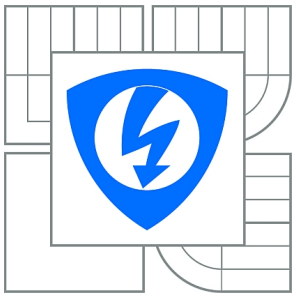
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ALGORITMY PŘEPOČTŮ GAMUTŮ VE SPRÁVĚ BAREV

GAMUT MAPPING ALGORITHMS IN COLOR MANAGEMENT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

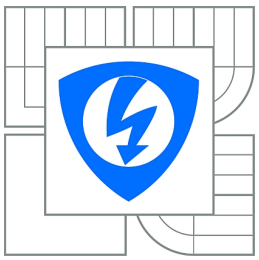
AUTOR PRÁCE — Bc. JAN SVOBODA

AUTHOR

VEDOUCÍ PRÁCE — Mgr. PAVEL RAJMIC, Ph.D.

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Jan Svoboda	***ID:***	125648
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Algoritmy přepočtů gamutů ve správě barev

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku barevných modelů a prostorů, měření barev a lidského vidění. Pochopte vztahy mezi prostory XYZ, LAB a CAM. Seznamte se s rozdíly mezi čtyřmi základními metodami pro přepočty gamutů: perceptuální, systostní, relativní a absolutní kolorimetrický. Seznamte se s měřicím hardware a software.

Z literatury nastudujte vybrané algoritmy přepočtu barevných gamutů, implementujte a ověřte v Matlabu. Srovnejte výstupy s obvyklými metodami (např. z Adobe Photoshopu).

DOPORUČENÁ LITERATURA:

[1] MOROVIČ, J. Color gamut mapping. Wiley, 2008.

[2] FRASER, B., MURPHY, C., BUNTING, F. Správa barev, Computer press, 2003.

[3] KAPLANOVÁ, M. a kol. Moderní polygrafie, Svaz polygrafických podnikatelů, 2010.

[4] HUNT, R., POINTER, M. Measuring color. Wiley, 2011.

Termín zadání:	10.2.2014	***Termín odevzdání:***	28.5.2014

Vedoucí práce: Mgr. Pavel Rajmic, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

ABSTRAKT

Práce se věnuje barvám – jejich reprezentaci v digitálních zařízeních, co nejvěrnějšímu zachování barev na rozdílných zařízeních a jakými postupy toho může být dosaženo. V první části práce stručně shrnuje poznatky o barvě a lidském vidění, rozvádí používané barevné modely a prostory a věnuje se hlavně těm nezávislým na zařízení. Gamut – rozsah barev, které je dané zařízení schopno zobrazit – se pro každé zařízení liší a v praxi je třeba zajistit správnou interpretaci nebo záznam barvy. Proto je stručně popsáno, jak funguje správa barev především z hlediska přepočtů gamutů a jakými způsoby jsou přepočty gamutů realizovány. V druhé části se práce zabývá již dvěma konkrétními algoritmy (HPMINDE a SCLIP) pro přepočty gamutů. Je popsána jejich implementace a ověření v prostředí MATLAB. Ve třetí, poslední části práce jsou uvedeny a diskutovány výsledky implementace těchto algoritmů v MATLABu a tyto výsledky jsou porovnány s běžně používanými přepočty gamutů (v Adobe Photoshop).

KLÍČOVÁ SLOVA

Barva, lidské vidění, kolorimetrie, správa barev, gamut, barevný model, barevný prostor, CIE XYZ, CIE LAB, CIE CAM02, ICC, způsoby vykreslení, MATLAB, sRGB, konvexní obálka, CIE LCh, přepočet, mapování barev, Delta E, HPMINDE, SCLIP, kompenzace černého bodu, BPC

ABSTRACT

The thesis deals with colors – their representation in digital devices and how to provide the best color preservation accross different devices. In the first part of the work, the knowledge of colors and human vision is briefly summarized. Then color models and color spaces are elaborated, mainly those device independent. Spectrum of colors viewable or printable on a device – the gamut – is different for every device and there's a need of precise reproduction or record of color. That's why the system of color management is described further and especially the gamut mapping approaches and algorithms are mentioned. In the second part of the work, the implementation of how two algorithms of color gamut mapping (HPMINDE, SCLIP) can be implemented in MATLAB is described. In the third and last part of the work, the results of implemented algorithms are presented and discussed. These results are compared to results of commonly used color gamut mapping technique (Adobe Photoshop).

KEYWORDS

Color, human vison, colorimetry, color management, gamut, color model, color space, CIE XYZ, CIE LAB, CIE CAM02, ICC, rendering intents, MATLAB, sRGB, convex hull, CIE LCh, color mapping, Delta E, HPMINDE, SCLIP, black point compensation, BPC

SVOBODA, Jan. *Algoritmy přepočtů gamutů ve správě barev*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 67 s. Vedoucí práce byl Mgr. Pavel Rajmic, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Algoritmy přepočtů gamutů ve správě barev“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Mgr. Pavlu Rajmicovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Technicka 12, CZ-61600 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Se správou barev se setkává každý dnes a denně, když pracuje s digitálními zařízeními, která zaznamenávají či zobrazují obraz – fotoaparát, skener, monitor, tiskárna, atd. Uživatel těchto zařízení nemusí o správě barev nic vědět, ale přesto uvnitř zařízení probíhají procesy, které zajišťují co nejvěrnější záznam či zobrazení barev. Lidské oko je schopno vidět určité spektrum barev a každé zařízení je také schopno zobrazit určité spektrum barev. Toto spektrum barev je možné číselně vyjádřit v barevném prostoru pomocí barevných modelů. Každé zařízení je však jiné, a proto když toto číslo zaznamenáme či zobrazíme, může mu ve skutečnosti odpovídat mírně odlišná barva. Červená bude pravděpodobně červenou, ale bude mít jiný odstín. Vyjádření všech barev, které je dané zařízení schopno zaznamenat či zobrazit, se nazývá gamut. Správa barev se pak zabývá přepočty mezi těmito gamuty jednotlivých zařízení tak, aby si barvy pokud možno vzájemně odpovídaly.

Práce je rozdělena na tři části. První z nich se zabývá správou barev – co tvoří barvu, jak ji vnímá člověk, jak se dá měřit, v jakých prostorech a jakými modely je možno ji číselně vyjádřit. Jak konkrétně probíhá správa barev v počítači či zařízeních a jak se pracuje s gamuty tam, kde nejsou stejné (např. kterou barvu by měla vytisknout tiskárna, která není schopna vytisknout svítivě zelenou, kterou zobrazuje monitor?). První část tedy uvádí nutný teoretický úvod a přehled k přepočtu gamutů. Druhá část práce se věnuje dvěma konkrétním algoritmům – HPMINDE a SCLIP. Nejprve jsou popsány nutné procedury úpravy obrazových dat v prostředí MATLAB, aby s těmito daty mohly algoritmy přepočtů gamutů pracovat. Následně je popsán princip obou algoritmů přepočtů gamutů a jejich implementace v prostředí MATLAB. V závěru druhé části je ještě popsána kompenzace černého bodu, která se v praxi používá spolu s přepočty gamutů. V poslední, třetí části práce, jsou uvedeny výsledky přepočtů gamutů pomocí implementovaných algoritmů a tyto jsou srovnány s výsledky z přepočtu gamutů v praxi používaném software – Adobe Photoshop.

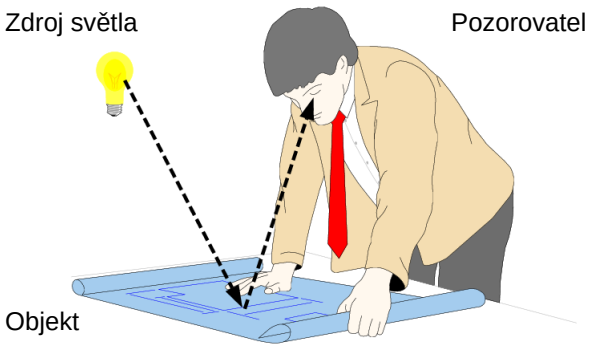
Barva samotná, správa barev a oblast přepočtů gamutů je velmi komplexní a složitá oblast, respektive používání správy barev již implementované v počítači může být jednoduché, řeší však mnoho složitých věcí. Do procesu správy barev vstupuje velké množství proměnných – zdroj světla, vlastnosti barevného objektu, pozorovatel a individuální odlišnosti, odlišnosti zařízení, zda jsou zařízení zkalibrována, odlišnosti v gamutech zařízení a přepočtech mezi nimi, atd. Přesto, že správa barev je na velmi pokročilé úrovni – byla vyvinuta vzájemnou spoluprácí výrobců zařízení, neexistuje zatím univerzální algoritmus nebo modul správy barev, který by vždy a správně bez zásahu člověka rozhodoval, kdy kterou metodu přepočtu gamutů a zobrazení použít. Jedná se o stále se rozvíjející oblast vědy a techniky. [2, 3]

1 SPRÁVA BAREV A PŘEPOČTY GAMUTŮ

1.1 Barva

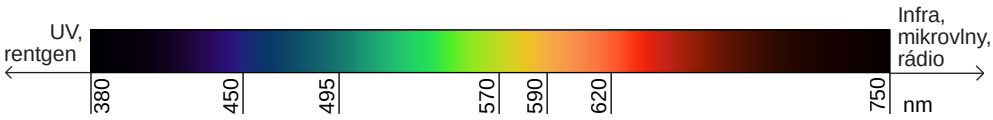
Barvu tvoří vnější podnět (fyzikální děj), ale i subjektivní výsledek vyhodnocení tohoto děje pomocí lidského oka a mozku [1]. V této úvodní kapitole budou zmíněny fyzikální aspekty barvy a v následující kapitole vizuální systém člověka.

Fyzikální děj, který vytváří barvu, je dán světelným zdrojem a vlastnostmi povrchu barevného objektu. Tuto takzvanou barevnou událost znázorňuje obrázek 1.1. Dá se tedy říci, že barva je světlo, že je vlastností objektu a také že vzniká v mysli pozorovatele. Všechna tři tvrzení jsou pravdivá, protože všechny tři složky hrají při barevné události roli. Barevná událost tak v sobě spojuje tři vědní obory - fyziku (zdroj světla – elektromagnetického záření), chemii (povrch a složení pozorovaného objektu) a biologii (funkce a části lidského oka a mozku) [2].



Obr. 1.1: Barevná událost a její tři složky – zdroj světla, vlastnosti objektu a lidské vidění. Převzato a upraveno z [4].

Z fyzikálního hlediska je tedy barva vždy světlem – elektromagnetickým zářením o určité vlnové délce. Viditelná část elektromagnetického záření s vlnovými délkami v nanometrech je znázorněna na obr. 1.2. Newton již v roce 1730 publikoval objev, že bílé světlo lze rozložit na jednotlivé barvy pomocí skleněného hranolu (disperze). Dokázal, že v bílém světle jsou zastoupeny všechny vlnové délky.



Obr. 1.2: Spektrum viditelného světla. Převzato a upraveno z [6].

To je patrno i z popisu světla pomocí spektrální charakteristiky, která vyjadřuje energii (množství světla) vyzářenou na každé vlnové délce. Stejný průběh bude mít

křivka odrazivosti, podíváme-li se na světlo jako na barvu – vlastnost barevného objektu. Při popisu ideálně bílého světla pomocí této křivky by byly stejnou měrou zastoupeny všechny vlnové délky, zatímco při popisu určité barvy by obsahovala pouze vlnové délky v okolí dané barvy, např. u zelené barvy budou ve velké míře zastoupeny složky kolem 550 nm a ostatní složky budou téměř nulové. Barva tak závisí na poměru a zastoupení jednotlivých vlnových délek. [1, 2]

Zdroj světla charakterizuje jeho barevná teplota – spektrální složky, které obsahuje. Organizace CIE (*Commission Internationale de l'Éclairage* – Mezinárodní komise pro osvětlování) [8] specifikovala třídy zdrojů světla. Teplota barvy zdroje světla (teplota chromatičnosti) se podle těchto specifikací udává absolutní teplotou černého tělesa v Kelvinech. Černé těleso je objekt (v reálném světě neexistuje), jehož barva nezávisí na jeho složení, ale pouze na jeho teplotě. Zvyšováním teploty je tak těleso nejdříve červené (3000 K, více energie vyzařují delší vlnové délky), potom bílé (5000 K, vlnové délky rovnoměrně zastoupeny) a s vyššími teplotami modré (10 000 K, více energie vyzařují krátké vlnové délky).[1] Komise CIE specifikovala třídy zdrojů světla A až F, které popisují typické charakteristiky různých zdrojů světla a svítidel od wolframové žárovky (3000 K) přes denní světlo (6500 K) po teoretické zdroje, které se používají ve výpočtech a fluorescenční svítidla jako např. zářivky. Pro správu barev je důležitá třída D, která specifikuje různé varianty denního slunečního světla. Nejčastěji používanými zdroji světla ve správě barev jsou tak D50 (5000 K) a D65 (6504 K).[2] Příklad spektrální charakteristiky zdroje světla (žárovky, teplota barvy nízká, do červena) je vidět v prvním grafu na obr. 1.3.

Povrch objektu, u kterého barvu pozorujeme, hraje také velkou roli. To, jak modifikuje světlo ze zdroje, se dá vyjádřit různými charakteristikami, jako je odrazivost (určité vlnové délky objekt odráží, jiné absorbuje), propustnost (světlo jím prochází, průsvitnost), rozptyl, fluorescence (schopnost absorbovat záření o vyšší vlnové délce

Obr. 1.3: Funkce, které znázorňují proces vnímání barvy. Na ose x je až na poslední graf vlnová délka. První graf je výkonovou spektrální charakteristikou zdroje světla, druhý je spektrem odrazivosti objektu, třetí představuje vynásobené dva předchozí. Toto odražené světlo vstupuje do oka skrze citlivost čípků (třetí graf), které světlo absorbují (čtvrtý graf, integrál křivek citlivosti). Převzato a upraveno z [4]

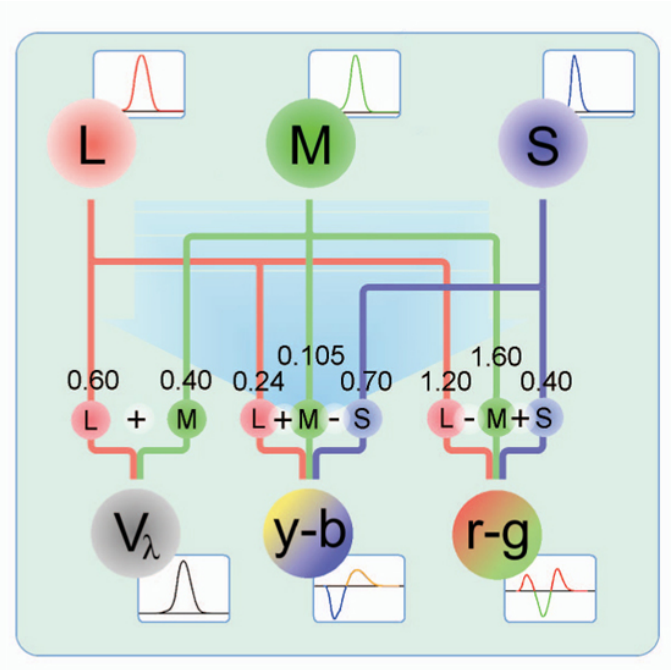
a vyzařovat je jako záření o nižší vlnové délce – používá se např. k bělení papíru a ztěžuje tak správu barev). [2] [1] Tyto vlastnosti povrchu objektu se dají popsat křivkami jako například spektrem odrazivosti jako je druhý graf na obr. 1.3. Když se toto spektrum odrazivosti vynásobí se spektrem zdroje světla, vyjde výsledné spektrum vlnových délek, které dopadají na sítnici lidského oka. Znázorňuje je třetí graf na obrázku 1.3 a velmi dobře je také možno proces znázornit pomocí apletu, který ve své diplomové práci vytvořil Stecík (2013) [9].

1.2 Lidské vidění

Vizuální systém člověka je tvořen okem a nervem, který z něj ústí do mozku, kde dochází k interpretaci informací. Lidské oko vidí trichromaticky – všechny barvy skládá ze 3 základních barev – červené, zelené a modré, ale tím, jak nervy informaci z oka zpracují, dochází ještě k úpravě těchto vjemů a do mozku jdou pak informace ještě v jiné podobě, viz dále. Na sítnici (vnitřní zadní straně) oka jsou umístěny dva druhy receptorů – tyčinky a čípky. Tyčinky umožňují pouze černobílé vidění za nízkého osvětlení, při běžném denním světle jsou oslepeny, nevyužity. Čípky jsou citlivé na běžné osvětlení, a je jich mnohem více než tyčinek (asi 120 milionů čípků a 6 milionů tyčinek) a jsou nejvíce koncentrovány v takzvané žluté skvrně, což je místo s nejostřejším viděním a základním barevným vjemem. Tři typy čípků umožňují trichromatické vnímání: čípků typu L (long – vnímají dlouhé vlnové délky a tedy červenou barvu) je nejvíce, čípků typu M (medium – vnímají střední vlnové délky a tedy zelenou barvu) je méně a čípků typu S (short – krátké vlnové délky, modrá barva) je nejméně. Citlivost čípků znázorňuje čtvrtý graf na obr. 1.3. Poměr čípků L:M:S je individuální, asi 6:3:1. [1, 2]

Teorii trichromatického vidění rozvinuli Young a Helmholtz (19. st.), později Hering (konec 19. st.) zkoumal zrakové vady a přišel s teorií protikladného vnímání. K tomu dochází tím, jak nervy sbírají informace z čípků, jak je celé ústrojí zapojeno. Člověk si totiž nedokáže představit červenou s nádechem zelené nebo žlutou s nádechem modré. Dochází zde ke sčítání a odčítání informací z čípků a výsledkem jsou opět tři složky, tentokrát ale protikladného charakteru: černá–bílá (světlost V_λ), žlutá–modrá (y–b) a červená–zelená (r–g). Barvy jsou tedy reprezentovány jedním luminiscenčním kanálem (světlost) a dvěma chromatickými, které udávají barvu. Literatura udává, že luminiscenční kanál V_λ je dán součtem signálů z čípků L a M [1, 2, 5], někdy součtem signálů ze všech tří typů čípků [3]. Chromatický kanál červené a zelené (y–b) je dán odečtením signálu z čípků M (zelená) od signálu z čípků L (červená) [2, 3], jeden zdroj uvádí, že se k nim ještě přičte signál z čípků S [5]. Chromatický kanál žluté a modré (y–b) je dán odečtením signálu z čípků S (modrá)

od součtu signálů z čípků L a M [2, 3, 5]. Signály se sčítají s různými koeficienty, výsledné protikladné signály jsou lineární kombinací vjemů z čípků L, M, S. Na obrázku 1.4 je patrný celý proces včetně koeficientů, jedná se o model transformace podle Inglinga a Tsou (1977) [5]. Na obrázku jsou naznačeny také spektrální charakteristiky citlivosti čípků na vstupu transformace a na výstupu výsledné spektrální charakteristiky citlivosti protikladných kanálů. Na tomto principu jsou postaveny barevné prostory CIE XYZ (souřadnice vypočteny pomocí hodnot z čípků L, M, S) a CIE LAB (vyjadřují protikladné hodnoty), které budou rozvedeny dále.



Obr. 1.4: Zpracování informací z čípků, převzato z [5].

1.3 Měření barvy

Měření barvy se děje v podstatě měřením světla, jeho spektrální charakteristiky. Barva jako taková se dá pojmenovat, nebo vyjádřit čísly až v barevném prostoru. Nejjednodušššími přístroji jsou denzitometry, které měří hustotu světla – míru, s jakou odrazný povrch absorbuje či propouští světlo. Úplné spektrální vlastnosti povrchu – kolik světla určité vlnové délky povrch propouští či absorbuje – je možno změřit pomocí spektrofotometru. Nejpodobněji lidskému vidění lze barvu změřit kolorimetrem, kdy měření probíhá pomocí barevných filtrů a jehož výstupem jsou čísla, která modelují reakci čípků v lidském oku. [2]

V rámci této práce bylo kolorimetrickou sondou naměřeno několik ICC profilů zařízení, konkrétně monitorů (viz dále správa barev, kap. 1.6). Tyto profily obsahují

gamuty zařízení vyjádřené v barevném prostoru (viz dále kap. 1.5.2). K naměření profilů byla použita kolorimetrická sonda i1 Pro od firmy X-Rite [7].

1.4 Kolorimetrie

Kolorimetrie se zabývá předpovídáním barevných shod, jak by je vnímal typický pozorovatel. Cílem kolorimetrie je vytvoření numerického modelu k porovnávání barev, na jehož základě by se dalo rozhodnout, zda dochází k metamerii či nikoliv [2]. *Metamerie* je jev, kdy stejné barvy můžeme dosáhnout různými způsoby – např. pomocí malířských barev, tiskárny či monitoru. Člověku se tyto barvy jeví totožné, ale kdyby byly změřena spektrofotometrem, ukázalo by se, že jejich spektrální charakteristiky se liší. Metamerie je tedy výhodou i nevýhodou zároveň. Výhodou proto, že je vůbec možno dosáhnout zobrazení totožné barvy jinou technologií, než jak se vyskytuje v přírodě, a nevýhodou proto, že se každá z takto uměle dosažených barev chová z fyzikálního hlediska (spektrum) jinak – např. pod jiným světelným zdrojem je už pozorovatel nevnímá jako totožné, ale vidí mezi nimi rozdíl [1]. Požadavky na kolorimetrický model jsou, aby v případech, kdy typický pozorovatel vidí shodné barvy, byly tyto reprezentovány shodnými číselnými hodnotami, a v případě, kdy pozorovatel vidí odlišné barvy, aby byly reprezentovány odlišnými číselnými hodnotami. Zároveň musí být možno vypočítat číslo, které vyjadřuje barevnou odlišnost, to jest o kolik se barvy liší [2].

Současná kolorimetrie a prakticky všechny systémy pro správu barev vycházejí z kolorimetrického systému CIE [8, 2]. Organizace CIE definovala kromě tříd zdrojů světla zmíněných v kapitole 1.1 také *standardního pozorovatele*, který představuje úplnou tristimulační reakci běžného lidského pozorovatele, tedy sadu všech barev, jež jsme schopni vidět. Na přelomu 20. a 30. let 20. st. ji naměřili Guild a Wright na skupině pozorovatelů a v roce 1931 byla organizací CIE ustanovena jako *1931 CIE standardní pozorovatel*. Barevný vjem byl měřen v zorném poli 2°, což odpovídá nejostřešímu vnímání, kdy paprsky dopadají do žluté skvrny na sítnici oka. Později bylo zjištěno, že ve žluté skvrně je nízká koncentrace čípků typu S, které jsou lépe zapojeny až při širším zorném poli a byl tedy změřen 1964 CIE standardní pozorovatel při zorném poli 10° [1]. Používanější ve správě barev je však stále starší standard z roku 1931, pouze když se počítá se zorným polem pozorovatele, vyšším než 4°, doporučuje se novější standard. [2] .

Hodnoty barev standardního pozorovatele specifikovaného organizací CIE odpovídají přibližně integrálu křivky spektrální citlivosti jednotlivých typů čípků L, M, S v lidském oku, jak to zobrazuje poslední graf z obrázku 1.3. Organizace CIE definovala barevný prostor CIE XYZ, kde hodnoty X, Y, Z představují mírně upravené

hodnoty L, M, S (více v kap. 1.5.2). Prostor XYZ představuje trojrozměrný ortogonální prostor, v němž množina bodů, kterou udává CIE 1931 standardní pozorovatel, tvoří gamut (rozsah) lidského barevného vidění. [1]

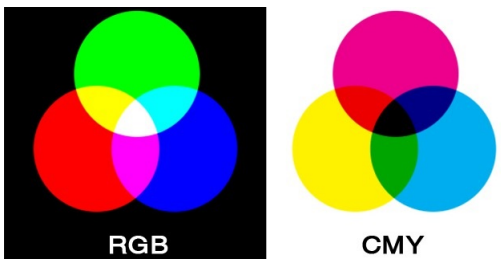
V takovém prostoru, kde body reprezentují barvy, je možno barvy mezi sebou nezávisle porovnávat prostým spočtením euklidovské vzdálenosti mezi body. Tato vzdálenost mezi barevnými body se obvykle značí ΔE, více v kap. 1.5.2 [2]. Prostor CIE XYZ vykazuje určité zkreslení těchto barevných vzdáleností (skutečné vzdálenosti bodů někdy neodpovídají subjektivnímu rozdílu mezi barvami jak je vnímá člověk), proto byl organizací CIE v r. 1976 definován prostor CIE LAB, který se snaží toto zkreslení eliminovat [2]. Tento prostor vychází z protikladného vnímání barev člověkem a na osách tedy nejsou hodnoty odpovídající tristimulaci čípků, ale hodnota jasu, a pak dvě chromatické složky červená–zelená a modrá–žlutá. Více k těmto barevným prostorům v následující kapitole.

1.5 Barevné modely a prostory

Jak již bylo nastíněno, barevné modely a prostory slouží k zaznamenání barvy, ať už k její interpretaci nebo k porovnání s jinou barvou. Díky nim je možno barvy číselně vyjádřit a uložit v počítači. *Barevný model* je abstraktní matematická struktura, která popisuje barvy jako n-tice čísel a definuje princip záznamu barev. *Barevný prostor* představuje už konkrétní interpretaci barevného modelu pro konkrétní pozorovací podmínky (např. teplotu bílé) nebo přímo konkrétní zařízení [10]. Barevný prostor vyjadřuje všechny barvy, které je zařízení schopno zobrazit či zaznamenat, *gamut* pak představuje hranice tohoto barevného prostoru, jeho obálku.

1.5.1 Barevné modely závislé na zařízení

Dvěma základními barevnými modely, které jsou závislé na zařízení, jsou RGB a CMYK. Závislé na zařízeních jsou proto, že skutečná barva, kterou získáme při stejných hodnotách RGB nebo CMYK, se na jednotlivých zařízeních může lišit[2].



Obr. 1.5: Aditivní model RGB a subtraktivní model CMY. Převzato z [11].

V modelu RGB představuje dle angličtiny R červenou, G zelenou a B modrou a tyto barvy jsou v počítači vyjádřeny čísly 0 až 255 (1 bajt), takže barvu v tomto modelu vyjadřují tři čísla. Jedná se o model aditivní, barvy se sčítají, nulová hodnota všech složek představuje černou. Model si lze představit jako tři zdroje světla dopadající na černé plátno, součet všech tří barevných složek dá bílou barvu, viz obr. 1.5. Tento model se používá v zařízeních, které vyzařují světlo – zobrazovací zařízení jako monitory, projektory, ale i fotoaparáty. Jeho protikladem je model CMY, resp. CMYK, kde C představuje azurovou (cyan), M purpurovou (magenta), Y žlutou a K černou (poslední písmeno black) – ta se z určitých důvodů v praxi používá jako čtvrtá barva. Analogicky zápis barvy v tomto modelu tvoří tři, resp. čtyři hodnoty 0 až 255. Jde o subtraktivní model, barvy se odčítají a základní barvy jsou barvami doplňkovými do bílé k RGB. Lze si jej představit jako míchání barev na bílém papíře, který světlo nevyzařuje, ale odráží, viz obr. 1.5. Nulové hodnoty zde představují bílou a maximální hodnoty černou. Tento model používají všechna tisková zařízení. [1, 11]

1.5.2 Barevné modely nezávislé na zařízení

Aby se dalo v praxi dosáhnout věrné shody barev na zobrazovacím i tiskovém zařízení, je nutno barvu definovat nezávisle na zařízení. K tomu slouží barevné modely nezávislé na zařízení, které se snaží vyjádřit barvu tak, jak ji vidí člověk, a používají se ve správě barev a při přepočtech gamutů. Následuje vysvětlení principů modelů CIE XYZ, CIE LAB včetně rovnic přepočtů mezi nimi, a také je stručně zmíněn novější model CIE CAM02.

Barevný model CIE XYZ a CIE xyY

Barevný model CIE XYZ vychází z charakteristik CIE 1931 pozorovatele a číselné hodnoty X, Y, Z popisují absorpci čípků L, M, S na základě modelu CIE 1931 pozorovatele. Hodnota Y v tomto modelu představuje jas [1]. Rovnice (1.1) až (1.3) popisují získání hodnot X, Y, Z [1]:

$$X = k \sum_{\lambda} S_\lambda R_\lambda \overline{x}_\lambda \Delta\lambda, \tag{1.1}$$

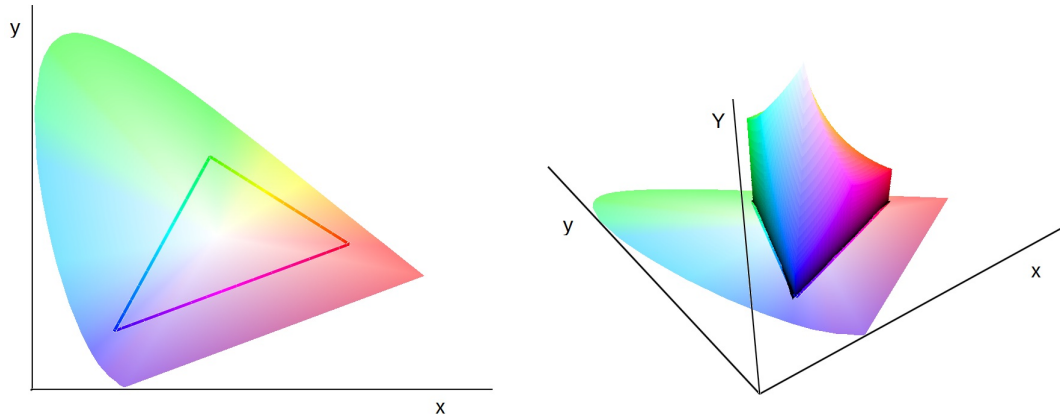
$$Y = k \sum_{\lambda} S_\lambda R_\lambda \overline{y}_\lambda \Delta\lambda, \tag{1.2}$$

$$Z = k \sum_{\lambda} S_\lambda R_\lambda \overline{z}_\lambda \Delta\lambda, \tag{1.3}$$

$$k = \frac{100}{\sum_\lambda S_\lambda \overline{y}_\lambda \Delta\lambda}. \tag{1.4}$$

S_λ je osvětlení, R_λ faktor odrazivosti, $\overline{x}_\lambda$, $\overline{y}_\lambda$ a $\overline{z}_\lambda$ jsou barevným vjemům odpovídající funkce, které definuje standardní pozorovatel. Sumy představují součet přes vlnové

délky, což nahrazuje integrál, plochu pod křivkou. Dle specifikací CIE má být odstup vlnových délek $\Delta\lambda = 1$ nm, ovšem v praxi při použití spektrofotometru se měří na intervalech 10 až 20 nm a využívá se interpolace chybějících hodnot. Každá hodnota je ještě násobena normalizačním koeficientem k dle rovnice (1.4), který upravuje hodnoty tak, aby maximální hodnota jasu Y byla 100, což odpovídá v tomto modelu bílému bodu [1]. Tyto rovnice platí pro světlo odražené a přijaté. V případě měření zobrazovacího zařízení, jako např. monitoru, by hodnoty nebyly násobeny faktorem odrazivosti R_λ [12].



Obr. 1.6: Chromatický diagram (vlevo) a gamut v modelu CIE xyY (vpravo). Naměřeno kolorimetrickou sondou [7] a zobrazeno pomocí programu Color Think Pro [13].

Tyto tři hodnoty je možno zobrazit jako bod v trojrozměrné kartézské soustavě souřadnic. Díky tomu, že hodnota Y představuje jasovou složku, je možno tyto tři souřadnice X, Y, Z upravit tak, aby se gamut dal zobrazit v dvojrozměrné rovině. Jde o *chromatický diagram* a je možno jej vykreslit pomocí hodnot x a y, které jsou vypočteny následovně [1]:

$$x = X/(X + Y + Z), \tag{1.5}$$

$$y = Y/(X + Y + Z). \tag{1.6}$$

Dochází zde však k nutné redukci informací, protože se tři proměnné transformují do dvou. Na obr. 1.6 vlevo je znázorněn chromatický diagram. Jedná se o rozsah lidského vidění dle CIE 1931 pozorovatele a trojúhelník uprostřed představuje gamut zařízení, zde konkrétně displej notebooku Lenovo ThinkPad Edge 0217-2NG. Je-li k hodnotám x a y přidána také původní složka jasu Y, je možno barevný prostor zobrazit v trojrozměrném prostoru, kdy svislá osa Y představuje jas a osy x a y chromatické složky. Vzniká tak barevný model xyY a gamut stejného notebooku je možno vidět na obr. 1.6 vpravo. Při srovnání obou obrázků je patrné, že chromatický

diagram (vlevo) je výsledkem pohledu shora na barevný prostor v modelu xyY a gamut vyjádřený pouze v chromatickém diagramu je tak velmi nepřesný, pouze orientační.

Barevný model CIE LAB

Tento barevný model specifikovala CIE v r. 1976 spolu s CIE LUV, který se ale používá čím dál méně. CIE LAB vychází matematicky z předchozího CIE XYZ, ale navazuje na protikladný systém zpracování barvy u člověka. I tento model umožňuje zobrazení v trojrozměrné kartézské soustavě souřadnic, ovšem osy jsou označovány L^*, a^*, b^*. L^* představuje pouze jasovou složku, ostatní jsou chrominanční: a^* představuje červenou-zelenou a b^* žlutou-modrou. Tyto hodnoty jsou vypočteny z hodnot X, Y, Z následujícím způsobem [1]:

$$L^* = 116(Y/Y_n)^{1/3} - 16, \tag{1.7}$$

$$a^* = 500\left[(X/X_n)^{1/3} - (Y/Y_n)^{1/3}\right], \tag{1.8}$$

$$b^* = 200\left[(Y/Y_n)^{1/3} - (Z/Z_n)^{1/3}\right], \tag{1.9}$$

kde X_n, Y_n a Z_n jsou hodnoty referenčního bílého bodu. Pro hodnoty X/X_n, Y/Y_n a Z/Z_n menší než 0,01 platí:

$$L^* = 116\left[f(Y/Y_n) - 16/116\right], \tag{1.10}$$

$$a^* = 500\left[f(X/X_n) - f(Y/Y_n)\right], \tag{1.11}$$

$$b^* = 200\left[f(Y/Y_n) - f(Z/Z_n)\right]. \tag{1.12}$$

Zde platí, že $f(Y/Y_n) = (Y/Y_n)^{1/3}$ pro Y/Y_n větší než 0,008856,

$$f(Y/Y_n) \;=\; 7,787(Y/Y_n) \;+\; 16/116$$

pro Y/Y_n menší nebo rovno 0,008856. Podobně pro $f(X/X_n)$ a $f(Z/Z_n)$ [1]. Chromatičnost C^*_{ab} a odstín h_{ab} jsou z hodnot a*, b* vypočteny následovně [1]:

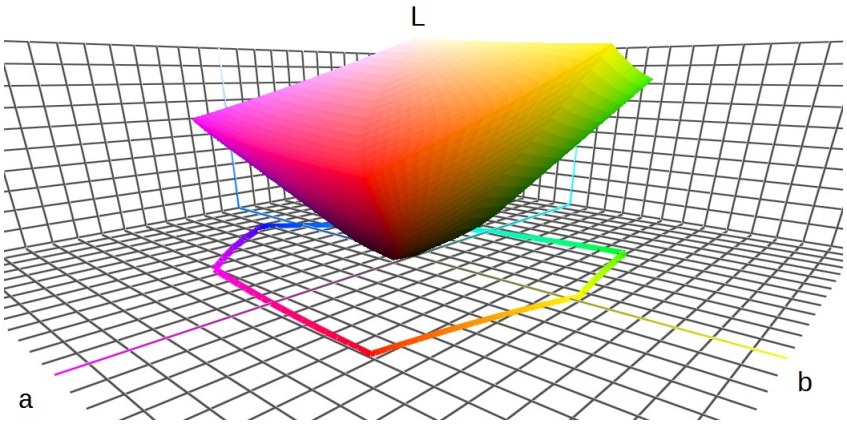
$$C^*_{ab} = \sqrt{a^{*2} + b^{*2}}, \tag{1.13}$$

$$h_{ab} = \arctan(b^*/a^*). \tag{1.14}$$

Na obrázku 1.7 je zobrazen v modelu CIE LAB stejný gamut jako na obrázku 1.6. Důvodem, proč CIE specifikuje v průběhu let nové barevné modely, je snaha, aby barevná odchylka ΔE byla v celém barevném prostoru rovna jedné, pokud se jedná o člověkem právě postřehnutelný rozdíl. Jde o to, aby byl prostor uniformní [14], jednolitý, aby byla barevná odchylka konzistentní v celém prostoru. Dle specifikace CIE z r. 1976 lze barevnou odchylku ΔE spočítat jako euklidovskou normu rozdílu barev:

$$\Delta E = \sqrt{\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}}. \tag{1.15}$$

V průběhu času se však ukázalo, že ani prostor CIE LAB není dostatečně uniformní, v některých případech je právě postřehnutelná odchylka i 2,3 ΔE [15]. Proto CIE později definovala nové způsoby výpočtu barevné odchylky [14].



Obr. 1.7: Gamut notebooku Lenovo v modelu CIE LAB. Zobrazeno pomocí [13].

Barevný model CIE CAM02

Tento model je jedním z nejnovějších modelů organizace CIE, byl publikován v roce 2002 a je používán k přepočtu gamutů ve správě barev Windows Vista a novějších (Windows Color System) [16]. CIE CAM02 bere v úvahu více proměnných, které do procesu vnímání barev člověkem vstupují – chromatickou adaptaci (žlutý citrón vnímáme jako žlutý i pod zeleným osvětlením), adaptaci na úroveň okolního osvětlení a vychází z modelu CIE CAT, který definuje šest matematicky popsatelných vlastností barvy. Kromě barevného stimulu se zde bere v potaz pozadí barevného stimulu a ještě okolí pozorování. Stejně jako u modelu CIE LAB, i zde je možno body zobrazit v systému kartézských souřadnic, osy se označují J, a, b, mají stejný význam jako u CIE LAB (J odpovídá L^*), ale výpočet hodnot je složitější a lépe zahrnuje výše zmíněné vlastnosti lidského vidění (kterých je využito například i u optických klamů). Model CIE CAM02 se chová uniformněji než model CIE LAB, barevná odchylka ΔE se zde skutečně rovná 1 při právě postřehnutelném rozdílu mezi barvami [17]. Morovič v knize Color Gamut Maping uvádí tento prostor jako jeden z nejlepších co se týče predikce barvy a příklady algoritmů přepočtů gamutů v této knize uvádí v tomto modelu [3].

1.6 Správa barev

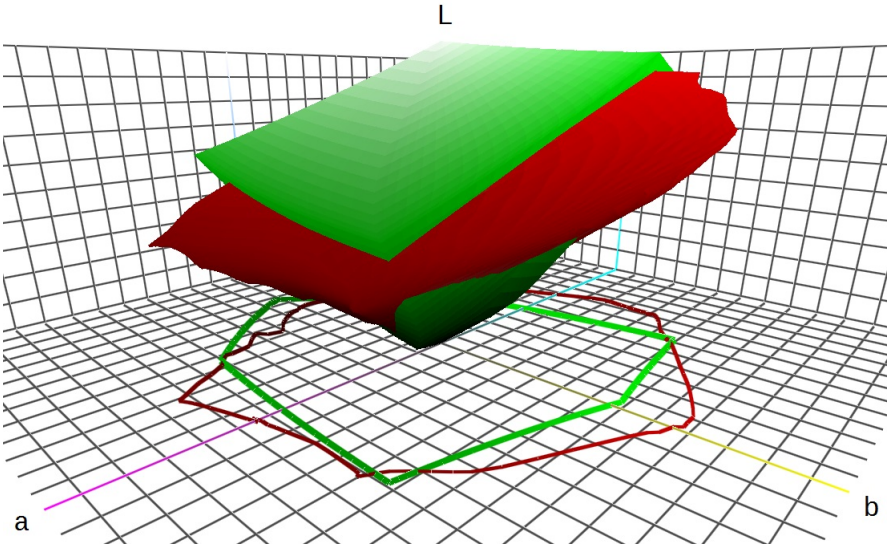
Správa barev je pojem, jímž se označuje systém v počítači nebo zařízeních, díky nemuž je možno dosahovat cílů kolorimetrie – barevných shod. Systém správy barev přiřazuje hodnotám RGB a CMYK jednoznačný význam (v modelech CIE XYZ či CIE LAB) a mění hodnoty RGB či CMYK, které jsou odesílány do různých zařízení (monitor, tiskárna) tak, aby došlo k barevné shodě či k zachování vztahů mezi barvami (viz způsoby vykreslení v další kapitole) [2]. Nutnost takových přepočtů je patrná z obrázku 1.8, na němž je ukázán rozdíl mezi gamutem displeje notebooku Lenovo, který je již na obrázku 1.7 a gamutem tiskárny Canon Pro 9000. Správa barev řeší problém, jak naložit s patrnými neshodami obou gamutů a přitom zachovat barvy co nejvěrněji. Nejrozšířenějším systémem správy barev je systém ICC, který definovalo sdružení ICC (Internationtal Color Consorcium), což je sdružení většiny velkých výrobců zařízení, kteří se rozhodli za účelem správy barev spolupracovat [18]. Systém správy barev dle specifikací ICC se dělí na čtyři základní části – profily ICC, prostor propojení profilů PCS, modul správy barev CMM a způsob vykreslení, jemuž se věnuje následující kapitola a je nejblíže tématu této práce. [2]

Profily ICC jsou soubory, které patří vždy ke konkrétnímu zařízení, obsahují informace o vztahu mezi RGB či CMYK hodnotami zařízení a hodnotami barev v nezávislém prostoru CIE XYZ či CIE LAB [2]. Profil obsahuje přesně specifikovanou hlavičku a samotná data. V hlavičce jsou uloženy informace o jaké zařízení se jedná, o barevném prostoru zařízení, preferovaném CMM modulu, do kterého prostoru se má převádět v PCS a použitý vztažný zdroj světla. Převod v PCS je pak možný dvěma způsoby – pomocí vyhledávání v tabulce, nebo přepočtem pomocí matice. Preferován je tabulkový převod, pokud to profil umožňuje. Profil neobsahuje informace o všech barvách, protože by pak byl datově příliš objemný. Obsahuje tedy pouze důležité body, které jsou v modulu CMM interpolovány, aby byla získána celá charakteristika. Soubor profilu musí mít malý datový objem, aby mohl být součástí souborů fotografií či dokumentů [19].

Propojení profilů PCS (Profile Connection Space) je část, která je jádrem správy barev, protože se jedná o převod z profilu vstupního přes nezávislý barevný prostor do profilu výstupního. Bez správy barev by muselo být mezi výrobci a zařízeními definováno obrovské množství převodů, ale díky tomuto převodu přes nezávislý prostor je proces zjednodušen a jednoznačně definován [2].

Modul správy barev CMM (Color Management Module) vykonává správu barev, je to software, který provádí přepočty převodu v PCS, pracuje tedy s daty o barvách uložených v profilu zařízení a provádí výpočty na základě způsobu vykreslení [2]. Díky tomu, že ICC systém správy barev specifikovalo, mohou různí výrobci nabízet své vlastní moduly CMM a to se také děje. Odlišné moduly CMM nabí-

zejí například výrobci Adobe, Agfa, Apple, Heidelberg, Kodak, X-Rite [2]. Modul CMM může barvy z nezávislého prostoru CIE XYZ či CIE LAB převést i do jiného barevného prostoru, kde může aplikovat přepočty mezi gamuty [2]. Např. modul CMM správy barev ve Windows Vista a novějších (WCS – Windows Color System), převádí gamuty do již zmiňovaného prostoru CIE CAM02, který má dobrou uniformitu a predikci barev na základě lidského vidění. V tomto prostoru jsou gamuty přepočteny a pak probíhá převod zpět do jednoho z prostorů PCS a pak podle profilu zařízení na konkrétní hodnoty RGB či CMYK [16]. Běžné operační systémy umožňují volbu preferovaného modulu CMM.



Obr. 1.8: Gamut displeje notebooku (zelený) a tiskárny (červený), zobrazeno v [13].

1.7 Přepočty gamutů ve správě barev

Téměř vždy nastává situace, že gamuty zařízení jsou rozdílné, podobně jako na obr. 1.8. Je tedy nutno nějakým způsobem gamuty přepočítat, převést rozsah barev jednoho zařízení do rozsahu barev druhého zařízení. Ve specifikacích ICC se tyto přepočty nazývají *způsoby vykreslení* a jsou definovány celkem čtyři: perceptuální, sytostní, relativní kolorimetrický a absolutní kolorimetrický [2]. V konkrétní implementaci ICC správy barev jsou tyto čtyři způsoby vykreslení zastoupeny již konkrétními algoritmy, ve specifikacích ICC se však jedná o reprodukční záměry, které vycházejí z některých reprodukčních záměrů tak, jak je specifikoval Hunt (1970), a jež jsou všeobecně uznávány [3]. Nyní podrobněji ke způsobům vykreslení dle ICC:

Perceptuální způsob vykreslení se snaží o zachování vzhledu všech barev jako celku tím, že mění všechny barvy zdrojového prostoru tak, aby odpovídaly nějakým barvám prostoru cílového. Nedochází k ořezání barev mimo cílový gamut, ale ke kompresi zdrojového gamutu tak, aby se vešel do cílového gamutu a vztahy mezi barvami byly zachovány. Výhodou zde je právě zachování vztahů mazi barvami, na které je lidské oko citlivé. Tento způsob je využíván u obrazů, které obsahují velké množství barev ležících mimo gamut cílového zařízení. [2]

Sytostní způsob vykreslení se snaží o zachování sytosti barev, ne jejich přesného odstínu či hodnoty. Nejsytější barvy ze zdrojového gamutu jsou mapovány na nejsytější barvy cílového gamutu. Tento způsob vykreslení je vhodný spíše pro diagramy, grafy, mapy se znázorněním výšek apod. Barevná reprodukce zde záměrně není přesná. Spolu s předchozím perceptuálním způsobem vykreslení se jedná o kompresi gamutu, následující kolorimetrické způsoby gamut ořezávají. [2]

Relativní kolorimetrický způsob vykreslení bere v potaz skutečnost, že naše oči se přizpůsobí bílé barvě daného média. Bílá zdrojového prostoru je tedy mapována na bílou cílového prostoru a všechny barvy jsou reprodukovány přesně, ve vztahu k bílému bodu. Barvy, které leží mimo gamut cílového prostoru jsou ořezány, neboli nahrazeny nejbližšími barvami na obálce gamutu. Způsobů, jakými probíhá projekce barev mimo cílový gamut na obálku cílového gamutu, je velké množství a toto řeší již konkrétní algoritmy. Tento způsob vykreslení umožňuje zachovat velké množství barev zdrojového obrazu a je velmi používán. [2]

Absolutní kolorimetrický způsob vykreslení je stejný jako předchozí v tom, že barvy ležící mimo cílový gamut jsou vhodným způsobem nahrazeny nejbližšími barvami v cílovém gamutu. Liší se však tím, že bílý bod zůstává zachován, není mapován na bílý bod cílového prostoru. Všechny barvy včetně bílé jsou tak absolutně přesně zachovány, může se to však u bílé barvy projevit např. nádechem azurové apod. Tento způsob vykreslení je používán např. u simulace výstupu tiskárny. [2]

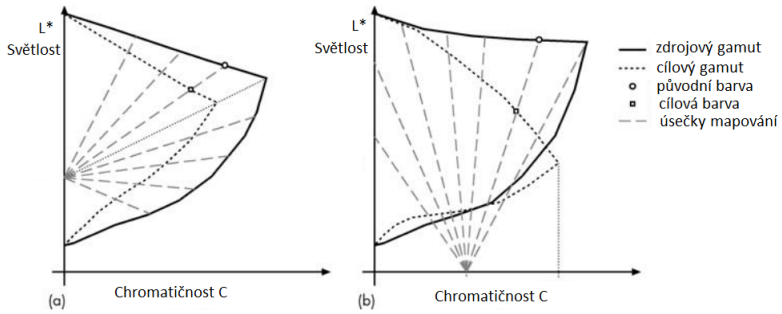
1.8 Algoritmy přepočtů gamutů

Algoritmy přepočtů gamutů mohou být rozděleny na základě vztahu mezi zdrojovým a cílovým gamutem a na základě typu informací, které jsou k přepočtu využity. V případě, že zdrojový a cílový gamut nemají společný průnik, algoritmus překládá jednotlivé barvy. V případě, kdy se gamuty částečně překrývají, algoritmus musí jednotlivé barvy adaptovat. Nejčastějším případem je však redukce jednotlivých barev, ke které dochází v případě, kdy cílový gamut je menší než zdrojový. V tomto přápadě se kromě redukce jednotlivých barev používají algoritmy redukce v prostoru a redukce ve spektru. Posledním možným případem vztahu zdrojového a cílového ga-

mutu je situace, kdy cílový gamut je větší než zdrojový, a zde se používají algoritmy expanze jednotlivých barev. [3]

Morovič ve své knize [3] uvádí stavební bloky, které jsou v algoritmech přepočtů gamutů používány. Nejjednoduším z nich je *minimální* ΔE, což je v podstatě oříznutí gamutu – zachování původních barev uvnitř a mapování barev zvenku na nejbližší barvy na obálce podle jejich barevné odchylky ΔE. Používají se však i jiné způsoby určení nejbližších barev, včetně manuálního výběru některých nejbližších barev pozorovatelem a následné interpolace barev zbývajících.

Dalším velmi používaným stavebním blokem je *mapování podél úsečky*, která spojuje bod barvy ve zdrojovém větším gamutu (bod, který chceme mapovat do menšího cílového gamutu) a některý význačný bod v barevném prostoru, např. střed osy světlosti, nebo bod na ose světlosti vypočtený poměrem velikostí obou gamutů, úsečka může vést i rovnoběžně s osou světlosti či rovnoběžně s osou chromatičnosti. Způsobů, jak vést úsečku, je celá řada. Na tuto úsečku je také možno mapovat barevné body různými způsoby. Barevné body ležící mimo cílový gamut mohou být oříznuty, nebo všechny barevné body zdrojového gamutu mohou být mapovány na úsečku v cílovém gamutu a to buď lineárně, nebo různými nelineárními způsoby.



Obr. 1.9: Příklad mapování podél úsečky (konkrétně se jedná o dvě ze tří komponent algoritmu CARISMA). Převzato a upraveno z [3].

Algoritmy přepočtů gamutů mohou také obsahovat různé *interpolační* nebo *morfující techniky*. V praxi se často přepočty provádí tak, že se provedou pouze pro několik barevných bodů, které stačí k tomu, aby zbývající cílové barevné body byly pomocí přesně spočtených bodů interpolovány.

Některé algoritmy také berou v potaz okolní pixely obrazu, nad kterým provádějí přepočet gamutů. Jsou zde využity *prostorové operace*, kdy je část obrazu rozdělena na více frekvenčních pásem, např. pomocí fourierovy transformace, následně je v každém tomto frekvenčním pásmu proveden přepočet do cílového gamutu a obraz je opět zkompletován. Obecně může být taková dekompozice provedena ve frekvenční i v prostorové oblasti a rozdílný přepočet jenotlivých frekvenčních pásem může přinést lepší výsledky u detailů v obraze.

V ideálním případě přepočet barev pro cílové zařízení probíhá se znalostí zdrojového i cílového gamutu, tedy obou zařízení. Přepočet však může proběhnou pouze se znalostí cílového gamutu zařízení, bez znalosti zdrojového gamutu. Potom jsou barvy mimo cílový gamut ořezány. Další možností je přepočet pouze pro konkrétní obraz, kdy se bere v potaz zdrojový gamut pouze daného obrazu (ne všechny barvy zdrojového zařízení, ale všechny barvy, které jsou v obraze přítomny při zobrazení na zdrojovém zařízení) a gamut cílového zařízení. V tomto případě mnohdy dochází k menší kompresi zdrojového gamutu a k lepším výsledkům, nevýhodou je však znovu probíhající přepočet pro každý další obraz a také možné nežádoucí barevné změny stejného objektu, který se na těchto dalších obrazech může vyskytovat.

Výše zmíněné stavební bloky jsou využívány v algoritmech přepočtů gamutů. Tyto algoritmy obsahují mnohdy více těchto stavebních bloků, a tak bývají někdy komplexní a složité. V případě redukce jednotlivých barev se používají např. algoritmy, které jednotlivé barvy v barevném prostoru přepočítávají postupně dimenzi po dimenzi, nebo se používají algoritmy, které zafixují určité jádro barevného prostoru, kde se barvy nepřepočítávají a zbytek zdrojového gamutu se pomocí komprese přepočítá do okrajové části cílového gamutu. Další algoritmy dělí barevný prostor na různé regiony (prostorově) a používají pro každý region jiný přepočet (mapování podél jiné úsečky). Jiné algoritmy využívají mapování podél zaoblených křivek. Některé algoritmy vycházejí také více z nových poznatků ve výzkumu lidského vidění. Klasickým případem přepočtu gamutů je přepočet gamutu monitoru na gamut tiskárny, což může být redukce gamutu, nebo adaptace (gamuty se částečně překrývají). Čím dál více se však v praxi objevuje také expanze gamutů, např. když je pomocí HDTV přehráván starý film apod.

V systému správy barev ICC pak může být použit v podstatě jakýkoli algoritmus, který splňuje daný způsob vykreslení dle předchozí kapitoly. Jak již bylo zmíněno, přepočet nemusí probíhat v barevném prostoru PCS (CIE XYZ, CIE LAB), ale výsledek přepočtu musí být převeden do barevného prostoru, který PCS používá. V případě relativního a absolutního kolorimetrického vykreslení může být použit kterýkoli algoritmus redukce jednotlivých barev (oříznutí zdrojového gamutu), Morovič doporučuje algoritmus *váhované* ΔE v barevném prostoru CIE CAM02 [3]. U perceptuálního vykreslení se naopak používají algoritmy komprese gamutů, protože zde jde o zachování poměru barev a ne o přesné zachování barvy samotné.

Co se týče porovnávání algoritmů přepočtů gamutů, situace je složitá, protože v podstatě neexistuje nějaká všeobecně uznávaná referenční sada dat, na základě které by se daly algoritmy jednoznačně porovnat. Existují výzkumy, které vždy několik algoritmů porovnávají, ale univerzální porovnání neexistuje, záměry a účely jednotlivých algoritmů se navíc mohou lišit. Existují však vodítka, instrukce od sdružení CIE, na jejichž základě se dá hodnocení algoritmů provádět. [3]

2 REALIZOVANÉ ALGORITMY PŘEPOČTŮ GAMUTŮ

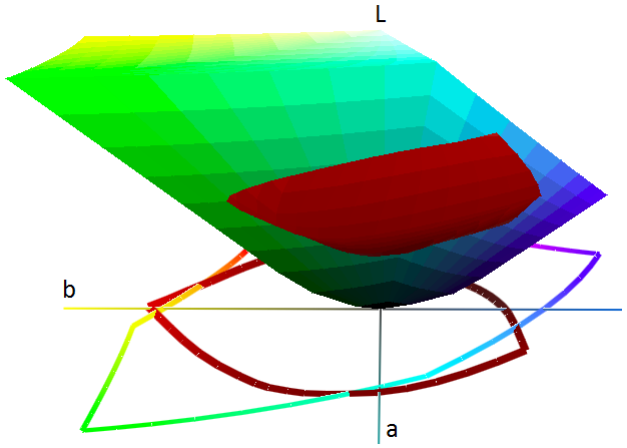
Druhá část práce popisuje dva algoritmy přepočtů gamutů, které byly implementovány v prostředí MATLAB. První z nich je HPMINDE (Hue Planes Minimum Delta E), což by se dalo přeložit jako minimální ΔE v odstínových rovinách. Znamená to, že jsou nejprve provedeny řezy po odstínech v barevném prostoru CIE LAB cílového gamutu a v těchto řezech probíhá mapování (přepočet) podle minimální ΔE. Druhý implementovaný algoritmus je v literatuře nazýván SCLIP a jedná se o ořez podle úsečky. Probíhá opět v řezech odstínů cílového gamutu, kde úsečka, podle které je prováděn ořez barvy, spojuje střed barevného prostoru a přepočítávaný barevný bod. [3]

Tyto algoritmy budou detailně popsány, včetně jejich implementace, v následujících kapitolách této části. Nejprve budou ovšem uvedeny nutné postupy přípravy dat pro přepočty gamutů v prostředí MATLAB, následně bude popsán algoritmus SCLIP, protože je jednodušší na implementaci a poté algoritmus HPMINDE. V závěru bude ještě popsána kompenzace černého bodu, jež je v součinnosti s algoritmy přepočtů gamutů hojně využívána a byla také implementována v prostředí MATLAB.

2.1 Příprava dat pro algoritmy přepočtů gamutů

Tato kapitola popisuje procedury, které je nutné provést s obrazovými daty předtím, než na ně mohou být aplikovány algoritmy přepočtů gamutů. Popisuje také zpětné procedury, aby bylo možno přepočtená data zobrazit opět jako obraz a porovnat jej s původním obrazem.

V prostředí MATLAB byl napsán skript `main.m`, který provádí přípravu dat i samotný přepočet – vykonává s daty veškeré procedury a volá další funkce, které byly napsány, nebo jsou již k dispozici v prostředí MATLAB. V textu budou popsány jen důležité části kódu tohoto skriptu a dalších funkcí. Kompletní komentovaný zdrojový kód, jehož obsahem je právě tento hlavní skript a další níže popsané funkce, je součástí elektronické přílohy na přiloženém CD (celý obsah tohoto média viz příloha). Při psaní kódu v MATLABu bylo čerpáno hlavně z dokumentace programu [25] a z dalších níže uvedených zdrojů. Použitá verze MATLABu byla 7.10.0 (R2010a).



Obr. 2.1: Gamut prostoru sRGB (barevný) a tiskárny (červený). Zobrazeno v [13].

2.1.1 Vstupní data

Vstupními daty přepočtu gamutů jsou dva gamuty, mezi kterými je přepočet prováděn. Jedním z gamutů je zdrojový gamut, který popisuje obrázek nebo zařízení, ze kterého je přepočet prováděn, druhým je cílový gamut, který popisuje zařízení, pro něž je přepočet prováděn. Konkrétně pro tuto práci byl zvolen jako zdrojový gamut samotný gamut obrázku v barevném prostoru sRGB, což je všeobecně uznávaný standardní barevný prostor, vytvořený původně ve spolupráci firem HP a Microsoft pro použití na monitoru, tiskárnách a internetu [26]. Cílový gamut, pro který bude přepočet probíhat, je možno zvolit v kódu na začátku skriptu `main.m` načtením jiného souboru s naměřenými daty, jež cílový gamut definují. Pro ilustraci výsledků této práce byl byl zvolen gamut tiskárny Epson Stylus DX5050, tisk inkoustem ABEL na papír Epson Photo Quality S041061. Porovnání zdrojového gamutu sRGB a cílového gamutu tiskárny je na obr. 2.1. Je vidět, že gamut prostoru sRGB je daleko větší než gamut tiskárny, existují ovšem i barvy, které umí vytisknout jenom tiskárna – ty zůstanou nevyužity, jelikož ve zdrojovém gamutu ani nemohou být obsaženy (algoritmy přepočtů typicky posuzují, zda barva v prostoru sRGB leží mimo cílový gamut tiskárny a pouze v tom případě barvu přepočítávají). Nutno však poznamenat, že samotný přepočet neprobíhá mezi těmito gamuty, ale mezi konkrétními barvami obrázku, jehož barvy jsou dány či omezeny gamutem sRGB a gamutem tiskárny. Načtení barev obrázku a gamutu tiskárny je popsáno hned v další podkapitole.

2.1.2 Převod vstupních dat do prostoru CIE LAB

Na začátku skriptu `main.m`, v sekci `vstupní data`, jsou do proměnných `obrazek` a `DataProfilu` uloženy řetězce názvů souborů, se kterými pracují níže popsané

funkce. Dále je ve skriptu sekce `výpočet barev v lab`, v níž jsou volány právě funkce, které data převádějí do prostoru CIE LAB. Jedná se o funkce `labObrazku` a `labZarizeni`.

Získání barev obrázku

Funkce `labObrazku` z načteného obrázku získá hodnoty všech barev v prostoru CIE LAB. Pracuje tak, že po načtení barev v RGB zařadí barvy pod sebe (do matice, kde jednomu řádku odpovídá jedna barva). Následně vytřídí opakující se barvy, přičemž zachová jejich původní indexy (jako ukazatele do tohoto vytříděného pole pro opětovné seskládání obrázku po přepočtu barev) a převede vytříděné barvy z RGB do CIE LAB prostoru.

Vstupním argumentem funkce je již načtený název souboru v proměnné `obrazek`. Nejdůležitějším výstupním argumentem je matice `lab0`, což jsou právě setříděné barvy v prostoru CIE LAB. První tři sloupce matice tvoří souřadnice L^*, a^*, b^*, poslední sloupec je pořadové číslo barvy v této setříděné matici (slouží k opětovnému seřazení po přepočtu k poskládání barev zpět do obrázku). Další výstupy funkce budou zmíněny dále při popisu funkce.

Funkce na začátku načítá podle názvu souboru obrázku jeho RGB hodnoty: `rgb0 = imread(obrazek)`, kde `imread` je funkce MATLAB Image Tools (formát souboru obrázku závisí tedy na možnostech této funkce, všechny běžné formáty jsou podporovány). Následně jsou získány rozměry obrázku pomocí MATLABovské funkce pro zjištění velikosti proměnné: `[sloupcu radku ~] = size(rgb0)`. Obrázek v hodnotách RGB a rozměry obrázku jsou pro další použití též výstupními argumenty funkce. RGB hodnoty jsou načteny jako trojrozměrná matice, kde počet sloupců a řádků odpovídá rozměrům obrázku a třetí rozměr je tři a odpovídá hodnotám R, G, B pro každý pixel obrázku. Funkce následně prochází obrázek pixel po pixelu a ukládá barvy do nové matice `data`, která má tři sloupce a na každém řádku je jedna barva. Hodonoty všech pixelů jsou tak seřazeny pod sebe:

```
data = uint8(zeros(sloupcu*radku,3)); %alokace paměti pro vyšší rychlost
q = 1; %řádek nové matice dat, v cyklu je inkrementován
for i=1:sloupcu
    for j=1:radku
        data(q,:) = [rgb0(i,j,1),rgb0(i,j,2),rgb0(i,j,3)];
        q = q+1;
    end
end
```

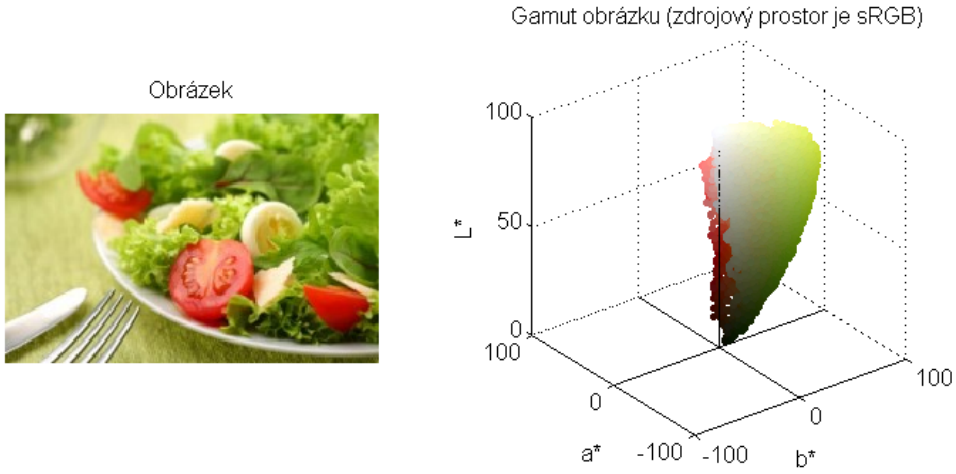
Dále je zavolána funkce MATLABu, která po řádcích vytřídí z matice data barvy, které se opakují a umí zároveň vrátit index do původní nevytříděné matice:

`[tridenaData, ~, index] = unique(data,'rows')`. Proměnná `index` je dalším výstupním argumentem funkce `labObrazku`, slouží pro opětovné sesazení obrázku po přepočtu. Poté jsou barvy již převedeny z hodnot RGB do hodnot CIE LAB a to s pomocí MATLABovských funkcí:

```
TrSrgb2Lab = makecform('srgb2lab'); %definice transformační struktury
labO = applycform(tridenaData,TrSrgb2Lab); %aplikace této struktury
labO = lab2double(labO); %převod z rozsahu 0–255 do rozsahu 0–100
```

Zde funkce `makecform` vytvoří strukturu barevné transformace a pomocí funkce `applycform` je možné ji na data aplikovat. Tyto funkce umožňují mnoho různých převodů nejen mezi typem barevného zápisu, ale i např. mezi ICC profily. Seznam dostupných převodů lze dohledat v dokumentaci [25]. V tomto okamžiku vlastně dochází k převodu z prostoru sRGB do CIE LAB, respektive na základě zvoleného předpokladu, že barvy obrázku jsou v prostoru sRGB, je aplikována barevná transformace, která převádí barvy z prostoru sRGB do prostoru CIE LAB.

Funkce `labObrazku` nakonec ještě zjišťuje délku vytříděné matice barev `labO` pomocí funkce `size` a tento počet barev vrací jako další výstupní argument. Na základě počtu barev je pak vygenerován sloupcový vektor (1 až počet barev), který je přiřazen jako čtvrtý sloupec k matici barev `labO`). To je kvůli možnosti seřazení po přepočtu tak, aby indexy (ukazatele do setříděných barev) souhlasily a bylo možno obrázek zpětně poskládat. Na obrázku 2.2 vpravo je vidět výsledek funkce `labObrazku`, resp. získané barvy v prostoru CIE LAB z obrázku vlevo. K vykreslení barev v prostoru CIE LAB byla použita funkce `plot_Lab` od pánů Eckharda a Dominga [27].



Obr. 2.2: Výsledek funkce `labObrazku` – vlevo obrázek, vpravo jeho barvy v prostoru CIE LAB.

Získání barev zařízení – tiskárny

Obdobně jako byly získány barvy obrázku, funkce `labZarizeni` byla napsána za účelem získání CIE LAB hodnot barev z naměřených kolorimetrických dat pro tiskové zařízení. Zde by mohla vyvstat otázka, proč by nemohl být načten přímo ICC profil dané tiskárny. Tento postup je možný, pro účely této práce byla však zvolena možnost výpočtu dat přímo z naměřených hodnot odrazivosti. Tato možnost je nejen jednodušší, ale také velice dobře demonstruje vznik a výpočet barevných hodnot. Práce s ICC profily je celkem složitá a zřejmě by tak byla nad rámec diplomové práce. ICC profily neobsahují přímo barevná data zařízení, ale pouze informace o převodu mezi barevným prostorem zařízení a barevným prostorem PCS (Profile Connection Space – CIE LAB nebo CIE XYZ), což jsou matice nebo tabulky k přepočtu jednotlivých barev, viz kap. 1.6.

Jediným vstupním argumentem funkce `labZarizeni` je proměnná `DataProfilu`, kde je uložen název souboru s naměřenými daty spekter odrazivosti barev. Výstupy funkce jsou pak proměnné `LabZ`, což jsou hodnoty naměřených barev v prostoru CIE LAB a `xyzZarizeni`, což jsou tytéž barvy v prostoru CIE XYZ.

Soubor s hodnotami spekter odrazivosti je výstupem měření pomocí již zmiňované kolorimetrické sondy i1 Pro od firmy X-Rite a k ní náležícímu měřicímu softwaru [7]. Tento software umožňuje jako výstup měření uložit nejen ICC profil, ale i samotná naměřená data odrazivosti, a to do souboru s příponou .mxf (Material Exchange Format), což je verze xml formátu používaná pro výměnu audiovizuálních dat [28]. Funkce `labZarizeni` tedy na začátku načítá tento .mxf soubor: `Soubor = xml_load(profil)`. Funkce `xml_load` je z XML Toolboxu pro MATLAB od Molinariho [29] a provádí načtení XML souboru do MATLABovské struktury. Následuje volání funkce `cieData`, která nemá žádný vstupní parametr, ale vrací kolorimetrická data: vlnovou délku odpovídající naměřeným hodnotám odrazivosti, násobek hodnot iluminantu D50 a CIE pozorovatele 1931 (2°). Funkce obsahuje pouze definice vektorů s daty, na základě dat CIE [30] a z britské normy ISO 13655:2009 [31].

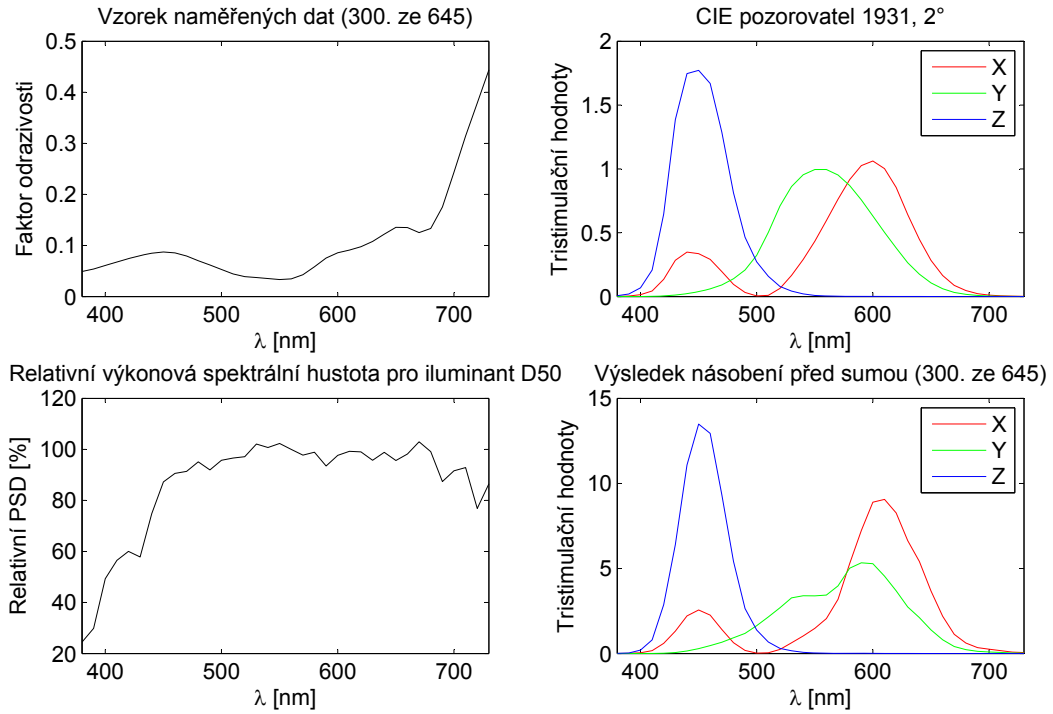
Samotný výpočet barevných hodnot v CIE XYZ potom probíhá následovně:

```
N = sum(Wy); %konstanta pro úpravy XYZ hodnot
Ref = zeros(36,645); %alokace, Ref = odrazivost (Reflectance)
xyzZarizeni = zeros(645,3); %alokace
for i = 1:645 %soubory s naměřenými daty obsahují 645 vzorků
%načtení naměřených hodnot odrazivosti ze struktury Soubor:
     Ref(:,i) = str2num(Soubor.Resources.ObjectCollection(1,i).Object ...
        .ColorValues.ReflectanceSpectrum)';
    X = (1/N) * sum(Wx .* Ref(:,i)); %výpočet hodnot X,Y,Z
```

```
    Y = (1/N) * sum(Wy .* Ref(:,i));
    Z = (1/N) * sum(Wz .* Ref(:,i));
    xyzZarizeni(i,:) = [X, Y, Z]; %řádek matice je barevná souřadnice
end
```

Vektory `Wx`, `Wy` a `Wz` jsou násobky hodnot iluminantu D50 a CIE pozorovatele 1931. Jsou výstupem již zmiňované funkce `cieData` a jedná se o 36 hodnot pro vlnové délky od 380 nm po 730 nm s rozestupem 10 nm. To proto, že každý ze 645 vzorků v souboru s naměřenými daty odrazivosti obsahuje právě 36 hodnot odpovídajících těmto vlnovým délkám. Výpočet hodnot X, Y, Z potom probíhá podle rovnic 1.1 až 1.4 v kapitole 1.5.2.

Na následujícím obrázku je průběh výpočtu graficky znázorněn. Poslední graf vpravo dole je výsledkem násobení hodnot průběhů z předchozích grafů (odrazivosti, CIE pozorovatele 1931 a iluminantu D50). Hodnoty barvy X, Y, Z potom představuje plocha pod křivkami výsledných průběhů (ve výpočtu suma).



Obr. 2.3: Průběhy výpočtu hodnot CIE XYZ z dat naměřené odrazovosti, CIE pozorovatele 1931 a iluminantu D50.

Funkce v závěru vypočtené barevné hodnoty v CIE XYZ převádí opět pomocí funkcí `makecform` a `applycform` do prostoru CIE LAB. Výsledek výpočtu barev je patrný na obrázku 2.5 vlevo dole (Barvy zařízení). Jedná se o oněch 645 naměřených barevných bodů uložených v proměnné `labZ`, které udávají cílový barevný prostor tiskárny.

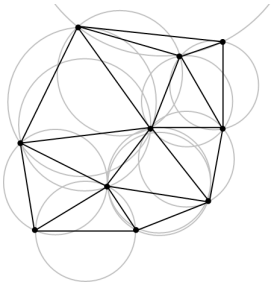
2.1.3 Výpočet konvexní obálky – cílového gamutu

Konvexní obálka množiny bodů je matematické formální vyjádření myšlenky obalení množiny bodů něčím elastickým. Ve dvojrozměrném prostoru by měla tato obálka formu nejmenšího možného mnohoúhelníku, v němž leží všechny body dané množiny. V trojrozměrném prostoru se pak bude jednat o nejmenší možný mnohostěn. [3]

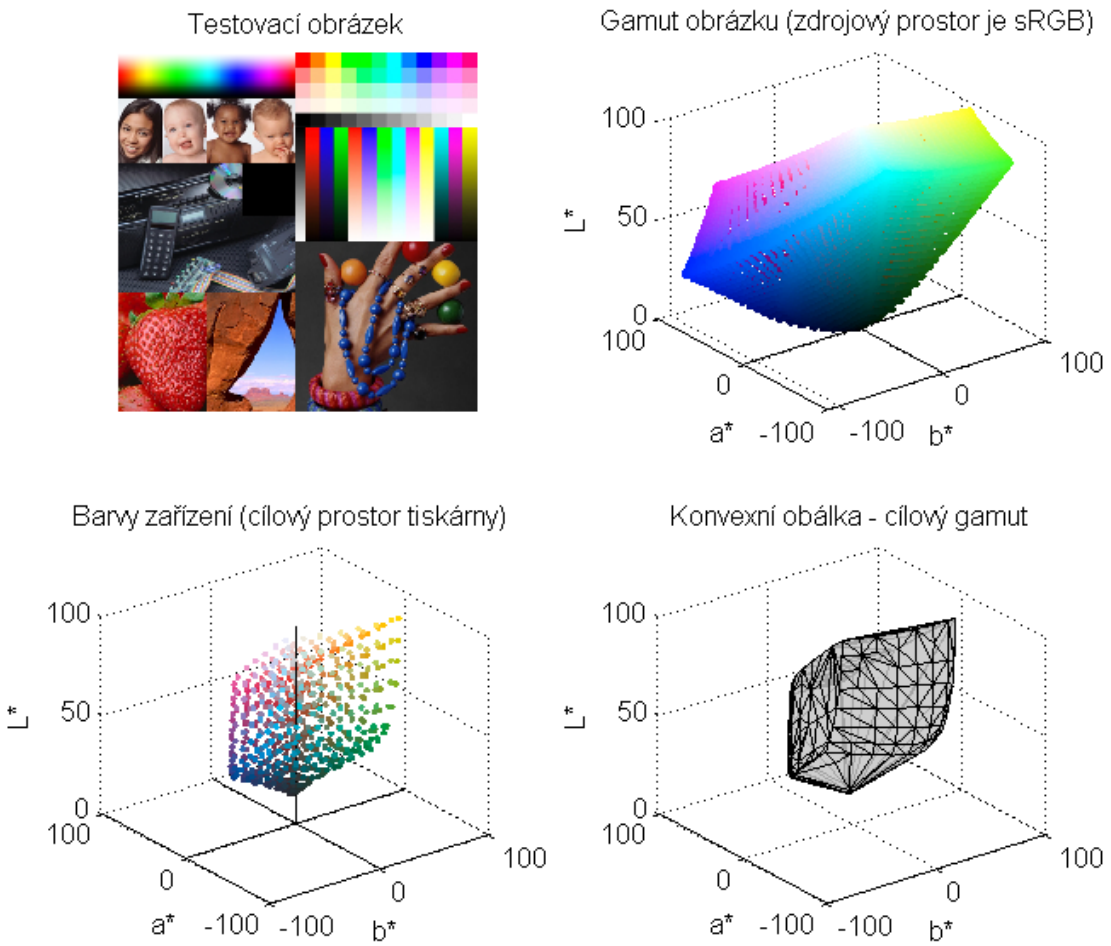
Nutnou a základní součástí algoritmů přepočtů gamutů je vhodný matematický popis hranic gamutu. Výpočet konvexní obálky je jednou z možností popisu hranic barevného prostoru – získání cílového gamutu. Tento způsob popisu u mnohých gamutů plně dostačuje, jsou však případy, kdy je tento popis nepřesný, gamuty některých zařízení by někdy lépe popisovaly hranice s konkávními oblastmi. Proto se v praxi často používá popis pomocí modifikované konvexní obálky, kdy jsou předem zjištěny konkávní oblasti hranic a výsledná konvexní obálka je potom modifikována tak, aby hranice gamutu lépe popisovala. [3]

Jako alternativu k popisu hranic gamutů lze místo konvexní obálky použít například tzv. *alfa útvary*, kdy algoritmus popisu hranic gamutu vychází z toho, že množina bodů je „objížděna“ kruhovým útvarem a to, zda se zde objeví konkávní oblasti, záleží na poloměru oné kružnice a vzdálenosti barevných bodů od sebe navzájem. Morovič také podrobněji rozvádí a doporučuje další alternativu a tou je *maximum segmentu*, kdy dochází k rozdělení barevného prostoru na sférické segmenty podle jednotlivých barev. Hranici gamutu pak popisují maximální, extrémní hodnoty v těchto segmentech (barevné body s největším poloměrem od středu sférických segmentů) a obálka je vytvořena na jejich základě. [3]

Pro tuto práci byla zvolena nejjednodušší metoda – popis cílového gamutu pouze pomocí konvexní obálky. Konvexní obálku určují body, které jsou jejími vrcholy. Máme-li množinu bodů v dvojrozměrném prostoru, nejjednodušší metodou, jak určit vrcholy konvexní obálky, je algoritmus *Jarvis March* (nebo také „balení dárku“) [3]. Ten postupuje tak, že jako výchozí je vzat bod s nejnižší hodnotou souřadnice y a pak je hledán bod s nejmenším úhlem od osy x. Tento bod je dalším vrcholem konvexní obálky a tyto dva body určují další přímku, od které je opět hledán bod, který s ní



Obr. 2.4: Delaunayova triangulace v rovině s opisujícími kružnicemi. Převzato z [22].



Obr. 2.5: Vstupy pro algoritmy přepočtů gamutů – nahoře testovací obrázek a všechny barvy, které obsahuje, dole naměřené barvy tiskárny a konvexní obálka těchto naměřených barev – cílový gamut.

svírá nejmenší úhel. Tak se postupuje, až je mnohoúhelník uzavřen návratem do výchozího bodu.

V praxi a v trojrozměrném prostoru se však k nalezení konvexní obálky používají efektivnější metody a algoritmy, jedním z oblíbených je *Q hull* [20], který využívají některé funkce MATLABu pro výpočet konvexní obálky a je také využíván pro výpočet Delaunayovy triangulace či Voronoiho diagramu. Výpočtu konvexní obálky předchází právě výpočet *Delaunayovy triangulace*. Delaunayova triangulace pro množinu bodů v rovině (na obr. 2.4) je taková triangulace, kdy kruh opsaný každému trojúhelníku neobsahuje uvnitř žádné body z množiny. Snahou je maximalizovat úhly všech trojúhelníků triangulace a dosáhnout toho, aby se tyto blížily rovnostranným trojúhelníkům [21]. Delaunayovu triangulaci velmi dobře ilustruje aplet [23] a výpočet konvexní obálky na jejím základě různými modifikovanými způsoby v trojrozměrném prostoru potom znázorňuje aplet [24].

V prostředí MATLAB [25] existuje několik již připravených funkcí, které fungují na základě výše zmíněných geometrických algoritmů. Výpočet konvexní obálky z barevných bodů cílového gamutu, které jsou uloženy v proměnné `labZ`, probíhá následovně: `konvObalkaTri = convexHull(DelaunayTri(labZ))`.

Jedná se o dvě vnořené funkce, nejprve je nad množinou barevných bodů provedena Delaunayova triangulace funkcí `DelaunayTri` a tato následně vstupuje do funkce `convexHull`, která z Delaunayovy triangulace všech bodů vypočítá konvexní obálku – tedy triangulaci bodů pouze na povrchu. Ve výsledné proměnné `konvObalkaTri` jsou potom uloženy indexy vrcholů mnohostěnu konvexní obálky – slouží jako ukazatele do původního pole barevných bodů `labZ`. Tyto body jsou (jako povrch který tvoří) vykresleny na obr. 2.5 vpravo dole.

2.1.4 Řez barevným prostorem podle odstínu

Protože algoritmy přepočtů probíhají v rovinách odstínu (pro jeho zachování), je nutné provést řez barevným prostorem na základě odstínu. Odstín představuje v prostoru CIE LAB úhel barevného bodu. Tyto řezy tedy není problém provést po převodu na systém souřadnic CIE LCh, kde barva má stále tři souřadnice – souřadnice L^* zůstane nezměněna, souřadnice C představuje chromatičnost (rovnice 1.13) a souřadnice h odstín (rovnice 1.14). V podstatě se jedná pouze o převod na jiný systém souřadnic. Řez rovinou odstínu je nutné provést jak ve zdrojovém, tak v cílovém barevném prostoru. Ve zdrojovém barevném prostoru je to možné udělat pouze vytřízením barev o stejném odstínu, v cílovém barevném prostoru jde o průsečík roviny odstínu a konvexní obálky, což je lomená čára. Nejprve bude vysvětlena právě tato funkce řezu konvexní obálkou a potom vytřídění barevných bodů zdrojového prostoru dle odstínu.

Řez cílovým gamutem dle odstínu

Funkce `rizniHue3` vypočte lomenou čáru průsečíku cílového gamutu a roviny odstínu. Vstupy funkce jsou barevné body tiskárny `labZ`, ukazatele na body, které tvoří konvexní obálku `konvObalkaTri` a odstín `hue`, ve kterém chceme řez provádět. Výstupem je proměnná `bodyRezu`, v níž jsou uloženy dvojrozměrné souřadnice bodů, které tvoří lomenou čáru řezu. Na začátku funkce jsou vygenerovány tři body, které tvoří řeznou rovinu – bod počátku, bod nahoře na ose L^* (rovina řezu je svislá) a bod obsahující informaci o odstínu. Následně je vypočítán normálový vektor roviny, který spolu s dalšími argumenty vstupuje do funkce `slice_iso_data`, jejímž autorem je D'Errico [32]. Tato funkce vrací trojrozměrné souřadnice bodů řezu. Funguje tak, že nejprve zjišťuje, které trojúhelníky triangulace jsou protnuty rovinou řezu (zjištěním, zda vrcholy trojúhelníků leží na té či opačné straně roviny

– protnuty jsou ty, jejichž vrcholy leží na obou stranách roviny). Tyto trojúhelníky pak v cyklu prochází a hledá, které strany těchto trojúhelníků jsou protnuty a kde přesně leží body průsečíků s rovinou.

```
pocatek = [0,0,0]; horniBod = [1,0,0]; %definice bodů řezné roviny
hueBod = [0,1,hue]; %třetí bod s odstínem dle vstupu
tr2lab = makecform('lch2lab'); %převod hueBodu na CIE LAB souřadnice
hueBod = applycform(hueBod,tr2lab);
normal = cross(pocatek-horniBod, pocatek-hueBod); %normálový vektor roviny
%řez pomocí funkce slice_iso_data
[hranicniBody,~]=slice_iso_data(pocatek,normal,labZ,konvObalkaTri);
```

Trojrozměrné souřadnice bodů řezu jsou teď uloženy v proměnné `hranicniBody` a tvoří polygon, protože řez proběhl ve svislé rovině počátkem i v úhlech posunutých o 180°. Funkce `rizniHue3` pak dále tyto body převádí opět pomocí dvojice `makecform` a `applycform` na souřadnice CIE LCh. Potom ještě tyto body prochází cyklem a zahazuje body, které jsou o 180° posunuty – ty nemají být ve výsledné lomené čáře průsečíku. Tak je vytvořena výstupní matice dvojrozměrných bodů řezu `bodyRezu` se souřadnicemi L^* a C.

Jelikož však není pravděpodobné, že by hrana některého trojúhelníku přesně protínala osu L^*, je nutné ještě dopočítat průsečíky konvexní obálky s osou L^* a k bodům je přidat. Jelikož tyto průsečíky jsou stejné pro řez jakýmkoli úhlem, není tento výpočet zahrnut ve funkci `rizniHue3`, ale počítá je předem a pouze jednou funkce `ziskejPrusecikyL`. Ta stejným způsobem jako výše vysvětlená `rizniHue3` provede řez konvexní obálkou – zvolen byl řez v rovině osy a^*. Potom body polygonu, který řezem vznikne, rozdělí na ty, které leží nalevo a napravo od osy L^*. V každé z těchto skupin nadále hledá dva body nejbližší ose L^* – jeden výše než bod `[50, 0, 0]`(polovina osy L^*, 50% šedá) a druhý níže. Tak jsou nalezeny čtyři body – dva spodní jsou spojeny úsečkou a dva horní taktéž. Průsečíky konvexní obálky s osou L^* jsou potom dva body, které jsou vypočteny jako průsečíky osy L^* a obou nalezených úseček. Průsečík osy L^* a úsečky počítá upravená funkce `findintersection` od Vhenganiho [33], a k nalezení průsečíku používá metodu determinantů [34]. Dva body průsečíků jseou tedy výstupy funkce `ziskejPrusecikyL`.

Výstupy funkce `rizniHue3` (body lomené čáry řezu) a funkce `ziskejPrusecikyL` pak nakonec vstupují do funkce `seradBody`, která k bodům lomené čáry přidá body průsečíků a zároveň body seřadí do matice tak, aby prvním bodem byl horní průsečík s osou L^* a dále aby body navazovaly na sebe a posledním byl spodní průsečík s osou L^*. To je provedeno tak, že je nalezen bod s nejvyšší chromatičností (nejvyšší souřadnicí C). Body vyšší (dle souřadnice L^*) jsou pak seřazeny vzestupně podle

chromatičnosti a body nižší jsou seřazeny sestupně podle chromatičnosti. Na začátek a na konec matice bodů jsou přidány průsečíky s osou L^*). Toto řazení probíhá kvůli tomu, aby algoritmy přepočtů gamutů v každém řezu odstínu mohly brát body jeden po druhém, jako na sebe navazující segmenty.

Výslednou lomenou čáru – hranici cílového gamutu v rovině určitého odstínu, zobrazuje obrázek 2.6. Řez cílovým gamutem je prováděn vždy pro určitou skupinu (kategorii) bodů se stejným odstínem. To probíhá v cyklu postupně pro všechny kategorie (odstíny) bodů a následně je volána funkce přepočtu pro body ležící mimo cílový gamut – viz další kapitola.

Řazení barevných bodů dle odstínu

Výše zmíněné funkce umožňují řez cílovým gamutem. Nyní bude vysvětlena funkce `rozdelHueKat`, která dosahuje téhož pro zdrojové barvy obrázku. Jelikož se zde nepracuje s geometrickým útvarem konvexní obálky, ale s konkrétními barevnými body obrázku, postačí tyto body roztřídit do kategorií podle odstínu barvy.

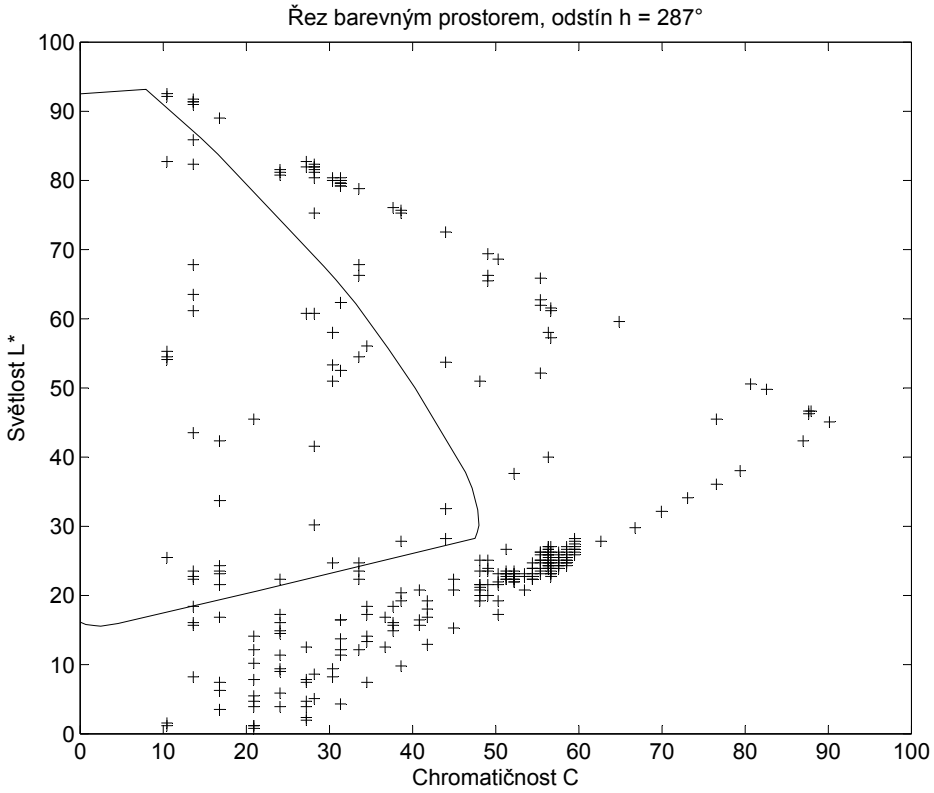
Nejprve jsou barevné body obrázku převedeny do souřadnic CIE LCh (`makecform`, `applycform`), kde třetí souřadnice mají přímo význam odstínu. Následně jsou barevné body pomocí MATLABovské funkce `sortrows` seřazeny právě podle třetího sloupce, aby změna odstínu byla při procházení barev cyklem dobře detekovatelná.

Vstupy funkce `rozdelHueKat` je matice seřazených barevných bodů `lch` ve formátu CIE LCh a počet barev v proměnné `barev`. Výstupem je struktura `k`, která obsahuje maximálně 360 matic s barevnými body vždy jednoho odstínu, vektor `hue` s odstíny těchto kategorií a počet kategorií. Nejdůležitější částí funkce je cyklus, který barevné body prochází, detekuje změnu odstínu a ukládá je do výsledné struktury.

```
for i = 2:barev %od 2. bodu – porovnáváme současný bod s předchozím
    lchKat(i,:) = [lch(i,1:2), round(lch(i,3)), lch(i,4) ]; %zaokrouhlení
    if(lchKat(i,3) > lchKat(i-1,3)) %pokud je odstín vyšší
        k.(sprintf('k%d', kat)) = temp; %uložíme barvy předchozí kat.
        hue(kat,1) = lchKat(i-1,3); % uložím odstín kategorie
        kat = kat + 1;  %posun na další kategorii
        temp = []; %vymažu pomocnou proměnnou
    end
    if(lchKat(i,2) == 0) %pokud bod leží na ose L, je v kat. 0
        k.k0 = [k.k0; lch(i,:)];
    else %ukládám body do pomocné proměnné
        temp = [ temp; lch(i,:)];
    end
end
```

K omezení počtu kategorií na 360 bylo přikročeno z toho důvodu, že barevné body obrázků obsahují obrovské množství odstínů a kategorií by tak bylo obrovské množství. Pro každou kategorii je nutné provádět výše popsaný řez cílovým gamutem a to je časově náročné na výpočet, navíc v případech, kdy se odstín liší o desetinná místa, je výsledný řez cílovým gamutem velmi podobný, téměř totožný. Byla porovnána přesná verze (reagující na každou změnu v odstínu barvy) s touto verzí a bylo shledáno, že výsledek je téměř totožný, rozdíl je lidským okem nerozeznatelný. V cyklu dochází k tomu, že detekce vyššího odstínu probíhá na základě porovnávání zaokrouhlených hodnot odstínů. Barevný prostor je tak rozdělen na kategorie s výsečí 1°. Původní hodnoty odstínů jsou však zachovány a uloženy do výsledné struktury, což je vidět z uvedeného kódu.

Řez barevným prostorem, který zobrazuje jednu kategorii bodů stejného odstínu, je vidět na následujícím obrázku. Z obrázku je patrné, že některé body kategorie leží uvnitř cílového gamutu a některé vně. Právě body ležící vně cílového gamutu je nutné přepočítat tak, aby ležely uvnitř.



Obr. 2.6: Řez barevným prostorem cílového gamutu (plná čára) a body kategorie příslušného odstínu zdrojového obrázku.

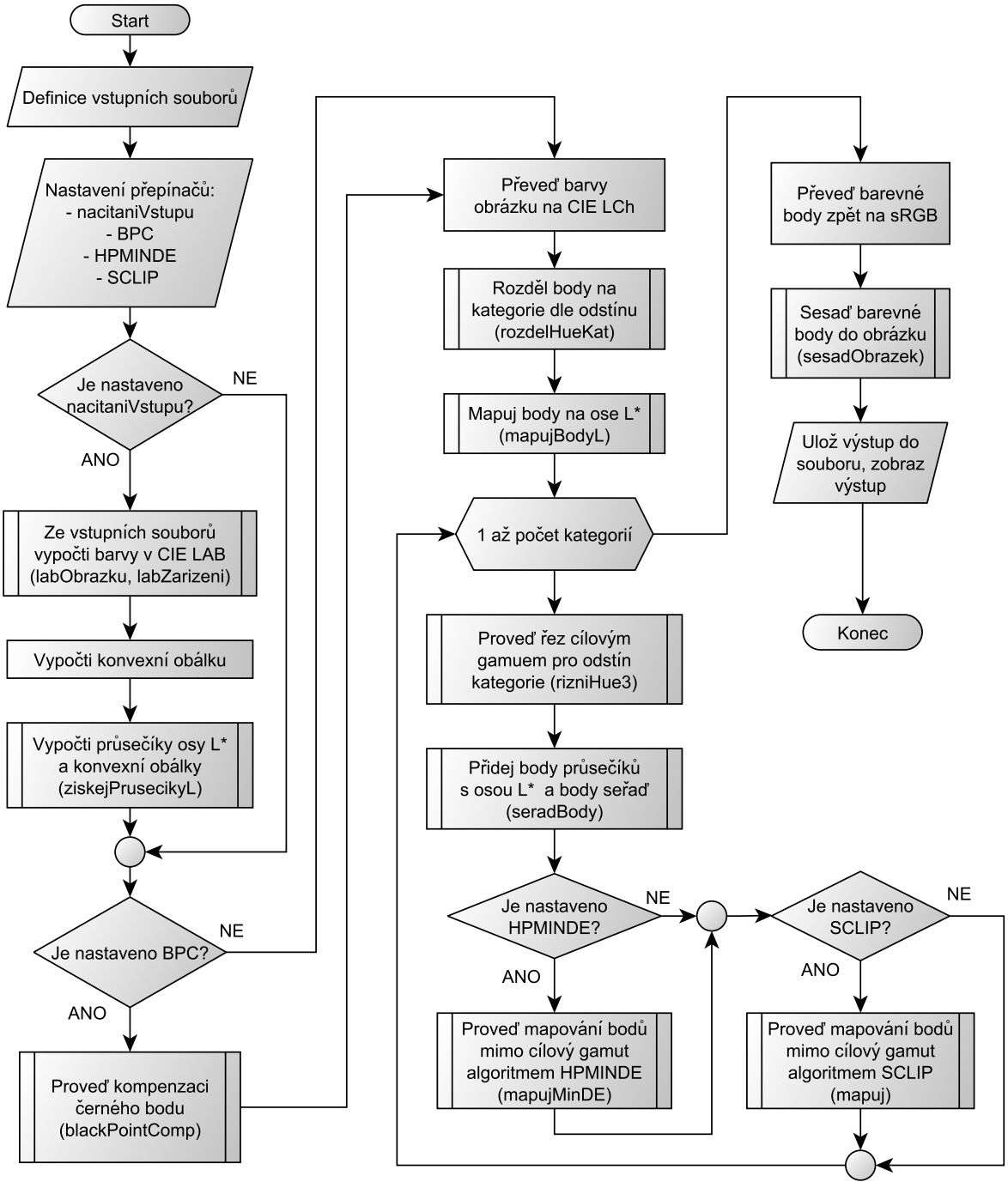
2.1.5 Mapování bodů mimo gamut – samotný přepočet

Nyní jsou všechna vstupní data pro algoritmy přepočtů gamutů připravena a může proběhnout samotný přepočet – mapování bodů mimo gamut na konvexní obálku. Přepočet tedy nyní probíhá v rovinách dle odstínu, pro každou kategorii bodů se stejným odstínem (resp. ve výseči odstínu 1°). Zde bude uveden pro přehlednost zjednodušený vývojový diagram skriptu `main.m` a bude vysvětlen cyklus, který postupně pro každou kategorii volá funkce přepočtu. Tyto funkce – algoritmy SCLIP a HPMINDE – budou vysvětleny v následujících samostatných kapitolách.

Na obrázku 2.7 je uveden pro přehlednost a lepší pochopení vývojový diagram skriptu `main.m`. Na začátku dochází k výše popsané definici vstupních souborů a k nastavení přepínačů (určují které části kódu se mají provádět a které přeskočit). Dále jsou vykonány výše popsané procedury. Samotné mapování zabírá skoro polovinu diagramu a jedná se hlavně o cyklus, který provádí mapování bodů ležících mimo cílový gamut pro každou kategorii. Po proběhnutí cyklu jsou barevné body sesazeny do výsledného obrazu, což podrobněji popisuje následující podkapitola. Mírně zjednodušený zdrojový kód mapování vypadá následovně:

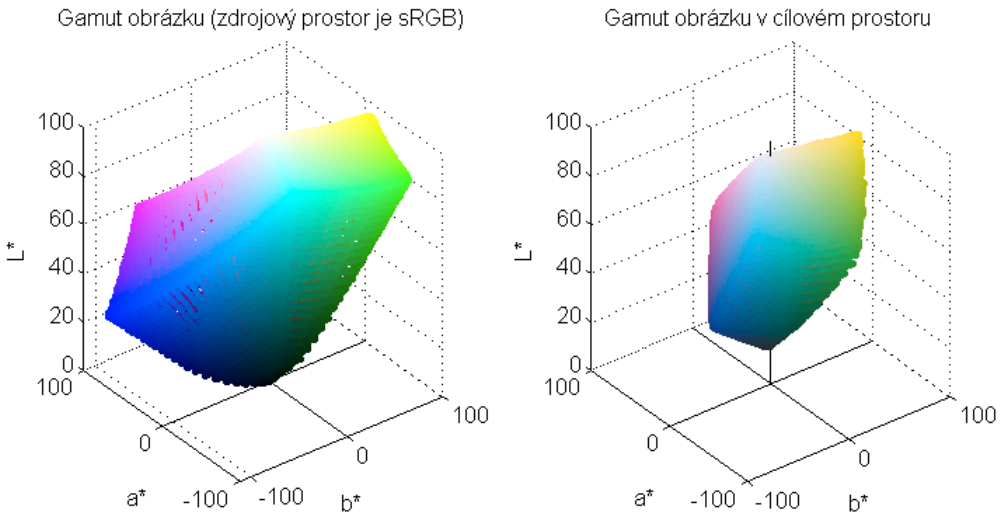
```
n.k0 = mapujBodyL( k, prusecikDole, prusecikHore ); %body na ose L*

for kat=1:kategorii %pro každou kategorii
    bodyRezu = rizniHue3(labZ,konvObalkaTri,hue(kat)); %řez cíl. gamutem
    bodyRezu = seradBody(bodyRezu, prusecikHore, prusecikDole);
    if(HPMINDE)
        n.(sprintf('k%d', kat)) = mapujMinDE(k.(sprintf('k%d', kat)), ...
          bodyRezu);
    end
    if(SCLIP)
        n.(sprintf('k%d', kat)) = mapuj(k.(sprintf('k%d', kat)), ...
          bodyRezu);
    end
end
```

Zde nejprve probíhá mapování bodů, které leží na ose L^*, pomocí funkce `mapujBodyL`. Ty jsou uloženy v proměnné `k0` struktury `k`. Funkce jednoduše cyklem prochází body této speciální kategorie a porovnává, zda leží níže nebo výše, než průsečíky osy L^* s cílovým gamutem. Pokud bod leží níže než spodní průsečík, je jím nahrazen, stejně pokud bod leží výše než horní průsečík, je mapován na tento průsečík. Body dalších kategorií, které nemají nulovou chromatičnost (tj. neleží na ose L^*) jsou uloženy v proměnných struktury `k`: `k1`, `k2`, `k3` ... až max. `k360`). Ty jsou mapovány v cyklu, který prochází všechny kategorie. Pro každou kategorii (odstín) je proveden řez cí-



Obr. 2.7: Vývojový diagram skriptu `main.m`

lovým gamutem a body řezu jsou seřazeny pro algoritmy mapování. Pak záleží na přepínačích zvolených na začátku skriptu, která funkce mapování je zavolána. Funkce `mapujMinDE` provádí přepočet pomocí algoritmu HPMINDE a funkce `mapuj` pomocí algoritmu SCLIP. Tyto funkce jsou popsány dále v kapitolách 2.2 a 2.3. Přepočtené barevné body jsou ukládány do struktury `n`, která tak ve výsledku zrcadlí vstupní strukturu `k`, ovšem všechny barevné body se již nacházejí uvnitř či na hranici cílového gamutu. To je názorně patrné z obrázku 2.8, kde jsou zobrazeny barevné body testovacího obrázku před a po mapování.



Obr. 2.8: Barevné body testovacího obrázku před mapováním (vlevo) a po mapování (vpravo) na cílový gamut.

2.1.6 Převod barevných bodů zpět do obrazu

Nyní je nutno z přepočtených barevných bodů obrázku vytvořit opět matici obrazu, ve které se nacházely předtím ve vstupním souboru. Body nacházející se ve struktuře `n`, jsou nyní uloženy do jediné proměnné opět ve formě 4 sloupců (3 sloupce barevných souřadnic, 1 sloupec pořadí). To je provedeno s využitím funkce `eval`, která vykoná vstupní řetězec jako příkaz. Vstupním řetězcem je zde seznam všech kategorií v nové struktuře obsažených. Následně jsou provedeny nutné zpětné převody barevných bodů. Barvy jsou pomocí `makecform` a `applycform` převedeny z CIE LCh zpět do prostoru CIE LAB. Následně jsou barvy seřazeny seřazeny podle čtvrtého sloupce s využitím funkce `sortrows` (všechny předchozí funkce nakládaly s body tak, že zachovávaly jejich čtvrtou souřadnici, i když byly první tři souřadnice přepočteny). Seřazení zpět umožňuje následovně využít proměnnou `index`, kde

jsou uloženy ukazatele do setříděné matice (lze si je představit jako pixely výsledného obrázku s odkazy na pozice barev, které mají na daném místě být). Nakonec jsou opět seřazené barevné body převedeny zpět do osmibitového rozsahu 0 až 255 pomocí funkce `lab2uint8` a opět pomocí `makecform`, `applycform` na sRGB hodnoty. Tyto body již spolu s proměnnou `index` a rozměry obrázku vstupují do funkce `sesadObrazek`. Ta provádí následovně sesazení obrázku do matice tak, aby jej bylo možno zobrazit:

```
rgbOUTmatice = uint8(zeros(sloupcu,radku,3)); %alokace
q = 1;
for i=1:sloupcu
    for j=1:radku
        rgbOUTmatice(i,j,1) = rgbOUT(index(q),1);
        rgbOUTmatice(i,j,2) = rgbOUT(index(q),2);
        rgbOUTmatice(i,j,3) = rgbOUT(index(q),3);
        q = q+1;
    end
end
```

Z kódu je patrné, že jde o vytváření trojrozměrné matice `rgbOUTmatice`, kdy cyklus, který obrázek sestavuje, je stejný jak ten, který jej rozebíral a hodnotám R, G, B každého pixelu jsou z vytřízěné matice barev (`rgbOUT`) přiřazovány hodnoty na základě již popisované proměnné `index`.

Tato výsledná matice je následně jako výsledek zobrazena s využitím funkce `imshow` a také uložena do souboru ve formátu .tif pomocí funkce `imwrite`.

2.1.7 Pomocné funkce

Tato krátká podkapitola ještě osvětluje některé další funkce, které byly vytvořeny a také obsahuje základní instrukce ke spuštění skriptu `main.m`.

Všechny soubory zdrojového kódu se nacházejí na přiloženém CD ve složce `MATLAB_zdrojovy_kod`. Pro spuštění v MATLABu stačí složku zkopírovat do pracovního adresáře a zajistit, aby byla složka včetně jejích podsložek přidána do pracovní cesty MATLABu. Pak je již možno přímo spustit skript `main`, nebo tento skript otevřít, upravit nastavení vstupních souborů a přepínačů funkcí a poté spustit. Po otevření skriptu k editaci lze na začátku nastavit vstupní soubory (obrázek a naměřená data tiskárny). Ve složce `zdrojove_obrazky` je na výběr několik obrázků a ve složce `namerena_data_tiskarny` je na výběr několik souborů `.mxf` s naměřenými daty odrazivosti. Také je zde možno nastavit základ názvu výstupního souboru, který se však celý tvoří podle nastavených přepínačů (není tak nutné

pro jeden obrázek a více možností přepínačů neustále nastavovat nový název výstupního souboru). Výstupní soubor je uložen do stejné složky, v níž se nachází skript `main.m`.

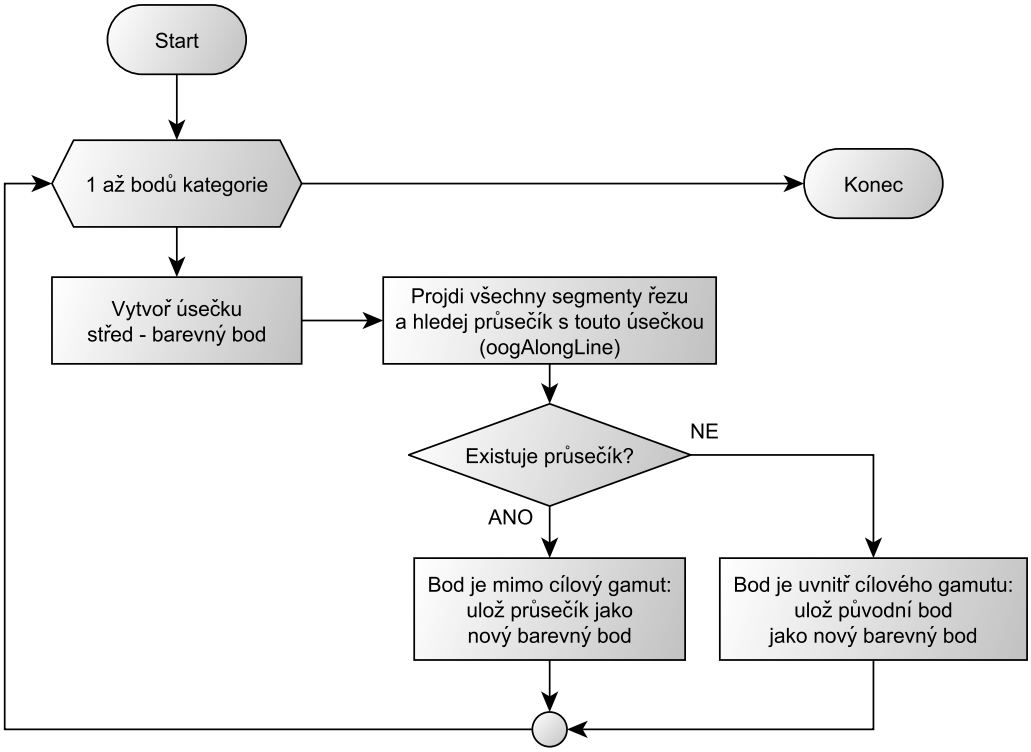
Dále je možno nastavit přepínače, jimiž lze ovlivnit to, zda budou do pracovního prostoru MATLABu načítána data zdrojového obrázku a cílového gamutu (umožňuje kratší dobu výpočtu při opakovaném spouštění), dále zda bude prováděna kompenzace černého bodu (viz dále kap. 2.4). Nakonec se nastavuje, který ze dvou algoritmů (HPMINDE či SCLIP) bude proveden (hodnotu `true` musí mít nastavenu pouze jeden z nich).

Na konci skriptu je v sekci `zobrazení` možnost odkomentovat některé části kódu, které volají funkce k zobrazení vstupních či výstupních dat. Kromě zobrazení výsledku mapování je tak možno zobrazit vstupy (jako na obr. 2.5), dvanáct řezů barevným prostorem před a po mapování (podobně jako na obrázku 2.11) nebo porovnat gamut obrázku před a po mapování jako na obr. 2.8.

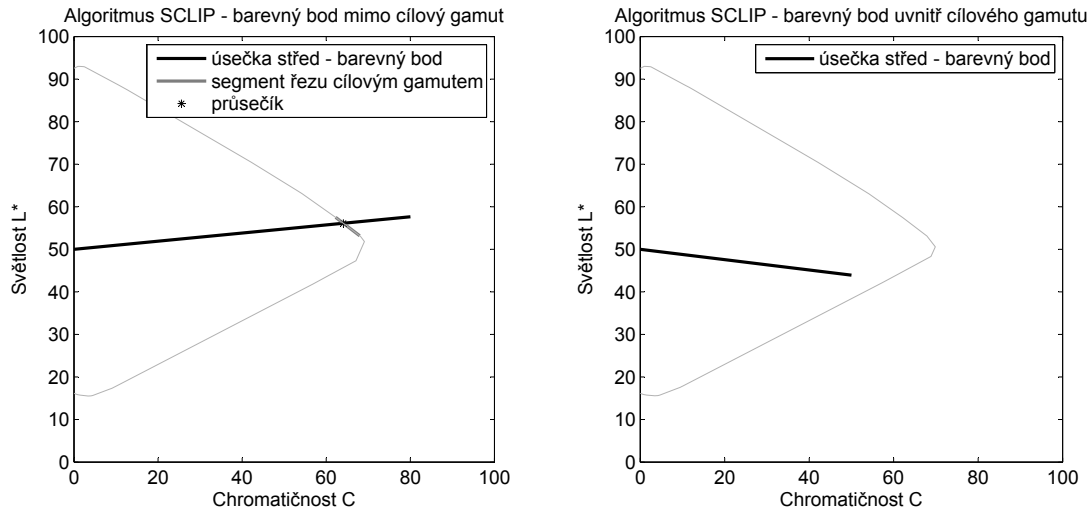
Nakonec je ještě nutno zmínit jeden skript, který byl napsán za účelem porovnání výsledků pomocí metriky ΔE. Ten se jmenuje `porovnejObrazky.m` a vykonává to, že načte dva obrázky (např. originál a výsledek přepočtu pomocí skriptu `main`) a vypočte hodnotu odchylky ΔE mezi všemi pixely obrázků. To provádí funkce `deltaE`, kterou skript volá. Tato funkce převede hodnoty barev obrázků do prostoru CIE LAB a dále obsahuje vnořený cyklus, kdy je na místě každého pixelu obrázku spočtena norma (odchylka ΔE) mezi pixely porovnávaných obrázků. Hodnoty této odchylky jsou ukládány do nové matice a jsou následně zobrazeny jako obrázková mapa, kterou skript také na konci ukládá do souboru .png. Výsledky tohoto skriptu jsou vidět v kapitole 3, kde jsou prezentovány výsledky algoritmů přepočtů.

2.2 Algoritmus SCLIP

Algoritmus SCLIP mapuje barvy, které leží mimo cílový gamut na hranici gamutu, přičemž zachovává odstín barvy a úhel elevace od středu osy světlosti. Střed osy světlosti je bod [50, 0, 0] – 50% šedá. Algoritmus poprvé navrhl Sara v roce 1984 (v kompresní podobě) [3]. Princip tohoto algoritmu naznačuje vývojový diagram na obr. 2.9. Zde je pro každý bod kategorie nejprve vytvořena úsečka, která spojuje střed osy světlosti a barevný bod. Následně jsou procházeny segmenty řezu konvexní obálkou pro tuto kategorii (odstín) a je hledán průsečík s některým z těchto segmentů. Kdyby střed osy světlosti a barevný body definoval přímku, byl by průsečík nalezen vždy. Zde jsou ovšem tyto body brány jako úsečka a prúsečík tak musí ležet uvnitř této úsečky. To umožňuje, aby bylo posouzeno, zda barevný bod leží mimo cílový gamut, nebo je uvnitř. Pokud je průsečík nalezen, potom barevný bod



Obr. 2.9: Vývojový diagram – princip algoritmu SCLIP



Obr. 2.10: Algoritmus SCLIP – barevný bod mimo cílový gamut (vlevo) a uvnitř cílového gamutu (vpravo). Světle šedá lomená čára je hranice cílového gamutu.

leží mimo gamut a musí být mapován. Nalezený průsečík je zároveň novým bodem, který leží na hranici cílového gamutu. V opačném případě není potřeba bod mapovat a jako nový bod je uložen bod původní. Oba případy ilustruje obrázek 2.10.

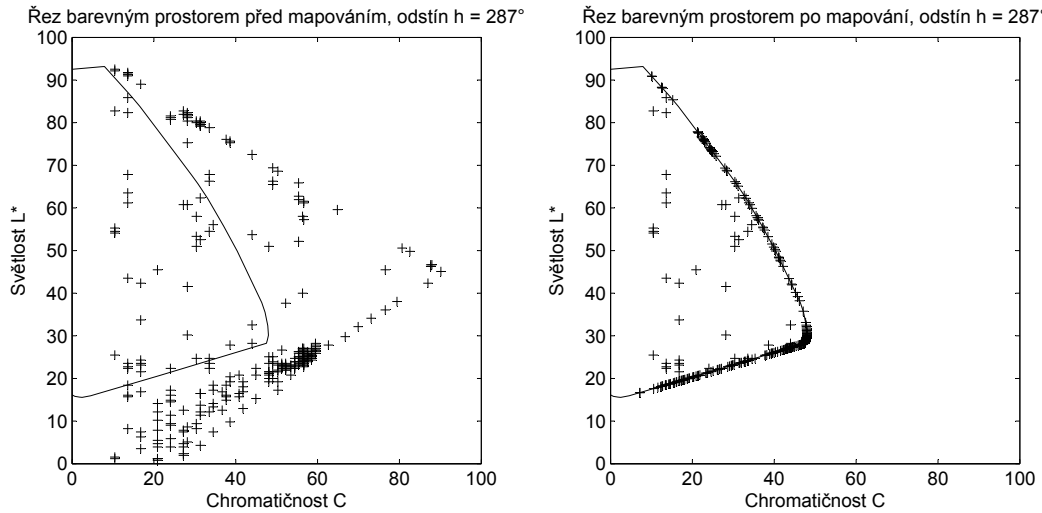
Dalo by se tedy říci, že jádro algoritmu tvoří nalezení průsečíku úsečky střed osy světlosti – barevný bod s řezem cílovým gamutem. Ve skriptu `main.m` vykonává algoritmus SCLIP funkce `mapuj`. Ta obsahuje cyklus, který prochází všechny body dané kategorie (odstínu). Pro každý bod volá funkci `oogAlongLine`, která v případě, že bod je mimo cílový gamut, vrací tuto informaci i nalezený průsečík. Ten je následně zapsán jako nový bod na hranici gamutu. Pokud funkce `oogAlongLine` nevrátí informaci o existujícím průsečíku, je jako nový bod zapsán původní bod (bod leží uvnitř cílového gamutu). Zdrojový kód funkce `oogAlongLine` vypadá následovně:

```
function [ isOOG novyBodAlongLine ] = oogAlongLine( bodyRezu , bod )
isOOG = false;
novyBodAlongLine = [];
stred = [0, 50];
seg = complex([stred(1); bod(1)],[stred(2); bod(2)]);
pline = complex(bodyRezu(:,1),bodyRezu(:,2));
[zi isOOG] = PolylineIntersectSegment(seg,pline);
novyBodAlongLine = [real(zi),imag(zi)];
end
```

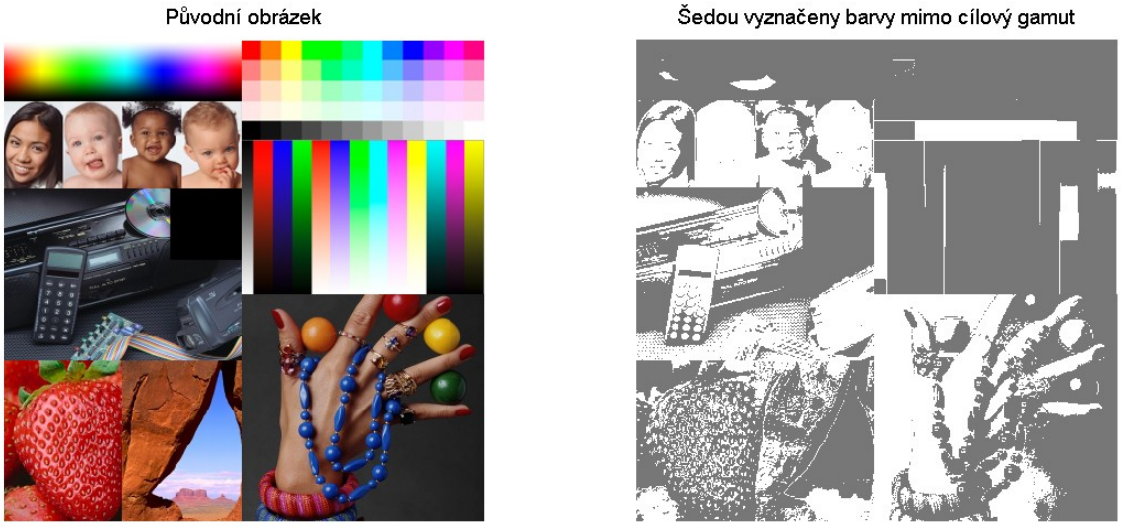
Z kódu je patrné, že zde probíhá převod bodů do komplexních čísel. Průsečík se segmenty řezu by mohl být nalezen pomocí již zmiňované funkce `findintersection`, která průsečík spolehlivě najde pomocí determinantů. Ve fázi optimalizace výpočtu však bylo zjištěno, že tento způsob nalezení průsečíku v komplexních číslech je na výpočet mnohem rychlejší a jednodušší. Úsečka střed – barevný bod je tedy převedena do komplexních čísel a uložena do proměnné `seg`. Převod, probíhající s využitím funkce `complex`, z první souřadnice bodu (x – chromatičnost C) vytvoří reálnou složku a z druhé (y – světlost L^*) složku imaginární. Stejně tak body řezu jsou uloženy jako komplexní čísla do vektoru `pline`. Jedno komplexní číslo tak reprezentuje bod se dvěma souřadnicemi. Dále je volána funkce `PolylineIntersectSegment` (převzata od Alvareze, teoretické odvození v [35]), která provádí nalezení a v případě existence výpočet průsečíku. Výhodou při práci s komplexními čísly reprezentujícími vektory je např. to, že normálový vektor je možno vypočítat prostým součinem s imaginární jednotkou i apod. Komentovaný zdrojový kód funkce je uveden níže. Výsledky přepočtu gamutů pomocí algoritmu SCLIP jsou uvedeny v kapitole 3.2.

```
nlines = numel(pline) - 1; %počet segmentů řezu o 1 nižší než bodů
P1 = seg(1)*ones(nlines,1); %generování kopií úsečky střed - barevný bod
d1  = (seg(2) - seg(1) )*ones(nlines,1); %zápis úsečky: P1 + s*d1
P2 = pline(1:end-1); %generování segmentů řezu
d2 = diff(pline); % zápis segmentů: P2 + t*d2
ns_1 = 1i*d1; % rotace diferenčních vektorů k získání kolmých vektorů
ns_2 = 1i*d2;
Ds = P2 - P1; %diferenční vektor počátečních bodů
s = zdot(Ds,ns_2)./zdot(d1,ns_2); %výpočet vzdáleností
t = -zdot(Ds,ns_1)./zdot(d2,ns_1); %počátečního bodu a průsečíku
intersectsOK = (s>=0)&(s<1)&(t>=0)&(t<1); % průsečík(y) existuje(jí)

if any(intersectsOK)
   zi = seg(1) + s(intersectsOK)*d1(1); %ulož průsečík
   is = true; %průsečík existuje
else
   zi = []; %ulož prázdnou množinu
   is = false; %průsečík neexistuje
end
% lokální funkce pro skalární součin komplexních vektorů:
    function d = zdot(z1,z2)
    d = real( z1(:) .* conj(z2(:)));
```

Pro názornost je ještě uveden obrázek 2.11, který zobrazuje řez barevným prostorem před a po mapování pomocí algoritmu SCLIP. Z obrázku je názorně vidět, že body uvnitř cílového gamutu zůstaly nezměněny, zatímco body mimo cílový gamut nyní leží na hranici cílového gamutu. Na obrázku 2.12 jsou potom názorně zobrazeny přímo pixely testovacího obrázku, kterých se přepočet týkal – šedou barvu mají ty pixely, které leží mimo cílový gamut.



Obr. 2.11: Řez bar. prostorem před mapováním (vlevo) a po mapování (vpravo).

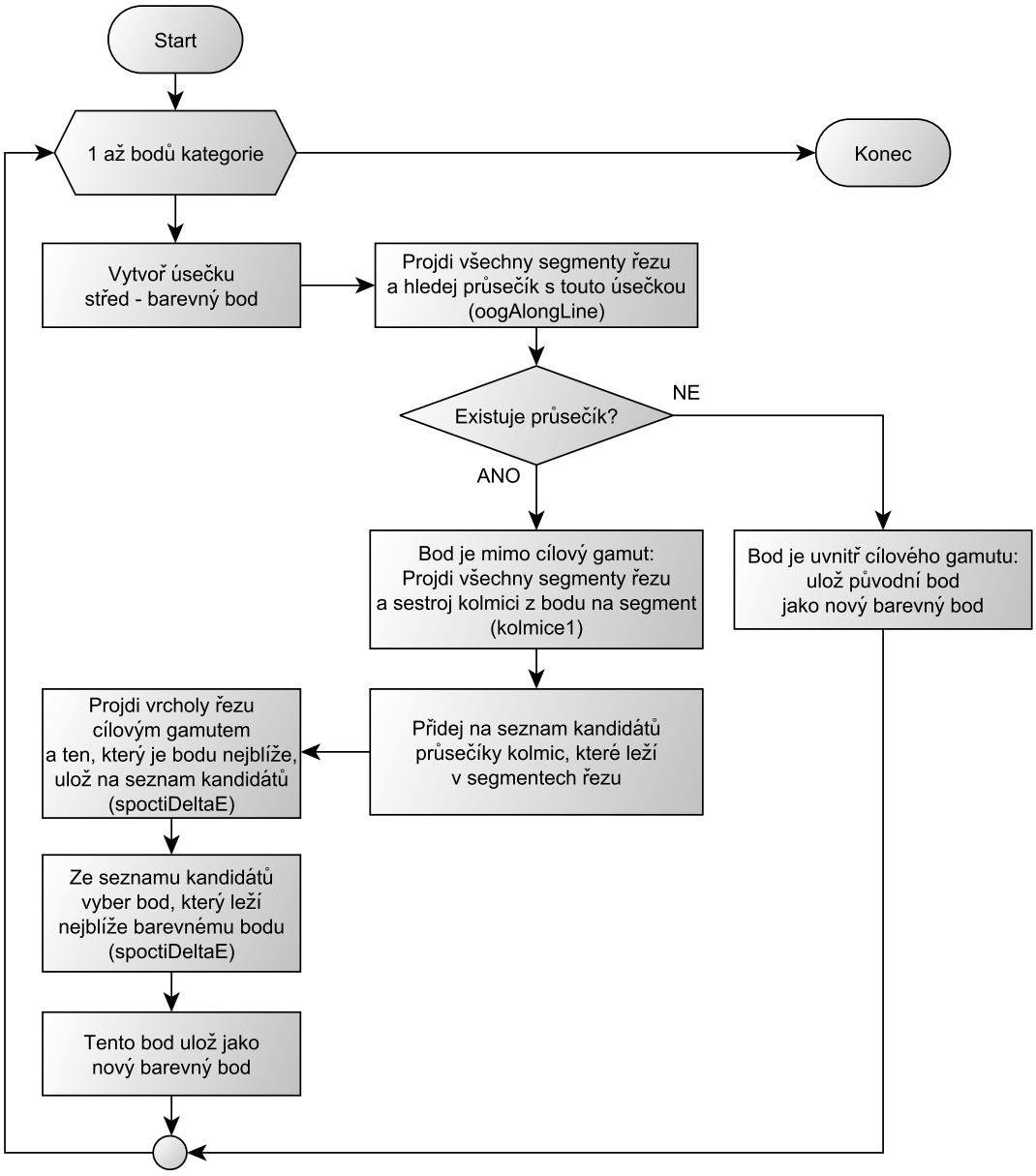


Obr. 2.12: Pixely, které leží mimo cílový gamut mají v pravém obrázku šedou barvu. Bílá barva v tomtéž obrázku představuje pixely beze změny.

2.3 Algoritmus HPMINDE

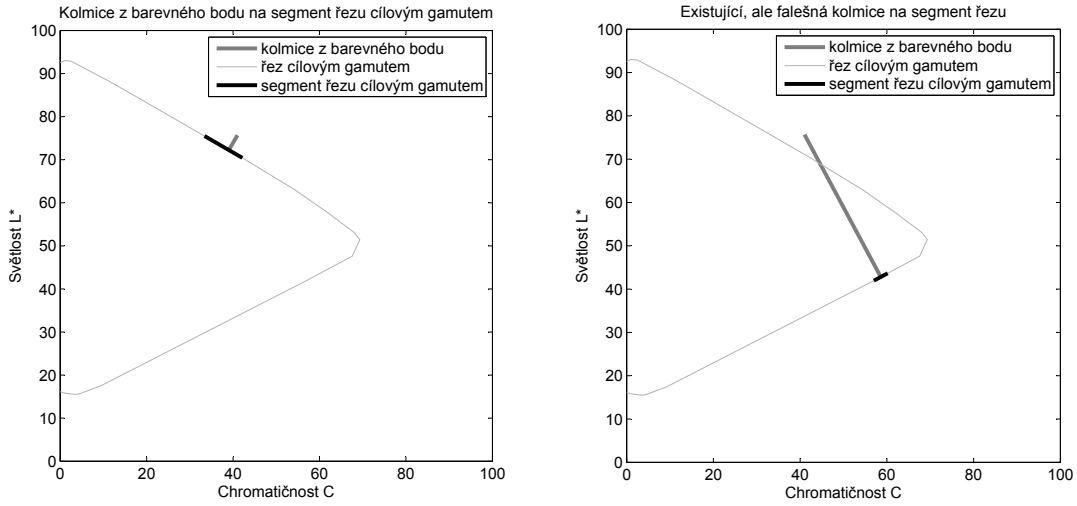
Algoritmus SCLIP byl vysvětlen jako první, protože je jednodušší na implementaci. I když následující algoritmus HPMINDE vychází z jednodušší myšlenky nalezení co nejbližší barvy podle ΔE, je do značné míry nástavbou na algoritmus SCLIP (využívá výše popsané funkce k určení, zda barva leží mimo nebo uvnitř cílového gamutu). Algoritmus HPMINDE je v podstatě ortogonální projekcí na konvexní obálku (cílový gamut), ovšem v rovinách dle odstínu (odtud i jeho název Hue Planes Minimum Delta E) a v roce 1989 s ním přišli Murch and Taylor [3]. Morovič ve své publikaci [3] podrobně popisuje algoritmus *minimální* ΔE, který ovšem probíhá v trojrozměrném prostoru. Toto je tedy zjednodušená verze v tom smyslu, že výpočet probíhá v dvojrozměrném prostoru.

Princip algoritmu je znázorněn na vývojovém diagramu (obr. 2.13). Opět zde probíhá cyklus, kdy je postupně brán každý barevný bod kategorie (odstínu) a je nejprve potřeba zjistit, zda bod leží uvnitř, nebo mimo cílový gamut. To stejně jako u algoritmu SCLIP zajišťuje funkce `oogAlongLine`, ovšem vypočtený průsečík je nyní ignorován. Pokud bod leží mimo gamut, jsou v cyklu znovu procházeny segmenty řezu a je hledán segment, na který existuje kolmice. To vykonává funkce `kolmice1`, která zároveň vrací bod, v němž se kolmice protíná se segmentem. Segmentů, pro něž existuje takový bod, může být více, proto jsou body průsečíku segmentu a kolmice ukládány na seznam kandidátů. Navíc nemusí kolmice na žádný ze segmentů existovat, nebo může kolmice existovat jen na segment, který není v blízkosti barevného

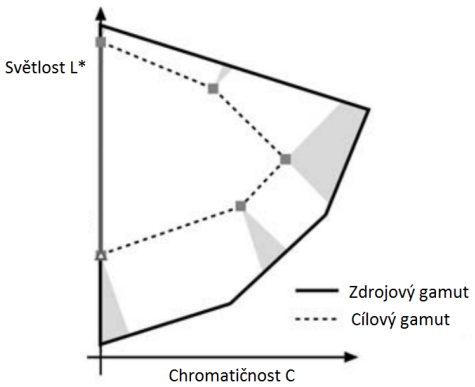


Obr. 2.13: Vývojový diagram – princip algoritmu HPMINDE

bodu, ale na vzdálené straně řezu cílovým gamutem. Proto je nutné projít i body řezu cílovým gamutem, vybrat ten, který je barevnému bodu nejblíže a také jej uložit na seznam vhodných kandidátů. To je vykonáno pomocí funkce `spoctiDeltaE`. Nakonec se ze seznamu kandidátů vybere bod, který je barevnému bodu nejblíže, podle normy ΔE (viz rovnice 1.15), opět pomocí funkce `spoctiDeltaE`. Z následujícího obrázku (2.14) je patrné, že může existovat více segmentů, na něž existuje kolmice a proč je nutné vybírat nejbližší z těchto bodů (průsečíků kolmice a segmentu). Na následujícím obrázku 2.15 je také vidět nutnost procházet body řezu cílovým gamutem, protože barevný bod může ležet v jedné z šedě vyznačených oblastí, kde neexistuje kolmice na žádný ze segmentů.



Obr. 2.14: Kolmice z barevného bodu na segment řezu cílovým gamutem. Vpravo hledaný bod v nejbližším segmentu, vlevo falešný bod ve vzdáleném segmentu.



Obr. 2.15: Oblasti mapování na body řezu cílovým gamutem (šedě). Převzato a upraveno z [3].

Níže je uveden komentovaný zdrojový kód zmíněných funkcí. Nejprve zdrojový kód funkce `mapujMinDE`:

```
[boduKat ~] = size(bodyKat); [boduRezu ~] = size(bodyRezu); % počet bodů
noveBodyKat = zeros(boduKat(1),4); % alokace

for i = 1:boduKat % procházíme body kategorie
     [isOOG ~] = oogAlongLine( bodyRezu , bodyKat(i,2:-1:1) );
     kand = []; % vynulujeme množinu vhodných kandidátů
     if(isOOG) % pokud je bod mimo cílový gamut
        for j = 1:(boduRezu-1) % projdi body řezu cílovým gamutem
            [ nejBod isInSeg] = kolmice1( bodyKat(i,2:-1:1), ...
                bodyRezu(j,:), bodyRezu(j+1,:) );
            if(isInSeg) % průsečík s kolmicí je v segmentu, uložíme
                kand = [kand; nejBod(2), nejBod(1), bodyKat(i,3:end)];
            end
        end
        %projdeme ještě body řezu cílovým gamutem, nejbližší uložíme
        [ nejVrchol ] = spoctiDeltaE( bodyKat(i,2:-1:1), bodyRezu );
        kand = [kand; nejVrchol(2:-1:1), bodyKat(i,3:end)];
        % vyber nejbližšího kandidáta
        [ novyBod ] = spoctiDeltaE( bodyKat(i,2:-1:1), kand(:,2:-1:1) );
        % zápis nejbližšího kandidáta jako nový bod
        noveBodyKat(i,:) = [ novyBod(2:-1:1), bodyKat(i,3:end)];
     else % bod leží uvnitř cílového gamutu, ponechat beze změny
        noveBodyKat(i,:) = bodyKat(i,:);
     end
end
```

Následuje zdrojový kód funkce `kolmice1`, při jejíž tvorbě bylo čerpáno z Bourkeho [36]. Body A a B jsou body úsečky segmentu řezu a bod C je barevný bod. Hledaný bod D je průsečíkem kolmice z barevného bodu C a segmentu AB.

```
function [ bodPrus isInSeg ] = kolmice1( bod, bod1seg, bod2seg )
Ax = bod1seg(1); Ay = bod1seg(2); % první bod segmentu
Bx = bod2seg(1); By = bod2seg(2); % druhý bod segmentu
Cx = bod(1); Cy = bod(2); % barevný bod mimo cílový gamut

% řešení rovnice [C - A - t(B - A)] x (B - A) = 0; x je skalární součin
t=( (Cx-Ax)*(Bx-Ax)+(Cy-Ay)*(By-Ay) )/( (Bx-Ax)^2+(By-Ay)^2 );
Dx=Ax+t*(Bx-Ax); % výpočet souřadnic průsečíku
Dy=Ay+t*(By-Ay);
bodPrus = [Dx Dy]; %sloučení souřadnic do vektoru - výsledný bod
```

```
    if(t >= 0 && t <= 1) % posouzení, zda průsečík leží v segmentu
        isInSeg = true;
    else
        isInSeg = false;
    end
end
```

Jde vlastně o nalezení nejmenší vzdálenosti mezi úsečkou a bodem. Následuje zdrojový kód funkce pro nalezení nejbližšího bodu pomocí normy ΔE:

```
function [ nejblizsiBod deltaE ] = spoctiDeltaE( bod, viceBodu )
% zkopírujeme bod do délky porovnávané matice s více body
bodOpakovany = repmat(bod,size(viceBodu,1),1);
% výpočet vektoru vzdáleností podle Delta E
vzdalenosti = (sum(((bodOpakovany-viceBodu).^2), 2)).^0.5;
[deltaE ind] = min(vzdalenosti); % vyhledání nejmenší vzdálenosti
nejblizsiBod = viceBodu(ind,:); % zápis výsledku
end
```

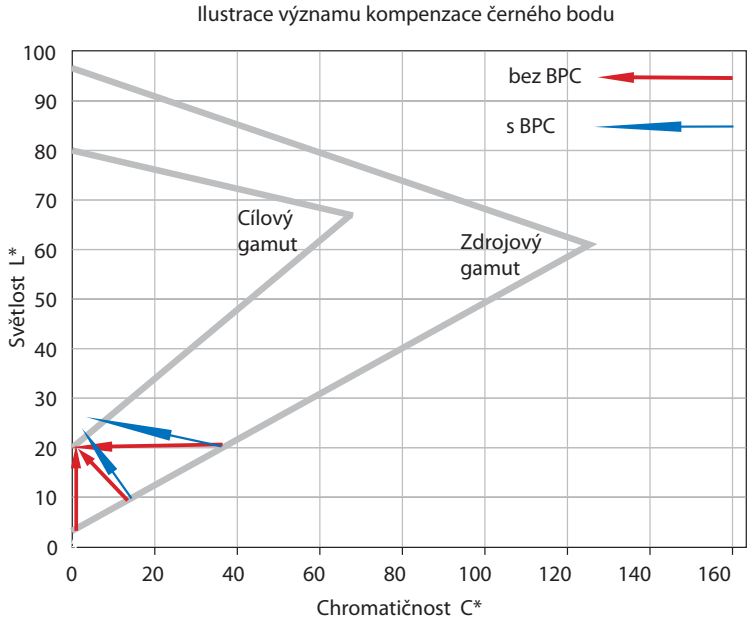
Výsledky přepočtu barev testovacího obrázku jsou uvedeny v kapitole 3.

2.4 Kompenzace černého bodu

Důvodem zavedení kompenzace černého bodu (BPC – Black Point Compensation) je lepší zachování detailů v tmavých odstínech a vyvarování se uměle vypadajících výsledků, kdy v důsledku menšího cílového gamutu je mnoho bodů mapováno na stejný bod cílového gamutu [3, 37]. Výsledky vlivu kopmenzace černého bodu viz kap. 3.4. Čeho je při kompenzaci černého bodu dosaženo také názorně přibližuje obr. 2.16. Při implementaci bylo čerpáno od Moroviče [3], který uvádí v praxi běžně užívanou kompenzaci černého bodu (např. v produktech Adobe). Jedná se o úpravu barevných bodů v prostoru CIE XYZ ještě před samotným přepočtem barevných bodů na cílový gamut (jak je patrno z vývojového diagramu skriptu `main.m` – obr. 2.7, kde je před přepočtem kompenzace černého bodu volitelně zavedena). Rozsah zdrojových barevných bodů je upraven na rozsah cílového gamutu a to takovým způsobem, že bílé body zdrojového i cílového prostoru jsou sesazeny na sebe (reprezentovány stejnými hodnotami CIE XYZ) a rozsah je upravován vzhledem k černému bodu (ten, který má nejnižší hodnotu L^* v prostoru CIE LAB). Úprava bodů probíhá následujícím způsobem:

$$\begin{bmatrix} X_{BPC} \\ Y_{BPC} \\ Z_{BPC} \end{bmatrix} = \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} - \begin{bmatrix} \frac{Y_W - Y_{BPD}}{Y_W - Y_{BPS}} & 0 & 0 \\ 0 & \frac{Y_W - Y_{BPD}}{Y_W - Y_{BPS}} & 0 \\ 0 & 0 & \frac{Y_W - Y_{BPD}}{Y_W - Y_{BPS}} \end{bmatrix} \begin{bmatrix} X_W - X_S \\ Y_W - Y_S \\ Z_W - Z_S \end{bmatrix} \tag{2.1}$$

Zde X_{BPC}, Y_{BPC}, Z_{BPC} jsou výsledné body po kompenzaci černého bodu a X_S, Y_S, Z_S jsou původní body před kompenzací (S dle anglického source – zdrojové, D jako destination – cílové). X_W, Y_W, Z_W jsou souřadnice bílého bodu (CIE iluminant D50). Ve členu, kterým jsou původní souřadnice modifikovány (v diagonále matice) pak figuruje jasová hodnota cílového černého bodu Y_{BPD} a zdrojového černého bodu Y_{BPS}. Odečtením od jasové souřadnice bílého bodu jsou jasové složky černých bodů normalizovány. Díky aplikaci tohoto lineárního škálování dojde ke kolorimetrické změně zdrojových barevných bodů, která je podobná jako u reálných objektů při změně (snížení) intenzity osvětlení, kdy se lidské oko adaptuje a detaily i při sníženém osvětlení vnímá. [3]



Obr. 2.16: Význam kompenzace černého bodu. Bez kompenzace jsou tmavé barevné body mapovány na nejtmavší bod cílového gamutu, s kompenzací je mezi tmavými barevnými body i v cílovém gamutu zachován rozdíl. Převzato a upraveno z [37].

Níže je uveden komentovaný zdrojový kód funkce, kterou volá skript `main.m` a jež kompenzaci černého bodu provádí. Do funkce vstupují barevné body obrázku i cílového gamutu a počet barev v obrázku. Výstupem jsou upravené barevné body obrázku.

```
function [ labO ] = blackPointComp( labO, labZ, barev )

TrLab2Xyz = makecform('lab2xyz'); % převod do CIE XYZ
xyzO = applycform(labO(:,1:3),TrLab2Xyz);

% konstanty pro výpočet:
[hodnotaBP indexBP] = min(labZ);
BP = labZ(indexBP(1),:);
BP = applycform(BP,TrLab2Xyz);
Ybpd = BP(2); % cílový černý bod
Ybps = min(xyzO(:,2)); % zdrojový černý bod
xyzW = whitepoint('d50'); % bílý bod
diagKonst = (xyzW(2) - Ybpd)/(xyzW(2) - Ybps); % diagonála matice
matice = diag([diagKonst diagKonst diagKonst]);

xyzBPC(3,barev) = 0; % alokace výsledných barevných bodů
% výpočet dle vztahu 2.1
for i = 1:barev
    xyzBPC(:,i) =  xyzW' - matice*(xyzW' - xyzO(i,:)');
end

TrXyz2Lab = makecform('xyz2lab'); % převod zpět do Lab
labO(:,1:3) = applycform(xyzBPC',TrXyz2Lab);

end
```

3 SROVNÁNÍ VÝSLEDKŮ PŘEPOČTŮ GAMUTŮ

Závěrečná kapitola uvádí výsledky výše popsaných algoritmů a jejich srovnání. Protože bylo hodnocení a srovnání výsledků prováděno na monitoru počítače (referenční barevný prostor byl sRGB), je možné, že v tištěné práci budou výsledky např. méně patrné, rozdíly méně výrazné. Některé listy však byly vytištěny na lesklý papír pro dosažení většího rozsahu barev. Pokud některé popisované rozdíly nebudou z tištěných výsledků zřejmé, všechny uvedené výsledky je možno nalézt v elektronické podobě na přiloženém CD ve složce `vysledky`. Jak již bylo zmíněno, zdrojovým gamutem je gamut obrázku převedený z prostoru sRGB, cílovým gamutem je gamut tiskárny Epson Stylus DX5050 – tisk inkoustem ABEL na papír Epson Photo Quality S041061. Z hlediska správy barev a způsobu vykreslení se potom jedná o absolutně kolorimetrický převod, protože bílý bod zdroje nebyl mapován na bílý bod cíle. Jedná se v podstatě o simulaci výstupu z tiskárny. Barevné obrázky jsou přímo jedním z výstupů skriptu `main.m`, obrázky znázorňující odchylku ΔE ve stupních šedi jsou pak výstupem již zmiňovaného skriptu `porovnejObrazky.m`.

3.1 Výsledky algoritmu HPMINDE

Na obrázku 3.1(b) je uveden výsledek algoritmu HPMINDE, původní obrázek je zobrazen na obr. 3.1(a). Morovič uvádí, že výsledek mapování algoritmem minimální ΔE má menší kontrast, vypadá méně barevně, v některých částech obrazu vykazuje ztrátu detailů atd. [3]. Zde se jedná o algoritmus HPMINDE, který zachovává odstín a minimální odchylka ΔE je aplikována pouze na hodnoty světlosti a chromatičnosti. I tak lze však pozorovat v literatuře popsané vlastnosti. Výsledný obrázek má méně kontrastu, hlavně v tmavých oblastech (magnetofon, kalkulačka), což vede ke ztrátě detailů v těchto oblastech. Menší barevnost je také patrná, barvy nedosahují takové chromatičnosti jako v originálním obrázku. Lze to pozorovat i u černé a bílé – černá je po mapování světlejší, bílá tmavší. Nejmarkantnějšího barevného rozdílu si pak lze všimnout v modrých a fialových barvách. Pravděpodobně to souvisí s již zmiňovanou uniformitou, resp. neuniformitou barevného modelu CIE LAB a také literatura poukazuje na největší neuniformitu tohoto modelu v modrých barvách [3]. Červené barvy také působí poněkud sytě. V horní levé části obrázku (přechody barev) si lze povšimnout jistých blokových artefaktů, jež jsou patrně způsobeny tím, že více barev zdroje bylo mapováno na jednu barvu cíle (např. na body řezu cílovým gamutem, obr. 2.15) a přechody pak nepůsobí plynule jako v originálu.

Na obrázku 3.2(b) je zobrazena barevná odchylka ΔE ve stupních šedi pro jednotlivé pixely. Škála je taková, že bílá odpovídá nulové odchylce ΔE a plná černá odpovídá odchylce 100 ΔE. Z obrázku je zřejmé, v jakých oblastech byla barevná odchylka nejvyšší, jedná se o modré a pak zelené a červené barvy. Nejvyšší odchylka v celém obrázku byla přibližně 84 ΔE.

3.2 Výsledky algoritmu SCLIP

Literatura hodnotí výsledky algoritmu SCLIP lépe, než výsledky algoritmu HPMINDE, především proto, že u výsledků algoritmu SCLIP se nevyskytují blokové artefakty [3]. Výsledek algoritmu SCLIP je vyobrazen na obrázku 3.1(c), barevná odchylka vůči originálu ΔE pak na obrázku 3.2(c). Při pohledu na výsledek je opět patrné velké zkreslení v modrých barvách, patrně ještě větší než u mapování pomocí algoritmu HPMINDE (tmavě modrá působí fialově). Ztráta barevnosti je zde srovnatelná s výsledkem algoritmu HPMINDE, ztráta detailu také (např, v tmavých barvách – magnetofon), ovšem červené barvy nejsou tak syté a jsou podobnější originálu. Např. detaily v dolní části jahody jsou zde lépe zachovány než u výsledku algoritmu HPMINDE.

Na obrázku 3.2(c) je opět se škálou 0 až 100 ΔE zobrazena barevná odchylka. Opět je patrné, že barevná odchylka je nejvyšší v modrých, následně v zelených a červených barvách. Nejvyšší barevná odchylka v obrázku činí téměř 89 ΔE, což je více než u algoritmu HPMINDE. Na první pohled je zde barevná odchylka téměř totožná jako u algoritmu HPMINDE, mírně pozorovatelný rozdíl je např. u jahody a červené koule, což potvrzuje i obrázek 3.4(a), který ve stejné škále zobrazuje barevnou odchylku mezi výsledky mapování obou algoritmů.

3.3 Porovnání s přepočtem v Adobe Photoshop

Na obrázku 3.1(d) je výsledek přepočtu, který byl realizován v programu Adobe Photoshop, verze 12 (CS5) [38]. Ten umožňuje obrázek tkzv. převést do profilu (v nabídce Úpravy se nachází možnost Převést do profilu). Tímto způsobem byl obrázek převeden z pracovního prostoru sRGB (standartně pracovní prostor programu Adobe Photoshop) do ICC profilu tiskárny, který vychází ze stejných naměřených dat, jež obsahuje vstupní soubor pro algoritmy přepočtů gamutů (když byla data pomocí programu X-Rite a kolorimetrické sondy [7] naměřena, byl spolu s nimi uložen také ICC profil na základě těchto dat). Převod do profilu v Adobe Photoshop umožňuje volbu způsobu vykreslení a také výběr modulu správy barev. Zde byl vybrán způsob převodu absolutně kolorimetrický a modul správy barev byl ponechán

(a) Testovací obrázek – originál

(b) Výsledek algoritmu HPMINDE

(c) Výsledek algoritmu SCLIP

(d) Převod pomocí Adobe Photoshop

Obr. 3.1: Výsledky přepočtů gamutů

(a) Originál – odchylka ΔE je nulová

(b) ΔE výsledku algoritmu HPMINDE

(c) ΔE výsledku algoritmu SCLIP

(d) ΔE převodu pomocí Adobe Photoshop

Obr. 3.2: Odchylka ΔE vůči originálnímu testovacímu obrázku. Bílá odpovídá 0 ΔE, černá 100 ΔE.

standartní Adobe. Po převodu do profilu tiskárny byl pak obrázek opět převeden do barevného prostoru sRGB (opět absolutně kolorimetricky), aby byl zachován stejný postup jako u algoritmů přepočtů realizovaných v MATLABu.

Při srovnání výsledků (obr. 3.1(d)) je patrné, že přepočet pomocí Adobe Photoshop výrazně lépe zachoval modré barvy, které nejsou tolik fialové jako u přepočtu pomocí algoritmů SCLIP nebo HPMINDE. Zachování detailů je srovnatelné s výsledkem algoritmu SCLIP (jahoda) a v tmavých oblastech (magnetofon) i s algoritmem HPMINDE. Barevnou odchylku opět ve škále 0 až 100 ΔE zobrazuje obr. 3.2(d). Nejvyšší odchylka v obrázku činila 87 ΔE, což je více, než u algoritmu HPMINDE a méně neý u algoritmu SCLIP. Z obrázku na první pohled není příliš patrný větší rozdíl oproti oběma realizovaným algoritmům, obr. 3.4(c) a (d) však ukazují, že rozdíly existují.

3.4 Vliv kompenzace černého bodu

Obrázek 3.3 demonstruje výsledky algoritmů HPMINDE a SCLIP s kompenzací černého bodu (BPC), obsahuje také srovnání s převodem do profilu tiskárny s kompenzací černého bodu provedené v Adobe Photoshop. V levém sloupci obr. 3.3 je originální obrázek a pod ním již prezentované výsledky algoritmů HPMINDE a SCLIP. V pravém sloupci jsou pak přepočty s kompenzací černého bodu – nahoře v Adobe Photoshop a pak příslušně k levým obrázkům pomocí obou algoritmů.

Rozdíl je patrný v tmavých místech obrázku (magnetofon a černý čtverec). U originálního obrázku lze rozeznat hranu černého čtverce od magnetofonu na pozadí, ve výsledcích algoritmů HPMINDE, SCLIP i převodu pomocí Adobe Photoshop bez kompenzace černého bodu se tato hrana však slévá v jednu barvu a není rozeznatelná. Napravo, v obrázcích s kompenzací černého bodu je však hrana černého čtverce opět patrná a tmavé části obrazu obsahují více detailů. Vliv kompenzace černého bodu je také patrný na pozadí ruky. V obrázcích bez kompenzace černého bodu působí pozadí jako tmavší, v obrázcích s kompenzací černého bodu podle očekávání světleji a kontrast ruky na pozadí tak lépe zachovává kontrast v originálním obrázku. Za povšimnutí stojí změna v červených barvách (jahoda, červená koule) u algoritmu HPMINDE s kompenzací černého bodu. Červená barva je zde tmavší, detaily v jahodě lépe zachovány, takže se tento výsledek blíží výsledku algoritmu SCLIP s kompenzací černého bodu.

Nutno ještě podotknout, že v případě výsledků realizovaných algoritmů s kompenzací černého bodu se již nejedná o absolutně kolorimetrický způsob vykreslení, ale tím, že jsou hodnoty barevných bodů upraveny, se jedná o verzi relativně kolorimetrického způsobu vykreslení (ovšem nepřesně, protože zde nedochází k úpravě

bílého bodu). Adobe Photoshop například logicky ani neumožňuje u absolutně kolorimetrického způsobu zvolit možnost kompenzace černého bodu, proto výsledek přepočtu s kompenzací černého bodu z Adobe Photoshop byl realizován relativně kolorimetrickým způsobem vykreslení (což je patrné např. na bílé barvě, která zde nemá tak studenou teplotu jako u výsledků obou algoritmů).

3.5 Porovnání výsledků mezi sebou

Srovnání některých výsledků pak ukazuje obr. 3.4. Zde opět bílá představuje odchylku 0 ΔE a sytě černá 100 ΔE. Rozdíl mezi algoritmem HPMINDE a SCLIP na obr. 3.4(a) byl již zmíněn, maximální odchylka je zde zhruba 23 ΔE. Z druhého obrázku 3.4(b) je vidět, že rozdíly mezi oběma realizovanými algoritmy s kompenzací černého bodu nejsou příliš velké, barevná odchylka zde dosahuje max. 20 ΔE. Na dalších obrázcích 3.4(c) a (d) jsou pal rozdíly mezi realizovanými algoritmy a převodem v Adobe Photoshop. Odchylka se zde pohybuje kolem 23 ΔE u algoritmu HPMINDE a 21 ΔE u algoritmu SCLIP. Poslední dvojice obrázků 3.4(e) a (f) představuje rozdíl mezi oběma realizovanými algoritmy s kompenzací černého bodu a bez této kompenzace. Zde je vidět, v jakých oblastech obrázku jsou změny, a je patrné, že změny nejsou příliš velké. Hodnoty barevné odchylky jsou zde 16 ΔE u algoritmu HPMINDE a 10 ΔE u algoritmu SCLIP.

Pro shrnutí ještě porovnání základních výsledků: Z obrázku 3.1 je patrné, že při přepočtu pomocí algoritmu HPMINDE působí některé barvy (např. červená) sytěji, než při přepočtu pomocí algoritmu SCLIP či Adobe Phtoshop a objevují se blokové artefakty. Je možno si všimnout, že však algoritmus HPMINDE lépe zachovává modrou barvu, než výsledek kalgoritmus SCLIP. Ten je na tom ze zobrazených výsledků s modrou barvou nejhůře, neobsahuje však narozdíl od HPMINDE blokové artefakty, což je předpokládáaný výsledek podle literatury [3]. Nejlépe modrou barvu zachovává převod pomocí Adobe Photoshop. Subjektivně i objektivně (obr. 3.4(d)) je výsledek algoritmu SCLIP podobnější převodu pomocí Adobe Photoshop. S kompenzací černého bodu jsou všechny výsledky obecně lepší (obr. 3.3), subjektivně více podobné originálu (zachované detaily v tmavých barvách vypadají realističtěji). Použití kompenzace černého bodu pak do značné míry stírá rozdíly mezi výsledky algoritmů HPMINDE a SCLIP.

(a) Testovací obrázek – originál

(b) Adobe Photoshop s BPC

(c) Algoritmus HPMINDE

(d) Algoritmus HPMINDE s BPC

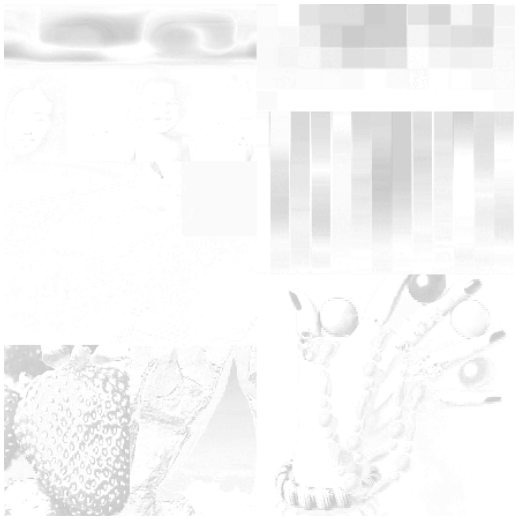
(e) Algoritmus SCLIP

(f) Algoritmus SCLIP s BPC

Obr. 3.3: Vliv kompenzace černého bodu

(a) HPMINDE a SCLIP

(b) HPMINDE s BPC a SCLIP s BPC



(c) HPMINDE a Adobe Photoshop

(d) SCLIP a Adobe Photoshop

(e) HPMINDE a HPMINDE s BPC

(f) SCLIP a SCLIP s BPC

Obr. 3.4: Porovnání výsledků pomocí metriky ΔE. Bílá odpovídá 0 ΔE, černá 100 ΔE.

4 ZÁVĚR

První část práce představila nutný teoretický základ pro algoritmy přepočtů gamutů. Byly v ní vysvětleny vlastnosti barvy, jak ji vnímá člověk, jak se měří. Byla představena kolorimetrie jako snaha o co nejvěrnější zachování barev napříč různými zařízeními. Byly vysvětleny barevné modely používané ve správě barev a převody mezi nimi. Naměření ICC profilů několika zařízení pomocí kolorimetrické sondy posloužilo k seznámení se správou barev. Byly rozvedeny čtyři základní způsoby přepočtů gamutů (způsoby vykreslení) ve správě barev dle specifikací ICC: perceptuální, systostní (komprese gamutu), relativní kolorimetrický a absolutní kolorimetrický (oříznutí gamutu). Dále byl z literatury (převážně [3]) nastudován a uveden přehled přístupů k přepočtům gamutů. Bylo zjištěno, že univerzální algoritmus, který by se hodil pro všechny aplikace přepočtů gamutů zatím neexistuje a porovnávání výsledků jednotlivých algoritmů mezi sebou je velmi složité, protože do procesu vstupuje obrovské množství proměnných. Konsorcium ICC nabízí určitá vodítka, na jejichž základě je možné algoritmy mezi sebou porovnávat.

Druhá část práce podrobně popsala přípravu dat pro algoritmy přepočtů gamutů a následně implementaci dvou algoritmů přepočtů gamutů v prostředí MATLAB, navíc s kompenzací černého bodu.

Výsledky převodů pomocí těchto algoritmů pak byly prezentovány ve třetí části práce, včetně srovnání s běžně používanými převody pomocí Adobe Photoshop.

Výsledky potvrdily vlastnosti implementovaných algoritmů, jež zmiňuje literatura [3]. Výsledek algoritmu HPMINDE vykazuje nejmenší barevnou odchylku ΔE, lépe zachovává modré barvy, ale může obsahovat blokové artefakty a působí méně přirozeně. Tyto artefakty neobsahuje výsledek algoritmu SCLIP, který je blíže převodu pomocí Adobe Photoshop, ovšem modré barvy zachovává hůře než HPMINDE. Použití kompenzace černého bodu výsledky obecně zlepšuje, zároveň do značné míry stírá rozdíly mezi algoritmy HPMINDE a SCLIP. Práce tak ověřila v literatuře zmiňovaný fakt, že přepočet pomocí nejbližších barev (minimální ΔE) nemusí být z hlediska lidského vnímání barev vždy nejlepší (což potvrzují i v praxi u fotografií používané způsoby vykreslení – perceptuální nebo relativně kolorimetrický s kompenzací černého bodu).

Do budoucna by bylo možné nadále optimalizovat zdrojový kód implementovaných algoritmů, aby výpočet probíhal ještě rychleji. Také by bylo možno navázat implementací přepočtu pomocí váhovaného ΔE či kompresní verze algoritmu SCLIP – SLIN (perceptuální způsob vykreslení) a bylo by možné na vytvořeném základě přípravy dat pro přepočty gamutů realizovat a srovnat mnoho dalších algoritmů.

LITERATURA

[1] BERNS, Roy S. *Billmeyer and Saltzman´s Principles of Color Technology.* 3rd ed. New York: John Wiley & Sons, Inc., 2000, ix, 247 s. ISBN 04-711-9459-X.

[2] FRASER, Bruce, Chris MURPHY a Fred BUNTING. *Správa barev: průvodce profesionála v grafice a pre-pressu.* Vyd. 1. Překlad Milan Daněk. Brno: Computer Press, 2003, 521 s. ISBN 80-722-6943-7.

[3] MOROVIČ, Ján. *Color gamut mapping.* Chichester: John Wiley, c2008, xiv, 287 s., [14] s. obr. příl. Wiley-IS. ISBN 978-0-470-03032-5.

[4] WANDELL, Brian A. STANFORD UNIVERSITY. *Foundations of Vision* [online]. [cit. 2013-11-25]. Dostupné z: <https://www.stanford.edu/group/vista/cgi-bin/FOV/>.

[5] MACHADO, G.M., M.M. OLIVEIRA a L. FERNANDES. A Physiologically-based Model for Simulation of Color Vision Deficiency. *IEEE Transactions on Visualization and Computer Graphics* [online]. 2009, vol. 15, issue 6, s. 1291-1298 [cit. 2014-05-29]. DOI: 10.1109/TVCG.2009.113. Dostupné z: <http://ieeexplore.ieee.org/ieee_pilot/articles/06/ttg2009061291/article.html>.

[6] Linear visible spectrum. In: Wikimedia Commons [online]. K dispozici pod licensí Public Domain [cit. 2013-12-30]. Dostupné z: <http://commons.wikimedia.org/wiki/File:Linear_visible_spectrum.svg>.

[7] *X-Rite – Right On Color* [online]. 2013 [cit. 2013-12-30]. Dostupné z: <http://www.xrite.com/>.

[8] *CIE – International Comission On Illumination* [online]. 2013 [cit. 2013-12-30]. Dostupné z: <http://www.cie.co.at/>.

[9] STECÍK, Július a Pavel RAJMIC. Vnímání barvy člověkem, prostory XYZ a Lab – applet. *Ústav Telekomunikací: Fakulta elektrotechniky a komunikačních technologií, Vysoké učení technické v Brně* [online]. 2013 [cit. 2013-12-30]. Dostupné z: <http://www.utko.feec.vutbr.cz/~rajmic/applets/clovek-spektrum.html>.

[10] RAJMIC, Pavel. *Základy počítačové sazby a grafiky.* Vyd. 1. Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací, 2012. ISBN 978-80-214-4451-5.

[11] RGB & CMYK – Concepts and Differences. *Mobilio: Impressive App Development* [online]. 2012 [cit. 2013-12-30]. Dostupné z: <http://www.mobiliodevelopment.com/rgb-cmyk-concepts-and-differences/>.

[12] Computing XYZ From Spectral Data (Emissive Case). *Bruce Lindbloom* [online]. 2003 [cit. 2013-12-30]. Dostupné z: <http://www.brucelindbloom.com/index.html?Eqn_Spect_to_XYZ.html>.

[13] Color Think Pro. *Chromix* [online]. 2013 [cit. 2013-12-30]. Dostupné z: <http://www.chromix.com/colorthink/>.

[14] CHROMiX ColorNews Issue #17 – Delta-E: The Color Difference. *Chromix* [online]. 2005 [cit. 2013-12-30]. Dostupné z: <http://www.chromix.com/colorsmarts/smartNote.cxsa?snid=1145>.

[15] Color difference. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013 [cit. 2013-12-30]. Dostupné z: <http://en.wikipedia.org/wiki/Color_difference>.

[16] WCS Gamut Map Model Profile Schema and Algorithms. *Windows: Dev Center – Desktop* [online]. 2010 [cit. 2013-12-30]. Dostupné z: <http://msdn.microsoft.com/en-us/library/windows/desktop/dd372430(v=vs.85).aspx>.

[17] Modern Color Models: CIECAM Color Appearance Model. MACEVOY. *Color Vision* [online]. 2005 [cit. 2013-12-30]. Dostupné z: <http://www.handprint.com/HP/WCL/color7.html#CIECAM>.

[18] *International Color Consortium* [online]. 2013 [cit. 2013-12-30]. Dostupné z: <http://www.color.org/>.

[19] STECÍK, Július *Algoritmy ve správě barev*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 58 s. Vedoucí práce byl Mgr. Pavel Rajmic, Ph.D.

[20] *Qhull* [online]. 1995-2013 [cit. 2013-12-30]. Dostupné z: <http://www.qhull.org/>.

[21] ZIKA, Jakub. *Algoritmus pro 3D Delaunayovu triangulaci.* Brno, 2012. Bakalářská práce. Masarykova Univerzita. Vedoucí práce RNDr. David Sehnal. Dostupné z: <http://is.muni.cz/th/374149/fi_b/>.

[22] Delaunay triangulation. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013 [cit. 2013-12-30]. Dostupné z: <http://en.wikipedia.org/wiki/Delaunay_triangulation>.

[23] Voronoi Diagram / Delaunay Triangulation. CHEW, Paul. *Cornell University: Department of Computer Science* [online]. 1997-2007 [cit. 2013-12-30]. Dostupné z: <http://www.cs.cornell.edu/home/chew/Delaunay.html>.

[24] Convex Hull Algorithms. LAMBERT, Tim. *School of Computer Science and Engineering, The University of New South Wales* [online]. 1998 [cit. 2013-12-30]. Dostupné z: <http://www.cse.unsw.edu.au/~lambert/java/3d/hull.html>.

[25] MATLAB: The Language of Technical Computing. *MathWorks* [online]. 1994-2013 [cit. 2013-12-30]. Dostupné z: <http://www.mathworks.com/products/matlab/>.

[26] A Standard Default Color Space for the Internet – sRGB. In: *World Wide Web Consortium (W3C)* [online]. 1996 [cit. 2014-05-28]. Dostupné z: <http://www.w3.org/Graphics/Color/sRGB>.

[27] ECKHARD, Timo. Plot Lab color coordinates. In: *File Exchange – MATLAB Central* [online]. 2014 [cit. 2014-05-28]. Dostupné z: <http://www.mathworks.com/matlabcentral/fileexchange/44965-plot-lab-color-coordinates>.

[28] Material Exchange Format. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2014 [cit. 2014-05-28]. Dostupné z: <http://en.wikipedia.org/wiki/Material_Exchange_Format>.

[29] MOLINARI, Marc. XML Toolbox. In: *File Exchange – MATLAB Central* [online]. 2003-2005 [cit. 2014-05-28]. Dostupné z: <http://www.mathworks.com/matlabcentral/fileexchange/4278-xml-toolbox>.

[30] Selected Colorimetric Tables. *CIE – International Commission On Illumination* [online]. 2000-2014 [cit. 2014-05-28]. Dostupné z: <http://www.cie.co.at/index.php/LEFTMENUE/index.php?i_ca_id=298>.

[31] BS ISO 13655:2009. *Graphic technology – Spectral measurement and colorimetric computation for graphic arts images.* London: BSI Group, 2010.

[32] D'ERRICO, John. Slice plane through a 3D object? In: *Newsreader – MATLAB Central* [online]. 2001 [cit. 2014-05-28]. Dostupné z: <http://www.mathworks.cn/matlabcentral/newsreader/view_thread/29075>.

[33] VHENGANI, Lufuno. Lines Intersection. In: *File Exchange – MATLAB Central* [online]. 2011 [cit. 2014-05-28]. Dostupné z: <http://www.mathworks.com/matlabcentral/fileexchange/32827-lines-intersection/content/findintersection.m>.

[34] WEISSTEIN, Eric W. Line-Line Intersection. In: *Wolfram MathWorld* [online]. 1999-2014 [cit. 2014-05-28]. Dostupné z: <http://mathworld.wolfram.com/Line-LineIntersection.html>.

[35] ALVAREZ, Robert E. Intersection of line segments using complex variables in Matlab. In: *Aprend Technology* [online]. 2011 [cit. 2014-05-28]. Dostupné z: <http://www.aprendtech.com/blog/Post12/P12segments.html>.

[36] BOURKE, Paul. Minimum Distance between a Point and a Line. In: *Paul Bourke: Research Associate Professor* [online]. 1988 [cit. 2014-05-28]. Dostupné z: <http://paulbourke.net/geometry/pointlineplane/>.

[37] URIBE, Jorge. Understanding black point compensation. *Test Targets 5.0* [online]. 2005, č. 5 [cit. 2014-05-28]. Dostupné z: <http://cias.rit.edu/~gravure/tt/pdf/cm/TT5_Jorge01.pdf>.

[38] ADOBE SYSTEMS INCORPORATED. *Adobe* [online]. 2014 [cit. 2014-05-28]. Dostupné z: <http://www.adobe.com/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

BPC	Black Point Compensation – kompenzace černého bodu
CIE	Commission Internationale de l'Éclairage — Mezinárodní komise pro osvětlování
CIE CAM02	Barevný model, nezávislý na zařízení
CIE LAB	Barevný model, nezávislý na zařízení
CIE LCh	Barevný model, nezávislý na zařízení
CIE xyY	Barevný model, nezávislý na zařízení
CIE XYZ	Barevný model, nezávislý na zařízení
CMM	Color Management Module – modul správy barev
CMYK	Barevný model, závislý na zařízení
D50	Iluminant (bílý bod) dle specifikace CIE
ΔE	Barevná odchylka dle specifikace CIE z r. 1976
HDTV	High-definition Television – televize s vysokým rozlišením
HPMINDE	Hue Planes Minimum Delta E – minimální ΔE v odstínových rovinách, algoritmus přepočtu gamutů
ICC	Internationtal Color Consorcium – sdružení výrobců, kteří spolupracují za účelem správy barev
MATLAB	Matrix Laboratory (maticová laboratoř) – interaktivní programové prostředí a skriptovací programovací jazyk
PCS	Profile Connection Space – propojení profilů
RGB	Barevný model, závislý na zařízení
SCLIP	Algoritmus přepočtu gamutů, ořez po úsečce do středu barevného prostoru
sRGB	Výrobci uznávaný standardní barevný prostor
XML	Extensible Markup Language – rozšiřitelný značkovací jazyk

A PŘÍLOHA – OBSAH PŘILOŽENÉHO CD MÉDIA

Přiložené CD médium obsahuje následující adresáře:

- `Elektronicka_verze_prace` – obsahuje elektronickou verzi práce ve formátu PDF.
- `ICC_profily` – obsahuje ICC profily tiskárny, které odpovídají naměřeným datům v adresáři `namerena_data_tiskarny` u zdrojového kódu. Obsahuje také naměřené profily monitorů, popsané v první části práce.
- `MATLAB_zdrojovy_kod` – obsahuje zdrojový kód pro MATLAB (pro spuštění zkopírovat, včetně podsložek přidat do pracovní cesty MATLABu a nastavit jako pracovní adresář MATLABu). Ve složce je hlavní skript `main.m` a všechny ostatní potřebné funkce, také skript na porovnání obrázků `porovnejObrazky.m`. Podadresář `namerena_data_tiskarny` obsahuje soubory s naměřenými hodnotami odrazivosti (těmto odpovídají ICC profily v adresáři `ICC_profily`. V podadresáři `zdrojove_obrazky` se nachází několik testovacích obrázků včetně toho, na němž byly v práci prezentovány výsledky algoritmů.
- `vysledky` – obsahuje v práci prezentované obrazové výsledky realizovaných algoritmů přepočtů gamutů.