Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Korekce složek barevných prostorů s využitím trojrozměrné interpolace

2020

Jiří Palacký

Vedoucí práce: RNDr. Eduard Bartl, Ph.D.

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Jiří Palacký
Název práce:	Korekce složek barevných prostorů s využitím trojrozměrné interpolace
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	RNDr. Eduard Bartl, Ph.D.
Počet stran:	49
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Jiří Palacký
Title:	Correction of color space elements by using three-dimensional interpolation
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, combined form
Supervisor:	RNDr. Eduard Bartl, Ph.D.
Page count:	49
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Cílem bakalářské práce je seznámit se s jednotlivými barevnými prostory, jejich využitím a dopadem na reálná schémata. Dále vysvětlit pojem barevné korekce a její využití, využití trojrozměrné interpolace v barevné korekci a vytvoření programu provádějící barevnou korekci za pomocí trojrozměrné interpolace.

Synopsis

Objective of this bachelor's thesis is to introduce color spaces and their usage in real applications. Further explain the concept of color correction, the use of three-dimensional interpolation in color correction and the creation of a program performing color correction using three-dimensional interpolation.

Klíčová slova: Korekce barev; barevný prostor; interpolace, LUT, RGB, HSL

Keywords: Color correction; color space; interpolation, LUT, RGB, HSL

Děkuji panu doktoru Bartlovi za vedení bakalářské práce a za volnost při práci. Dále pak odbornému konzultantu panu Hejzlarovi za pomoc v krušných začátcích tvorby programu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

S příchodem moderních informačních technologií způsobujících digitalizaci grafiky a neustálým zvyšováním výkonu moderních počítačů se neustále posouvají možnosti grafika využívat výpočetně a paměťově náročné operace. Tato bakalářská práce slouží jako úvod do problematiky barevné korekce s využitím trojrozměrné interpolace.

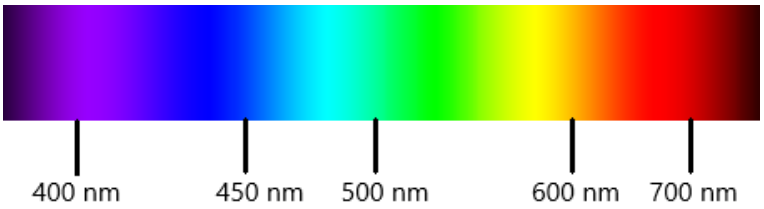
V teoretické části se nejprve zaměřím na barvy a barevné prostory, jejich využití a případné převody. Dále objasním, co všechno zahrnuje pojem „barevná korekce“. V poslední teoretické kapitole vysvětlím, co je to trojrozměrná interpolace, jaké struktury k ní bude potřeba vytvořit a jak s nimi pracovat.

Cílem praktické části práce je vytvoření programu, který umožňuje vytvářet, upravovat a zobrazovat trojrozměrnou vyhledávací tabulku, do které je zabalen barevný prostor. Pomocí tabulky pak přetransformuji vstupní obraz a porovnám pomocí referenčního obrazu přesnost interpolovaných hodnot a vhodnost použité barevné korekce.

2 Barvy

Dříve něž se budu bavit o barvách a jejich korekcích v počítačové grafice, je vhodné si nejdříve objasnit, co je to barva a jak jí vnímáme. Základem barevné informace je část spektra elektromagnetického záření ze světelného zdroje, která obsahuje záření o vlnových délkách v tzv. viditelném spektru. Působením takového světla na povrch materiálu se poté odráží podmnožina elektromagnetického záření podle povrchové charakteristiky materiálu, která je pak zachycena lidským okem [**5**].

Barvu je možné charakterizovat jako vjem vytvářený na sítnici oka. Tato schopnost vnímat a rozeznávat barvy vzniká pomocí jednotlivých fotoreceptorů, tyčinek a čípků [**5**]. Tyčinky reagují především na intenzitu světla a to i na velmi malé změny intenzity za nízkého osvětlení a tedy slouží primárně jako noční vidění. Barevné vidění zpostředkovávají tři druhy čípků. Každý druh čípku je citlivý na rozdílný rozsah vlnových délek elektromagnetického záření ve viditelném spektru, viz obrázek 1. Jeden druh čípků, označován jako modrý, je citlivý na kratší vlnové délky viditelného spektra okolo 445 nm až 520 nm, druhé čípky označovány jako zelené jsou citlivé na vlnové délky okolo 535 nm až 680 nm a třetí čípky označovány jako červené začínající taktéž okolo 535 nm a končící až 750 nm (intervaly vlnových délek se z velké části pro zelené i červené čípky překrývají, ale s tím, že zelené čipky jsou citlivější na kratší vlnové délky a červené na delší) [**5**].



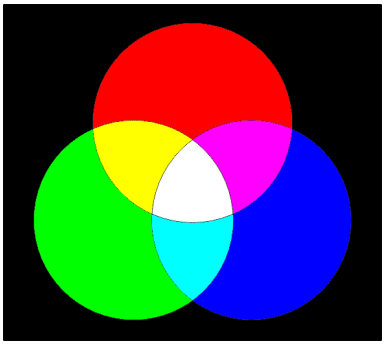
Obrázek 1: Viditelné spektrum

2.1 Aditivní míchání barev

Aditivní míchání barev je založeno na sčítání jednotlivých vlnových délek světla. Nejjednodušeji se tato metoda dá popsat např. skládáním paprsků ze tří světel a to červeným, zeleným a modrým. Kombinací různých intenzit světel jsme pak schopni vytvořit velikou škálu barev. Hlavní využití aditivního míchání je v počítačových monitorech, televizních obrazovkách a v počítačové grafice.

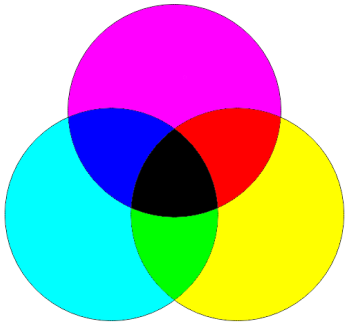
2.2 Subtraktivní míchání barev

Oproti aditivnímu míchání je subtraktivní míchání založeno na odčítávání neboli absorpci daných vlnových délek světla. Docílit absorpce se dá např. pomocí filtrů. Zeleným filtrem odstraníme červenou z barevného spektra, fialovým filtrem



Obrázek 2: Aditivní míchání barev

odstraníme zelenou a například žlutým filtrem odstraníme modrou část spektra. Subtraktivní míchání využívají malíři k namíchání chtěné barvy a barevný model CMY(K), který je používán k tisku.



Obrázek 3: Subtraktivní míchání barev

3 Barevné prostory

3.1 RGB

Nejznámější a nejpoužívanější barevný prostor, ze kterého vychází další podobné barevné prostory, je založen na aditivním míchání tří barev - červené (R, *red*), zelené (G, *green*) a modré (B, *blue*). Každá složka je popsána svojí intenzitou často uváděnou v rozsahu $0-1$. V počítačové grafice se intenzita barevné složky nejčastěji kóduje do jednoho bytu. Kódováním intenzity barevné složky do jednoho bytu se dá zobrazit $256^3 = 16\ 777\ 216$ barev. Používají se však i jiná kódování, např. pomocí 12 nebo 16 bitů [**5**]. Prostor se často graficky znázorňuje jako barevná krychle, kde jednotlivé osy reprezentují základní barvy R,G a B.

Výpočet jasu by bylo možné naivně popsat jako vážený průměr RGB složek, avšak lidské oko vnímá intenzitu jednotlivých barevných složek různým způsobem (nejcitlivější je na zelenožlutou). Pro výpočet jasu I se tedy nejčastěji

používá empirický vztah [**5**]:

$$I = 0,229R + 0,587G + 0,114B. \tag{1}$$

3.2 RGBA

Dalším často používaným barevným prostorem je RGBA (red, green, blue a alpha), který je RGB prostor obohacený o další kanál *alpha*, který uchovává hodnotu (např. v rozmezí 0 až 1) o průhlednosti zobrazovaného bodu [**5**].

3.3 CMY, CMYK

Je barevný prostor založený narozdíl od RGB na subtraktivním míchání barev. Základními barvami jsou - tyrkysová (C, *cyan*), purpurová (M, *magenta*), žlutá (Y, *yellow*) a často přidávaná černá (K, *key* nebo *black*; pak mluvíme o barevném modelu CMYK). Stejně jako RGB se dá CMY zobrazit jako krychle. Převod z RGB do CMY se dá jednoduše popsat rovnící:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{2}$$

Hlavním důvodem vzniku CMY prostoru byla realita, že tiskárny nemohou využít aditivní míchání barev kvůli tomu, že papír nevydává žadné záření. Černá barva je přidávaná hlavně z toho důvodu, že tiskárna nemůže použít plné barvy (C,M,Y), protože by poslední barva, co tiskne plně překryla ty dvě předchozí. Takže by se nenatiskla plná barva, ale pouze hnědá. [**16**].

3.4 HSV, HSL

Oba barevné prostory reprezentují alternativní zobrazení barevného modelu RGB. Hlavní myšlenkou pro vznik obou modelů je snaha přizpůsobit model lidskému vnímání barev. Oba prostory HSL i HSV definují barvu pomocí tří složek [**20**].

Pro HSV jsou to složky barevný tón (H, *hue*), sytost (S, *saturation*) a jasová hodnota (V, *value*) a pro HSL místo jasové hodnoty je složka světlosti (L, *lightness*). Prostor HSV je geometricky reprezentován jako šestiboký jehlan s vrcholem ležícím v počátku soustavy souřadnic. Jehlan je odvozen z RGB kostky (prostor RGB) pro hodnoty R, G, B z intervalu $[0, 1]$ tak, že se pootočí RGB kostka, aby bod reprezentující černou ležel v počátku soustavy souřadnic. Poté jsou ostatní barvy přemapovány do podstavy jehlanu. Bod reprezentující bílou je pak hlavní vrchol jehlanu. Složka V je výška jehnalu nabývající hodnot od 0 do 1 a složka reprezentující H je úhel měřený ze středu podstavy ke stěně jehnalu popisován v intervalu $(0°, 360°)$ [**20**].

Naproti tomu je HSL reprezentován místo jednoho jehlanu dvěma jehlany. Odvození je velice podobné jako u HSV akorát s tím rozdílem, že přemapování barev do podstavy obou jehlanů se provede tentokrát, i bez bodu bílé barvy. Podstava obou jehlanů bude ležet ve výšce dané souřadnicí $l = 0,5$. Vrcholem jednoho jehlanu bude bod bílé barvy ležící na souřadnici $l = 1$ a vrcholem druhého jehlanu bude bod černé barvy ležící na souřadnici $l = 0$ [**20**].

Pro převod z RGB do HSL/HSV je důležité si objasnit nový pojem *Chroma*. Projekcí pootočené RGB kostky, aby bod reprezentující černou ležel na počátku soustavy souřadnic, do roviny kolmé k ose L nebo V vznikne v počátku soustavy souřadnic pravidelný šestiúhelník s vrcholy reprezentujícími barvy, známý taktéž jako „chromatická rovina“. Chromou, budu rozumět vzdálenost od středu vzniklého šestiúhelníku k promítnutému bodu [**20**].

Pro kterýkoliv bod, kde platí $R = G = B$, je možné si všimnout, že výsledkem promítnutí do chromatické roviny, bude střed a tedy *chroma* rovna 0. Tedy pokud se odebere nebo přičte stejná hodnota ke všem složkám R, G a B, tak se pohnu vertikálně po mnou pootočené RGB kostce, aniž bych jakkoliv ovlivnil pozici promítnutého bodu. *Chroma* bodu, který má některou složku rovnou 0, je tedy rovna větší ze dvou zbývajících složek. Pro situaci, kde není nejmenší složka nulová, vypočítáme *chromu*, jako rozdíl největší a nejmenší složky [**20**].

Mohu tedy definovat:

$$\mathbf{Max} = max(R, G, B), \tag{3}$$

$$\mathbf{Min} = min(R, G, B), \tag{4}$$

$$Chroma = \mathbf{Max} - \mathbf{Min}. \tag{5}$$

Barevný tón H se pak počítá jako poměr vzdálenosti hrany šestiuhelníku, procházejícím promítnutým bodem, a vzdáleností šestiuhelníku. V minulosti popisováno na intervalu $[0, 1)$ ale v dnešní době spíše převáděno na stupně.

$$H' = \begin{cases} \text{nedefinováno}, & \text{pro } \mathit{Chroma} = 0, \\ \frac{G-B}{\mathit{Chroma}} \bmod 6, & \text{pro } \mathbf{Max} = R, \\ \frac{B-R}{\mathit{Chroma}} + 2, & \text{pro } \mathbf{Max} = G, \\ \frac{R-G}{\mathit{Chroma}} + 4, & \text{pro } \mathbf{Max} = B, \end{cases} \tag{6}$$

$$H = 60° \cdot H'. \tag{7}$$

V populární četbě se HSL i HSV graficky zobrazuje jako válec a v tomto kroku se přemapují souřadnice ze šestiuhelníku do kruhu [**20**].

Dalším krokem je vypočítání jasu.

- V HSV prostoru, se k vypočítání jasové složky V, využívá největší složka **Max**.

$$V = \mathbf{Max} \tag{8}$$

- V HSL prostoru, je jas L definována jako průměr největší složky **Max** a nejnižší složky **Min**.

$$L = 1/2(\mathbf{Max} + \mathbf{Min}) \tag{9}$$

Vyobrazování barev pomocí chromy by pro hodnoty $H \in [0°, 360°)$ $C \in [0, 1]$ a $V \in [0, 1]$ a polovina výsledných hodnot by spadla mimo zobrazitelné barvy. Výpočet saturace se tohoto problému snaží zbavit vydělením chromy maximální hodnotou chromy, daného jasu [**20**].

- Pro HSV:

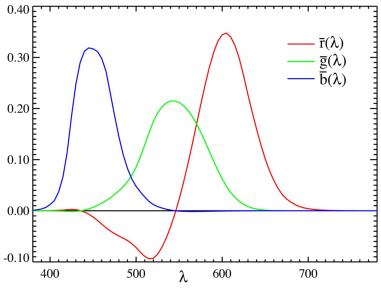
$$S = \begin{cases} 0, & \text{pokud } V = 0 \\ \frac{\mathbf{Chroma}}{V}, & \text{jinak} \end{cases} \tag{10}$$

- Pro HSL:

$$S = \begin{cases} 0, & \text{pokud } L = 1 \vee L = 0 \\ \frac{\mathbf{Chroma}}{1-|2L-1|}, & \text{jinak} \end{cases} \tag{11}$$

3.5 CIE XYZ

V roce 1931 byl vytvořen mezinárodní standard, aby bylo možné barvy měřit a vyjadřovat bez subjektivního lidského vnímání. Základem jsou tři primární barvy (červená, zelená a modrá). Pomocí kolometrických experimentů zaměřených na srovnávání vytvořených barev (skládání barev) byly stanoveny srovnávací funkce $(r(A), g(A), b(A))$ obrázek 4 [**5**], [**15**].



Obrázek 4: Srovnávací funkce [**6**]

Problémem takového vyjádření barvy je, že průběh funkce červené barvy nabývá záporných hodnot, protože reálně není možné skládáním vytvořit všechny barvy pouze třemi prímárními. Z toho důvodu byl vytvořený nový model CIE XYZ, ve kterém je barva určená srovnávacími funkcemi, tvořené třemi imaginárními barvami, které nenabývají nulových hodnot [**5**].

Následně bylo celé spektrum přetransformováno do jednotkové roviny XYZ vytvářející tzv. „spektrální podkovu“ ve které je barva určena souřadnicemi x a y, tvořící chromatický diagram. Z souřadnice bývá často zanedbáváva a nazývá se osvětlení [**5**].

4 Obrazové formáty

Nejjednodušší, ale málo úspornou reprezentací diskrétního obrazu, je dvourozměrná matice pixelů, kde každý pixel nabývá hodnot podle typu obrazu. Obrazu, skládajícímu se z pixelů, které jsou popsány jedním bitem, se říká, že je obraz jednobitový. Jednobitový obraz obsahuje dvě barvy, nejčaštěji bílou a černou barvu.

Další reprezentací je „indexový mód“. U takové reprezentace nereprezentuje pixel přímo barvu, ale je ukazatelem do tabulky tzv. barevné palety. Index v indexovém módu je reprezentován většinou jedním bytem, a tedy tabulka má jen většinou 256 řádků. Nejčastěji používaná paleta označovaná jako „8 - 8 - 8“ reprezentuje barvu RGB (po jednom bytu na barevný kanál). Tedy každý pixel obrazu nabývá některé barvy z 256 barev palety, které jsou vybrány z celkového množství 2^{24} barev. Toto přiřazení barev bývá označováno jako „pseudo color“.

Dále je také možné reprezentovat obraz jen v odstínech šedi, tzv. „grayscale“, kde je každý bod popsán přímo odstínem šedé.

Poslední případem je reprezentace pixelu všemi třemi barvami, nejčastěji v modelu RGB. Buď se jedná o tzv. „true color“ obraz, kde každý pixel obsahuje hodnotu všech tří barevných složek, nebo naproti tomu se může jednat o tzv. „direct color“ pracující s RGB hodnotami, které slouží jako odkaz do barevných palet pro každou jednotlivou barevnou složku.

4.1 BMP

BMP je grafický formát (zkratka „bitmap“), který i přesto, že nenabízí moc užitečných vlastností a je složitý na zpracování, je jeden z nejpoužívanějších rastrových formátů. Hlavním důvodem je fakt, že tento formát byl navržen firmami IBM a Microsoftem, a proto je načítání a ukládání BMP obrázku přímo podporováno v aplikačním rozhraní jejich operačních systémů (OS/2 a Microsoft Windows).

Každý BMP formát obsahuje následující struktury:

- **BITMAPFILEHEADER** - Datová struktura obsahující základní informace. Velikost vždy 14 bytů.
- **BITMAPINFOHEADER** - Datová struktura obsahující metainformace o obraze. Velikost vždy 40 bytů.
- **RGBQUAD** - Barevná paleta skládající se ze složek RGB.
- **BITS** - pole bitů obsahující pixely nebo jiná rastrová data.

Formát BMP se dá ukládat ve čtyřech různých formátech:

- **1 bit na pixel** - dvoubarevný obrázek (používá se barevná paleta).
- **4 bity na pixel** - obrázek skládající se ze 16 barev (používá se barevná paleta).
- **8 bitů na pixel** - obrázek skládající se ze 256 barev (používá se barevná paleta).
- **24 bitů na pixel** - je už výše popsaný „true color“ obrázek a tedy barevná paleta se nepoužívá.

Jedním z hlavních problémů tohoto formátu je neúspornost kvůli třem důvodům. Prvním je velikost hlavičkových struktur, které jsou neměnné. Druhým je velikost barevné palety, která si místo 3 bytů na barvu rezervuje 4. Třetím a hlavním důvodem je špatně komprimační schéma, kde u mnoha obrázků nastane nárůst velikosti [**12**].

4.2 JPEG

JPEG je grafický formát, pojmenován podle skupiny „Joint Photographic Experts Group“, která ho navrhla. JPEG je ztrátový formát, který se zakládá na lidském vnímání barvy. V první řádě si přetransformuje RGB data na tři složky - Y (informace o světelnosti) a C_b, a C_a (nesoucí informace o barvě). V dalším kroku se rozdělí obraz do čtverců 8×8 pixelů, ve kterých se pomocí DCT vypočítají koeficienty, které jsou pak vynásobeny kvantizační maticí a výsledek se zaokrouhlí. Kvantizační matice je daná zvolenou kompresí a je pro celý obrázek stejná.

Vzhledem ke komprimační metodě se JPEG formát nejvíce používá pro ukládání fotografií, protože blízké pixely ve fotografii jsou většinou rozdílné, ale podobné. JPEG je v dnešní době stále ještě nejpoužívanější rastrový formát pro přenos obrazu na internetu, ale kvůli ztrátovosti a nemožnosti uložení „alpha“ kanálu je nahrazován formátem PNG.

4.3 PNG

Formát PNG (zkratka „Portable Graphics Network“) byl navržen primárně pro přenos obrazu po síti. Od roku 2004 normou ISO a oproti formátu JPEG, používá bezeztrátovou kompresi, která se skládá ze dvou procesů: tzv. „filtering“ a použití kompresního algoritmu DEFLATE [**22**]. Proces filtering předzpracuje každý pixel podle pěti způsobů:

- Typ 0: **None** - Pixel není upravován.
- Typ 1: **Sub** - Je ukládán pouze rozdíl s předchozím pixelem vlevo.
- Typ 2: **Up** - Je ukládán pouze rozdíl s pixelem nahoře.

- Typ 3: **Average** - Je ukládán průměr pixelu s levým i pravým sousedem.
- Typ 4: **Paeth** -Je ukládaná hodnota získaná z daného pixelu a jeho tří sousedů, tedy levého, pravého a horního, pomocí metody navržené A. W. Paethem [**5**].

Po předzpracování všech pixelu se použije bezeztrátová komprese DEFLATE, která je kombinací slovníkové komprese LZ77 [**5**] a Huffmanova kódování [**5**].

Důležitým rozdílem oproti JPEG je tedy schopnost bezeztrátově ukládat v rozlišení „true color“ nebo možnost uchovávat barevné složky v rozsahu až 16 bitů. Taktéž může uchovávat údaj o průhlednost.

4.4 GIF

Jeden z nejstarších a stále používaných formátů, používající paletu s jedním bytem na pixel. Používá slovníkovou bezeztrátovou kompresi LZW [**5**]. Mezi hlavní charakteristiky patří:

- Možnost uchování více obrázků v jednom souboru, přičemž každý obrázek může mít svojí vlastní paletu.
- Možnost prokládání řádků, aby když se obrázek posílá po sítí, mohl uživatel už po získání částečného objemu dat rozpoznat vzhled obrazu.
- Řídící prvky, které umožňují interaktivní práci s obrazem.

5 Barevná korekce

Abych se mohl zabývat barevnou korekcí, je důležité si nejdříve objasnit, co si pod barevnou korekcí představit. Intuitivně by se dalo říci, že se jedná o dosáhnutí co nejvíce přirozené barevnosti, avšak ne vždy je snahou přirozená barevnost, ale barevnost požadovaná autorem [**18**].

Požadavek na barevnou korekci nevyplývá jen z potřeb uzpůsobit si obraz tak, aby vypadal podle představ autora, ale aby byl obraz přizpůsoben pro akci některého ze zařízení (např. tiskárny), které je schopno vytisknout nebo zobrazit jen určitý počet barev, který se neshoduje s počtem barev v obrazu. Barevnou korekcí mohu, také chtít docílit opravy barevných odchylek mezi obrazem zachyceným kamerou a reálně vnímaném obrazu lidským okem.

Digitální obrazy jsou nejčastěji zakódovány pomocí některé variace RGB, avšak jak už bylo v kapitole výše popsáno, lidskému vnímání je barva popsána pomocí HSL bližší. Přidávání saturace a světlosti barevným odstínům nebo zaměnění barevných odstínů za jiné je jedna z nejčastějších funkcí moderních grafických editorů. Je však důležité dát si pozor na ukládání obrazu, protože po převodu např. modré barvy zpět do RGB, které bych nastavil maximální jas, ztratím informaci o odstínu a saturaci barvy.

Všechny operace popsané v této kapitole se dají popsat transformační funkcí, která mění hodnoty pixelů. Často používanou optimalizací, aby se nemusela transformační funkce provádět nad každým pixelem je použití tzv. Lookup table (vyhledávací tabulka, zkráceně LUT), kde se transformační funkce použije jen nad řádky LUT. Hodnota pixelu slouží pak jen jako odkaz do LUT.

5.1 Úprava intenzit barevných kanálů

Úprava intenzit barevných kanálů je asi nejzákladnější a nejjednodušší barevná korekce. Hodí se k tomu, aby se dalo navýšit např. intezitu modrého kanálu čímž se docílí toho, aby obraz dostal modrý nádech. Takovéto úpravy jsou však málo používané kvůli často nechtěnému zabarvení bílých bodů.

5.2 Úprava jasu a kontrastu

Úprava jasu a kontrastu jsou základní běžně používané barevné korekce. Přidáním jasu budu rozumět zesvětlení cílových barev a snížením jasu naproti tomu budu rozumět ztmavení cílových barev.

Kontrastem budu rozumět rozlišitelnost mezi světlejšími a tmavšími barvami.

Zvyšováním kontrastu se docílí zesvětlení světlých barev a zároveň ztmavení tmavých barev. Snížením kontrastu se světlé a tmavé barvy příliš nezmění, ale obraz bude působit, jako by barvy začaly do sebe splývat [**7**]. V praktické části je pro výpočet kontrastu použit následující postup:

- Pro $L_{in} > 0,5$:

$$L_{out} = L_{in} + (L_{in} - 0,5) \cdot f. \tag{12}$$

  Pokud $L_{out} < 0,5$, pak:

$$L_{out} = 0,5. \tag{13}$$

- Pro $L_{in} < 0,5$:

$$L_{out} = L_{in} - L_{in} \cdot f. \tag{14}$$

  pokud $L_{out} > 0,5$, pak:

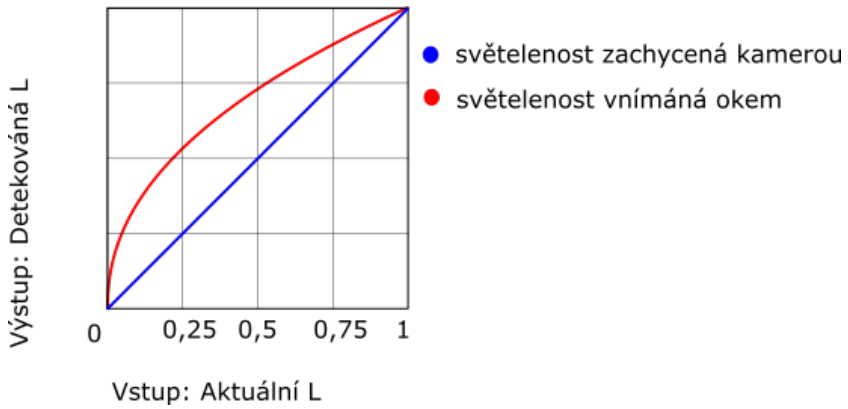
$$L_{out} = 0,5. \tag{15}$$

Kde L_{in} je původní hodnota jasu v rozsahu $\langle 0, 1\rangle$, f je libovolně zvolený koeficient v rozsahu $\langle -1, 1\rangle$ a L_{out} je výsledná hodnota jasu po úpravě kontrastu.

5.3 Gamma korekce

Gamma je velice důležitá charaketristika ve většině digitálních zobrazovacích systémech. Gamma korekce se definuje jako vztah mezi numerickou hodnotou pixelu a reálným jasem. Bez gammy, by odstíny zachycené kamerou nepůsobily stejně, jako by je v reálném světě vnímal člověk.

Hlavním důvodem použití je, že lidské oči nevnímají světlo stejně jako digitální kamery, např. při dopadu dvojnásobného počtu fotonů na senzor kamera zašle dvakrát signál a lineárně se dopracuje k výsledné hodnotě pixelu. Lidské oči ovšem takto lineární navýšení jasu nevnímají a místo toho, se nám obraz jeví, jen o zlomek jasnější a s přibývajícím světlem se nelineárně navyšuje [**2**].



Obrázek 5: Vnímaný jas okem a kamerou

Funkce gamma korekce je definovaná takto:

$$L_{out} = L_{in}^{gamma}. \tag{16}$$

Kde L_{in} je vstupní hodnota jasu umocněná koeficientem *gamma*, tak abych dostal požadovaný výsledný jas L_{out}. Hodnoty $gamma < 1$ se často používají k zakódování jasu obrazu a naproti tomu se hodnoty $gamma > 1$ používají k dekódování jasu obrazu.

5.4 Vyvážení bílé

Vyvážení bílé je další barevná korekce, pomocí které se snažím docílit obrazu zachyceného např. fotoaparátem, aby vypadal tak, jak by ho vnímal člověk okem. Každý světelný zdroj v závislosti na barevné teplotě, zanechává na zachycovacím zařízení daleko větší rozdíl barevnosti nežli zachytí lidské oko. Rozdíl se pak projeví tak, že světlem o malé hřejivosti vznikne na zachycovacím zařízení namodralý obraz a naopak hodně hřejivým světlem vznikne mírně oranžovo-načervenalý obraz [**1**], [**17**].

5.5 Barevná teplota

Barevná teplota světelného zdroje je teplota tzv. absolutně černého tělesa [**4**], které vyzařuje barvy podobné světelnému zdroji. Teplota barvy se stala běžnou charakteristikou viditelného světla používaná v osvětlovací technice, fotografování, filmování, počítačové grafice a ve spoustě dalších oborů. Teplota barvy má smysl pouze pro ty barvy, které dokáže vyzářit absolutně černé těleso za různých teplot, tudíž se jedná o světla barev jdoucích od červené do oranžové, z oranžové do žluté, ze žluté do bílé a z bílé do modro-bíle viz obrázek 17. Teplota barvy se udává v kelvinech. Světlu o barevných teplotách od 1800 K do 4000 K se říká teplé světlo, od 4000 K do 6500 K neutrální (běžné denní světlo) a od 6500 K výše studené světlo.

Obrázek 6: Barevné teploty

6 Lookup table

Lookup table (vyhledávací tabulka, zkráceně LUT) [**3**], [**9**] je vyhledávací datová struktura, která pro daný vstup vrátí náležící výstup. Vyhledávací tabulky jsou v grafice vhodné optimalizační struktury, které se využívájí v momentě, kdy vypočet funkce je náročný nebo transformace prováděné nad daty jsou obtížně uložitelné (např. mnoho ručních úprav barev obrazu). Nejjednodušší variantou jsou tzv. jednorozměrné LUT, které jsou charakterizovány právě jedním vstupním parametrem. Pro barevnou korekci *vstupem/výstupem* se rozumí např. barevný kanál. Může se také stát, že vstupní hodnota padne mezi navzorkované hodnoty a je zapotřebí pomocí aproximační metody, v mém případě interpolace, dopočítat přibližné hodnoty.

6.1 Jednorozměrná LUT

Hlavním problémem této variace LUT je, že ne vždy je možné namapovat výstup pro všechny barevné operace právě pro jediný vstupní parametr. Příkladem by mohla být operace, která převede barevný obraz do korespondencujícího *„greyscale“* obrazu. K takové operaci se používá hodnota jasu (luminance), která je složena z váženého průměru všech tří barevných kanálů. Jako řešení takového omezení, může sloužit trojrozměrná LUT [**9**].

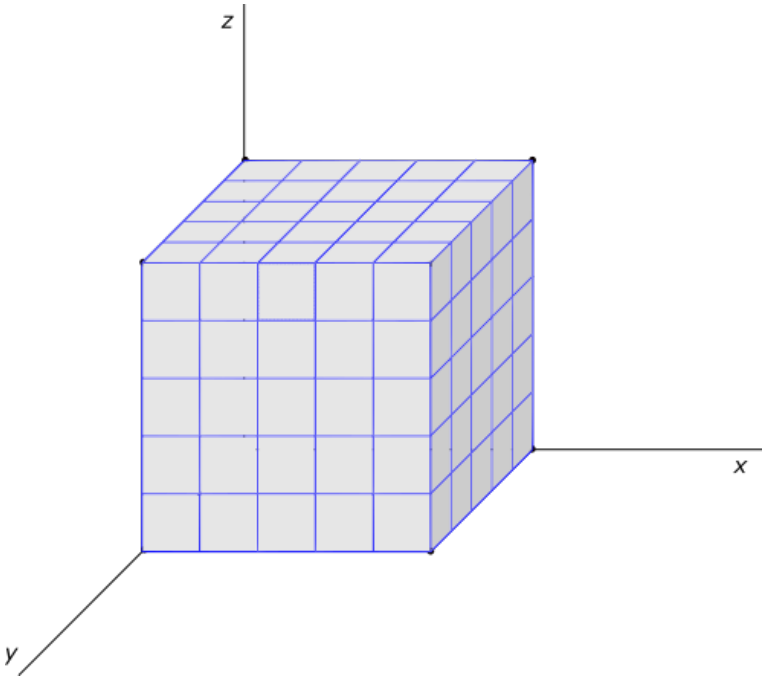
6.2 Trojrozměrná LUT

Jak už název napovídá, trojrozměrná LUT očekává právě tři vstupní parametry. Největším problémem trojrozměrné LUT oproti jednorozměrné LUT je mnohonásobně větší náročnost na paměť. K vyhledání např. požadované RGB hodnoty za pomocí jednorozměrné LUT je zapotřebí vytvořit jednorozměrnou LUT pro každý barevný kanál, o počtu položek B^2 (B - počet bitů na barevný kanál). Výsledný počet položek pak bude pro 24-bitové kódování do RGB obsahovat 256×3 a naproti tomu trojrozměrná LUT pro stejné kódování bude obsahovat 256^3 položek [**3**].

Tvorba a užití trojrozměrné LUT se nejčastěji dělí na tři části - zabalení, extrakci a vypočítání hledané hodnoty (v mém případě se bude jednat o interpolaci). Zabalení je proces, který zabalí zvolený barevný prostor a vhodně ho navzorkuje. Extrakce je další krok, ve kterém se nalezne pozice vstupního bodu a k němu nejbližší body. Body jsou pak použity k výpočtu interpolace vstupního bodu.

6.2.1 Omezení trojrozměrné LUT

Nejčastější používané barevné operace jako například úprava jasu, kontrastu atd. se dají namapovat, stejně tak i složitější transformace jako změna *saturace*, *barevného tónu*, nebo ještě složitější namapování na uživatelem nebo systémem vytvořený (pokřivený) barevný prostor. Bohužel, ani trojrozměrná LUT není schopna



Obrázek 7: Uniformě navzorkovaná trojrozměrná LUT

namapovat všechny reálně možné transformace, protože ne všechny transformace se dají jednoduše popsat přímo jako vstup/výstup funkce. Je tedy vhodné si nadefinovat podmínky transformace, které musí být dodrženy, aby bylo možné trojrozměrnou LUT použít [**9**].

- Počítání pixelů musí být prostorově nezávislé na pozici v obrázku. Barevné operace, které jsou ovlivněné sousedními hodnotami, nejsou vyjádřitelné vyhledávací tabulkou.
- Navzorkované hodnoty barevnou transformací musí být ideálně co nejvíce spojité. Pokud navzorkované hodnoty jsou málo spojité, tak interpolace může vracet nepřípustné hodnoty.

6.2.2 Neuniformní LUT

Pokud se nachází převážné množství hledaných bodů jen v určítých mezích, např. průměrná fotografie má nejvíce barev v odstínech zelené, tak interpolace zapříčiňuje větší průměrnou odchylku výpočtu výsledného bodu. Pro rovnoměrně navzorkovanou LUT se odchylka může snížit jen zvětšením velikosti LUT, které není vždy žádoucí. Řešením takového problému je nerovnoměrné navzorkování LUT za pomocí vhodného mapujícího algoritmu [**9**].

6.2.3 Aplikace trojrozměrné LUT

V současné době se v digitální grafice používání trojrozměrné LUT stalo běžnou záležitostí, kterou nabízí všechny známější programy na úpravu obrazu jako

je Adobe Photoshop, DaVinci Resolve atd. LUT taktéž mohou využívat grafické karty k úpravě obrazu videa v reálném čase, ve snaze docílit reálně vypadajícího obrazu nebo požadovaného efektu [**9**].

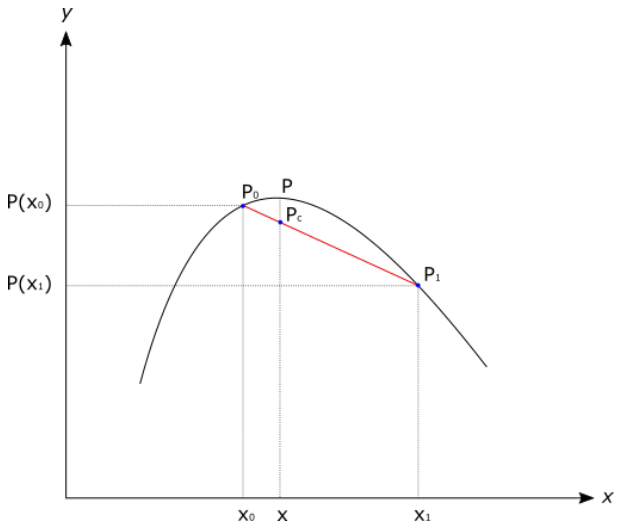
6.3 Interpolace

Obecně se pod pojmem interpolace [**3**], [**11**] dá představit nalezení přibližné hodnoty dané funkce v kterémkoliv intervalu, kde jsou známé hodnoty jen v některých bodech.

Jak už název práce napovídá, budu se hlavně zabývat geometrickou trojrozměrnou interpolací. K tomu, aby se dala trojrozměrná interpolace pochopit, nejdříve objasním lineární a poté bilineární interpolace, ze kterých se trojrozměrná interpolace skládá.

6.3.1 Lineární interpolace

V grafu na obrázku 8 je zobrazený bod P, ležící na křivce mezi body (v tomto případě mnou vhodně vybranými body mřížky vyhledávací tabulky) P_1 a P_2. Interpolovaná hodnota $P_c(x)$ je lineárně úměrná poměru $(x-x_0)/(x_1-x_0)$ [**3**], [**11**].



Obrázek 8: Lineární interpolace

Z grafu si odvodím rovnici:

$$\frac{P_c(x) - P(x_0)}{P(x_1) - P(x_0)} = \frac{x - x_0}{x_1 - x_0}. \tag{17}$$

Vyřešením rovnice pro $P_c(x)$ dostanu:

$$P_c(x) = P(x_0) + (x - x_0)\frac{P(x_1) - P(x_0)}{x_1 - x_0}. \tag{18}$$

Odchylka je definováná jako:

$$\delta = P(x) - P_c(x). \tag{19}$$

6.3.2 Bilineární interpolace

Bilineární interpolace je funkce $P_c(x, y)$ ve dvou rozměrech a čtyřmi body P_{00}, P_{01},P_{10}, P_{11}, aneb jak je znázorněno na grafu v obrázku 9. Abych mohl dostat hodnotu P, tak nejdříve musím získat bod P_0 lineární interpolací nad body P_{00} a P_{10} pro konstatní y_0 [**3**], [**11**].

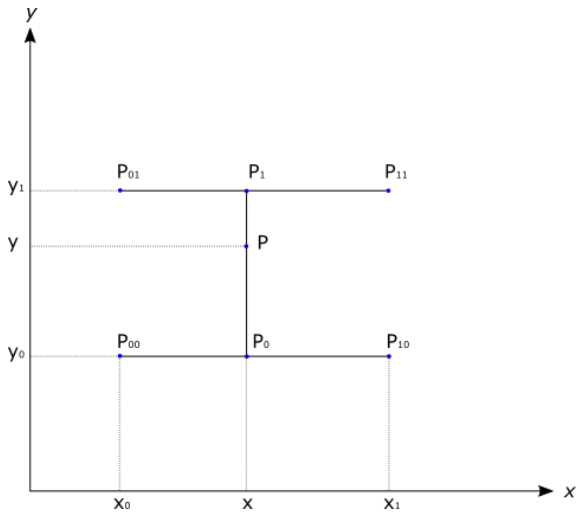
$$P_0 = P_{00} + (P_{10} - P_{00})\frac{x - x_0}{x_1 - x_0}. \tag{20}$$

Stejně tak vypočítám P_1 pro konstantní y_1.

$$P_1 = P_{01} + (P_{11} - P_{01})\frac{x - x_0}{x_1 - x_0}. \tag{21}$$

Poté dopočítané body se využijí k aplikování třetí lineární interpolace, tentokrát pro konstatní x:

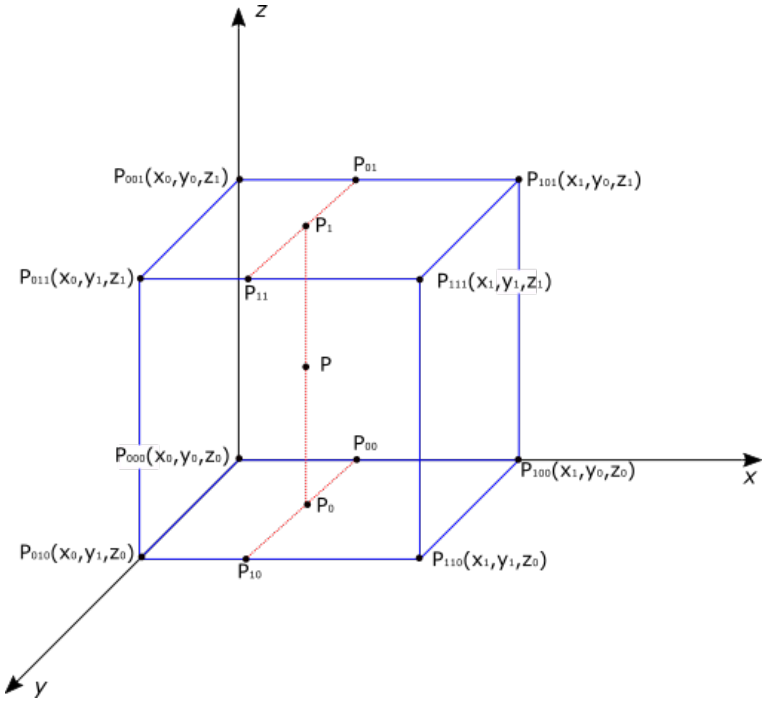
$$P(x, y) = P_0 + (P_1 - P_0)\frac{y - y_0}{y_1 - y_0}. \tag{22}$$



Obrázek 9: Bilineární interpolace

6.3.3 Trilineární interpolace

Trilineární interpolace je odvozena podobně jako bilineární interpolace, jen přibývá další rozměr, a pro vypočítání $P(x, y, z)$ je potřeba aplikovat lineární interpolaci sedmkrát.



Obrázek 10: Trilineární interpolace

Rovnici trilineární interpolace je možné obecně zapsat [**3**]:

$$\begin{aligned} P(x,y,z) = {} & c_0 + c_1\Delta x + c_2\Delta y + c_3\Delta z + c_4\Delta x\Delta y \\ & + c_5\Delta y\Delta z + c_6\Delta z\Delta x + c_7\Delta x\Delta y\Delta z, \end{aligned} \tag{23}$$

kde Δx, Δy a Δz jsou vzdálenosti bodů vzhledem k nulovému bodu P_{000} na osách x, y a z [**3**]:

$$\Delta x = \frac{x - x_0}{x_1 - x_0}. \tag{24}$$

$$\Delta y = \frac{y - y_0}{y_1 - y_0}. \tag{25}$$

$$\Delta z = \frac{z - z_0}{z_1 - z_0}. \tag{26}$$

Koeficienty c_j jsou dopočítány z hodnot vrcholů [**3**].

$$\begin{aligned}
c_0 =& P_{000}, \\
c_1 =& (P_{100} - P_{000}), \\
c_2 =& (P_{010} - P_{000}), \\
c_3 =& (P_{001} - P_{000}), \\
c_4 =& (P_{110} - P_{010} - P_{100} + P_{000}), \\
c_5 =& (P_{011} - P_{001} - P_{010} + P_{000}), \\
c_6 =& (P_{101} - P_{001} - P_{010} + P_{000}), \\
c_7 =& (P_{111} - P_{011} - P_{101} - P_{110} + P_{100} + P_{001} + P_{010} - P_{000}).
\end{aligned} \tag{27}$$

Rovnice by se dala zapsat zkráceně v lineárním tvaru [**3**] jako:

$$p = C^T Q_1, \tag{28}$$

Kde C je vektor koeficientů:

$$C = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{bmatrix}^T, \tag{29}$$

Q_1 je vektor všech vzdáleností vzhledem k Δx, Δy a Δz, takže C můžeme napsat jako:

$$C = \begin{bmatrix} 1 & \Delta x & \Delta y & \Delta z & \Delta x \Delta y & \Delta y \Delta z & \Delta z \Delta x & \Delta x \Delta y \Delta z \end{bmatrix}^T. \tag{30}$$

Je důležité si uvědomit, že vektory Q_1 a C musí mít stejnou velikost. Dále jsem schopen vložit koeficienty C do maticového tvaru jak je ukázano vnásledující rovnici:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} P_{000} \\ P_{001} \\ P_{010} \\ P_{011} \\ P_{100} \\ P_{101} \\ P_{110} \\ P_{111} \end{bmatrix} \tag{31}$$

nebo zkráceně:

$$C = B_1 P. \tag{32}$$

kde vektor P obsahuje všech osm vrcholů,

$$P = \begin{bmatrix} P_{000} & P_{001} & P_{010} & P_{011} & P_{100} & P_{101} & P_{110} & P_{111} \end{bmatrix}^T. \tag{33}$$

A matice B_1 o velikosti 8×8, reprezentuje binární konstanty. Substitucí rovnice (32) do (28), dostanu rovnici na výpočet cílové hodnoty p [**3**].

$$p = C^T Q_1 = P^T B_1^T Q_1. \tag{34}$$

7 Program ColorCorrection

K tomu abych mohl otestovat poznatky o barevné korekci za pomocí trojrozměrné interpolace je vytvořen v rámci bakalářské práce program *ColorCorrection*, který slouží jako podpora pro dosažené poznatky a jejich možné reálné aplikace. V první části této kapitoly se věnuji použitým technologiím a návrhu, další část slouží jako manuál k užití programu a závěr je zaměřen na testování, tedy srovnání dosažených výsledků s očekávanými výsledky a možnými optimalizacemi.

7.1 Použité technologie

7.1.1 Jazyk C#

Program je psaný na platformě .NET v jazyce C#, první myšlenka vývoje však byla pro jazyk C++, který by byl z výkonostných důvodů vhodnější, avšak na základě preference, možnosti pracovat s pointerovou aritmetikou i v jazyce C# a výkonu moderních počítačových zařízení, nebylo potřeba jej zvolit.

7.1.2 NCalc

Je volně dostupná knihovna (PCL - *portable class library*), která parsuje a zpracovává textové výrazy, které si uloží do třídy *NCalc.expression*, nad kterou se dají volat metody, vypočítající výraz dle nastaveného parametru [**13**].

7.1.3 OxyPlot

OxyPlot je cross-platform knihovna (PLC) pro .NET vykreslující grafy. Libovolně upravitelný ovládací prvek je implementován pro WPF, Windows 8, Windows Phone, Windows Phone Silverlight, Windows Forms, Xamarin.iOS, Xamarin.Forms, Xamarin.Android a GTK# [**10**].

7.2 Návrh programu

Z důvodu jednoduchosti práce s obrazy spustitelný program nevyužívá RGB, ale HSL LUT, jinak by vznikalo zbytečně veliké množství zaokrohlouvacích chyb, které by nebyly vhodné pro testování správnosti vypočítaných hodnot pomocí trojrozměrné interpolace.

7.2.1 Logická část

- **Třída *Algorithm***
  - Je třída obsahující pouze statické metody, které jsou převážně určeny pro matematické operace.

- **Třída *_3DLut***
  - obsahuje struktury pro uchování HSL bodu, RGB bodu a veškeré převody a práce s nimi.
  - obsahuje převážnou logickou část práce. Instance třídy si uchovává adresu LUT, a obsahuje veškerou logiku pracující s ní.
  - Stěžejní metodou celého programu je metoda provádějící trilineární interpolaci ***triInterpolateHslColor()***.
- **Třída *ImgProcessor***
  - obsahuje veškerou práci s obrazy i aplikace LUT, je-li instance třídy _3DLut předaná

7.2.2 Obsluha oken

- **Třída *CCForm***
  - obstarává a vytváří hlavní okno uživatelského prostředí programu.
- **Třída *ModifyForm***
  - Obsluhuje okno podle stavu měnící buď nahranou LUT nebo nahraný obrázek.
- **Třída *ContrastForm***
  - Obsluhuje okno podle stavu měnící kontrast buď nahranou LUT nebo nahraný obrázek.
- **Třída *ToneFunctionForm***
  - Obsluhuje okno podle stavu měnící buď nahranou LUT nebo nahraný obrázek za pomocí funkce.
- **Třída *ColorRangeChange***
  - Obsluhuje okno, které konvertuje barevný rozsah odstínů na jiný.
- **Třída *ModifyGamma***
  - Obsluhuje okno, které mění jas pomocí gamma funkce.
- **Třída *ColorTemperature***

– Obsluhuje okno, které mění barevnou teplotu.

- **Třída *ColorTemperature***

  – Obsluhuje okno, které mění barevnou teplotu.

- **Třída *Gridmodif***

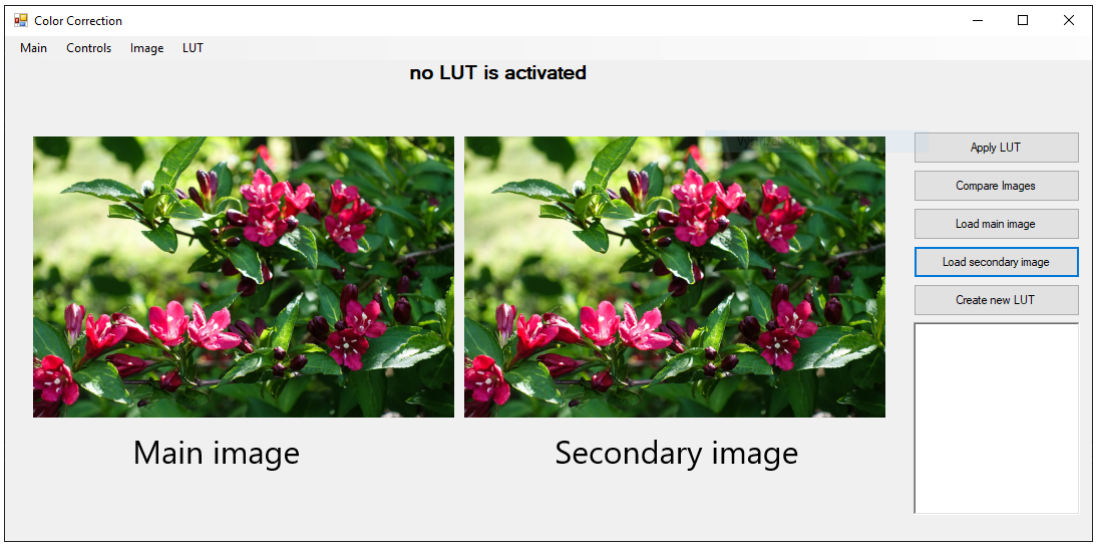
  – Obsluhuje okno, které vizualizuje pomocí barevné mřížky LUT.

7.3 Manuál

Program je psaný na platformě .NET s využitím *Window Forms*, tedy vzhled programu připomíná běžné *Windows* aplikace. Ke spuštění programu není potřeba žádná instalace, stačí pouze spustit sestavený *ColorCorrection.exe* soubor.

7.3.1 Hlavní okno – Color Correction

Na obrázku 11 je ukázáno úvodní a taktéž hlavní okno programu. Hlavní obrázek slouží pro účely aplikace načtené, nebo vytvořené LUT a druhý obrázek slouží buď k úpravě nebo jako reference k hlavnímu obrázku.



Obrázek 11: Hlavní okno programu

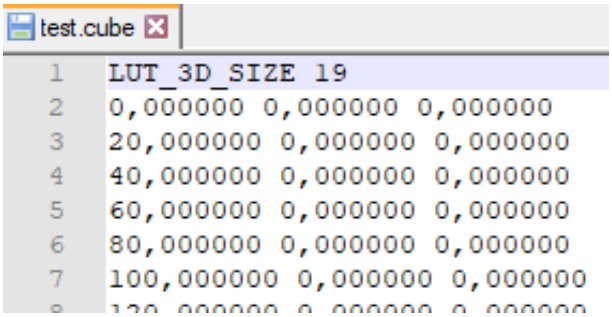
Tlačítka a textové pole na pravé straně:

- **Tlačítko *Create new LUT*** - Slouží k vytvoření nové LUT, kde je možné zvolit velikost z intervalu $\langle 4, 64 \rangle$, pro jiné hodnoty by LUT neměla smysl.

- **Tlačítko *Load main image*** - Slouží k nahrání hlavního obrázku z libovolné složky.
- **Tlačítko *Load secondary image*** - Slouží k nahrání druhého obrázku z libovolné složky.
- **Tlačítko *Apply LUT*** - Tlačítko aplikuje LUT na hlavní obrázek.
- **Tlačítko *Compare Images*** - Vypočítá rozdíl mezi hlavním a druhým obrázkem. Přidáno z analytických důvodů.
- **Textové pole** - Slouží k logování analytických informací, jako je čas provedených akcí, nebo rozdíl dvou obrázků v procentech.

Hlavní menu:

- **Main**
  - *New LUT* - Stejně jako tlačítko slouží k vytvoření nové LUT.
  - *Save LUT* - Slouží k uložení vytvořené nebo poupravené LUT do souboru ve formátu *.cube*. Formát *.cube* je charakterizován tím, že obsahuje na prvním řádku direktivu, jaká trojrozměrné data (neboli kostku), bude soubor obsahovat a celé číslo značící *velikost* strany kostky, tak jak je vidět na obrázku 12.



Obrázek 12: Ukázka .cube souboru

  - *Load LUT* - Slouží k načtení LUT ve formátu *.cube*.
- **Controls**
  - *Apply LUT to image* - Stejně jako tlačítko slouží aplikování LUT na hlavní obrázek.
  - *Compare Images* - Umožňuje podobně jako tlačítko **Tlačítko *Compare images*** výpočet rozdílu obrázků, navíc však s možnosti vypočítat rozdíl jen ve vybráné části obrázku.

- *Load main image* - Stejná funkce jako tlačítko.
- *Load secondary image* - Stejná funkce jako tlačítko.
- *Save main image* - Uloží hlavní obrázek.
- *Save secondary image* - Uloží druhý obrázek.

- **Image**
  - *Modify images* - Přepne uživatele do nového okna *Color modification* v módu úpravy obrázku.
  - *Modify Image contrast* - Přepne uživatele do nového okna *Contrast modification* v módu úpravy obrázku.
  - *Modify gamma* - Přepne uživatele do nového okna *ModifyGamma* v módu úpravy obrázku.
  - *Color temperature* - Přepne uživatele do nového okna *ColorTemperature* v módu úpravy obrázku.
  - *Color range swap* - Přepne uživatele do nového okna *ColorRangeSwap* v módu úpravy obrázku.
  - *Color tone curves* - Po výběru atributu (hue, saturation, luminence), který chceme modifikovat, přepne uživatele do nového okna *Tone curve modification* v módu úpravy obrázku.

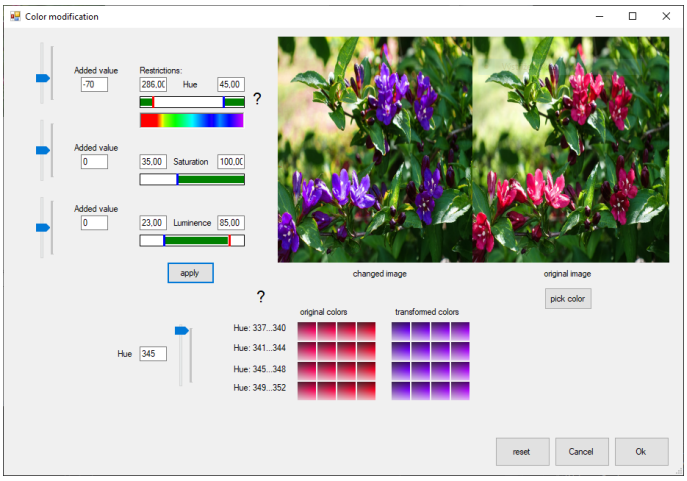
- **LUT**
  - *Modify LUT* - stejné jako *modify images* akorát pro LUT.
  - *Modify contrast* - stejné jako Modify *Image contrast* akorát pro LUT.
  - *Modify gamma* - stejné jako předchoz *Modify gamma* akorát pro LUT.
  - *Color range swap* - stejné jako předchoz *Color range swap* akorát pro LUT.
  - *LUT grid view* - Přepne uživatele do okna, které zobrazuje hodnoty LUT barevnou mřížkou.
  - *Color tone curves* - stejné jako předchozí *Color tone curves* akorát pro LUT.

7.3.2 Okno – Color modification

Jak je vidět na obrázku 13, okno Color modification, je jednoduché okno nabízející buď úpravu obrázku nebo LUT podle vybraného tlačítka. Pokud už byl na hlavním okně načtený obrázek (pro LUT hlavní a pro modifikaci obrázků druhý), tak se zobrazí v okně. Okno nabízí úpravu HSL složek pomocí jednoduché matematiky a to tak, že přidá vyplněnou hodnotu dané složce, pokud bod daného obrázku nebo LUT splní všechny podmínky určené pomocí *slidebaru*. Tlačítko *Pick color* slouží k vybrání (kliknutí na obrázek) barvy. Výběr barvy se projeví ve *sliderbarech*.

Pokud se jedná o úpravu LUT, je okno obohaceno o vizualizaci výsledných barev. Každý čtvereček reprezentuje jeden odstín *hue* (16 odstínu pro všechny velikosti LUT), vodorovná osa čtverečku udává barvu pro výše zadané rozmezí *luminiscence* a vodorvná osa udává barvu stejně, akorát pro *saturaci*.

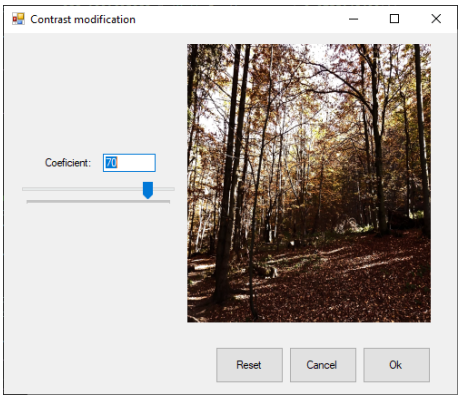
Tláčítkem *reset* vrátíme nastavení do původního stavu a tlačítkem *apply* aplikujeme změny na obrázek nebo LUT. Tlačítkem *Ok* potvrdím všechny změny.



Obrázek 13: Okno – Color modification

7.3.3 Okno – Contrast modification

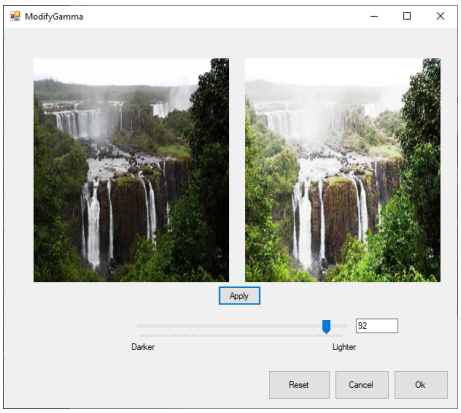
Další okno, nabízející běžně používanou úpravu kontrastu pomocí koeficientu zadaného *trackbarem*.



Obrázek 14: Okno – Contrast modification

7.3.4 Okno – Gamma modification

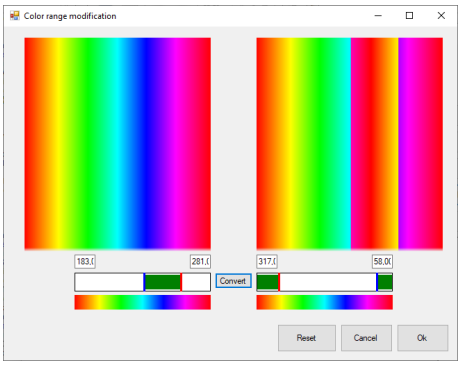
Okno nelineárně upravující jas pomoci gamma funkce, zadáním $koeficientu$ pomocí *trackbaru* nebo vepsáním do *Textboxu*. $Koeficient$ má rozsah 0 až 100, pro nižší hodnoty než 50 ztmavuje a pro vyšší zesvětluje. Pro $koeficient < 50$ se gamma vypočítá přemapováním koeficientu z intervalu $\langle 0, 50\rangle$ do intervalu $\langle 1, 8\rangle$ a pro $koeficient > 50$ se gamma vypočítá přemapováním koeficientu z intervalu $\langle 50, 100\rangle$ do intervalu $\langle 0,01; 1\rangle$.



Obrázek 15: Okno – ModifyGamma

7.3.5 Okno – Hue range convertion

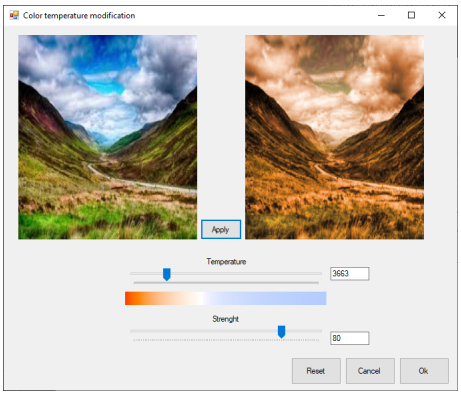
Okno, které konvertuje rozsah odstínů na jiný rozsah odstínů, jak je ukázáno na obrázku 16. Je taktéž možné nastavit rozsah jako doplněk, pokud nastavím druhou hodnotu rozsahu menší než tu první. Tím docílím, že jsem např. zkonvertoval všechny červené odstíny.



Obrázek 16: Okno – ColorRangeChange

7.3.6 Okno – Color temperature

Okno upravující barevnou teplotu pomocí dvou *trackbarů*. Prvním *trackbarem* se nastaví barevná teplota pomocí které, se vypočítá algoritmem [**14**] referenční barva. Vstupní barvu (barva pixelu obrázku nebo barva vrcholu LUT) pak promíchám pomocí hodnoty z druhého *trackbaru*, která určuje jak moc se bude výsledná barva přibližovat referenční barvě. Vzhledem k tomu, že míchání barev probíhá pro barvy v RGB modelu, bude vznikat pro okrajové hodnoty LUT veliká odchylka, viz kapitola testování.

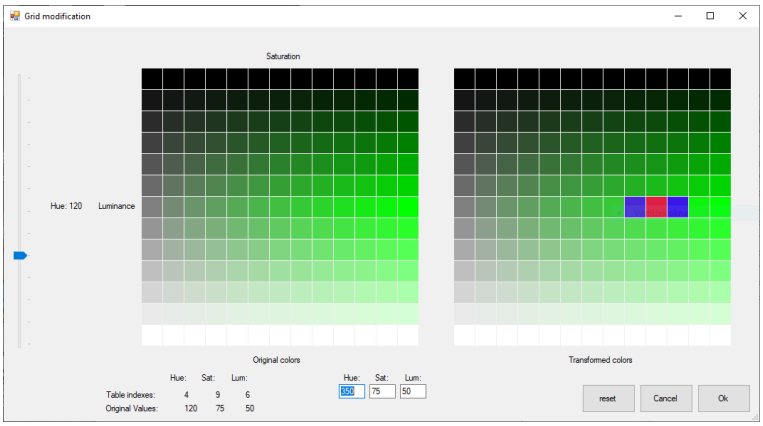


Obrázek 17: Okno – ColorTemperature

7.3.7 Okno – LUT grid view

Okno zobrazující LUT pomocí dvou barevných mřížek a to mřížky obsahující nepřetransformované barvy a mřížky obsahující transformované barvy. Výběr odstínu mřížky se uskutečňuje pomocí trackbaru na levé straně okna. Každý čtverec začínající vlevo má nulovou saturaci, která se postupně každému čtverci navyšuje směrem doprava, a naproti tomu každý čtverec začínající nahoře má nulovou luminiscenci, která se navyšuje směrem dolů.

Kliknutím na barevný čtverec je možné si libovolně upravit hodnoty transformované barvy.



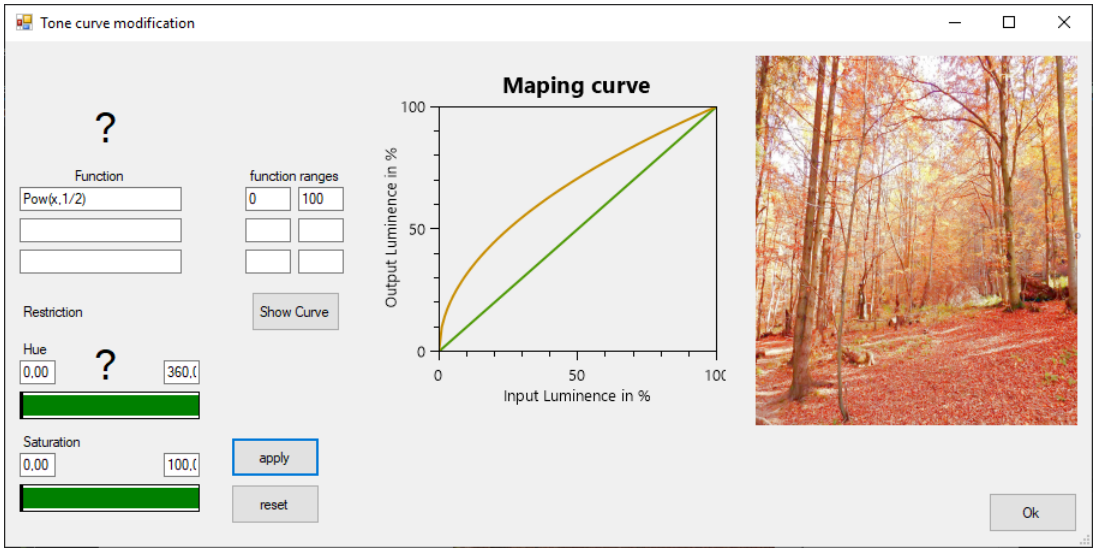
Obrázek 18: Okno – LUT grid view

7.3.8 Okno – Tone curve modification

Je poslední okno, ukázáno na obrázku 19. Okno je spíše určené pro experimentální účely, není tedy vhodné na reálnou barevnou korekci. Účelem okna je aplikovat na vybraný atribut (hue, saturation, luminence) funkci, která bude aplikovaná na všechny hodnoty daného atributu ve vybraných mezích. Funkce se aplikuje buďto na LUT nebo na druhý obrázek podle zmáčnutého tlačítka v hlavním okně.

Funkce může obsahovat operátory a funkce vypsány níže:

- +
- -
- *
- \
- Abs(-1)
- Ceiling(1.5)
- Cos(0)
- Exp(0)
- Log(1, 10)
- Min(1, 2)
- Pow(3, 2)
- Sin(0)
- Sqrt(4)



Obrázek 19: Okno – Tone curve modification

Další použitelné funkce jsou popsány v programu. Každá funkce má jen jeden parametr x a je možné ji použít jak je uvedeno na obrázku 19. Je vhodné si uvědomit, že graf znázorňující funkci a omezení jsou pro saturaci a jas v procentech, zatímco hodnota parametru x bude desetinné číslo. Program též povoluje napsat až tři funkce, které jsou aplikované postupně od shora.

7.4 Testování

Vzhledem k tomu, že LUT používá interpolaci k výpočtu tak, jak bylo popsáno výše v kapitole o interpolaci, bude vznikat odchylka. Tuto odchylku a čas potřebný k provedení akcí otestuji v této kapitole. Testy budou prováděny pro LUT o velikostech 9, 17 a 33, jejíž výsledky porovnám s referenčním obrázkem.

Odchylkou δrgb, rozumím průměrnou vzdáleností RGB hodnot pixelů dvou obrazků na stejné pozici. Vzhledem k tomu, že RGB tvoří krychli, použiji na měření vzdálenosti dvou pixelů rovnici výpočtu vzdálenosti dvou bodů v trojrozměrném euklidovském prostoru:

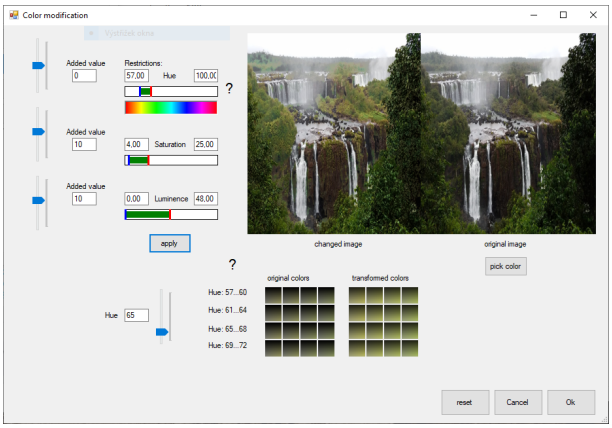
$$d = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}. \tag{35}$$

Kde r_1, g_1, b_1 jsou barevné složky pixelu z prvního obrázku a r_2, g_2, b_2 jsou barevné složky pixelu z druhého obrázku. Další měřený údaj, uveden spíše pro zajímavost, bude čas potřebný, jak pro aplikaci úpravy LUT, tak pro aplikaci LUT na obrázek.

7.4.1 Test – Lineární úpravy

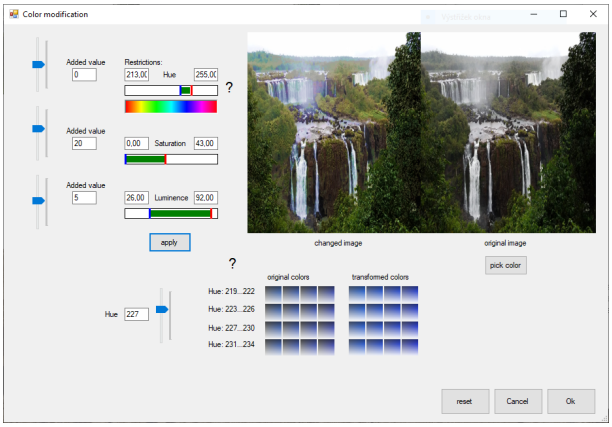
Jako první a nejjednodušší test provedu dvě lineární úpravy LUT:

- **Operace 1** - Zkusím navýšit saturaci a jas odstínům zelené, aby obrázek mírně ožil (obrázek 20).



Obrázek 20: Aplikace první operace

- **Operace 2** - Pokusím se ještě navýšit kontrast zeleně a vodopádu (obrázek 21).



Obrázek 21: Aplikace druhé operace

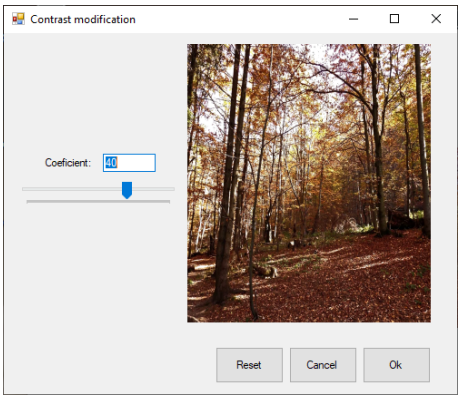
Obrázek 22: Srovnání výsledných obrázků

Tabulka 1: Výsledky prvního testu

	LUT-9	LUT-17	LUT-33	bez LUT
δrgb	21,848815	8,241944	4,43249	-
Doba modifikace LUT	1 ms	1 ms	8 ms	-
Doba modifikace obrázku	656 ms	658 ms	672 ms	437 ms

7.4.2 Test – Úprava kontrastu

V tomto testu se zaměřím na změnu kontrastu. Funkce nepřemapovává jas lineárně a pro používanou uniformní LUT je očekávaná daleko větší odchylka pro LUT o malé velikosti. Test provádím pro koeficient rovnající se 40, tak jak je ukázáno na obrázku 23.



Obrázek 23: Kontrast test

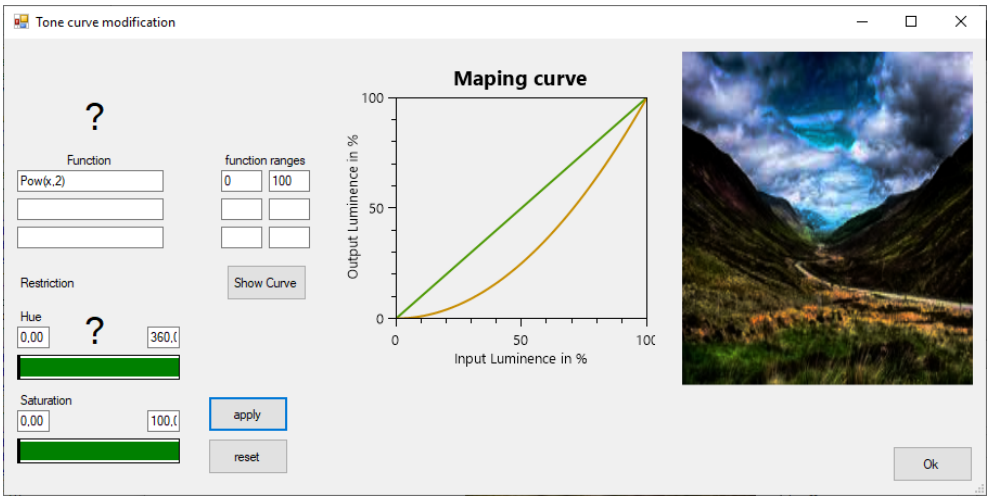
Obrázek 24: Srovnání výsledných obrázků kontrast testu

Tabulka 2: Výsledky druhého testu

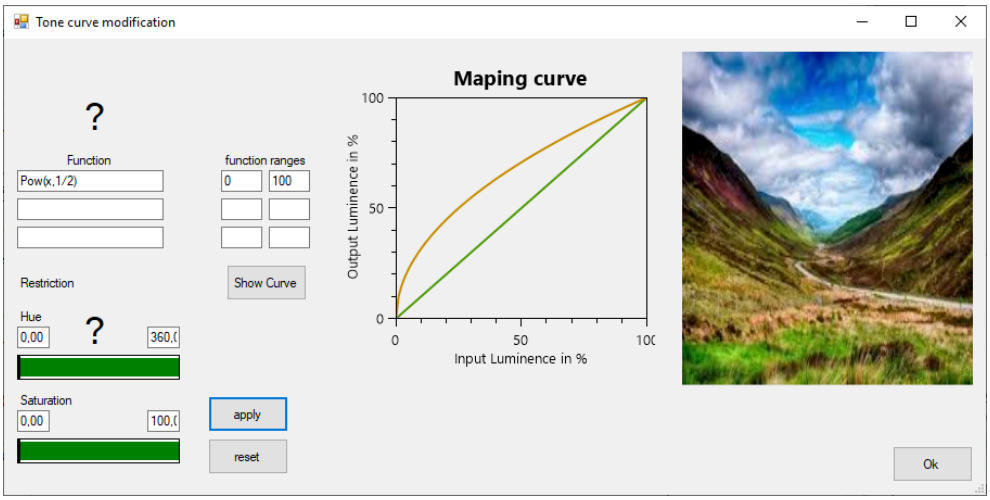
	LUT-9	LUT-17	LUT-33	bez LUT
δrgb	9,699285	4,984726	2,846646	-
Doba modifikace LUT	1 ms	1 ms	4 ms	-
Doba modifikace obrázku	861 ms	891 ms	897ms	275 ms

7.4.3 Test – Gamma korekce

Jako další test provedu gamma korekci a to tak, že použiji funkce $f(x) = x^{1/2}$ pro x v intervalu $\langle 0, 1 \rangle$ (obrázek 26). Nejdříve si ale obrázek upravím inverzní funkcí $f'(x) = x^2$, taktéž pro x v intervalu $\langle 0, 1 \rangle$ (obrázek 25). Vzhledem k tomu, že gamma korekce by byla znovu pouze korekcí jasu, zkusím si ještě k tomu aplikovat úplně stejné funkce na saturaci, aby byl test komplexnější.

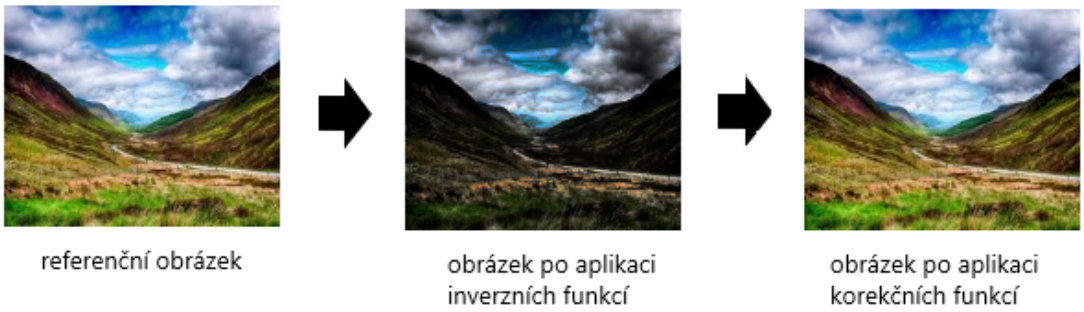


Obrázek 25: Aplikace inverzní gamma korekce na obrázek



Obrázek 26: Aplikace gamma korekce na obrázek

Abych výsledná data podrobil komplexnější zkoušce, použiji na každý test dvě LUT, jednu pro inverzní funkce a další pro korekční funkce.



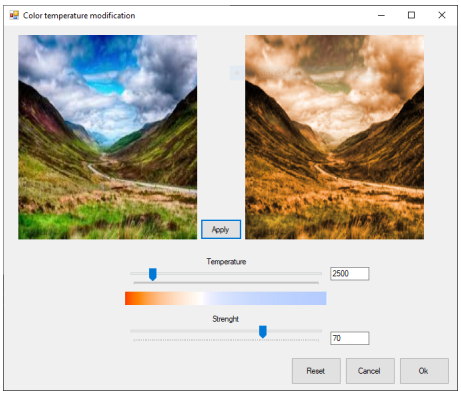
Obrázek 27: Aplikace funkcí na obrázek

Tabulka 3: Výsledky třetího testu

	LUT-9	LUT-17	LUT-33	bez LUT
δrgb	12,56876	7,951461	3,451694	-
Doba modifikace LUT	24 ms	156 ms	1104 ms	-
Doba modifikace obrázku	88 ms	86 ms	86 ms	256 ms

7.4.4 Test – Úprava barevné teploty

Jako poslední test jsem vybral úpravu pomocí barevné teplity. Vzhledem k tomu, jak bylo popsáno v kapitole 7.3.6, je očekávaná chyba. Prvně vyzkouším upravit barevnou teplotu obrázku, který neobsahuje odstíny červené, a pak obrázku obsahující odstíny červené. Na obrázky i LUT (velikost 33) aplikuji korekci s nastavenými hodnotami tak, jako na obrázku 28.



Obrázek 28: Aplikace barevné teploty na LUT

- Srovnání obrázků bez odstínů červené: průměrná vzdálenost RGB hodnot celého obrazu je $\delta rgb = 2,243964$.

Obrázek 29: První test korekce barevné teploty

- Srovnání obrázků s odstíny červené: průměrná vzdálenost RGB hodnot celého obrazu je $\delta rgb = 8,304681$ a průměrná vzdálenost RGB hodnot jen ve vyznačené ploše na obrázku 30 je $\delta rgb = 59,584689$.

Obrázek 30: Druhý test korekce barevné teploty

7.4.5 Zhodnocení

Z prvních tří testů jsem schopen usoudit, že zvolení velikosti LUT hraje významnou roli v přesnosti výsledných barev. Ve všech třech prvních testech je průměrná vzdálenost mezi LUT o velikosti 9 a 33 až 5× větší, tedy až na LUT o velikosti 9 je rozdíl oku těžce zpozorovatelný. Problémem malé LUT je, že se změny barevnosti projeví především i na barvách, které bý neměli být upravené.

Ve čtvrtém testu se taktéž ukázalo, že průměrná barevná vzdálenost nemusí mít vždy vypovídající hodnotu a to především, když se významně liší jen malá část barev. Dalším postřehem může být rychlost modifikace obrázku, kde až na třetí test je běžná aplikace rychlejší. To může být způsobeno aplikací parseru NCalc, který není napsán pro výpočetně náročné aplikace. Avšak i tato informace dává dobrou představu, že výhoda použití LUT, bude záviset na počtu použitých operací.

Jak je nastíneno výše, čtvrtým a posledním testem jsem dokázal, že LUT není všemocná. Hlavním důvodem proč čtvrtý test vyšel špatně pro červené odstíny je fakt, že k výpočtu se převádí HSL hodnoty na RGB, které se upravují tak, aby se výsledná barva blížila k vypočítané barvě (barvě absolutně černého tělesa zadané barevné teploty). To zapřičiňuje, že některé body (v mém případě červené odstíny) se posouvají k výslednému odstínu opačnou stranou než sousední body. Test tedy dobře posloužil k tomu, že výběr barevného prostoru, který by se měl zabalit do LUT, by měl záležet na barevných korekcích, které chci použít.

Závěr

Cílem této práce bylo popsat barevnou korekci pomocí trojrozměrné interpolace. Bylo potřeba si objasnit nad čím, a jak bude aplikovaná trojrozměrná interpolace. Výsledkem praktické části je program, na kterém si může kdokoliv, kdo má zájem o tématiku barevné korekce za pomocí LUT ozkoušet teorii nabytou čtením této práce.

Osobně jsem se v minulosti částečně zabýval úpravami fotografií a LUT vnímal jen jako předpřipravené filtry a nic více. Proto mě, i přes krušné začátky při vypracovávání této bakalářské práce, výběr daného tématu velice nadchl. Hlavním přínosem bylo vypracování interpolace a převodů, které jsem musel z neustálých zaokrouhlovacích chyb přepisovat, kde i malá chyba měla podstatný dopad na výsledek. Velice mě překvapila, výsledná přesnost, ať už malých, či velkých LUT. Důvodem, tak velké přesnosti je nejspíš použití LUT tvořené nad HSL prostorem, na radu odborného konzultanta pana Hejzlara, který mě tím ušetřil o spousty odchylek.

Na bakalářskou práci by se dalo navázat diplomovou prací, ve které bych se mohl pokusit o vytvoření bohatšího GUI, pro práci a zobrazení LUT, nebo vytvořit plugin, pracující s více druhy LUT, do Adobe Photoshopu.

Conclusions

The aim of this work was to describe color correction using three-dimensional interpolation. It was necessary to clarify what three-dimensional interpolation is and how would it be appliedt. The result of the practical part is a program where anyone who is interested in the topic of color correction with the help of LUT can test the theory acquired by reading this work.

Being a person who has worked with photo editing in the past and perceived LUT as just pre-made filters and nothing more, I was very excited about the topic, despite the difficult beginnings. The main benefit was the development of interpolation and transfers, which I had to rewrite from constant rounding errors, where even a small error had a tremendous impact on the result. I was very surprised by the resulting accuracy, whether small or large LUT. The reason for such a great accuracy is probably the use of LUT formed over HSL space, on the advice of expert consultant Mr. Hejzlar, who thus saved me of many rounding errors.

The bachelor's thesis could be followed by a master's thesis in which I could try to create a richer GUI to work and display LUT, or create a plugin working with multiple types of LUT, to Adobe Photoshop.

A Obsah přiloženého CD/DVD

Na samotném konci textu práce je uveden stručný popis obsahu přiloženého CD/DVD, tj. jeho závazné adresářové struktury, důležitých souborů apod.

`bin/`
: Adresář obsahuje spustitelný soubor programu.

`doc/`
: Adresář obsahuje dokumentaci jak v souboru pdf, tak její zdrojové soubory.

`src/`
: Adresář obsahuje zdrojový kód programu.

`readme.txt`
: Soubor readme.txt obsahuje postup spuštění programu. Dále obsahuje odkazy na veškeré cizí, převzaté materiály.

Reference

[1] Cambridge in colour - White balance. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://www.cambridgeincolour.com/tutorials/white-balance.htm

[2] Cambridge in colour - understanding gamma correction. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://www.cambridgeincolour.com/tutorials/gamma-correction.htm

[3] Henry R. Kang - Computational Color Technology, SPIE Publications, 2006. ISBN-13 978-0819461193

[4] Jaroslav Reichl and Martin Všetička - Záření absolutně černého tělesa [online] 2019-10-08 [cit. 2019-10-08]. Dostupné z: http://fyzika.jreichl.com/main.article/view/538-zareni-absolutne-cerneho-telesa

[5] Jiří Žára, Bedřich Beneš, Jiří Sochor, Petr Felkel - Moderní počítačová grafika, Computer Press Brno 2004. ISBN 80-251-0454-0.

[6] Julian Villegas, Michael Cohen - Synæsthetic Music or the Ultimate Ocular Harpsichord, ISBN-978-0-7695-2983-7 [online]. 2020-04-05 [cit. 2020-04-05]. Obrázek ve formátu PPM. Dostupné z: https://www.researchgate.net/figure/CIE-1931-Standard-Colorimetric-Observer-XYZ-functions-between-380-nm-and-780-nm-at-5-nm_fig2_4292613

[7] Lighting Design and Simulation Knowledgebase. Lighting Design Glossary. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://www.schorsch.com/en/kbase/glossary/contrast.html

[8] Matt Pharr, Randima Fernando - GPU Gems 2, Addison-Wesley Professional 2005. ISBN-13 978-0321335593

[9] Nvidia. GPU Gems - chapter 24. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://developer.nvidia.com/gpugems/GPUGems2/gpugems2_chapter24.html

[10] Oystein Bjorke - oxyPlot dokumentace. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://oxyplot.github.io/

[11] Paul Borke - Paulbourke.net interpolation methods, written in 1999. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: http://paulbourke.net/miscellaneous/interpolation/

[12] Pavel Tišnovský, root.cz - Grafický formát BMP. [online] 2019-06-08 [cit. 2020-05-03]. Dostupné z: https://www.root.cz/clanky/graficky-format-bmp-pouzivany-a-pritom-neoblibeny/

[13] Sebastien Ros - NCalc dokumentace. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://archive.codeplex.com/?p=ncalc

[14] Tanner Helland. Converting temperature (Kelvin) to RGB. [online] 2019-06-08 [cit. 2020-05-03]. Dostupné z: https://tannerhelland.com/2012/09/18/convert-temperature-rgb-algorithm-code.html

[15] Wikipedia. CIE 1931 color space. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://en.wikipedia.org/wiki/CIE_1931_color_space

[16] Wikipedia. CMYK. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://cs.wikipedia.org/wiki/CMYK

[17] Wikipedia. Color balance. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://en.wikipedia.org/wiki/Color_balance

[18] Wikipedia. Color grading. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://en.wikipedia.org/wiki/Color_gradingl

[19] Wikipedia. Color Space. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://en.wikipedia.org/wiki/Color_space

[20] Wikipedia. HSL and HSV. [online] 2019-06-08 [cit. 2019-06-08]. Dostupné z: https://en.wikipedia.org/wiki/HSL_and_HSV

[21] Wikipedia. Portable Network Graphics. [online] 2019-06-08 [cit. 2020-05-03]. Dostupné z: https://en.wikipedia.org/wiki/Portable_Network_Graphics

[22] Wikipedia. Portable Network Graphics. [online] 2019-06-08 [cit. 2020-05-03]. Dostupné z: https://en.wikipedia.org/wiki/Portable_Network_Graphics