FRAKTÁLNA KOMPRESIA OBRAZU

ĽUBOMÍR MAŤÚŠ

2006

FRAKTÁLNA KOMPRESIA OBRAZU

DIPLOMOVÁ PRÁCA

ĽUBOMÍR MAŤÚŠ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY
A INFORMATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

ODBOR: INFORMATIKA – POČÍTAČOVÁ GRAFIKA

VEDÚCI DIPLOMOVEJ PRÁCE
DOC. RNDr. ANDREJ FERKO, PhD.

BRATISLAVA 2006

Zadanie diplomovej práce

Naštudovať a implementovať metódu kompresie obrazu pomocou fraktálnej geometrie, spočítať súbor experimentov, vyhodnotiť prípadne publikovať.

Ďakujem svojmu školiteľovi, doc. RNDr. Andrejovi Ferkovi, PhD. za cenné rady
a pripomienky pri písaní tejto práce.

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry.

V Bratislave 10.5.2006 .....

Ľubomír Maťúš

Abstrakt

Názov práce: Fraktálna kompresia obrazu
Autor: Ľubomír Maťúš
Katedra: Katedra aplikovanej informatiky
Vedúci diplomovej práce: doc. RNDr. Andrej Ferko, PhD.

Abstrakt: Táto práca sa venuje metóde kompresie obrazu na báze fraktálnej geometrie. Pretože v tejto oblasti sa už dosiahli určité výsledky okolo šedo tónových (aj farebných) obrazov, zameriava sa hlavne na kompresiu farebných obrazov. Súčasťou práce je aj implementácia horizontálno-vertikálneho fraktálneho kodéra s vizualizáciou práce algoritmu, a dekodéra s možnosťou dekódovania do zvolenej mierky. Práca sa zaoberá rôznymi metódami predspracovania a úpravy farebného obrazu. Ďalej skúma vplyvy rôznych hodnôt parametrov fraktálneho kodéra na veľkosť a množstvo výstupných transformácií, a na kvalitu výstupného obrazu. Taktiež sa zameriava na optimalizáciu ukladania výstupných transformácií použitím rôznych kompresných metód. Nakoniec testuje výstupné dáta a obrazy vzhľadom na kvalitu a kompresný pomer, a výsledky porovnáva so známymi stratovými aj bezstratovými kompresnými metódami ako JPEG, JPEG2000, PNG, GIF, RAR, ZIP.

Kľúčové slová: fraktálna kompresia, fraktálna transformácia, IFS, HV schéma.

Abstract

Title: Fractal image compression
Author: Ľubomír Maťúš
Department: Department of applied informatics
Supervisor: doc. RNDr. Andrej Ferko, PhD.

Abstract: This work is dedicated to method of compression based on fractal geometry. Because there exist some results with grayscale images (also with color images) in this area, work is dedicated to color images compression. Part of this work is implementation of horizontal-vertical fractal encoder with visualization of algoritm progress, and implementation of decoder with possibility to decode to selected scale. Work deals with various methods of preprocessing and modification of color images. Investigates influence of various values of parameters of fractal encoder to size and quantity of output transformations, and to quality of output image. Also is dedicated to optimization of storing output transformations using various compression methods. Performs tests of quality and compression ratio on output images, results compares with popular lossy and loseless compression methods like JPEG, JPEG2000, PNG, GIF, RAR, ZIP.

Keywords: fractal compression, fractal transformation, IFS, HV scheme.

Obsah

Zoznam symbolov a skratiek

DCT	Discrete Cosine Transform, diskrétna kosínusová transformácia
DWT	Discrete Wavelet Transform, diskrétna waveletová transformácia
LZW	Lempel-Ziv-Welch, kompresný algoritmus
RLE	Run-Length Encoding, kódovanie pomocou dĺžky behov
JPEG	Joint Photographic Experts Group
MRCM	Multiple Reduction Copy Machine, kopírovací stroj
v_i	rovinná časť priestorovej transformácie
w_i	priestorová transformácia
$a_i,\ldots,f_i$	koeficienty rovinnej affinej transformácie
K:1	kompresný pomer
IFS	Iterated Function System, systém iterovaných funkcií
PIFS	Partitioned IFS, delené IFS
RIFS	Recurent IFS, rekurentné IFS
R	množina reálnych čísel, tiež RMS chyba
W	zobrazenie obsahujúce zjednotenie transformácií, zobrazenie realizujúce IFS
x_W	pevný bod alebo atraktor zobrazenie W
f,g	obrazová funkcia, funkcia $R^2 \to R$ alebo $I^2 \to I$
I	jednotkový interval, $I = \langle 0,1 \rangle$
$d, d_{\sup}, d_{rms}$	metrika: všeobecná, suprémová, RMS
RMS	Root Mean Square, stredná kvadratická odchýlka
PSNR	Peek Signal-to-Noise Ratio, pomer signálu a šumu
D_i	i-ta doména (Domain)
R_i	i-ty blok (Range)
HV	horizontálno-vertikálny
h_d	Hausdorffova metrika v priestore s metrikou d
$H(X)$	Hausdorffov priestor nad metrickým priestorom X
NTSC	National Television System(s) Committee, americká televízna norma
PAL	Phase Alternation Line, európska televízna norma
SECAM	Séquentiel Couleur Avec Mémoire (sequential color with memory), francúzska televízna norma
MPEG	Moving Picture Experts Group
DLL	Dynamic-Link Library, dynamicky linkovateľná knižnica
HDR	hlavičková štruktúra, tiež High Dynamic Range, vysoký dynamický rozsah
SI	Scale Indicies, indexy škálovacích faktorov
OI	Offset Indicies, indexy posunutí
TRN	Transformations, transformačné dáta
DP	Domain Positions, pozície domén
DF	Domain Factors, faktory domén
RP	Range Positions, pozície blokov
RS	Range Sizes, veľkosti blokov
RTD	Range Tree Distanties, deliace vzdialenosti v strome blokov
RTT	Range Tree Types, typy delení v strome blokov
AD	Average Difference, priemerná odchýlka
MD	Maximal Difference, maximálna odchýlka

HVE-Pi	HV encoder profil, profil nastavení triedy HV encoder
HVD-Pi	HV decoder profil, profil nastavení triedy HV decoder
MTF	Move-To-Front transformácia
BWT	Burrows-Wheeler Transform
CABAC	Context-Adaptive Binary Arithmetic Coding
CAVLC	Context-Adaptive Variable-Length Coding
dB	decibel

Zoznam obrázkov

Zoznam tabuliek

Zoznam grafov

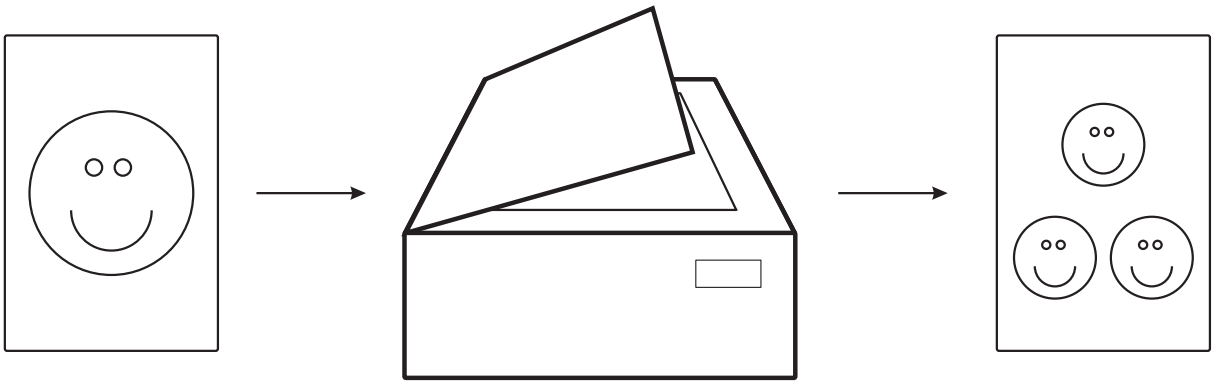
1. Úvod

1.1 Prečo kompresia obrazu

V súčasnosti sa do značnej miery rozvinulo používanie grafiky a grafických prvkov v počítačových systémoch, v porovnaní s nedávnou minulosťou, kedy sa používalo často len textové rozhranie. Obrazové informácie v rastrovej podobe zaberajú veľké objemy dát. Napr. obyčajná dovolenková fotka v rozlíšení 2048 x 1536 x 24bit zaberie 9 MB v hrubej nekomprimovanej podobe. Popritom sú obrazové informácie vo veľkej väčšine prípadov značne redundantné. Túto nadbytočnosť redukujeme kompresiou, ktorá oddelí podstatné informácie od nepodstatných a uloží len množstvo dát potrebné na rekonštrukciu. Kompresné metódy sa delia na bezstratové a stratové. Bezstratové kompresie dokážu z uložených dát rekonštruovať obraz na 100%, stratové rekonštruujú pôvodný obraz len do určitej miery podobnosti, ktorú si používateľ zvolí pred kompresiou. Súčasne používané metódy dokážu stlačiť obraz až na stotinu a viac jeho pôvodnej veľkosti (t.j. kompresný pomer 100:1). Najviac používané a najznámejšie sú napr. LZW (použitá v GIF formáte), PNG (bezstratový formát), JPEG (založený na DCT), JPEG2000 (založený na DWT), RLE (používaná v BMP formáte). V tejto práci by som sa chcel venovať menej známej metóde nazývanej „Fraktálna“. Nazýva sa takto kvôli použitým princípom a štruktúram. Táto kapitola sa z veľkej časti opiera o [2] a [1].

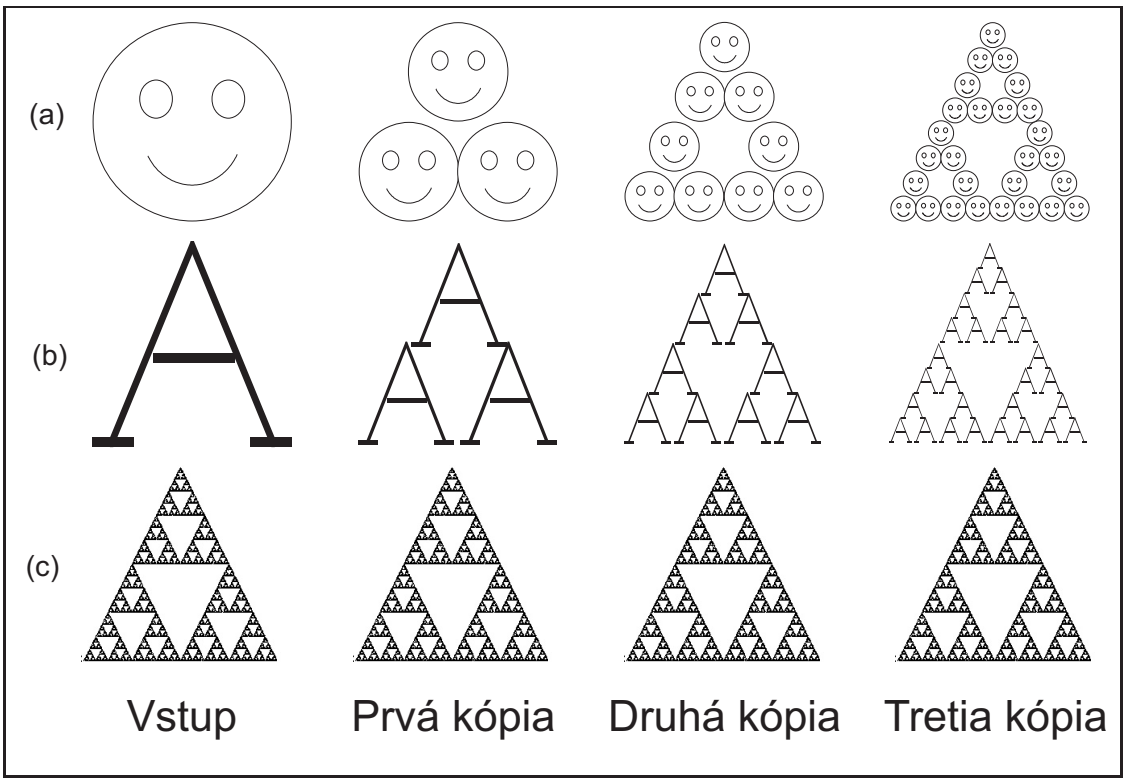
1.2 Čo je to fraktálna kompresia obrazu ?

Predstavme si špeciálny kopírovací stroj, ktorý zoberie vstup a vytlačí ho trikrát na jeden papier v polovičnej veľkosti na presne určené rôzne pozície.



Obrázok 1: Kopírovací stroj.

Čo sa stane keď výstup použijeme ako nový vstup a postup zopakujeme? Keď to budeme opakovať nekonečne veľakrát, postupnosť výstupných obrázkov sa začne stále viac podobať na jeden presne určený obrázok. Takisto pri použití iného vstupného obrázku a zopakovaní tohto procesu dostaneme znova ten istý obrázok ako v ostatných prípadoch. Tento obrázok sa nazýva *atraktor* a je pre daný kopírovací stroj špecifický.



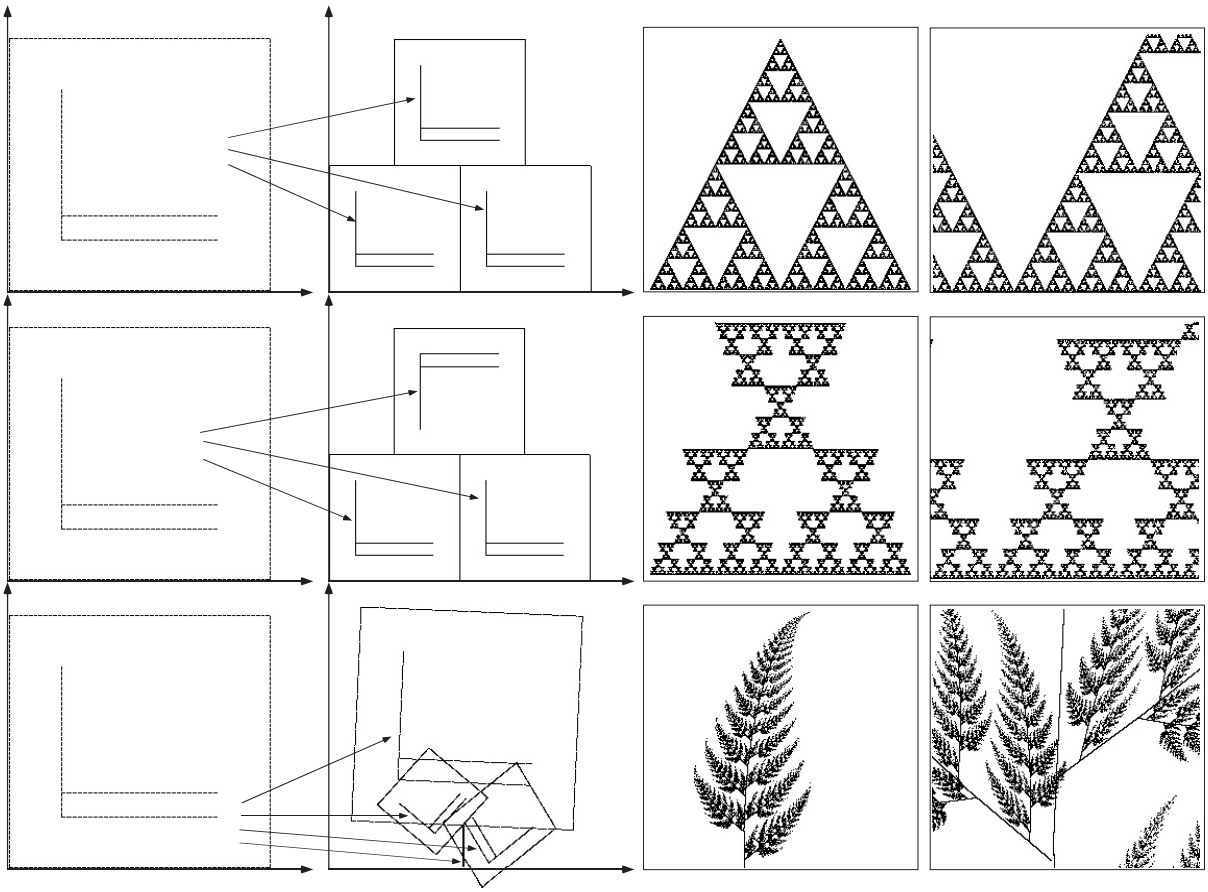
Obrázok 2: Prvé tri kópie pre rôzne vstupné obrazy.

Na obr. 2 vidíme, že obrázky (a) a (b) sa menia a blížia sa k jednému tvaru, pritom obrázok (c) sa nemení – je to atraktor tohto stroja. Obr.2 (c) má detail v ľubovoľnom priblížení a teda je to – fraktál. Vidíme tiež, že výsledný atraktor nie je ovplyvnený voľbou počiatočného vstupu.

Atraktor závisí od transformácií, ktoré realizuje kopírovací stroj. Aby postupnosť výstupných obrázkov konvergovala treba aby dané transformácie boli kontraktívne, t.j. aby vzdialenosť dvoch zobrazených bodov bola menšia ako v pôvodnom obraze, inak by atraktor mohol mať nekonečnú veľkosť. Transformácie môžu byť ľubovoľnej formy, ale v praxi sa obmedzíme na transformácie tvaru

$$w_i\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix},$$

nazývané *afinné* transformácie roviny. Touto formou sa dá vyjadriť napr. skosenie, zmena mierky, rotácia, posunutie a ich kombinácie v zmysle skladania. Afinné transformácie dokážu pokryť dosť veľkú množinu rôznych atraktorov. Na popis jednej transformácie si stačí zapamätať šesť reálnych koeficientov a_i, b_i, c_i, d_i, e_i, f_i. Týmto spôsobom reprezentácie obrazu si stačí uložiť niekoľko transformácií, ktoré generujú obraz namiesto ukladania všetkých obrazových bodov (pixelov). Pri rekonštrukcii sa vyberie iniciálny obraz a aplikujú sa transformácie, výstup použijeme ako vstup a opakujeme, kým nedosiahneme požadovanú presnosť.



Obrázok 3: Rôzne typy transformácií a ich atraktory.

1.2.1 Prečo „fraktálna“ kompresia ?

Dekódovaciu postupnosť môžme „naťahovať“ na dosiahnutie ľubovoľnej presnosti, teda môžme dosiahnuť detail v každom priblížení. Túto vlastnosť majú fraktály – preto ten názov. Takisto môžme obrázok dekódovať do ľubovoľnej veľkosti a rozmerov pri zachovaní detailov. Problém nastáva pri kódovaní z rastrovej predlohy, pri ktorej sa ťažko generujú detaily, ktoré v predlohe neboli.

1.2.2 Prečo fraktálna „kompresia“ ?

Metódy kompresie obrazu sa posudzujú podľa ich schopnosti stlačiť veľkosť obrazu, pri určitej miere podobnosti s predlohou. Pomer objemu nestlačených dát v rastrovej podobe ku objemu dát v stlačenej podobe sa nazýva *kompresný pomer* (*K*) a zapisuje sa pomerom *K:1*. Pri fraktálnej metóde je tento pomer diskutabilný, pretože môžme obraz dekódovať do ľubovoľnej veľkosti a tým sa mení objem nestlačených dát (napr. pri zväčšení 4x sa objem zväčší 16x). Takže pri pojme *kompresný pomer* budeme používať pôvodný rozmer predlohy.

1.2.3 Systém iterovaných funkcií (Iterated Function System - IFS)

Teraz si ukážeme formálnu notáciu pre náš kopírovací stroj. Vo fraktálnej geometrii máme štruktúru popisujúcu takýto spôsob práce - IFS. IFS pozostáva z množiny kontraktívnych transformácií mapujúcich R^2 na seba

$$A = \{w_i : R^2 \to R^2 \mid i = 1,...,n\}.$$

Obraz je definovaný ako množina bodov v R^2 (pixely sa kreslia čiernou farbou). Množina transformácií *A* definuje zobrazenie (množiny bodov na inú množinu bodov)

$$W(\bullet) = \bigcup_{i=1}^{n} w_i(\bullet).$$

Toto zobrazenie neaplikujeme na celú rovinu, ale len na množinu bodov *S* a výsledkom je nová množina bodov *S'*, ktorá sa použije ako vstup ďalšej iterácie.
Pre takéto zobrazenie platia dva dôležité fakty:
1. Ak sú transformácie w_i kontraktívne v rovine, tak je aj *W* kontraktívne.
2. Máme dané kontraktívne zobrazenie *W*. Potom existuje špeciálna množina (obraz) x_W nazývaný atraktor s nasledujúcimi vlastnosťami:

a) ak aplikujeme *W* na atraktor x_W, dostaneme nezmenený atraktor, x_W je pevným bodom zobrazenia *W*,

$$W(x_W) = x_W .$$

b) máme daný vstupný obraz S_0, aplikujeme *W* na S_0, dostaneme $S_1=W(S_0)$, postupným aplikovaním zobrazenia *W* dostávame postupnosť $S_i = W(S_{i-1}) = W^i(S_0)$.
Pevný bod

$$x_W = S_\infty = \lim_{n\to\infty} W^n(S_0)$$

nezávisí od voľby S_0.

c) pevný bod x_W je jediným pevným bodom zobrazenia *W*.
(tieto tri podmienky a na nich založené tvrdenie sú známe ako „veta o pevnom bode kontraktíveho zobrazenia“ (Contractive Mapping Fixed-Point Theorem)).

Takto definované IFS pracuje s čierno-bielymi obrazmi. Neskôr rozšírime tento model na prácu so šedo tónovými obrazmi.

1.3 Samopodobnosť v obrazoch

Ďalej budeme pod pojmom obraz (alebo obrázok) rozumieť šedo tónový obraz.

1.3.1 Obraz ako graf funkcie

Na prácu s obrazom budeme potrebovať jeho matematický model. Budeme pracovať na intervale $I = \langle 0,1 \rangle$, obraz bude spojitá funkcia $f : I^2 \to I$. Hodnota *f(x,y)* znamená jasovú úroveň na pozícii *(x,y)* v I^2. Takto navrhnutá reprezentácia sa dá chápať aj ako podmnožina priestoru I^3 , $S = \{(x, y, f(x, y)) \mid x, y \in I\}$. Diskrétna podoba obrazovej funkcie *f* je matica $F=[f_{ij}]$ typu *MxN*, kde $f_{ij} = f\left(\frac{i}{M}, \frac{j}{N}\right)$.

1.3.2 Obrazová metrika

V ďalšej práci budeme merať, ako sú si dva obrazy podobné (vzdialené). Na túto úlohu je určená metrika. Poznáme veľa rôznych metrík, ale v praxi sa používa len málo vybraných. Napr. suprémová metrika pre dva obrazy *f* a *g*

$$d_{\sup}(f, g) = \sup_{(x,y)\in I^2} |f(x, y) - g(x, y)|$$

určuje vzdialenosť podľa maximálnej odchýlky obrazových funkcií, alebo v praxi často používaná RMS metrika (Root Mean Square)

$$d_{rms}(f,g)=\sqrt{\int_{I^2}(f(x,y)-g(x,y))dxdy}\,,$$

ktorá určuje podobnosť podľa strednej kvadratickej odchýlky.

1.3.3 Modifikácia kopírovacieho stroja (Partitioned IFS - PIFS)

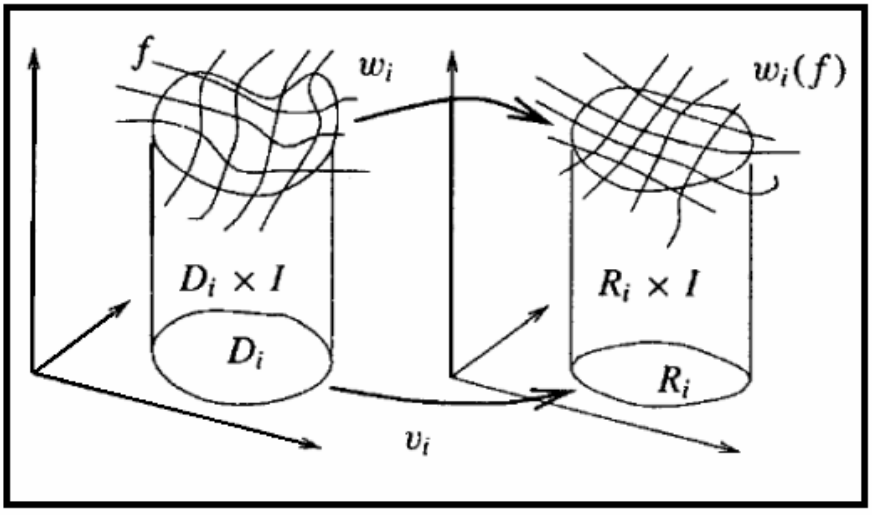
Obrazy reálneho sveta nie sú celkom samopodobné, nie sú kópiami ani transformáciami samých seba. Napriek tomu obsahujú vnútornú samopodobnosť v malých častiach (napr. okraje oblakov majú podobné kontúry, listy rastlín majú podobné vzory, ...atď.). Pôvodný model kopírovacieho stroja upravíme tak, aby pokrýval naše požiadavky. Upravíme ho tak, že bude vedieť kopírovať a transformovať oblasť vstupného obrazu na inú oblasť výstupného obrazu, a tiež mu pridáme možnosť upravovať kontrast a jas transformovanej oblasti. Oblasť vstupného obrazu budeme nazývať *doména* (domain) a označovať D_i , a oblasť výstupného obrazu budeme nazývať *blok* (range) a označovať R_i. Potom transformácia w_i je daná doménou D_i , nad ktorou pracuje. Zobrazenie $W(f)=w_1(f)\cup\ldots\cup w_n(f)$ je rovnaké ako pre IFS. Množina takýchto transformácií sa nazýva Partitioned IFS (PIFS). Typ transformácií nie je obmedzený, ale v praxi budeme používať len afinné transformácie tvaru

$$w_i\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}a_i & b_i & 0\\ c_i & d_i & 0\\ 0 & 0 & s_i\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}+\begin{bmatrix}e_i\\f_i\\o_i\end{bmatrix},$$

kde koeficienty a_i, b_i, c_i, d_i, e_i, f_i majú rovnaký význam ako pri IFS, koeficient s_i kontroluje kontrast a o_i jas výsledného bodu. Priestorová časť transformácie má tvar

$$v_i\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}a_i & b_i\\ c_i & d_i\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}e_i\\f_i\end{bmatrix}.$$

Transformácia $w_i(f)$ sa realizuje ako $w_i(f)=w_i(x,y,f(x,y))$, a pracuje len v priestore D_ixI (t.j. vertikálny priestor nad D_i). V rovine to znamená, že $v_i(D_i) = R_i$.



Obrázok 4: Transformácia w_i mapuje graf nad doménou D_i na graf nad blokom R_i.

Chceme, aby $W(f)$ bol obraz, a preto musí platiť $\bigcup R_i = I^2$ (bloky pokrývajú celý obraz) a $R_i \cap R_j = \varnothing, i \neq j$ (bloky sa neprekrývajú). Postupnosť $f_i = W^i(f_0) = W(f_{i-1})$ je postupnosťou obrazov generovaných modifikovaným strojom (PIFS).

1.3.4 Pevný bod PIFS

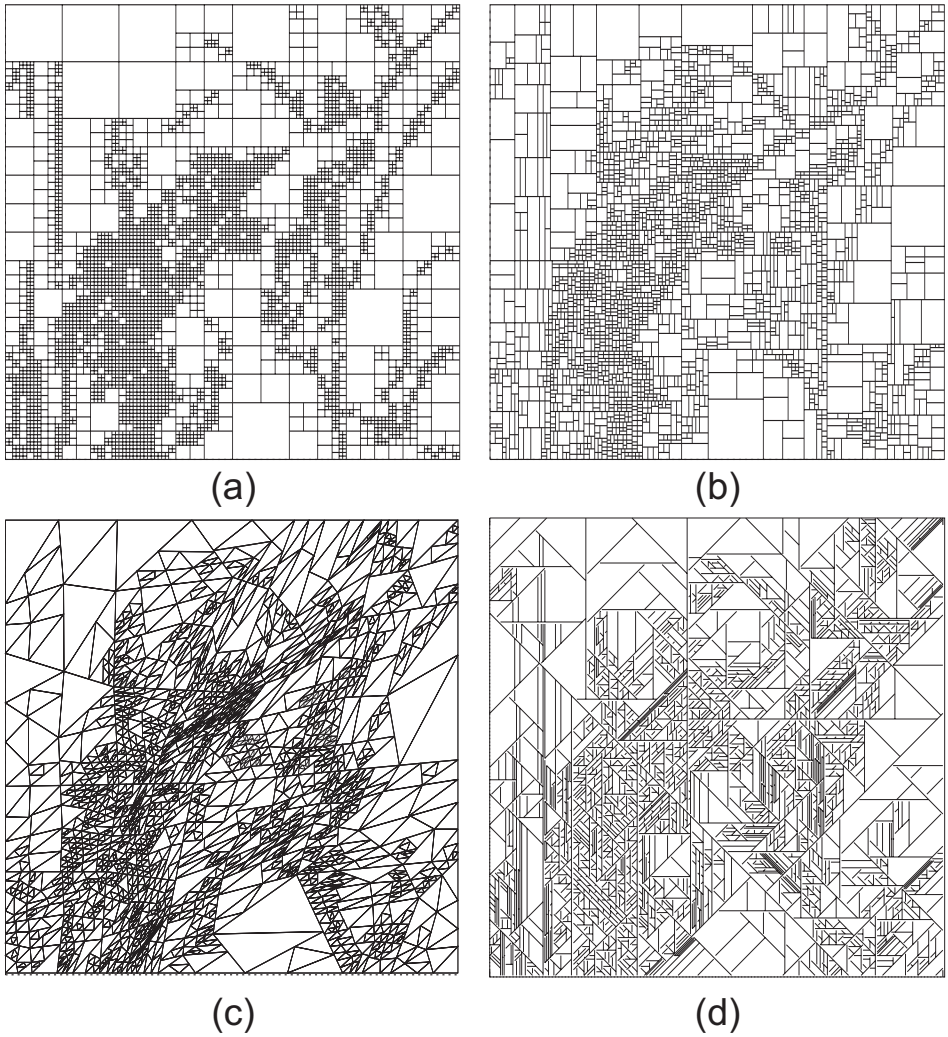
Pevný bod je obraz f spĺňajúci $W(f) = f$. Ak je W kontraktívne, tak podľa vety o pevnom bode kontraktívneho zobrazenia sa dokážeme ľubovoľne blízko k nemu priblížiť postupnosťou $f_i=W(f_{i-1})$. Ak si zvolíme suprémovú metriku, ktorá je citlivá len v smere z, nie je potrebné sa zaoberať kontraktivitou v smeroch x a y. Zobrazenie W bude určite kontraktívne, ak všetky $s_i<1$ (t.j. keď vzdialenosť bude škálovaná faktorom $0<= s < 1$). V skutočnosti môžme vetu o pevnom bode aplikovať aj na zobrazenie W^m pre nejaké m. V tomto prípade môže byť W^m kontraktívne, aj keď koeficienty s_i vo W nebudú menšie ako 1.

1.4 Kódovanie obrazu

Máme daný obraz f, ktorý chceme zakódovať ako zobrazenie $W = \bigcup_{i=1}^{N} w_i$ také, že $f = W(f)$. Potrebujeme nájsť rozdelenie obrazu na bloky R_i , transformácie w_i (aj ich domény D_i), pokrývajúce všetky bloky. V praxi je ťažké nájsť také W, že f bude jeho pevným bodom, preto sa snažíme nájsť W' pre f', ktoré je blízke f vzhľadom na danú metriku napr. d_{rms} (t.j. chceme aby $d_{rms}(f,f')$ bolo čo najmenšie).

1.5 Spôsoby delenia obrazu

Poznáme viacero rôznych spôsobov delenia obrazu: pevné, kvadrantové stromy, horizontálno-vertikálne (HV), trojuholníkové, šesťuholníkové, polygonálne, regiónové, ...atď., ale ukázalo sa, že je len málo z nich vhodných pre prax. Dôležité pre schému delenia je, aby výsledné tvary blokov dokázali pokryť celý obraz bez toho aby sa navzájom prekrývali, pretože v oblastiach prekryvu by nebolo možné obraz jednoznačne dekódovať alebo by bolo treba definovať stmeľovaciu funkciu, ktorá by spojila prekrývajúce sa oblasti. So zložitosťou možných tvarov blokov narastá množstvo informácií potrebných na ich uloženie. Napr. na uloženie kvadrantového stromu postačuje uložiť si informáciu z vrcholu o tom či sa blok delí alebo nie, ale na uloženie stromu obdĺžnikových blokov potrebujeme uložiť aj informáciu o type delenia (horizontálne alebo vertikálne) a o vzdialenosti, v ktorej je blok rozdelený. Takisto rôzne tvary blokov poskytujú rôzne množiny transformácií, ktoré sa dajú vhodne využiť a nie sú náročné na kódovanie. Napr. pre trojuholníkové bloky môžme uvažovať viac uhlov otočenia ako pre štvorcové alebo obdĺžnikové bloky, ale za cenu potreby väčšieho množstva dát na ich uchovanie.



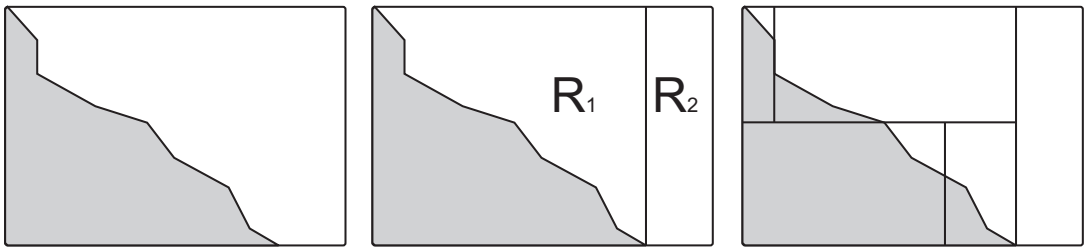
Obrázok 5: Rôzne spôsoby delenia obrazu (a) kvadrantový strom, (b) HV delenie, (c) trojuholníkové, (d) polygonálne.

1.5.1 Kvadrantový strom

Tento spôsob delenia rieši nedostatky pevného rozdelenia (pevne určené bloky). Obraz f rozdelíme na štyri základné kvadranty (štvorce), ktoré budú predstavovať prvé bloky. Pri nedostatočnom pokrytí bloku R_i (t.j. ak nenájdeme doménu D_i a transformáciu w_i takú, že $w_i(D_i)$ je dosť blízko fxR_i) ho rozdelíme na štyri menšie štvorcové bloky a pracujeme s nimi osobitne. Bloky delíme kým nie sú dostatočne pokryté alebo nemajú najmenší povolený rozmer (napr. 4x4 pixely).

1.5.2 Horizontálno-Vertikálne (HV) delenie

Toto delenie odstraňuje niektoré nedostatky kvadrantových stromov. Takisto sa bloky delia na menšie v prípade nedostatočného pokrytia, ale delia sa len na dve časti, buď v horizontálnom alebo vertikálnom smere a tiež sa prihliada aj na obsah daného bloku.



Obrázok 6: Ukážka horizontálno-vertikálneho delenia blokov.

1.5.3 Iné spôsoby delenia

K ďalším metódam delenia patrí napr. trojuholníkové delenie (obr. 5 (c)). Bloky sú trojuholníkového tvaru. Trojuholníkový blok sa rozdelí na štyri menšie trojuholníky. Trojuholníky môžu byť otočené aj v inom ako 90° uhle. Táto schéma je trochu flexibilnejšia ako dve predošlé. Taktiež môžme rovinu deliť na šesťuholníky alebo použiť nepravidelné štruktúry (napr. osemuholníky dopĺňané štvorcami).

1.6 Implementácia

Ukážeme si dva postupy kódovania obrazu prihliadajúcich na rôzne požiadavky, konkrétne to budú kvalita alebo kompresný pomer. Pred kódovaním potrebujeme zvoliť schému delenia blokov R_i na menšie časti, a množinu domén *D* (domain pool), z ktorej budeme vyberať domény na pokrývanie. Množina *D* môže byť množina všetkých rôzne veľkých obdĺžnikov (štvorcov alebo trojuholníkov) v obraze, v tomto prípade ide o kompletné prehľadávanie, ktoré je ale časovo náročné. Alebo do *D* vyberieme len úzku podmnožinu zo všetkých možných častí obrazu, tým sa značne urýchli prehľadávanie, ale môže sa znížiť kvalita pokrytia.

1.6.1 Kódovanie s maximálnou povolenou chybou

1. zvolíme toleranciu *e*

2. do zoznamu *R* pridáme $R_1 = I^2$ a označíme ho ako nepokrytý (zoznam *R* je zoznam blokov)

3. kým sú v *R* nepokryté bloky, tak robíme {

    4. z množiny domén *D* nájdeme doménu D_i a korešpondujúcu transformáciu w_i , ktorá najlepšie pokrýva blok R_i (t.j. minimalizuje chybu d_{rms})

    5. ak (je chyba pokrytia $d_{rms} < e$) alebo (veľkosť $R_i \le r_{\min}$) tak
        označíme R_i ako pokrytý, zapíšeme transformáciu w_i
    inak
        rozdelíme R_i na menšie časti, označíme ich ako nepokryté, a pridáme do *R*, odstránime R_i z *R*

}

Parameter r_{min} znamená minimálnu dovolenú veľkosť bloku. Parameter *e* ovplyvňuje chybu pokrytia. Väčšie hodnoty *e* znižujú vernosť obrazu, ale kódovanie vytvára menší súbor blokov a transformácií. Na druhej strane menšie hodnoty *e* zvyšujú vernosť ale produkujú väčší súbor informácií.

1.6.2 Kódovanie s maximálnym povoleným počtom transformácií

1. zvolíme maximálny povolený počet transformácií N

2. do zoznamu R pridáme $R_1 = I^2$ a označíme ho ako nepokrytý

3. kým sú v R nepokryté bloky, tak robíme {

    4. pre každý nepokrytý blok R_i v R nájdeme a uložíme doménu D_i a transformáciu w_i , ktorá najlepšie pokrýva R_i a označíme R_i ako pokrytý

    5. zo zoznamu R vyberieme blok R_j taký, že veľkosť $R_j > r_{\min}$ a chyba pokrytia d_{rms} je najväčšia (najhoršie pokrytý)

    6. ak počet blok v R je < N, tak {

        7. rozdelíme R_j na menšie časti, pridáme ich do R, označíme ich ako nepokryté

        8. vypustíme R_j ,w_j, D_j zo zoznamov

    }

}

9. zapíšme všetky transformácie w_i v zozname.

Parameter N ovplyvňuje veľkosť výstupnej informácie. Celkový počet bitov závisí od použitej schémy delenia, počtu bitov použitých na uloženie domény a transformácie. Čím väčšie zvolíme N, tým presnejšie bude obraz pokrytý (vo väčšine prípadov).

1.6.3 Dekódovanie

Dekódovanie je jednoduché. Zoberieme blok R_i, jeho transformáciu w_i, a doménu D_i. Doménu D_i transformujeme podľa w_i , vynásobíme s_i , pripočítame o_i , upravíme veľkosť na veľkosť R_i a výsledné pixle vložíme na pozíciu R_i. Toto spravíme pre všetky bloky R_i. Väčšinou stačí okolo 10 iterácií na dosiahnutie dobrej aproximácie pevného bodu.

1.6.4 Metrika RMS

V praxi porovnávame domény s blokmi pomocou RMS metriky, pretože poskytuje dobrú mieru odhadu vzdialenosti dvoch obrazov a optimálne koeficienty s_i a o_i sa vypočítajú ľahko. Máme dva štvorce (alebo kompatibilné obrazy) D_i s pixelmi a_1, ... , a_n (doména je zmenšená na rozmer bloku) a R_i s pixelmi b_1, ..., b_n. Hľadáme hodnoty s a o minimalizujúce hodnotu R

$$R = \sum_{i=1}^{n} \left(s \cdot a_i + o - b_i \right)^2 .$$

Položíme parciálne derivácie podľa s a o rovné nule. Hodnota R dosahuje minimum, keď

$$s = \frac{\left[n\sum_{i=1}^{n} a_i b_i - \sum_{i=1}^{n} a_i \sum_{i=1}^{n} b_i \right]}{\left[n\sum_{i=1}^{n} a_i^2 - \left(\sum_{i=1}^{n} a_i \right)^2 \right]} \ , \ o = \frac{1}{n}\left[\sum_{i=1}^{n} b_i - s\sum_{i=1}^{n} a_i \right].$$

Ak menovateľ vo vzťahu pre s je rovný nule, tak položíme $s = 0$, $o = \frac{1}{n}\sum_{i=1}^{n} b_i$. Potom RMS chyba je $d_{rms} = \sqrt{R}$.

Ukážeme efektívny výpočet hodnoty d_{rms}:

Nech D_i je doména transformácie w_i . Doménu D_i zmenšíme na veľkosť bloku R_i (napr. priemerovaním, alebo metódou najbližšieho suseda) a dostaneme novú maticu pixelov F_i.

1. ak w_i obsahuje rotáciu alebo preklopenie pixelov, tak podľa toho upravíme pixle F_i (permutujeme, alebo transformujeme).

2. vypočítame sumu $\sum_{a \in F_i} a$, a sumu $\sum_{a \in F_i} a^2$.

3. vypočítame sumu $\sum_{b \in R_i} b$, a sumu $\sum_{b \in R_i} b^2$.

4. vypočítame sumu $\sum_{a \in F_i, b \in R_i} ab$, kde násobíme len hodnoty obrazových bodov na rovnakej pozícii.

5. tieto sumy použijeme na výpočet hodnôt s_i, o_i a R.

6. vypočítame hodnotu $d_{rms} = \sqrt{R}$.

Prvé štyri sumy sa dajú vypočítať v čase predspracovania a netreba ich počítať pri každom novom porovnávaní. Jedine posledná suma závisí aj od domény aj od bloku.

1.6.5 Efektívne ukladanie transformácií

Na presné určenie transformácie w_i sú potrebné jej koeficienty, doména a blok. Koeficienty s_i a o_i si musíme uložiť, ostatné koeficienty (okrem informácií o transformácii ako rotácia alebo preklopenia) možno určiť z pozície domény a bloku. Ak uvažujeme delenie kvadrantovým stromom, do úvahy pripadá osem možností ako otočiť alebo preklopiť štvorcovú doménu a namapovať ju na blok (štyri možnosti otočenia o uhly 0°, 90°, 180°, 270°, alebo preklopenie domény a znova tie isté možnosti otočenia). Na zakódovanie tohto potrebujeme 3 bity. Na koeficienty s_i a o_i použijeme kvantovanie a dve rôzne inštancie entropických kodérov (napr. huffmanov alebo aritmetický kodér), pretože nemusia mať rovnakú distribúciu hodnôt. V praxi sa používa kvantovanie koeficientov s_i na 5 bitov a o_i na 7 bitov. V takomto prípade je dobré počítať RMS chybu pri pokrývaní bloku z kvantovaných hodnôt s a o. Dosiahne sa tak väčšia vernosť ako v prípade kvantovania až po pokrytí bloku. Pozíciu a veľkosť bloku uložíme ako cestu z koreňa k nemu v kvadrantovom strome. Cesta v strome presne určuje aj pozíciu (podľa navigácie) aj veľkosť (podľa hĺbky). Na jeden vrchol v strome sú potrebné 2 bity. Pozíciu a veľkosť domény uložíme diferenčným spôsobom, rozdielom pozície aj veľkosti od predchádzajúcej domény (ak predpokladáme, že domény máme usporiadané). Celkovo to vychádza na jednu transformáciu cca 31 až 34 bitov.

1.6.6 Časová optimalizácia kódovania

Kódovanie je možné urýchliť určitou klasifikáciou domén a blokov napr. podľa pozície svetlého kvadrantu (blok alebo doménu rozdelíme na kvadranty a určíme priemernú svetlosť v každom z nich, počet možných kombinácií priestorového usporiadania kvadrantov je konečný a zdá sa byť vhodným atribútom na jemnejšiu klasifikáciu domén a blokov). Týmto spôsobom sa dá kódovanie podstatne urýchliť tým, že z porovnávania vylúčime celé triedy nepodstatných domén.

2. Matematické základy

V tejto časti formálne zadefinujeme použité matematické štruktúry a uvedieme tvrdenia, ktoré sú dokázané o týchto štruktúrach. Dôkazy týchto tvrdení nebudeme uvádzať. Táto kapitola sa opiera hlavne o [2] a [5].

2.1 IFS

2.1.1 Úplný metrický priestor

Definícia: Metrický priestor je množina *X*, na ktorej má reálna funkcia $d : X \times X \to R$ nasledujúce vlastnosti:

1. $d(a,b) \geq 0, pre \forall a,b \in X$
2. $d(a,b) = 0 \Leftrightarrow a = b, pre \forall a,b \in X$
3. $d(a,b) = d(b,a), pre \forall a,b \in X$
4. $d(a,c) \leq d(a,b) + d(b,c), pre \forall a,b,c \in X$

Takáto funkcia *d* sa nazýva *metrika*.

2.1.2 Hausdorffova metrika

Uvedieme najprv niekoľko príkladov metrík.

1. Euklidovská metrika v rovine

$$d((x_1,y_1),(x_2,y_2)) = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}\ .$$

2. Maximová metrika v rovine

$$d((x_1,y_1),(x_2,y_2)) = \max\{|x_1-x_2|,|y_1-y_2|\}.$$

3. Manhattanská metrika v rovine

$$d((x_1,y_1),(x_2,y_2)) = |x_1-x_2| + |y_1-y_2|\ .$$

4. Suprémová metrika pre (merateľné) funkcie $f,g : R^2 \to R$

$$d_{\sup}(f,g) = \sup_{x \in R^2} |f(x) - g(x)|\ .$$

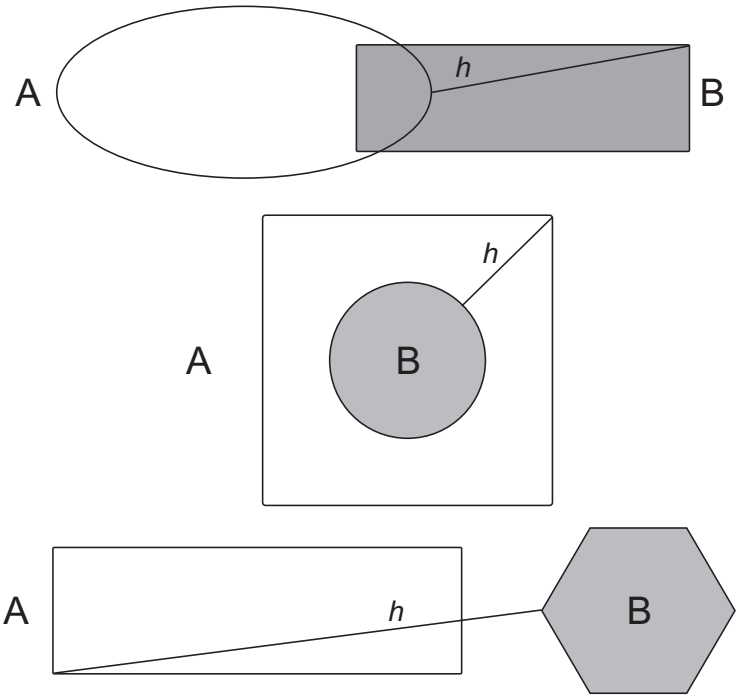
5. Ak *x* je bod v rovine a *A* je podmnožina roviny, definujeme

$$\delta_d(x,A) = \inf_{y \in A} d(x,y),$$

kde *d(x,y)* je rovinná metrika. Ak *B* je tiež podmnožina roviny, tak definujeme

$$h_d(A,B) = \max\left\{\sup_{x \in A} \delta_d(x,B), \sup_{y \in B} \delta_d(y,A)\right\}.$$

Toto je *Hausdorffova metrika v rovine*. Určuje vzdialenosť dvoch množín bodov v rovine.



Obrázok 7: Hausdorffova vzdialenosť pre rôzne vzájomné polohy dvoch množín.

Definícia: Postupnosť bodov $\{x_n\}$ v metrickom priestore sa nazýva *Cauchyovská*, ak platí

$$\forall \varepsilon > 0, \exists N \in N : \forall m, n > N : d(x_m, x_n) < \varepsilon$$

(pre každé *epsilon* existuje *N* také, že rozdiel ľubovoľných členov postupnosti za indexom *N* je menší ako *epsilon*).

Definícia: Metrický priestor *X* sa nazýva *kompletný* (*úplný*), ak každá *Cauchyovská* postupnosť má limitu v *X*.

Definícia: Metrický priestor *X* sa nazýva *kompaktný*, keď je *uzavretý* a *ohraničený*.

Teraz zadefinujeme priestor, s ktorým budeme pracovať.
Definícia: *Haussdorffov priestor* nad metrickým priestorom *X* je množina

$$H(X) = \{ S \subset X \mid S \text{ je kompaktná} \}$$

všetkých kompaktných podmnožín metrického priestoru *X*.

Ak $A \in H(X)$, potom definujeme $A_d(\varepsilon) = \{x \mid d(x, y) \leq \varepsilon, y \in A\}$ ako epsilonové okolie množiny *A*.

Hausdorffova vzdialenosť dvoch množín $A, B \in H(X)$ je definovaná ako

$$h_d(A, B) = \max\{\inf\{\varepsilon \mid B \subset A_d(\varepsilon)\}, \inf\{\varepsilon \mid A \subset B_d(\varepsilon)\}\}.$$

(pre body oboch množín zistíme minimálnu vzdialenosť do druhej množiny, a z týchto vzdialeností vyberieme maximálnu).

Veta: Nech *X* je kompletný metrický priestor s metrikou *d*, potom *H(X)* s Hausdorffovou metrikou h_d je kompletný metrický priestor.

2.1.3 Kontraktívne mapy (zobrazenia) a IFS

Definícia: Nech *X* je metrický priestor s metrikou *d*. Zobrazenie (*mapa*) $w : X \to X$ sa nazýva *Lipschitzovské* s *Lipschitzovým faktorom s*, ak existuje $s > 0$ také, že

$$\text{pre } \forall x, y \in X: \; d(w(x), w(y)) \leq sd(x, y).$$

Ak je $s < 1$, hovoríme, že w je *kontraktívne* s *kontraktivitou s*.

Lema: Ak $f : X \to X$ je Lipschitzovská, potom f je spojitá.

Definícia: Nech $w_i : R^2 \to R^2$, $i = 1,...,n$ sú kontraktívne transformácie, potom definujeme zobrazenie $W : H(R^2) \to H(R^2)$ ako

$$W(S) = \bigcup_{i=1}^{n} w_i(S).$$

Veta: Ak $w_i : R^2 \to R^2$, $i = 1,...,n$ sú kontraktívne transformácie s faktormi s_i, potom je aj zobrazenie $W : H(R^2) \to H(R^2)$, $W(S) = \bigcup_{i=1}^{n} w_i(S)$, kontraktívne vzhľadom na Hausdorffovu metriku s faktorom $s = \max_{i=1,...,n} \{s_i\}$.

2.1.4 Veta o pevnom bode kontraktívneho zobrazenia

Veta: (veta o pevnom bode kontraktívneho zobrazenia, Contractive Mapping Fixed-Point Theorem) Nech X je kompletný metrický priestor a $f : X \to X$ je kontraktívne zobrazenie. Potom existuje práve jeden bod $x_f \in X$ taký, že pre $\forall x \in X$ platí

$$x_f = f(x_f) = \lim_{n \to \infty} f^n(x).$$

Tento bod sa nazýva *pevný bod* alebo *atraktor* zobrazenia *f*.

Dôsledok: (Collage Theorem) $d(x, x_f) \leq \frac{1}{1-s} d(x, f(x))$.

Dôsledok hovorí o vzťahu vzdialenosti bodu x a atraktora x_f ku vzdialenosti dvoch po sebe nasledujúcich hodnôt v postupnosti aproximácií x a $f(x)$.

Definícia: Nech *f* je Lipschitzovské zobrazenie. Ak existuje *n* také, že f^n je kontraktívne, tak *f* nazýva *eventuálne kontraktívne*. Číslo *n* nazývame *exponent eventuálnej kontraktivity*.

Dôsledok: Nech *f* je eventuálne kontraktívne s exponentom *n*. Potom pre vzdialenosť susedných aproximácií platí

$$d(x, x_f) \leq \frac{1}{1-s} \frac{1-\sigma^n}{1-\sigma} d(x, f(x)),$$

kde *s* je kontraktívny faktor zobrazenia f^n a σ je Lipschitzov faktor zobrazenia *f*.

Definícia: (Systém iterovaných funkcií – Iterated Function System - IFS)
Nech X je kompletný metrický priestor. IFS je množina kontraktívnych transformácií $w_i : X \to X, i = 1,..., n$.

Poznáme viaceré alternatívy IFS, napr. rekurentné IFS (RIFS), ktoré sa ale dajú modelovať na pôvodných IFS, preto sa nimi nebudeme ďalej zaoberať.

2.2 Matematické modely obrazu

Budeme sa zaoberať matematickým popisom monochromatických obrazov (gray-scale, nie black-and-white), farebné obrazy sú zložené z viacero monochromatických rovín. Obraz môžme popísať ako priestor s mierou, maticu intenzít (pixely) alebo funkciu.

2.2.1 Priestor s mierou

Obraz reprezentujeme ako mieru μ v rovine, kde intenzita na podmnožine A roviny je

$$\mu(A) = \int_A d\mu .$$

Intenzita bodu v tomto modeli je nulová, miera dáva zmysel len pre plôšky. Tento model používať nebudeme.

2.2.2 Maticová forma

Obraz sa v tomto modeli reprezentuje ako vektor diskrétnych hodnôt intenzít v bodoch. Vektor má rozmer $n = M.N$, kde M a N sú rozmery obrazu, je to vektor z priestoru R^n.
Bežne používané normy v tomto priestore sú tzv. *p-normy* tvaru

$$\|x\|_p = \sqrt[p]{\left(|x_1|^p + \ldots + |x_n|^p\right)},$$

ktoré definujú metriku

$$d_p(x,y) = \|x - y\|_p .$$

Špeciálna norma ∞ – *norma* je definovaná ako

$$\|x\|_\infty = \max_{i=1,\ldots,n} |x_i| .$$

Budeme používať 2-normu nazývanú tiež l^2 norma, ktorá definuje metriku nazývanú RMS, t.j. ak $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ a $\boldsymbol{y} = \langle y_1, \ldots, y_n \rangle$ sú obrazy potom ich RMS vzdialenosť je

$$d_{rms}(x,y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} .$$

V praxi sa na meranie podobnosti dvoch obrazov používa aj PSNR (Peak Signal-to-Noise Ratio, pomer signál / šum). Ide o pomer signálu a pridaného šumu.

$$PSNR = 20 \log_{10}\left(\frac{b}{rms}\right),$$

kde b je maximálna možná hodnota signálu (pre bitmapy je to napr. 255) a *rms* je RMS vzdialenosť predlohy a obrazu dekódovaného z kódovaných dát predlohy. Čím je toto číslo väčšie, tým je váha šumu menšia a naopak, čím je menšie tým je miera pridaného šumu vyššia.

2.2.3 Funkcie nad $\mathbf{R^2}$

Funkcia $f : R^2 \to R$ môže byť považovaná za obraz s nekonečným rozlíšením. Pretože reálne pracujeme s konečnými obrazmi, obmedzíme sa na interval $I=\langle 0,1 \rangle$. Budeme pracovať len s funkciami $f : I^2 \to I$. Hodnota *f(x,y)* je jasovou hodnotou v bode *(x,y)*. Metriky pre tieto funkcie sú podobné ako metriky pre maticové vyjadrenie obrazu. L^p metrika má tvar

$$d_p(f,g) = \|f - g\|_p = \sqrt[p]{\int_{I^2} |f - g|^p} ,$$

RMS metrika je potom L^2 metrika.
Používajú sa ešte dve ďalšie metriky a to L^∞ (maximová) a suprémová metrika.

$$d_{\sup}(f,g) = \sup_{(x,y)\in I^2} |f(x,y) - g(x,y)|.$$

Vzdialenosť dvoch obrazov f, g je maximálny rozdiel medzi jasovými hodnotami meranými v tom istom bode obrazov.

Nakoniec môžme definovať obraz aj ako podmnožinu priestoru R^3,

$$A = \{(x, y, f(x,y)) \mid x, y \in I\}$$ ako graf funkcie f.

2.2.4 Farebné obrazy

Ľudia vnímajú obraz len v troch farebných hladinách s rôznymi vlnovými dĺžkami, preto stačí farebný obraz popisovať napr. len troma funkciami pre R, G a B hladiny [4]. V technickej praxi sa používajú aj iné farebné modely, ktoré oddeľujú jasovú informáciu od informácií o farbe. Takýmito modelmi sú napr. modely YIQ (použitý v americkej televíznej norme NTSC), YUV (použitý v európskych televíznych normách PAL a SECAM), YCbCr (použitý v digitálnych systémoch JPEG a MPEG). Prevod z RGB do týchto modelov a naspäť je jednoduchá lineárna kombinácia vyjadrená maticou a inverznou maticou

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix},$$

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.141 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix},$$

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & 0.418668 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

Tieto modely sú vhodnejšie pre kódovanie farebných obrázkov. Môžme si dovoliť použiť nižšiu kvalitu pre farebné hladiny ako pre jasovú hladinu bez straty vizuálnej kvality. Podobným spôsobom sa farebná informácia spracúva aj v ľudskom vizuálnom systéme.

2.3 Afinné transformácie

Afinná transformácia $w: R^n \to R^n$ sa dá zapísať v maticovom tvare ako $w(x) = Ax + b$, kde A je matica typu n x n a b je n-rozmerný vektor posunutia. Transformácia bude kontraktívna práve vtedy, keď bude kontraktívna jej lineárna časť, t.j. ak

$$\|A\| = \sup_{x\in R^n} \|Ax\| / \|x\| < 1,$$

kde $\|A\|$ je norma matice A.

2.4 Partitioned IFS

Definíciu IFS rozšírime tak, aby bola vhodná pre naše použitie. Definičné obory (domény) zobrazení w_i obmedzíme na podmnožiny priestoru X.

Definícia: (PIFS) Nech X je kompletný metrický priestor, $D_i \subset X, i = 1,...,n$. Potom PIFS je množina kontraktívnych zobrazení $w_i : D_i \to X$, $i = 1,...,n$.

2.4.1 Kódovanie obrazu pomocou PIFS

Budeme pracovať s priestorom funkcií (obrazov) $F = \{f : I^2 \to R\}$, a použijeme suprémovú metriku d_{sup}.

Veta: Priestor F s metrikou d_{sup} je kompletný.

Definícia: Hovoríme, že transformácie $w_1, ..., w_n$ pokrývajú I^2, ak pre všetky $f \in F$ platí, že $\bigcup_{i=1}^{n} w_i(f) \in F$.

To znamená, že ak aplikujeme w_i na časť obrazu $f \cap (D_i \times I)$ nad doménou D_i, tak výsledok bude graf funkcie nad R_i , $I^2 = \bigcup_{i=1}^{n} R_i$ (bloky pokrývajú celý obraz) a bloky R_i budú disjunktné.

Definícia: Nech $w : R^3 \to R^3$ je transformácia, kde $w(x,y,z_1) = (x' \; y', z_1')$ a $w(x,y,z_2) = (x', y', z_2')$. Potom w sa nazýva *z-kontraktívne*, ak existuje $0 < s < 1$ také, že

$$|z_1' - z_2'| \le s|z_1 - z_2|,$$

a x' a y' sú nezávislé od z_1 alebo z_2 pre všetky x, y, z_1, z_2.

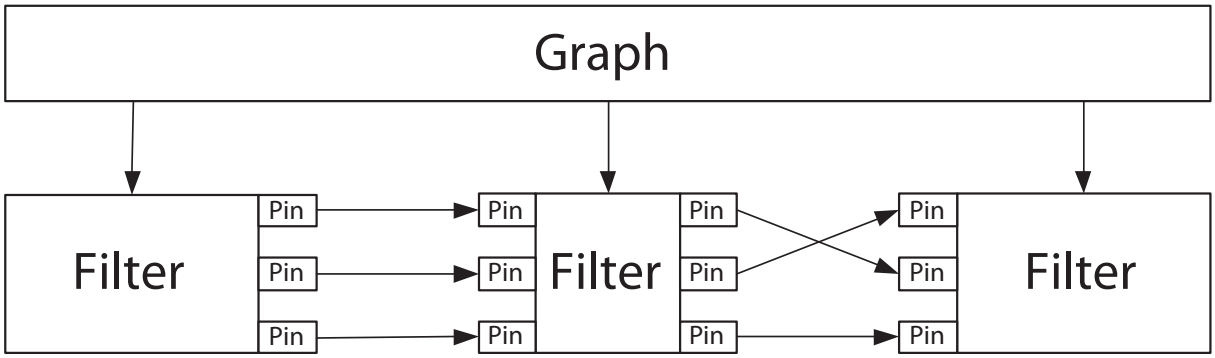
Lema: Ak $w_1, ..., w_n$ sú z-kontraktívne, potom $W = \bigcup_{i=1}^{n} w_i$ je kontraktívne v priestore F so suprémovou metrikou.

Vlastnosť pokrytia nám zabezpečuje, že výsledok W bude znovu obrazová funkcia, a môžeme ju preto použiť v iteračnom cykle. Kódovanie predlohy f spočíva v rozdelení f na vhodné bloky R_i a nájdenie vhodných kontraktívnych zobrazení w_i s doménami D_i tak, aby výsledný pevný bod zobrazenia W, f', bol čo najbližšie k pôvodnej f (minimalizácia $d(f, f')$).

3. Návrh a implementácia

3.1 Architektúra a implementácia

Ako pomocnú architektúru pre podporu riadenia postupnosti operácií použijeme architektúru „graf filtrov“ (Filtergraph). Použijeme ju kvôli možnosti ľahko meniť proces spracovania dát, napr. zmenou grafu alebo nastavením filtrov. Tento návrh je inšpirovaný technológiou DirectShow z balíka Microsoft DirectX používanou na spracovanie a riadenie toku multimediálnych dát (na rozdiel od DirectShow, tento návrh nepodporuje spracovanie tokov dát (streaming)). Základom architektúry sú tri základné triedy: TGraph, TFilter a TPin. Trieda TPin je základné komunikačné rozhranie poskytujúce funkcie spájania pinov a predávania dát medzi nimi. Trieda TFilter je virtuálna trieda, ktorá poskytuje priestor pre implementáciu vlastných algoritmov. Objekty triedy TFilter komunikujú medzi sebou pomocou objektov triedy TPin. TFilter riadi základné operácie s pinmi ako pridávanie a odoberanie pinov. Každá odvodená trieda má priestor na implementáciu hlavného algoritmu, konfiguračných funkcií ako napr. konfiguračný dialóg, inicializačnej a ukončovacej funkcie. Trieda TGraph riadi spúšťanie jednotlivých filtrov, zabezpečuje beh a ukončenie spracovania, monitoruje priebeh spracovania.

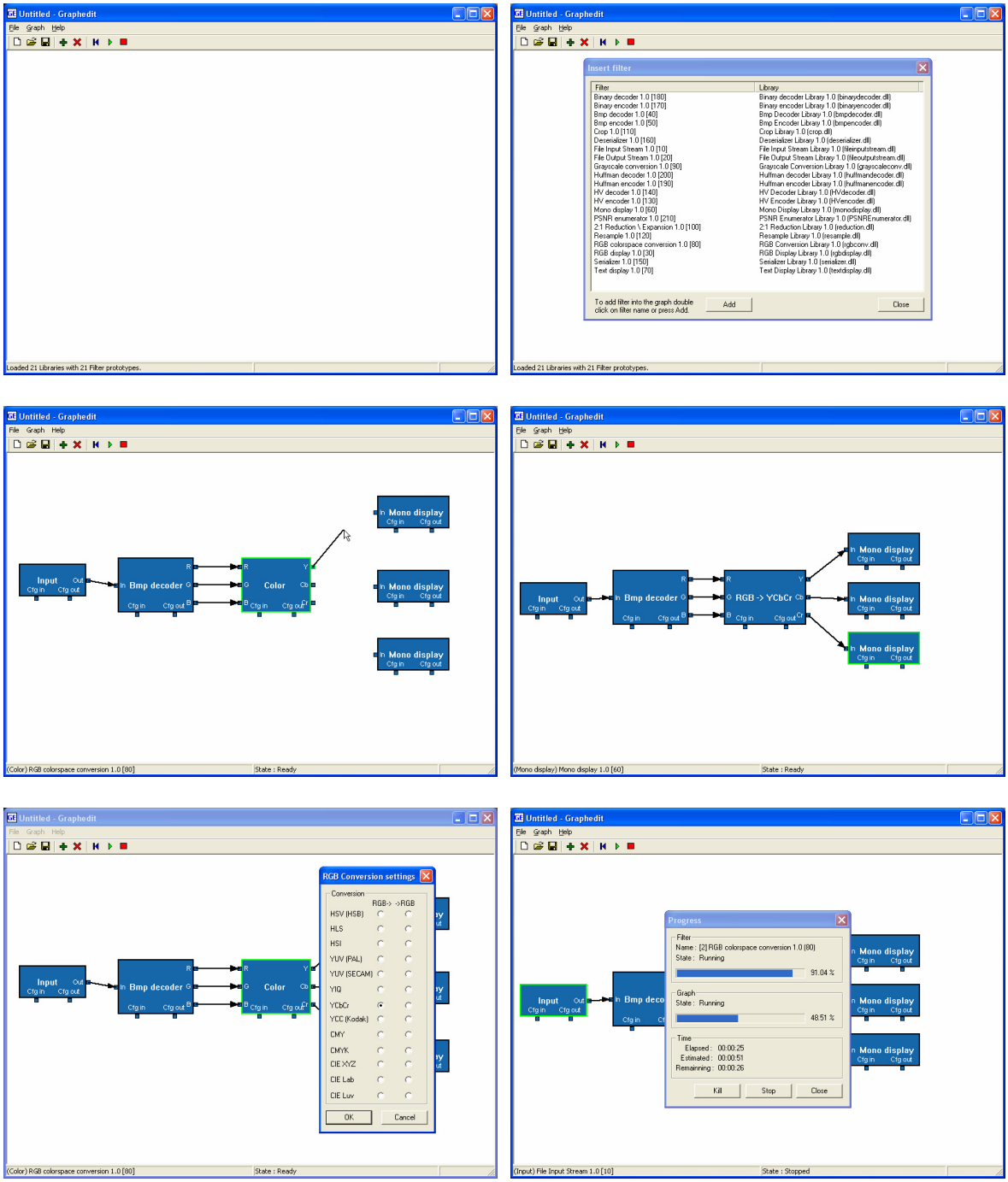


Obrázok 8: Architektúra „graf filtrov“.

Všetky knižnice a programy v tomto projekte sú implementované v jazyku C++. Väčšina knižníc je závislá na platforme Microsoft Windows. Triedy filtrov sú zabalené v dynamicky linkovateľných knižniciach (DLL). Jedna knižnica filtrov môže obsahovať niekoľko tried filtrov. Každá trieda má jednoznačný identifikátor – Magic number. Magic number by malo byť jednoznačné v rámci všetkých použitých tried filtrov (t.j. nielen v rámci jednej knižnice ale aj v rámci ostatných knižníc). Na prácu s grafmi slúži aplikácia Graphedit, ktorá dokáže pracovať s knižnicami filtrov.

3.2 Program Graphedit

Program Graphedit slúži na prácu s grafmi. Umožňuje jednoduchú interaktívnu vizuálnu editáciu grafu, pridávanie a odoberanie filtrov, spájanie vybraných pinov, nastavovanie parametrov filtrov, uloženie grafu do súboru (aj s nastaveniami filtrov) a jeho opätovné načítanie. Taktiež poskytuje funkcie spúšťania, ukončenia a monitorovania behu grafu. Jednotlivé knižnice filtrov sú pripojiteľné do aplikácie vo forme zásuvných modulov (Plugin DLLs).



Obrázok 9: Program Graphedit.

3.3 Triedy filtrov

Triedy filtrov sú triedy odvodené od základnej triedy TFilter. Každá trieda by mala mať svoje vlastné identifikačné číslo (Magic number) a musí implementovať virtuálne funkcie predpísané v triede TFilter. V implementácii architektúry Filergraph sú preddefinované základné komunikačné dátové typy podporujúce spracovanie obrazovej informácie (obrazová rovina, obrazová rovina trojíc alebo štvoríc, pole celých alebo desatinných čísel, pamäťové buffre, ...). Všetky grafické operácie sú implementované s použitím operácií v plávajúcej rádovej čiarke (floating point) s dvojnásobnou presnosťou (v jazyku C++ je to 64 bitový typ *double*) teda je možné spracovávať obrazy vyššej presnosti (ako tradičná 8 bitová) alebo väčšieho dynamického rozsahu (High Dynamic Range - HDR).

Projekt obsahuje tieto triedy filtrov:

Súborový vstup – výstup:
File input stream – vstup zo súboru, načíta súbor do pamäťového buffra
File output stream – výstup do súboru, uloží pamäťový buffer do súboru

Kódovanie a dekódovanie formátov:
Bmp decoder – dekodér súborov BMP, rozdelí obraz vo formáte BMP na farebné roviny R, G a B, podporuje len 24 bitový formát bez kompresie
Bmp encoder – kodér súborov BMP, skombinuje farebné roviny RGB do formátu BMP

Zobrazovanie:
RGB display – farebné zobrazovacie okno, okno s tromi vstupmi pre kanály R, G a B
Mono display – monochromatické zobrazovacie okno, okno s jedným mono vstupom, ktorý zobrazuje v odtieňoch šedej
Text display – zobrazovač textovej informácie, dokáže zobraziť celé číslo, desatinné číslo a reťazec znakov

Spracovanie farby:
RGB conversion – konverzia farebných modelov z a do farebného modelu RGB
Grayscale conversion – konverzia farebného modelu RGB na šedo tónovú škálu

Zmena rozlíšenia:
2:1 reduction / expansion – redukcia alebo expanzia rozlíšenia na polovicu alebo dvojnásobok, ide o rýchlu redukciu priemerovaním a rýchlu expanziu interpoláciou
Crop – orezanie obrazu, vyreže z obrazu istú časť
Resample – zmena rozlíšenia obrazu v ľubovoľnej mierke, prevzorkuje obraz do ľubovoľného rozlíšenia s použitím nastaveného konvolučného filtra

Fraktálne kódovanie:
HV encoder – fraktálny kodér založený na horizontálno-vertikálnej schéme
HV decoder – dekodér fraktálne kódovaných dát

Práca s pamäťovými bufframi:
Serializer – spájajúci filter, spojí niekoľko pamäťových buffrov do jedného a pridá informáciu o ich usporiadaní
Deserializer – rozdeľujúci filter, rozdelí spojené pamäťové buffre

Kódovanie polí celých čísel:
Binary encoder – binárny kodér, binárne zakóduje pole celých čísel použitím iba potrebného počtu bitov
Binary decoder – binárny dekodér, dekóduje binárne zakódované dáta
Huffman encoder – huffmanov kodér, zakóduje pole celých čísel huffmanovým kódom
Huffman decoder – huffmanov dekodér, dekóduje dáta zakódované huffmanovým kódom

Meranie:
PSNR enumerator – merač podobnosti dvoch obrazov na základe RMS metriky

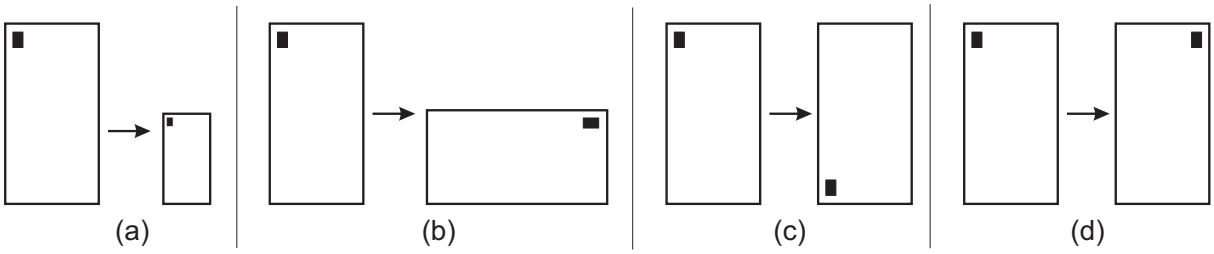
Bližší popis filtrov a ich nastavení možno nájsť v dokumentácii k jednotlivým filtrom.

3.4 Trieda HV encoder (fraktálny kodér)

Trieda HV encoder je implementáciou horizontálno-vertikálnej schémy fraktálnej transformácie. Vstupom je šedo tónový obraz (v ľubovoľnom dynamickom rozsahu) a výstupom je sada afinných transformácií popisujúca IFS systém, ktorého atraktor je podobný vstupnému obrazu.

3.4.1 Transformácie

HV encoder je kvôli časovej náročnosti prehľadávania obmedzený len na afinné transformácie. HV schéma rozdeľuje obraz na obdĺžnikové bloky. Kvôli tomu je množina afinných transformácie ešte zúžená len na určité vhodné typy. Uvažujeme len tieto typy afinných transformácií: posunutie, škálovanie (kvôli kontraktívnosti len zmenšenie), preklopenie okolo horizontálnej osi (horizontal flip), preklopenie okolo vertikálnej osi (vertical flip), otočenie o 90° doprava a ich kombinácie. Zložením týchto transformácií môžme docieliť otočenie aj o iné uhly (180°, 270°). Týmto sme množinu možných transformácií dostatočne zúžili, čím sme zabezpečili ľahké kódovanie transformácie, ale tiež sme ponechali dostatočnú voľnosť, čím sme pokryli veľkú časť množiny atraktorov. Transformácia je teda určená blokom (pozícia a rozmery), doménou (pozícia a faktory zväčšenia šírky a výšky vzhľadom na veľkosť bloku) a modifikáciou domény (otočenie o 90°, horizontálne a vertikálne preklopenie). Pozícia bloku a domény určuje posúvaciu časť transformácie, faktory zväčšenia šírky a výšky určujú škálovaciu časť transformácie, modifikáciu domény treba uložiť osobitne.

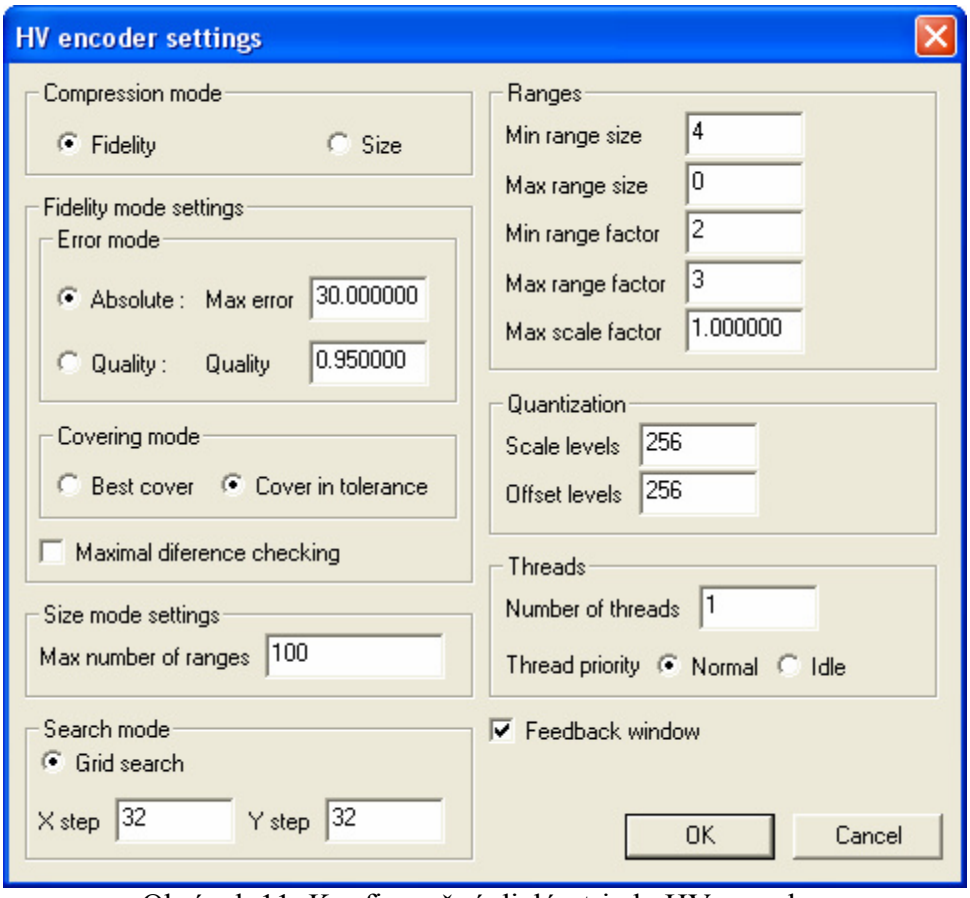


Obrázok 10: Použité transformácie: (a) škálovanie, (b) otočenie o 90°doprava, (c) horizontálne preklopenie, (d) vertikálne preklopenie.

Modifikáciu domény zakódujeme tromi bitmi: 1 bit pre otočenie o 90°, 1 bit pre horizontálne preklopenie a 1 bit pre vertikálne preklopenie. Posúvaciu a škálovaciu časť transformácie budeme kódovať iným spôsobom.

3.4.2 Nastavenia a parametre

Prácu filtra HV encoder môžme ovládať nastavením množiny parametrov pred jeho spustením. Zmenami parametrov je možné ovládať hlavne cieľovú kvalitu výstupu, veľkosť výstupu a čas potrebný na spracovanie. Parametre je možné nastaviť v konfiguračnom dialógu filtra (dvojklikom na box filtra v programe Graphedit).



Obrázok 11: Konfiguračný dialóg triedy HV encoder.

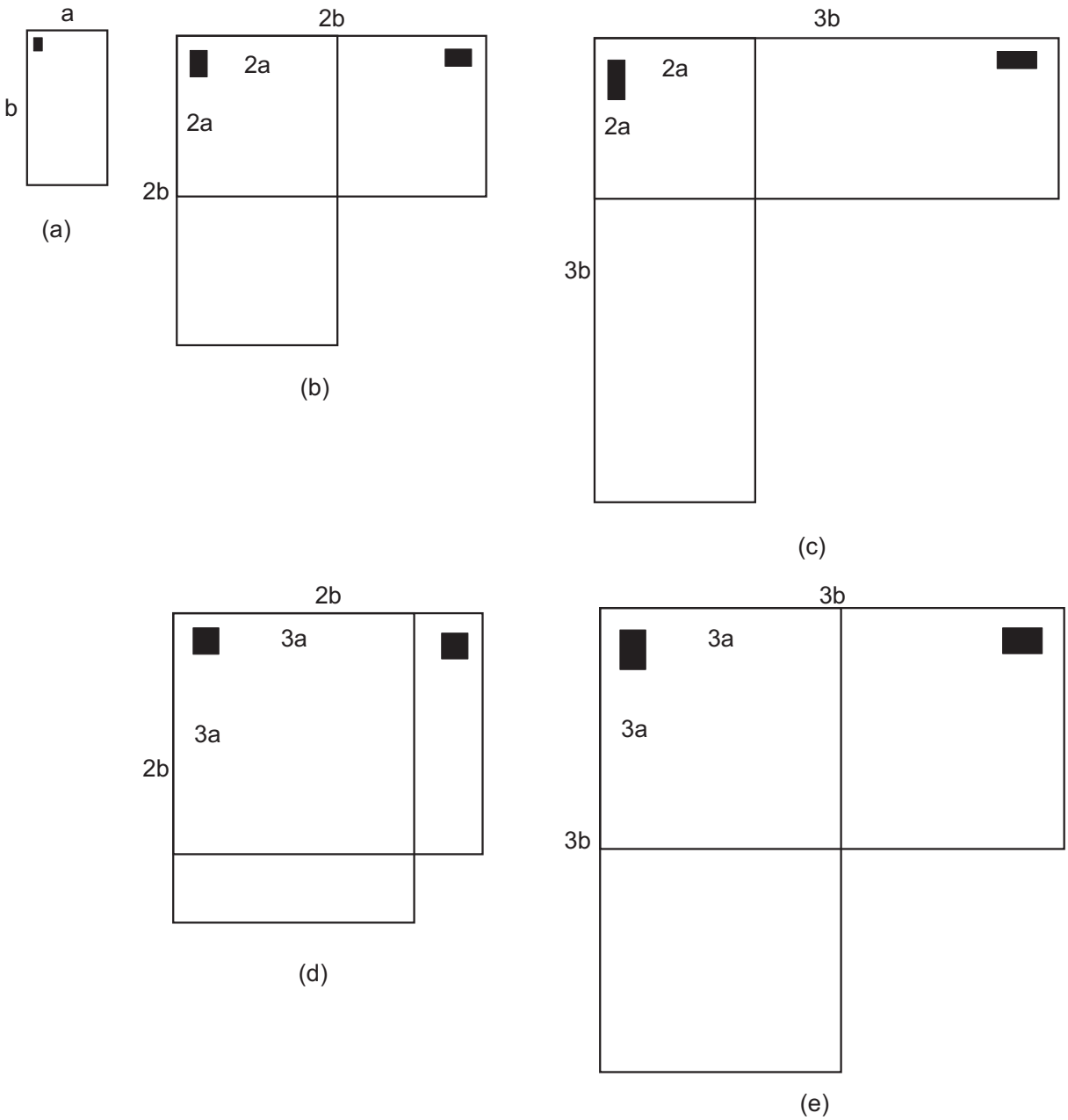
Compression mode: Mód kompresie alebo voľba kompresného algoritmu. Tu máme na výber či je našim cieľom dosiahnuť stanovenú vernosť (voľba *Fidelity*) alebo veľkosť (voľba *Size*). Mód *Fidelity* neprihliada na množstvo transformácií, ktoré generuje ale snaží sa nájsť pokrytie v nastavenej tolerancii. Mód *Size* hľadá najlepšie pokrytie s použitím ohraničeného počtu blokov (transformácií), čím zabezpečí, že veľkosť výstupu bude zhora ohraničená.

Fidelity mode settings: Nastavenia módu *Fidelity*.
Error mode: Mód chyby. Algoritmus používa RMS metriku na porovnanie podobnosti medzi pokrývaným blokom a zmenšenou doménou. Voľba *Absolute* nastaví priamo hodnotu maximálnej povolenej RMS odchýlky (alebo chyby) na hodnotu *Max error* ($MaxError \geq 0$). Pri voľbe *Quality* si algoritmus vypočíta sám maximálnu povolenú odchýlku z dynamického rozsahu vstupného obrazu. Hodnotou *Quality* ($Quality \in <0.0,1.0>$) môžme túto toleranciu ovplyvniť. Hodnota 1.0 znamená maximálnu kvalitu (žiadna tolerancia, akceptovaná len RMS odchýlka 0.0 t.j. žiadna chyba), naopak hodnota 0.0 znamená najnižšiu kvalitu. Väčšie hodnoty *Quality* spôsobia generovanie väčšieho množstva blokov a tým väčšiu vernosť ale aj veľkosť výstupu.
Covering mode: Mód pokrývania. Voľba *Best cover* dovoľuje algoritmu pokračovať a dokončiť prehľadávanie aj keď už bolo nájdené pokrytie v danej tolerancii. Algoritmus sa snaží nájsť najlepšie pokrytie bloku aj za cenu väčšej časovej náročnosti. Voľba *Cover in tolerance* túto možnosť nepovoľuje a prehľadávanie sa zastaví pri nájdení prvého pokrytia v danej tolerancii.
Maximal diference checking: Kontrola maximálnej odchýlky. Hovorí algoritmu aby udržiaval tiež maximálnu odchýlku bloku v tolerancii (nie len RMS odchýlku). Týmto je udržiavaná vyššia vernosť (väčšinou za cenu zväčšenia veľkosti výstupu). Tento parameter je zatiaľ len v experimentálnom stave a nie je zaručená 100%-ná funkčnosť.

Size mode settings: Nastavenia módu *Size*.
Max number of ranges: Maximálny povolený počet blokov ($MaxNumRanges \geq 0$). Po dosiahnutí nastaveného počtu blokov sa algoritmus zastaví.

Search mode: Mód prehľadávania. Určuje akým spôsobom sa budú vyberať domény pre bloky. Momentálne je k dispozícii len voľba *Grid search*. V móde *Grid search* sú domény vyberané zaradom z pozícií v mriežke s nastavenou hustotou každých *X step* pixelov v horizontálnom smere a *Y step* pixelov vo vertikálnom smere. Tento spôsob prehľadávania je časovo náročný.

Ranges: Nastavenia obmedzujúce veľkosť, škálovacie faktory bloku a škálovací faktor domény. Tieto nastavenia sú striktne dodržiavané a ich nevhodné nastavenie môže spôsobiť nedosiahnutie cieľa (t.j. strata cieľovej kvality alebo prekročenie cieľovej veľkosti, v závislosti od nastaveného módu kompresie) alebo nájdenie nevhodného atraktora.
Min range size: Minimálna povolená veľkosť bloku ($MinRangeSize \geq 0$). Blok, ktorý má obidva rozmery menšie alebo rovné ako nastavená hodnota, už nebude rozdelený na menšie bloky (aj keby to bolo potrebné). Hodnota *MinRangeSize=0* znamená neobmedzené delenie bloku (t.j. dovoľuje algoritmu deliť blok až do rozmerov 1x1). Väčšie hodnoty zamedzujú generovaniu malých blokov, ale môžu spôsobiť stratu kvality.
Max range size: Maximálna povolená veľkosť bloku ($MaxRangeSize \geq 0$). Blok, ktorý má aspoň jeden rozmer väčší ako nastavená hodnota, nebude pokrývaný a bude hneď rozdelený na menšie bloky (aj keby počet blokov presiahol povolenú hranicu). Hodnota *MaxRangeSize=0* povoľuje neobmedzenú veľkosť bloku. Menšie hodnoty zamedzujú pokrývaniu veľkých blokov (pre ktoré sa väčšinou nenájde vhodné pokrytie), čím sa ušetrí čas, ale môžu spôsobiť generovanie veľkého množstva blokov, čím sa zväčší veľkosť výstupu.
Min range factor: Minimálny pomer veľkosti bloku a domény ($MinRangeFactor \geq 2$).
Max range factor: Maximálny pomer veľkosti bloku a domény ($MaxRangeFactor \geq 2$). Tieto dva parametre ovplyvňujú výber domén. Na jednej pozícii sú vybraté domény so všetkými možnými kombináciami škálovacích faktorov, ktoré sú ešte kombinované s otočením (t.j. ak *MinRangeFactor=2* a *MaxRangeFactor=3* tak na jednej pozícii je vybratých a porovnaných 8 domén škálovaných nasledovne: 2x2, 2x2o, 2x3, 2x3o, 3x2, 3x2o, 3x3, 3x3o, vzhľadom na veľkosť bloku). Veľký rozsah faktorov spôsobí značné predĺženie pokrývania (čas potrebný na pokrytie rastie exponenciálne). Takisto veľké hodnoty faktorov spomaľujú pokrývanie.
Max scale factor: Maximálny povolený škálovací faktor domény ($MaxScaleFactor \geq 0$, mal by byť menší ako *1.0* ale nie je to podmienkou). Pri pokrývaní bloku vybratou doménou sa vypočíta najlepší škálovací faktor a posunutie (v jasovej zložke obrazu, nie v priestorovej) pre dosiahnutie minimálnej RMS odchýlky. Ak ale škálovací faktor je väčší ako *1.0* výsledný IFS systém nemusí konvergovať resp. atraktor môže mať neobmedzené rozmery. Preto, ak vypočítaný škálovací faktor je väčší ako nastavená hodnota, pokrytie je odmietnuté.



Obrázok 12: Veľkosti domén pre *MinRangeFactor=2* a *MaxRangeFactor=3* (a) pôvodný blok s rozmermi *a*x*b*. Kombinácie: (b) 2x2 (*2a*x*2b*) a 2x2o (*2b*x*2a*), (c) 2x3 (*2a*x*3b*) a 2x3o (*3b*x*2a*), (d) 3x2 (*3a*x*2b*) a 3x2o(*2b*x*3a*), (e) 3x3 (*3a*x*3b*) a 3x3o (*3b*x*3a*).

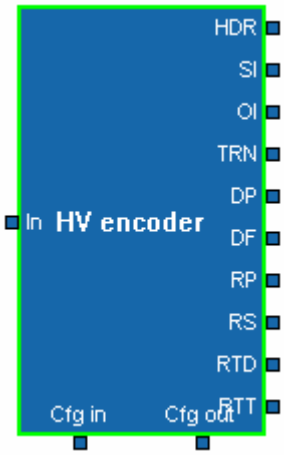
Quantization: Nastavenia kvantizácie. Hodnoty škálovacieho faktoru a posunutia vypočítané pri pokrývaní sa ďalej kvantujú lineárnym kvantizátorom a z kvantovaných hodnôt sa vypočíta RMS odchýlka. Väčší počet kvantizačných úrovní dovolí zachovať kvalitu, ale môže zväčšiť veľkosť výstupu. Menší počet úrovní znižuje kvalitu, ale zmenšuje veľkosť výstupu. V istých prípadoch menší počet úrovní môže spôsobiť generovanie väčšieho množstva blokov a tým zväčšiť výstup (hlavne pri vyšších kvalitách, kedy kvantovanie značne zníži kvalitu pokrytia bloku).
Scale levels: Počet kvantizačných úrovní pre škálovací faktor domény ($ScaleLevels \geq 1$).
Offset levels: Počet kvantizačných úrovní pre posunutie ($OffsetLevels \geq 1$).

Threads: Nastavenia výpočtových vlákien. Trieda HV encoder je navrhnutá a implementovaná s možnosťou spolupráce viacerých výpočtových vlákien, ktoré zabezpečujú paralelné pokrývanie jedného bloku. Týmto dovoľuje využiť možnosti viac

jadrových a viac procesorových systémov a urýchliť proces pokrývania. Táto možnosť je v experimentálnom stave a nie je zaručená 100%-ná funkčnosť (odporúčame používať len jedno vlákno kvôli korektnosti výsledkov).
Number of threads: Počet paralelných spolupracujúcich vlákien ($NumThreads \geq 1$).
Thread priority: Nastavenie systémovej priority výpočtových vlákien. Voľba *Normal* nastaví normálnu prioritu, čo môže spôsobiť dlhšie nereagovanie systému kvôli vyťaženiu procesora. Voľba *Idle* nastaví prioritu na minimálnu hodnotu, čo zaručí reakcie systému, ale môže skresliť (predĺžiť) merané časy výpočtov.

Feedback window: Okno vizualizácie algoritmu. Voľba povolí alebo zakáže zobrazenie vizualizačného okna, cez ktoré je možné monitorovať stav a priebeh algoritmu. Zapnutá vizualizácia môže výrazne spomaliť priebeh pokrývania.

3.4.3 Výstupy



Obrázok 13: HV encoder.

Výstup triedy HV encoder je množina transformácií. Každá transformácia je určená blokom, doménou a modifikáciou domény (kódovanou vyššie uvedeným spôsobom). Trieda poskytuje výstup rozdelený do niekoľkých výstupných štruktúr podľa charakteru výstupnej informácie. Informácie o transformáciách sú rozdelené kvôli ich rôznorodosti. Rôzne časti majú rôzny charakter a je vhodné ich kódovať rôznym spôsobom.

Výstupy triedy HV encoder podľa pinov:
HDR (Header): Hlavičková štruktúra. Obsahuje základné informácie o predlohe ako rozlíšenie, dynamický rozsah, niektoré parametre použité pri kódovaní, a dáta potrebné na korektné dekódovanie.

SI (Scale Indicies): Kvantované indexy škálovacích faktorov, kódované ako pole celých čísel.

OI (Offset Indicies): Kvantované indexy posunutí, kódované ako pole celých čísel.

TRN (Transformations): Zakódované informácie o modifikáciách domén, kódované ako pole celých čísel.

DP (Domain Positions): Pozície domén predelené hodnotami *X step* pre x-ovú pozíciu a *Y step* pre y-ovú pozíciu, kvôli zmenšeniu hodnôt a možnosti efektívnejšieho kódovania. Kódované po dvojiciach ako pole celých čísel.

DF (Domain Factors): Škálovacie faktory blokov, kódované po dvojiciach ako pole celých čísel.

RP (Range Positions): Pozície blokov, kódované po dvojiciach ako pole celých čísel.

RS (Range Sizes): Veľkosti blokov, kódované po dvojiciach ako pole celých čísel.

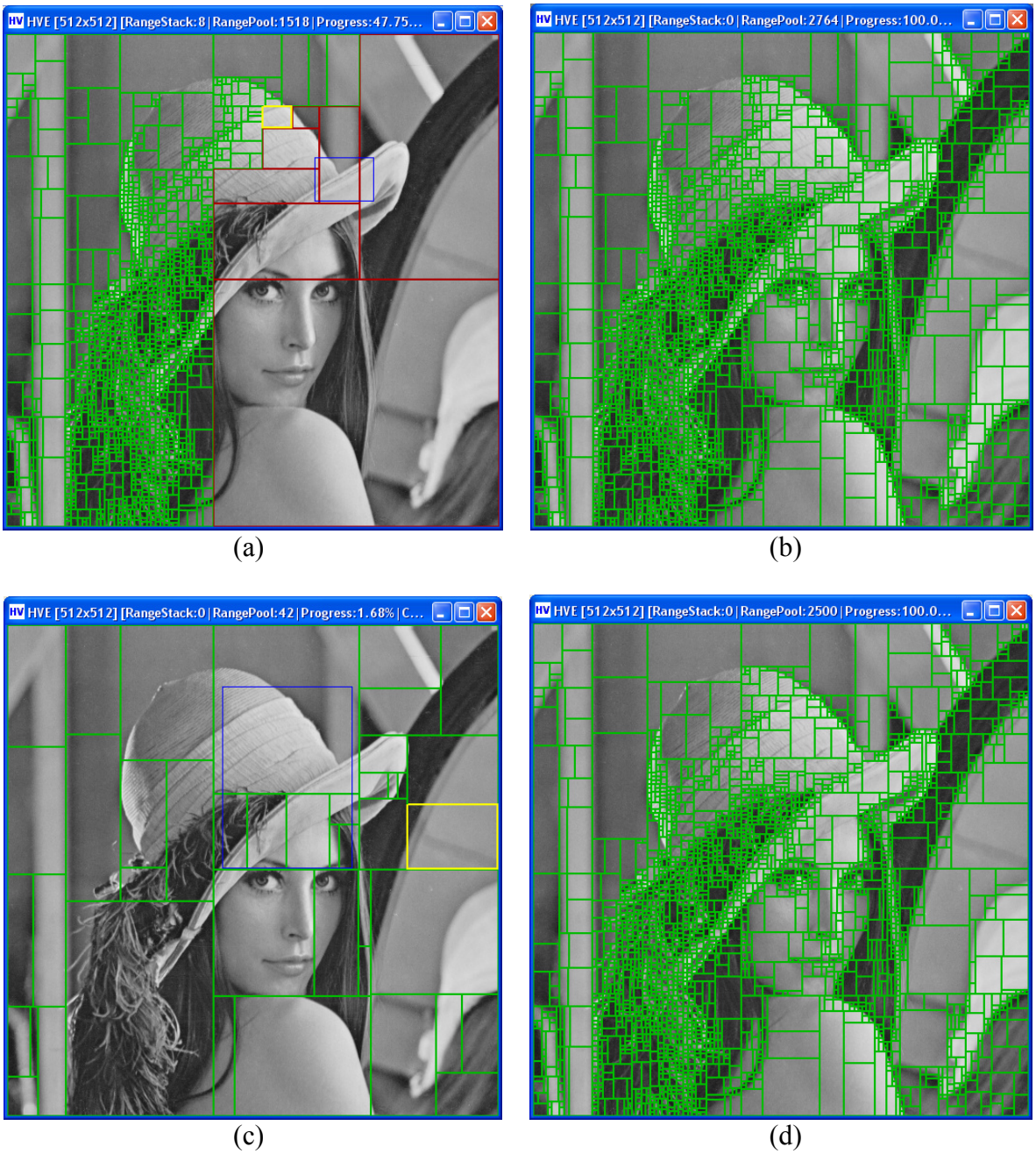
RTD (Range Tree Distancies): Deliace vzdialenosti v strome blokov, kódované traverzom stromu ako pole celých čísel.

RTT (Range Tree Types): Typy delení v strome blokov (typ určuje či je blok rozdelený horizontálne alebo vertikálne), kódované traverzom stromu ako pole celých čísel.

Všetky výstupné polia celých čísel sú synchronizované, teda paralelným prechodom cez všetky polia dostaneme prislúchajúce informácie o transformáciách. Napr. transformáciu *i* dostaneme zložením informácií z SI[*i*], OI[*i*], TRN[*i*], DP[*2i*], DP[*2i+1*], DF[*2i*], DF[*2i+1*], RP[*2i*], RP[*2i+1*], RS[*2i*], RS[*2i+1*]. Dvojica výstupných polí (RP, RS) obsahuje rovnaké informácie ako dvojica (RTD, RTT), ktoré sú však zapísané rôznou formou. Dvojica (RP, RS) kóduje pozície a veľkosti blokov priamo. Dvojica (RTD, RTT) ich kóduje stromom, ktorý sa vytváral pri výpočte. Pozície a veľkosti blokov môžme rekonštruovať z týchto údajov tak, že znovu vytvoríme strom blokov a vyberieme z neho listové vrcholy. Tieto dve dvojice poskytujú redundantné informácie a preto je vhodné kódovať len jednu z nich. Prvky z dvojice (RP, RS) majú približne rovnaký charakter. Naopak prvky z poľa RTD (vzdialenosti) majú iný charakter ako prvky poľa RTT (typy – čísla 0 a 1). Dvojicu (RTD, RTT) je možné efektívnejšie kódovať a preto budeme používať ďalej len ju.

3.4.4 Vizualizácia algoritmu

Vizualizácia sa zapne len pri zapnutí voľby *Feedback window* v konfiguračnom dialógu. Potom môžme v okne sledovať priebeh delenia a pokrývania blokov. V pozadí je zobrazený vstupný obraz. V popredí sa vykresľuje strom blokov. Zelenou farbou sa zobrazujú už pokryté bloky, červenou ešte nepokryté bloky, a žltou blok, ktorý je práve pokrývaný. Modrou farbou sa zobrazujú práve testované domény (môže ich byť aj viac v závislosti od počtu výpočtových vlákien). V záhlaví okna sa zobrazujú štatistické informácie ako počet blokov v zásobníku (*RangeStack*), počet pokrytých blokov (*RangePool*), percentuálna kompletnosť operácie (*Progress*) a korektnosť pokrytia (*CC* – Correct Cover, ak je hodnota TRUE znamená to, že pokrytie je zatiaľ korektné, ak FALSE, znamená to, že algoritmus bol donútený akceptovať pokrytie bloku, ktoré nevyhovuje nastaveným požiadavkám, napr. bolo potrebné rozdeliť blok ale kvôli jeho malej veľkosti to nebolo dovolené).



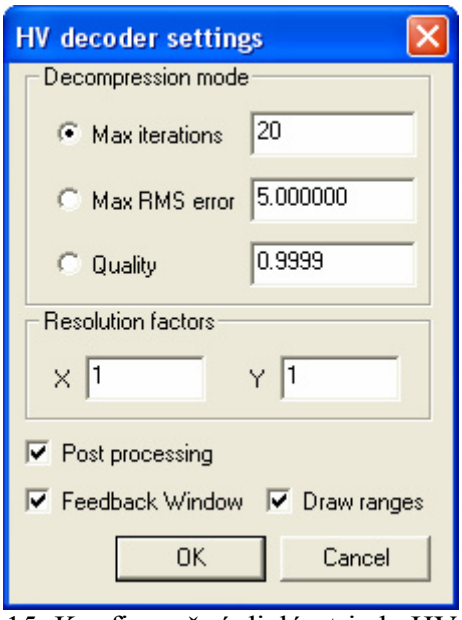
Obrázok 14: Vizualizácia pokrývania (a) priebeh pokrývania v móde *Fidelity* , (b) ukončené pokrývanie (a), (c) priebeh pokrývania v móde *Size* , (d) ukončené pokrývanie (c).

3.5 Trieda HV decoder (fraktálny dekodér)

Trieda HV decoder je implementáciou iteračného dekodéra pre IFS systém popísaný triedou HV encoder. Vstupom je IFS systém zakódovaný vo výstupných štruktúrach triedy HV encoder a výstupom je šedo tónový obraz blízky atraktoru vstupného IFS. Dekódovací proces prebieha v iteráciách, kedy sa aplikujú jednotlivé transformácie na predošlú iteráciu. Ako počiatočný vstup do dekodéra je použitý čierny obraz (obsahujúci len nuly). Iteračným prístupom nie je možné dosiahnuť atraktor, teda výstup je len priblíženie sa k tomuto atraktoru. Vzdialenosť výstupu a atraktora môžeme kontrolovať nastavením parametrov dekodéra.

3.5.1 Nastavenia a parametre

Dekódovací proces môžeme doladiť množinou nastavení dekodéra. Taktiež môžeme zapnúť vizualizáciu práce dekodéra.



Obrázok 15: Konfiguračný dialóg triedy HV decoder.

Decompression mode: Mód dekompresie. Určuje za akých podmienok sa má dekódovací proces skončiť. Voľba *Max iterations* ($MaxIterations \geq 0$) nastaví maximálny povolený počet iterácií. Po dosiahnutí tohto počtu sa proces zastaví. Voľba *Max RMS error* ($MaxRMSError \geq 0$) nastaví maximálnu povolenú odchýlku od atraktora na danú hodnotu (odchýlku od atraktora môžeme z hora odhadnúť z RMS odchýlky dvoch po sebe idúcich iterácií). Proces končí pri dosiahnutí danej blízkosti k atraktoru. Voľba *Quality* ($Quality \in < 0.0, 1.0 >$) nastaví maximálnu povolenú RMS odchýlku od atraktora v závislosti od dynamického rozsahu kódovanej predlohy načítaného z hlavičkovej štruktúry. Hodnota *1.0* znamená najvyššiu kvalitu (žiadna odchýlka od atraktora), hodnota *0.0* znamená najnižšiu kvalitu.

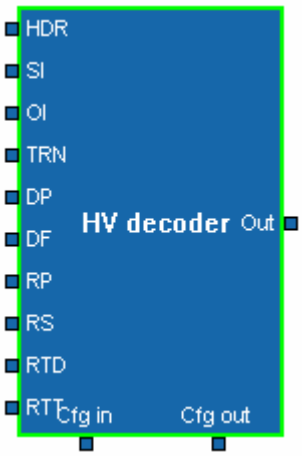
Resolution factors: Zmena výstupného rozlíšenia. Výstupné rozlíšenie možno škálovať celočíselnými faktormi v smere x faktorom X ($X \geq 1$) a v smere y faktorom Y ($Y \geq 1$).

Post processing: Dodatočné spracovanie. Voľba povolí dodatočné vyhladenie iteračného výstupu. Vyhladenie sa aplikuje na okolie hrán blokov, kvôli zmierneniu nežiaducich blokovitých artefaktov, ktoré vznikajú pri kódovaní v nižších kvalitách.

Feedback window: Vizualizačné okno. Voľba zapne vizualizačné okno, v ktorom môžeme sledovať priebeh dekódovacieho procesu a vidieť jednotlivé iterácie. Zapnutá vizualizácia môže výrazne spomaliť dekódovací proces.

Draw ranges: Zobrazovanie blokov. Voľba povolí zobrazovanie stromu blokov vstupného IFS vo vizualizačnom okne.

3.5.2 Vstupy

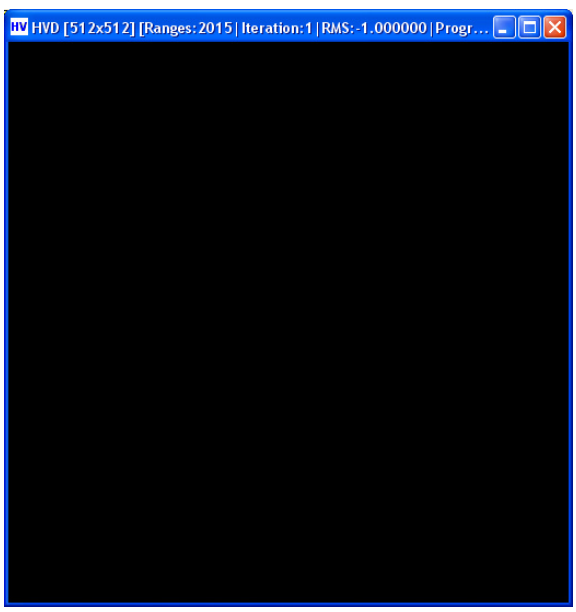


Obrázok 16: HV decoder.

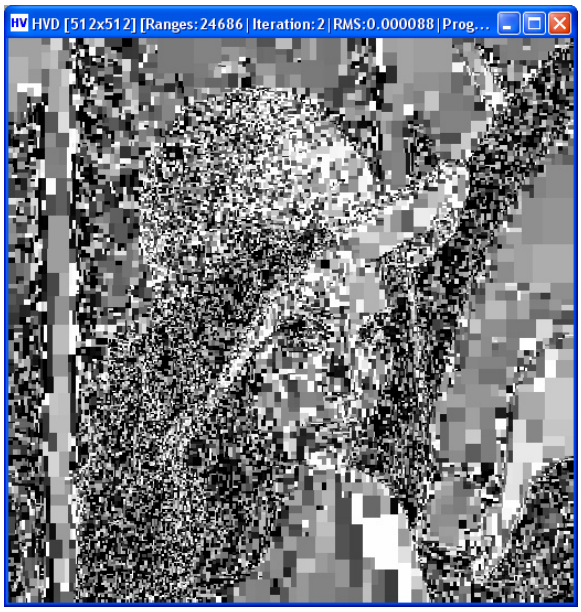
Vstupom dekodéra je IFS systém zakódovaný vo výstupných štruktúrach triedy HV encoder. Na rekonštrukciu IFS systému sú potrebné vstupy HDR, SI, OI, TRN, DP, DF, a jedna z dvojíc (RP,RS) , (RTD,RTT).

3.5.3 Vizualizácia algoritmu

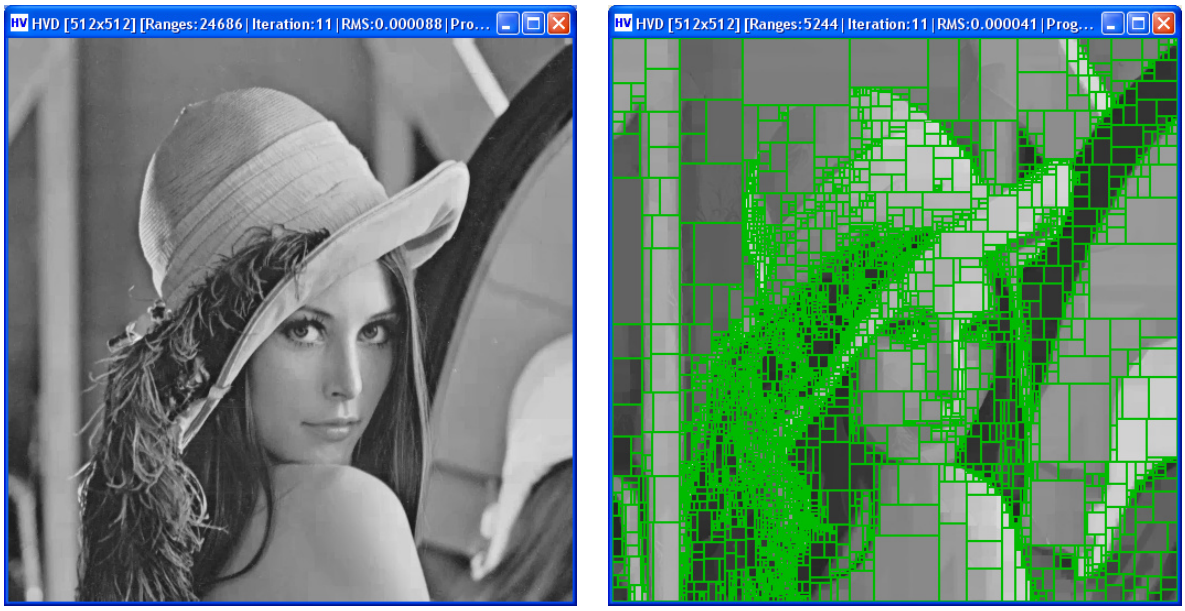
V okne môžeme sledovať postupnosť iteračných krokov. V záhlaví okna sa zobrazujú priebežné štatistické informácie ako rozlíšenie výstupu, počet blokov (*Ranges*), iterácia (*Iteration*), RMS odchýlka od atraktora (*RMS*) a percentuálna kompletnosť operácie (*Progress*).



(a)



(b)



(c) (d)

Obrázok 17: Vizualizácia iterovania, iterácia (a) 0, (b) 1, (c) 10. (d) strom blokov.

3.6 Kódovanie výstupov triedy HV encoder

3.6.1 Triedy Binary encoder a Binary decoder

Trieda Binary encoder analyzuje pole celých nezáporných čísel, vypočíta maximálny počet potrebných bitov *M* na uloženie najväčšieho prvku. Potom prvky poľa uloží do pamäťového buffra v zhustenej podobe použitím *M* bitov na každý prvok. Trieda Binary decoder dekóduje takto zakódované dáta späť na pole celých čísel.

3.6.2 Triedy Huffman encoder a Huffman decoder

Trieda Huffman encoder analyzuje pole celých nezáporných čísel, vytvorí frekvenčnú tabuľku znakov a na jej základe vytvorí Huffmanov kód. Potom zakóduje prvky poľa v zhustenej podobe použitím vytvorených kódových slov. Trieda umožňuje analýzu poľa na základe viac prvkových znakov (*n*-gramov, kedy *n* po sebe idúcich prvkov poľa tvorí jeden znak), ale v ďalších testoch budeme používať len jednoduché znaky (1-gramy). Trieda Huffman decoder dekóduje takto zakódované dáta späť na pole celých čísel.

3.7 Iné použité triedy

3.7.1 RGB conversion (zmena farebného modelu)

Trieda slúžiaca na konverziu farebných modelov z a do farebného modelu RGB. Dokáže pracovať s farebnými modelmi HSV (HSB), HLS, HSI, YUV (PAL), YUV (SECAM), YIQ, YCbCr, YCC (Kodak), CMY, CMYK, CIE XYZ, CIE Lab a CIE Luv. Vstupom je trojica šedo tónových rovín reprezentujúca RGB model a výstupom je n-tica šedo tónových rovín reprezentujúca výstupný model (alebo opačne, podľa zvoleného smeru prevodu). V ďalších testoch budeme používať hlavne model YCbCr.

3.7.2 Grayscale conversion (šedo tónový prevod)

Trieda slúži na konverziu farebného modelu RGB na šedo tónovú škálu. Poskytuje dve metódy prevodu farby na úrovne šedej: maximovú a trilineárnu kombináciu (s nastaviteľnými aj preddefinovanými koeficientmi).

3.7.3 2:1 reduction / expansion (redukcia rozlíšenia)

Trieda poskytuje redukciu rozlíšenia na polovicu. Vstupom je šedo tónový obraz. Redukciu môžme robiť v horizontálnom smere, vertikálnom smere alebo oboch (v tomto prípade sa objem vstupného obrazu zredukuje až na 1/4 pôvodnej veľkosti). Redukujeme jednoduchým priemerovaním dvojíc alebo štvoríc pixelov. Taktiež môžme robiť inverznú operáciu – expanziu rozlíšenia na dvojnásobok (v prípade oboch smerov až na štvornásobok). Expandujeme jednoduchou interpoláciou chýbajúcich hodnôt.

3.7.4 PSNR enumerator (merač PSNR)

Trieda poskytujúca štatistické informácie o podobnosti dvoch vstupných šedo tónových obrazov. Poskytuje hodnoty PSNR (*Peek Signal to Noise Ratio*) vychádzajúcu z dynamického rozsahu vstupných obrazov, RMS odchýlku (*Root Mean Square*), MD (*Maximal Diference*) – maximálnu odchýlku pixelov na rovnakej pozícii, AD (*Average Diference*) – priemernú odchýlku pixelov na rovnakej pozícii.

4. Testy

V tejto kapitole opíšeme a vyhodnotíme niekoľko testov. Testy sú zamerané hlavne na porovnanie kvality, stupňa kompresie a času komprimácie. Testy na šedo tónových obrazoch sa zameriavajú na porovnanie vplyvu rôznych hodnôt parametrov tried HV encoder a HV decoder, testy na farebných obrazoch sa zameriavajú na farebný model YCbCr v rôznych módoch. Množinu nastavení triedy HV encoder a triedy HV decoder môžme zhrnúť do profilov použitých pri testoch.

Parameter \ Profil	HVE-P1(q)	HVE-P2(q)	HVE-P3(n)	HVE-P4(n)
Compression mode	Fidelity	Fidelity	Size	Size
Error mode	Quality	Quality	–	–
Quality	q	q	–	–
Covering mode	Cover in tolerance	Cover in tolerance	–	–
Max diference checking	nie	nie	nie	nie
Max number of ranges	–	–	n	n
Grid search X step	32	32	32	32
Grid search Y step	32	32	32	32
Min range size	1	1	1	1
Max range size	0	0	0	0
Min range factor	2	2	2	2
Max range factor	2	2	2	2
Max scale factor	1.0	1.0	1.0	1.0
Quantization scale levels	256	64	256	64
Quantization offset levels	256	64	256	64
Number of threads	1	1	1	1
Thread priority	Normal	Normal	Normal	Normal
Feedback window	nie	nie	nie	nie

Tabuľka 1: Profily nastavení triedy HV encoder.

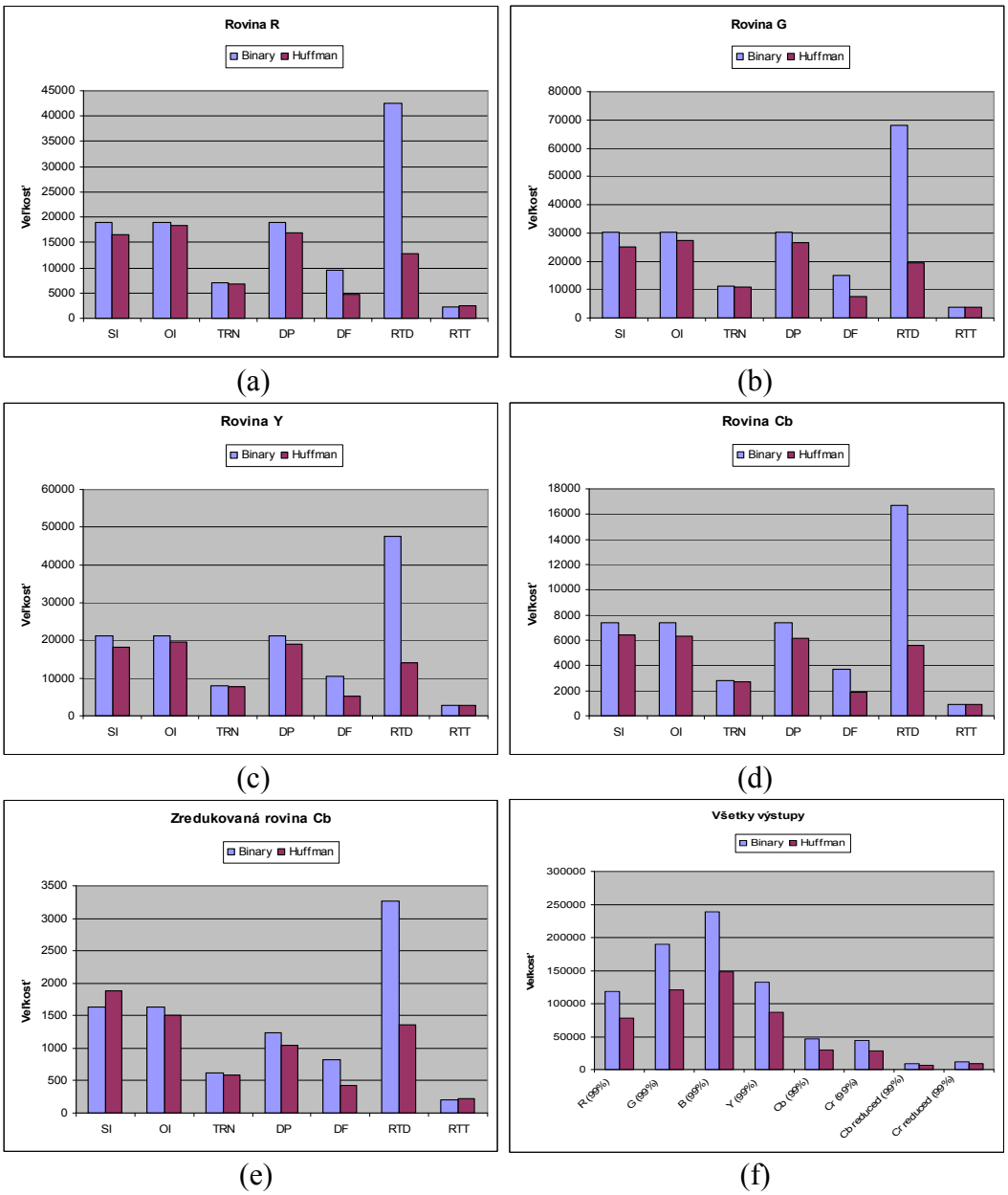
Parameter \ Profil	HVD-P1	HVD-P2
Decompression mode	Quality	Quality
Max iterations	–	–
Max RMS error	–	–
Quality	0.9999	0.9999
Resolution factor X	1	1
Resolution factor Y	1	1
Postprocessing	nie	áno
Feedback window	nie	nie
Draw ranges	nie	nie

Tabuľka 2: Profily nastavení triedy HV decoder.

Všetky časové údaje boli merané na systéme s procesorom AMD AthlonXP 2800+ (jadro Barton, 2250 MHz, 512 kB cache), a pamäťou 512 MB (DDR, 432 MHz). V tejto kapitole budeme pod pojmom *Lena24bit* rozumieť testovací farebný obraz Lena o rozmeroch 512 x 512 pixelov a farebnej hĺbke 8bit na každú farebnú zložku (v móde RGB888) a veľkosti 786 486 bajtov (v BMP formáte) a pod pojmom *Lena8bit* obraz Lena24bit spracovaný filtrom Grayscale conversion na šedo tónový obraz hĺbky 8bit o veľkosti 262 162 bajtov (Y zložka v modeli YCbCr).

4.1 Efektívnosť ukladania výstupov triedy HV encoder

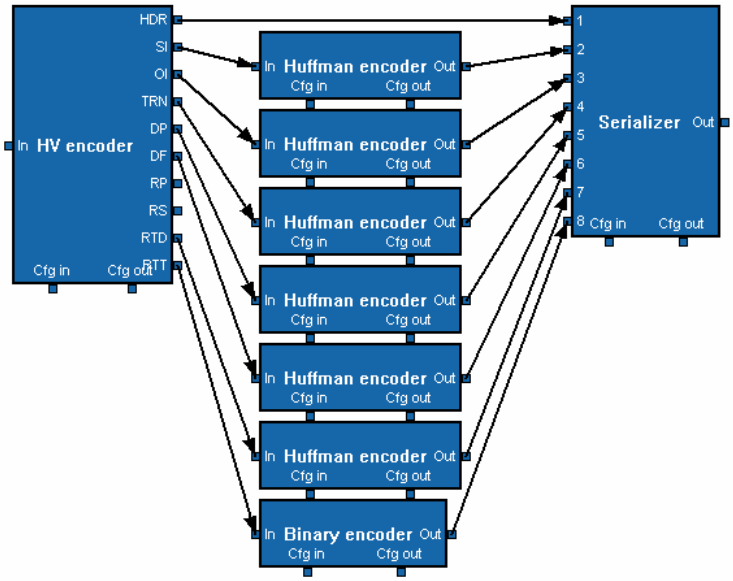
Tento test sa zameriava na zistenie, aké kódovanie je vhodné použiť na konkrétne výstupy, v kombinácii s farebným modelom. Ako zdroj použijeme testovací obrázok Lena24bit. Na kódovanie použijeme profil HVE-P1(99%). Budeme testovať veľkosť výstupu pre farebné roviny R, G, B, Y, Cb, Cr, Cb zredukované na 1/4 a Cr zredukované na 1/4. Výstupy budeme kódovať binárnym a huffmanovým kodérom. Zakódované veľkosti porovnáme.



Graf 1: Porovnanie veľkosti zakódovaných výstupov triedy HV encoder. Roviny: (a) R, (b) G, (c) Y, (d) Cb, (e) zredukované Cb. (f) Všetky výstupy kódované spoločne jedným typom kodéru.

Výsledky dosiahnuté pre roviny R, G a B sú veľmi podobné (preto je uvedené len porovnanie rovín R a G v Grafe 1). Takisto výsledky pre roviny Cb a Cr sú veľmi podobné. Z grafu 1 (f) je vidieť, že roviny Y, Cb, Cr, sa dajú zakódovať efektívnejšie ako roviny R, G a B. Keď porovnávame efektivitu kodérov pre jednotlivé výstupy, vidieť, že huffmanov kodér je vhodné použiť na výstupy SI, OI, TRN, DP, DF, RTD a binárny kodér na výstup RTT

(huffmanov kodér tu dáva menej dobré výsledky, kvôli nutnosti ukladania frekvenčnej tabuľky, a kvôli rovnomernej distribúcii prvkov RTT).



Obrázok 18: Zapojenie filtrov pre kódovanie výstupov triedy HV encoder.

Preto v ďalších testoch budeme používať zapojenie filtrov z obr. 18 na kódovanie výstupov triedy HV encoder.

4.2 Test triedy HV encoder, kvalita a veľkosť výstupu

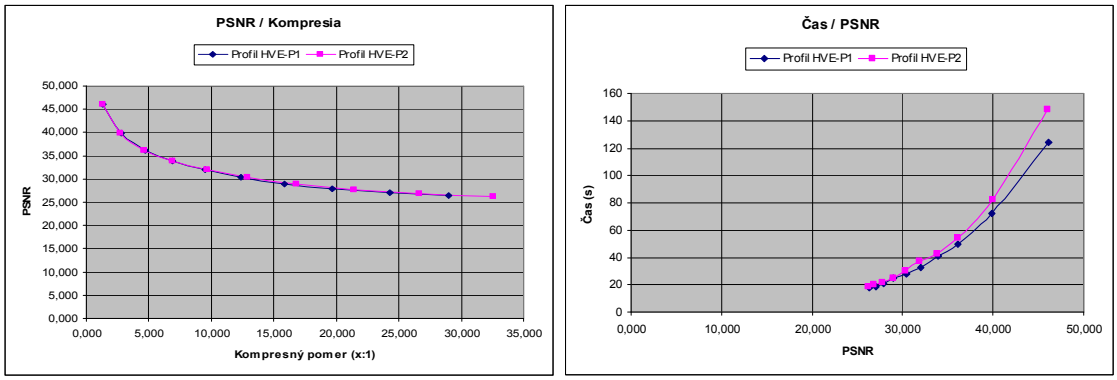
V tomto teste porovnáme kvalitu a veľkosť komprimovaného obrazu Lena8bit v profiloch HVE-P1 a HVE-P2 pre rôzne nastavenia kvality, a HVE-P3 a HVE-P4 pre rôzny počet blokov. Budeme merať aj čas spracovania. V ďalšom texte budeme pod pojmom *Kompresia* meraného v percentách rozumieť pomer *stlačená veľkosť / pôvodná veľkosť* (napr. kompresia 20% znamená, že obraz sa podarilo zakódovať na 20% pôvodnej veľkosti).

Kvalita (%)	Bitov na pixel	Kompresný pomer (x:1)	PSNR (dB)	Čas (s)	Bloky
95	0,276	29,019	26,369	18	2015
95,5	0,330	24,272	27,030	19	2472
96	0,406	19,691	27,868	21	3136
96,5	0,504	15,864	28,960	25	3995
97	0,648	12,344	30,474	28	5244
97,5	0,844	9,479	31,983	33	6958
98	1,160	6,894	33,932	41	9721
98,5	1,689	4,736	36,124	50	14377
99	2,853	2,805	39,836	72	24686
99,5	5,991	1,335	46,139	124	53260

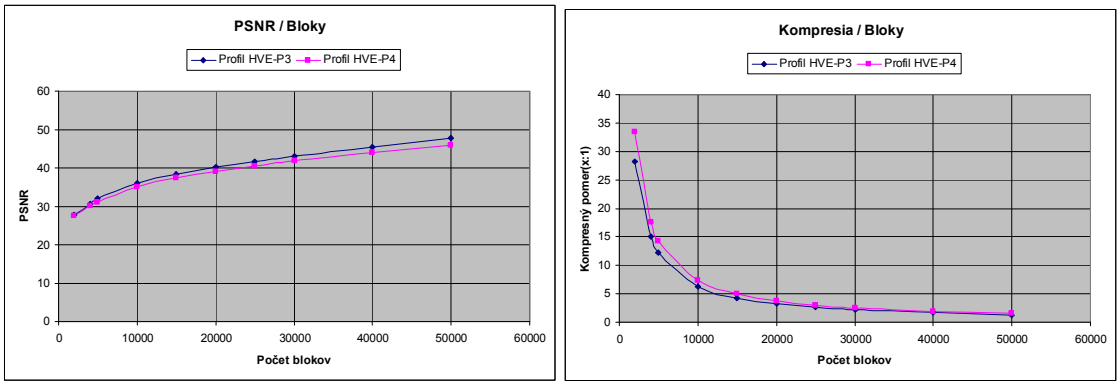
Tabuľka 3: Výsledky pre obraz Lena8bit v profile HVE-P1.

Kvalita (%)	Bitov na pixel	Kompresný pomer (x:1)	PSNR (dB)	Čas (s)	Bloky
95	0,246	32,571	26,209	19	2139
95,5	0,300	26,653	26,820	20	2665
96	0,374	21,404	27,726	22	3393
96,5	0,475	16,838	29,019	25	4386
97	0,622	12,855	30,312	30	5832
97,5	0,826	9,689	31,942	37	7841
98	1,165	6,868	33,848	43	11168
98,5	1,729	4,628	36,085	54	16716
99	2,969	2,695	39,920	82	29126
99,5	6,262	1,278	46,041	148	62744

Tabuľka 4: Výsledky pre obraz Lena8bit v profile HVE-P2.



Graf 2: Porovnanie kompresie, PSNR a času pre profily HVE-P1 a HVE-P2.

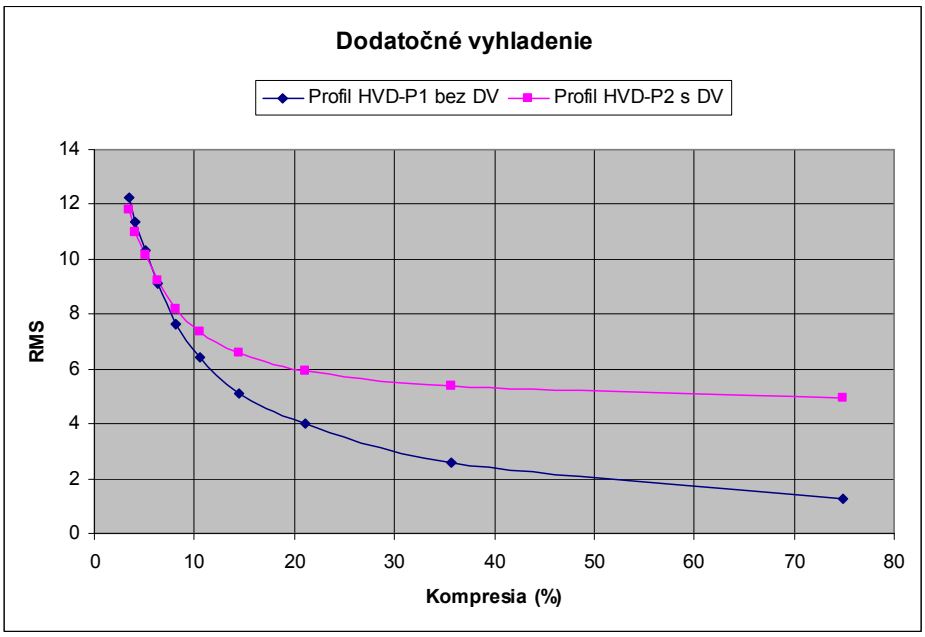


Graf 3: Porovnanie PSNR, kompresie a počtu blokov pre profily HVE-P3 a HVE-P4.

Z grafu 2 vidíme, že s profilom HVE-P2 môžme v istých prípadoch dosiahnuť vyšší kompresný pomer s rovnakou kvalitou za cenu dlhšieho času spracovania. Použiteľné hodnoty kvality sú z intervalu 95%-99,5% (z praktických skúseností). Z grafu 3 vidíme, že pre dosiahnutie vyšších hodnôt PSNR rastie počet potrebných blokov exponenciálne. Výstupy boli dekódované použitím profilu HVD-P1.

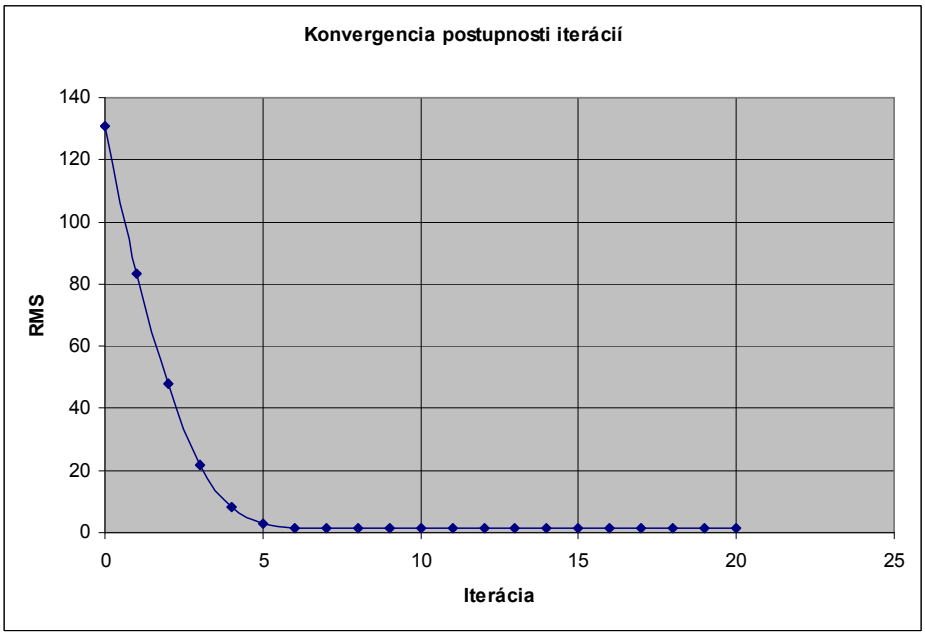
4.3 Test triedy HV decoder, kvalita výstupu a konvergencia iterácií

V tomto teste zistíme vplyv dodatočného vyhladenia obrazu na kvalitu. Pozrieme sa aj na rýchlosť konvergencie postupnosti iteračných krokov. Vychádzať budeme z obrazu Lena8bit zakódovaného v rôznych stupňoch kvality. Vstupom pre test iteračnej postupnosti bude obraz Lena8bit zakódovaný profilom HVE-P1(99,5%).



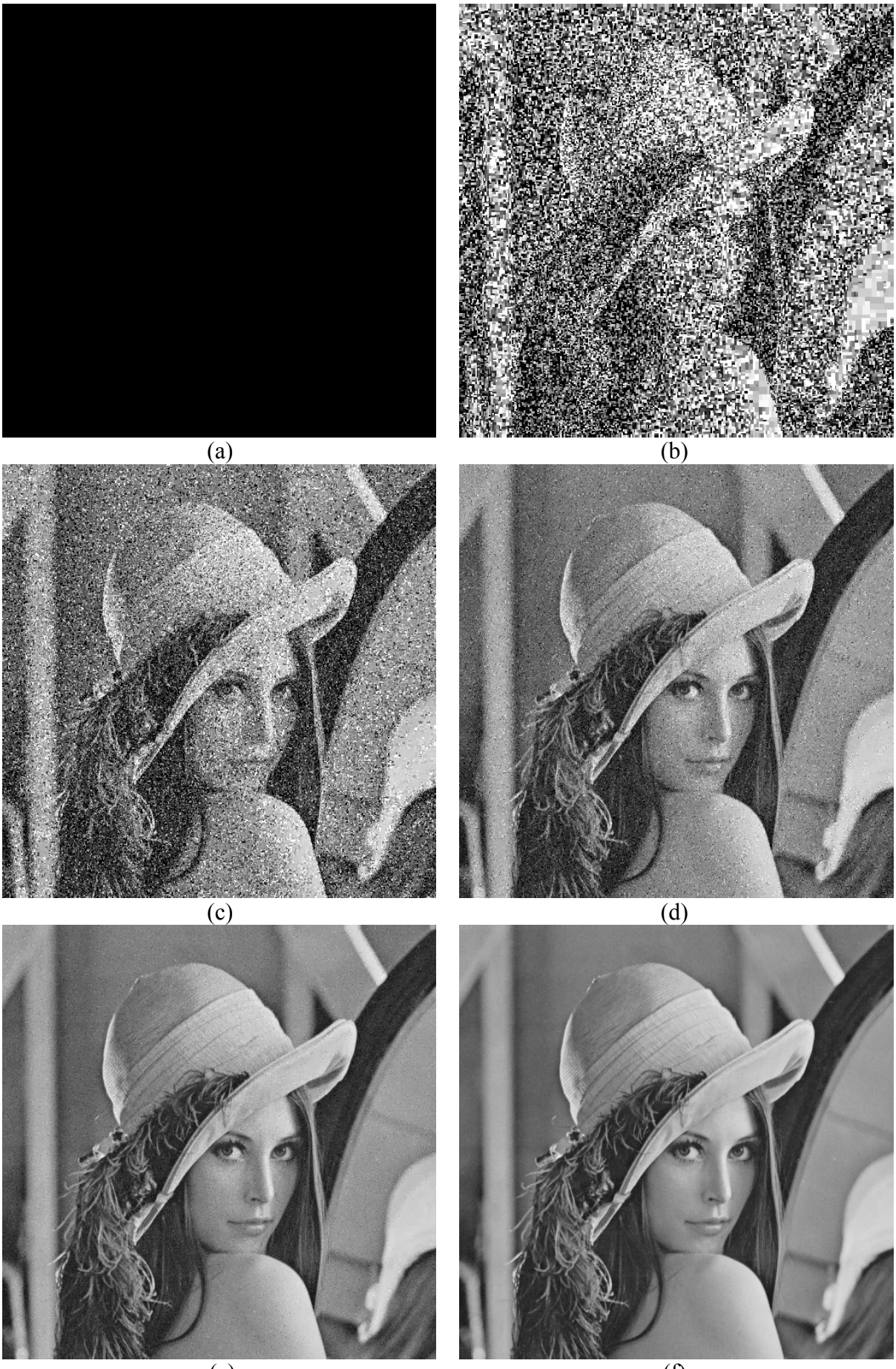
Graf 4: Vplyv dodatočného vyhladenia na RMS odchýlku pre rôzne stupne kvality.

Z grafu 4 vidíme, že pri vyšších kompresných pomeroch vyhladenie dokáže jemne zlepšiť kvalitu obrazu, ale naopak, pri nižších, kvalitu výrazne zhoršuje.



Graf 5: Konvergencia postupnosti iterácií.

Z grafu 5 vidíme, že iteračný proces rýchlo konverguje. Rozdiel medzi 5. a 6. iteráciou už nie je pozorovateľný voľným okom. Väčšinou stačí do 10 iterácií na dostatočné priblíženie sa k atraktoru.



(a) (b)

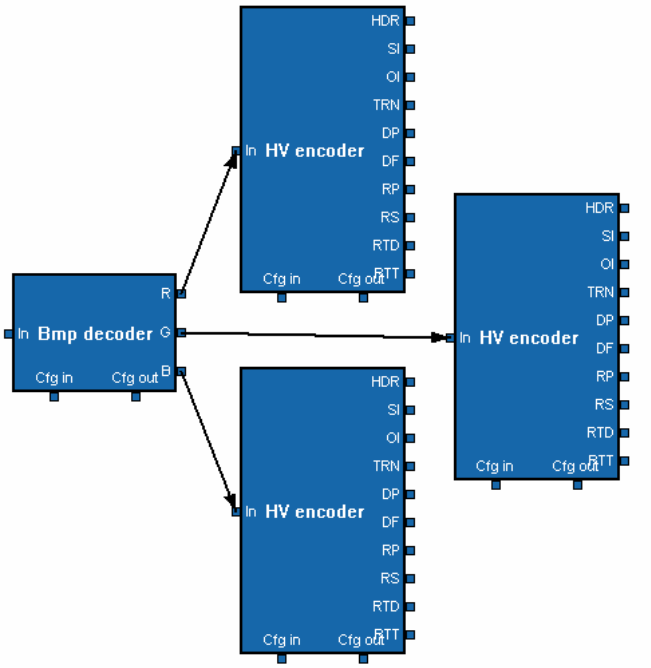
(c) (d)

(e) (f)

Obrázok 19: Postupnosť iterácií. Iterácia (a) 0, (b) 1, (c) 2, (d) 3, (e) 4, (f) 5.

4.4 Test farebného modelu RGB v móde 4:4:4

V tomto teste si predstavíme a otestujeme prvý spôsob ukladania farebného obrazu – mód RGB 4:4:4. Vstupom budú tri farebné roviny R,G a B, ktoré nie sú žiadnym spôsobom upravené (ani farba ani veľkosť). Prevedieme testy s profilmi HVE-P1 a HVE-P2 s rôznymi úrovňami kvality. Použijeme testovací obraz Lena24bit.



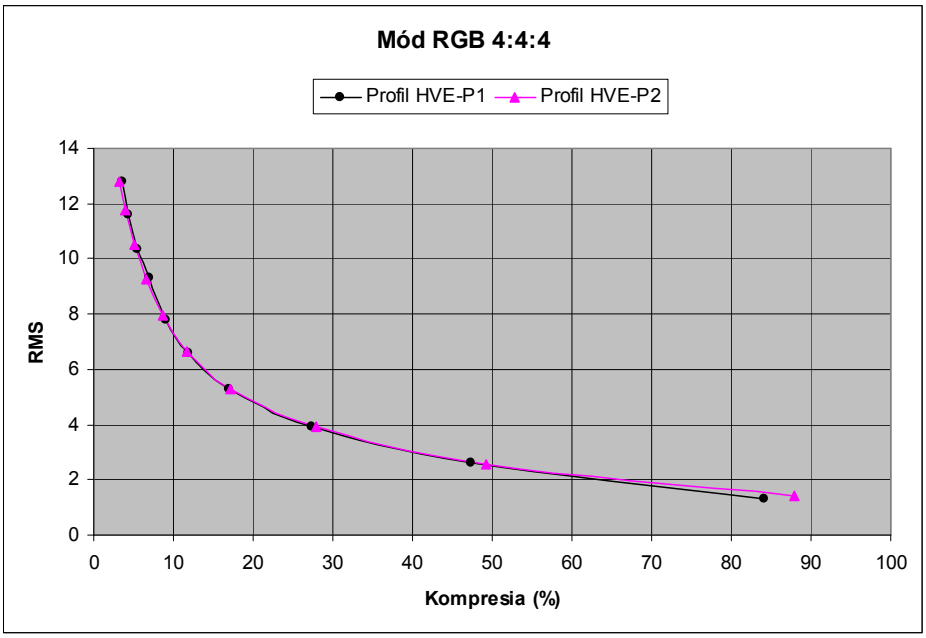
Obrázok 20: Zapojenie filtrov v móde RGB 4:4:4.

Kvalita (%)	Bitov na pixel	Kompresný pomer (x:1)	PSNR (dB)	Čas (s)
95	0,868	27,654	25,999	54
95,5	1,056	22,737	26,838	57
96	1,322	18,158	27,827	67
96,5	1,665	14,412	28,735	78
97	2,163	11,099	30,279	91
97,5	2,870	8,364	31,773	108
98	4,091	5,867	33,665	137
98,5	6,551	3,664	36,235	189
99	11,363	2,112	39,793	277
99,5	20,216	1,187	45,727	419

Tabuľka 5: Výsledky pre profil HVE-P1 v móde RGB 4:4:4.

Kvalita (%)	Bitov na pixel	Kompresný pomer (x:1)	PSNR (dB)	Čas (s)
95	0,774	30,995	25,988	54
95,5	0,957	25,084	26,709	60
96	1,223	19,625	27,680	70
96,5	1,576	15,232	28,804	83
97	2,064	11,627	30,105	95
97,5	2,811	8,537	31,653	115
98	4,099	5,855	33,666	148
98,5	6,708	3,578	36,230	209
99	11,827	2,029	39,890	312
99,5	21,083	1,138	45,169	498

Tabuľka 6: Výsledky pre profil HVE-P2 v móde RGB 4:4:4.

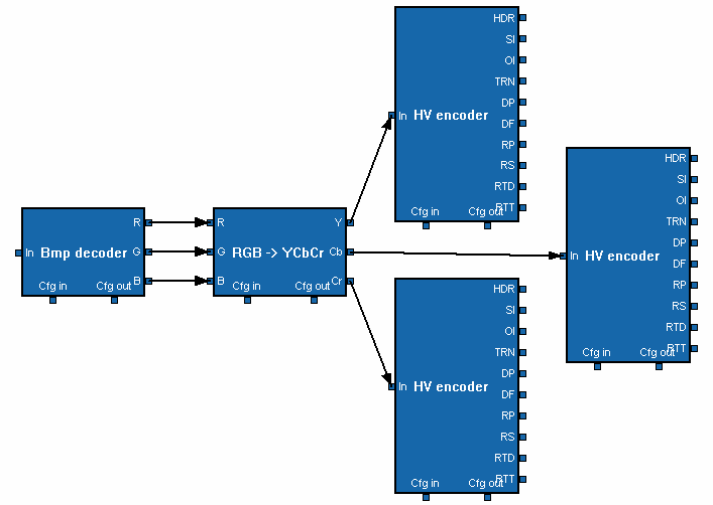


Graf 6: Porovnanie profilov HVE-P1 a HVE-P2 v móde RGB 4:4:4.

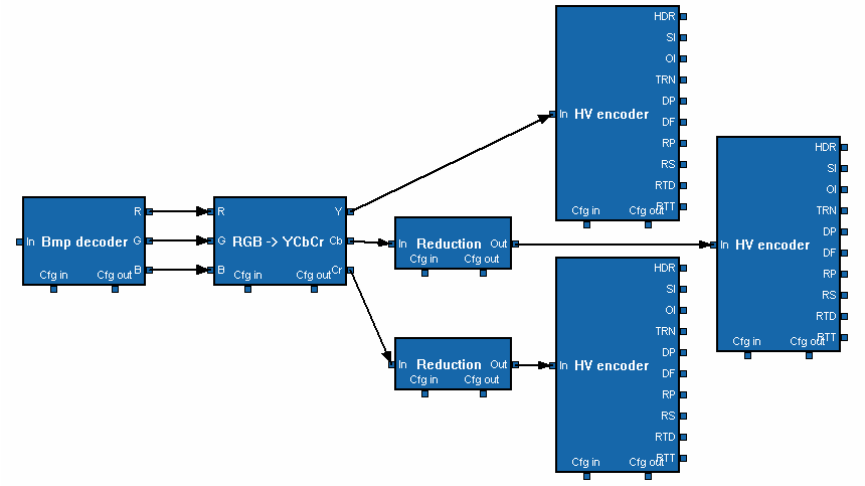
Z grafu 6 vidíme, že profil HVE-P1 dosahuje lepšiu kompresiu pri vyšších kvalitách, naopak lepšiu kompresiu pre nižšie kvality dosahuje profil HVE-P2.

4.5 Test farebného modelu YCbCr v módoch 4:4:4, 4:2:2 a 4:1:1

Tento test sa zameriava na farebný model YCbCr v rôznych módoch. Vstupom budú farebné roviny Y, Cb, Cr. V móde 4:4:4 nebudeme robiť žiadnu redukciu, v móde 4:2:2 zredukujeme roviny Cb a Cr na polovicu veľkosti vo vertikálnom smere, a v móde 4:1:1 zredukujeme roviny Cb a Cr na štvrtinu veľkosti (preto označenie 4:1:1, zo štvorca 2x2 pixely vyberieme štyri vzorky Y, jednu Cb a jednu Cr). Redukciu budeme robiť triedou *2:1 reduction / expansion*. Prevedieme testy s profilom HVE-P1 s rôznymi úrovňami kvality. Použijeme testovací obraz Lena24bit.

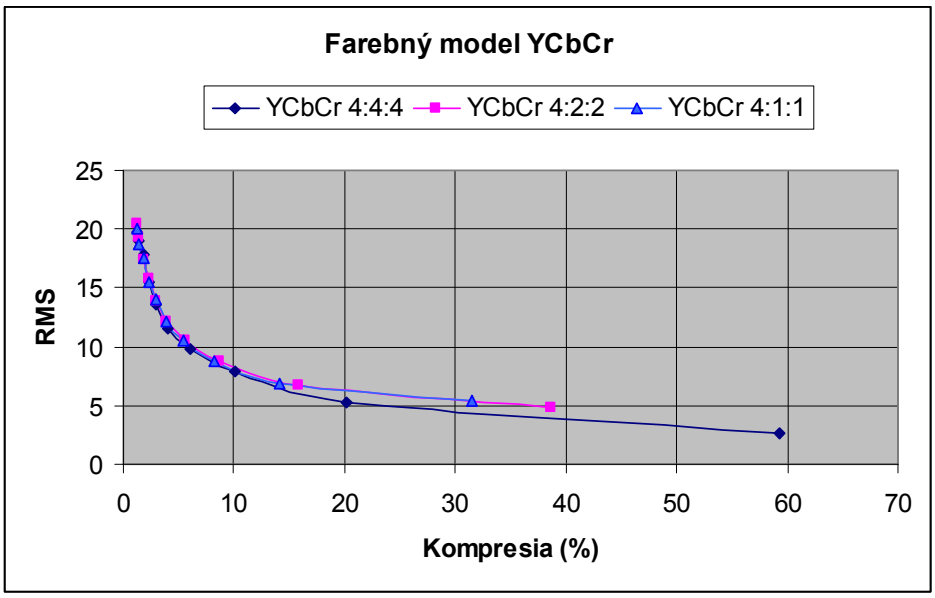


(a)



(b)

Obrázok 21: Zapojenie filtrov pre farebný model YCbCr v módoch (a) 4:4:4 (b) 4:2:2 a 4:1:1.



Graf 7: Porovnanie módov YCbCr 4:4:4, 4:2:2 a 4:1:1.

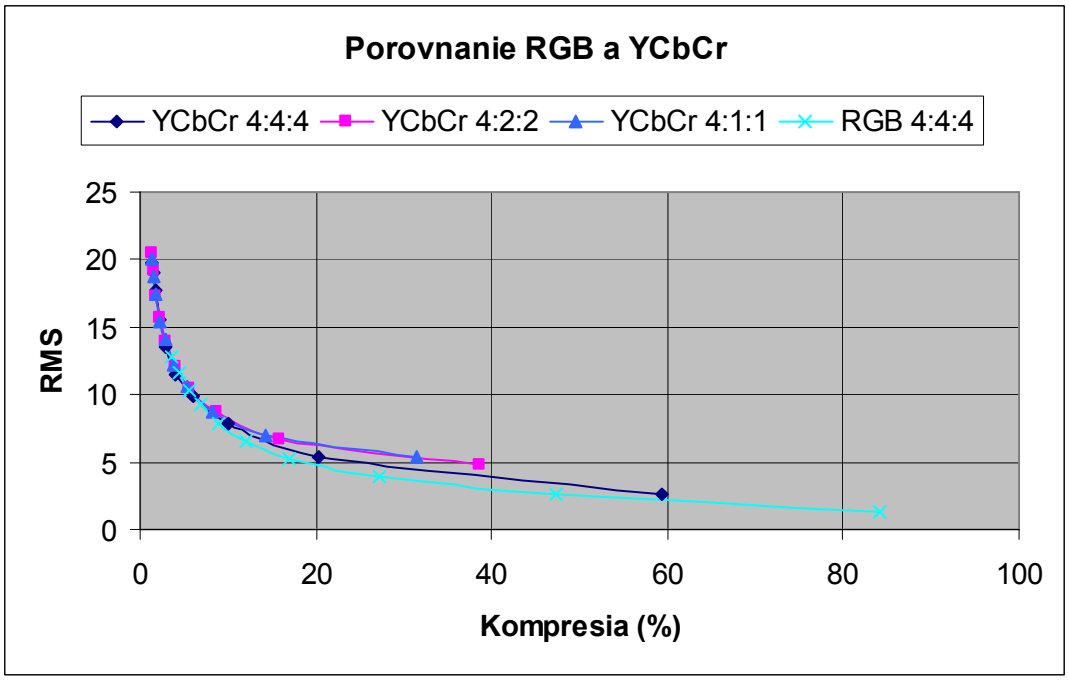
	Kvalita (%)	Bitov na pixel	Kompresný pomer (x:1)	PSNR (dB)	Čas (s)
YCbCr 4:4:4	95	0,298	80,607	22,224	20
	95,5	0,352	68,112	22,562	20
	96	0,435	55,223	23,126	25
	96,5	0,546	43,921	24,300	30
	97	0,717	33,456	25,476	37
	97,5	0,981	24,469	26,898	48
	98	1,448	16,574	28,273	63
	98,5	2,411	9,953	30,222	92
	99	4,853	4,946	33,615	152
	99,5	14,246	1,685	39,543	332
YCbCr 4:2:2	Kvalita (%)	Bitov na pixel	Kompresný pomer (x:1)	PSNR (dB)	Čas (s)
	95	0,298	80,550	21,907	18
	95,5	0,352	68,153	22,448	21
	96	0,433	55,394	23,345	22
	96,5	0,543	44,172	24,198	26
	97	0,710	33,792	25,278	31
	97,5	0,945	25,410	26,501	35
	98	1,349	17,787	27,712	44
	98,5	2,069	11,602	29,339	58
	99	3,784	6,343	31,640	88
	99,5	9,267	2,590	34,527	168
YCbCr 4:1:1	Kvalita (%)	Bitov na pixel	Kompresný pomer (x:1)	PSNR (dB)	Čas (s)
	95	0,298	80,459	22,089	18
	95,5	0,352	68,124	22,670	20
	96	0,433	55,441	23,274	22
	96,5	0,542	44,249	24,342	25
	97	0,700	34,299	25,176	29
	97,5	0,924	25,975	26,404	35
	98	1,303	18,424	27,633	42
	98,5	1,964	12,220	29,311	54
	99	3,403	7,054	31,326	78
	99,5	7,576	3,168	33,445	140

Tabuľka 7: Výsledky pre farebný model YCbCr v módoch 4:4:4, 4:2:2 a 4:1:1.

Z grafu 7 vidíme, že mód 4:4:4 dosahuje vysokú vernosť, ale za cenu nízkej úrovne kompresie. Mód 4:2:2 dosahuje nižšiu vernosť, ale so značne lepšou kompresiou a mód 4:1:1 dosahuje len jemne menšiu vernosť s ešte lepšou kompresiou. Mód 4:1:1 dosahuje aj značne vyššiu rýchlosť, kvôli tomu, že spracúva menší objem dát. Z testovaných módov je teda najvýhodnejší mód 4:1:1 pretože poskytuje najlepšiu kompresiu s nepoznateľným zhoršením kvality oproti módom 4:4:4 a 4:2:2.

5. Výsledky a porovnanie

5.1 Porovnanie farebných modelov RGB a YCbCr

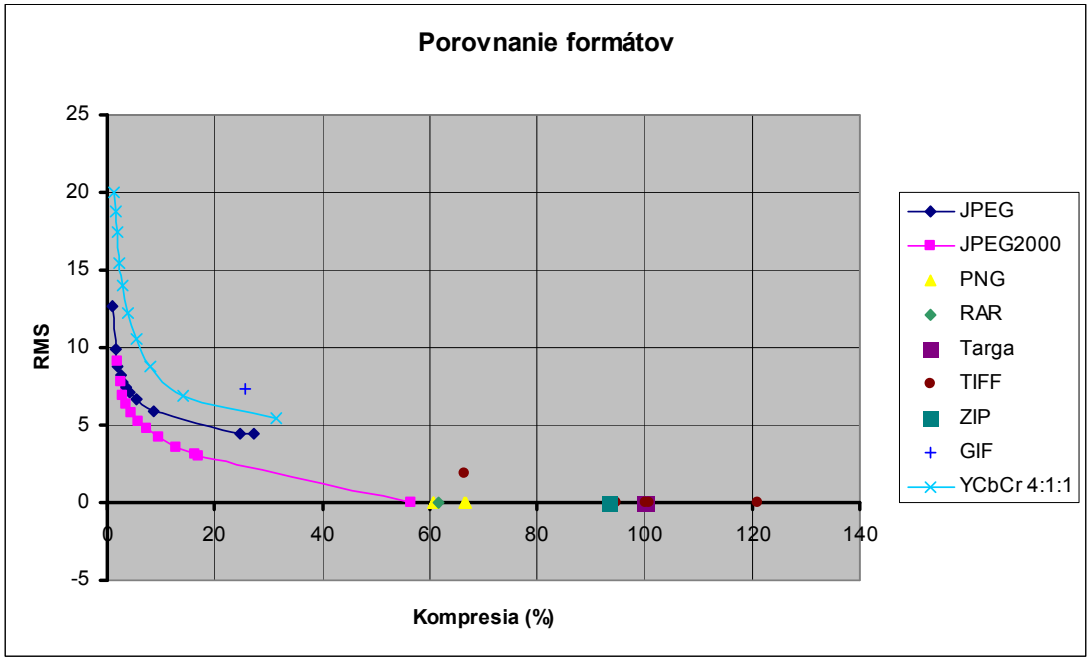


Graf 8: Porovnanie farebných modelov RGB a YCbCr.

Model RGB v móde 4:4:4 dosahuje vyššej vernosti pri nízkych kompresných pomeroch. Model YCbCr v móde 4:4:4 dosahuje podobnej vernosti pri značnom zlepšení kompresie, a v móde 4:1:1 dosahuje najlepšej kompresie pri malej strate vernosti. Preto sa model YCbCr v móde 4:1:1 zdá byť najvhodnejšou alternatívou. Testované na obraze Lena24bit.

5.2 Porovnanie so známymi formátmi

Porovnáme známe formáty a metódy kompresie často používané v praxi. Zo všetkých formátov sme vybrali zástupcov obrazových kompresorov so stratovou aj bezstratovou kompresiou (JPEG, JPEG2000, PNG, Targa, TIFF, GIF), ako aj zástupcov klasických dátových bezstratových kompresorov (RAR, ZIP). Vstupom bol obraz Lena24bit. Merali sme dosiahnutú kompresiu a kvalitu pre rôzne nastavenia kodérov (väčšinou to bol koeficient kvality alebo úroveň kompresie).



Graf 9: Porovnanie formátov.

Z grafu 9 je zrejmé, že najlepšiu kompresiu pri zachovaní kvality poskytuje formát JPEG2000 (založený na waveletovej transformácii), ktorý poskytuje aj možnosť bezstratovej kompresie. Ďalším v poradí je starší formát JPEG (založený na kosínusovej transformácii), ktorý je predchodcom formátu JPEG2000. Z bezstratových formátov najlepšie výsledky dosahuje JPEG2000, o trochu horšie výsledky poskytuje PNG a RAR. Niektoré formáty pri istých nastaveniach objem dát dokonca zväčšili (TIFF). Fraktálna metóda v tejto implementácii dosahuje o niečo nižšiu kvalitu obrazu pri rovnakých stupňoch kompresie v porovnaní s formátom JPEG, ku ktorému je najbližšie.

5.3 Záver a pokračovanie práce

Z prevedených testov vidíme, že model RGB nie je vhodný pre ukladanie fraktálne transformovaných dát. Vhodnejším sa ukázal model YCbCr pretože je bližší ľudskému vnímaniu farby (vyššia citlivosť na zmenu jasu ako na zmenu farebného odtieňu alebo sýtosti). Ukázalo sa, že v móde YCbCr 4:1:1 je fraktálna kompresia istým „konkurentom“ kompresii JPEG. Pri hlbšom výskume by bolo možné tieto výsledky ešte vylepšiť. V práci je možné pokračovať rôznymi smermi. Môžme sa zamerať na optimalizáciu kvality obrazu, vyskúšať iné metódy predspracovania, použiť inú doménu (nie priestorovú ale napr. spolupracovať s kosínusovou alebo waveletovou transformáciou), vyskúšať iné fraktálne schémy (trojuholníkovú). Môžme sa tiež zamerať na optimalizáciu ukladania výstupných transformácií, použiť iné metódy kódovania (aritmetické, LZW, RLE, CABAC, CAVLC, MTF, BWT). A nakoniec sa môžme zamerať aj na optimalizáciu rýchlosti kódovania, vylepšiť algoritmus prehľadávania, použiť klasifikáciu domén a blokov, využiť možnosti CPU (SIMD, MMX, 3DNOW, SSE, SSE2, SSE3, VMX, AltiVec,...), využiť paralelizáciu (viac jadrové systémy, viac procesorové systémy, distribuované systémy, klastre, ...).

Pramene

[1] PEITGEN, JÜRGENS, SAUPE. 1992. *Chaos and Fractals, New Frontiers of Science*. New York : Springer-Verlag Inc., 1998. ISBN 3-540-97903-4.

[2] FISHER, Y. 1995. *Fractal Image Compression, Theory and Application*. New York : Springer-Verlag Inc., 1995. ISBN 3-540-94211-4.

[3] POLEC, J. a kolektív. 2000. *Vybrané metódy kompresie dát, kódovanie obrazov*. Bratislava : Fakulta Matematiky, Fyziky a Informatiky Univerzity Komenského, 2000. ISBN 80-223-1392-0.

[4] GLASSNER, A. *Principles of Digital Image Synthesis*. Kapitola 1, The Human Visual System And Color.

[5] MEDVEĎ, M. *Prednášky z Matematickej analýzy III* na FMFI UK.

[6] *Waterloo Fractal Compression Project*. Dostupné na internete: <http://links.uwaterloo.ca/fractals.home.html>.

[7] *Microsoft Developer Network*. Dostupné na internete: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directshow/htm/directshow.asp>.

[8] STANEK, M. Prílohy k prednáške *Kódovanie a kryptológia* na FMFI UK. Dostupné na internete: <http://www.dcs.fmph.uniba.sk/~stanek/pedag.html>.

[9] ŠIKUDOVÁ, E. Prednáška *Kódovanie a spracovanie obrazu* na FMFI UK. Dostupné na internete: <http://www.sccg.sk/~sikudova/courses.html>.

[10] PETZOLD, Ch. 1999. *Programování ve Windows*. Praha : Computer Press, 1999. ISBN 80-7226-206-8.

[11] POLEC, J. , PAVLOVIČOVÁ, J. , ORAVEC, M. , VARGIC, R. , KARLUBÍKOVÁ, T. 2001. *Medzinárodné štandardy pre kódovanie obrazu I.,Princípy kódovania obrazu*. Bratislava : FEI STU, 2001. ISBN 80-227-1629-4.

[12] VARGIC, R. 2003. *Wavelety a banky filtrov*. Bratislava : FEI STU, 2003. Dostupné na internete: <http://www.ktl.elf.stuba.sk/~vargic/wabf/skripta>.

[13] POLEC, J. , PAVLOVIČOVÁ, J. , KARLUBÍKOVÁ, T. 2001. *Medzinárodné štandardy pre kompresiu obrazu II, H.261,MPEG-1,MPEG-2,H.263,MPEG-4*. Bratislava : FEI STU, 2001. ISBN 80-227-1784-3.

[14] MURRAY, J.D. , VANRYPER, W. 1997. *Encyklopedie Grafických Formátů, Druhé vydání*. Praha : Computer Press, 1997. ISBN 80-7226-033-2.

[15] SAUPE, D. 1995. *Accelerating Fractal Image Compression by Multi-Dimensional Nearest Neigbour Search*. Proceedings DCC'95 (IEEE Data Compression Conference).

[16] GIANG, N.K. , SAUPE, D. *Adaptive Post-Processing For Fractal Image Compression*.

[17] HARTENSTEIN, H. , RUHL, M. , SAUPE, D. 1999. *Region-Based Fractal Image Compression*. Submission to IEEE Transaction on Image Processing, March 1999.

[18] RUHL M. , HARTENSTEIN, H. , SAUPE, D. 1997. *Adaptive Partitionings for Fractal Image Compression*. Proceedings ICIP-97 (IEEE International Conference on Image Processing), Santa Barbara, Oct. 1997.

[19] KIM, Chang-Su, KIM, Rin-Chul, LEE, Sang-Uk. 1995. *Novel Fractal Image Compression Method With Non-Iterative Decoder*. Proceedings ICIP-95 (IEEE International Conference on Image Processing).

[20] SAUPE, D. , RUHL, M. 1996. *Evolutionary Fractal Image Compression*. Proceedings ICIP-96 (IEEE International Conference on Image Processing), Lausanne, Sept. 1996.

[21] SAUPE, D. , HAMZAOUI, R. 1998. *A Bibliography on Fractal Image Compression*. Nov. 1998.

[22] ZHANG, Ying, PO, Lai-Man. 1995. *Fractal Color Image Compression Using Vector Distorsion Measure*. Proceedings ICIP-95 (IEEE International Conference on Image Processing).

[23] LEVI, P. , MACHE, N. , HARRER, T. *Applications of fractal image encoding*.

[24] HAMZAOUI, R. , SAUPE, D. 1998. *Rate-Distortion Based Fractal Image Compression*.

[25] RABIEE, H.R. , KASHYAP, R.L. , RADHA, H. 1995. *Multiresolution Image Compression With BSP Trees and Multilevel BTC*. IEEE ICIP 1995, Washington D.C., Oct. 1995.

[26] HART, J.C. 1995. *Fractal Image Compression and the Inverse Problem of Recurrent Iterated Function Systems*.

[27] Centrum voor Wiskunde en Informatica. Dostupné na internete: <http://www.cwi.nl/archive/projects/fractals/index.html>.

[28] BAHARAV, Z. , KRUPNIK, H. , MALAH, D. , KARNIN, E. *A Multi-Resolution Framework for Fractal Image Representation and its applications*.

[29] FISHER, Y. , ROGOVIN, D. , SHEN, T.P. *A Comparison of Fractal Methods with DCT and Wavelets*.