Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Komprese barevných digitálních obrazů s využitím triangulace

Plzeň, 2008 Radek Sýkora

Originál zadání práce, (ten s červeným kulatým razítkem) v jednom výtisku práce, kopie zadání ve druhém.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 16.5.2008 …….

Radek Sýkora

Poděkování

Touto formou bych chtěl vyjádřit poděkování těm, kteří mi při psaní bakalářské práce pomáhali. Děkuji panu Ing. Josefu Kohoutovi, Ph.D, vedoucímu Projektu 5 (na PRJ5 tato práce navazuje). Dále bych rád poděkoval vedoucí této bakalářské práce, paní Doc. Dr. Ing. Ivaně Kolingerové. Oběma děkuji za velmi cenné rady, odborné názory a komentáře, které pro mě byly velmi užitečné a také za jejich kritické přečtení a komentování textu. Dále děkuji svým rodičům nejen za trpělivost během trávení mého času u bakalářské práce. Děkuji také Haně Ševčíkové, která mi zapůjčila počítač k ukládání velkého množství dat.

Abstrakt

The title of this bachelor thesis is “Compression of Colour Digital Images with Use of Triangulation”. The theoretical part describes colours, their perception, and description of individual colour systems. In addition, it briefly summarises placement of colours and raster data in a computer, and methods of raster data compression. The manner of objective measurement of visual quality of image data compression, so-called PSNR, is described in an abbreviated form. The conclusion of the theoretical part is intended for data presentation by means of triangulations, co-triangulations, and their subsequent coding and compression. The RAW, VXPATH, and FVXPATH coding methods are described. The practical part describes solutions, transfer between individual colour systems, and data processing. The practical part is ended with a chapter containing the analysis of the results achieved.

Obsah

1. Úvod

V dnešní době se s digitálními obrazy setkáváme téměř na každém kroku. Došlo k masivnímu rozvoji digitalizace fotoaparátů a videokamer, které jsou součástí řady domácností. Spolu s tímto rozvojem je řešena problematika ukládání velkého množství dat z digitálních zařízení do počítače pomocí různých kompresních metod. Komprese barevných digitálních obrazů s využitím triangulace (kotriangulace) spolu s použitím různých barevných systémů je tématem této bakalářské práce. Cílem je zhodnocení vlivu použitých barevných systémů na dosažený kompresní poměr triangulací (kotriangulací) a na kvalitu jednotlivých obrázků, kde jsou jednotlivé barevné komponenty zpracovávány odděleně jako tři nezávislé šedotónové obrazy a výsledkem procesu jsou tři nezávislé triangulace nebo tři nezávislé triangulace následně kombinované do jedné kotriangulace.

Bakalářská práce by se dala rozdělit na čtyři celky a to:

- úvod
- teoretická část
- realizační část
- závěr

Teoretická část se skládá celkem ze čtyř kapitol: *Světlo, barva a barevné systémy*, *Reprezentace rastrových obrazů*, *Triangulace a kotriangulace* a *Komprese a kódování triangulací*. První část je věnována barvám, jejich reprezentaci a jednotlivým barevným systémům, které jsou následně testovány. Další kapitola se zabývá reprezentací digitálních obrazů, jednotlivými způsoby uložení rastrových dat (pomocí ztrátové nebo bezeztrátové komprese), dále se kapitola věnuje formátu PNG a způsobu objektivního měření vizuální kvality komprese (PSNR). Následuje kapitola popisující způsob reprezentace obrazových dat v podobě triangulací a kotriangulací. Poslední kapitola teoretické části je věnována kompresím a kódováním těchto triangulací (kotriangulací).

Realizační část se skládá ze dvou kapitol: *Implementace* a *Analýza výsledků*. Implementace se zabývá popisem řešení jednotlivých částí programu, zatímco analýza výsledků je věnována zhodnocení dosažených výsledků během celého testování z pohledu triangulací, kotriangulací a následnému porovnání těchto dvou přístupů.

Poslední kapitola (*Závěr*) uzavírá celou práci shrnutím dosažených výsledků.

2. Světlo, barva a barevné systémy

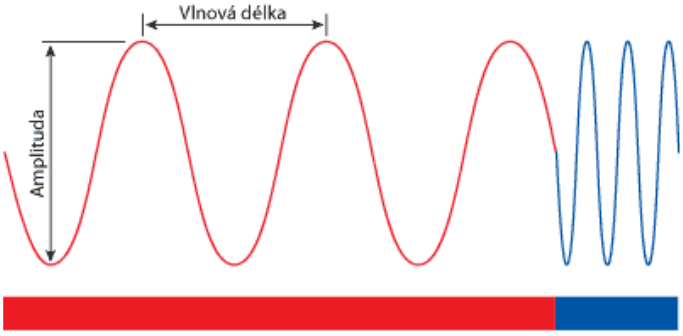
2.1 Světlo

Podle definice [10][1] je světlo viditelná část elektromagnetického záření o určité vlnové délce. Člověk je však schopen registrovat jen velmi malou část na zemi existujícího záření a ještě menší část záření existujícího ve vesmíru. Rozlišujeme zdroje světla:

- sálání tepla (sluneční světlo, záření žárovky)
- záření plazmatu (oheň, oblouková lampa)
- monochromatické záření (laser, světlo LED diody, plynové výbojky)
- luminiscence (luminiscenční diody, stínítka obrazovek apod.)

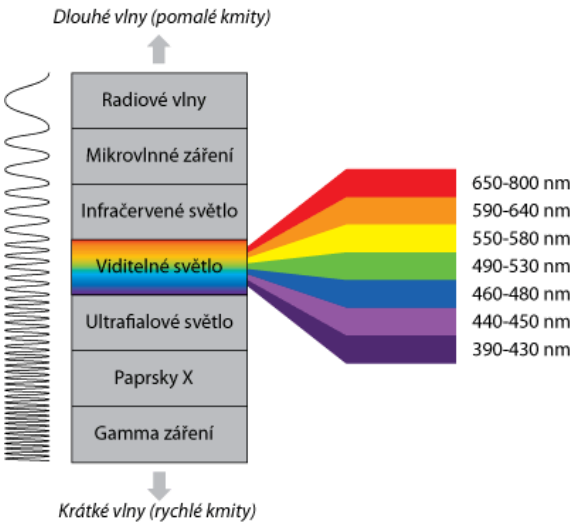
Základní charakteristika světla (Obr. 2.1):

- vlnová délka (rychlost či frekvence kmitání)
- intenzita (jas světla, amplituda vlny)
- polarizace (směr kmitání)



Obr. 2.1 Základní charakteristika světla [15].

Části viditelného světla vnímáme jako jednotlivé barvy (Obr. 2.2). Odstín barvy záleží na vlnové délce světla, která se udává v nanometrech (nm). Nejkratší vlnové délky (okolo 400 nm) odpovídají barvě modré či fialové, naopak nejdelší vlnové délky (okolo 700 nm) určují barvu červenou. Vlnové délky zelené se pohybují kolem 500 nm.



Obr. 2.2 Obrázek znázorňuje spektrum viditelného světla (monochromatické záření) rozdělené podle barev a odpovídající vlnové délky. Za hranicemi červené, resp. fialové barvy, již lidské oko barvu nevnímá - zde leží infračervené a ultrafialové světlo [15].

[1] podkladem pro většinu citací v kapitole č.2 je [10], pokud není uveden jiný zdroj

2.2 Barva

Fyzikálně je tedy barva vjem, který vytváří část spektra viditelného záření dopadajícího na sítnici lidského oka. Barevné vidění zprostředkovávají receptory oka [4] zvané čípky, které jsou trojího druhu - citlivé na tři základní barvy (červenou, zelenou a modrou). Vedle tří druhů barvocitlivých čípků jsou na sítnici ještě tyčinky, které jsou citlivé pouze v zeleno-modré oblasti (neslouží k registraci barvy). Uplatní se zejména při nočním vidění, kdy barvocitlivé čípky díky své nízké citlivosti ztrácejí schopnost vidět. To je důvod, proč v noci a ve tmě vidíme jen černobíle (nejsme schopní rozlišit barvy) a proč nám noc připadá celkově lehce modrá.

Každá jedna konkrétní vlnová délka světla je okem vnímána jako jedna konkrétní barva. Barvy, které je takto možné vytvořit, jsou tzv. spektrální barvy. Mícháním různých vlnových délek vzniká řada barev, které nikdy nemohou být vytvořeny jednou vlnovou délkou. Ty se nazývají nespektrální, neboť nejsou obsaženy v čistém spektru světla. Typickými nespektrálními barvami jsou např. růžové či purpurové odstíny, které jsou směsí červené a fialové z opačných konců barevné stupnice. Pokud je směs vyvážená, tj. obsahuje shodné množství základních barev, dostaneme odstín šedé barvy (krajními odstíny šedé jsou černá a bílá).

Barva, přesněji řečeno to, co člověk jako barvu vnímá, je závislá na mnoha okolních podmínkách. Mezi hlavní patří spektrální složení dopadajícího světla, směr jeho dopadu, vlastnosti povrchu, celková barevnost scény, směr pohledu pozorovatele a vlastnosti pozorovatele (např. kvalita zraku, přizpůsobení okolnímu světlu, věk…). Objektivní posuzování barev dále komplikuje skutečnost, že se lidské smysly nechávají oklamat očekáváním a pamětí. Při běžném pohledu člověk nevnímá barevné změny známých předmětů v závislosti na změnách okolního osvětlení. Člověk si automaticky dorovnává jak intenzitu osvětlení, tak i hodnotu bílé barvy. Vnímání barvy ovlivňují barvy v okolí pozorovaného místa. Typickým příkladem jsou optické klamy (Obr. 2.3).

Obr. 2.3 Jednoduchý optický klam ukazuje vliv okolní plochy. Střední pruh je v celé své délce stejně šedý, přesto může být vnímán jako přechod od světle šedé (vlevo) po tmavší šedou (vpravo) [16].

Tyto klamy dobře demonstrují jednotlivé funkce dvojice oko-mozek. Pokud tedy chceme barvy (v digitální podobě) sdílet s ostatními, potřebujeme objektivní a přesný návod, jak je popisovat, měřit a kontrolovat (přesné matematické metody pro popisování barev). Poptávka po standardizaci modelů barev vedla v roce 1931 k založení komise CIE (Commission Internationale de l'Eclairage), která je zodpovědná za stanovení a udržování mezinárodních standardů. Výstupy práce CIE jsou, kromě jiného, definice barevných prostorů, normy definující metodologii měření, vlastnosti pozorovatele a vlastnosti osvětlení.

2.3 Reprezentace barev

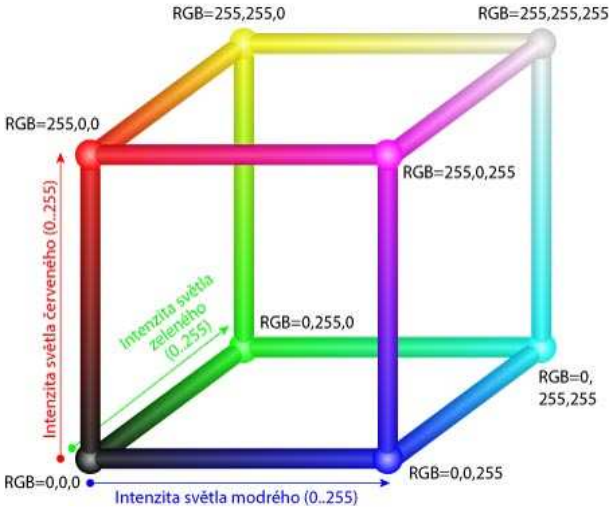
Nejčastěji jsou barvy reprezentovány vektorem o třech nebo čtyřech složkách [12]. Významy jednotlivých složek vektoru závisí na tom, s jakým barevným prostorem pracujeme (viz následující kapitoly). Výsledné barvy v digitálním obrazu jsou zpravidla tvořeny kombinací několika základních barev z barevného spektra.

2.4 Barevné systémy

Mezi nejznámější barevné systémy patří model RGB. Je to proto, že v tomto modelu pracují digitální fotoaparáty a většina fotografií je v tomto modelu uložena. I když populární JPEG používá pro svojí vnitřní potřebu trochu jiný barevný systém (YCbCr), navenek se pro běžného uživatele tváří také jako RGB. Asi druhý nejznámější model je model CMYK (nebude zde podrobně popisován) určený zejména pro tisk. Model HSV je obvykle používán v grafických aplikacích stejně tak jako L*u*v.

2.4.1 Barevný systém RGB

Barevný systém RGB je vhodný pro zobrazení barev v počítači i na jiných zobrazovacích zařízeních. Nejčastěji používané je skládání tří složek - červené (R, red), zelené (G, green) a modré (B, blue). Barva je tedy v tomto případě vyjádřena trojsložkovým vektorem (R, G, B), kde hodnoty jednotlivých složek jsou nejčastěji z intervalu (0, 255). Hodnota nula znamená minimum (příslušná složka není v barvě obsažena), hodnota 255 maximum (maximální možný podíl barevné složky ve výsledné barvě). Černá barva je vyjádřena vektorem (0, 0, 0), bílá barva (255, 255, 255). RGB model lze zobrazit jako krychli (Obr. 2.4). Barva vyjádřená tímto způsobem určuje také intenzitu světla.

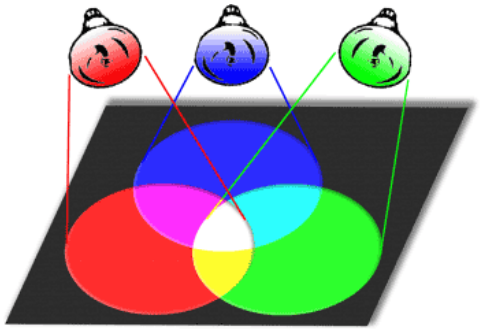


Obr. 2.4 RGB model znázorněný pomocí krychle, jednotlivé osy odpovídají modrému, červenému a zelenému světlu., na úhlopříčce krychle je stav, kdy všechna tři světla svítí na maximum, tedy vytvoří bílou barvu (255, 255, 255) [17].

RGB model je aditivní[2] model, tedy založený na přidávání RGB světel na tmavou nesvítící (Obr. 2.5) podložku (typické pro klasický monitor či televizi). Přidáním všech světel naplno se vytvoří bílá. Bohužel sám model RGB nemá žádnou přesnou specifikaci svých základních barev (červené, zelené a modré), a proto vzniklo více RGB

[2] aditivní míchání barev je takový způsob, kdy se jednotlivé složky barev sčítají a vytváří světlo větší intenzity, výsledná intenzita se rovná součtu intenzit jednotlivých složek

modelů. Nejznámější a nejrozšířenější je zřejmě varianta sRGB, která je standardem Windows. V tomto standardu jsou přesně definovány jak základní barvy RGB, tak i bílý bod. Barevný model sRGB je praktický zejména proto, že odpovídá reálným možnostem zobrazení většiny monitorů.



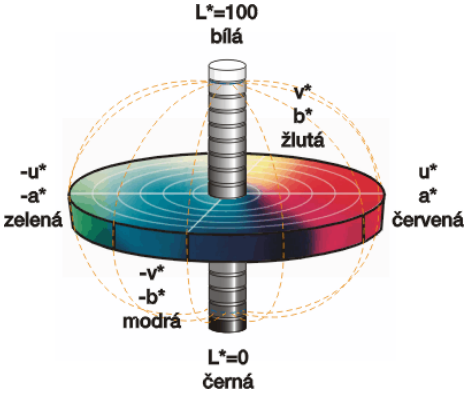
Obr. 2.5 Aditivní míchání barev (přidávání světel na tmavou podložku) [17].

2.4.2 Barevný systém YCbCr

Dalším barevným prostorem, který zde bude zmíněn, je barevný prostor YCbCr. Nejčastěji se používá u videa a digitální fotografie. Je to barevný prostor, původně vytvořený pro televizní normu SECAM. CRT video je zobrazováno pomocí červených, zelených a modrých potenciálových signálů, ale tento model signálů není účinný pro zobrazení a zprostředkování (mnoho nadbytečnosti). YCbCr se vyznačuje oddělením jasové složky Y (může být ukládána s vysokým rozlišením, nebo vysílána s velkou šířkou pásma) od složek Cb, Cr (mohou být vysílány s redukovanou šířkou pásma, s dílčími vzorky, komprimované, nebo jinak odděleně zpracovávané pro zlepšenou účinnost systému) definujících barevný odstín, čehož se používá v kompresním algoritmu JPEG.

*2.4.3 Barevný systém L*u*v*

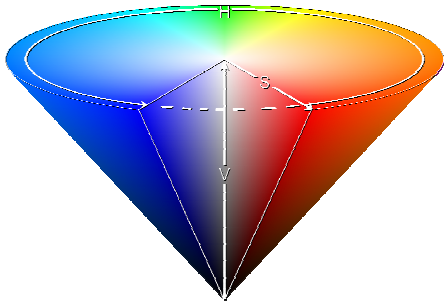
Pro názorné a matematicky snadné určování barev byl navržen barevný prostor CIE L*u*v, který si získal výsadní postavení při popisu barev nezávislých na zařízení. Jeho konstrukce je založena na faktu, že barva nemůže být zároveň zelená a červená, stejně tak modrá a žlutá. Proto lze zavést hodnoty, popisující polohu odstínu mezi zelenou a červenou, respektive mezi modrou a žlutou. Máme-li barvu definovanou v CIE L*u*v, pak L definuje jas, *u udává polohu mezi primárními barvami R/G a *v udává polohu mezi primárními barvami Y/B. Ve středu kruhového diagramu (Obr. 2.6) je neutrální oblast (středem této oblasti procházejí neutrální barvy černá, stupně šedé a bílá). Polohu barvy v souřadnicích *u a *v vynášíme do kruhového diagramu, skutečnou polohu barvy obdržíme po zahrnutí hodnoty L a umístěním barvy do prostoru.



Obr. 2.6 Barevný systém L*u*v [18].

2.4.4 Barevný systém HSV

Pro některé situace, zejména pro některé případy práce s obrázky, je výhodný model HSV. Jeho přínos je v tom, že odpovídá lidskému vnímání barev. Zatímco RGB je model založený na míchání barev, HSV model definuje barvy pro člověka přirozeným způsobem. Je tedy velmi intuitivní a pro některé případy velmi názorný. Barevný model HSV používá (podobně jako např. model RGB) tři základní složky pro popis barvy, dává jim ale jiný význam [6] (Obr. 2.7).



Obr. 2.7 Kuželovitá reprezentace HSV modelu [10].

Odstín barvy (H, Hue) je vlastnost, s jejíž pomocí rozlišujeme jednu barvu od druhé (tedy např. červenou, zelenou, modrou). Pro popis barvy se používá úhel na barevném kole (0-360°). Dohodou se za úhel 0° (360°) považuje červená, 120° odpovídá zelené a 240° modré barvě.

Sytost barvy (S, Saturation) popisuje, zda-li je barva bez přimíchání bílé (šedé). Čím více má v sobě barva bílé (šedé), tím více její sytost klesá. Udává se v procentech přičemž sytost 100 % znamená čistou barvu, sytost 50 % znamená poloviční příměs bílé (šedé) a sytost 0 % znamená odstín šedé (od bílé po černou) tedy již zcela bez barvy. Na barevném kole vzrůstá sytost od středu k okrajům (např. červená s 50% sytostí bude růžová).

Jas (V, Value) popisuje vlastnost barvy podle měřítka "tmavá - světlá". Nabývá hodnot od 0 (pro všechny barvy čistá černá) do 100% (pro všechny barvy čistá bílá). Černá a bílá, spolu s odstíny šedé mezi nimi, se označují jako neutrální barvy (nemají odstín).

3. Reprezentace rastrových obrazů

3.1 Způsoby uložení barev v počítači

Ukládání barev v podobě rastrových obrázků souvisí s maximálním možným (teoretickým) počtem barev, které se mohou v obrázku vyskytovat. Tento počet barev je udáván v tzv. barevné hloubce[3]. Mezi používané barevné hloubky patří [12]:

- 1bit: 2^1 = 2 barvy
- 4bity: 2^4 = 16 barev
- 8bitů: 2^8 = 256 barev
- 16bitů: 2^{16} = 65 536 barev, High Color
- 24bitů: 2^{24} = 16 777 216 barev, True Color, RGB
- 32bitů: 2^{32} = 4 294 967 296 barev, True Color, RGBA

Barvy lze ukládat přímo (svou hodnotou), nebo odkazem (indexem) do palety[4] barev. Ukládání barev hodnotou je používáno u True Color obrázků, ukládání odkazem pro ostatní barevné hloubky. Barvy jsou v paletě uloženy jako 24 nebo 32 bitové hodnoty.

3.2 Způsoby uložení rastrových dat

Obrázky jsou ukládány v podobě rastrových souborů [11]. Obrazová data lze mimo jiné do souboru zapsat v nekomprimované a nebo komprimované podobě. V případě nekomprimovaných dat jsou ukládány přímo hodnoty jednotlivých obrazových bodů (pixelů[5]), v případě komprimovaných dat jsou obrazová data před zápisem do souboru zpracována některým z kompresních algoritmů. Existují dva základní typy kompresních algoritmů [10] - bezeztrátové a ztrátové. Bezeztrátové jsou takové kompresní algoritmy, kdy jsou původní obrazová data a data po kompresi a následné dekompresi totožná. Naopak ztrátové jsou takové kompresní algoritmy, kdy původní obrazová data a data po kompresi a následné dekompresi nejsou totožná, ale jsou do jisté míry podobná.

3.2.1 Bezeztrátové metody komprese

RLE - jedna z nejjednodušších komprimačních metod [10] pro kompresi nejen obrazových dat. Tato metoda zredukuje jakýkoliv typ sekvence opakujících se obrazových bodů stejné barvy hned, jak jejich počet dosáhne předepsaného počtu výskytů. Do souboru se pak zapisuje hodnota příslušné barvy a počet opakování. Tato metoda je symetrická[6] a vyznačuje se velkou rychlostí kódování za cenu poměrně nízkého kompresního poměru. Kompresní poměr však záleží na zpracovávaných datech a může být v některých případech naopak velmi vysoký. Použití: např. grafické formáty PCX, PSD, BMP…

LZ77 - princip tohoto kompresního algoritmu [10] spočívá v náhradě původních sekvencí obrazových dat binárními kódy. Tyto binární kódy se během kódování průběžně doplňují do tabulek (slovníků - slovníkové kódování). Kódovací tabulky se vždy po svém zaplnění plní znovu od začátku. To má tu výhodu, že lze provádět

[3] počet bitů sloužících k uložení barevné informace
[4] seznam indexů barev, každý pixel obrazu uchovává číslo, které odkazuje na odpovídající položku palety
[5] obrazový prvek, miniaturní světlocitlivá buňka
[6] komprese i dekomprese trvá zhruba stejně tak dlouho

dekomprimaci od míst, kde se začíná s novými tabulkami, pokud jsou tato místa někde zapamatována. Kódovací tabulky se nemusí zapisovat do souboru, počáteční stav tabulek je vždy pro určitou délku kódování stejný a dále se plní při rozbalování opačným způsobem než při komprimaci. Použití: např. grafický formát PNG (podrobnější popis viz podkapitola 3.3 Formát PNG).

LZW - je to modifikovaný LZ77 algoritmus, používá [10] se v kompresním programu ZIP. Použití: např. grafické formáty GIF, TIFF, PostScript…

Huffmanovo kódování - základní princip [10] spočívá v nahrazení různě dlouhých sekvencí bitů kratšími bitovými, prefixovými[7] kódy, při jejichž tvorbě se využívá znalost pravděpodobnosti výskytu jednotlivých kódovaných znaků (častěji se vyskytujícím znakům přiřazuje kratší kódy a naopak). Výhody této metody jsou velmi rychlá komprese i dekomprese a nepříliš velké nároky na paměť. Nevýhodou je nutnost uložení binárního stromu a slabší kompresní poměr (algoritmus si nevšímá opakování řetězců). Použití: např. grafické formáty CIT, TIFF…

3.2.2 Ztrátové metody komprese

JPEG - kompresní metoda [5] využívající rozdílné citlivosti lidského oka, které je citlivější vůči malým změnám jasu než vůči jemným změnám barevných odstínů. Vstupní barvy jsou nejprve převedeny do barevného prostoru YCbCr a následně je jasová složka Y zpracována odděleně od zbylých dvou složek. V dalším kroku se sníží objem dat zprůměrováním barevných složek z několika sousedních bodů. Poté je obraz rozdělen na čtverce 8x8 pixelů, pro něž jsou spočteny koeficienty diskrétní kosinové transformace. Tyto čtverce jsou v komprimovaném souboru uloženy v pořadí, odpovídajícím rozkladu obrázku směrem shora dolů.

JPEG2000 – upravený [5] formát JPEG vyskytující se i v bezeztrátové variantě. Využívá předzpracování obrazu vlnkovými (wavelet) transformacemi, nevytváří čtverce v obraze. Zatím se příliš nerozšířil.

3.3 Formát PNG

Všechny výstupy z programu (který je mimo jiné náplní této BP a který bude podrobně popsán v následujících kapitolách) v podobě rastrového obrazu jsou ukládány v binárním souborovém formátu PNG [10] (Portable Network Graphics). Tento formát byl zvolen především proto, že data ukládá pomocí modifikované bezeztrátové komprese LZ77 a zároveň dosahuje výborných kompresních poměrů pro nejrůznější True Color obrázky. Nabízí barevnou hloubku 24bitů a 8bitový alfa kanál[8]. PNG se často používá na internetu, stejně jako formát GIF a JPEG. Celý binární soubor s uloženým obrázkem se skládá z hlavičky a z libovolného množství tzv. chunků[9], což jsou pojmenované bloky, z nichž každý je opatřen svou délkou, typem a kontrolním součtem CRC[10].

[7] je to takový kód, kde žádný symbol kódové abecedy není předponou jiného (delšího) symbolu abecedy
[8] obrázek může být v různých částech různě průhledný
[9] shluky
[10] speciální funkce používaná k detekci chyb během přenosu či ukládání dat

Hlavička souboru - má délku 8 bytů. Tyto byty mají vždy stejné hodnoty, jejich hexadecimální vyjádření vypadá následovně: 89 50 4E 47 0D 0A 1A 0A. Podrobnější popis znázorňuje Tab. 3.1.

BYTE	VÝZNAM BYTU
89	Byte s nejvýše nastaveným bitem, detekce podpory 8bit. přenosu dat
50 4E 47	v ASCII řetězec „PNG“, slouží k identifikaci souborového formátu
0D 0A	v DOSu označující konec řádku, detekce náhrady za jinou sekvenci
1A	Byte zastavující výpis souboru v DOSu
0A	v UNIXu detekce konce přenosu (LF)

Tab. 3.1 Hlavička souboru, význam jednotlivých bytů [10].

Shluky - po hlavičce přichází série shluků, z nichž každý zprostředkovává jistou informaci o obrazu. Shluky deklarují sebe jako rozhodující nebo pomocné. Pokud se program setká se shlukem, kterému nerozumí, může ho bezpečně ignorovat. Struktura tohoto shluku je založena tak, aby dovolovala slučitelnost PNG formátů se staršími verzemi. Každý shluk má hlavičku specifické velikosti a typu. Pak ihned následují aktuální data, posléze probíhá kontrolní součet dat. Shluky jsou dány čtyřmi znaky v ASCII s vlastností case-sensitiv[11]. Rozdílnost znaků (pátý bit je číselná hodnota) poskytuje možnost dešifrovat informaci o povaze shluku v případě jeho nerozeznatelnosti.

3.4 Objektivní měření vizuální kvality komprese

Ztrátově komprimovaný obrázek se vždy liší od originálu. Jedna z mnoha používaných metod měření vizuální kvality je střední kvadratická chyba (MSE - mean square error), což je v podstatě zkreslení komprimovaného obrázku vůči originálnímu obrázku. Tato chyba je definována následujícími vztahy [10]:

$$a) \quad MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left\| I(i,j) - K(i,j) \right\|^2$$

$$b) \quad MSE = \frac{1}{3 \cdot m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \sum_{k=0}^{2} \left\| I_k(i,j) - K_k(i,j) \right\|^2$$

kde *m* a *n* je výška a šířka (rozlišení) porovnávaných obrázků, *K* je komprimovaný obrázek, *I* originální obrázek a *(i, j)* jsou souřadnice aktuálního pixelu. Vzorec v případě a) je pro výpočet MSE šedotónových obrázků, případ b) se týká barevných obrázků.

Obvyklejší metoda měření vizuální kvality je poměr signálu vůči šumu (PSNR - peak signal to noise ratio), což je metoda, která lépe vypovídá o kvalitě obrazu, nicméně konkrétní hodnota této veličiny o ničem nevypovídá, míru kvality rekonstrukce získáme porovnáním výsledků dvou rozdílně zrekonstruovaných obrázků. Vztah pro výpočet PSNR vychází z MSE a je definován takto [10]:

[11] záleží na velikosti písmen

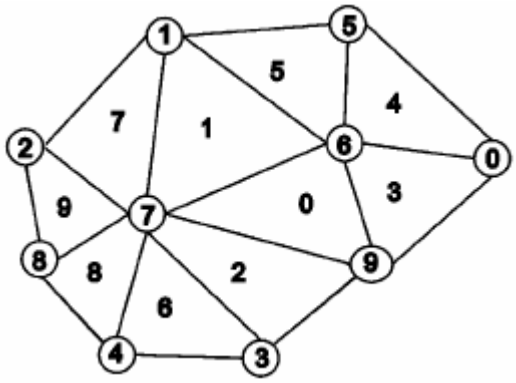
$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

kde MAX_I je maximální hodnota pixelu v obrázku. Pokud jsou pixely reprezentovány v podobě 8bit/pixel, potom MAX_I = 255. Všeobecně je tato hodnota definována jako $MAX_I = 2^b-1$. Vzorec pro výpočet PSNR je shodný jak pro šedotónové, tak pro barevné obrázky. Typické hodnoty PSNR se pohybují mezi 30-50dB, kde vyšší hodnota znamená lepší kvalitu komprimovaného obrázku.

4. Triangulace a kotriangulace

4.1 Tvorba trojúhelníkové sítě

Tvorba trojúhelníkové sítě je proces, kdy je vybrán určitý typ triangularizace a z množiny bodů je vytvořena trojúhelníková síť (TIN - Triangulated Irregular Network). TIN [7] model reprezentuje obrázek jako skupinu trojúhelníků, které mohou mít různou velikost a tvar. Tato reprezentace ukládá, mimo jiné, vztahy mezi jednotlivými trojúhelníky. Příklad použití TIN modelu znázorňuje následující obrázek (Obr. 4.1). Pro uložení této sítě v počítači je potřeba ukládat všechny informace o trojúhelnících (Tab. 4.1).



Obr. 4.1 Příklad sítě TIN [7]

Tab. 4.1a:

- obsahuje seznam vrcholů + Z = barva

Tab. 4.1b:

- obsahuje seznam sousedů jednotlivých trojúhelníků
- příznak -1 označuje hranu ležící na okraji sítě

Tab. 4.1c:

- obsahuje seznam jednotlivých vrcholů trojúhelníka

Index vrcholu	Souřadnice X	Y	Z
0	X0	Y0	Z0
1	X1	Y1	Z1
2	X2	Y2	Z2
3	X3	Y3	Z3
4	X4	Y4	Z4
5	X5	Y5	Z5
6	X6	Y6	Z6
7	X7	Y7	Z7
8	X8	Y8	Z8
9	X9	Y9	Z9

a) souřadnice vrcholů

Index trojúhelníku	Indexy okolních trojúhelníků		
0	3	1	2
1	0	5	7
2	-1	0	6
3	-1	4	0
4	3	-1	5
5	4	-1	1
6	2	8	-1
7	1	-1	9
8	6	9	-1
9	7	-1	8

b) seznam sousedů

Index trojúhelníku	Vrcholy 0	1	2
0	7	9	6
1	1	7	6
2	7	3	9
3	6	9	0
4	5	6	0
5	1	6	5
6	4	3	7
7	2	7	1
8	8	4	7
9	8	7	2

c) seznam vrcholů

Tab. 4.1 Datová struktura TIN modelu [7]

4.2 Triangulace

4.2.1 Definice triangulace

Triangulace je jednou ze základních oblastí ve výpočetní geometrii. Má velmi širokou oblast použití v počítačové grafice, vizualizaci vědeckých dat, modelování terénu, robotice, počítačovém vidění apod.

Pojem triangulace v našem případě znamená interpolaci založenou na jednotlivých bodech *P* komprimovaného obrázku.

Definice [9]: množina trojúhelníků $T = \{T_i\}$, $i = 1...n$ se nazývá triangulace oblasti Ω, pokud:

- libovolná dvojice trojúhelníků z T se vzájemně protíná nejvýše v jednom společném vrcholu nebo má nejvýše jednu společnou hranu
- sjednocení všech trojúhelníků vytvoří souvislou množinu v 2D

Z takto formulované definice triangulace vyplývá, že oblast Ω bude ohraničena souvislým polygonem[12], který nemusí být konvexní a může obsahovat díry. To lze odstranit pomocí zúžené definice triangulace:

Definice [9]: námi požadovaná triangulace je vytvořena, pokud množinu bodů $P = \{P_i\}$, $i = 0, ... n$ v rovině spojíme úsečkami, které se vzájemně neprotínají, takovým způsobem, že každá vnitřní oblast konvexní obálky těchto bodů je součástí trojúhelníka. Vlastnosti takto definované triangulace jsou:

- každý pod $P_i \in P$ je vrcholem minimálně jednoho trojúhelníka
- průnik vnitřních oblastí libovolných dvou trojúhelníků je prázdný
- sjednocení všech trojúhelníků tvoří konvexní obálku bodů množiny P

Takto definovaná triangulace však zdaleka není jednoznačná, a proto se při konstrukci triangulací volí další, tzv. optimalizační podmínky, které závisí na účelu, pro nějž je triangulace vytvářena. Na základě těchto kritérií se konstruují tzv. optimální triangulace z hlediska určitých optimalizačních podmínek. Optimální triangulace je pak taková, která je z hlediska těchto optimalizačních kritérií lepší, než všechny ostatní triangulace.

4.2.2 Delaunayova triangulace

Na základě optimalizačních kritérií se dají vytvářet různé druhy triangulací. Z nichž nejznámější je Delaunayova triangulace (DT - Delaunay triangulation).

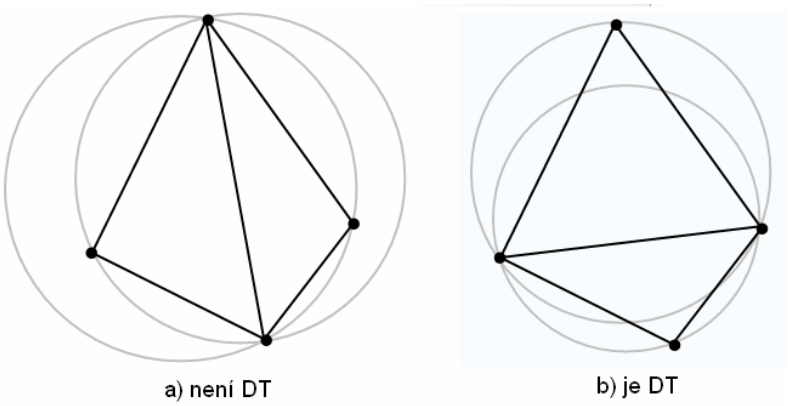
Definice [1]: triangulace je Delaunayovou triangulací, pokud platí, že uvnitř kružnice opsané libovolnému trojúhelníku neleží žádný další vrchol jiného trojúhelníka téže triangulace. Takto vytvořená triangulace má následující vlastnosti:

- je optimální z hlediska minimalizace poloměru kružnice opsané
- je optimální z hlediska maximalizace minimálního úhlu
- pokud žádné čtyři body neleží na kružnici, je jednoznačná

Pomocí této definice pracuje většina algoritmů, a to tak, že testují prázdnou opsanou kružnici. Obr. 4.2 tento test na prázdnou opsanou kružnici zachycuje. Kružnice opsaná

[12] mnohoúhelník

každému trojúhelníku v Delaunayově triangulaci neobsahuje žádný další bod z množiny bodů (Obr 4.2b). V určitém singulárním případě je možné, aby na kružnici leželo více bodů z Delaunayovy triangulace, tento případ nastává, pokud více než tři body v triangulaci leží na kružnici.



Obr. 4.2 Test na prázdnou kružnici [10]

Ke konstrukci Delaunayovy triangulace lze použít několik již existujících algoritmů. Jde o algoritmy typu [1]:

- inkrementální vkládání
- inkrementální konstrukce
- lokální zlepšování
- rozděl a panuj
- včlenění do vyšší dimenze
- nepřímá metoda přes tvorbu Voronoiova diagramu

4.2.3 Některé další triangulace

Datově závislá triangulace (DDT) - většina triangulací [1] je vytvořena pouze na základě znalosti umístění a vzdálenosti vstupních bodů v rovině, někdy je však výhodné při konstrukci triangulace použít i třetí souřadnici (výšku, barvu, atd.). Vstupem je již hotová síť. Obvykle se jako vstupní triangulace používá Delaunayova triangulace a to z těchto důvodů:

- dobré vlastnosti sítě
- časovou nenáročnost

Triangulace s minimální váhou (MWT) - minimalizuje [9] celkovou délku všech hran v triangulaci. Při konstrukci je potřeba sestrojit všechny možné triangulace a vybrat z nich tu s minimální celkovou délkou hran. Metoda má exponenciální složitost, je časově velmi náročná.

Hltavá (greedy, žravá) triangulace - tato [9] triangulace se skládá z nejkratších hran v triangulaci. Vytvoří se všechny možné hrany (i ty, které se navzájem budou křížit), seřadí se podle délky a poté se vkládají do výsledné triangulace od nejkratší tak dlouho, dokud není triangulace úplná.

4.3 Kotriangulace

Kotriangulace [8] několika funkcí jsou velmi důležité pro modelování komplexních přírodních jevů. Mohou být použity k reprezentaci jakéhokoliv počtu vlastností v jedné samostatné oblasti a jsou použitelné např. v následujících oborech:

- lékařství, např. modelování tkání
- reprezentace geologických dat
- reprezentace klimatických dat, např. tlak vzduchu a teplota vzduchu
- zobrazení a analýza N-rozměrných dat

Základní myšlenkou algoritmu vytvoření kotriangulace [8] je aproximace pomocí po částech lineárních funkcí vyšší dimenze. Vstupem je N datových množin, které reprezentují N různých D-rozměrných závislých funkcí. Výstupem algoritmu je D-rozměrný po částech lineární objekt v D + N rozměrném prostoru takový, že N ortogonálních projekcí tohoto objektu reprezentuje N vstupních datových množin s předepsanou tolerancí.

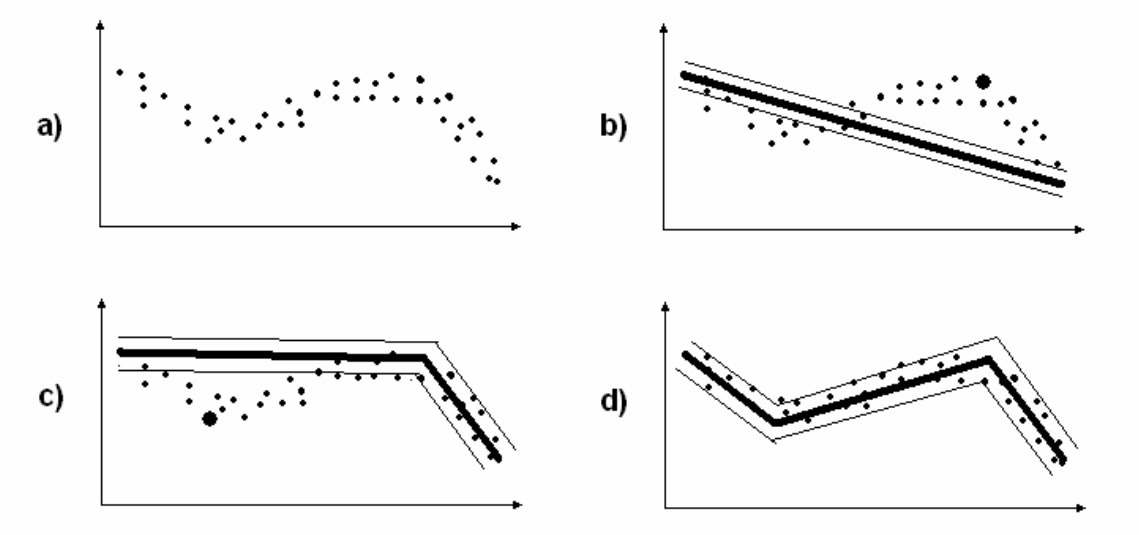
Stručný popis algoritmu pro aproximaci kotriangulace ve smyslu článku [8]:

1. Vstupem je N datových množin Si, kde i = 1, …, N, které reprezentují N různých D-rozměrných závislých funkcí.
2. Inicializace počáteční triangulace T_0 s vrcholy v R^D.
3. Vytvoření prvního odhadu aproximace dat S_1 přiřazením atributu S_1 (hodnota počátečního vzorku dat) všem vrcholům triangulace T_0.
4. Kvalita současné aproximace je zhodnocena stanovením maximální odchylky mezi aktuální aproximací a zbývajícími vzorky dat z S_1. Tato odchylka je měřena jako absolutní hodnota rozdílu v poslední souřadnici (souřadnice posledního vloženého vrcholu do triangulace) mezi interpolací atributů přiřazených vrcholům triangulace a zbývajícími vzorky dat z S_1. Nové vrcholy jsou do triangulace vkládány greedyho (žravým) algoritmem tak dlouho, dokud není splněno toleranční kritérium ε_1[13] pro všechny vzorky dat z S_1. Nové vrcholy vkládané do triangulace T_0 jsou vybírány přímo ze vzorků dat S_1, to znamená, že souřadnice těchto vrcholů jsou brány z těchto vzorků a atribut přiřazený s novým vrcholem je datová hodnota tohoto vzorku. Výslednou triangulaci nazýváme T_1. Je to aproximační triangulace pro skupinu dat S_1, která garantuje, že všechny vzorky dat z S_1 splňují toleranční kritérium ε_1 pro aproximaci první skupiny atributů přiřazených vrcholům T_1.

   Příklad: Aproximační algoritmus je ukázán na Obr. 4.3 ve 2D. S_1 obsahuje vzorky dat z křivky (2D), tj. tento objekt má jednu dimenzi. Vzorky s maximální odchylkou od současné aproximace jsou vkládány do triangulace T_0 jako nové vrcholy, dokud všechna data neleží uvnitř žádoucí tolerance (oblast označená tenkou linií). Atributy přiřazené těmto vrcholům jsou hodnotou těchto vzorků. Situace a) zobrazuje pouze body. Situace b) ukazuje prvotní aproximaci daných bodů a zároveň je tučně označen bod s maximální odchylkou od aktuální aproximace, který bude vkládán do triangulace T_0 v dalším kroku. Situace c) zobrazuje zařazení tohoto bodu do triangulace T_0 a zároveň je tučně označen

[13] maximální (nastavitelná) tolerovaná odchylka dat od aproximace, data spadající do tohoto tolerančního kritéria nejsou již pro následnou aproximaci uvažována (nejsou vkládána jako vrcholy do triangulace)

další bod s maximální odchylkou od aktuální aproximace, který bude vkládán do triangulace T_0 v následujícím kroku. Situace d) znázorňuje kompletní aproximaci původních dat. Tenká černá linie zobrazuje toleranční kritérium.



Obr. 4.3 Ukázka aproximačního algoritmu [8].

5. Triangulace T_1 je použita jako počáteční triangulace pro data S_2 (viz druhý bod aproximačního algoritmu) a aproximace začíná znovu (viz body 3-5 aproximačního algoritmu). Tento proces probíhá tak dlouho, dokud nejsou všechny skupiny dat N začleněny do aproximace.

   Příklad: Vrcholům T_1 jsou přiřazeny nové atributy k vytvoření prvního odhadu aproximace pro data S_2. Poté je triangulace upřesňována opakovaným vkládáním nových vzorků z S_2 do T_1 v podobě nových vrcholů. Opět jsou souřadnice vrcholu vybrány ze vzorků s maximální odchylkou od aktuální aproximace a přiřazený nový atribut je datová hodnota tohoto vzorku. Nové vrcholy jsou do triangulace T_1 vkládány greedyho (žravým) algoritmem tak dlouho, dokud není splněno toleranční kritérium ε_2 pro všechny vzorky dat z S_2. Výsledná triangulace se nazývá T_2. Triangulace T_2 je použita jako počáteční triangulace pro data S_3 (viz druhý bod aproximačního algoritmu) a aproximace začíná znovu (viz body 3-5 aproximačního algoritmu).

6. Výsledkem je triangulace dat, kde každý vrchol má N přiřazených atributů, které mohou být použity k aproximování všech dat v rámci tolerančního kritéria.

Algoritmus je jednoduchý a intuitivní. To má několik vlastností:

- skupiny dat jsou zpracovávány postupně, paměť je pro každou skupinu dat použita pouze jednou, proto je celková spotřeba paměti závislá na velikosti největší skupiny vzorků, ne na součtu velikostí všech vzorků dohromady
- algoritmus zaručuje konečnost; v nejhorším případě musí být do triangulace vloženy všechny vzorky (tento případ nastává pouze pokud jsou všechna toleranční kritéria jednotlivých skupin dat nastavena na nulu)
- vložení jednoho vzorku v jednom kroku do Delaunayho triangulace způsobí pouze lokální změnu aproximace, to je ideální pro efektivní určení datového vzorku s maximální odchylkou od aproximace pro další upřesňující krok

5. Komprese a kódování triangulací

5.1 Komprese triangulací

Při reprezentaci obrázku v podobě triangulace je nutné ukládat jak souřadnice, tak i hodnoty intenzit jednotlivých vrcholů a trojúhelníků. V případě Delaunayho triangulace potřebujeme na uložení v podobě souboru mnoho bytů, proto není vhodná pro přímé ukládání. Je nutné triangulaci vhodně kódovat [3], případně již zakódovanou triangulaci následně komprimovat některým z existujících algoritmů komprese dat, jako je bzip2, deflate (používaný v ZIPu), ppdm, lzma (oba používané v 7z), paq8o, lpaq1 a quad (všechny dostupné na přiloženém CD). Způsob použití jednotlivých algoritmů komprese dat znázorňuje Tabulka 5.1.

algoritmus	kompresní program	parametry potřebné k použití požadovaného kompresního algoritmu
bzip2	7z.exe	a -tbzip2 -mmt=on -mx9 -md=900k -mpass=5
deflate	7z.exe	a -tzip -mm=Deflate64 -mx9 -md=64k -mfb=128 -mpass=5 -mem=AES256
ppdm	7z.exe	a -t7z -m0=PPMd -mx9 -mmem=192m -mo=32 -ms=on
lzma	7z.exe	a -t7z -m0=LZMA -mmt=on -mx9 -md=32m -mfb=64 -ms=on
paq8o	paq8o.exe	-1
lpaq1	lpaq1.exe	6
quad	quad.exe	-x

Tabulka 5.1 Algoritmy používané ke kompresi dat a jejich konkrétní použití [3].

5.2 Kódování triangulace metodou RAW

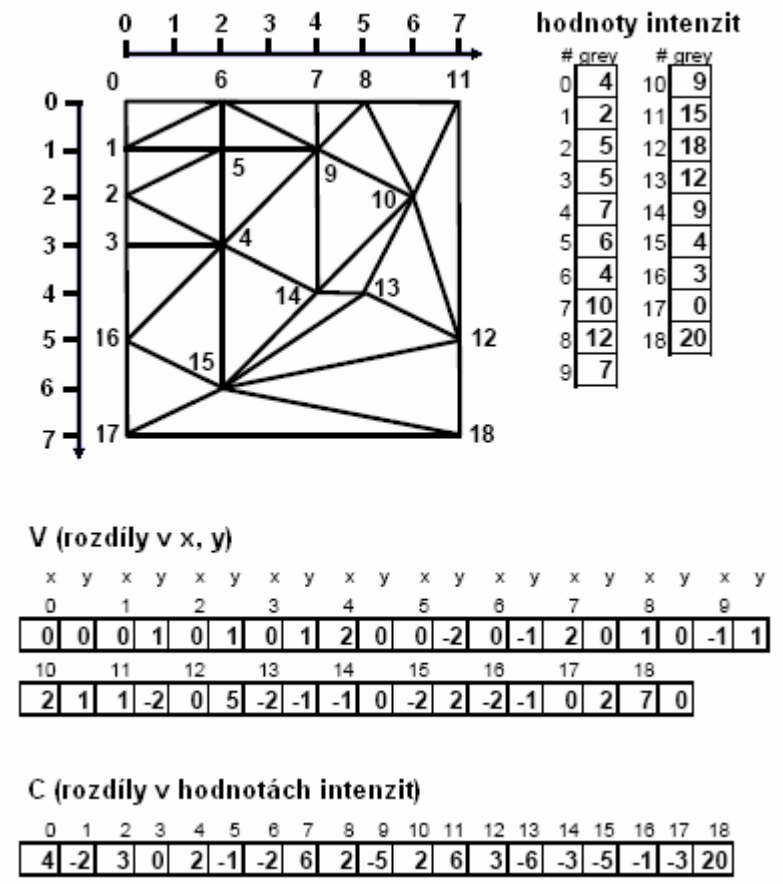
Výstupní soubor se skládá z hlavičky, která obsahuje velikost obrázku a počet vrcholů, po které následuje ukládání vrcholů jednoho po druhém v nekomprimované podobě. RAW [3] ukládá pro každý vrchol jeho (x, y) souřadnice následované jeho hodnotou intenzity používající 16-bitové celé číslo (integer) pro jednu souřadnici a 8-bitové celé číslo pro hodnotu intenzity, tj. 5 bytů je spotřebováno na jeden vrchol. Topologie není vůbec uložená, proto je tato metoda vhodná pouze pro Delaunayovu triangulaci. Vytvořený soubor je poté komprimovaný jedním z algoritmů komprese dat, očekávaný kompresní poměr je spíše malý.

5.3 Kódování triangulace metodou Vertex Path (VXPATH)

Tato metoda [3] postupně prochází všechny vrcholy a ukládá rozdíly mezi aktuálně kontrolovaným a naposled navštíveným vrcholem do dvou polí. V prvním poli V jsou uloženy rozdíly souřadnic (x, y), v druhém poli C jsou uloženy rozdíly hodnot intenzity. Při rekonstrukci obrazu se pozice vrcholů načítají z pole V, zatímco druhé pole C je použito pro výpočet hodnot intenzit. Vrcholy jsou procházeny v takovém pořadí, aby byl rozdíl souřadnic (x, y) minimální. Pokud jsme ve vrcholu p(px, py, pgrey), algoritmus pokračuje vrcholem q(qx, qy, qgrey), který ještě nebyl navštíven a Minkowského vzdálenost mezi těmito vrcholy, tj. hodnota:

$$|p_x - q_x| + |p_y - q_y|$$

je minimální. Obr. 5.1 zobrazuje triangulaci obrazu 8x8 pixelů, pořadí vrcholů, jejich hodnoty intenzit a odpovídající obsah obou polí V a C.



Obr. 5.1 Postup ukládání vrcholů metodou VXPATH [3].

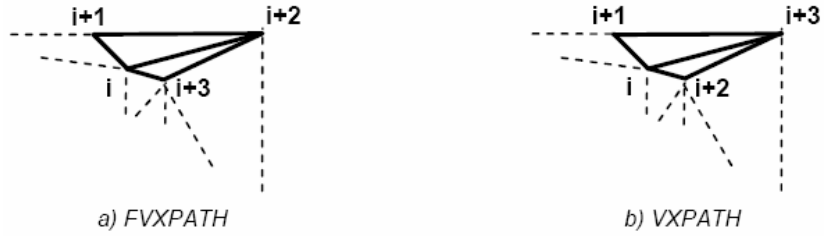
Metoda poté uloží hlavičku (viz metoda RAW) a minimum v poli rozdílů V. Poté jsou všechny hodnoty z tohoto pole, snížené na minimum, uloženy za použití tak malého počtu bitů, jak je to jen možné. V našem případě (Obr. 5.1) je min = -2 a max = 7, tzn. rozpětí je 9, a proto potřebujeme 4 bity $(1001)_2$. To je více, než bychom potřebovali, pokud bychom ukládali souřadnice místo rozdílů. Nicméně pro vetší obrazy je vysoce nepravděpodobné, že ukládání rozdílů bude spotřebovávat více bitů/vrchol než ukládání samotných souřadnic. Tak jako tak, tento přístup snižuje entropii[14] dat. Při použití kompresní metody dat na výstupním souboru, můžeme očekávat vyšší poměr komprese. Rozdíly v hodnotách intenzit jsou ukládány jinak. Očekává se, že dva sousedící vrcholy mohou mít úplně jinou hodnotu intenzity (v nejhorším případě může být rozdíl 255). Pokud aplikujeme stejnou kódující strategii pro pole rozdílů C, mohli bychom potřebovat min. 8 bitů pro každou hodnotu. Proto jsou hodnoty jednoduše ukládány s použitím 8 bitů. Výstupní soubor může být dále komprimovaný jedním z algoritmů komprese dat. Na druhou stranu, tento algoritmus hrubé síly běží v $O(N^2)$, což znamená, že zabírá mnoho času (zvláště pokud triangulace obsahuje velké množství vrcholů).

5.4 Kódování triangulace metodou Faster Vertex Path (FVXPATH)

Jedná se o modifikovanou [3] verzi metody VXPATH. Pro vrchol P algoritmus spočítá jeho Minkowského vzdálenost s každým vrcholem, který ještě nebyl navštíven a je spojený s tímto vrcholem jedním vrcholem v triangulaci a pro který je vzdálenost minimální. Pokud žádný takový vrchol neexistuje, pokračuje se jako v metodě

[14] veličina udávající míru neuspořádanosti zkoumaného systému nebo míru neurčitosti daného procesu

VXPATH. Algoritmus proto běží mnohem rychleji než VXPATH, ale generuje trochu rozdílné pole rozdílů (mimo jiné vytváří dvě pole Vx a Vy místo jediného pole V), které může obsahovat větší hodnoty (Obr. 5.2). Hodnoty intenzit jsou ukládány pomocí 9 bitů/vrchol.



Obr. 5.2 Rozdíl mezi pořadím vrcholů v metodě FVXPATH a VXPATH [3].

5.5 Některé další metody kódování triangulace

Mezi další často používané metody kódování [3] triangulace patří např.:

- Triangle Path
  - modifikace VXPATH
- Hilbert Space Filling Curve (BEHEC)
  - časová náročnost O(N)
  - malé soubory
- LZ Hilbert Space Filling Curve (LZHEC)
  - vychází z BEHEC
- KORILA
- LZ Image 3D Matrix (LZIM)
- Mueller (MUEKD)
  - paměťová náročnost

6. Implementace

6.1 Použitý software & hardware

Veškeré programy a pomocné utility vytvořené v rámci této bakalářské práce byly naprogramovány za použití následující kombinace programovacího jazyka a příslušného softwaru:

- programovací jazyk Microsoft Visual C# .NET
- vývojové prostředí Microsoft Visual Studio .NET 2003
- aplikační rámec Microsoft .NET Framework potřebný ke spuštění vytvořených programů (možnost bezplatného stáhnutí ze stránek Microsoftu)

Následuje specifikace počítačů, na kterých vznikala tato bakalářská práce. Veškeré programy byly naprogramovány na osobním počítači ACER TravelMate 2701LC_533:

- Intel Pentium 4 2.8GHz
- ATI RADEON 9000 IGP
- 40GB HDD
- 512MB RAM
- Operační systém Microsoft Windows XP Professional + SP2, verze 2002

Většina testů, výpočtů, zpracování výsledků, ale především veškerá výstupní data (obrázky, triangulace, zakódované triangulace, zakódované a následně komprimované triangulace, kotriangulace, atd.) potřebná k analyzování a zpracování výsledků, byla uložena na počítači ACER Aspire 5103NWLMi (byl zvolen především kvůli velikosti pevného disku, výstupní data dosahují řádově několika desítek GB):

- AMD Turion 1.6GHz
- ATI RADEON X1300
- 120GB HDD
- 512MB RAM
- Operační systém Microsoft Windows XP Professional + SP2, verze 2002

6.2 Popis řešení

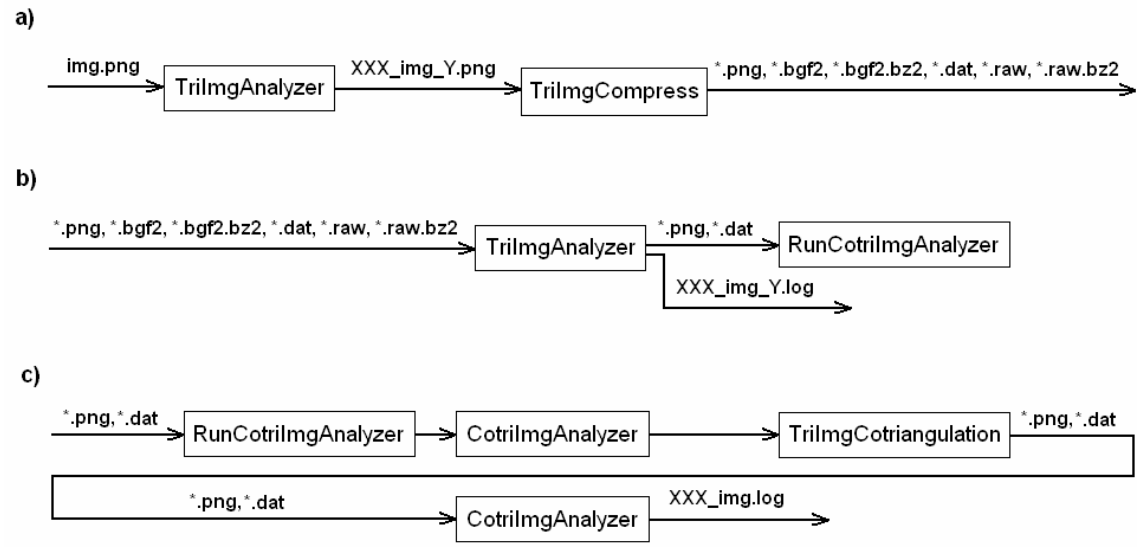
Cílem veškerého snažení je komprimovat vstupní obrázek v požadované kvalitě za použití triangulace (kotriangulace). Za tímto účelem bylo vyvinuto několik programů a utilit, které se postupně na tomto procesu podílejí. Od vlastní implementace reprezentace obrázků pomocí triangulací a kotriangulací bylo upuštěno, neboť tato metoda byla naimplementována v již hotovém softwaru, který byl poskytnut k testování. Autorem těchto převzatých programů je pan Ing. Josef Kohout, Ph.D. Mezi převzaté a použité programy pro testování a zpracovávání rastrového obrazu patří:

- TriImgCompress [19]
- TriImgCotriangulation [20]

Mezi programy, které byly vytvořeny autorem této bakalářské práce a které byly použity při analyzování výstupních dat patří:

- CotriImgAnalyzer (RunCotriImgAnalyzer)
- TriImgAnalyzer

Vzájemný vztah jmenovaných programů popisuje následující obrázek (Obr. 6.1).

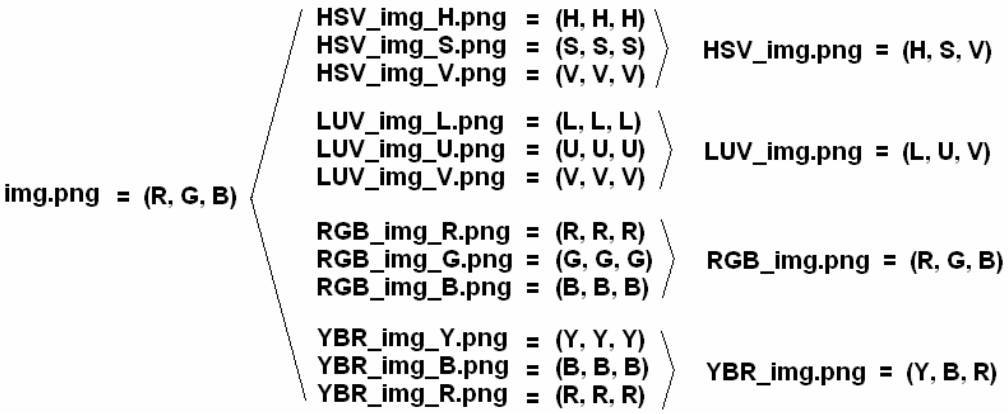


Obr. 6.1 Vzájemný vztah programů, použitých v rámci BP + jejich vstupy/výstupy [zdroj: vlastní].

Vstupem programu TriImgAnalyzer (Obr. 6.1a) je rastrový obraz, který je následně rozložen pomocí čtyř testovaných barevných systémů. Jedná se o barevné systémy:

- HSV
- L*u*v
- YCbCr
- RGB (barevný systém RGB je zde použit pouze jako výchozí model pro porovnání, zda byla aplikace ostatních barevných systémů účinnější či nikoli)

Po převodu vstupního rastrového obrazu do všech čtyř testovaných barevných systémů jsou tyto obrazy uloženy na disk po jednotlivých složkách (vždy tři soubory pro jeden vstupní obrázek a jeden barevný systém) pod zvoleným názvem souboru (Obr. 6.2).



Obr. 6.2 Program TriImgAnalyzer rozloží vstupní obrázek vždy na 12 obrázků, které jsou následujícího formátu: XXX_yyy_Z.png = (Z, Z, Z), kde XXX je použitý barevný systém, yyy je název vstupního souboru a Z je jedna ze tří složek barevného systému XXX. Barva každého pixelu v jednom konkrétním obrázku je uložena jako vektor (Z, Z, Z), kde Z je opět jedna ze tří složek barevného systému XXX. Takto rozložené obrázky umí program i následně opět složit do původní podoby [zdroj: vlastní].

Takto vzniklé soubory (šedotónové obrázky) jsou následně použity jako vstup programu TriImgCompress, který nad nimi vytvoří množinu dalších výstupních souborů a uloží je na disk. Tyto soubory mají následující význam (přípony jednotlivých souborů):

- *.dat obsahuje triangulaci vstupního obrázku
- *.bgf2 obsahuje zakódovanou triangulaci metodou FVXPATH
- *.raw obsahuje zakódovanou triangulaci metodou RAW
- *.bgf2.bz2 obsahuje komprimovanou, již zakódovanou triangulaci (*.bgf2)
- *.raw.bz2 obsahuje komprimovanou, již zakódovanou triangulaci (*.raw)
- *.png výstupní rastrový obrázek zrekonstruovaný z triangulace (*.dat)

Všechny tyto soubory jsou poté použity opět jako vstup programu TriImgAnalyzer (Obr. 6.1b), který na jejich základě vytvoří statistický soubor, obsahující všechny potřebné informace k analyzování výhodnosti jednotlivých barevných systémů, a také vytvoří vstupní data programům RunCotriImgAnalyzer, CotriImgAnalyzer a TriImgCotriangulation. Tyto data mají následující formát a význam:

- XXX_yyy_nnn-COTRI.png obrázek složený v RGB barevném systému
- XXX_yyy_nnn-COTRI_R.dat triangulace obrázku (první barevná složka)
- XXX_yyy_nnn-COTRI_G.dat triangulace obrázku (druhá barevná složka)
- XXX_yyy_nnn-COTRI_B.dat triangulace obrázku (třetí barevná složka)

kde XXX je použitý barevný systém, yyy je název vstupního souboru a nnn slouží jako počítadlo obrázků (0, 1, 2…).

Program RunCotriImgAnalyzer (Obr. 6.1c) tyto data postupně předává programům CotriImgAnalyzer a TriImgCotriangulation. Na základě těchto vstupních dat (složený obrázek a tři triangulace) vytvoří program TriImgCotriangulation sadu výstupních kotriangulací a obrázků vzniklých rekonstruováním dat konkrétní kotriangulace. Tento výstup je vstupem programu CotriImgAnalyzer, který tato data zpracuje a zakóduje metodami FVXPATH, RAW a takto zakódované kotriangulace komprimuje algoritmem komprese bzip2. Z těchto dat vytvoří statistický soubor, obsahující všechny potřebné informace k analyzování výhodnosti jednotlivých barevných systémů.

6.3 Převody mezi jednotlivými barevnými systémy

V principu není žádný problém převést obrázek uložený např. v modelu RGB na stejný obrázek uložený v jiném modelu (např. HSV). Jedná se o matematický přepočet nejčastěji 3 čísel, která definují barvu každého bodu. Jinými slovy: místo tří čísel typu RGB u každého bodu (pixelu) budou v novém modelu opět tři čísla, ale s významem HSV. Při praktickém převodu vždy nastane malá ztráta kvality. Dochází k tomu jednak proto, že různé modely mají obvykle různé gamuty[15], ale také nutnost zaokrouhlování na celá čísla v reálné 8 bitové reprezentaci provede malé posuny. Posuny způsobené zaokrouhlováním jsou ale prakticky nepostřehnutelné. Následující kapitoly se budou věnovat implementovaným převodům jednotlivých barevných systémů. Jedná se vždy o převod mezi barevným systémem RGB a cílovým barevným systémem (HSV, L*u*v, YCbCr) a naopak, tedy o převod mezi výchozími barevnými systémy (HSV, L*u*v, YCbCr) a cílovým barevným systémem RGB.

[15] dosažitelná oblast barev v určitém barevném prostoru

6.3.1 RGB

Barevný systém RGB je výchozím stavem všech námi použitých barevných obrázků. Není tedy potřeba nějakým složitým způsobem převádět hodnoty jednotlivých pixelů do podoby RGB, neboť všechny pixely jsou reprezentovány právě v podobě barevného vektoru (R, G, B). Barevný systém RGB je reprezentován třemi hodnotami:

- R intenzita červené barvy
- G intenzita zelené barvy
- B intenzita modré barvy

RGB -> R, G, B

Vstup: barevný obrázek (RGB)
Výstup: tři šedotónové obrázky obsahující vždy pouze jednu barevnou složku R, G a B

Cílem je rozdělit vstupní obrázek (barevný systém RGB) na jednotlivé barevné složky a takto oddělené složky následně uložit jako tři šedotónové obrázky. Vstupní obrázek je načten pomocí standardní komponenty (Bitmap). Následně je ve dvou vnořených FOR cyklech (0…(height-1), 0…(width-1)) zjištěna a uložena do proměnné hodnota každého pixelu vstupního obrázku v podobě vektoru (R, G, B). Dále je do pomocných proměnných uložena hodnota každé barevné složky zvlášť tak, že nám vzniknou tři šedotónové vektory (R, R, R), (G, G, G) a (B, B, B), které jsou použity jako vektory barvy pro uložení tří výstupních šedotónových (Obr. 6.2) obrázků (obsahují vždy jen jednu ze tří barevných složek, tedy jeden ze tří obrázků obsahuje pouze složku R, G a nebo B). Výstupní obrázky jsou ukládány vždy pomocí standardní komponenty ve formátu PNG.

R, G, B -> RGB

Vstup: tři šedotónové obrázky obsahující vždy pouze jednu barevnou složku R, G a B
Výstup: barevný obrázek (RGB)

Cílem je složit tři šedotónové obrázky (každý obsahuje vždy jen jednu ze tří barevných složek) v jeden výstupní barevný obrázek (RGB). Vstupní obrázky jsou načteny pomocí standardní komponenty (Bitmap). Následně jsou ve dvou vnořených FOR cyklech (0…(height-1), 0…(width-1)) zjištěny a uloženy do proměnných hodnoty všech pixelů ze vstupních obrázků v podobě vektorů (R, R, R), (G, G, G) a (B, B, B). Dále je do pomocné proměnné uložena hodnota každé barevné složky zvlášť tak, že nám vznikne jeden barevný vektor (R, G, B), který je použit jako vektor barvy pro uložení výstupního barevného obrázku (Obr. 6.2).

6.3.2 HSV

Barevný systém HSV je další z barevných systémů použitých k rozkládání a skládání obrázků po složkách. Situace už zde není tak jednoduchá, jako v případě barevného systému RGB. Hodnoty jednotlivých barevných složek H, S a V jsou vypočítány z původních hodnot R, G a B níže popsaným způsobem. Barevný systém HSV je v našem případě reprezentován třemi hodnotami:

- H odstín barvy
- S sytost barvy
- V jas barvy

RGB -> H, S, V

Vstup: barevný obrázek (RGB)
Výstup: tři šedotónové obrázky obsahující vždy pouze jednu barevnou složku H, S a V

Cílem je rozdělit vstupní obrázek (barevný systém RGB) na jednotlivé barevné složky a takto oddělené složky následně uložit jako tři šedotónové obrázky. Vstupní obrázek je načten pomocí standardní komponenty (Bitmap). Následně je ve dvou vnořených FOR cyklech (0…(height-1), 0…(width-1)) zjištěna a uložena do proměnné hodnota každého pixelu vstupního obrázku v podobě vektoru (R, G, B). Až do této chvíle byl postup stejný, jako v předchozím případě (RGB -> R, G, B). Hodnoty R, G a B jsou převedeny na interval <0, 1> prostým vydělením původní hodnoty číslem 255 (maximální možná hodnota barvy). Dále je zjištěno minimum a maximum z hodnot R, G a B. Následně jsou spočítány hodnoty H, S a V (kde H $\in$ <0, 360> je úhel odstínu, a S, V $\in$ <0,1> jsou sytost a jas) podle těchto vzorečků [10]:

$$H = \begin{cases} 0°, & \text{jestliže } max = min \\ 60° \times \frac{g-b}{max-min} + 0°, & \text{jestliže } max = R \text{ a } G \geq B \\ 60° \times \frac{g-b}{max-min} + 360°, & \text{jestliže } max = R \text{ a } G < B \\ 60° \times \frac{b-r}{max-min} + 120°, & \text{jestliže } max = G \\ 60° \times \frac{r-g}{max-min} + 240°, & \text{jestliže } max = B \end{cases}$$

$$S = \begin{cases} 0, & \text{jestliže } max = 0 \\ \frac{max-min}{max} = 1 - \frac{min}{max}, & \text{jinak} \end{cases}$$

$$V = max$$

Takto získané hodnoty barevných složek H, S a V jsou dále uloženy do pomocných proměnných zvlášť tak, že nám vzniknou tři šedotónové vektory (H, H, H), (S, S, S) a (V, V, V), které jsou použity jako vektory barvy pro uložení tří výstupních šedotónových (Obr. 6.2) obrázků (obsahují vždy jen jednu ze tří barevných složek, tedy jeden ze tří obrázků obsahuje pouze složku H, S a nebo V).

H, S, V -> RGB

Vstup: tři šedotónové obrázky obsahující vždy pouze jednu barevnou složku H, S a V
Výstup: barevný obrázek (RGB)

Cílem je složit tři šedotónové obrázky (každý obsahuje vždy jen jednu ze tří barevných složek) v jeden výstupní barevný obrázek (RGB). Vstupní obrázky jsou načteny pomocí standardní komponenty (Bitmap). Následně jsou ve dvou vnořených FOR cyklech (0…(height-1), 0…(width-1)) zjištěny a uloženy do proměnných hodnoty všech pixelů ze vstupních obrázků v podobě vektorů (H, H, H), (S, S, S) a (V, V, V). Jednotlivé hodnoty barevných složek S a V jsou převedeny na interval <0, 1> prostým vydělením původní hodnoty číslem 255, hodnota H je převedena na stupně <0, 360°>. Jestliže S = 0, pak je výsledná barva bezbarvá nebo šedá. V tomto zvláštním případě se všechny

složky R, G, a B rovnají hodnotě V. Pro případ, kdy je S ≠ 0, může být k výpočtu hodnot R, G a B použito následujících vzorečků [10]:

$$Hi = \left[\frac{H}{60}\right] \bmod 6 \qquad P = V \times (1 - S)$$

$$Q = V \times (1 - F \times S)$$

$$F = \frac{H}{60} - Hi \qquad T = V \times (1 - (1 - F) \times S)$$

Na základě těchto hodnot můžeme dopočítat barevný vektor (R, G, B) následujícím způsobem [10]:

$$(R, G, B) = \begin{cases} (V, T, P), & \text{jestliže } Hi = 0 \\ (Q, V, P), & \text{jestliže } Hi = 1 \\ (P, V, T), & \text{jestliže } Hi = 2 \\ (P, Q, V), & \text{jestliže } Hi = 3 \\ (T, P, V), & \text{jestliže } Hi = 4 \\ (V, P, Q), & \text{jestliže } Hi = 5 \end{cases}$$

Tento vektor barvy (R, G, B) je použit pro uložení výstupního barevného obrázku (Obr. 6.2).

*6.3.3 L*u*v*

Barevný systém L*u*v je z použitých barevných systémů nejkomplikovanější. Pro převod mezi RGB -> L*u*v a L*u*v -> RGB musí být použity dva mezikroky: barevný systém sRGB[16] a barevný systém XYZ [2]. Hodnoty jednotlivých barevných složek L, U a V jsou vypočítány z původních hodnot R, G a B níže popsaným způsobem. Barevný systém L*u*v je v našem případě reprezentován třemi hodnotami:

- L jas barvy
- U udává polohu mezi primárními barvami R/G
- V udává polohu mezi primárními barvami Y/B

RGB -> L, U, V

Vstup: barevný obrázek (RGB)
Výstup: tři šedotónové obrázky obsahující vždy pouze jednu barevnou složku L, U a V

Cílem je rozdělit vstupní obrázek (barevný systém RGB) na jednotlivé barevné složky a takto oddělené složky následně uložit jako tři šedotónové obrázky. Vstupní obrázek je načten pomocí standardní komponenty (Bitmap). Následně je ve dvou vnořených FOR cyklech (0…(height-1), 0…(width-1)) zjištěna a uložena do proměnné hodnota každého pixelu vstupního obrázku v podobě vektoru (R, G, B). Hodnoty R, G a B jsou převedeny na interval <0, 1> vydělením původní hodnoty číslem 255 (maximální

[16] sRGB je standardní RGB barevný prostor vytvořený pro použití na monitoru, tiskárnách a internetu

možná hodnota barvy). Dále jsou hodnoty R, G a B převedeny do barevného systému sRGB [13] podle následujícího vzorce [13]:

$$jestliže \quad R > 0.04045, \; pak \quad R = \left(\frac{R + 0.055}{1.055} \right)^{2.4}$$

$$jinak \quad R = \frac{R}{12.92}$$

Analogicky se vypočítají hodnoty G a B. Takto získané hodnoty R, G a B jsou vynásobeny hodnotou 100. Následují vzorce pro převod do barevného systému XYZ, po kterém už přijde na řadu převod do L*u*v [13]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Takto získané hodnoty X, Y a Z jsou dále použity pro výpočet pomocných proměnných pom_U, pom_V a pom_Y [13]:

$$pom\_U = \frac{4 \cdot X}{X + 15 \cdot Y + 3 \cdot Z}, \quad pom\_V = \frac{9 \cdot Y}{X + 15 \cdot Y + 3 \cdot Z}$$

$$pom\_Y = \frac{Y}{100}$$

$$jestliže \quad pom\_Y > 0.008856, \; pak \quad pom\_Y = pom\_Y^{1/3}$$

$$jinak \quad pom\_Y = 7.787 \cdot pom\_Y + \frac{16}{116}$$

Dále si zavedeme hodnoty ref_X, ref_Y a ref_Z, které použijeme pro výpočet pomocných proměnných ref_U a ref_V [13]:

$$ref\_X = 95.047, \quad ref\_Y = 100.000, \quad ref\_Z = 108.883$$

$$ref\_U = \frac{4 \cdot ref\_X}{ref\_X + 15 \cdot ref\_Y + 3 \cdot ref\_Z}, \quad ref\_V = \frac{9 \cdot ref\_Y}{ref\_X + 15 \cdot ref\_Y + 3 \cdot ref\_Z}$$

Nyní již můžeme vypočítat hodnotu jednotlivých složek barevného systému L*u*v, podle následujících vzorečků [13]:

$$L = (116 \cdot var\_Y) - 16$$

$$U = 13 \cdot L \cdot (var\_U - ref\_U)$$

$$V = 13 \cdot L \cdot (var\_V - ref\_V)$$

Takto získané hodnoty uložíme do vektorů (L, L, L), (U, U, U) a (V, V, V) a použijeme je jako vektory barvy pro uložení tří výstupních šedotónových (Obr. 6.2) obrázků.

L, U, V -> RGB

Vstup: tři šedotónové obrázky obsahující vždy pouze jednu barevnou složku L, U a V
Výstup: barevný obrázek (RGB)

Cílem je složit tři šedotónové obrázky (každý obsahuje vždy jen jednu ze tří barevných složek) v jeden výstupní barevný obrázek (RGB). Vstupní obrázky jsou načteny pomocí standardní komponenty (Bitmap). Následně jsou ve dvou vnořených FOR cyklech (0…(height-1), 0…(width-1)) zjištěny a uloženy do proměnných hodnoty všech pixelů ze vstupních obrázků v podobě vektorů (L, L, L), (U, U, U) a (V, V, V). Následný postup je víceméně analogický s převodem z RGB -> L, U, V, jen vše probíhá v opačném pořadí s drobnými změnami, proto zde bude ukázána kompletní sada vzorečků.
Nejprve je nutné zavést několik pomocných proměnných [13]:

$$pom\_Y = \frac{L+16}{116}$$

$$jestliže \quad pom\_Y^3 > 0.008856, \; pak \quad pom\_Y = pom\_Y^3$$

$$jinak \quad pom\_Y = \frac{\frac{pom\_Y - 16}{116}}{7.787}$$

$$ref\_X = 95.047, \quad ref\_Y = 100.000, \quad ref\_Z = 108.883$$

$$ref\_U = \frac{4 \cdot ref\_X}{ref\_X + 15 \cdot ref\_Y + 3 \cdot ref\_Z}, \quad ref\_V = \frac{9 \cdot ref\_Y}{ref\_X + 15 \cdot ref\_Y + 3 \cdot ref\_Z}$$

$$pom\_U = \frac{U}{13 \cdot L} + ref\_U, \quad ref\_V = \frac{V}{13 \cdot L} + ref\_V$$

Na základě těchto hodnot můžeme dopočítat barevný vektor (X, Y, Z) následujícím způsobem [13]:

$$Y = pom\_Y \cdot 100$$

$$X = \frac{-9 \cdot Y \cdot pom\_U}{(pom\_U - 4) \cdot pom\_V - pom\_U \cdot pom\_V}$$

$$Z = \frac{9 \cdot Y - 15 \cdot pom\_V \cdot Y - pom\_V \cdot X}{3 \cdot pom\_V}$$

Nyní si zavedeme pomocné proměnné pom_X, pom_Y a pom_Z, které získáme tak, že vydělíme hodnoty X, Y a Z číslicí 100, pomocí těchto hodnot vypočítáme pomocné proměnné pom_R, pom_G a pom_B, ze kterých dopočítáme konečný vektor (R, G, B):

$$\begin{bmatrix} pom\_R \\ pom\_G \\ pom\_B \end{bmatrix} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} \cdot \begin{bmatrix} pom\_X \\ pom\_Y \\ pom\_Z \end{bmatrix}$$

$$jestliže \quad pom\_R > 0.0031308, \; pak \quad pom\_R = 1.055 \cdot pom\_R^{1/2.4} - 0.055$$
$$jinak \quad pom\_R = 12.92 \cdot pom\_R$$

$$R = pom\_R \cdot 255$$

Analogicky se vypočítají hodnoty pom_G a pom_B, které jsou dále vynásobeny konstantou 255, čímž získáme barevné složky G a B. Takto získaný vektor (R, G, B) je použit pro uložení výstupního barevného obrázku (Obr. 6.2).

6.3.4 YCbCr

Barevný systém YCbCr je posledním z barevných systémů použitých k rozkládání a skládání obrázků po složkách. Hodnoty jednotlivých barevných složek Y, Cb a Cr jsou vypočítány z původních hodnot R, G a B níže popsaným způsobem. Barevný systém YCbCr je v našem případě reprezentován třemi hodnotami:

- Y jas barvy
- Cb modrá chrominanční komponenta
- Cr červená chrominanční komponenta

RGB -> Y, Cb, Cr

Vstup: barevný obrázek (RGB)
Výstup: tři šedotónové obrázy obsahující vždy pouze jednu barevnou složku Y, Cb a Cr

Cílem je rozdělit vstupní obrázek (barevný systém RGB) na jednotlivé barevné složky a takto oddělené složky následně uložit jako tři šedotónové obrázky. Vstupní obrázek je načten pomocí standardní komponenty (Bitmap). Následně je ve dvou vnořených FOR cyklech (0…(height-1), 0…(width-1)) zjištěna a uložena do proměnné hodnota každého pixelu vstupního obrázku v podobě vektoru (R, G, B). Pomocí následujícího vzorečku [10] jsou vypočteny hodnoty Y, Cb a Cr:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Takto získané hodnoty barevných složek Y, Cb a Cr jsou dále uloženy do pomocných proměnných zvlášť tak, že nám vzniknou tři šedotónové vektory (Y, Y, Y), (Cb, Cb, Cb) a (Cr, Cr, Cr), které jsou použity jako vektory barvy pro uložení tří výstupních šedotónových (Obr. 6.2) obrázků (obsahují vždy jen jednu ze tří barevných složek, tedy jeden ze tří obrázků obsahuje pouze složku Y, Cb a nebo Cr).

Y, Cb, Cr -> RGB

Vstup: tři šedotónové obrázky obsahující vždy pouze jednu barevnou složku Y, Cb a Cr
Výstup: barevný obrázek (RGB)

Cílem je složit tři šedotónové obrázky (každý obsahuje vždy jen jednu ze tří barevných složek) v jeden výstupní barevný obrázek (RGB). Vstupní obrázky jsou načteny pomocí

standardní komponenty (Bitmap). Následně jsou ve dvou vnořených FOR cyklech (0…(height-1), 0…(width-1)) zjištěny a uloženy do proměnných hodnoty všech pixelů ze vstupních obrázků v podobě vektorů (Y, Y, Y), (Cb, Cb, Cb) a (Cr, Cr, Cr). Pomocí následujících vzorečků [10] jsou vypočteny hodnoty R, G a B:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.772 & 0 \end{bmatrix} \cdot \begin{bmatrix} Y \\ (Cb-128) \\ (Cr-128) \end{bmatrix}$$

Tento vektor barvy (R, G, B) je použit pro uložení výstupního barevného obrázku (Obr. 6.2).

6.4 Zpracování dat - triangulace

Tato podkapitola se bude věnovat otázce zpracování veškerých dat, která poskytuje program TriImgCompress. Na základě těchto dat umí program TriImgAnalyzer vytvořit statistický soubor, který obsahuje všechny potřebné informace k vytvoření pomocných tabulek v programu MS Excel, na základě kterých mohou být zkonstruovány grafy, které jsou tématem diskuse v následující kapitole (7. Analýza výsledků).

Program TriImgCompress má naimplementováno:

- reprezentaci obrázků Delaunayho triangulací
- metodu kódování FVXPATH
- metodu kódování RAW

Vstup: šedotónový obrázek

Výstup:

- triangulace vstupního obrázku (*.dat)
- zakódované triangulace vstupního obrázku metodou FVXPATH (*.bgf2)
- zakódované triangulace vstupního obrázku metodou RAW (*.raw)
- komprimované, již zakódované triangulace metodou FVXPATH (*.bgf2.bz2)
- komprimované, již zakódované triangulace metodou RAW (*.raw.bz2)
- obrázek zrekonstruovaný z Delaunayho triangulace (*.png)

Formát výstupních souborů je následující: filename_REC_PM600_FM0_M_PSNR.*

kde jednotlivé symboly znamenají:

- **filename** název vstupního obrázku
- **M** počet vrcholů, které byly z obrázku vypuštěny
- **PSNR** PSNR zrekonstruovaného obrázku z Delaunayho triangulace
- ***** jedna z koncovek dat, bgf2, raw, bgf2.bz2, raw.bz2, png

Takto vzniklé soubory slouží jako vstup programu TriImgAnalyzer. Ten je zpracuje níže popsaným způsobem. Cílem zájmu jsou především hodnoty PSNR, M, velikost jednotlivých souborů v bytech, barevný systém zpracovávaného obrázku a konkrétní barevná složka tohoto obrázku.

Program v první fázi načte všechny soubory do pole, které následně setřídí podle názvu jednotlivých souborů. Každý název souboru tohoto pole je rozparsován na jednotlivé části. Takto získané informace jsou dále ukládány do struktury, obsahující položky:

- **fileFullName** plný název souboru
- **originalFileName** originální název souboru
- **colorSystem** barevný systém
- **colorSystemComponent** barevná složka
- **PSNR** PSNR
- **M** počet vypuštěných vrcholů
- **resolution** rozlišení
- **fileSize_bgf2, fileSize_bgf2bz2, fileSize_dat** velikost souboru
- **fileSize_png, fileSize_raw, fileSize_rawbz2** velikost souboru

Nyní máme všechny potřebné informace o souborech, které generuje program TriImgCompress, uloženy v poli struktur. Na tomto poli nyní můžeme spustit funkci vytvářející výstupní statistický soubor. Program TriImgAnalyzer umí pracovat ve dvou režimech:

- PSNR
- N

Každý přistupuje k získaným informacím odlišným způsobem, viz následující dvě podkapitoly.

6.4.1 Přístup podle PSNR

Přístup k získaným informacím podle PSNR funguje tak, že se k první barevné složce (např. R složka v případě RGB) barevného systému hledá druhá a třetí barevná složka (G a B v případě RGB) ze všech možných, která má PSNR nejbližší k PSNR první barevné složky, viz následující příklad. Tyto tři barevné složky jsou následně kompletovány dohromady a vzniká tak výstupní barevný obrázek.

Příklad

1. RGB_Lena_R_PM600_FM0_100_10.10.png
2. RGB_Lena_R_PM600_FM0_200_20.10.png
3. RGB_Lena_R_PM600_FM0_400_25.10.png
4. RGB_Lena_G_PM600_FM0_100_08.10.png
5. RGB_Lena_G_PM600_FM0_200_15.10.png
6. RGB_Lena_G_PM600_FM0_400_22.10.png
7. RGB_Lena_B_PM600_FM0_100_20.10.png
8. RGB_Lena_B_PM600_FM0_200_26.10.png
9. RGB_Lena_B_PM600_FM0_400_30.10.png

atd.

Podle výše popsaného postupu budou kombinace barevných složek následující:

1-4-7 2-6-7 3-6-8

Z toho plyne, že druhá barevná složka č.5 vůbec nebude při zpracování použita, stejně tak že nebude při zpracovávání statistik použita třetí barevná složka č.9, neboť jejich PSNR není nejbližší k PSNR první barevné složky. V případě první barevné složky budou vždy použity všechny soubory. Tyto kombinace souborů jsou ukládány do pole struktur. Struktura má následující tvar:

- **colorComponent1** pozice v poli první barevné komponenty
- **colorComponent2** pozice v poli druhé barevné komponenty
- **colorComponent3** pozice v poli třetí barevné komponenty

V momentě, kdy máme zkombinované příslušné soubory dohromady, můžeme dokončit vytváření statistického souboru. Ten má následující výstupní formát:

XXX_filename_stats_PSNR.log

kde jednotlivé symboly znamenají:

- **XXX** zpracovávaný barevný systém
- **filename** název vstupního obrázku
- **PSNR** označuje, že se jedná o přístup podle PSNR

Výstupní soubor XXX_filename_stats_PSNR.log obsahuje tyto informace: PSNR, PSNR_slozka1, PSNR_slozka2, PSNR_slozka3, N_celkem, N_slozka1, N_slozka2, N_slozka3, bgf2_celkem, bgf2_slozka1, bgf2_slozka2, bgf2_slozka3, bgf2.bz2_celkem, bgf2.bz2_slozka1, bgf2.bz2_slozka2, bgf2.bz2_slozka3, dat_celkem, dat_slozka1, dat_slozka2, dat_slozka3, png_celkem, png_slozka1, png_slozka2, png_slozka3, raw_celkem, raw_slozka1, raw_slozka2, raw_slozka3, raw.bz2_celkem, raw.bz2_slozka1, raw.bz2_slozka2, raw.bz2_slozka3.

Význam je ve zkratce následující: první položka obsahuje PSNR zrekonstruovaného obrázku; PSNR_slozkaX obsahuje PSNR X-té komponenty; N_slozkaX obsahuje počet vrcholů triangulace X-té komponenty, v N_celkem jsou tyto hodnoty sečteny, dat_slozkaX je velikost příslušného *.dat souboru X-té komponenty (v Bytech), atd.

Program dále generuje vstupní data programům RunCotriImgAnalyzer, CotriImgAnalyzer a TriImgCotriangulation. Jedná se o barevný obrázek a tři triangulace jednotlivých barevných složek tohoto barevného obrázku. Tento obrázek je skládán barevným systémem RGB i v případě, že jednotlivé barevné složky jsou uloženy v barevném systému HSV, L*u*v či YCbCr, neboť program TriImgCotriangulation pracuje pouze v režimu RGB.

6.4.2 Přístup podle počtu vrcholů triangulace N

Přístup k získaným informacím podle počtu vrcholů triangulace N funguje tak, že se k první barevné složce (např. R složka v případě RGB) barevného systému hledá druhá a třetí barevná složka (G a B v případě RGB) ze všech možných, která má stejné N jako první barevná složka, viz následující příklad. Tyto tři barevné složky jsou následně kompletovány dohromady a vzniká tak výstupní barevný obrázek.

Příklad

1. RGB_Lena_R_PM600_FM0_100_10.10.png
2. RGB_Lena_R_PM600_FM0_200_20.10.png
3. RGB_Lena_R_PM600_FM0_400_25.10.png
4. RGB_Lena_G_PM600_FM0_100_08.10.png
5. RGB_Lena_G_PM600_FM0_200_15.10.png
6. RGB_Lena_G_PM600_FM0_400_22.10.png
7. RGB_Lena_B_PM600_FM0_100_20.10.png
8. RGB_Lena_B_PM600_FM0_200_26.10.png
9. RGB_Lena_B_PM600_FM0_400_30.10.png

atd.

Podle výše popsaného postupu budou kombinace barevných složek následující:

1-4-7 2-5-8 3-6-9

Z toho plyne, že budou při zpracování statistik použity vždy všechny soubory a informace, které obsahují. Tyto kombinace souborů jsou ukládány do pole struktur (struktura má stejný tvar, jako v předchozí kapitole 6.4.1 Přístup podle PSNR). V momentě, kdy máme zkombinované příslušné soubory dohromady, můžeme dokončit vytváření statistického souboru. Ten má následující výstupní formát:

XXX_filename_stats_N.log

kde jednotlivé symboly znamenají:

- **XXX** zpracovávaný barevný systém
- **filename** název vstupního obrázku
- **N** označuje, že se jedná o přístup podle N

Výstupní soubor XXX_filename_stats_N.log obsahuje stejné informace jako výstupní soubor popsaný v předcházející kapitole 6.4.1 Přístup podle PSNR.

Program dále generuje vstupní data programům RunCotriImgAnalyzer, CotriImgAnalyzer a TriImgCotriangulation. Jedná se o barevný obrázek a tři triangulace jednotlivých barevných složek tohoto barevného obrázku. Tento obrázek je skládán barevným systémem RGB i v případě, že jednotlivé barevné složky jsou uloženy v barevném systému HSV, L*u*v či YCbCr, neboť program TriImgCotriangulation pracuje pouze v režimu RGB.

6.5 Zpracování dat - kotriangulace

Tato podkapitola je věnována otázce zpracování veškerých dat, která poskytuje program TriImgCotriangulation. Na základě těchto dat umí program CotriImgAnalyzer vytvořit statistický soubor, který obsahuje všechny potřebné informace k vytvoření pomocných tabulek v programu MS Excel, na základě kterých mohou být zkonstruovány grafy, které jsou tématem diskuse v následující kapitole (7. Analýza výsledků).

Program TriImgCotriangulation má naimplementováno:

- reprezentaci obrázků pomocí kotriangulací

Vstup:

- barevný obrázek složený v RGB barevném systému
- triangulace obrázku (první barevná složka)
- triangulace obrázku (druhá barevná složka)
- triangulace obrázku (třetí barevná složka)

Výstup:

- množina kotriangulací ze tří vstupních triangulací (*.dat)
- obrázek zrekonstruovaný z kotriangulace (*.png)

Formát výstupních souborů je následující: REC_N_PSNR.*

kde jednotlivé symboly znamenají:

- **N** počet vrcholů v kotriangulaci
- **PSNR** PSNR zrekonstruovaného obrázku z kotriangulace
- ***** jedna z koncovek dat, png

Takto vzniklé soubory slouží jako vstup programu CotriImgAnalyzer. Ten je zpracuje níže popsaným způsobem. Cílem zájmu jsou především hodnoty PSNR, N a velikost jednotlivých souborů v bytech.

Program vezme množinu všech kotriangulací a rastrových obrázků, vybere z ní pouze jednu dvojici kotriangulace/rastrový obrázek, jejíž PSNR je nejbližší PSNR obrázku, který byl vstupem programu TriImgCotriangulation. Tuto dvojici uloží na disk do adresáře Results_cotri. Dále zakóduje kotriangulaci metodou FVXPATH, RAW a takto zakódované kotriangulace komprimuje algoritmem komprese bzip2. Tím nám vnikne sada šesti souborů dále určených ke zpracování. Výstupní formát těchto souborů je následující:

XXX_filename_nnn-COTRI_N_PSNR.*

kde jednotlivé symboly znamenají:

- **XXX** barevný systém původního obrázku
- **filename** název původního obrázku
- **nnn** počítadlo obrázků (0, 1, 2…)
- **N** počet vrcholů v kotriangulaci
- **PSNR** PSNR zrekonstruovaného obrázku z kotriangulace
- ***** jedna z koncovek bgf2, bgf2.bz2, dat, png, raw, raw.bz2

V další fázi program načte a zpracuje všechny soubory (viz předcházející odstavec) do pole, které následně setřídí podle názvu jednotlivých souborů. Každý název souboru v takto setříděném poli je rozparsován na jednotlivé části. Takto získané informace jsou dále ukládány do struktury, obsahující položky:

- **PSNR** PSNR
- **N** počet vrcholů triangulace
- **fileSize_bgf2, fileSize_bgf2bz2, fileSize_dat** velikost souboru
- **fileSize_png, fileSize_raw, fileSize_rawbz2** velikost souboru

Nyní máme všechny potřebné informace o souborech, které generují programy TriImgCotriangulation a CotriImgAnalyzer, uloženy v poli struktur. Na tomto poli nyní můžeme spustit funkci vytvářející výstupní statistický soubor. Ten má následující výstupní formát:

XXX_filename_stats.log

kde jednotlivé symboly znamenají:

- **XXX** barevný systém původního obrázku
- **filename** název původního obrázku

Výstupní soubor XXX_filename_stats.log obsahuje tyto informace: PSNR, N, bgf2, bgf2.bz2, dat, png, raw, raw.bz2.

Význam jednotlivých položek je následující: PSNR obsahuje PSNR zrekonstruovaného obrázku z kotriangulace, N obsahuje počet vrcholů kotriangulace, bgf2 obsahuje velikost příslušného *.bgf2 souboru (v Bytech), bgf2.bz2 obsahuje velikost příslušného *.bgf2.bz2 souboru (v Bytech), atd.

6.6 PSNR

Další pomocná utilita načítající dva obrázky, ze kterých následně spočítá a zobrazí jejich odchylku (PSNR). Po spuštění je testována správnost syntaxe na vstupu. Proběhne-li vše v pořádku, je zavolána hlavní metoda programu, které jsou jako parametry předány barevné obrázky, mezi nimiž se má PSNR spočítat. Tato metoda vytvoří objekt třídy BitmapClass, která pracuje s barevnými obrázky (ověřuje existenci obrázků, vrací jejich výšku, šířku, barvu pixelu atd.). Následuje výkonná část kódu. Ve dvou vnořených FOR cyklech je počítána mezihodnota SE, potřebná pro výpočet MSE a tedy i celkové hodnoty PSNR (viz kapitola 3.4 Objektivní měření vizuální kvality komprese). Program řeší i případ, že by byly porovnávány dva shodné obrázky. V takovém případě je PSNR přiřazena hodnota 100dB, kvalita obrazu po rekonstrukci v porovnání s původním obrazem je maximální, tedy beze změn.

6.7 RunCotriImgAnalyzer

Pomocná utilita spouštějící program CotriImgAnalyzer. Slouží především k ulehčení práce s velkým množstvím dat, která generuje program TriImgCotriangulation. Vstupem je adresář obsahující vstup programů CotriImgAnalyzer a TriImgCotriangulation. Příkazem FOREACH jsou postupně zpracovány všechny soubory z tohoto vstupního adresáře. Před přechodem na další soubor je výstupní adresář Results, který obsahuje data generovaná programem TriImgCotriangulation smazán, neboť obsahuje řádově několik GB dat, která jsou již v tuto chvíli „zbytečná“. Program je také možné spustit s parametrem „-stats“, který vyvolá spuštění programu CotriImgAnalyzer s tímto parametrem, na základě kterého dojde k vytvoření statistického souboru.

7. Analýza výsledků

7.1 Použité obrázky

Všechny testy, experimenty a výpočty v rámci této bakalářské práce byly prováděny na sadě 15 obrázků ve formátu PNG (viz 3.3 Formát PNG). Výběr byl proveden s přihlédnutím na typovou různorodost (např. obrázky s ucelenějšími plochami barev apod.). Data získaná z těchto obrázků (kvalita, kompresní poměr, velikost výstupních souborů, atd.) jsou díky tomu věrohodná a výsledky experimentů všeobecně použitelné a zobecnitelné.

Vybrané obrázky lze rozdělit na dvě základní kategorie, na fotografie a uměle vytvořené rastrové soubory, a následně na několik podkategorií:

- obrázky známé a často používané (baboon, fruits, lena)
- obrázky barevně bohaté (girl, peppers, pool, redmagic, tulips, yacht)
- obrázky barevně nevýrazné (aerials, airplane, car, drop, earth, woman)

Obrázky girl.png a tulips.png mají formát 768x512 pixelů, redmagic.png je ve formátu 594x766 pixelů a na konec obrázek pool.png má formát 510x383 pixelů. V ostatních případech se jedná o rastrové obrázky formátu 512x512 pixelů. Všech 15 testovacích souborů má barevnou hloubku 24 bitů/pixel.

7.2 Triangulace

Tato kapitola se věnuje zhodnocení kvality jednotlivých obrázků, kde jsou jednotlivé barevné komponenty zpracovávány odděleně jako tři nezávislé šedotónové obrazy a výsledkem procesu jsou tři nezávislé triangulace. Bylo experimentováno s různými barevnými systémy (HSV, L*u*v, RGB, YCbCr), které se nemalou částí podílejí na dosaženém kompresním poměru.

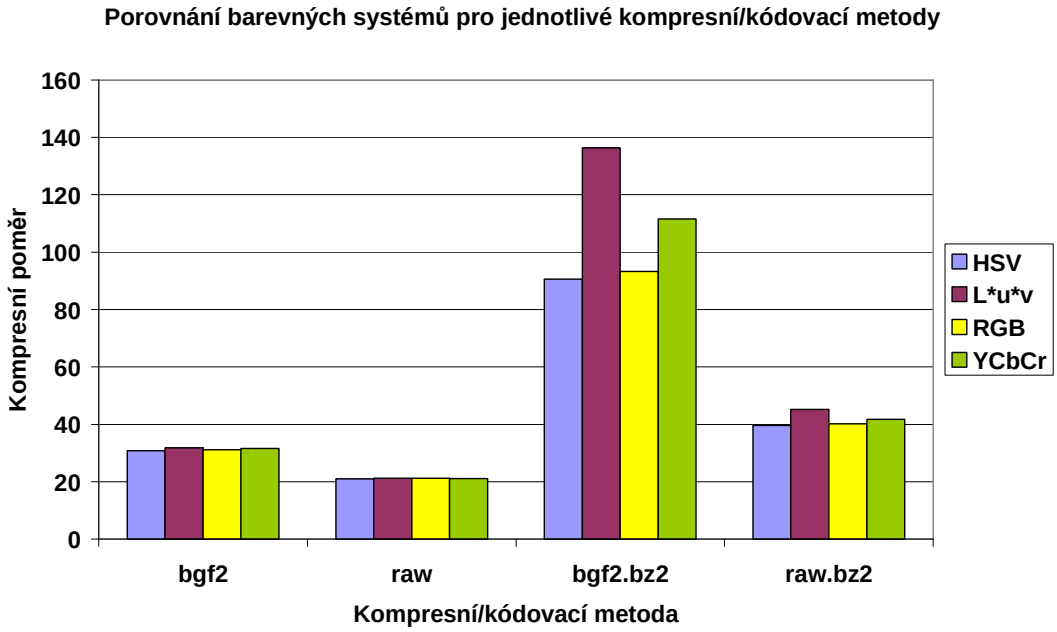
Poznámka

Dále v textu je mluveno pouze o jedné jediné triangulaci. Tato skutečnost je zapříčiněna tím, že je na tyto tři nezávislé triangulace nahlíženo jako na součet počtu vrcholů těchto triangulací, součet velikostí jednotlivých souborů, atd. Pro názornost: hodnota počet vrcholů triangulace N = N1 + N2 + N3, kde N1 je počet vrcholů triangulace první barevné složky, N2 je počet vrcholů triangulace druhé barevné složky a N3 je počet vrcholů triangulace třetí barevné složky. Analogicky je mluveno o jedné velikosti souborů *.png, *.bgf2, *.bgf2.bz2, *.raw, *.raw.bz2, *.dat.

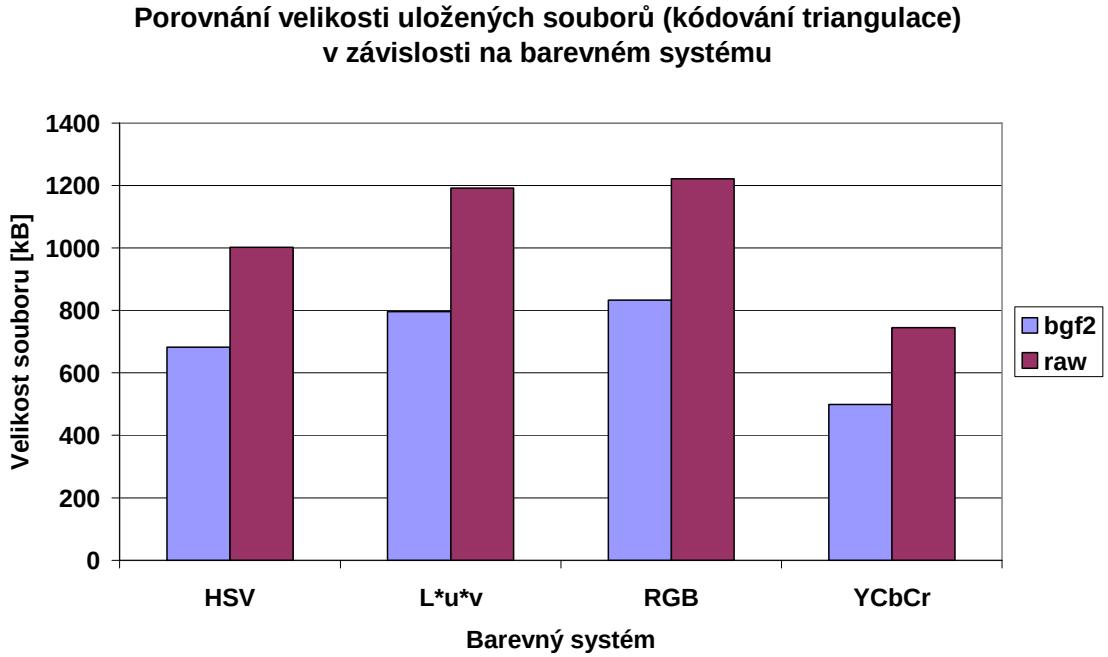
7.2.1 Vliv barevného systému na typ kódování triangulace

Experimenty s různými obrázky ukázaly, že při porovnání námi zvolených metod kódování triangulací (bgf2 - FVXPATH, raw - RAW) vychází lépe metoda FVXPATH. Vliv barevného systému je v tomto porovnání nezajímavý. Odchylky v kompresním poměru zakódovaných vůči nezakódovaným triangulacím (Delaunayova triangulace) jsou v rámci jedné metody minimální (Obr. 7.1), tedy zanedbatelné. Typické chování vykazuje např. obrázek fruits.png, vyloženě odlišné chování nevykazoval žádný z testovaných obrázků.

Jinak tomu je při porovnávání velikosti jednotlivých souborů (zakódovaných triangulací) uložených na disku (Obr. 7.2). Zde už je vliv barevných systémů poměrně znatelný. Jednoznačně nejlepším je z tohoto ohledu barevný systém YCbCr. Triangulace kódovány jak metodou FVXPATH, tak i metodou RAW, jsou při použití tohoto barevného systému a při následném ukládání na disk v podobě souborů nejmenší. Dále nám z hlediska velikosti zakódovaných triangulací jako nejvhodnější vychází barevný systém HSV. Za těmito dvěma barevnými systémy zaostávají RGB a L*u*v, ty už se od sebe příliš neliší, ale v porovnání s HSV a YCbCr jsou výrazně horší. Typické chování (Obr. 7.2) vykazují např. obrázky fruits.png a baboon.png, oproti tomu se atypicky choval obrázek car.png, kde byl nejvýhodnější barevný systém L*u*v, následovaný barevnými systémy YCbCr, RGB a HSV. Velikosti triangulací zakódovaných metodou RAW jsou přibližně 1.4x větší oproti metodě FVXPATH.

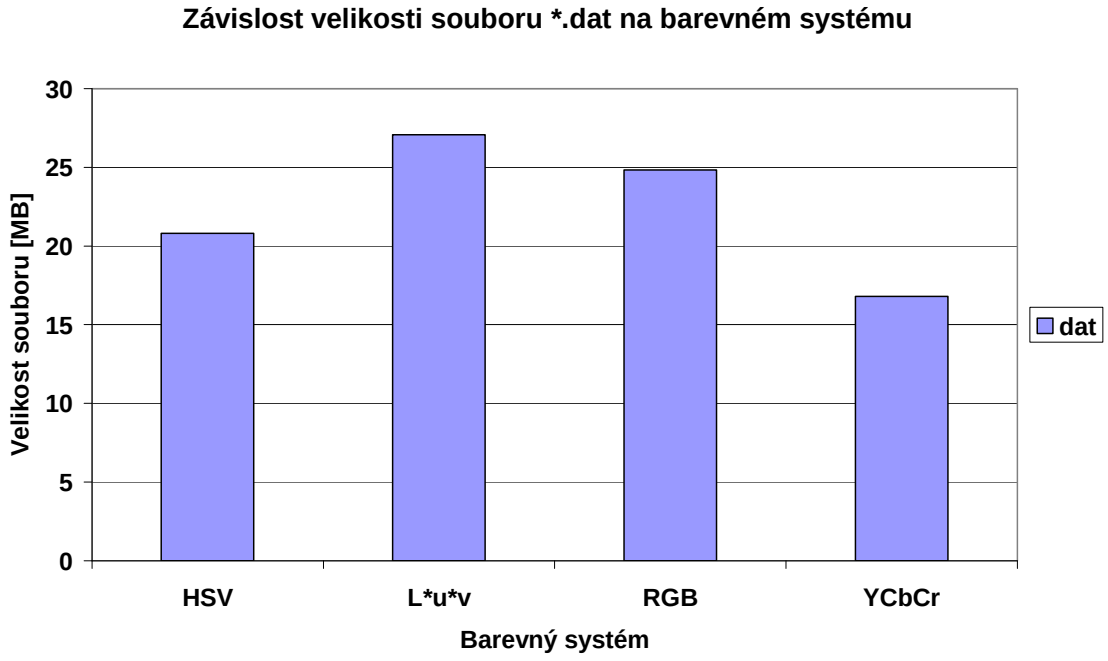


Obr. 7.1 Charakteristiky tohoto obrázku (fruits.png) znázorňují typické chování vlivu jednotlivých barevných systémů na typ kódování (skupina bgf2 a raw) a vliv jednotlivých barevných systémů na kompresi takto zakódovaných triangulací (skupina bgf2.bz2 a raw.bz2), ve všech případech je kompresní poměr porovnáván vzhledem k základní triangulaci [zdroj: vlastní].



Obr 7.2 Na tomto obrázku (fruits.png) je ukázán vliv jednotlivých barevných systémů na velikost ukládaného souboru (zakódované triangulace) [zdroj: vlastní].

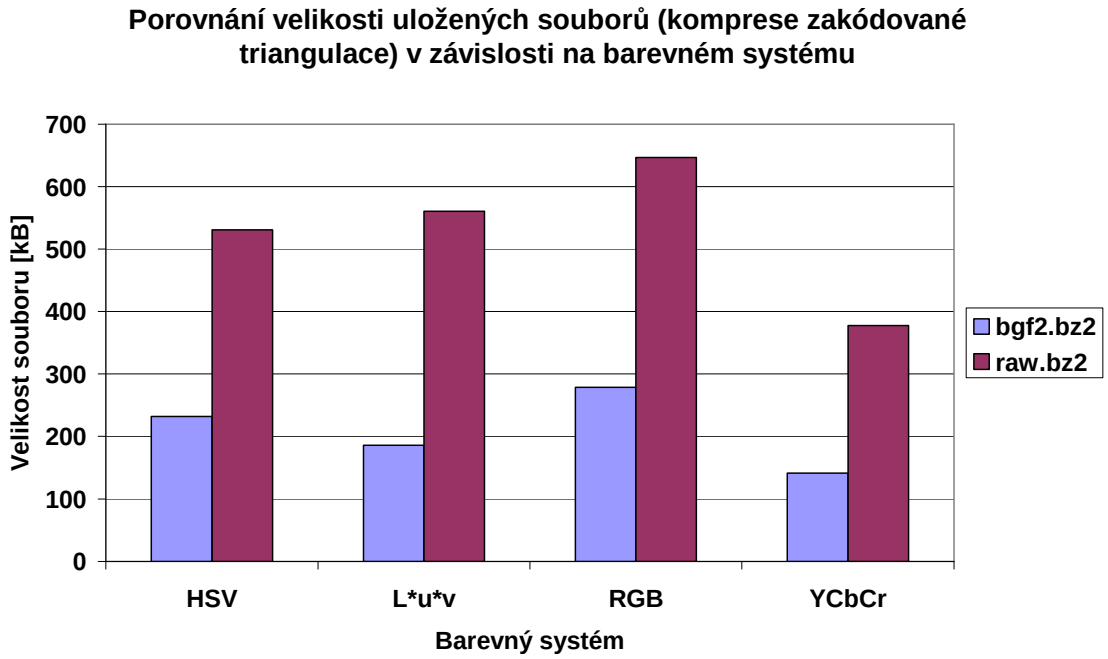
Při porovnávání velikosti jednotlivých souborů uložených na disku v nezakódované podobě (Obr 7.3), tedy v podobě Delaunayovy triangulace, vychází nejlépe opět YCbCr, následovaný barevným systémem HSV, RGB a L*u*v. Typické chování (Obr. 7.3) vykazují např. obrázky fruits.png a baboon.png, atypicky se opět choval obrázek car.png, kde byl nejvýhodnější barevný systém L*u*v, následovaný barevnými systémy YCbCr, RGB a HSV.



Obr. 7.3 Obrázek (baboon.png) znázorňuje nejčastější vliv barevného systému na velikost ukládaného souboru s triangulací *.dat [zdroj: vlastní].

7.2.2 Vliv barevného systému na kompresi již zakódované triangulace

K výrazným změnám dochází při použití komprese na již zakódované triangulace pomocí výše popsaných metod (FVXPATH a RAW). Metoda FVXPATH vychází opět mnohem lépe než metoda RAW. Vliv barevného systému začíná hrát svou významnou roli jak při porovnání kompresních poměrů (Obr. 7.1), tak při porovnání velikostí jednotlivých uložených souborů na disku (Obr. 7.4).



Obr. 7.4 Na tomto obrázku (fruits.png) je ukázán vliv jednotlivých barevných systémů na velikost ukládaného souboru (komprimované již zakódované triangulace) [zdroj: vlastní].

Kompresní poměr je nejlepší při použití barevného systému L*u*v. Takto zakódované a následně komprimované soubory vykazují v průměru kompresní poměr cca. 130 vůči originální triangulaci a kompresní poměr cca 4.5 vůči již zakódované triangulaci. Při použití barevného systému YCbCr získáme o něco horší, nicméně také velmi dobré výsledky. Kompresní poměr je cca. 110 vůči originální triangulaci a 3.5 vůči již zakódované triangulaci. Barevné systémy RGB a HSV vykazují navzájem podobné výsledky, ale opět výrazně zaostávají za barevnými systémy L*u*v a YCbCr co se kompresních poměrů týče (cca 80 vůči originální triangulaci a 2.5 vzhledem k již zakódované triangulaci). Typické chování (Obr. 7.1) vykazují obrázky airplane.png, baboon.png, fruits.png či tulips.png, oproti tomu se atypicky chová obrázek drop.png, kde byl z pohledu kompresního poměru nejvýhodnější barevný systém RGB, následovaný barevnými systémy L*u*v, YCbCr a HSV.

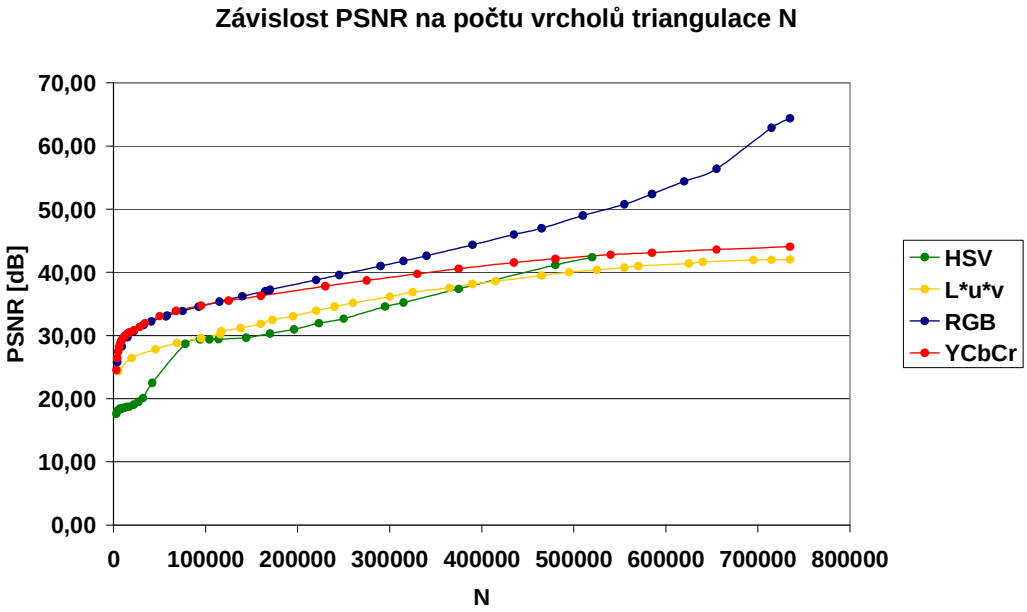
Znatelně se na velikosti jednotlivých souborů (zakódovaných a následně komprimovaných triangulací) uložených na disku podepisuje jak vliv barevného systému (Obr. 7.4), tak vliv použité kódovací metody (velikost výsledných souborů získaných metodou kódování RAW jsou přibližně 1.4x větší než při použití metody FVXPATH). Jednoznačně nejlepším je z tohoto ohledu opět barevný systém YCbCr. Triangulace kódovány jak metodou FVXPATH, tak i metodou RAW, které byly následně komprimovány jsou při použití tohoto barevného systému a při následném ukládání na disk v podobě souborů nejmenší. Dále jako nejvýhodnější vychází barevné systémy HSV a L*u*v. Za těmito dvěma barevnými systémy zaostává (co se velikosti

jednotlivých souborů týče) RGB. Typické chování (Obr. 7.4) vykazují obrázky fruits.png, redmagic.png či tulips.png, oproti tomu se atypicky chová obrázek aerials.png, kde byl z pohledu velikosti ukládaného souboru nejvýhodnější barevný systém RGB, následovaný barevnými systémy YCbCr, L*u*v a HSV.

7.2.3 Další možná hodnotící kritéria

Závislost PSNR na počtu vrcholů triangulace

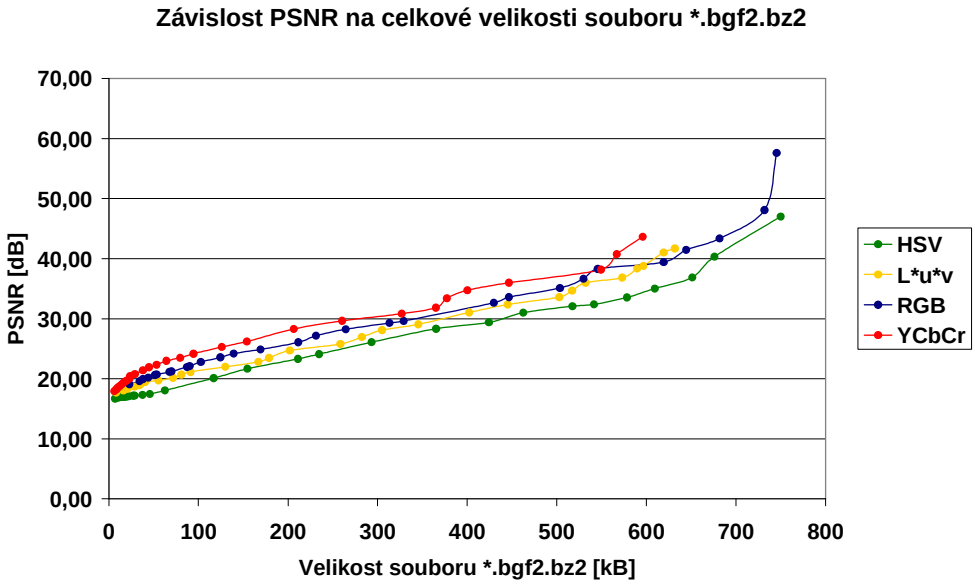
Z křivek jednotlivých barevných systémů znázorňujících závislost velikosti PSNR na počtu vrcholů N (Obr. 7.5) vyplývá, že při velkém počtu vrcholů v triangulaci (nad 200.000) je z hlediska PSNR závislého na počtu vrcholů nejvýhodnější barevný systém RGB. Pro menší počet vrcholů v triangulaci je s RGB srovnatelný jak systém YCbCr tak i L*u*v. Systém HSV je s ostatními porovnatelný pouze na úseku vysokého počtu vrcholů v triangulaci, v ostatních případech je víceméně nepoužitelný. Typické chování (Obr. 7.5) vykazují obrázky airplane.png, baboon.png, lena.png, peppers.png či redmagic.png, oproti tomu se atypicky chová obrázek aerials.png, girl.png či woman.png, kde byl z pohledu kvality PSNR na počtu vrcholů triangulace nejvýhodnější barevný systém HSV, následovaný barevnými systémy L*u*v, RGB a YCbCr.



Obr. 7.5 Na tomto obrázku (lena.png) je ukázán typický vliv jednotlivých barevných systémů na výslednou kvalitu obrázku (PSNR) v závislosti na počtu vrcholů triangulace [zdroj: vlastní].

Z pohledu závislosti PSNR na velikosti jednotlivých souborů (*.dat, *.bgf2, *.raw, *.bgf2.bz2, *.raw.bz2) je situace naprosto stejná. Obr. 7.6 znázorňuje závislost hodnoty PSNR na velikosti souboru *.bgf2.bz2. Chování PSNR v závislosti na velikosti ostatních souborů (*.dat, *.bgf2, *.raw, *.raw.bz2) je víceméně shodné s chováním PSNR v závislosti na velikosti souboru *.bgf2.bz2, proto je typické chování ukázáno pouze na případu, kde dochází k nejlepšímu kompresnímu poměru původní triangulace *.dat. Z obrázku je patrné, že pro malé velikosti souboru (nízký počet vrcholů v triangulaci) se barevný systém příliš neprojevuje. Pro vyšší počet vrcholů vychází z pohledu PSNR nejlépe barevný systém YCbCr, následovaný barevnými systémy RGB

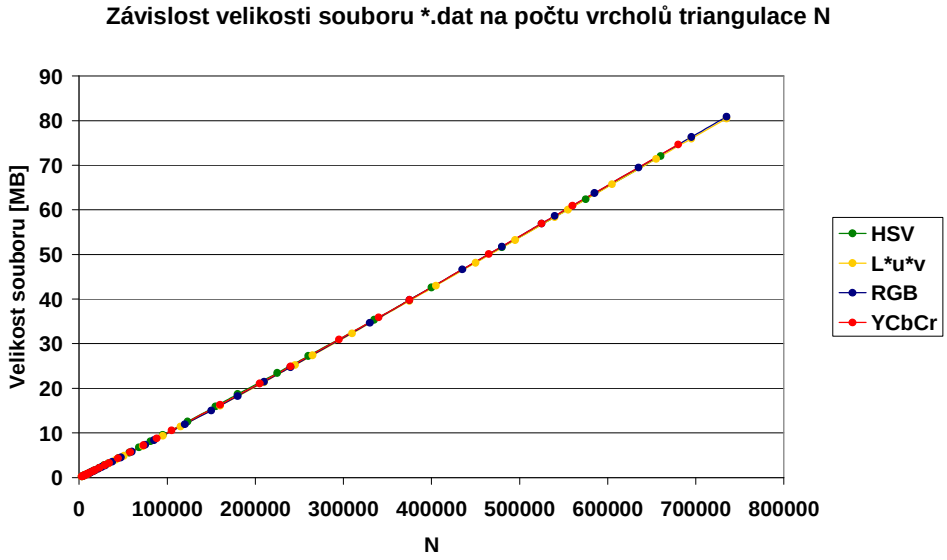
a L*u*v. Systém HSV za ostatními opět zaostává o cca 5dB. Typické a atypické chování obrázků je stejné jako u závislosti PSNR/N.



Obr. 7.6 Charakteristiky tohoto obrázku (baboon.png) znázorňují typické chování vlivu jednotlivých barevných systémů na PSNR v závislosti na velikosti souboru *.bgf2.bz2 [zdroj: vlastní].

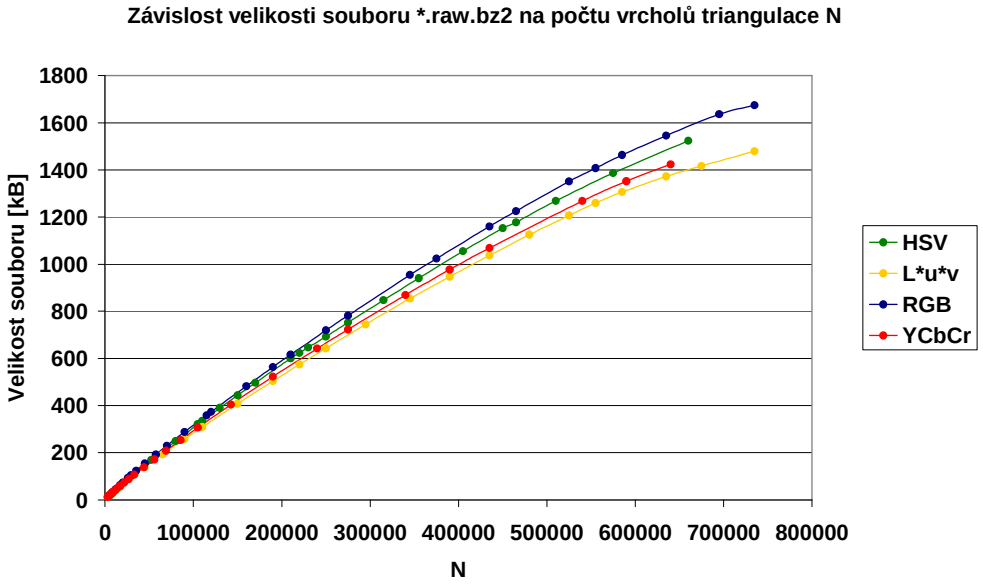
Závislost velikosti souborů na počtu vrcholů triangulace

Závislost velikosti souboru *.dat (Obr. 7.7), obsahujícího nezakódovanou a nekomprimovanou triangulaci, na počtu vrcholů triangulace je lineární pro všechny barevné systémy (HSV, L*u*v, RGB, YCbCr). Typické chování (Obr. 7.7) vykazují např. obrázky airplane.png, či fruits.png. Tento ukazatel je pro určení výhodnosti barevného systému bezvýznamný stejně tak jako závislost velikosti zakódovaných triangulací jednou z metod (FVXPATH, RAW) na počtu vrcholů triangulace. Zde vykazují typické chování např. obrázky baboon.png či car.png.

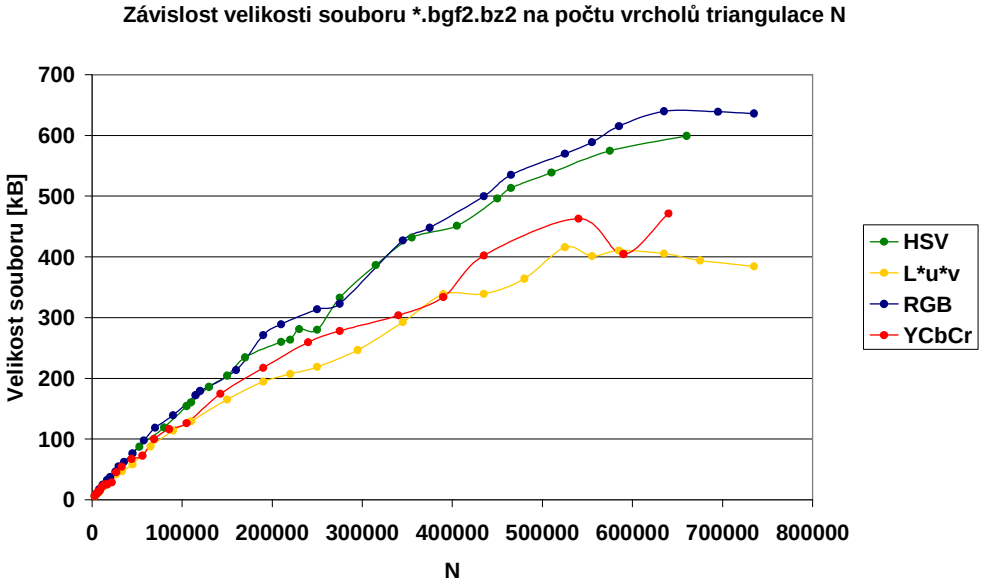


Obr. 7.7 Charakteristiky tohoto obrázku (airplane.png) znázorňují typické chování vlivu jednotlivých barevných systémů na velikost souboru *.dat v závislosti na počtu vrcholů triangulace. Pro soubory *.bgf2 a *.raw je graf analogický, pouze velikosti jednotlivých souborů jsou odlišné, nicméně vliv barevných systémů je stejný [zdroj: vlastní].

Porovnání závislosti velikosti souborů komprimovaných (již zakódovaných) triangulací na počtu vrcholů triangulace je z pohledu vlivu barevného systému také zanedbatelné, nicméně nepatrně výhodněji vychází použití barevného systému L*u*v a YCbCr. Barevné systémy HSV a RGB jsou z tohoto ohledu méně výhodné. Při malém počtu vrcholů (cca 100.000 a méně) je vliv barevného systému bezvýznamný. Při vyšším počtu vrcholů (okolo 300.000) se již začíná výhodnost různých barevných systémů projevovat. Pro vysoký počet vrcholů (nad 300.000) se začíná vliv barevných systémů projevovat jak u metody RAW (Obr. 7.8) tak u metody FVXPATH (Obr. 7.9), nicméně v případě druhé jmenované metody je tento vliv o poznání výraznější. U obou metod kódování vykazují typické chování např. obrázky baboon.png, fruits.png či redmagic.png.

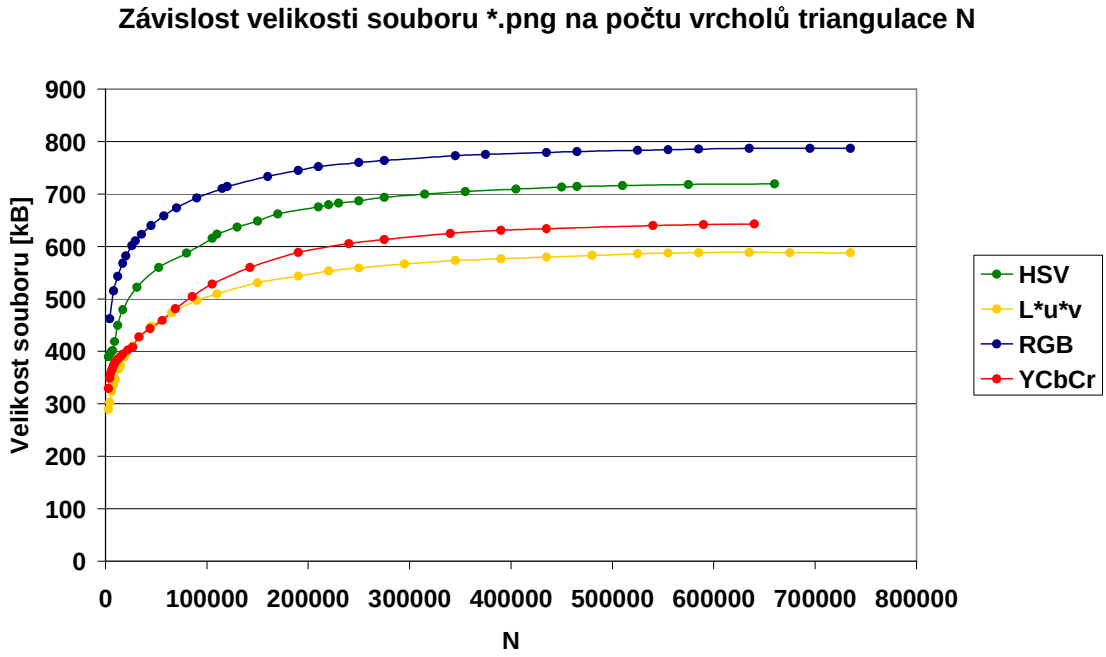


Obr. 7.8 Charakteristiky tohoto obrázku (fruits.png) znázorňují typické chování vlivu jednotlivých barevných systémů na velikost souboru *.raw.bz2 (ukládajícího triangulaci obrázku v zakódované a následně komprimované podobě) v závislosti na počtu vrcholů triangulace [zdroj: vlastní].



Obr. 7.9 Charakteristiky tohoto obrázku (fruits.png) znázorňují typické chování vlivu jednotlivých barevných systémů na velikost souboru *.bgf2.bz2 (ukládajícího triangulaci obrázku v zakódované a následně komprimované podobě) v závislosti na počtu vrcholů triangulace [zdroj: vlastní].

Z hlediska závislosti velikosti výstupního souboru (rastrového obrázku ve formátu PNG) na počtu vrcholů v triangulaci (Obr. 7.10) jsou opět na celém úseku, bez ohledu na počet vrcholů, zcela jasně nejvýhodnější systémy L*u*v a YCbCr. Systém HSV a RGB vychází z porovnávané čtveřice opět nejhůře. Křivky této závislosti mají konkávní tvar. Typické chování (Obr. 7.10) vykazují např. obrázky baboon.png, či redmagic.png.



Obr. 7.10 Charakteristiky tohoto obrázku (fruits.png) znázorňují typické chování vlivu jednotlivých barevných systémů na velikost ukládaného rastrového souboru *.png v závislosti na počtu vrcholů triangulace [zdroj: vlastní].

7.2.4 Celkové zhodnocení

Při rozhodování, který barevný systém je nejvýhodnější, nelze dát jednoznačnou odpověď. Je zde mnoho faktorů, které mohou při takovémto rozhodování hrát významnou roli, ať už je to kvalita výsledného obrazu (PSNR), velikost rastrových obrazů nebo jednotlivých souborů.

Pokud ale vezmeme všechna tato kritéria dohromady, tak nám jako nejlepší kombinace vycházejí barevné systémy L*u*v a YCbCr. Během celého testování vykazovaly nejlepší výsledky, co se velikosti jednotlivých souborů týče, ale také výsledné kvality a kompresního poměru (jak při použití samotného kódování, tak při použití kódování a následné komprese takto uložených souborů), nicméně kompresní poměr u barevného systému L*u*v je z testované čtveřice nejlepší. Dále jako nejvýhodnější vychází barevný systém RGB, který vykazuje velmi dobré výsledky ohledně PSNR, ale špatné výsledky v kategorii velikosti jednotlivých souborů a komprese ho řadí až na třetí místo. HSV je v porovnání s ostatními nejméně výhodný.

Na základě dosažených výsledků dále vychází lépe metoda kódování FVXPATH oproti metodě RAW. Při použití komprese na takto zakódovanou triangulaci je tento rozdíl ještě výraznější.

Shrnutí

Pro kompresi digitálních obrazů za použití triangulace je nejvýhodnější zvolit barevný systém L*u*v (případně YCbCr), metodu kódování FVXPATH a následně takto zakódovanou triangulaci komprimovat dostupnými metodami komprese dat.

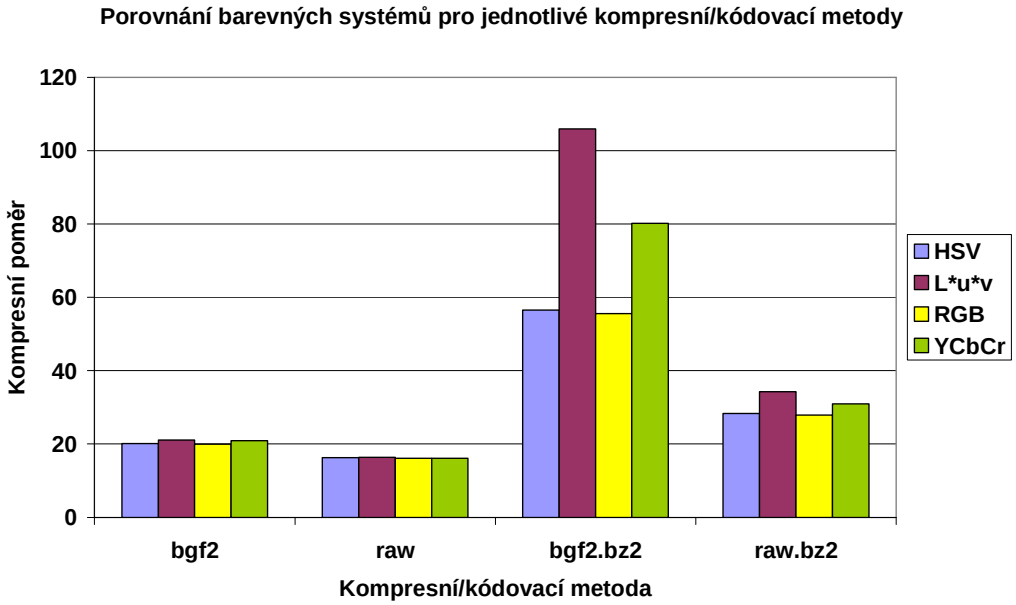
Jednotlivé typy obrázků (uměle vytvořené, fotografie) se chovají obdobně. U uměle vytvořených obrázků je mnohem méně pestrosti, než u reálných obrázků (fotografií). Např. pro barevný systém YCbCr je u uměle vytvořených obrázků (např. redmagic.png či tulips.png) dobře vidět, že ani vysoká ztráta bodů v kanálech Cb a Cr nemá moc velký vliv na výslednou kvalitu obrazu. Ztráta barvy není až tak podstatná jako ztráta jasu.

7.3 Kotriangulace

Tato kapitola se věnuje zhodnocení kvality jednotlivých obrázků, kde jsou jednotlivé barevné komponenty zpracovávány odděleně jako tři nezávislé šedotónové obrazy, ale vzniklé triangulace jsou následně kombinovány dohromady do jedné kotriangulace. Opět bylo experimentováno s různými barevnými systémy a byla provedena příslušná zhodnocení.

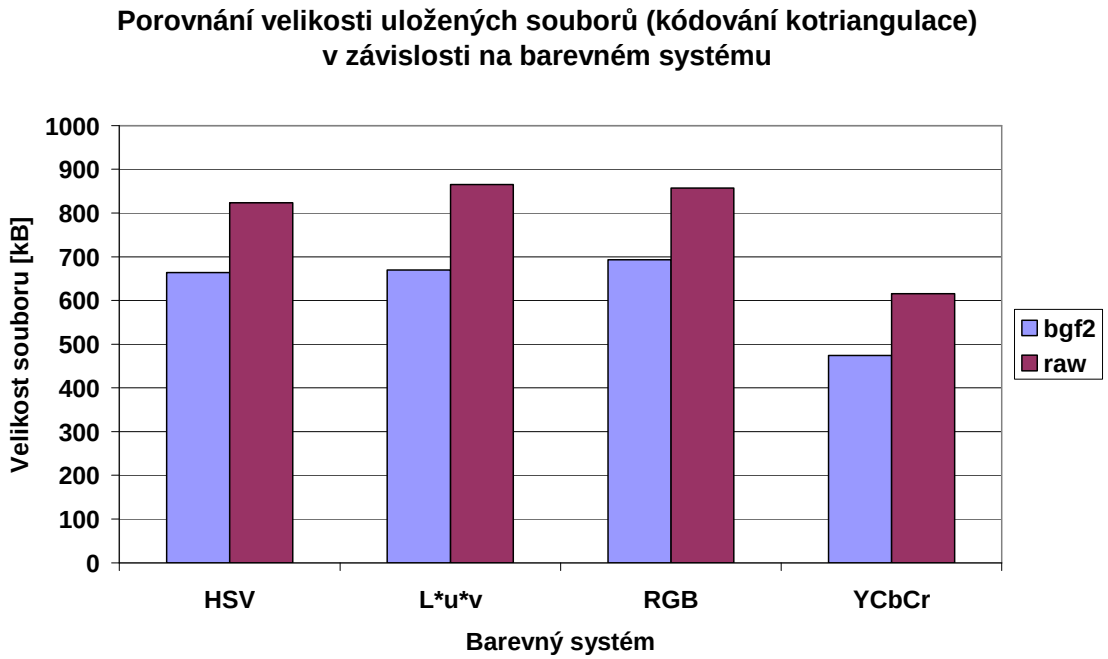
7.3.1 Vliv barevného systému na kotriangulace

Experimenty s různými obrázky ukázaly, že při porovnání námi zvolených metod kódování kotriangulací (bgf2 - FVXPATH, raw - RAW) vychází lépe metoda FVXPATH (stejné jako u triangulací). Vliv barevného systému je v tomto porovnání opět nezajímavý. Odchylky v kompresním poměru zakódovaných vůči nezakódovaným kotriangulacím jsou v rámci jedné metody minimální (Obr. 7.11), tedy zanedbatelné. Typické chování vykazuje např. obrázek fruits.png, vyloženě odlišné chování nevykazoval žádný z testovaných obrázků.



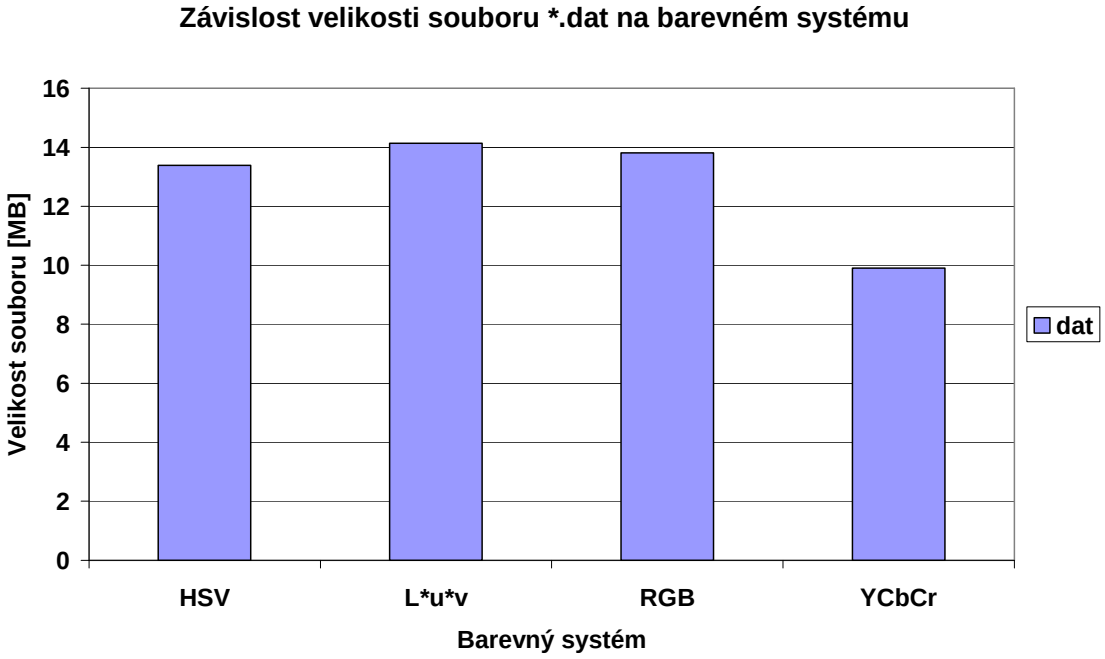
Obr. 7.11 Charakteristiky tohoto obrázku (fruits.png) znázorňují typické chování vlivu jednotlivých barevných systémů na typ kódování (skupina bgf2 a raw) a vliv jednotlivých barevných systémů na kompresi takto zakódovaných kotriangulací (skupina bgf2.bz2 a raw.bz2), ve všech případech je kompresní poměr porovnáván vzhledem k základní kotriangulaci [zdroj: vlastní].

Jiná situace nastává při porovnávání velikosti jednotlivých souborů (zakódovaných kotriangulací) uložených na disku (Obr. 7.12). Zde už je vliv barevných systémů poměrně znatelný. Jednoznačně nejlepším je z tohoto ohledu barevný systém YCbCr. Kotriangulace kódovány jak metodou FVXPATH, tak i metodou RAW, jsou při použití tohoto barevného systému a při následném ukládání na disk v podobě souborů nejmenší. Ostatní barevné systémy (HSV, L*u*v a RGB) vycházejí víceméně stejně, nicméně v porovnání s YCbCr nepatrně zaostávají. Typické chování (Obr. 7.12) vykazují např. obrázky fruits.png a baboon.png, oproti tomu se atypicky choval obrázek drop.png. Velikosti kotriangulací zakódovaných metodou RAW jsou přibližně 1.3x větší oproti metodě FVXPATH.



Obr 7.12 Na tomto obrázku (fruits.png) je ukázán vliv jednotlivých barevných systémů na velikost ukládaného souboru (zakódované kotriangulace) [zdroj: vlastní].

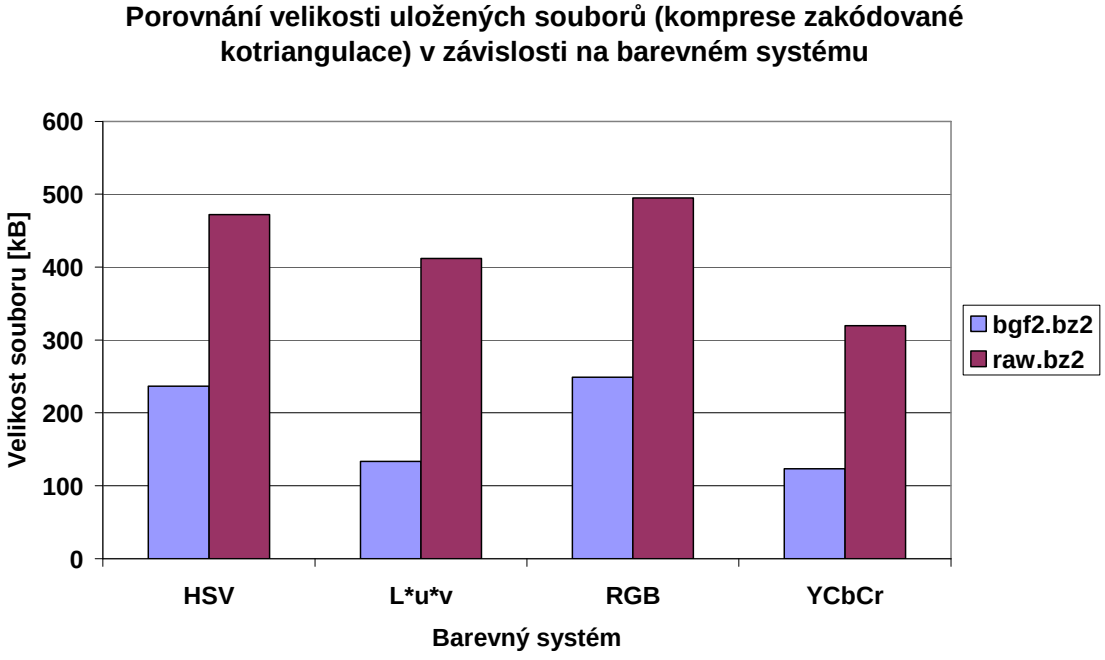
Při porovnávání velikosti jednotlivých souborů uložených na disku v nezakódované podobě (Obr 7.13), tedy v podobě kotriangulace, vychází nejlépe opět YCbCr, následovaný barevným systémem HSV, RGB a L*u*v. Typické chování (Obr. 7.13) vykazují např. obrázky fruits.png a baboon.png, atypicky se choval obrázek car.png, kde byl nejvýhodnější barevný systém L*u*v, následovaný barevnými systémy YCbCr, RGB a HSV.



Obr. 7.13 Obrázek (fruits.png) znázorňuje nejčastější vliv barevného systému na velikost ukládaného souboru s kotriangulací *.dat [zdroj: vlastní].

7.3.2 Vliv barevného systému na kompresi již zakódované kotriangulace

K výrazným změnám dochází při použití komprese na již zakódované kotriangulace pomocí výše popsaných metod (FVXPATH a RAW). Metoda FVXPATH vychází opět mnohem lépe než metoda RAW. Vliv barevného systému začíná hrát svou významnou roli jak při porovnání kompresních poměrů (Obr. 7.11), tak při porovnání velikostí jednotlivých uložených souborů na disku (Obr. 7.14).



Obr. 7.14 Na tomto obrázku (fruits.png) je ukázán vliv jednotlivých barevných systémů na velikost ukládaného souboru (komprimované již zakódované kotriangulace) [zdroj: vlastní].

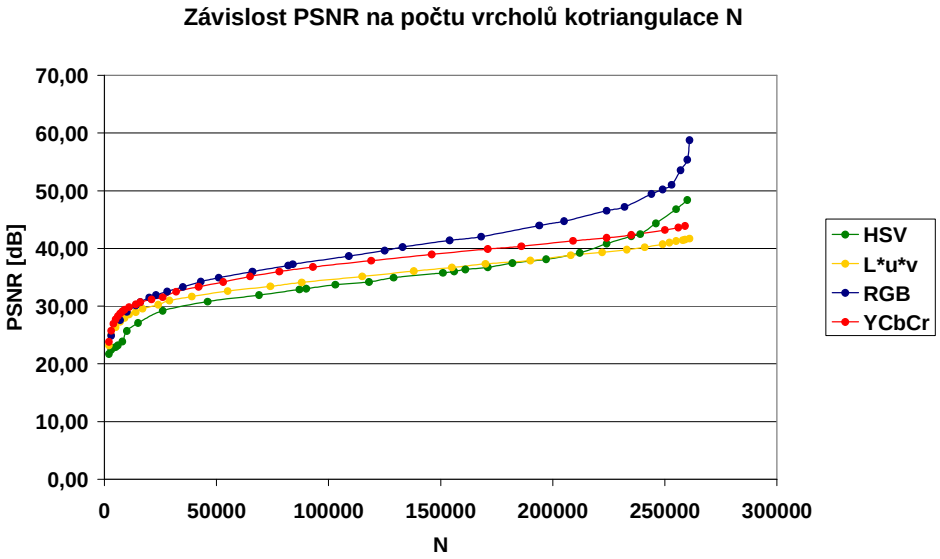
Kompresní poměr je nejlepší při použití barevného systému L*u*v. Takto zakódované a následně komprimované soubory vykazují v průměru kompresní poměr cca. 105 vůči originální kotriangulaci a kompresní poměr cca 5.2 vůči již zakódované kotriangulaci. Při použití barevného systému YCbCr získáme o něco horší, nicméně také velmi dobré výsledky. Kompresní poměr je cca. 80 vůči originální kotriangulaci a 4.1 vůči již zakódované kotriangulaci. Barevné systémy RGB a HSV vykazují navzájem podobné výsledky, ale za barevnými systémy L*u*v a YCbCr výrazně zaostávají, co se kompresních poměrů týče (cca 60 vůči originální kotriangulaci a 2.9 vzhledem k již zakódované kotriangulaci). Typické chování (Obr. 7.11) vykazují obrázky baboon.png, fruits.png či redmagic.png, oproti tomu se atypicky chová obrázek pool.png, kde byl z pohledu kompresního poměru nejvýhodnější barevný systém RGB, následovaný barevnými systémy L*u*v, HSV a YCbCr.

Znatelně se na velikosti jednotlivých souborů (zakódovaných a následně komprimovaných kotriangulací) uložených na disku podepisuje jak vliv barevného systému (Obr. 7.14), tak vliv použité kódovací metody. Velikost výsledných souborů získaných metodou kódování RAW a následně komprimovaných algoritmem bzip2 jsou přibližně 2.5x větší než při použití metody kódování FVXPATH a následné kompresi algoritmem bzip2. Jednoznačně nejlepším je z tohoto ohledu barevný systém YCbCr. Kotriangulace kódovány jak metodou FVXPATH, tak i metodou RAW, které byly následně komprimovány, jsou při použití tohoto barevného systému a při následném ukládání na disk v podobě souborů nejmenší. Dále jako nejvýhodnější vychází barevný systém L*u*v. Za těmito barevnými systémy zaostávají (co se velikosti jednotlivých souborů týče) barevné systémy HSV a RGB. Typické chování (Obr. 7.14) vykazují obrázky fruits.png, redmagic.png či tulips.png, oproti tomu se atypicky chová obrázek yacht.png, kde byl z pohledu velikosti ukládaného souboru nejvýhodnější barevný systém RGB, následovaný barevnými systémy L*u*v, YCbCr a HSV.

7.3.3 Další možná hodnotící kritéria

Závislost PSNR na počtu vrcholů kotriangulace

Z křivek jednotlivých barevných systémů znázorňujících závislost velikosti PSNR na počtu vrcholů kotriangulace N (Obr. 7.15) vyplývá, že při velkém počtu vrcholů v kotriangulaci (nad 50000) je z hlediska PSNR závislého na počtu vrcholů nejvýhodnější barevný systém RGB. Pro menší počet vrcholů v kotriangulaci je s RGB srovnatelný jak systém YCbCr, tak i L*u*v. Systém HSV je s ostatními porovnatelný pouze na úseku vysokého počtu vrcholů. Typické chování (Obr. 7.15) vykazují obrázky baboon.png, lena.png či tulips.png, oproti tomu se atypicky chová obrázek aerials.png, girl.png či woman.png, kde byl z pohledu kvality PSNR na počtu vrcholů kotriangulace nejvýhodnější barevný systém RGB, následovaný barevnými systémy HSV, L*u*v a YCbCr.

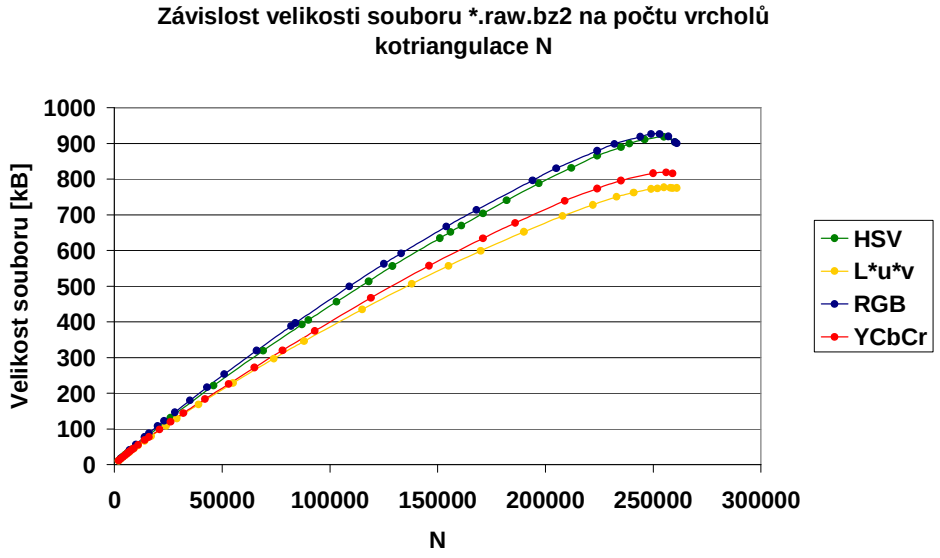


Obr. 7.15 Na tomto obrázku (fruits.png) je ukázán typický vliv jednotlivých barevných systémů na výslednou kvalitu obrázku (PSNR) v závislosti na počtu vrcholů kotriangulace [zdroj: vlastní].

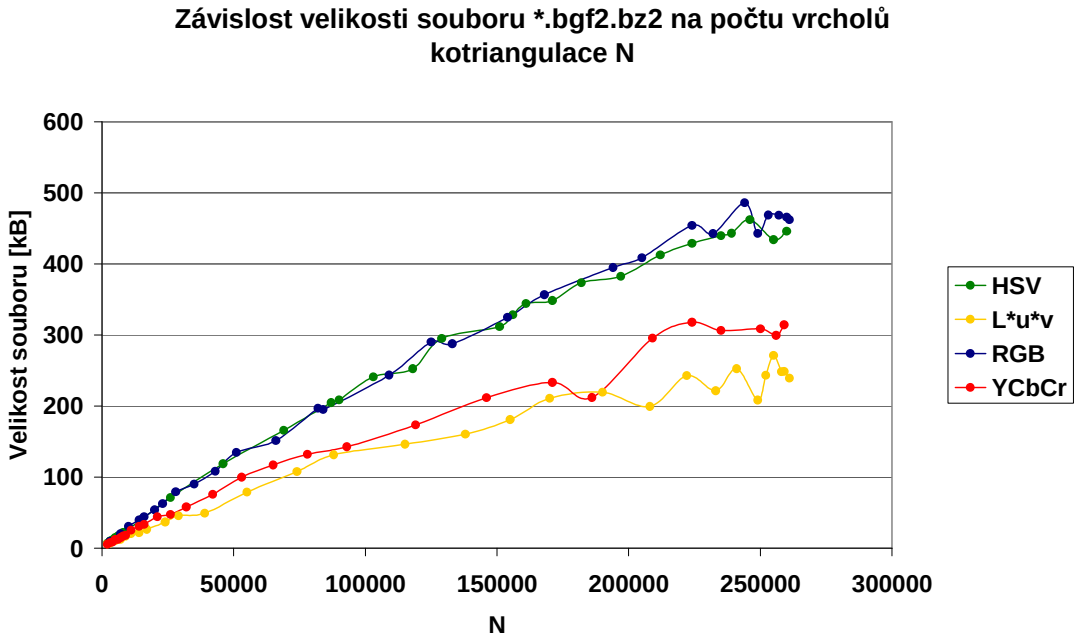
Závislost velikosti souborů na počtu vrcholů kotriangulace

Závislost velikosti kotriangulace (soubor *.dat) na počtu vrcholů této kotriangulace (podobný obrázek viz triangulace Obr. 7.7) je lineární pro všechny barevné systémy (HSV, L*u*v, RGB, YCbCr). Tento ukazatel je pro určení výhodnosti barevného systému bezvýznamný, stejně tak jako závislost velikosti zakódovaných kotriangulací jednou z metod (FVXPATH, RAW) na počtu vrcholů kotriangulace.

U porovnání závislosti velikostí souborů komprimovaných (již zakódovaných) kotriangulací na počtu vrcholů kotriangulace vychází nejvýhodněji použití barevných systémů L*u*v a YCbCr. Systémy HSV a RGB jsou z tohoto ohledu méně výhodné. Při malém počtu vrcholů (okolo 50000) je vliv barevného systému bezvýznamný. Pro vyšší počet vrcholů kotriangulace je tento vliv výrazný při použití obou metod kódování: RAW (Obr. 7.16), FVXPATH (Obr. 7.17).

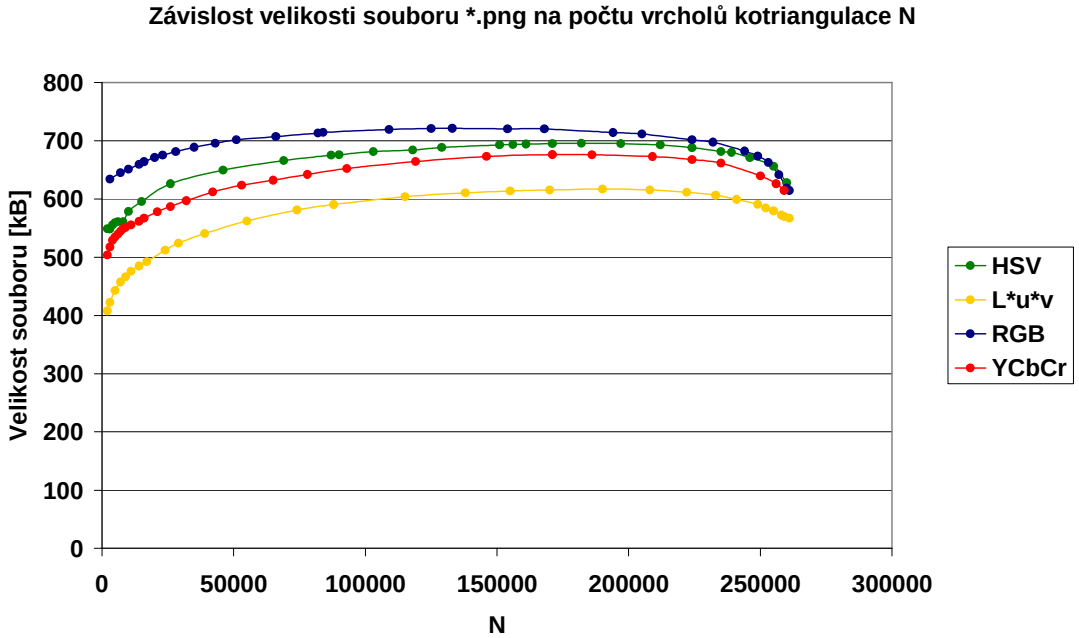


Obr. 7.16 Charakteristiky tohoto obrázku (fruits.png) znázorňují typické chování vlivu jednotlivých barevných systémů na velikost souboru *.raw.bz2 (ukládajícího kotriangulaci obrázku v zakódované a následně komprimované podobě) v závislosti na počtu vrcholů kotriangulace [zdroj: vlastní].



Obr. 7.17 Charakteristiky tohoto obrázku (fruits.png) znázorňují typické chování vlivu jednotlivých barevných systémů na velikost souboru *.bgf2.bz2 (ukládajícího kotriangulaci obrázku v zakódované a následně komprimované podobě) v závislosti na počtu vrcholů kotriangulace [zdroj: vlastní].

Z hlediska závislosti velikosti výstupního souboru (rastrového obrázku ve formátu PNG) na počtu vrcholů kotriangulace (Obr. 7.18) jsou opět na celém úseku, bez ohledu na počet vrcholů, zcela jasně nejvýhodnější systémy L*u*v a YCbCr. Systém RGB a HSV vychází z porovnávané čtveřice opět nejhůře. Typické chování (Obr. 7.18) vykazují např. obrázky baboon.png či fruits.png.



Obr. 7.18 Charakteristiky tohoto obrázku (fruits.png) znázorňují typické chování vlivu jednotlivých barevných systémů na velikost ukládaného rastrového souboru *.png v závislosti na počtu vrcholů kotriangulace [zdroj: vlastní].

7.3.4 Celkové zhodnocení

Při rozhodování, který barevný systém je nejvýhodnější, opět nelze dát jednoznačnou odpověď. Je zde mnoho faktorů, které mohou při takovémto rozhodování hrát významnou roli, ať už je to kvalita výsledného obrazu (PSNR), velikost rastrových obrazů nebo jednotlivých souborů.

Pokud ale vezmeme všechna tato kritéria dohromady, jako nejlepší kombinace vycházejí barevné systémy YCbCr a L*u*v. Během celého testování vykazovaly nejlepší výsledky, co se velikosti jednotlivých souborů týče, ale také výsledné kvality a kompresního poměru (jak při použití samotného kódování, tak při použití kódování a následné komprese takto uložených souborů), nicméně kompresní poměr u barevného systému L*u*v je z testované čtveřice nejlepší. Dále jako nejvýhodnější vychází barevný systém RGB, který vykazuje velmi dobré výsledky ohledně PSNR, ale špatné výsledky v kategorii velikosti jednotlivých souborů a komprese ho řadí až na třetí místo. HSV je v porovnání s ostatními nejméně výhodný.

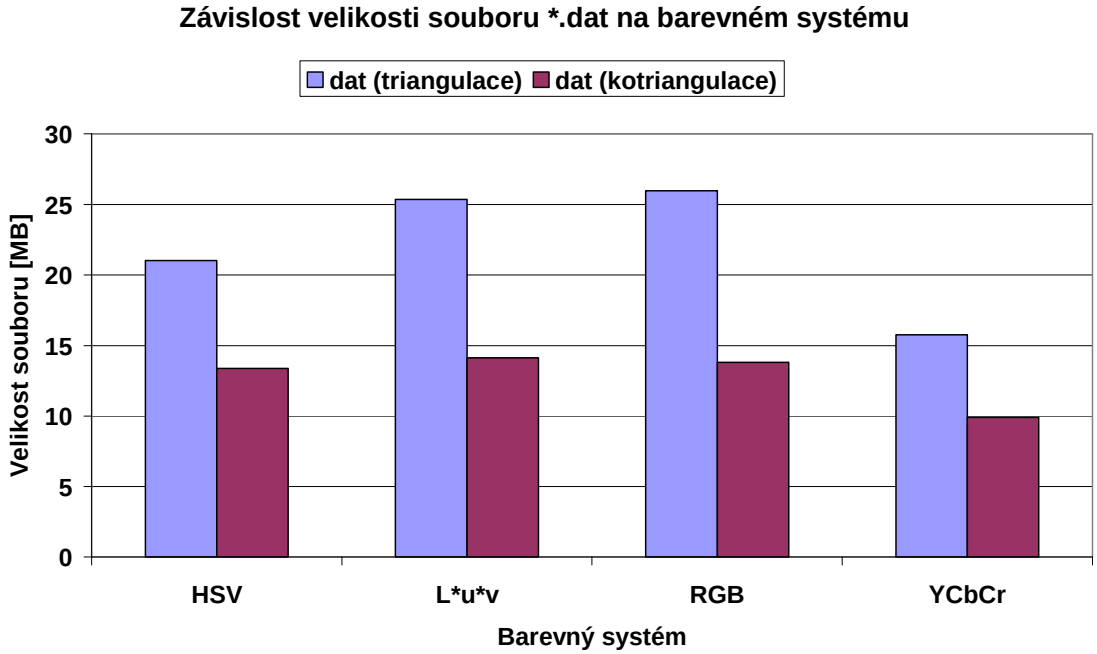
Na základě dosažených výsledků dále vychází lépe metoda kódování FVXPATH oproti metodě RAW. Při použití komprese na takto zakódovanou kotriangulaci je tento rozdíl ještě výraznější.

Shrnutí

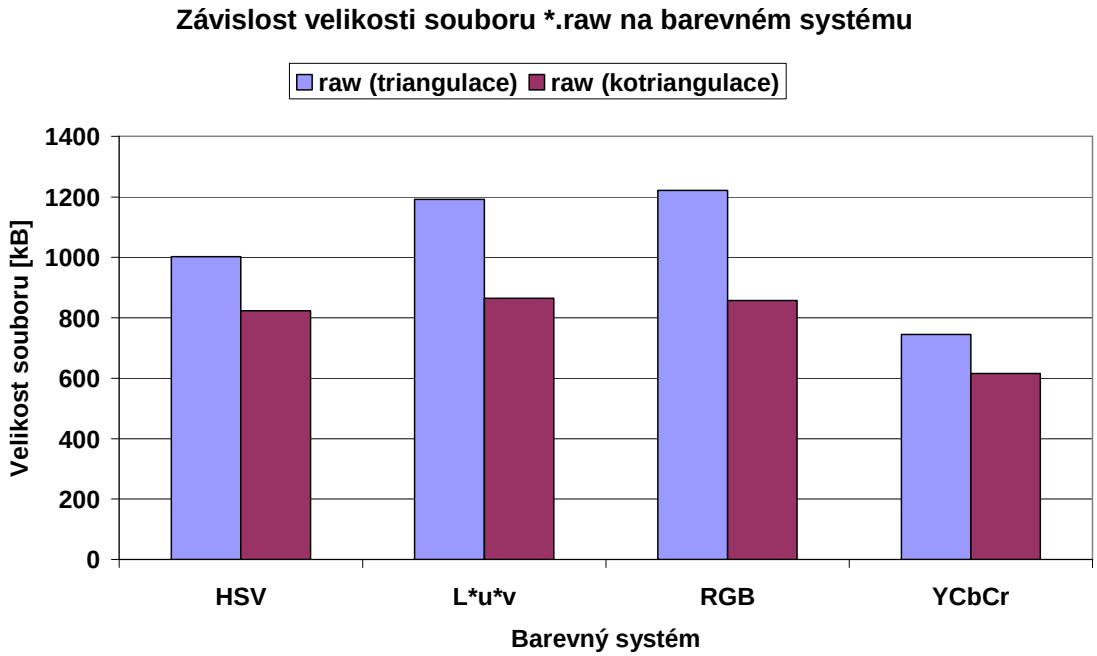
Pro kompresi digitálních obrazů za použití kotriangulace je nejvýhodnější zvolit barevný systém L*u*v (případně YCbCr), metodu kódování FVXPATH a následně takto zakódovanou kotriangulaci komprimovat dostupnými metodami komprese dat.

7.4 Porovnání dosažených výsledků

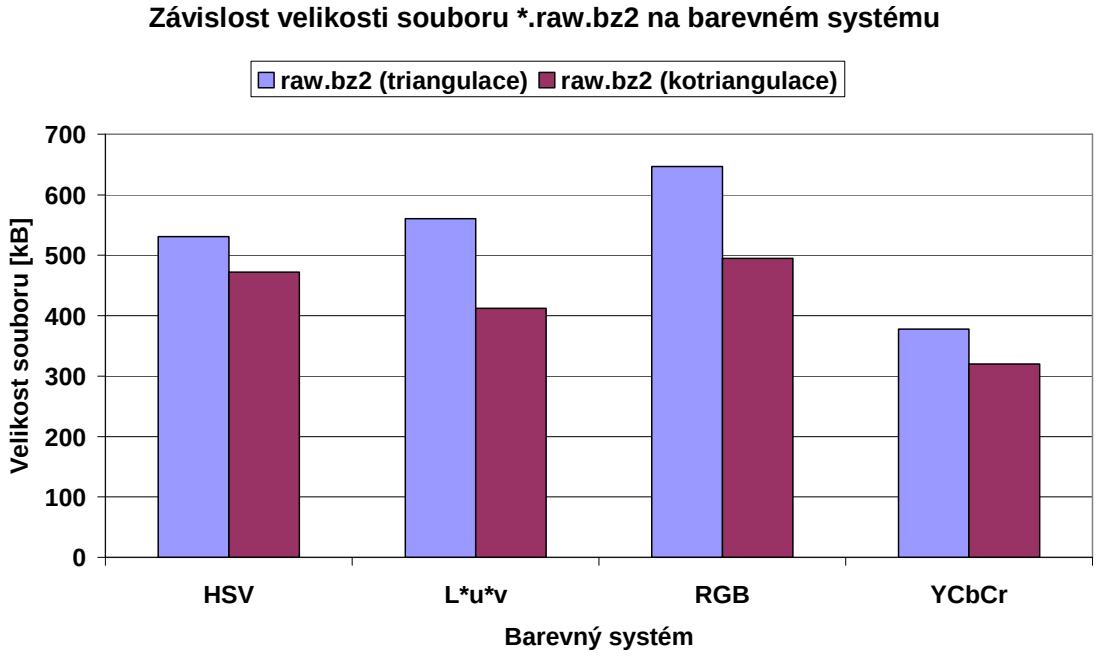
Jak u komprese digitálních barevných obrazů pomocí triangulace, tak i u komprese digitálních barevných obrazů pomocí kotriangulace bylo dosaženo velice podobných výsledků, co se týče výhodnosti barevného systému, metody kódování a následné komprese dat. Jediné, co může převážit ručičku vah ve prospěch triangulací či kotriangulací, je velikost výstupních datových souborů. Toto porovnání vyznívá poměrně jednoznačně ve prospěch kotriangulací, viz následující obrázky (Obr. 7.19, Obr. 7.20, Obr. 7.21, Obr. 7.22 a Obr. 7.23). Jedná se o obrázky, kde jsou porovnány velikosti jednotlivých datových souborů z pohledu triangulací a kotriangulací. Na první pohled je zřejmé, že komprese digitálních barevných obrazů pomocí kotriangulací uspoří na disku, v případě nezakódovaných a nekomprimovaných dat (Obr. 7.19), až několik MB oproti triangulacím. U ostatních případů (porovnání triangulací s kotriangulacemi z pohledu datových souborů *.raw, *.raw.bz2 a *.bgf2, *.bgf2.bz2) je situace obdobná, pouze úspora se pohybuje řádově v kB.



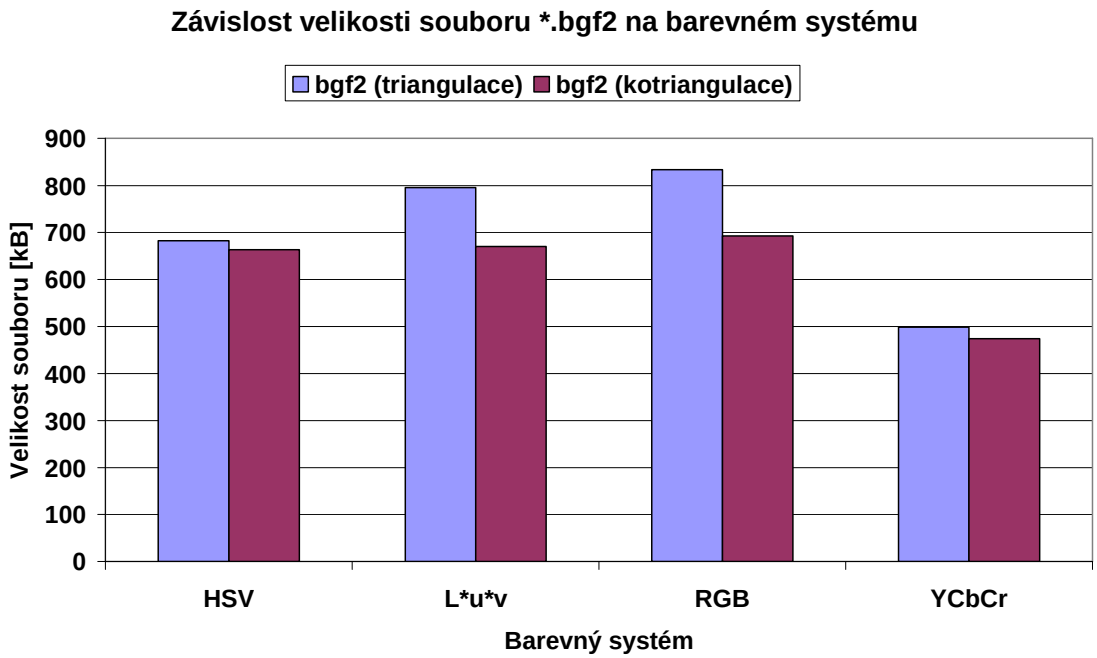
Obr. 7.19 Charakteristiky tohoto obrázku (fruits.png) znázorňují typické chování vlivu jednotlivých barevných systémů na velikost ukládaného souboru *.dat [zdroj: vlastní].



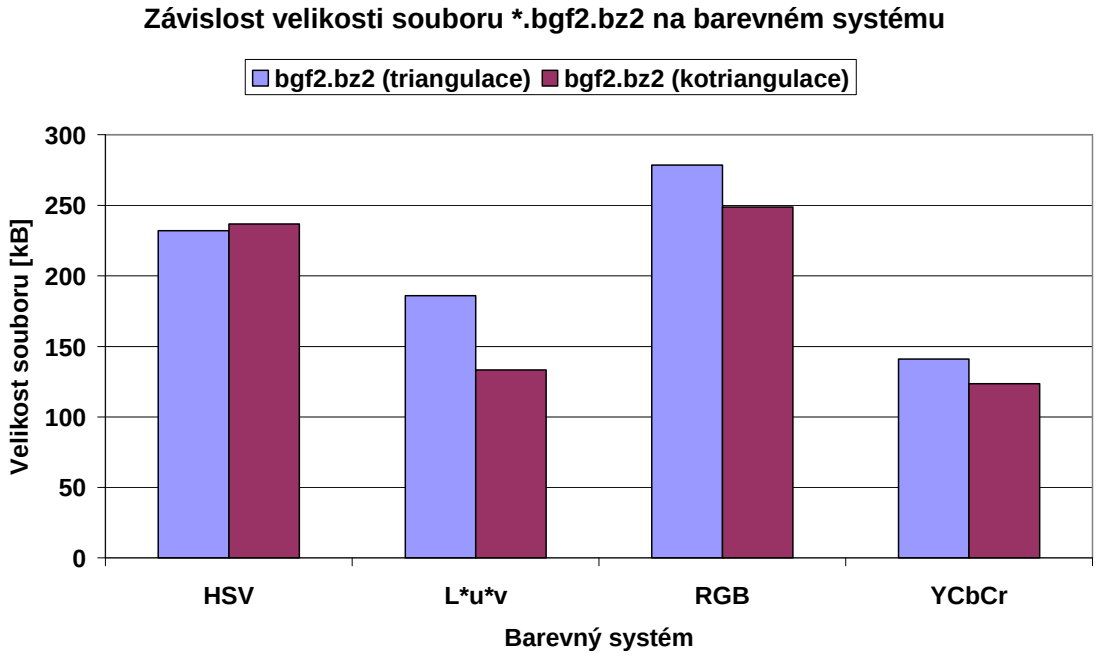
Obr 7.20 Na tomto obrázku (fruits.png) je ukázán vliv jednotlivých barevných systémů na velikost ukládaného souboru (zakódované triangulace/kotriangulace) [zdroj: vlastní].



Obr. 7.21 Na tomto obrázku (fruits.png) je ukázán vliv jednotlivých barevných systémů na velikost ukládaného souboru (komprimované již zakódované triangulace/kotriangulace) [zdroj: vlastní].

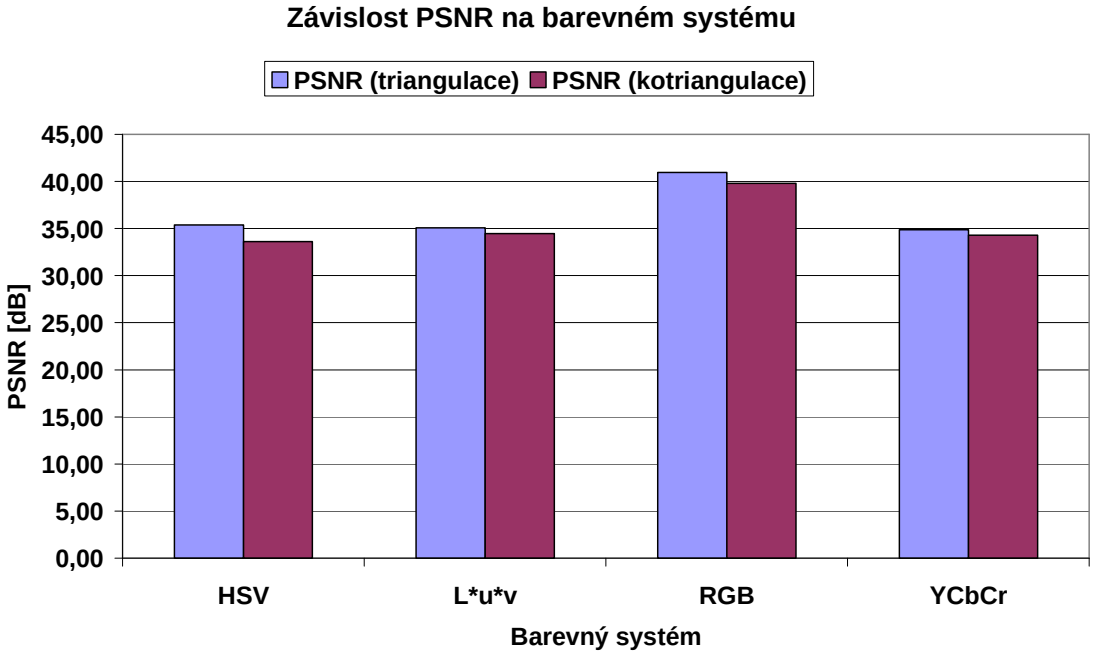


Obr 7.22 Na tomto obrázku (fruits.png) je ukázán vliv jednotlivých barevných systémů na velikost ukládaného souboru (zakódované triangulace/kotriangulace) [zdroj: vlastní].



Obr. 7.23 Na tomto obrázku (fruits.png) je ukázán vliv jednotlivých barevných systémů na velikost ukládaného souboru (komprimované již zakódované triangulace/kotriangulace) [zdroj: vlastní].

Dalším možným měřítkem výhodnosti triangulací či kotriangulací může být objektivní měření vizuální kvality komprese barevného digitálního obrazu pomocí PSNR. Obr. 7.24 ukazuje, že triangulace vycházejí o něco lépe, nicméně kotriangulace zabraňují roztékání barev, což je občas nežádoucím jevem v případě použití triangulací.



Obr. 7.24 Na tomto obrázku (fruits.png) je ukázán typický vliv jednotlivých barevných systémů na výslednou kvalitu obrázku (PSNR) [zdroj: vlastní].

Shrnutí

Pro kompresi digitálních barevných obrazů je výhodnější použít kotriangulace namísto triangulací. Díky kotriangulacím nedochází k roztékání barev v komprimovaném obrázku, navíc lze za použití kotriangulací dosáhnout menších výstupních souborů (cca 1.2x menší soubory) než v případě použití tří triangulací jednotlivých barevných složek použitého barevného systému. Z hlediska barevného systému je nejvýhodnější použít L*u*v (případně YCbCr), metodu kódování FVXPATH a následně takto zakódovanou kotriangulaci komprimovat dostupnými metodami komprese dat (např. bzip2).

8. Závěr

Cílem této práce byla analýza komprese digitálních barevných obrazů s využitím triangulací. Na základě metod pro reprezentaci šedotónových obrázků triangulací a metod pro kompresi těchto triangulací bylo vyvinuto několik utilit, které jsou schopny reprezentovat vstupní barevný digitální obraz triangulací, případně kotriangulací, a takto vzniklá data následně kódovat a komprimovat do úspornější podoby.

K analýze dat, vzniklých reprezentací obrazových dat pomocí tří nezávislých triangulací pro jednotlivé barevné složky, vznikl software, který umí tato data vyhodnotit a poskytnout statistický soubor, potřebný k podrobné analýze výhodnosti tohoto přístupu.

K analýze dat, vzniklých reprezentací obrazových dat pomocí tří nezávislých triangulací pro jednotlivé barevné složky, které byly následně kombinovány dohromady do jedné kotriangulace, vznikl obdobný software, který vyhodnocuje a poskytuje statistický soubor, potřebný k podrobné analýze výhodnosti tohoto přístupu.

Během práce bylo pracováno s několika významnými barevnými systémy (HSV, L*u*v, RGB, YCbCr), které se nemalou částí podílejí na dosaženém kompresním poměru a vizuální kvalitě výsledného komprimovaného obrazu pomocí triangulací/kotriangulací. Popis převodu mezi jednotlivými barevnými systémy a práce s nimi, který je mimo jiné popsán v této bakalářské práci, může užitečně posloužit všem, kteří se o tuto problematiku zajímají.

V závěru práce (kapitola č.7) byla analýzou výsledků zjištěna optimální kombinace barevného systému, reprezentace obrazových dat, typu kódování a komprese výsledných dat.

Použitá literatura

[1] FUKSA, M. *Delaunayova triangulace s omezením (CDT) v E^2 a E^3*. Plzeň, 2006. Diplomová práce na fakultě aplikovaných věd na Západočeské univerzitě v Plzni na katedře informatiky a výpočetní techniky. Vedoucí diplomové práce Doc. Dr. Ing. Ivana Kolingerová.

[2] HUNT, R. W.G. *The reproduction of colour*. 6. vyd. John Wiley & Sons, 2004. ISBN 0-470-02425-9

[3] KOHOUT, J. *Alternative Representation of Image Information*. Plzeň: University of West Bohemia, dosud nevydáno. Technical Report.

[4] MÍKOVEC, P. *Doporučení pro používání barev*. Plzeň, 2006. Projekt5 na fakultě aplikovaných věd na Západočeské univerzitě v Plzni na katedře informatiky.

[5] MURRAY, J.D. *Encyklopedie grafických formátů*. 2. vyd. Praha, 2005. ISBN 80-7226-033-2

[6] SKALA, V. *Světlo, barvy a barevné systémy v počítačové grafice*. 1. vyd. Praha: Academia, 1993. ISBN 80-200-0463-7

[7] STRYCH, V. *Triangulace a editování vrstevnic*. Plzeň, 2003. Diplomová práce na fakultě aplikovaných věd na Západočeské univerzitě v Plzni na katedře matematiky. Vedoucí diplomové práce Doc. Dr. Ing. Ivana Kolingerová.

[8] WEIMER, H.; WARREN, J.; TROUTNER, J.; WIGGINS, W.; SHROUT, J. *Efficient co-triangulation of large data sets*. 1. vyd. 1998. Digital Object Identifier: 10.1109/VISUAL.1998.745293

[9] ZSOLT, T. Rekonštrukcia obrazu pomocou triangulácie. Bratislava, 2004. Diplomová práce na fakultě matematiky, fyziky a informatiky na univerzitě Komenského. Vedoucí práce RNDr. Andrej Ferko, PhD.

WWW zdroje

[10] Wikipedia [online].
URL: <http://wikipedia.org> [cit. 2008-04-15].

[11] Rastrové formáty [online].
URL: <http://nlp.fi.muni.cz/nlp/aisa/NlpCz/Rastrove_formaty.html> [cit. 2008-01-27].

[12] Rastrové grafické formáty [online].
URL: <http://cgg.ms.mff.cuni.cz/~vajicek/presentations/rasform.pdf> [cit. 2008-01-24].

[13] EasyRGB - Color mathematics and conversion formulas [online].
URL: <http://easyrgb.com/math.html> [cit. 2008-03-12].

[14] ABZ.cz: slovník cizích slov - on-line hledání [online].
URL: <http://slovnik-cizich-slov.abz.cz> [cit. 2008-03-20].

[15] Vše o světle - 1. Co je to světlo (fotogtafovani.cz) [online].
URL: <http://www.fotografovani.cz/art/fozak_df/rom_1_01_cojetosvetlo.html>
[cit. 2008-04-17].

[16] Vše o světle - 2. Světlo, oko a mozek (fotogtafovani.cz) [online].
URL: <http://www.fotografovani.cz/art/fozak_df/rom_1_02_svetlookomozek.html>
[cit. 2008-05-02].

[17] Vše o světle - 5. Barevné modely (fotogtafovani.cz) [online].
URL: <http://www.fotografovani.cz/art/fozak_df/rom_1_05_colormodels.html>
[cit. 2008-05-01].

[18] Printing.cz - Barvy a barevné modely II [online].
URL: <http://www.printing.cz/art/colormanagement/barvy_a_modely_2.html>
[cit. 2008-05-02].

Použité programy

[19] TriImgCompress [počítačový program]. 10.3.2008. Plzeň: University of West Bohemia, Ing. Josef Kohout, Ph.D, 10.3.2008. Požadavky na systém: x86 Windows platform.

[20] TriImgCotriangulation [počítačový program]. 10.3.2008. Plzeň: University of West Bohemia, Ing. Josef Kohout, Ph.D, 10.3.2008. x86, Windows platform.

Přehled použitých zkratek

BP	bakalářská práce
DDT	datově závislá triangulace
DT	Delaunayova triangulace
HSV	barevný systém HSV
Chunk	shluk
L*u*v	barevný systém L*u*v
MSE	střední kvadratická chyba
PSNR	poměr signálu ku šumu
px	pixel
RGB	barevný systém RGB
TIN	trojúhelníková síť
YCbCr	barevný systém YCbCr
***.bgf2**	přípona souboru se zakódovanou (ko)triangulací metodou FVXPATH
***.bgf2.bz2**	přípona souboru s komprimovanou, již zakódovanou, (ko)triangulací
***.dat**	přípona souboru s (ko)triangulací
***.log**	přípona souboru obsahujícího statistické výpisy
***.png**	přípona rastrového obrázku ve formátu PNG
***.raw**	přípona souboru se zakódovanou (ko)triangulací metodou RAW
***.raw.bz2**	přípona souboru s komprimovanou, již zakódovanou, (ko)triangulací

Přílohy

Příloha A (uživatelská dokumentace)

Utilita PSNR

Pomocná utilita načítající dva šedotónové/barevné obrázky. Program na základě dat, která načte ze vstupních souborů, umožňuje následující funkce:

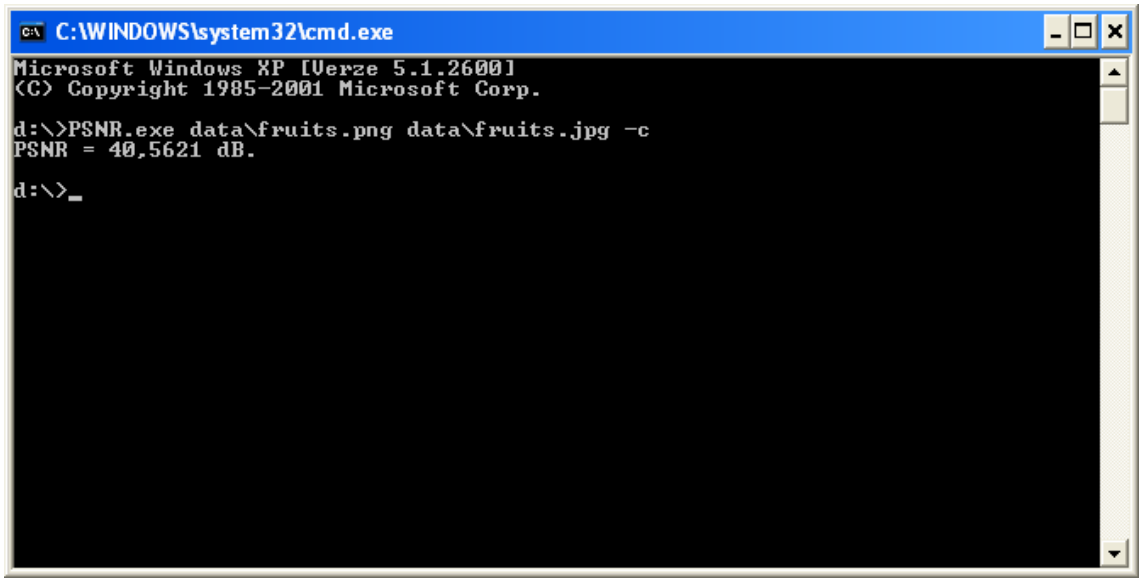
- výpočet odchylky (PSNR) mezi dvěma šedotónovými/ barevnými obrázky

Název programu je „PSNR.exe“, je spouštěn z příkazového řádku se třemi parametry. První a druhý parametr jsou vstupní obrázky. Třetím parametrem uživatel rozhodne o tom, jak bude na vstup nahlíženo (dva barevné/šedotónové obrázky => -c/g). Jsou-li zadány všechny parametry správně, program je zpracuje bez jakýchkoliv dalších vstupů uživatele, vypíše hodnotu PSNR na obrazovku a skončí.

Příklady vstupů s následným bezchybným chodem programu (Obr. A1):

.. \PSNR.exe data\ fruits.png data\fruits.jpg -c
.. \PSNR.exe data\ fruits.png data\fruits.jpg -g

- vypíše hodnotu PSNR a skončí



Obr. A1 Příklady vstupů s následným bezchybným chodem programu

Příklady vstupů s následným chybovým chodem programu (Obr. A2):

.. \PSNR.exe data\fruits.png -c
.. \PSNR.exe data\fruits.png -g

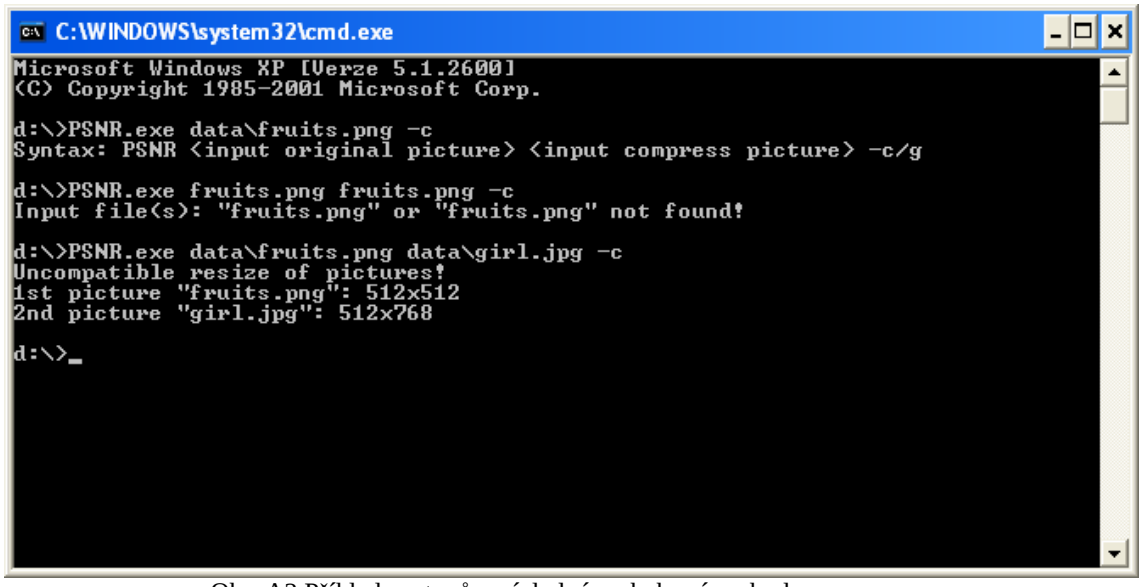
- vypíše zprávu o tom, jak má syntaxe vstupních parametrů vypadat a skončí

.. \PSNR.exe fruits.png fruits.jpg -c
.. \PSNR.exe fruits.png fruits.jpg -g

- vypíše zprávu o tom, že zadaný soubor(y) nebyl(y) nalezen(y) a skončí

.. \PSNR.exe data\fruits.png data\girl.jpg -c
.. \PSNR.exe data\fruits.png data\girl.jpg -g

- vypíše zprávu o tom, že zadané soubory jsou nekompatibilní a skončí



Obr. A2 Příklady vstupů s následným chybovým chodem programu

Utilita TriImgAnalyzer

Hlavní program umožňuje následující funkce:

- rozložení obrázku po složkách (HSV, L*u*v, RGB a YCbCr)
- rekonstrukce obrázku ze tří oddělených složek (HSV, L*u*v, RGB a YCbCr)
- vytvoření statistického souboru na základě vstupních dat

Název programu je „TriImgAnalyzer.exe“, je spouštěn z příkazového řádku s několika parametry - význam jednotlivých parametrů bude pospán na názorných příkladech. Jsou-li zadány všechny parametry správně, program je zpracuje bez jakýchkoliv dalších vstupů uživatele (s výjimkou módu vytvářejícího statistický soubor), provede požadovanou akci a skončí.

Příklady vstupů s následným bezchybným chodem programu (Obr. A3):

..\TriImgAnalyzer.exe data
..\TriImgAnalyzer.exe data\fruits.png

- rozloží všechny obrázky ze vstupního adresáře nebo pouze jeden obrázek zadaný jako parametr na vstupu, výstup uloží do DataCS a skončí
- adresář data obsahuje obrázek(y), které chcete rozložit

..\TriImgAnalyzer.exe DataCS Results
..\TriImgAnalyzer.exe DataCS\fruits.png Results

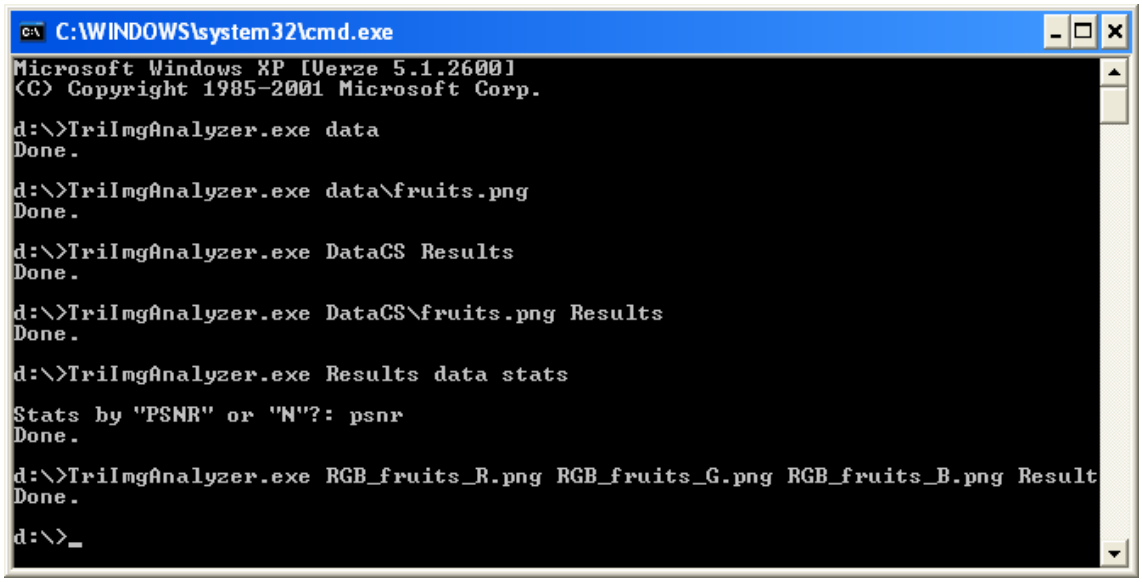
- složí všechny obrázky ze vstupního adresáře nebo pouze jeden obrázek zadaný jako parametr na vstupu, výstup uloží do výstupního adresáře (Results) a skončí
- adresář DataCS obsahuje jednotlivé obrázky uložené po složkách, ze kterých chcete složit původní obraz

..\TriImgAnalyzer.exe Results data stats

- zpracuje všechny obrázky ze vstupního adresáře (Results), pokud jsou správného formátu, tak je porovná s referenčním adresářem (data) a výstup uloží do výstupního adresáře (stats)
- uživatel je dotázán, zda-li chce k datům přistupovat podle PSNR a nebo podle počtu vrcholů v triangulaci N, druhý adresář (data) je v tomto případě brán jako referenční, tedy musí obsahovat originální předlohy, ze kterých byly obrázky původně rozloženy na jednotlivé složky (nutné pro výpočet PSNR)

..\TriImgAnalyzer.exe RGB_fruits_R.png RGB_fruits_G.png RGB_fruits_B.png Result

- složí tři obrázky zadané na vstupu, výstup uloží do adresáře (Result) a skončí



Obr. A3 Příklady vstupů s následným bezchybným chodem programu

Příklady vstupů s následným chybovým chodem programu (Obr. A4):

..\TriImgAnalyzer.exe

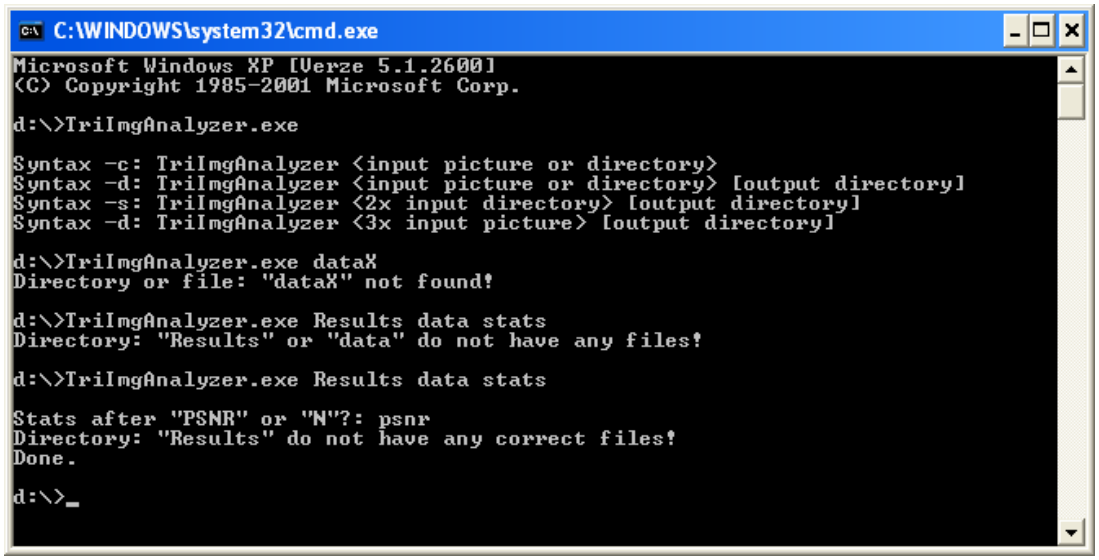
- vypíše zprávu o tom, jak má syntaxe vstupních parametrů vypadat a skončí

..\TriImgAnalyzer.exe data**X**
..\TriImgAnalyzer.exe data\fruits**.BMP**
..\TriImgAnalyzer.exe DataC**X** Results
..\TriImgAnalyzer.exe DataCS\fruits**X**.png Results

- vypíše zprávu o tom, že zadaný adresář nebo soubor nebyl nalezen a skončí
- všeobecně neexistující adresář/soubor zadaný chybně na vstupu jako parametr vede k vypsání této chybové hlášky a následnému ukončení běhu programu
- pokud je vstupní adresář prázdný (neobsahuje žádné soubory), program vypíše příslušnou zprávu o této skutečnosti a ukončí svou činnost

..\TriImgAnalyzer.exe Results data stats

- v případě, že adresář Results (*.dat, *.bgf2, *.raw, *.bgf2.bz2, *.raw.bz2, *.png) a nebo data (*.png) neobsahuje potřebné (korektní) soubory, tak je uživatel o této skutečnosti informován vypsáním příslušné hlášky a program ukončí svou činnost



Obr. A4 Příklady vstupů s následným chybovým chodem programu

Utilita RunCotriImgAnalyzer

Pomocná utilita spouštějící program CotriImgAnalyzer. Program na základě dat, která načte ze vstupu, umožňuje následující funkce:

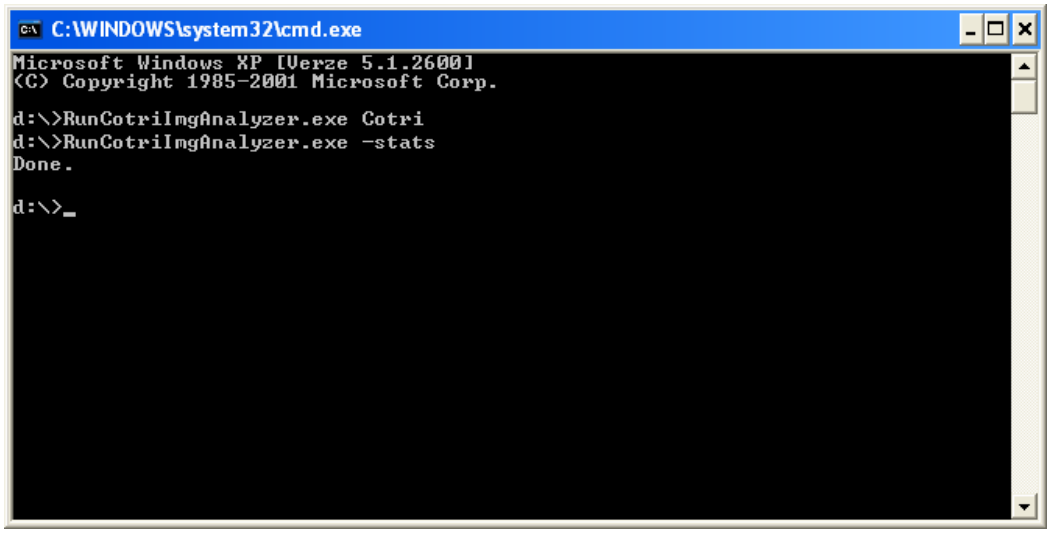
- spuštění programu CotriImgAnalyzer

Název programu je „RunCotriImgAnalyzer.exe“, je spouštěn z příkazového řádku s jedním parametrem. Tím je buď název vstupního adresáře, který obsahuje data ke zpracování programy CotriImgAnalyzer a TriImgCotriangulation, nebo parametr „-stats“ (vytvoření statistického souboru). Je-li zadán parametr na vstupu správně, program jej zpracuje bez jakýchkoliv dalších vstupů uživatele a skončí.

Příklad vstupu s následným bezchybným chodem programu (Obr. A5):

.. \RunCotriImgAnalyzer.exe Cotri
.. \RunCotriImgAnalyzer.exe -stats

- provede potřebné akce a skončí



Obr. A5 Příklady vstupů s následným bezchybným chodem programu

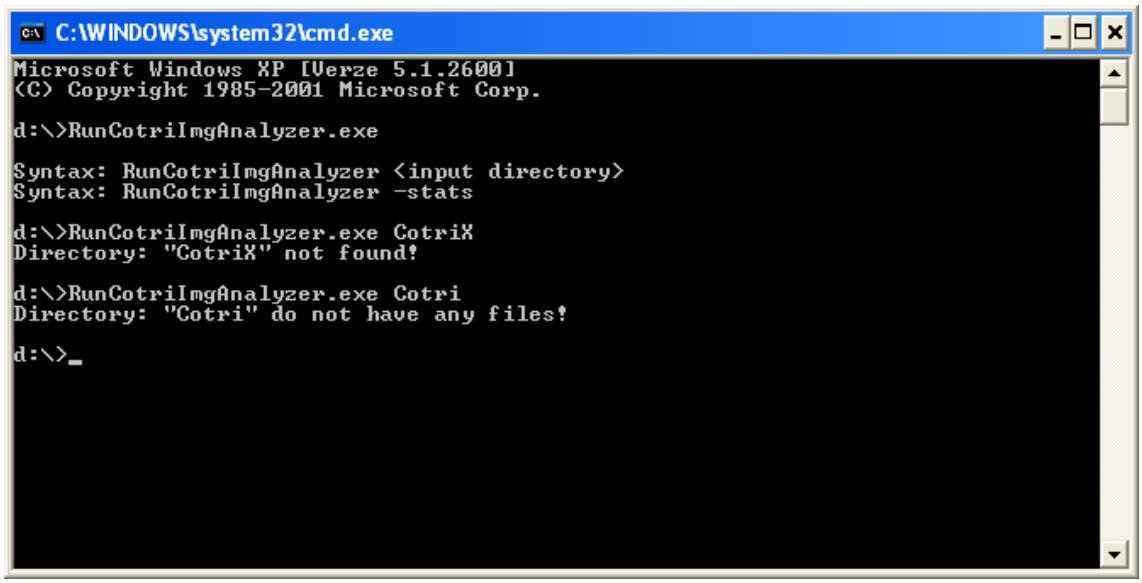
Příklady vstupů s následným chybovým chodem programu (Obr. A6):

..\RunCotriImgAnalyzer.exe

- vypíše zprávu o tom, jak má syntaxe vstupních parametrů vypadat a skončí

..\RunCotriImgAnalyzer.exe Cotri**X**
..\RunCotriImgAnalyzer.exe Cotri

- vypíše zprávu o tom, že zadaný adresář nebyl nalezen a skončí
- všeobecně neexistující adresář zadaný chybně na vstupu jako parametr vede k vypsání této chybové hlášky a následnému ukončení běhu programu
- pokud je vstupní adresář prázdný (neobsahuje žádné soubory), program vypíše příslušnou zprávu o této skutečnosti a ukončí svou činnost



Obr. A6 Příklady vstupů s následným chybovým chodem programu

Utilita CotriImgAnalyzer

Hlavní program umožňuje následující funkce:

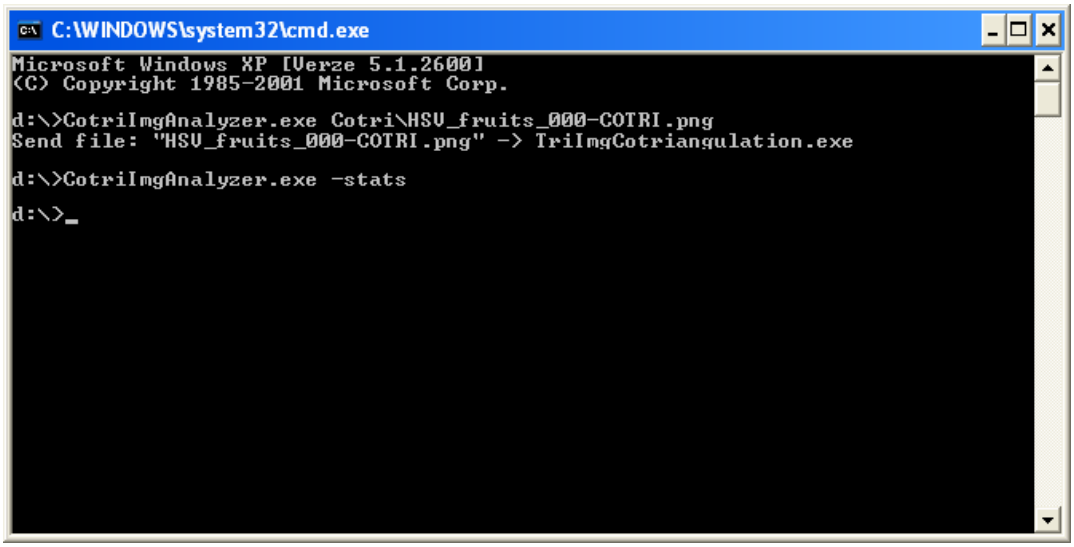
- spuštění programu TriImgCotriangulation
- zpracování dat, vygenerovaných programem TriImgCotriangulation
- zakódování kotriangulací metodou FVXPATH a RAW
- komprese již zakódovaných kotriangulací metodou bzip2
- vytvoření statistického souboru

Název programu je „CotriImgAnalyzer.exe“, je spouštěn z příkazového řádku s jedním parametrem. Tím je buď název vstupního obrázku, který je pojmenovaný stejně jako tři triangulace jednotlivých složek tohoto obrázku, nebo parametr „-stats“ (vytvoření statistického souboru na základě dat, která jsou generována programem TriImgCotriangulation). Ke spuštění je nutná existence adresáře Data, který musí obsahovat originální obrázek (výpočet PSNR). Je-li zadán parametr na vstupu správně, program jej zpracuje bez jakýchkoliv dalších vstupů uživatele a skončí.

Příklad vstupu s následným bezchybným chodem programu (Obr. A7):

.. \CotriImgAnalyzer.exe Cotri\HSV_ fruits _000-COTRI.png
.. \CotriImgAnalyzer.exe -stats

- provede potřebné akce a skončí



Obr. A7 Příklady vstupů s následným bezchybným chodem programu

Příklady vstupů s následným chybovým chodem programu (Obr. A8):

..\CotriImgAnalyzer.exe

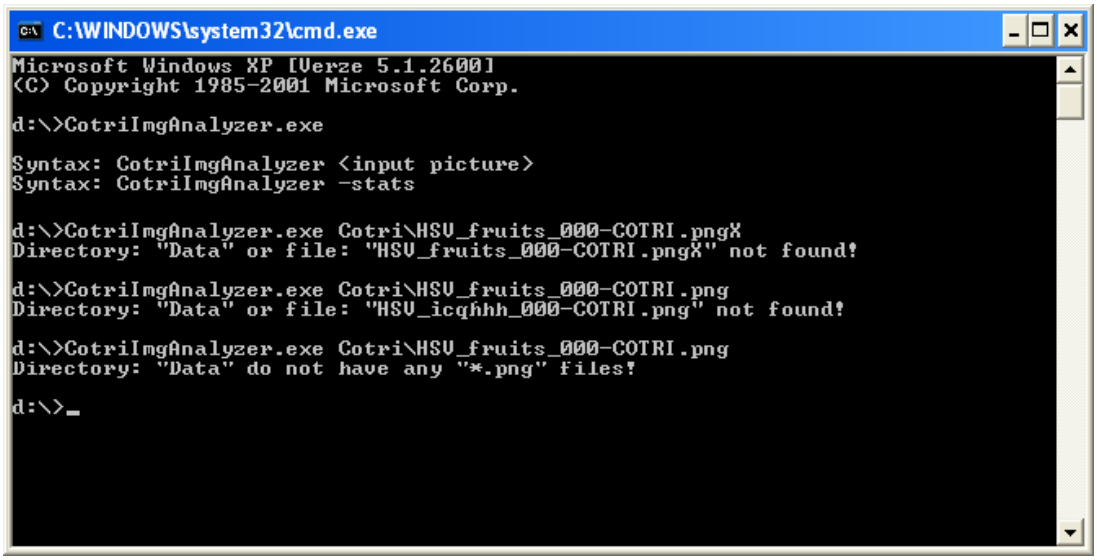
- vypíše zprávu o tom, jak má syntaxe vstupních parametrů vypadat a skončí

..\CotriImgAnalyzer.exe Cotri\HSV_fruits_000-COTRI.png

- neexistuje-li adresář Data, vypíše zprávu o tom, že tento adresář nebyl nalezen
- v případě, že tento adresář existuje, ale neobsahuje potřebné soubory, je opět vypsána zpráva o této skutečnosti

..\CotriImgAnalyzer.exe Cotri\HSV_ fruits _000-COTRI.png**X**

- vypíše zprávu o tom, že zadaný soubor nebyl nalezen a skončí



Obr. A8 Příklady vstupů s následným chybovým chodem programu

Příloha B (programátorská dokumentace)

Utilita PSNR

Zdrojové soubory:

- **BitmapClass.cs** metody pro práci s bitmapou
- **MainClass.cs** vstupní bod programu
- **RunClass.cs** metody potřebné pro výpočet PSNR

Důležité metody třídy BitmapClass.cs:

public bool GetLoadBitmap()
- načítá vstupní obrázky z parametrů příkazové řádky

public int GetWidth(int imgIndex)
- vrací šířku obrázku

public int GetHeight(int imgIndex)
- vrací výšku obrázku

public Color GetBitmapPixel(int imgIndex, int sirka, int vyska)
- metoda vrací barvu pixelu na pozici (šířka, výška)

Důležité metody třídy MainClass.cs:

static void Main(string[] args)
- vstupní bod programu, metoda zkontroluje, zda-li byl na vstupu zadán správný počet parametrů, pokud ne, tak je na výstup vypsána zprávu o tom, jak by měl vstup parametrů vypadat, v opačném případě zavolá metodu RunApplication a předá jí parametry args[0] a args[1]

Důležité metody třídy RunClass.cs:

public void RunApplication(string arg1, string arg2)
- metoda počítající PSNR dvou obrázků (parametrů) zadaných při spuštění programu z příkazového řádku, ve dvou vnořených FOR cyklech sčítá rozdíly barev v jednotlivých pixelech obou obrázků, zároveň testuje kompatibilitu obou obrázků, při nesrovnalosti rozměrů výpočet nespustí a vypíše zprávu o chybě, v případě, že je vše v pořádku (obrázky mají shodné rozměry), tak metoda spustí výpočet PSNR a po jeho ukončení vypíše vypočítanou hodnotu na obrazovku ve formátu: PSNR = XXX dB, při porovnávání dvou shodných obrázků je PSNR přiřazena hodnota 100dB

Utilita TriImgAnalyzer

Zdrojové soubory:

- **BitmapClass.cs** metody pro práci s bitmapou
- **BitmapClassComposition.cs** metody pro práci s bitmapou - kompozice
- **BitmapClassDecomposition.cs** metody pro práci s bitmapou - dekompozice
- **DirectoryClass.cs** metody pro práci s adresáři
- **ExceptionClass.cs** metoda vypisující chybové zprávy
- **MainClass.cs** vstupní bod programu
- **PSNRClass.cs** metody pro výpočet PSNR
- **StatisticsClassN.cs** metody pro zpracování statistik (N)
- **StatisticsClassPSNR.cs** metody pro zpracování statistik (PSNR)

Důležité metody třídy BitmapClass.cs:

public bool GetLoadBitmap()
- načítá vstupní obrázky z parametrů příkazové řádky

public Bitmap GetBitmap(int imgIndex)
- vrací bitmapu

public int GetWidth(int imgIndex)
- vrací šířku obrázku

public int GetHeight(int imgIndex)
- vrací výšku obrázku

public Color GetBitmapPixel(int imgIndex, int sirka, int vyska)
- metoda vrací barvu pixelu na pozici (šířka, výška)

Důležité metody třídy BitmapClassComposition.cs:

private string SaveBitmap(string colorSystem)
- ukládá zrekonstruovaný "originální" obrázek na disk

private string HLSComposition()
private string HSVComposition()
private string LABComposition()
private string LUVComposition()
private string RGBComposition()
private string YBRComposition()
private string YIQComposition()
private string YUVComposition()
- pracuje s obrázky jednotlivých barevných složek

public void RunClass()
- metoda vybírající barevný systém

Důležité metody třídy BitmapClassDecomposition.cs:

private void SetBitmapPixel()
- nastaví barvu pixelu obrázku na pozici (šířka, výška)

private string SaveBitmap(string colorSystem)

- ukládá obrázky po složkách na disk

private string HLSDecomposition()
private string HSVDecomposition()
private string LABDecomposition()
private string LUVDecomposition()
private string RGBDecomposition()
private string YBRDecomposition()
private string YIQDecomposition()
private string YUVDecomposition()

- rozkládá obrázek na jednotlivé barevné složky

public void RunClass()

- metoda vybírající barevný systém

Důležité metody třídy DirectoryClass.cs:

public void SetCreateDirectory(string adresarCreate)

- snaží se vytvořit výstupní adresář, pokud takový adresář existuje, tak ho nastaví jako výchozí adresář pro ukládání výsledků, pokud takový adresář neexistuje, tak ho vytvoří a posléze ho nastaví jako výchozí adresář pro ukládání souboru

Důležité metody třídy MainClass.cs:

static void Main(string[] args)

- vstupní bod programu, metoda zkontroluje, zda-li byl na vstupu zadán správný počet parametrů, pokud ne, tak je na výstup vypsána zprávu o tom, jak by měl vstup parametrů vypadat, v opačném případě zavolá příslušné metody dle zadaného vstupu a tyto vstupní parametry jim předá

Důležité metody třídy PSNRClass.cs:

public float RunClassColor()
public float RunClassGrey()

- výpočet PSNR

Důležité metody třídy StatisticsClassN.cs a StatisticsClassPSNR.cs:

```
private struct FileInfoStruct
{
        public string fileFullName;
        public string originalFileName;
        public string colorSystem;
        public string colorSystemComponent;
        public double PSNR;
        public long N, resolution;
        public long fileSize_bgf2, fileSize_bgf2bz2, fileSize_dat;
        public long fileSize_png, fileSize_raw, fileSize_rawbz2;
}
```

- hlavní struktura

```
private struct FinalInfoStruct
{
        public int colorComponent1;
        public int colorComponent2;
        public int colorComponent3;
}
```

- pomocná struktura

private void LoadData()

- výkonná část programu, analyzuje vstupní data a vytváří statistický soubor

public void RunClass()

- volá metodu LoadData()

Utilita RunCotriImgAnalyzer

Zdrojové soubory:

- **MainClass.cs** vstupní bod programu
- **ExceptionClass.cs** metoda vypisující chybové zprávy

Důležité metody třídy MainClass.cs:

static void Main(string[] args)

- vstupní bod programu, metoda zkontroluje, zda-li byl na vstupu zadán správný počet parametrů, pokud ne, tak je na výstup vypsána zprávu o tom, jak by měl vstup parametrů vypadat, v opačném případě pomocí příkazu FOREACH projde všechny soubory ze vstupního adresáře a pošle je ke zpracování programu CotriImgAnalyzer

Utilita CotriImgAnalyzer

Zdrojové soubory:

- **BitmapClass.cs** metody pro práci s bitmapou
- **DirectoryClass.cs** metody pro práci s adresáři
- **EncodingClass.cs** metody pro kódování kotriangulace
- **ExceptionClass.cs** metoda vypisující chybové zprávy
- **MainClass.cs** vstupní bod programu
- **PreprocessingClassCotri.cs** metody pro zpracování dat
- **PSNRClass.cs** metody pro výpočet PSNR
- **StatisticsClassCotri.cs** metody pro zpracování statistik

Důležité metody třídy BitmapClass.cs:

public bool GetLoadBitmap()

- načtení obrázku

public Bitmap GetBitmap(int imgIndex)

- vrací bitmapu

public int GetWidth(int imgIndex)

- vrací šířku obrázku

public int GetHeight(int imgIndex)

- vrací výšku obrázku

public Color GetBitmapPixel(int imgIndex, int sirka, int vyska)

- metoda vrací barvu pixelu na pozici (šířka, výška)

Důležité metody třídy DirectoryClass.cs:

public void SetCreateDirectory(string adresarCreate)

- snaží se vytvořit výstupní adresář, pokud takový adresář existuje, tak ho nastaví jako výchozí adresář pro ukládání výsledků, pokud takový adresář neexistuje, tak ho vytvoří a posléze ho nastaví jako výchozí adresář pro ukládání souboru

Důležité metody třídy EncodingClass.cs:

```
private struct VertexArray
{
        public short X;
        public short Y;
        public byte color_R;
        public byte color_G;
        public byte color_B;
        public bool visited;
}
```

- hlavní struktura pro vrchol kotriangulace

private void AdjustData(ref int[] pData, int nCount, ref int nMin, ref int nMax)

- načtení a posunutí vstupních dat odečtením minima

private void WriteDataIntoStream(ref int[] pData, int n, int nMax, ref BinaryWriter bw)

- zjistí maximální počet bitů dat a na takový počet bitů uloží data ze vstupního pole rozdílů do výstupního binárního souboru

private void FVXPATHandBZ2(string fullDATFileName)

- kódování kotriangulace metodou FVXPATH + komprese algoritmem bzip2

private void RAWandBZ2(string fullDATFileName)

- kódování kotriangulace metodou RAW + komprese algoritmem bzip2

public void RunClass()

- volá příslušné metody potřebné k zakódování kotriangulace

Důležité metody třídy MainClass.cs:

static void Main(string[] args)

- vstupní bod programu, metoda zkontroluje, zda-li byl na vstupu zadán správný počet parametrů, zároveň testuje existenci kontrolního adresáře Data, pokud jedna z podmínek není splněna, tak je na výstup vypsána zprávu o tom, jak by měl vstup parametrů vypadat, v opačném případě zavolá metody potřebné k analyzování dat a k vytvoření statistického souboru

Důležité metody třídy PreprocessingClassCotri.cs:

private string SaveBitmap(string colorSystem)

- ukládá obrázky na disk

private void SetBitmapPixel()

- nastaví barvu pixelu obrázku na pozici (šířka, výška)

private string RGBtoHSV(string picture)
private string RGBtoLUV(string picture)
private string RGBtoRGB(string picture)
private string RGBtoYBR(string picture)

- rozkládá výstupní obrázky z programu TriImgCotriangulation barevným systémem RGB a znovu je skládá buď barevným systémem HSV, L*u*v, RGB a nebo YCbCr

public void RunClass()

- metoda spouští program TriImgCotriangulation a zpracovává výstupní data generovaná tímto programem

Důležité metody třídy PSNRClass.cs:

public float RunClassColor()
public float RunClassGrey()

- výpočet PSNR

Důležité metody třídy StatisticsClassCotri.cs:

```
private struct FileInfoStruct
{
        public string PSNR;
        public string N;
        public long fileSize_bgf2, fileSize_bgf2bz2, fileSize_dat;
        public long fileSize_png, fileSize_raw, fileSize_rawbz2;
}
```

- hlavní struktura

private void LoadData()

- výkonná část programu, analyzuje vstupní data a vytváří statistický soubor

public void RunClass()

- volá metodu LoadData()

Příloha C (obsah přiloženého CD)

- **data**
  - ***images*** použité obrázky
  - ***stats*** statistiky získané během psaní BP

- **exe**
  - ***encoders*** kodéry
  - ***CotriImgAnalyzer.exe*** aplikace CotriImgAnalyzer
  - ***msvcm90.dll*** knihovna dll
  - ***msvcp90.dll*** knihovna dll
  - ***msvcr90.dll*** knihovna dll
  - ***RunCotriImgAnalyzer.exe*** aplikace RunCotriImgAnalyzer
  - ***TriImgAnalyzer.exe*** aplikace TriImgAnalyzer
  - ***TriImgCompress.exe*** aplikace TriImgCompress
  - ***TriImgCotriangulations.exe*** aplikace TriImgCotriangulation

- **src**
  - ***CotriImgAnalyzer*** zdrojové kódy projektu
  - ***PSNR*** zdrojové kódy projektu
  - ***RunCotriImgAnalyzer*** zdrojové kódy projektu
  - ***TriImgAnalyzer*** zdrojové kódy projektu

- **text**
  - ***BP.doc*** text bakalářské práce v doc
  - ***BP.pdf*** text bakalářské práce v pdf

- **web**
  - ***CotriImgAnalyzer*** webová dokumentace projektu
  - ***PSNR*** webová dokumentace projektu
  - ***RunCotriImgAnalyzer*** webová dokumentace projektu
  - ***TriImgAnalyzer*** webová dokumentace projektu

- **readme.txt** popis struktury CD