Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Barevná korekce digitálního obrazu

pomocí zásuvného modulu pro Adobe Photoshop

Plzeň, 2007 Oldřich Petřík

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

V Plzni dne 17.5.2007 Oldřich Petřík

Color Correction of a Digital Image

Implemented as an Adobe Photoshop Plug-in

The subject of this thesis is color corrections performed on digital images. Focus is given on clarifying the need for color corrections and explaining some of their most used methods. Further, several color correction tools used in Adobe Photoshop but also in other applications are introduced.

An evaluation of their capabilities results in designing and implementing of a Photoshop plug-in module for advanced color correction. The techniques used for implementing the module are described – the utilization of Adobe Photoshop SDK, the image filtering algorithm and building of the user interface above all. The resulting plug-in is tested on several examples and it's capabilities are appraised.

Obsah

1

Úvod

V této práci se budu zabývat problémem barevných korekcí digitálního obrazu. Nejprve se soustředím na okolnosti, které zapříčiňují potřebu barevných korekcí. Objasním různé druhy digitální reprezentace barev. Následně popíši důvody pro provádění barevných korekcí a přiblížím některé metody, které se k tomu účelu používají. Uvedu nástroje Adobe Photoshopu a zásuvné moduly pro něj určené, které využívají tyto i jiné metody, a zhodnotím jejich schopnosti. Zmíním také několik aplikací barevných korekcí v jiných programech, například v oblasti digitálního videa.

Na základě zhodnocení schopností zmíněných nástrojů se pokusím navrhnout zásuvný modul Photoshopu pro pokročilou korekci barev. Navržený modul pak implementuji a popíši použité postupy. Při tom se soustředím na popis využití Adobe Photoshop SDK, algoritmu filtrace a sestavení uživatelského rozhraní. Na závěr otestuji takto vytvořený zásuvný modul na praktických příkladech a porovnám jeho schopnosti s ostatními uvedenými nástroji.

2

Barvy a jejich korekce

2.1 Vnímání barev

Abychom se mohli zabývat barevnými korekcemi v digitálním světě, je nejdříve potřeba pochopit systém, kterým vnímáme barvy my sami. To, co nazýváme barvou, je ve skutečnosti světlo o specifické vlnové délce nebo taková směs světel různých vlnových délek, na kterou reagujeme stejným způsobem.

Pouť barevné informace začíná na povrchu předmětu, na který se díváme. Světlo ze světelného zdroje, které (obvykle) představuje mnoho různých vlnových délek (složek), dopadá na povrch sledovaného předmětu a od tohoto povrchu se odráží. Nemusí se ovšem odrážet všechny složky dopadajícího světla, ale pouze jejich podmnožina.

Odražené světlo potom putuje do našeho oka. Skrz zornici pronikne pouze tenký svazek paprsků, který je čočkou zaostřen na sítnici. Sítnice obsahuje světločivé buňky – tyčinky a čípky. Většina její plochy je pokryta téměř pouze tyčinkami, které zajišťují vidění v noci a periferní vidění. Výjimku tvoří dvě místa. Jedním z nich je ústí zrakového nervu – slepá skvrna, kde se žádné receptory nenacházejí. Druhým a mnohem důležitějším je tzv. žlutá skvrna, malá jamka v jinak hladké sítnici, která se nachází přesně naproti čočce.

Ve žluté skvrně je velmi vysoká hustota čípků (kolem 150 000 buněk na 1 mm^2). A právě čípky jsou zodpovědné za barevný zrakový vjem. Na lidské sítnici se vyskytují tři druhy těchto buněk, které se liší pouze druhem pigmentu v nich obsaženého. Pro světlo nejnižších vlnových délek jsou čípky s modrým pigmentem, pro vyšší vlnové délky jsou zde receptory se zeleným a pro nejvyšší se zelenožlutým pigmentem. Lidské oko tedy vnímá barvu jako míru vybuzení těchto tří druhů receptorů, jde o tzv. trichromatické vidění. Zde je vidět, že nezáleží na tom, zda do oka dopadá světlo jediné vlnové délky, nebo směs více světel o různých vlnových délkách, protože pokud budou buňky v obou případech drážděny stejnou měrou, uvidíme naprosto stejnou barvu.

Barevná informace následně putuje ve formě elektrických signálů ze světločivých buněk do neuronů na sítnici, kde se provádí prvotní zpracování. Barva je zde převedena na tři signály, kterými jsou informace o světlosti, poloha barvy mezi červenou a zelenou a poloha mezi modrou a žlutou. Takto upravená data pak pokračují do mozku vlákny očního nervu. Po cestě jsou nervová vlákna z vnitřních polovin sítnic obou očí překřížena (vlevo jdou tedy vlákna z obou levých polovin a opačně).

Nervová vlákna končí v týlním laloku mozku. Tady dochází k analýze barevné informace, především na základě kontrastu mezi daty ze sousedních buněk. Mozek si tak vlastně vytváří jakousi mapu postavenou na lokálních kontrastech v příchozích signálech, kterou nakonec „znormalizuje“ na určitý rozsah hodnot. Tímto postupem dosahuje velké přizpůsobivosti jak v rozsahu jasů, tak také v barvě osvětlení. Bílý papír nám pak připadá bílý ať už je osvětlen slunečními paprsky nebo žárovkou, ačkoliv v každém z těchto případů má světlo od něj odražené jiné zastoupení vlnových délek. Tato vlastnost, nazývaná chromatická adaptace, je právě tím, co přináší potřebu barevných korekcí. Důležité je také připomenout, že barevně vidí z celé sítnice jen oblast žluté skvrny, což je jen velmi malá část zorného pole. Mozek sice záměrně rychle kmitá pohledem v určitém rozsahu, aby co nejčastěji obnovil obraz na větší ploše, ale přesto si barvy v periferních částech zorného pole pouze „pamatuje“. Pokud tedy dochází ke chromatické adaptaci, je to především na základě zorného pole snímaného žlutou skvrnou. Více informací o fungování lidského zraku lze najít v [2][1].

2.2 Barevné systémy

2.2.1 Historie

Od chvíle, kdy byl vytvořen první počítač, bylo potřeba nějak zobrazovat informace jím produkované. Zpočátku byla data jen číselného případně znakového charakteru, většinou šlo o výsledky výpočtů, ať vědeckých či vojenských, výstup se tedy omezoval na prostý text, obvykle vytištěný na tiskárně. Postupem času byla k počítači připojena také obrazovka, na které se tato data zobrazovala. Stále se však jednalo víceméně o monochromatický znakový výstup.

Přelom přišel v sedmdesátých letech dvacátého století, kdy se začaly objevovat první osobní počítače a také herní konzole, které přinesly grafický výstup. V roce 1977 představený počítač Apple II zvládal zobrazovat kromě bílé také 6 barev [1]. V následujících letech se počet barev zvyšoval přes 16 u Commodore 64 z roku 1982 až k 256 barvám u počítačů z konce osmdesátých let. Společným znakem těchto barevných režimů byla pevně definovaná tabulka barev, ze které se barvy vybíraly pomocí indexu.

Počátkem devadesátých let minulého století přišly první grafické adaptéry s podporou 16bitových barev, které už nepracovaly přímo s tabulkou barev, ale výsledná barva obrazového bodu byla definována třemi složkami – červenou, zelenou a modrou. Jejich intenzita byla určena 16b číslem tak, že prvních 5 bitů představovalo množství červené barvy, dalších 6 bitů množství zelené a posledních 5 bitů modré. Lidské oko je nejcitlivější na zelenou barvu, proto právě zelené byl přiřazen přebývající bit. V tomto režimu je možné zobrazit až 65 536 různých barevných odstínů, což je ale stále výrazně méně, než kolik dokáže rozeznat lidské oko. Z toho důvodu byly vytvořeny další režimy s 8 nebo 16 bity pro každou barevnou složku (24 resp. 48 bitů na obrazový bod). V poslední době se začínají objevovat také režimy s vysokým dynamickým rozsahem (high dynamic range, HDR), které umožňují zaznamenat v obraze mnohem větší škálu jasů, a podávají tak přirozeněji vypadající výsledky.

2.2.2 RGB

Jedná se o aditivní model, ve kterém se výsledná barva vytváří sečtením červeného, zeleném a modrého (red, green, blue – RGB) světla. Představa je taková, že svítí-li červené a zelené světlo do jednoho místa, složí se a vidíme světlo žluté, taktéž u zeleného a modrého dojde ke složení a to na světlo azurové barvy, nakonec červené a modré vytvoří světlo purpurové. Regulací intenzit jednotlivých složek pak lze dosáhnout libovolného odstínu mezi nimi.

RGB byl vůbec prvním barevným systémem použitým ve světě počítačů. Bylo tomu tak především proto, že barevné obrazovky používají stejný systém, ušetřil se tedy převod z barevné reprezentace v počítači na reprezentaci v monitoru. Pro korektní zobrazení barev je ale nutné pevně definovat vlnové délky všech tří složek. Z toho důvodu byla vytvořena řada standardů, obvykle buď se zaměřením na fyzické možnosti monitorů (sRGB, Apple RGB), na co nejlepší pokrytí viditelných barev (ProPhoto RGB), nebo kompromis obou (Adobe RGB).

Poprvé však systém založený na červené, zelené a modré použili koncem dvacátých let minulého století W. David Wright a John Guild [1]. Zkoumali schopnosti lidského zraku v barevné oblasti za pomoci obrazovky, která byla rozdělena na dvě poloviny. V jedné se zobrazovala vzorová barva, zatímco ve druhé směs červené, zelené a modré, jejichž intenzity měl pozorovatel nastavit tak, aby byl výsledek shodný se vzorovou barvou. Zároveň bylo možné do vzorové barvy přimíchat jednu z těchto tří složek v případě, že se ve druhé polovině nepodařilo najít odpovídající kombinaci. V takových případech byla intenzita u přidané složky považována za zápornou. Výsledkem jejich výzkumu pak byly funkce určující, kolik červeného, zeleného a modrého světla (o vlnových délkách 700, 546,1 a 435,8 nm) je potřeba k simulaci světla o dané vlnové délce. Teprve v roce 1997 pak byl na základě použitých vlnových délek přijat Mezinárodní komisí pro osvětlení (Commission Internationale de l'Éclairage, CIE) [1] standard CIE RGB.

2.2.3 CIE XYZ, CIExy

Vzhledem k tomu, že ve standardu CIE RGB nabývá funkce intenzity červeného světla záporných hodnot, snažili se členové CIE vytvořit takový standard, kde všechny funkce budou nabývat v celém intervalu pouze hodnot kladných. Toho se podařilo dosáhnout vytvořením imaginárních světelných zdrojů z původních tří složek systému RGB. Vznikl tak systém CIE XYZ, kde hodnoty X, Y a Z odpovídají intenzitám složek x, y, z podobně jako je tomu u prostoru RGB, ale funkce jejich intenzity jsou nezáporné.

Protože se barvy v systému XYZ vytvářejí smícháním určitých intenzit tří dílčích složek, tvoří množina těchto barev trojúhelník s každou ze složek samostatně v jednom vrcholu. Z toho důvodu byly složky umístěny do roviny a byly jim přiřazeny normalizované souřadnice $[1; 0]$, $[0; 0]$ a $[0; 1]$. Bod v rovině je ale jednoznačně určen už dvěma souřadnicemi, proto byla poslední (z) vypuštěna. Takto byl vytvořen barevný prostor CIExy [1], v němž lze vyjádřit barvu viditelného světla o libovolné vlnové délce. Nelze v něm už ale vyjádřit intenzitu tohoto světla v důsledku absence třetí souřadnice. Znázornění barevného systému CIExy v rovině xy (tzv. chromatický diagram) se často používá k zobrazení gamutu jiných barevných prostorů. Gamut je množina všech barev, pro které existuje odpovídající kombinace intenzit složek daného barevného prostoru.

2.2.4 CIELAB

Systém CIELAB, či podle jeho složek L*a*b*, vychází z výše zmíněného CIE XYZ. Snaží se ale svým návrhem přiblížit principům lidskému zraku. Jeho složkami jsou proto světlost (lightness, L*) a dva barevné kanály, jeden (a*) udávající polohu barvy mezi zelenou a purpurovou a druhý (b*) pro polohu mezi modrou a žlutou. Lze tedy říci, že pomocí kanálů a*, b* je vyjádřena barva podobně jako pomocí x, y v systému CIExy, kanál L* je pak onou chybějící souřadnicí, která udává intenzitu (světlost) odstínu. Hlavním záměrem bylo vytvoření takového barevného prostoru, ve kterém konstantní změna některého z kanálů vyvolá vizuálně konstantní změnu výsledné barvy. Předchůdcem tohoto barevného prostoru je systém Hunter Lab [1], který aproximuje lidský zrak o něco hůře než CIELAB a je dnes používán jen zřídka.

2.2.5 HSL

Pokud bychom základní složky systému RGB umístili na osy trojrozměrného prostoru, vymezíme krychli odstínů, které lze získat jejich smícháním při různých intenzitách. Krychli můžeme pak „postavit“ na počátek souřadnic, tedy bod, kde jsou všechny tři složky nulové, a zdeformovat na dvojkužel. Tento útvar představuje znázornění barevného modelu HSL. Po obvodu podstavy dvojkužele jsou umístěny čisté barvy od červené přes žlutou, zelenou, azurovou, modrou a purpurovou až zpátky k červené, jde o složku odstínovou (hue, H). Směrem ke středu pak klesá sytost barvy (saturation, S), aby v ose útvaru byla nulová, v řeči barev tedy odstín šedé. Zároveň odspoda nahoru roste světlost (lignтness případně luminance, L).

2.2.6 HSV (HSB)

Jde o velice podobný systém jako HSL, ale zde je ke znázornění barvy použit pouze jeden kužel postavený na vrchol. Lze si to představit tak, že bychom zatlačili vrchol horního kužele u modelu HSL dolů tak, že by se stěny kužele dostaly do roviny. Zde je tedy opět po obvodu podstavy umístěn odstín a vzdálenost od osy kužele určuje sytost. Ve svislém směru se pak mění intenzita barvy (value, V, případně brightness, B) od černé dole až po plnou intenzitu nahoře.

Oba barevné modely – jak HSV, tak HSL – byly navrženy hlavně pro jednodušší práci v barevném prostoru RGB. Pro člověka je mnohem jednodušší představit si, jak bude vypadat výsledná barva při změně odstínu, sytosti nebo světlosti, než při změně intenzity červené, zelené a modré složky.

2.2.7 CMY, CMYK

Barevný model CMY je přesným opakem modelu RGB. Jeho složkami jsou azurová (cyan, C), purpurová (magenta, M) a žlutá (yellow, Y), které jsou opačnými barvami k červené, zelené a modré. Tento systém vznikl z potřeby barevného tisku, kde nelze aplikovat model aditivní (tedy takový, kde se složky sčítají), ale je potřeba model subtraktivní. Na začátku je bílý papír, který odráží všechny vlnové délky stejně. Aplikací pigmentu jedné ze složek se zamezí odrážení vlnových délek odpovídajícím jejímu protějšku v systému RGB (jeho

barva se „odečte“). Aplikací pigmentu další ze složek se přestane odrážet další část spektra a nakonec při všech složkách vytištěných přes sebe se neodráží žádné světlo, vidíme černou. Problémem je, že nelze vyrobit úplně dokonalé pigmenty, zejména správný azurový je obtížné získat. Také nelze při tisku tekutými inkousty nanést na papír příliš velké množství inkoustu, neboť by se rozpíjel a papír by se zvlnil nebo i protrhl.

Jako řešení zmíněných problémů byl ke zmíněným třem pigmentům přidán ještě jeden – černý (black případně key, K). Při soutisku všech tří pigmentů se díky jejich nedokonalosti nedařilo vytvořit přesnou černou barvu, což se použitím černého inkoustu vyřešilo. Zároveň se přidáním černé barvy omezila nutnost použití velkého množství ostatních inkoustů naráz. Přidáním černé složky ke zbylým třem byl tedy vytvořen barevný model CMYK, který je dnes používán ve většině barevných tiskáren.

Velkou nevýhodou těchto systémů je právě to, že jsou opačné k systému RGB. Barvy jimi produkované leží v trojúhelníku, stejně jako barvy produkované systémem RGB. Tyto dva trojúhelníky jsou ale navzájem o 180° otočené, což způsobuje problémy při převodu mezi oběma systémy, protože všechny odstíny, které leží mimo průnik obou množin, jsou „ořezány“ – nahrazeny nejbližší barvou z tohoto průniku.

2.2.8 Další barevné systémy

Existuje řada dalších, více či méně používaných, barevných modelů, které ale většinou staví na stejných principech jako ty výše zmíněné. Jedním z nich je například Munsellův barevný model [1] vytvořený Albertem H. Munsellem počátkem minulého století, který se velice podobá modelu HSL. Jako další bych mohl jmenovat zejména ve výtvarném umění využívaný barevný model **RYB** stavějící na míchání tzv. základních barev – červené, žluté a modré. Systému L*a*b* se velmi podobá v televizním vysílání používaný model **YUV** a z něj odvozený **YCbCr** používaný pro digitální video.

2.3 Barevné korekce

Pod pojmem barevná korekce obrazu se rozumí změna barevnosti obrazu. Ve většině případů by se dalo říci oprava barevnosti ve snaze o co nejpřirozenější barevné podání. Někdy se ale termínem barevná korekce označuje i jakákoliv jiná změna barevnosti, kdy výsledné barvy nemusí odpovídat skutečnosti. Pak to nemusí být oprava vzhledem k přirozeným barvám, ale například k fantazii autora. Termín přirozené barvy nebo barevné podání se nyní pokusím blíže specifikovat.

Jak jsem popsal výše, lidský zrakový aparát je schopný se adaptovat na zabarvení scény v zorném poli, takže vnímá přibližně konstantní barvu předmětů bez ohledu na barvu světla, kterým je scéna osvětlena. To ale nefunguje u fotoaparátů, kamer a dalších zařízení na snímání obrazu, které zachycují barvu předmětů takovou, jaká je dána světlem odraženým od jejich povrchu. A tato barva závisí samozřejmě také na složení světla, které na daný předmět dopadá. Mají-li v dopadajícím světle některé vlnové délky slabší intenzitu než ostatní, nemohou se tyto vyskytnout v plné míře ani v odraženém světle. Mozek si postiženou část spektra dokáže „zesílit“, snímač fotoaparátu nebo filmové políčko však nikoliv. Jestliže se potom díváme na fotografii, která nebyla nijak upravena, mají předměty na ní takovou barvu, jaké světlo se od nich odrazilo, ale ne takovou, jakou bychom

„viděli na vlastní oči“. Jedinou výjimku tvoří případ, kdy by fotografovaná scéna byla osvětlena světlem s rovnoměrným zastoupením vlnových délek, jinak řečeno bílým. V takovém případě „vidí“ fotoaparát stejné barvy jako člověk a fotografie má tedy pro člověka přirozené barevné podání. V ostatních případech je nutné barvy opravit, aby se dosáhlo stejného efektu.

Digitální fotoaparáty za tím účelem obsahují funkci zvanou vyvážení bílé, která se právě snaží „zneutralizovat“ zabarvení snímku. Toho se dosahuje různými algoritmy, ovšem vždy jde jen o nedokonalou aproximaci lidského vnímání. V závislosti na kvalitě algoritmu v konkrétním přístroji se proto vždy vyskytne určité procento snímků, kde byly barvy vyhodnoceny jinak, než by to udělal náš mozek. I zde je potom nutné použít barevnou korekci.

Další potíže pak přináší normalizace, kterou mozek provádí nad informacemi z očí. Pokud se ve scéně nachází předmět s barvou obsahující složku, která jinak není ve scéně zastoupena, připadne nám jeho barva sytější, než ve skutečnosti je. Jako příklad mohu uvést zámek Červená Lhota, jenž je jediným červeným objektem ve svém okolí. Při pohledu na něj se nám zdá poměrně sytě červený, ale ve skutečnosti je sytost omítky mnohem nižší. Ke stejnému efektu často dochází u oblohy, která se nám zdá sytě modrá, ale na fotografii pak vypadá bledě. V těchto případech je také možné využít barevných korekcí pro dosažení stejného efektu, jaký bychom viděli na vlastní oči.

2.4 Metody barevných korekcí

Než přistoupím k vlastnímu popisu jednotlivých metod, chtěl bych přiblížit reprezentaci barevného obrazu v počítači. Protože počítač pracuje na digitálním tj. číselném principu, je velice obtížné použití spojité reprezentace. Celý obraz je proto při převodu do digitální podoby nutné rozdělit (vzorkovat) do bodů, tzv. pixelů. Každému pixelu je pak podle charakteru obrazu přiřazena buď intenzita (černobílý obraz) nebo barevná informace (barevný obraz). Barva, tak jak ji vidíme v reálném světě, je směsí světel o různé vlnové délce a různé intenzitě. Pokud bychom chtěli popisovat barvy v tomto stavu, museli bychom pro každý obrazový bod vytvořit spojitou křivku popisující intenzity světla v rozsahu viditelných vlnových délek. Zde opět narážíme na problém se spojitými daty, který ale nemůžeme dostatečně efektivně obejít dalším vzorkováním, neboť by takto reprezentovaný obraz zabíral příliš mnoho paměti. Řešení je ale jednoduché a využívá další vlastnosti našeho zraku a sice té, že barvu tvořenou směsí světla různých vlnových délek lze nahradit směsí pouze několika málo dostatečně vzdálených vlnových délek (barev) s určitou intenzitou. Díky tomu byl vytvořen systém RGB, který je popsán výše. Barevný obraz je v něm složen z dílčích obrazů – kanálů, z nichž každý je vlastně mapou intenzit pro danou složku systému. Postupem času se pak, většinou pro přirozenější práci s barevnou informací, objevily i další způsoby reprezentace, jako jsou systémy L*a*b* nebo HSL. V těchto systémech, zejména v L*a*b*, je možné provádět barevné korekce s lepšími výsledky než v systému RGB.

2.4.1 Změna intenzity barevných kanálů

Nejjednodušší a nejméně přesná metoda barevné korekce. Výsledek je víceméně závislý pouze na vjemu člověka, který korekci provádí. Využívá se zde úpravy intenzit jednotlivých barevných kanálů. Pokud má například obraz nádech do červena, snížením intenzity červeného kanálu (nebo zvýšením intenzity modrého a zeleného) dosáhneme zlepšení barevnosti. Tato metoda nemusí fungovat vždy přesvědčivě a může dojít např. k nevhodnému zabarvení bílých oblastí.

2.4.2 Úprava křivek barevných kanálů

Pro každý kanál je výsledná intenzita definována funkcí původní intenzity. Při použití této metody obvykle uživatel zadává body, kterými definuje hodnotu funkce pro konkrétní vstupní intenzity kanálů. Tyto body jsou následně proloženy funkcí. Ve většině případů se používá interpolace kubickou křivkou, kdy se každé dva body spojí polynomiální funkcí nejvýše třetího řádu (kubickou funkcí) takovou, že se v každém bodě vyjma krajních rovnají tečny (derivace) obou funkcí, které na sebe v tomto bodě navazují. Je ale možné použít také interpolační polynom nebo lomenou čáru (lineární křivku). Stejně jako u předchozí metody je i zde výstup závislý na člověku, který obraz upravuje.

2.4.3 Referenční barvy

Velmi oblíbená metoda, která využívá oblastí v obrázku, u kterých známe jejich skutečnou barvu. Tyto barvy pak nazýváme referenční. Korekce spočívá v nalezení a označení takových oblastí v obrázku. Následně je třeba těmto oblastem přiřadit jejich referenční barvu. Barvy v obrázku jsou pak přepočítány na základě rozdílů mezi původními barvami a jejich požadovanou podobou. Přepočet může využívat různé interpolační mechanismy, třeba již zmíněné kubické křivky, rozdíl oproti předchozí metodě pak spočívá vlastně jen ve způsobu definice bodů na křivkách. Jako referenční barvy jsou nejčastěji používány barvy neutrální, tedy šedé, protože u nich známe dobře poměry intenzit kanálů a korekce tedy bývá většinou poměrně přesná. Někdy se používají také pleťové odstíny nebo barvy zeleně, ovšem zde už nemusíme dosáhnout takové přesnosti.

2.4.4 Míchání kanálů

V tomto případě jsou výsledné barevné kanály tvořeny kombinací vstupních kanálů. Každý kanál je tedy součtem všech vstupních kanálů, přičemž každý z nich je vynásoben zadaným koeficientem. Tyto koeficienty nemusí být omezeny jen na rozsah 0–1, ale mohou být i větší a také záporné. Metoda zahrnuje veškeré úpravy, které je možné provést pomocí změny intenzity, ale její schopnosti jsou větší.

3

Nástroje pro korekci barev

3.1 Korekce barev v Adobe Photoshopu

Photoshop je sám o sobě velice silný nástroj pro úpravu obrazu, barevné korekce nevyjímaje. Proto nejprve uvedu nástroje nebo postupy, kterými lze provádět úpravu barev v samotném Photoshopu. Potom se zaměřím na dodatečné zásuvné moduly pro Photoshop, které jsou v tomto oboru dostupné.

3.1.1 Interní nástroje

Míchání kanálů

Jednoduchý nástroj, který pracuje podle stejnojmenné korekční metody. Pro každý kanál je možné měnit vlivy vstupních kanálů v rozmezí -200% až +200%, navíc lze k výsledné intenzitě přičíst konstantní hodnotu ve stejném rozmezí (Obr. 3.1a).

Vyvážení barev

Tato funkce je založena na principu změny intenzity barevných kanálů, zmíněnou metodu ale rozšiřuje o rozdělení změn podle světlosti zpracovávaných pixelů. Lze tedy nastavit změny zvlášť pro stíny, střední tóny a světla (Obr. 3.1b). Je také možné zachování původní světlosti jednotlivých bodů, i když se změní jejich barevný odstín.

Křivky

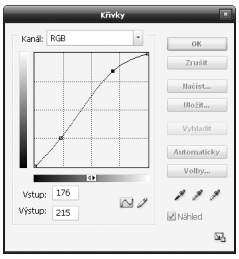
Křivky jsou nejsilnějším nástrojem na barevné korekce, kterým Photoshop disponuje. Jejich funkce je podle očekávání shodná s metodou úpravy křivek. Pro jednotlivé kanály (ale i pro všechny najednou) je možné definovat transformační křivku, a to buď pomocí bodů, které budou proloženy kubickou interpolační křivkou, nebo volným kreslením. Tato křivka pak pro každou vstupní intenzitu daného kanálu určuje intenzitu výslednou.

Nástroj křivky obsahuje také tlačítko pro automatickou korekci. Její funkci lze přepínat v dialogu (Obr. 3.1c) pod tlačítkem Volby mezi třemi módy, které odpovídají funkcím

Kontrast automaticky, Úrovně automaticky a Barvy automaticky.

(a) Míchání kanálů

(b) Vyvážení barev



(c) Křivky

Obr. 3.1: dialogy nástrojů standardně dodávaných v Adobe Photoshopu

ostatní

Poměrně specifickým, ale celkem silným, nástrojem pro barevné korekce je dialog Použít obraz, který umožňuje do obrazu vložit samostatný kanál nebo i celý obraz z libovolného otevřeného souboru (tedy například stejného obrázku, ale převedeného do jiného barevného prostoru). V dialogu lze nastavit, s jakým efektem prolnutí a jakou průhledností se obrazová data vloží.

Dále Photoshop nabízí několik automatických korekčních funkcí, které jsem již zmínil u nástroje Křivky. A snad kromě funkce Kontrast automaticky je lepší k nim také přes Křivky přistupovat, neboť je pak možné jejich funkci ovlivnit nebo opravit změnou patřičných křivek.

V poslední řadě bych uvedl nástroj Odstín a sytost, který je sice pro samotnou korekci barev velmi slabý, ale lze jím, v kombinaci s funkcí Kontrast automaticky nebo Jas a kontrast, upravit celkovou barevnost obrázku po použití jiných korekcí.

3.1.2 Zásuvné moduly

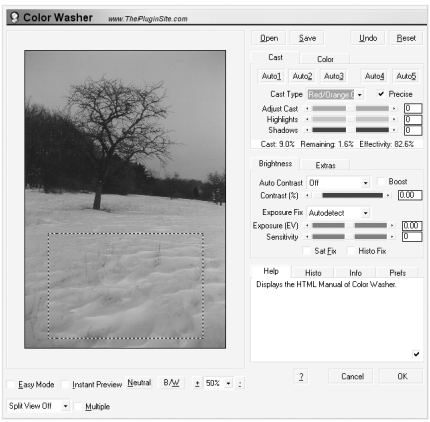
Color Pilot

Prvním modulem plug-in pro Adobe Photoshop, na který narazíte, pokud zadáte do vyhledávače termín „color correction“, bude pravděpodobně Color Pilot (Obr. 3.2) od společnosti Two Pilots. Tento nástroj, pracuje na principu referenčních barev. Pomocí obdélníčků vymezíte oblasti a přiřadíte jim referenční barvu. Modul se pak postará o to, aby ve výsledném obrázku měly tyto oblasti požadovanou barvu. Všechny ostatní barvy v souladu s tímto samozřejmě přepočítá a tím dojde k barevné korekci. Pro každou nadefinovanou barvu lze dále nastavit, zda se bude upravovat pouze odstín, nebo i světlost. Výsledky tohoto pluginu jsou dobré, označil bych ho za jeden z lepších.

ColorWasher

Druhým v pořadí je modul ColorWasher (Obr. 3.3), který pochází ze stránky ThePluginSite.com. Je založen na stejném principu jako předchozí Color Pilot. Zde se ovšem

Obr. 3.2: dialog modulu Color Pilot



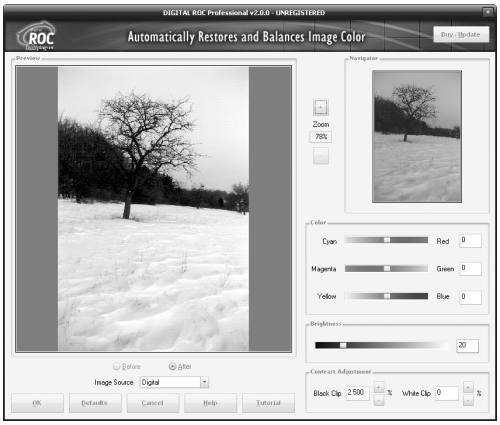
Obr. 3.3: dialog modulu ColorWasher

předpokládají pouze oblasti neutrálních barev (šedé), ačkoliv se do jisté míry dají nadefinovat i jiné barvy. Plusem tohoto modulu je možnost vybrat nádech, který měl obrázek před korekcí (i když jen z několika možností), což má následně vliv na to, jakým způsobem jsou interpolovány barvy v obrázku. Dále si uživatel může nastavit i automatickou korekci kontrastu a to buď v procentech, nebo výběrem z připravených předvoleb.

Color Mechanic

Dále jsem se zabýval pluginem Color Mechanic (Obr. 3.4). Je to velice jednoduchý nástroj, který se ale příliš nehodí na barevné korekce jako takové. Jeho síla spočívá spíše v záměně barev v obrázku. Tohoto efektu dosahuje použitím šestiúhelníkového barevného diagramu, ve kterém je možné zvolit zdrojový odstín (bod v šestiúhelníku) a následně vektor, jehož koncový bod udává odstín cílový, kterému lze pomocí posuvníků ještě upravit světlost (ta v diagramu chybí) a míru změny. Bohužel ale modul vybírá barvy až příliš ostře, takže i mírný šum ve zdrojovém obrázku se může ve výsledku silně zvýraznit.

Obr. 3.4: dialog modulu Color Mechanic



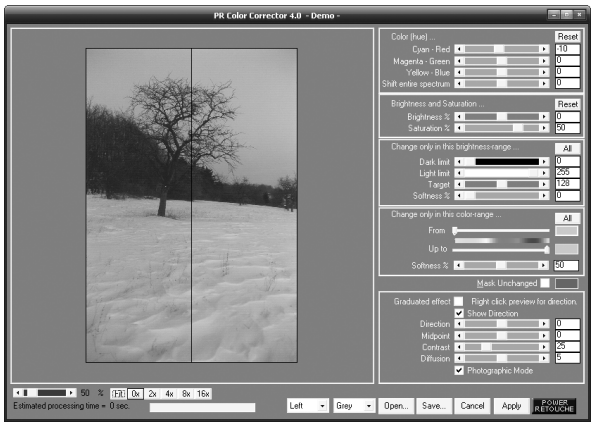
Obr. 3.5: dialog modulu Kodak Digital ROC

Digital ROC

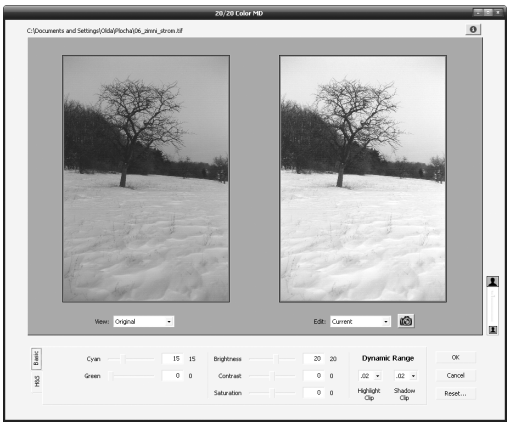
Společnost Kodak nabízí nástroj Digital ROC (Obr. 3.5), který provádí barevnou korekci plně automaticky. Ve verzi Professional lze navíc dodatečně upravit vyvážení barev a jas obrázku. Nepřišel jsem na to, jakým způsobem tento plugin barevné korekce provádí, nicméně dosahuje překvapivě dobrých výsledků (vzhledem k tomu, že pracuje automaticky).

PowerRetouche

Ze sady PowerRetouche lze zmínit hned několik zásuvných modulů. Za všechny je to modul Color Corrector (Obr. 3.6), který nabízí odstranění barevného nádechu. Princip je velice jednoduchý – uživatel pomocí posuvníků mění poměr červené/azurové, zelené/purpurové a modré/žluté barvy, případně všech najednou. Je tedy založen na změně intenzity barevných kanálů. Dále je možné upravit jas a sytost výsledného obrázku (kontrast chybí). Změny lze omezit pouze na určitý rozsah jasů nebo barev v obrázku a také jen na určité místo obrázku s odezněním do zvoleného směru (jako při použití masky s lineárním přechodem). Velice příjemná je možnost zobrazit vedle sebe upravenou a neupravenou polovinu obrázku pro srovnání. Sada PowerRetouche obsahuje ještě další nástroje pro barevné korekce, jako například White Balance, který pracuje na principu referenčních barev s omezením pouze na šedé tóny, a dále jednoduché nástroje suplující standardní výbavu Photoshopu – Contrast, Brightness, Saturation, Histogram Repair, Exposure a Dynamic Range Compressor.



Obr. 3.6: dialog modulu Color Corrector



Obr. 3.7: dialog modulu 20/20 Color MD

20/20 Color MD

Na podobném principu jako výše zmíněný Color Corrector pracuje i plugin 20/20 Color MD od Photo Tune Software (Obr. 3.7). Jeho pojetí připomíná jakéhosi průvodce, který v několika krocích provede uživatele korekcí obrázku. Postupně za sebou následuje vyrovnání dynamického rozsahu, oprava expozice, posun barev mezi červenou a azurovou, mezi zelenou a purpurovou, mezi modrou a žlutou a na závěr je možné nastavené parametry ještě manuálně doladit. Schopnosti tohoto nástroje jsou, stejně jako u Color Correctoru, vcelku slabé, pro začátečníky ale mohou být dostačující.

EditLab Pro

Posledním ze zásuvných modulů, kterými jsem se zabýval, a nejspíše nejkomplexnějším z nich je plugin společnosti iCorrect s názvem EditLab Pro. Tento nástroj provádí barevnou korekci ve čtyřech nezávislých fázích, kterým jsou přiděleny jednotlivé záložky v uživatelském rozhraní (Obr. 3.8). První fáze umožňuje vyrovnat barvy v obrázku pomocí definice referenčních barev, případně s volbou Silders také změnou intenzity barevných kanálů. Na druhé záložce pak lze provést vyrovnání dynamického rozsahu pomocí histogramu, třetí záložka se pak zabývá úpravou jasu, kontrastu a sytosti. A konečně v poslední fázi je možné na barevném prstenci posouvat odstíny barev. Na každé záložce navíc existuje tlačítko SmartColor, které danou proceduru provede automaticky, ovšem ne vždy s uspokojivým výsledkem (hlavně v případě vyrovnání barev). Trochu nepříjemně působí absence jakékoliv možnosti přiblížení náhledu, stejně jako nemožnost zvětšit okno pluginu (v nastavení je ale skryta volba pro roztažení na celou obrazovku).

Obr. 3.8: dialog modulu EditLab Pro

3.2 Korekce barev mimo Photoshop

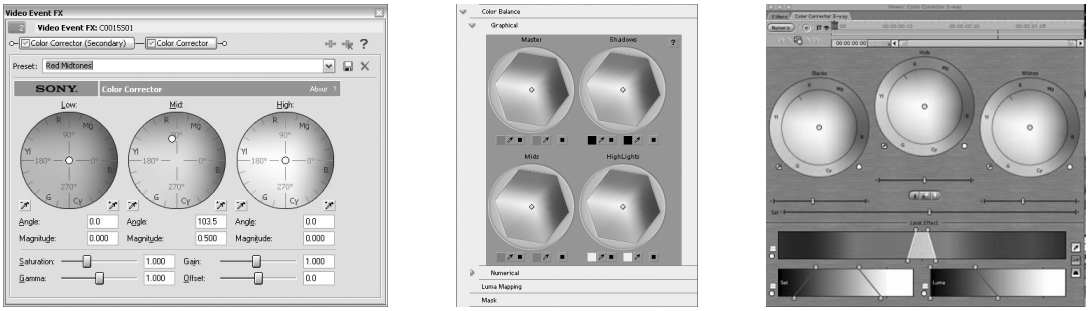
3.2.1 Samostatné programy

Tato kategorie se v zásadě příliš neliší od zásuvných modulů pro Photoshop, protože většina z nich je dostupná také jako samostatný program. Snad jediný, který jsem našel a není ve formě zásuvného modulu, je program KlearVision Photo-D. Ten zpracovává obrázky automaticky v dávkách a podle údajů na svých webových stránkách provádí snad všechny myslitelné úpravy obrázku od odstranění barevného nádechu přes vyrovnání barevnosti, jasu a kontrastu až po zvětšení/zmenšení obrázků na požadované rozměry a jejich doostření. Program bohužel není k dispozici pro vyzkoušení, o jeho schopnostech tedy nemohu ze své strany říci nic.

3.2.2 Digitální video

U videa nastává trochu jiná situace, než u fotografie. Zatímco ve fotografii se obvykle snažíme barvy co nejvíce přiblížit realitě, pro video to nemusí vždy platit a někdy je

barevný nádech na místě, neboť dotváří atmosféru záběru. Také se zde častěji najde využití pro nahrazení některé barvy jinou. Programy pro střih videa proto obsahují trochu odlišné nástroje pro korekci barev, které jsou podobné výše uvedenému pluginu Color Mechanic. Zde ale bývá nastavení rozděleno zvlášť pro stíny, střední tóny a světla a někdy je ještě k dispozici celková úprava (Obr. 3.9). Tento postup bývá implementován pomocí barevných kruhů a posouvání bodů na nich (tak jako u již zmíněného Color Mechanicu, nebo ve formě, kterou najdeme v modulu EditLab), případně pomocí škály barev v jedné ose a intenzity nebo světlosti barvy v ose druhé. Výsledný efekt může být navíc omezen pouze na určitý rozsah barev nebo jasů. Samozřejmě jsou tu i jednodušší nástroje jako vyrovnání histogramu, gamma korekce a podobně.



(a) Sony Vegas (b) Adobe Premiere (c) Apple Final Cut

Obr. 3.9: korekce barev v programech pro střih digitálního videa

4

Plugin pro pokročilou korekci barev

4.1 Úvod do zásuvných modulů

Adobe Photoshop je možné rozšířit pomocí zásuvných modulů (modulů plug-in, „pluginů“). Takový zásuvný modul je vlastně běžná dynamická knihovna (DLL), která ale navíc obsahuje tzv. PiPL (Plug-in Property List). PiPL je vlastně blok informací, ze kterých Photoshop zjistí, o jaký typ pluginu se jedná, jak se jmenuje, do které nabídky má být zařazen, jaké barevné režimy podporuje a spoustu dalších.

Existuje devět typů zásuvných modulů, které mohou rozšiřovat možnosti Photoshopu o podporu nových souborových formátů, vstupních i výstupních zařízení, nové možnosti výběru oblasti v obrázku, výběru barvy z palety, přidávají možnosti automatizace nebo propojení s jinými aplikacemi a také přinášejí možnosti úprav samotných obrazových dat otevřeného souboru. Právě posledním jmenovaným typem pluginů, tedy filtry, se budu zabývat blíže, neboť do této kategorie spadá i plugin pro barevnou korekci. Podrobný popis jednotlivých typů zásuvných modulů ve Photoshopu je dostupný v [3].

4.2 Analýza

4.2.1 Co chybí na trhu?

To je první otázka, kterou je třeba se zabývat. Nemá smysl vytvářet něco, co už existuje, mnohem lepší je najít „díru“ a tu zaplnit. Ovšem pokud se zaměřím pouze na pokročilou korekci barev, neexistuje žádný plugin, který by do této kategorie spadal, nebo jsem alespoň žádný takový nenašel. Téměř všechna dostupná řešení neposkytují takřka žádnou funkčnost navíc oproti standardním nástrojům Photoshopu, v lepším případě nabízejí pohodlnější ovládání nebo určitou míru automatizace. Tato „usnadnění“ mohou být ale pro zkušenějšího uživatele spíše na škodu, neboť ztrácí určitý přehled nad tím, co se s obrazovými daty děje (jak vypadají korekční křivky, v jakém poměru se míchají kanály, ...). Odpověď na tuto otázku by tedy mohla znít: „Na trhu chybí modul, který nabízí v oblasti barevné korekce širší (pokročilejší) funkce než sám Photoshop a ponechává uživateli nad korekcemi plnou kontrolu.“ S tím souvisí další otázka a sice...

4.2.2 Co chybí ve Photoshopu?

Nástroje Photoshopu nabízejí pro barevné korekce silnou základnu. Mnohdy je ale nutné použít několik nástrojů najednou, což činí proces zdlouhavým a obtížněji reprodukovatelným. Hodila by se tedy určitá forma integrace nejpoužívanějších nástrojů pro tuto oblast do jediného. Dále neumožňuje Photoshop při práci s obrazem až na výjimky použít jiné barevné kanály než ty přítomné v obrázku. Existují způsoby, jak toho dosáhnout lze, ale opět jde o velice zdlouhavý proces. To jsou tedy dva zřejmě nejdůležitější nedostatky, které by bylo dobré pokrýt.

4.2.3 Požadavky na plugin

Požadavky již byly víceméně vzneseny, přesto neuškodí je trochu více rozvést. Má-li být modul spojením více nástrojů, které to budou? V první řadě jednoznačně Křivky, jakožto nejsilnější nástroj na barevné korekce ve Photoshopu. Dále bych přidal Vyvážení barev a potažmo Míchání barev. Zajímavé, ale pravděpodobně obtížné by bylo alespoň částečné začlenění nástroje Použít obraz.

V otázce vstupu více barevných kanálů, než obrázek sám obsahuje, budou jistě figurovat kanály barevných prostorů RGB, CMYK, L*a*b* a HSB. Co se týče uživatelského rozhraní, bylo by dobré v rámci intuitivního ovládání zachovat podobnost s nástroji Photoshopu.

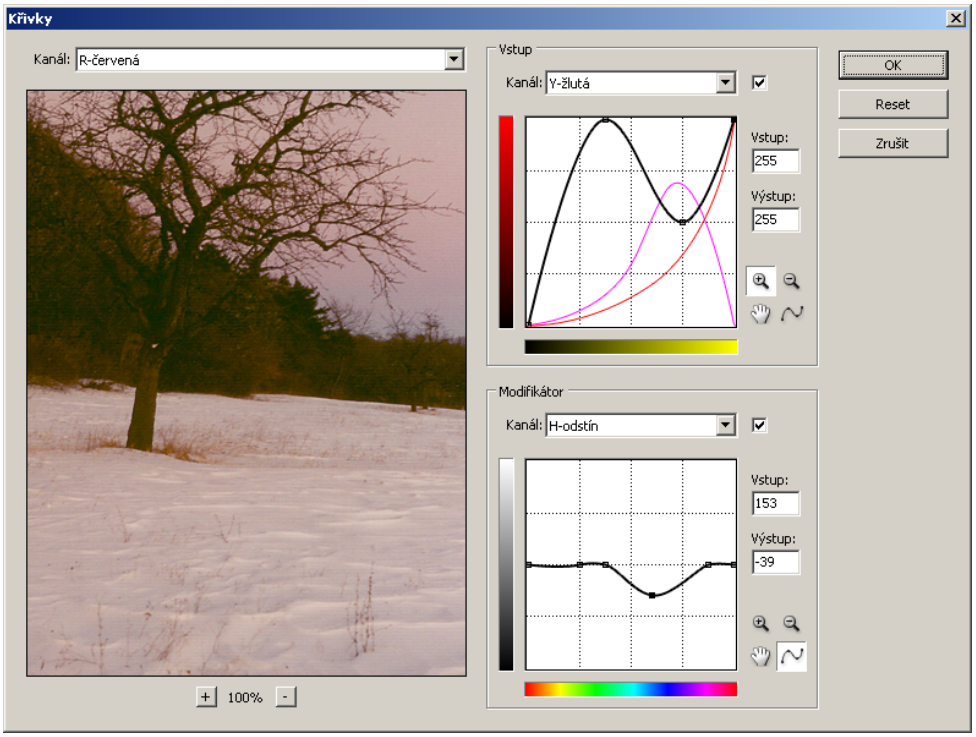
4.3 Návrh modulu

V návrhu se zaměřím rovnou na uživatelské rozhraní, ze kterého následně vyplyne i fungování modulu. Jak již bylo řečeno, plugin bude stavět hlavně na nástroji Křivky, proto bych použil pole s křivkou jako základní ovládací prvek. Zde ale bude pole obsahovat křivek víc, přičemž uživatel si bude moci zvolit kanály, jejichž křivky chce v obraze kombinovat a tedy je i zobrazit. V nabízených kanálech budou zahrnuty všechny ze zmíněných barevných prostorů. Stejně jako v modulu Křivky budou i zde oddělená nastavení pro jednotlivé kanály obrázku, pouze nastavení pro všechny kanály dohromady vynechám. Pro každý bod obrázku se tedy zjistí hodnoty potřebných barevných kanálů, pro jiné barevné prostory, než v jakém obrázek leží, se hodnoty dopočítají. Následně se najdou pro tyto hodnoty jejich obrazy na příslušných křivkách a tyto obrazy se pak sečtou do výsledné hodnoty dané barevné složky.

Zpracování obrazu tím ale ještě nekončí. Modul bude obsahovat ještě druhé pole s křivkami, které bude fungovat jako modifikátor výstupu z předchozí části. Modifikátor bude fungovat podobně jako způsob prolnutí vrstev „překrýt“. Pro tyto účely bude svislá stupnice druhého pole posunuta, nula bude uprostřed a nikoli na spodním okraji. Výpočet hodnoty bude shodný s první částí, pouze bude výsledek umístěn na změněné stupnici. Výstup druhé části bude modifikovat výstup první části tím způsobem, že pokud bude nižší než nula, hodnota z první části se sníží, pokud bude naopak větší než nula, hodnota se zvýší. Pro nulu se pak výsledek první části nemění. Takto upravená hodnota je nakonec uložena do dané barevné složky obrazového bodu.

Obě pole s křivkami bude možné přiblížit pro větší přesnost při manipulaci s body. Kromě

vstupního a modifikačního pole křivek musí modul obsahovat ještě náhled změněného obrázku, který bude taktéž možné přibližovat a oddalovat. Výsledný návrh vzhledu je vidět na Obr. 4.1.



Obr. 4.1: Návrh vzhledu dialogu zásuvného modulu

4.4 Použité technologie

4.4.1 Programovací jazyk

Sada Adobe Photoshop SDK (Software Development Kit) je v současnosti dostupná pouze pro jazyk C++, plugin je tedy nutné psát v tomto jazyce. Jako vývojové prostředí jsem zvolil Microsoft Visual Studio, neboť většina dostupných zdrojů informací, ze kterých jsem čerpal, počítá právě s tímto prostředím.

4.4.2 Photoshop API

V souladu s aplikačním rozhraním (API, Application Interface) má zásuvný modul pro Photoshop jednu vstupní metodu, která musí být vidět i zvenčí knihovny. Pokud uživatel plugin použije (vybere ho z nabídky, spustí naposledy použitý filtr nebo přehraje makro), Photoshop načte knihovnu do paměti a zavolá právě tuto její metodu. Zároveň jí předá čtyři parametry. Prvním z nich je tzv. selektor, ten určuje, jaká fáze běhu pluginu má být vykonána (příprava, filtrování obrazu s případným zobrazením dialogu, zobrazení dialogu s informacemi o pluginu apod.). Druhý parametr obsahuje adresu v paměti, na které se nachází struktura zprostředkovávající rozhraní mezi Photoshopem a pluginem, v případě

filtrů se jedná o datový typ `FilterRecord` [3]. Skrze atributy této struktury je možné načíst a uložit obrazová data, zjistit informace o obrázku, ale třeba také přistupovat ke speciálním sadám pokročilých funkcí (tzv. suites). Třetím parametrem je paměťová adresa bloku dat pluginu. Do tohoto bloku si může modul ukládat data, která bude potřebovat při dalších spuštěních. Poslední parametr je odkaz na paměťové místo, do kterého musí plugin uložit svůj výstupní stav. Výstupní stav je číslo, které může být buď nula (konstanta `noErr`), pokud vše proběhlo v pořádku, kladné v případě, že se vyskytla chyba, ale modul již její výskyt uživateli oznámil, anebo záporné tehdy, kdy došlo k chybě a chybové hlášení má zobrazit sám Photoshop.

Navržený zásuvný modul bude potřebovat data z obrázku nejen v jeho původním barevném prostoru, ale i v dalších. Proto bude potřeba funkce, která zajistí převod mezi těmito barevnými prostory. Vzhledem k tomu, že u některých barevných systémů Photoshop podporuje několik různých standardů, přičemž Photoshop API nenabízí žádnou možnost, jak tyto standardy rozeznat, bude mnohem výhodnější použít pro převod interní funkce Photoshopu. Tyto berou samozřejmě aktuální standard v potaz, takže zamezíme nepřesnostem v barevnosti, ke kterým by mohlo dojít použitím vlastních metod. Převody barev mezi jednotlivými prostory a tři další operace týkající se barev zajišťuje funkce `colorServices`, která je pluginu dostupná ve struktuře `FilterRecord`. Jedná se o referenci funkčního typu `ColorServicesProc`. Volání funkce vypadá následovně:

```
OSerr (*ColorServicesProc)(ColorServicesInfo *csinfo)
```

Jako parametr přebírá metoda odkaz na strukturu `ColorServicesInfo`. V položkách této struktury jsou umístěny informace nutné pro vykonání metody a po jejím dokončení jsou zde také uloženy výsledky. Následující výčet shrnuje popis jednotlivých položek struktury.

- `int32 infoSize`
  Udává velikost celé struktury `ColorServicesInfo`. Tato položka je zde z důvodu kompatibility s případnými vyššími verzemi Photoshopu, kde by struktura mohla obsahovat některé nové položky a její velikost by se tím zvětšila.

- `int16 selector`
  Na základě jeho hodnoty se vykoná jedna ze čtyř možných operací, které tato metoda poskytuje. Může nabývat následujících hodnot:

  - `plugIncolorServicesChooseColor`
    Volání funkce s touto hodnotou způsobí zobrazení dialogu pro výběr barvy. Lze použít jak interní dialog Photoshopu, tak standardní dialog operačního systému a dokonce i jiný, který byl nainstalován ve formě pluginu. Barvu vybranou v dialogu je možné získat buď v barevném prostoru, ve kterém byla vybírána (a zároveň zjistit, o který prostor se jedná) nebo převedenou do libovolného zadaného.
  - `plugIncolorServicesConvertColor`
    Tato hodnota vyvolá převod zadané barvy z jednoho barevného prostoru do druhého. Je to pravděpodobně nejpoužívanější ze všech čtyř možností.
  - `plugIncolorServicesSamplePoint`
    Umožňuje získání barvy obrázku na zadaném bodě. Tímto způsobem může

plugin jednorázově zjistit barvu libovolného pixelu v obrázku a zároveň ji převést do požadovaného barevného modelu. To je vhodné například v reakci na stisknutí tlačítka myši, pro načtení bloku obrazových dat ale existují vhodnější postupy.

  - `plugIncolorServicesGetSpecialColor`
    Pomocí této hodnoty je možné zjistit aktuálně nastavené barvy popředí a pozadí opět s možností převodu do jiného barevného prostoru. U zásuvných modulů typu filtr je možné obě barvy zjistit přímo ze struktury `FilterRecord`, ale například u modulů typu export to možné není a je třeba využít této funkce.

- `int16 sourceSpace`
  Obsahuje identifikátor zdrojového barevného prostoru. V případě výběru barvy z palety určuje barevný prostor výchozí barvy. Při konverzi barvy mezi dvěma barevnými prostory určuje ten prostor, ze kterého se bude převádět. Pro zbylé dvě operace udává barevný prostor, ve kterém bude uložena výsledná barva. Je možné nastavit kteroukoliv z následujících hodnot: `plugIncolorServicesRGBSpace`, `plugIncolorServicesHSBSpace`, `plugIncolorServicesCMYKSpace`, `plugIncolorServicesLabSpace`, `plugIncolorServicesGraySpace`, `plugIncolorServicesHSLSpace`, `plugIncolorServicesXYZSpace`. Pokud bychom chtěli automaticky zjistit hodnotu odpovídající barevnému prostoru obrázku, lze použít funkci `CSModeToSpace`.

- `int16 resultSpace`
  Určuje cílový barevný prostor, ve kterém bude vrácena barva při výběru nebo konverzi. Ve druhých dvou případech je tato položka ignorována. Oproti předchozí položce je zde možné pro výběr barvy z palety nastavit navíc ještě hodnotu `plugInColorServicesChosenSpace`, která zajistí, že barva bude vrácena v tom barevném prostoru, ve kterém ji uživatel v dialogu nastavil. Potom je zde po provedení výběru uložen identifikátor tohoto prostoru.

- `Boolean resultGamutInfoValid`
  Po dokončení operace je zde uložena logická hodnota, která informuje o tom, jestli daná operace nastavovala položku `resultInGamut` nebo ne.

- `Boolean resultInGamut`
  Informuje o tom, zda vstupní barva leží uvnitř gamutu výstupního barevného prostoru, nebo byla během převodu „oříznuta“ na nejbližší barvu, která toto splňuje. Hodnota zde uložená má smysl pouze tehdy, je-li položka `resultGamutInfoValid` nastavená na `true`.

- `void *reservedSourceSpaceInfo`
  Tato položka je vyhrazena pro interní potřeby a musí být vždy nastavena na `NULL`.

- `void *reservedResultSpaceInfo`
  Tato položka je vyhrazena pro interní potřeby a musí být vždy nastavena na `NULL`.

- `int16 colorComponents[4]`
  Pole obsahující čtyři prvky pro jednotlivé kanály barvy. Při konverzi barevných prostorů určuje vstupní barvu, při volání dialogu pro výběr barvy udává výchozí barvu nastavenou v dialogu po zobrazení. V obou případech je formát barvy určen položkou `sourceSpace`, u dalších dvou operací je pole ignorováno.

Po vykonání operace je zde vždy uložena výsledná barva, jejíž formát udává pro první dvě operace `resultSpace` a pro druhé dvě `sourceSpace`.

Pole má vždy čtyři položky bez ohledu na to, kolik kanálů daný barevný prostor obsahuje. Jestliže má daný prostor méně než čtyři kanály, jsou prvky obsazovány od začátku pole a přebývající prvky jsou ignorovány.

- `void *reserved`
  Tato položka je vyhrazena pro interní potřeby a musí být vždy nastavena na `NULL`.
- `union selectorParameter`
  Obsahuje tři podpoložky, které určují dodatečný parametr v jednotlivých typech operací.
  - `Str255 *pickerPrompt`
    Řetězec, který bude zobrazen v titulku dialogu pro výběr barvy.
  - `Point *globalSamplePoint`
    Bod v obraze, ze kterého má být načtena barva.
  - `int32 specialColorID`
    Identifikátor speciální barvy, může mít buď hodnotu `plugIncolorServicesForegroundColor` pro barvu popředí, nebo `plugIncolorServicesBackgroundColor` pro barvu pozadí.

Strukturu `ColorServicesInfo` je tedy nutné nejprve naplnit v souladu s tím, jakou operaci chceme vykonat, a následně ji odkazem předat metodě `colorServices`. K usnadnění inicializace struktury je v Photoshop SDK připravená metoda `CSInitInfo`, stejně tak pro jednotlivé operace je možné použít předpřipravené metody `CSPickColor` nebo `CSConvertColor`. Nevýhodou jejich používání je ale přílišná robustnost (hlavně neustálá kontrola správnosti parametrů), což proces zbytečně zpomaluje.

Nyní, když už dokážeme převést obrazová data, která získáme od Photoshopu, do všech potřebných barevných prostorů, může nastoupit vlastní filtrační algoritmus. Uživatel určil pro každý z kanálů obrázku, ze kterých složek dostupných barevných prostorů se budou jeho hodnoty skládat. Pro každou takovou složku pak nadefinoval body určující, jak bude daná hodnota této složky ovlivňovat příslušný kanál obrázku. Stejný postup provedl i se druhou sadou složek pro modifikátor. Důležité je říci, že svislá souřadnice bodu určující míru ovlivnění kanálu je normalizována do rozsahu $0 \ldots 1$. Definované body je pak potřeba v rámci každé složky proložit interpolační křivkou. K tomu jsem použil kubickou spline funkci (kubickou křivku), jejíž výpočet je popsán níže. Výsledný proces tedy vypadá následovně:

1. Vytvoříme mapu hodnot pro každou aktivní složku v každém kanálu. Tedy pro všechny hodnoty, kterých může daná složka nabývat, vypočítáme hodnotu interpolační křivky a uložíme do pole.
2. Projdeme všechny pixely obrázku a barvu každého z nich převedeme do všech potřebných barevných prostorů, abychom získali konkrétní hodnoty barevných složek.
3. Pro každý obrazových bodů a každý z jeho kanálů vyhledáme hodnoty aktivních složek v příslušných mapách. Tím získáme příspěvky těchto složek do daného kanálu bodu. Všechny příspěvky sečteme a uložíme pro další zpracování.

4. Nyní provedeme tentýž postup jako v předchozím kroku, pouze budeme vyhledávat v mapách složek modifikátoru namísto složek vstupu. Ve výsledku dostaneme tedy k hodnotě vstupu z předchozího kroku také hodnotu modifikátoru. Tu vynásobíme dvěma a výsledkem nakonec vynásobíme vstup. Násobením dvěma zajistíme, že pro hodnoty uprostřed rozsahu $(0, 5)$ nebude ovlivňovat hodnotu vstupu, pro nižší hodnoty bude vstup zeslabovat a pro vyšší naopak zesilovat. Na závěr je nutné výsledek ještě denormalizovat na rozsah hodnot daného kanálu a zaokrouhlit na celé číslo. To pak můžeme uložit jako novou hodnotu zpracovávaného kanálu obrazového bodu.

4.4.3 Kubická interpolační spline funkce

Mějme zadanou funkci f pomocí tabulky hodnot $\{x_i, f(x_i)\}$; $i = 0, 1, \ldots n$. Funkci $s(x)$ nazveme kubickou spline interpolací funkce f, splňuje-li následující vlastnosti:

1. je na každém intervalu $\langle x_i, x_{i+1}\rangle$; $i = 0, 1, \ldots n-1$ polynomem třetího řádu ve tvaru
$$s_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3$$
2. splňuje podmínky interpolace
$$s(x_i) = f(x_i) \quad \text{tj.} \quad s_i(x_i) = f(x_i)\ ;\ i = 0, 1, \ldots n-1\ \wedge\ s_{n-1}(x_n) = f(x_n)$$
3. je spojitá na celém intervalu $\langle x_0, x_n\rangle$, čili v uzlech platí
$$s_i(x_{i+1}) = s_{i+1}(x_{i+1})\ ;\ i = 0, 1, \ldots n-2$$
4. má spojitou první derivaci na $\langle x_0, x_n\rangle$, tedy
$$s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})\ ;\ i = 0, 1, \ldots n-2$$
5. má spojitou druhou derivaci na $\langle x_0, x_n\rangle$
$$s_i''(x_{i+1}) = s_{i+1}''(x_{i+1})\ ;\ i = 0, 1, \ldots n-2$$

Pro konstrukci funkce potřebujeme zjistit 4 koeficienty a_i, b_i, c_i, d_i pro každý z n intervalů, celkem $4n$ neznámých. Z vlastností 2.–5. však vyplývá pouze $4n - 2$ podmínek, je proto nutné ještě určit dvě dodatečné podmínky. Existuje několik variant těchto dodatečných podmínek. Nejčastěji používané jsou tzv. přirozené podmínky, které jsou pro tuto aplikaci vhodné a jsou užity také v nástroji Křivky ve Photoshopu. Přirozené podmínky jsou určeny nulovou druhou derivací v krajních bodech funkce, tedy $s''(x_0) = 0$ a $s''(x_n) = 0$.

Nyní máme tedy sadu podmínek kompletní a můžeme jednoznačně určit koeficienty polynomů. Funkci s(x) si přepíšeme do tvaru

$$s(x) = \eta_0 s_0(x) + \eta_1 s_1(x) + \cdots + \eta_{n-1} s_{n-1}(x)$$

kde $\eta_i = \eta_i(x)$ jsou charakteristické funkce intervalů, tj.

$$\eta_i(x) = \begin{cases} 1\ ;\ x \in \langle x_i, x_{i+1}\rangle \\ \\ 0 \text{ jinak} \end{cases}$$

Z první vlastnosti vyvodíme vztahy v uzlových bodech:

$$s(x_i) = a_i \qquad s'(x_i) = b_i \qquad s''(x_i) = c_i \qquad s'''(x_i-) = d_{i-1} \quad \text{a} \quad s'''(x_i+) = d_i$$

označíme h_i velikosti intervalů mezi uzly:

$$h_i = x_{i+1} - x_i \ ; \ i = 0, 1, \ldots n-1$$

a pomocí těchto vztahů pak přepíšeme podmínky 2. až 5.:

2. interpolační podmínky

$$f(x_i) = a_i \ ; \ i = 0, 1, \ldots n-1 \qquad f(x_n) = a_{n-1} + b_{n-1}h_{n-1} + \frac{c_{n-1}}{2}\, h_{n-1}^2 + \frac{d_{n-1}}{6}\, h_{n-1}^3$$

3. spojitost

$$f(x_{i+1}) = a_i + b_i h_i + \frac{c_i}{2}\, h_i^2 + \frac{d_i}{6}\, h_i^3 \ ; \ i = 0, 1, \ldots n-2$$

4. spojitá první derivace

$$b_{i+1} = b_i + c_i h_i + \frac{d_i}{2}\, h_i^2 \ ; \ i = 0, 1, \ldots n-2$$

5. spojitá druhá derivace

$$c_{i+1} = c_i + d_i h_i \ ; \ i = 0, 1, \ldots n-2$$

Tyto podmínky můžeme pak použít k sestavení soustavy lineárních algebraických rovnic, ze kterých vypočítáme koeficienty c_i a na jejich základě potom i b_i a d_i. Po úpravách získáme $n-1$ rovnic:

$$\alpha_i c_{i-1} + 2c_i + \beta_i c_{i+1} = g_i \ ; \ i = 1, 2, \ldots n-1$$

$$\begin{aligned} \text{kde} \quad \beta_i &= \frac{h_i}{h_i + h_{i-1}} \\ \alpha_i &= 1 - \beta_i \\ g_i &= \frac{6}{h_i + h_{i-1}} \left(\frac{f(x_{i+1}) - f(x_i)}{h_i} - \frac{f(x_i) - f(x_{i-1})}{h_{i-1}} \right) \end{aligned}$$

Abychom dostali jednoznačné řešení, potřebujeme dvě další rovnice. Ty získáme z dodatečných podmínek:

$$s''(x_0) = s''(x_n) = 0 \ \wedge \ s''(x_i) = c_i \quad \Rightarrow \quad c_0 = c_n = 0$$

V maticovém zápisu pak soustava vypadá takto:

$$\begin{bmatrix} 2 & \beta_0 & & 0 & \cdots & 0 \\ \alpha_1 & 2 & \beta_1 & & \ddots & \vdots \\ & \alpha_2 & 2 & \beta_2 & & 0 \\ 0 & & \ddots & \ddots & \ddots & \\ \vdots & \ddots & & \alpha_{n-1} & 2 & \beta_{n-1} \\ 0 & \cdots & 0 & & \alpha_n & 2 \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{n-1} \\ g_n \end{bmatrix}$$

kde první a poslední rovnice soustavy jsou definovány dodatečnými podmínkami. Matice soustavy je třídiagonální. Na diagonále jsou dvojky a součet prvků na vedlejších diagonálách je vždy 1 ($\alpha_i + \beta_i = 1$), matice je proto tzv. ostře diagonálně dominantní [7]. Z těchto tvrzení plyne, že je matice regulární, tedy že tato soustava má právě jedno řešení, konstanty c_i jsou jí určeny jednoznačně.

Z dřívějších vztahů je zřejmé, že $a_i = f(x_i)$, ostatní koeficienty pak dopočítáme ze získaných c_i

$$b_i = \frac{f(x_{i+1}) - f(x_i)}{h_i} - \frac{2c_i + c_{i+1}}{6} h_i \; ; \quad d_i = \frac{c_{i+1} - c_i}{h_i}$$

a můžeme dosadit do výše uvedeného vztahu, pomocí kterého pak získáme hodnotu křivky v libovolném bodě na intervalu $\langle x_i, x_{i+1} \rangle$

$$s_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3$$

4.4.4 Windows API

Uživatelské rozhraní zásuvného modulu pro Photoshop lze vytvořit dvěma cestami. První z nich je použití Adobe Dialog Manageru (ADM), sady funkcí pro práci s dialogovými okny a jejich ovládacími prvky. Předností ADM je hlavně možnost použití takto vytvořeného GUI (Graphical User Interface, grafické uživatelské rozhraní) jak na Windows, tak na Mac OS, ale také jednotný vzhled dialogu a standardních nástrojů Photoshopu. Obecně ale použití ADM není autory pluginů příliš doporučováno kvůli jeho chybám. Sám jsem se setkal s problémy, kdy se dialog vytvořený pomocí ADM vůbec nezobrazil, ačkoliv jinak pracoval zásuvný modul v pořádku. Proto jsem se rozhodl namísto ADM využít druhého řešení, kterým je použití nativních systémových služeb pro vytvoření a ovládání GUI. To s sebou nese nevýhodu, kterou je nutnost dvou různých verzí zdrojových kódů, pokud bychom chtěli plugin používat na obou zmíněných operačních systémech. Omezím se tedy pouze na Microsoft Windows a využiji služeb jeho aplikačního rozhraní (Windows API, zkráceně WinAPI).

Windows Application Interface je nejnižší úroveň interakce mezi běžnými aplikacemi a systémem. Na nižších úrovních pracují kromě součástí systému většinou jen ovladače. Z důvodů poměrně zdlouhavé implementace některých funkcí v programech pomocí WinAPI a také tomu, že není objektově orientované, byla vytvořena řada nadstaveb. Mezi takové nadstavby patří například MFC (Microsoft Foundation Classes) a WTL (Windows Template Library), ale i VCL (Visual Component Library) používaná v nástrojích společnosti Borland. Nakonec i zmíněný Adobe Dialog Manager musí na systému Windows používat jeho aplikační rozhraní. Výhodou přímého použití WinAPI oproti zmíněným nadstavbám je zejména vyšší rychlost.

Funkce nabízené WinAPI lze rozdělit do sedmi skupin:

- **Základní funkce**
  Poskytují přístup k nejdůležitějším systémovým prostředkům, jako jsou souborové systémy, připojená zařízení, procesy a vlákna, systémový registr a správa chyb.

- **Grafické funkce**
  Souhrnně se označují Graphics Device Interface (GDI) a umožňují grafický výstup

na monitor, ale i tiskárny a jiné grafické periférie. V novějších verzích Windows se lze setkat také s inovovanou sadou GDI+, která poskytuje mnohem komplexnější grafické služby.

- **Uživatelské rozhraní**
  Tyto služby zajišťují vytváření aplikačních oken, uživatelských dialogů a nejzákladnějších ovládacích prvků jako jsou tlačítka nebo posuvníky. Dále umožňují reakce na události klávesnice a myši.

- **Knihovna dialogů**
  Obsahuje funkce sloužící k vyvolání systémových dialogů pro otevírání a ukládání souborů, výběr písma, barvy, nastavení tiskárny apod.

- **Knihovna ovládacích prvků**
  Je známá obvykle pod názvem Common Controls Library a přináší definice pokročilejších ovládacích prvků okna – stavového řádku, panelů nástrojů, záložek, ukazatelů průběhu a dalších.

- **Funkce shellu**
  Shell je softwarová vrstva zprostředkovávající rozhraní mezi jádrem systému a aplikacemi. V této kategorii jsou přítomny různé dodatečné funkce a ovládací prvky.

- **Síťové služby**
  Umožňují programům síťovou komunikaci pomocí různých protokolů přítomných v systému.

Z těchto skupin využijeme zejména funkce pro uživatelské rozhraní, knihovny ovládacích prvků a také grafické funkce.

Se všemi dialogy, ovládacími prvky i aplikačními okny je ve WinAPI nakládáno naprosto stejně. Jednotlivé prvky (budu je souhrnně označovat jako okna) jsou uspořádány v hierarchii. Na nejvyšší úrovni jsou hlavní okna programů, která nemají přiřazené žádné rodičovské okno. Ta jsou pak rodiči oknům na nich umístěným, ale také dialogovým oknům, které vyvolávají. Tito jejich „potomci“ mohou zase obsahovat další prvky a tak dále. Každé okno určuje jeho unikátní identifikátor, tzv. handle. Ten získáme při jeho vytváření, které se děje pomocí funkce

```
HWND CreateWindowEx(
    DWORD dwExStyle, LPCTSTR lpClassName, LPCTSTR lpWindowName,
    DWORD dwStyle, int x, int y, int nWidth, int nHeight,
    HWND hWndParent, HMENU hMenu, HINSTANCE hInstance, LPVOID lpParam
)
```

Jednotlivé parametry mají následující význam:

- `DWORD dwExStyle`
  Přináší nové styly oken, které byly představeny s 32bitovou verzí Windows. Z důvodů zpětné kompatibility nebylo možné tyto styly přidat mezi hodnoty parametru `dwStyle`. Tímto parametrem lze například umožnit oknu přijímat soubory na něj přetažené (tzv. drag & drop) nebo nastavit okno jako „vždy navrchu“. Funkce `CreateWindow` používaná na 16bitových Windows tuto položku neobsahuje.

- `LPCTSTR lpClassName`
  Jméno třídy oken, ke které bude vytvořené okno náležet, určuje typ tohoto okna. Možnými hodnotami pro systémové třídy jsou:
  - `"BUTTON"` – tlačítko
  - `"COMBOBOX"` – rozbalovací seznam
  - `"EDIT"` – textové pole
  - `"LISTBOX"` – okno se seznamem
  - `"MDICHILD"` – podokno MDI (Multiple Document Interface, vícedokumentové rozhraní) hlavního okna obvykle obsahující otevřený dokument
  - `"SCROLLBAR"` – posuvník
  - `"STATIC"` – statická komponenta, která může sloužit jako textový popisek, zobrazovat ikonu či bitmapu, nebo tvořit rodičovskou komponentu pro skupinu ovládacích prvků.

  Kromě uvedených tříd lze také vytvořit a zaregistrovat třídu vlastní.

- `LPCTSTR lpWindowName`
  Text, který se zobrazí v závislosti na typu okna třeba jako popisek tlačítka nebo titulek dialogu.

- `DWORD dwStyle`
  Styl, který může být kombinací několika různých hodnot určujících vlastnosti okna – zda má mít titulkový pruh, je nebo není roztažitelné, jaký má typ okraje a další.

- `int x, y`
  Pozice okna vzhledem k hornímu levému rohu klientské oblasti rodičovského okna. Klientská oblast okna se může lišit od celkové oblasti zabrané oknem. Například titulkový pruh nebo okraje jsou součástí okna, ale do jeho klientské oblasti se nepočítají. Pokud okno nemá definovaného rodiče (je navrchu hierarchie), jsou tyto parametry vztaženy k hornímu levému okraji obrazovky.

- `int nWidth, nHeight`
  Šířka a výška celé oblasti okna. Pokud bychom chtěli nastavit přesně rozměry a pozici klientské oblasti, můžeme k tomu využít funkce `AdjustWindowRectEx`, která zjistí pozici a rozměry celého okna ze zadané klientské oblasti.

- `HWND hWndParent`
  Zde je možné oknu přiřadit handle rodičovského okna. Pokud je uvedena hodnota `NULL`, bude okno nejvýše v hierarchii a to dokonce i v případě, že se jedná pouze o některý z ovládacích prvků.

- `HMENU hMenu`
  Odkaz na hlavní menu okna nebo identifikátor ovládacího prvku v rámci dialogu. Tímto parametrem můžeme buď oknu aplikace přiřadit hlavní menu, nebo ovládacímu prvku přiřadit celočíselný identifikátor (ID), pod kterým bude přístupný v dialogovém okně (používá se často pro tlačítka, kterými se dialog zavírá). Pokud jsme pro okno zaregistrovali vlastní třídu, je možné přiřadit hlavní menu i v její definici a zde ponechat `NULL`.

- `HINSTANCE hInstance`
  Identifikátor instance programu, který je programu přidělován při spuštění a je vždy jednoznačný. Díky tomu lze například rozlišit okna dvou spuštěných instancí téhož programu.

- `LPVOID lpParam`
  Odkaz na další data, která mohou být potřeba při vytváření okna, například v případě oken MDI.

Jako základ uživatelského rozhraní zásuvného modulu bude použit modální dialog. Tento typ dialogu zamezí po svém zobrazení používání okna, které ho vyvolalo, dokud není uzavřen. Stejný typ bychom získali i při použití ADM. Dialog je možné vytvořit mnohem snazší cestou, než je tomu u běžného aplikačního okna. Pomocí editoru dialogů Visual Studia můžeme snadno vytvořit šablonu dialogového okna v souboru zdrojů pluginu. Z ní pak lze vytvořit vlastní modální dialog zavoláním funkce

```
INT_PTR DialogBox(
    HINSTANCE hInstance, LPCTSTR lpTemplate,
    HWND hWndParent, DLGPROC lpDialogFunc
)
```

Její parametry jsou tyto:

- `HINSTANCE hInstance`
  Podobně jako u `CreateWindowEx` jde o identifikátor instance programu. Ten zjistíme pomocí funkce `GetDLLInstance`, která je součástí Photoshop SDK.

- `LPCTSTR lpTemplate`
  Název šablony, podle které bude dialog vytvořen. Název získáme použitím makra `MAKEINTRESOURCE`, kterému předáme identifikátor vytvořené šablony.

- `HWND hWndParent`
  Handle rodičovského okna. Protože bude náš dialog modální vzhledem k hlavnímu oknu Photoshopu, potřebujeme znát handle tohoto okna. Získáme jej z položky `platformData`, která je součástí struktury `FilterRecord`.

- `DLGPROC lpDialogFunc`
  Odkaz na metodu obsluhující tzv. smyčku zpráv dialogu. Smyčka zpráv je podrobněji popsána dále v dokumentu.

Výstupní hodnotou metody `DialogBox` je číslo nesoucí obvykle informaci o tom, kterým tlačítkem byl dialog ukončen. Kromě této metody poskytuje WinAPI ještě metodu `CreateDialog`, která přijímá shodné parametry. Výsledkem jejího volání je ale dialog nemodální, po jehož zobrazení lze dále pracovat s jeho rodičovským oknem.

Komunikace mezi systémem a oknem, ale i mezi okny navzájem, probíhá pomocí tzv. zpráv. Zpráva informuje dané okno o nějaké akci nebo požaduje vykonání nějaké akce. Je implementována jako struktura `MSG` s následujícími položkami:

- `HWND hwnd`
  Obsahuje handle okna, kterému je zpráva určena.

- `UINT message`
  Číslo, které určuje druh zprávy. Může to být jedna z předdefinovaných konstant pro běžné systémové zprávy (např. `WM_LBUTTONDOWN` pro stisknutí levého tlačítka myši), případně konstanta definovaná uživatelem, pak jde o uživatelskou zprávu.

- `WPARAM wParam`
  Je to 32bitové číslo obsahující dodatečné informace o zprávě. Je rozděleno na dvě části po 16 bitech. Například pro zprávu typu `WM_COMMAND` je v první části uložen číselný identifikátor prvku na dialogu, který zprávu vyvolal (tlačítko, na které bylo kliknuto apod.), ve druhé pak druh vykonané akce (v tomto případě tedy kliknutí). V 16bitových Windows měla tato položka pouze 16 bitů (datový typ `WORD`, proto `WPARAM`).

- `LPARAM lParam`
  Druhá pozice pro dodatečné informace. Má opět délku 32 bitů, ale na rozdíl od předchozí položky ji měla vždy (typ `LONG`, tedy `LPARAM`). Obvykle obsahuje handle prvku, který vyvolal zprávu. Pokud je potřeba ke zprávě přiložit více informací, než jsou schopny pojmout položky `wParam` a `lParam`, bývá zde uložen odkaz na datovou strukturu obsahující všechny tyto informace.

- `DWORD time`
  Udává čas odeslání zprávy.

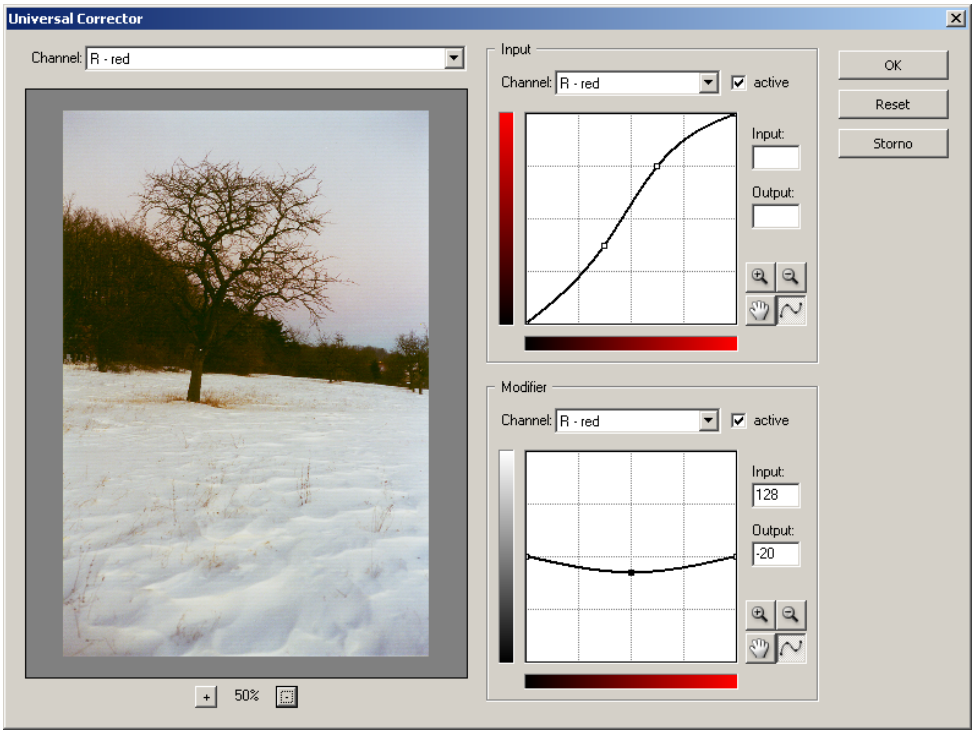
- `POINT pt`
  Pozice kurzoru myši ve chvíli, kdy byla zpráva odeslána. Pozice je uložena v absolutních souřadnicích obrazovky.

Zpracování zpráv v programu probíhá uvnitř smyčky zpráv. Všechny zprávy určené oknům programu jsou ukládány do fronty, ze které jsou v nekonečném cyklu po jedné vybírány a zpracovávány. Pokud vytváříme obyčejný program, musíme smyčku implementovat sami. V případě zásuvného modulu ji za nás obstarává Photoshop. Při vytváření dialogového okna je pouze třeba předat odkaz na funkci, která bude provádět samotné zpracování uvnitř smyčky. Taková metoda obvykle obsahuje konstrukci příkazu `switch`, pomocí které rozlišuje jednotlivé typy zpráv a pro každý z nich provádí příslušnou akci. Do ní přicházejí zprávy určené samotnému dialogovému oknu, ale také všem oknům v hierarchii pod ním. Metoda však nemusí reagovat na všechny typy zpráv – pokud zprávu nezpracuje dialogové okno, může ji přenechat ke zpracování svému rodičovskému oknu. Funguje také opačný postup. Můžeme totiž libovolnému ovládacímu prvku dialogu přiřadit jeho vlastní metodu pro zpracování zpráv. Tomuto postupu se říká subclassing. Ovládací prvek pak bude zpracovávat zprávy určené jemu a všem oknům, které jsou v hierarchii pod ním.

4.5 Uživatelská příručka

Plugin nainstalujete jednoduchým zkopírováním souboru `ucorrect.8bf` do složky zásuvných modulů Photoshopu, standardně `Plug-Ins`, nebo některé její podsložky. Po spuštění Photoshopu je pak dostupný přes nabídku *Filter* | *Color Corrections* | *Universal Corrector*. Modul pracuje v barevných režimech RGB, CMYK a Lab jak s 8, tak se 16 bity na kanál, v ostatních režimech není v nabídce dostupný.

Po vybrání filtru z nabídky se zobrazí dialog s jeho volbami, který můžete vidět na Obr. 4.2. Vlevo je zobrazen náhled upravovaného obrázku. Pod ním jsou tlačítka, kte-



Obr. 4.2: Dialog zásuvného modulu

rými lze ovládat zvětšení náhledu. Pokud není obraz při aktuálně nastaveném zvětšení zobrazen celý, je možné ho posouvat klepnutím a tažením myší v prostoru náhledu. Nad náhledem je rozbalovací seznam pro výběr kanálu obrazu, který se bude upravovat. Po dokončení úprav lze vybrat jiný kanál, přičemž nastavení předchozího zůstane zachováno.

V pravé části jsou pak dvě skupiny ovládacích prvků. Horní slouží k nastavení vstupních hodnot barevného kanálu, druhá pak určuje hodnotu modifikátoru. Ve vrchní části obou skupin je umístěn rozbalovací seznam pro výběr barevné složky. Vedle něj je pak zaškrtávací pole, jehož zaškrtnutím složku aktivujeme, tedy zajistíme, že se bude podílet na výsledném obrazu. Pod seznamem složek je čtvercové pole, ve kterém jsou vykreslovány křivky pro jednotlivé složky. Křivka právě vybrané složky je vždy zobrazena silnou černou čarou, tenkými barevnými čarami jsou pak zobrazeny křivky všech aktivních složek v rámci vybraného barevného kanálu. Po levé straně pole v horní skupině je umístěn pruh s barevným přechodem odpovídajícím právě vybranému barevnému kanálu. V dolní skupině je přechod vždy černobílý. Pod každým z polí je podobný pruh s přechodem, který v obou případech odpovídá právě vybrané složce. Z obou barevných přechodů by měl uživatel získat představu, jak se bude určitá hodnota vybrané barevné složky podílet na vstupu daného kanálu obrázku resp. modifikátoru tohoto vstupu. Napravo každé ze skupin je pak dvojice textových polí, kde se zobrazují a také dají upravovat souřadnice právě vybraného bodu křivky. Pod nimi se nachází čtveřice tlačítek po výběr funkce myši v poli pro křivky. Horní levé tlačítko přepíná do režimu přibližování, klepnutím myši do pole s křivkami pak toto v místě klepnutí dvakrát přiblížíme. Tlačítko vedle něj má opačný efekt, po jeho zapnutí se klepnutím myši křivky dvakrát oddálí. Tlačítko vlevo dole pak umožňuje přiblížené křivky posouvat. Nakonec poslední tlačítko přepíná do režimu editace

křivky. Toto je implicitně zapnuto při spuštění zásuvného modulu.

V editačním režimu je možné přidávat, odebírat a posouvat body na křivce. Stisknutím levého tlačítka myši v blízkosti některého ze stávajících bodů tento označíme. Tažením ho můžeme přesunout na jiné místo. Jestliže bod přesuneme mimo vyhrazenou oblast nebo za sousední bod, bude bod z křivky odebrán. Můžeme ho do křivky vrátit přetažením zpět. Jestliže byl bod odebrán, způsobí uvolnění tlačítka myši jeho trvalé odstranění. Krajní body křivky nelze odebrat.

Úplně napravo dialogu se nachází poslední trojice tlačítek. Shora je to tlačítko *OK* pro potvrzení a provedení nastavených korekcí v obrázku. Pod ním je umístěno tlačítko *Reset*, které obnoví nastavení korekcí na výchozí hodnotu. A posledním je tlačítko *Cancel*, které ukončí plugin bez provedení změn v obrázku.

Poznámka:
Po aktivování složky barevného prostoru, který zatím nebyl použit, je třeba počkat na přepočítání hodnot pro tento prostor. To může pro větší obrázky trvat i desítky sekund, hlavně v případě prostorů CMYK a Lab.

4.6 Zhodnocení vytvořeného zásuvného modulu

Jak bylo řečeno, plugin staví zejména na ve Photoshopu zabudovaném nástroji Křivky, který ale svými schopnostmi překračuje. Možnost použití barevných kanálů, které nejsou součástí obrázku, rozšiřuje pole barevných korekcí, kterých je možné dosáhnout. Například kanály obrázku v systému HSB nelze v Photoshopu bez dodatečných nástrojů nikde získat a tedy ani využít k úpravě obrázku. Přitom právě tyto kanály mohou být velice užitečné, pokud chceme třeba změnit odstín nebo intenzitu některé barvy na obrázku, aniž bychom narušili zbytek barev. Navíc poskytuje plugin ještě funkci modifikátoru, který se dá využít k zesílení nebo zeslabení výsledné hodnoty opět na základě libovolné kombinace dostupných barevných složek.

Oproti testovaným zásuvným modulům nabízí vytvořené řešení poměrně silnou kontrolu nad výslednou podobou obrazu. Na jedné straně je to výhoda, protože lze s barvami v obrázku nakládat přesně tak, jak si to uživatel přeje. Na druhou stranu by ale úprava většího množství fotografií zabrala mnohem více času, než u nástrojů, které nabízejí plně automatickou nebo poloautomatickou barevnou korekci, byť ne vždy uspokojivou. Také je nasnadě říci, že k plnému a efektivnímu využití všech schopností pluginu by bylo potřeba si používání pluginu dobře osvojit, neboť jde o vcelku netradiční řešení.

Modul jsem otestoval na několika obrázcích, které jevily různé druhy barevného poškození. Použil jsem jak naskenované fotografie, kde došlo k rozladění barev při skenování, tak i snímek z digitálního fotoaparátu, u kterého pochybil systém vyvážení bílé. Zdrojové obrázky a výsledky jejich korekce jsou připojeny v příloze.

5

Závěr

V úvodu práce jsem přiblížil procesy, jimiž člověk vnímá barvy objektů, na které se dívá. Dále jsem se zaměřil na různé způsoby reprezentace barev v počítači. Popsal jsem barevné systémy, které se nejčastěji používají k ukládání digitálního obrazu, a zmínil jsem několik dalších významných. Byly vysvětleny důvody, které vedou k potřebě barevných korekcí, především rozdíl mezi fotografií a lidským zrakem. Následně bylo uvedeno několik metod, které se používají k napravení barev v obrázku.

Druhá kapitola byla zaměřena na nástroje, kterými lze barevné korekce provádět. Nejdříve byly představeny součásti Adobe Photoshopu – *Křivky*, *Míchání barev* a *Vyvážení barev*, které se k tomuto účelu nejčastěji používají, připomenuty byly i některé další. Poté jsem uvedl řadu korekčních zásuvných modulů určených pro Photoshop. Schopnosti všech uvedených nástrojů jsem otestoval a popsal. Nakonec jsem se zaměřil i na nástroje pro úpravy barev mimo Adobe Photoshop, zejména pak v oblasti digitálního videa.

Po zhodnocení situace v oblasti nástrojů pro barevné korekce jsem uvedl požadavky, které by měl vytvářený plugin splňovat. Na jejich základě jsem potom vytvořil návrh pluginu a jeho uživatelského rozhraní. Podle návrhu jsem modul také implementoval a popsal technologie využité k jeho vytvoření. V první řadě to bylo aplikační rozhraní Adobe Photoshopu. To je poměrně dobře zdokumentováno, proto jsem se zaměřil pouze na jeho součásti, které jsou klíčové v implementovaném procesu filtrace obrázku. Jako další jsem popsal výpočet kubické interpolační křivky, která tvoří další z nezbytných součástí filtrační metody. Třetí uvedenou technologií je aplikační rozhraní Windows (WinAPI), pomocí něhož je vytvořeno a ovládáno uživatelské rozhraní pluginu. Pokusil jsem se o jeho stručný popis a poté jsem představil metody, které byly využity k vytvoření ovládacích prvků a samotného dialogu. Vysvětlena byla také komunikace aplikací se systémem a aplikací navzájem ve WinAPI, která se děje pomocí předávání zpráv. Závěrem bylo popsáno také ovládání vytvořeného pluginu a plugin byl otestován na několika obrázcích.

Implementovaný zásuvný modul je schopen dosáhnout velmi dobrých výsledků s nejrůznějšími druhy barevné defromace v obrázcích. Jeho schopnosti jsou podle očekávání větší než schopnosti testovaných nástrojů. Jeho nevýhodou je kompletně manuální ovládání korekcí, díky čemuž může jeho použití vyžadovat více času než použití zmíněných nástrojů.

Literatura

[1] *Wikipedia, the free encyclopedia.*
Dostupné z: http://www.wikipedia.org

[2] PIHAN, Roman. *Oko versus fotoaparát.*
Poslední revize 1.7.2005.
Dostupné z: http://www.fotoroman.cz/techniques2/light_eye_camera.htm

[3] SKÁLA, Jiří.
Masking Images for DTP Needs, Implemented as Adobe Photoshop Plug-in.
Diplomová práce na Západočeské univerzitě v Plzni, 2006. 68 s.

[4] *Adobe Photoshop Application Programming Interface Guide.*
Verze CS, říjen 2003.
Dostupné na požádání z: http://www.adobe.com

[5] *Adobe Technical Journal: Color Services and the Color Picker Plug-in, Rev. 2.*
Vydáno 10.8.1997.
Dostupné na požádání z: http://www.adobe.com

[6] KAS, Thomas. *How to Write a Photoshop Plug-In, Part 1, 2.*
Dostupné z:
http://www.mactech.com/articles/mactech/Vol.15/15.04/PhotoshopPlug-InsPart1
http://www.mactech.com/articles/mactech/Vol.15/15.05/PhotoshopPlug-InsPart2

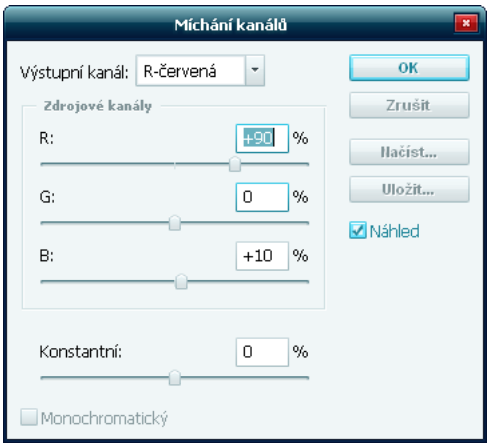
[7] PŘIKRYL, Petr – BRANDNER, Marek. *Numerické metody II.*
Skripta, Západočeská univerzita v Plzni, 2000.

[8] CHALUPA, Radek. *Učíme se WinAPI.*
Poslední revize 22.3.2003.
Dostupné z: http://www.radekchalupa.cz/clanky.aspx?kod=213&obsah=1

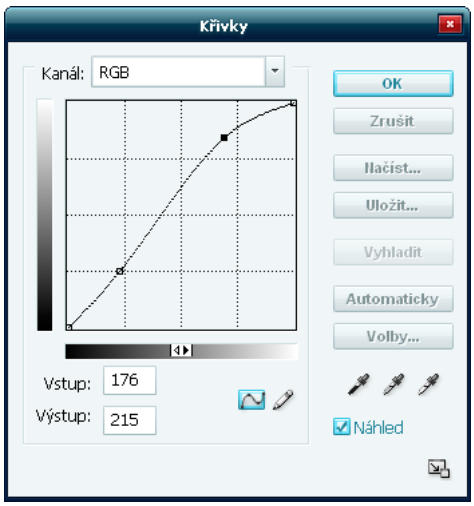
Příloha A

Uživatelská rozhraní testovaných nástrojů

A.1 Interní nástroje Photoshopu



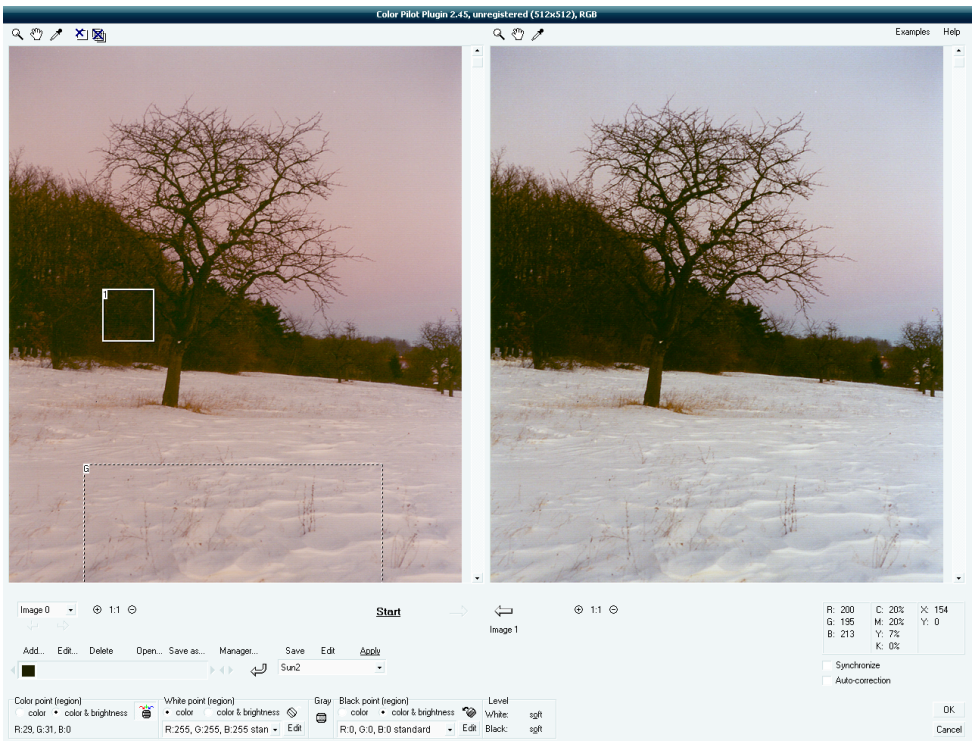
Obr. A.1: dialog nástroje Míchání kanálů



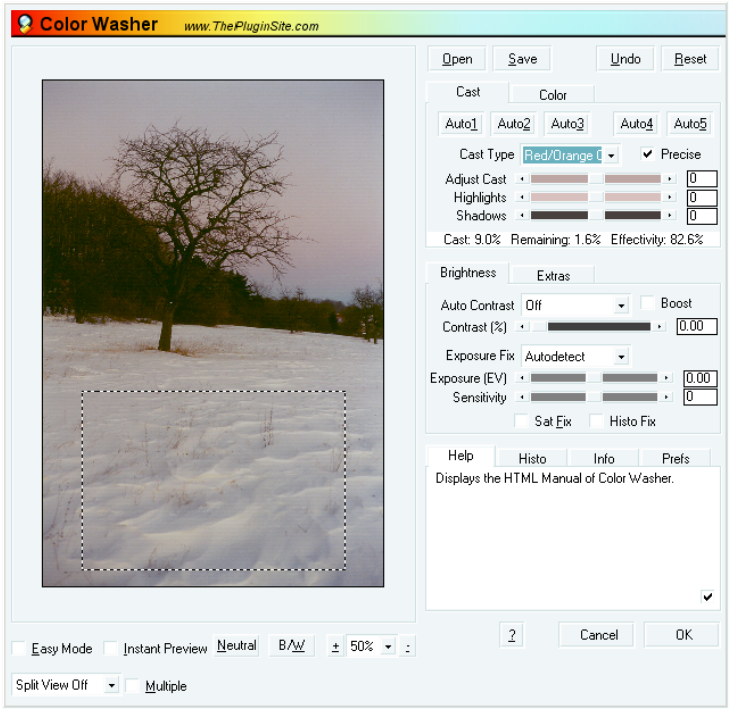
Obr. A.2: dialog nástroje Křivky

Obr. A.3: dialog nástroje Vyvážení barev

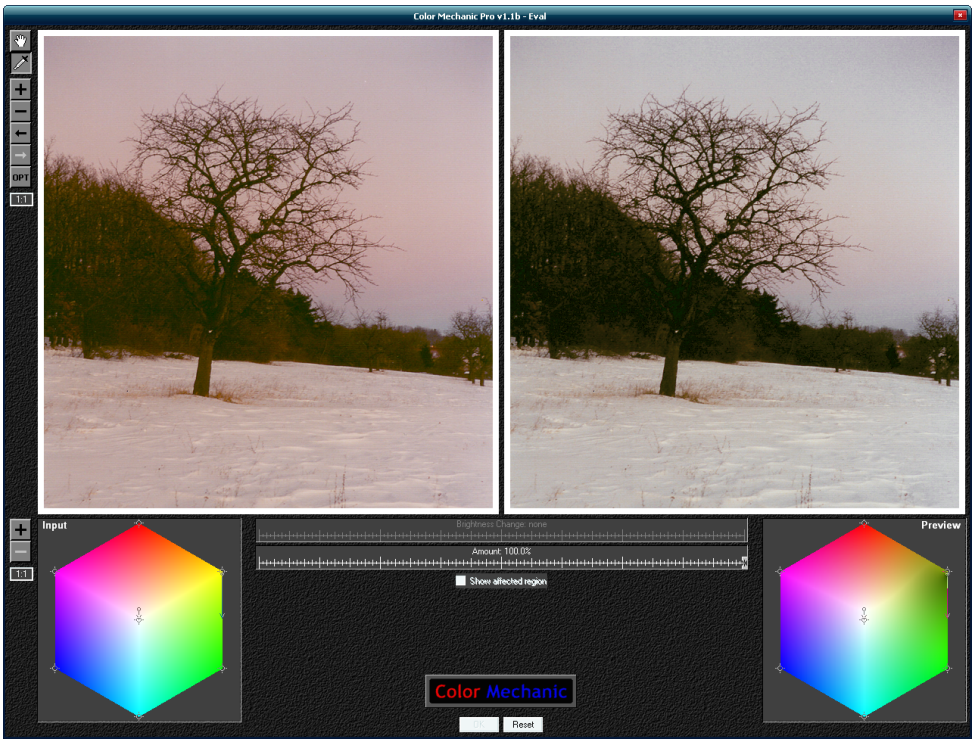
A.2 Zásuvné moduly



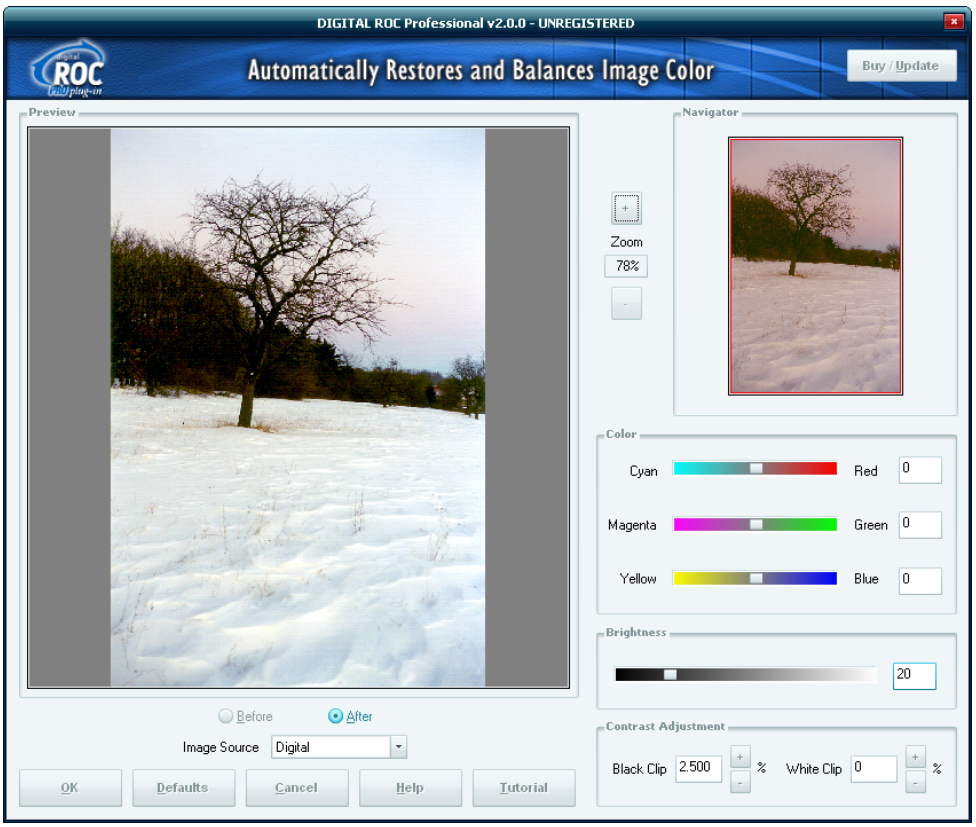
Obr. A.4: dialog modulu Color Pilot



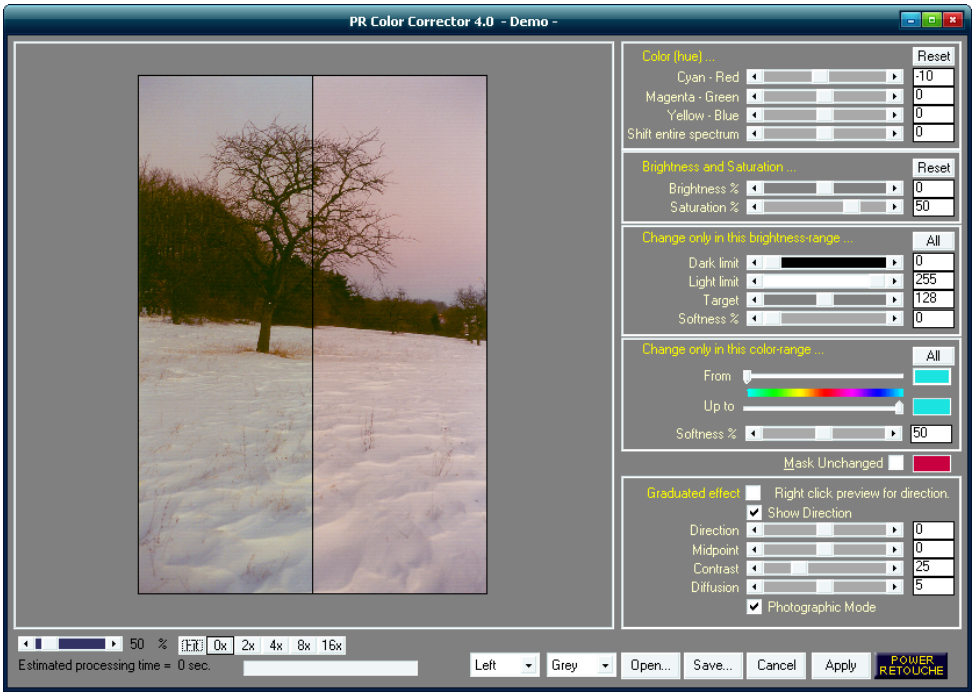
Obr. A.5: dialog modulu ColorWasher



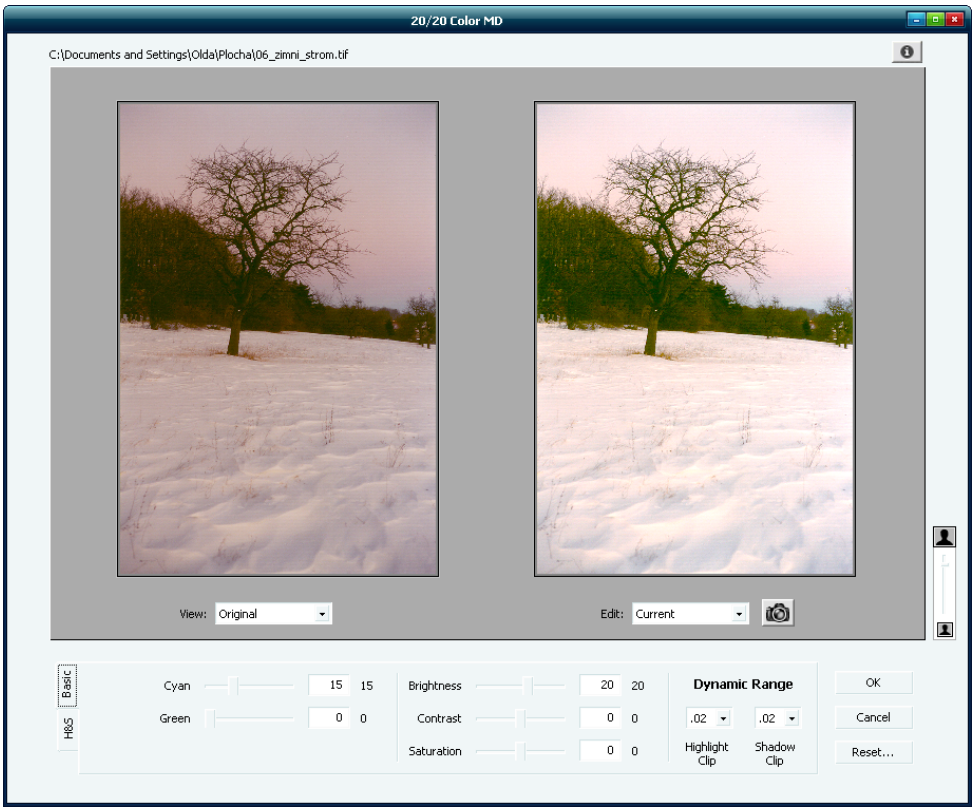
Obr. A.6: dialog modulu Color Mechanic



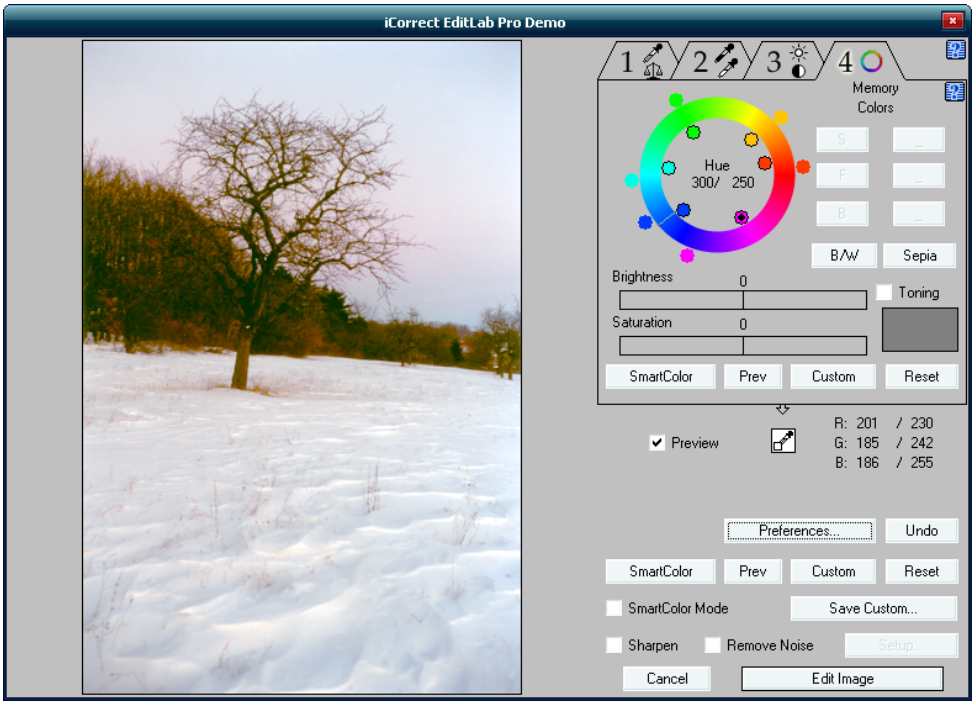
Obr. A.7: dialog modulu Kodak Digital ROC



Obr. A.8: dialog modulu Color Corrector

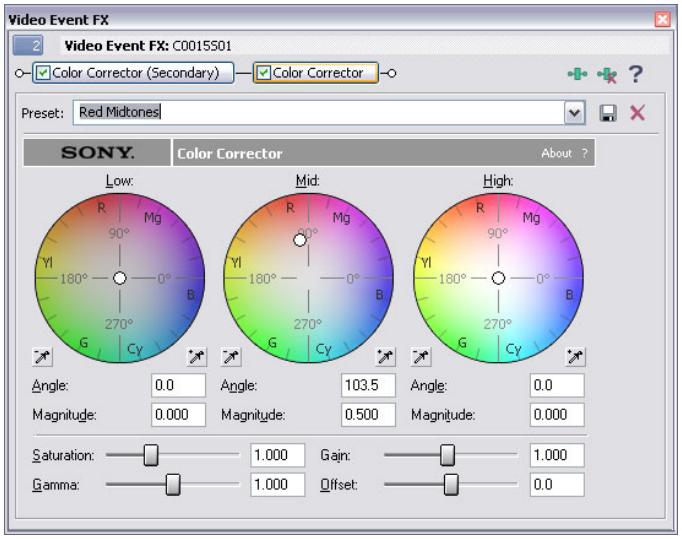


Obr. A.9: dialog modulu 20/20 Color MD

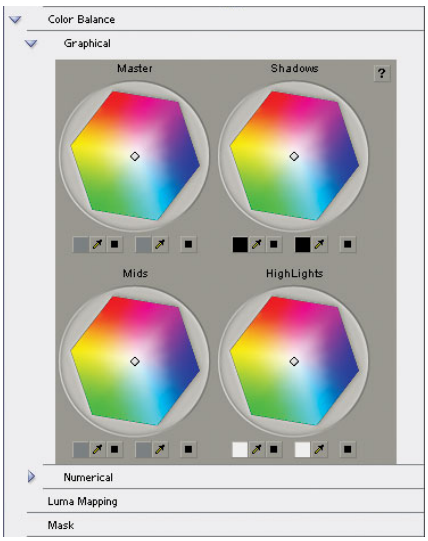


Obr. A.10: dialog modulu EditLab Pro

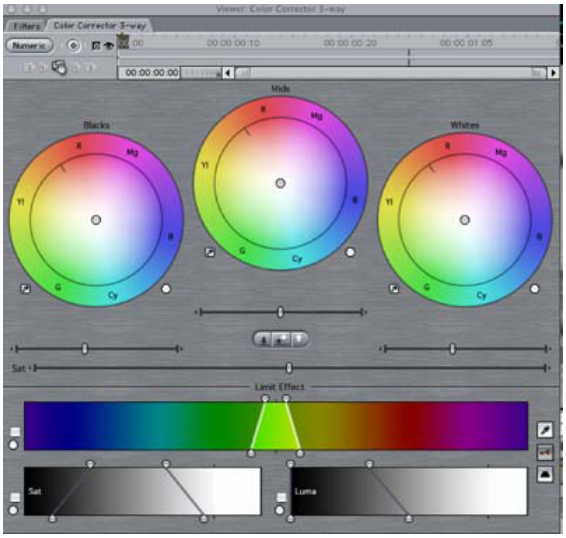
A.3 Nástroje pro střih digitálního videa



Obr. A.11: korekce barev v programu Sony Vegas



Obr. A.12: korekce barev v programu Adobe Premiere



Obr. A.13: korekce barev v programu Apple Final Cut

Příloha B

Výsledky práce pluginu

Obrázek vlevo je vždy původní, vpravo je pak jeho podoba po provedení korekcí.