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

Bakalářská práce

Barevné korekce obrazu a videa

Plzeň, 2010 Jan Zeman

Prohlášení

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

V Plzni dne ....

Abstract

The purpose of the Dikobraz application is to integrate so far separated methods of image and video color correction. In this thesis, we will first explore main differences between them. The same example will be corrected in all our reference applications, Adobe Photoshop CS4, Adobe Photoshop Lightroom 2, Iridas Speedgrade DI 2009 and Autodesk Lustre 2009. By that, also the GUIs of these programs will be presented. Afterwards, we will design a GUI for our application Dikobraz, that will have to combine them. We will also test different methods of memory management for the application.

Obsah

1. Úvod

Reprodukované obrazy, film či fotografie, provázejí člověka celým dnem, ať chce nebo ne. Pokud bude znát principy jejich výroby, může se oprostit od důvěry v to nejpovrchnější, že jsou jaksi samozřejmé a že za nimi stojí nějaká abstraktní moc a ne konkrétní lidé. Cílem práce koloristy je obrázek vylepšit a cílem naší práce je poskytnout mu k tomu nástroje. Poznamenejme však, že Margulisova definice, že obrázek je lepší pokud se více líbí zákazníkovi, není v žádném případě samozřejmá.

Hlavní motivací byla myšlenka propojit programy pro úpravu barev videa a programy pro úpravu barev obrázku do jediné aplikace. Standardní postupy pro barevné korekce ve videu se totiž od těch v obraze velmi liší, ač jde o tutéž činnost. Mají rozdílné GUI a jsou používány jiné filtry a jiné analyzační nástroje. Profesionálové z jednoho oboru si tak nerozumějí s kolegy z druhého.

Jako referenční berme aplikace Adobe Photoshop CS4 a Adobe Photoshop Lightroom 2 ze světa korekce obrazu a aplikace Iridas Speedgrade DI 2009 a Autodesk Lustre 2009, kterými jsou v postprodukci upravovány hollywoodské filmy. Aplikace Dikobraz má za cíl je sjednotit a poskytnout možnost provádět jak barevné korekce obrazu, tak i videa na profesionální úrovni. Umožní také lidem zvyklým upravovat video použít funkce známé z barevných korekcí obrazu, a naopak. Základní sada filtrů nabídne obě možnosti. Ačkoliv je nástroj zaměřen primárně na postprodukci, bude ho možno použít i pro kompletní výrobu amatérského videa, neboť bude osahovat i základní možnosti střihu.

Propojením tak vzniknou nové možnosti k dosažení požadovaného výsledku. Zatím žádná aplikace pro úpravu videa nenabízí možnost provádět korekce v různých barevných prostorech. Všechny úpravy budou prováděny ve vrstvách, úpravy tak budou nedestruktivní a bude tak zaznamenán jejich postup. Aplikace bude plně rozšiřitelná pomocí pluginů.

2. Teoretická část

Nejdříve popíšeme, jakým způsobem jsou barevné korekce prováděny v obrázku a jakým ve videu. Na konkrétním příkladu nastíníme typický postup práce v jednotlivých programech. Tak si zároveň názorně představíme nejdůležitější prvky jejich GUI, které musí Dikobraz integrovat také. Nebudeme používat formu referenční příručky, neboť hlavním cílem teoretické části je ujasnit si typické potřeby profesionálních koloristů.

2.1 Barevné korekce v obrázku

Následující postup korekce je převzat z [Margulis]. Jedná se už o složitější obrázek, který nestačí vylepšit použitím jediného filtru. Uvedený způsob však není jediný možný, cest k témuž výsledku je více. Úprava obrázku je navíc kreativní činnost, kde má konkrétní představu o výsledku jen každý sám. Od automatických nástrojů kvalitu očekávat nemůžeme, každý obrázek je jiný, má jiný obsah, jiné rozvržení barev. Jen člověk ví, kam by obrázek chtěl barevně posunout, a jednou částí jeho práce je také určit moment, kdy už je obrázek dostatečně dobrý.

2.1.1 Adobe Photoshop

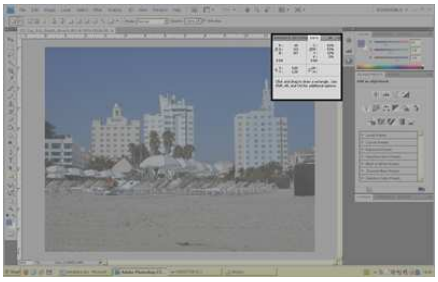
Na CD ke knize je soubor s obrázkem 2.1. Zkopírujeme si jej nejdřív do složky s fotografiemi. Spustíme Photoshopu a po kliku na File->Open... se otevře standardní okno Windows pro vybrání souboru. To je pro fotografy celkem nevýhodné, protože nemůžeme ihned při výběru souboru vidět metadata souboru, např. jakým fotoaparátem byla fotografie pořízena. Dnes se navíc každý uživatel, nejen profesionální fotograf, musí zároveň potýkat s velkým množstvím obrázků. Pokud se tak hledaný soubor jmenuje např. DSC02250.JPG, stojí nějaký čas ho vůbec najít. Proto se používá aplikace Adobe Bridge, která zobrazuje náhledy na jednotlivé soubory a po nalezení hledaného souboru spustí přímo další Adobe aplikaci pro jeho úpravu, u obrázků tedy Adobe Photoshop. Přesto máme pak spuštěnou aplikaci navíc. Dikobraz kolem sebe žádnou takovou rodinu produktů mít nebude, a správa souborů tak musí být součástí aplikace. Zapamatujme si také, že Photoshop pracuje přímo na konkrétních souborech, pokud obrázek po korekci uložíme, přepíše se také soubor.

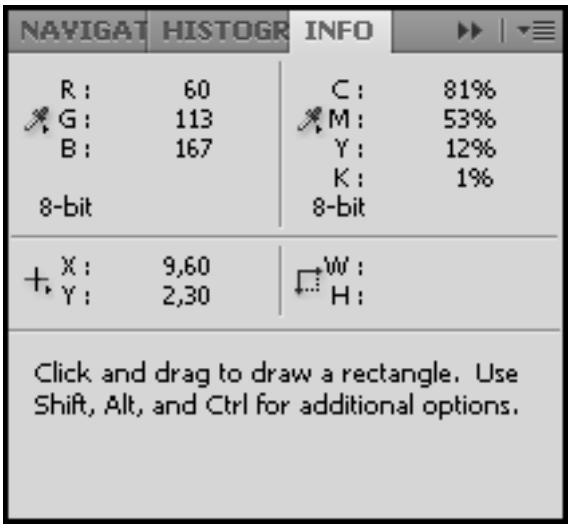
Obrázek 2.1 – Původní obrázek, který budeme v aplikacích upravovat

Barevné prostory

Na první pohled je naše fotografie v pořádku a laik si nevšimne chyb. Obloha je modrá, stromy zelené, přesně jako ve skutečnosti. Může tomu ale být obráceně, jsme zvyklí na to, že obloha bývá modrá a stromy zelené. Oko nemusí poznat i poměrně výrazný barevný posuv, protože tyto barvy v obrázku očekáváme. Jak tedy poznáme, co je ve skutečnosti na obrázku za barvu? Ve Photoshopu se k tomu používá standardní nástroj Info-paleta (obr. 2.2). Bez ní je jakákoli barevná korekce obrázku nepřesná a ani zkušený kolorista se bez ní neobejde. Je skrytá pod ikonou (i) na panelu v pravé části. Po jejím rozkliknutí se otevře nové okno přes obrázek. To je celkem nevýhodné, Info-paleta se používá prakticky pořád a není důvod jí zabalovat. Bude tedy vždy zakrývat kus obrázku. V GUI by správně měla by být vždy viditelná a neměla by rušit.

Info-paleta zobrazuje potřebné informace o pixelu na pozici kurzoru. V její levé části vidíme jistá čísla u R, G a B. To jsou jednotlivé barevné kanály, červený, zelený a modrý, které dohromady určují barvu pixelu. Každý kanál pixelu je jedno číslo, standardně ukládané na 1 bajt, tedy nabývající hodnoty 0-255. Číslo 0 znamená, že pixel postrádá barvu kanálu, 255 znamená nejsytější možnou barvu. Fotoaparát nám dodává snímky právě v barevném prostoru RGB, fungují v něm monitory a zařízení produkující světlo. Není to ale jediná možnost reprezentace barvy pixelu. Podívejme se na pravou část Info-palety, kde jsou hodnoty pixelu v prostoru CMYK. Je to prostor, užívaný v digitálním tisku, tiskárna potřebuje obrázek v tomto prostoru, aby určila, jak velkým množstvím barvy má být dané místo na papíru pokryto. Jednotlivé kanály jsou azurová, purpurová, žlutá a černá. Jsou taktéž ukládány každý do jednoho bajtu, jen není standardní je zobrazovat číselně, nýbrž procentuelně. Čím více procent, tím větší pokrytí danou barvou. Rozdíl je v míchání barev, pokud bude ve všech kanálech 0%, nebude se nic tisknout, papír tedy zůstane bílý a výsledkem bude bílá barva. V RGB třikrát 0 znamená černou, žádné světlo nebude svítit.

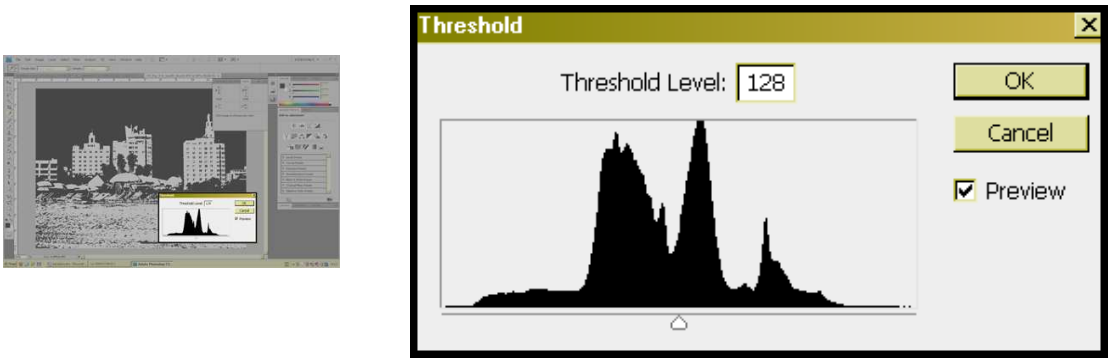




Obrázek 2.2 – Info-paleta

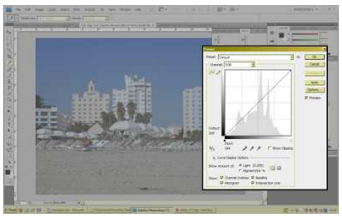
První úpravy

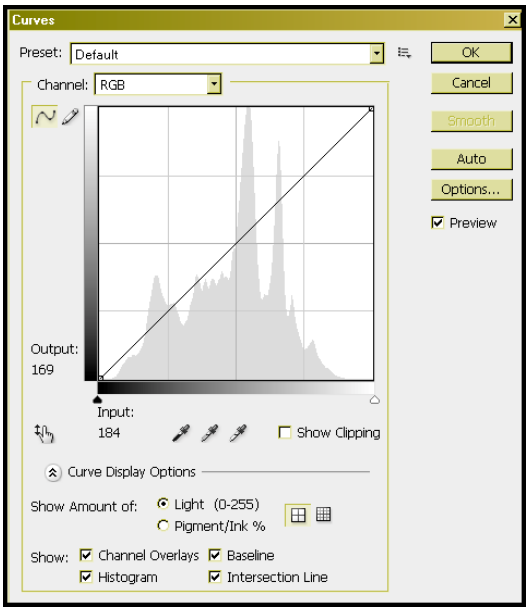
Ač jsme výše hovořili o různých postupech pro dosažení kvalitního výsledku, první krok je neměnný : vyrovnání černé a bílé. Nejčernější černá má hodnoty 0R 0G 0B, nejbělejší bílá 255R 255G 255B. Pokud mají pixely ve tmavých a světlých místech nevyrovnané hodnoty kanálů, je pravděpodobné že fotoaparát špatně nastavil světelné podmínky a obrázek vytvořil se zabarvením. Pokud rozsah mezi nejčernější černou a nejbělejší bílou je menší než 0-255, ochuzujeme tím i všechny barvy obrázku. Tak i např. nejintenzivnější modrá nikdy nebude tak intenzivní, jak by mohla být. Nižší rozsah jasů je běžný jev, fotoaparáty nastavují při pořízení snímku krajní body podle vlastního algoritmu, který neurčí, zda jsou body významové, či nikoliv. Stalo se tomu i v našem obrázku, ve Photoshopu přejíždíme kurzorem přes objekty, které by měly být bílé, slunečníky a hotel v pozadí, a hledáme, u kterých bodů zobrazuje Info-paleta největší hodnoty. Nikdy si ale nebudeme jisti, že jsme nějaké místo obrázku neminuli. Přesnější je, použít pro přesnou identifikaci krajních bodů nástroj Image->Adjustments->Threshold... (obr. 2.3) Ten má jediný posuvník. Pixely obrázku, jejichž celkový jas (vážený průměr hodnot tří kanálů) je větší než posuvníkem vybraná hodnota, jsou nad prahem a budou bílé, ostatní černé. Při posunutí posuvníku na krajní hodnoty zůstanou zvýrazněné pouze hledané krajní body, nejsvětlejší a nejtmavší. Tyto body je vhodné zafixovat, abychom je měli vždy na očích a viděli, jak se při korekcích budou měnit. Klikneme tedy na ně za držení klávesy Alt a jejich hodnoty se objeví v dolní části Info-palety. Zde zůstanou i poté, co kurzorem přejedeme na jiný pixel, i poté, co nástroj Threshold zavřeme. Změny nepotvrzujeme, jinak bychom přepsali data obrázku, nástroj Threshold sloužil pouze pro zjištění pozice kýžených bodů.



Obrázek 2.3 – nástroj Threshold (práh)

Pro samotné vyrovnání těchto bodů, a tedy úpravu dat obrázku, použijeme gradační křivky. Nástroj otevřeme z menu Image->Adjustments->Curves... (obr. 2.4). Máme k dispozici čtyři křivky, pro každý kanál R, G a B a jednu společnou, které vybíráme z rozbalovacího menu Channel. Mřížku pod ním nahlížejme jako soustavu souřadnic, kde osa x znamená původní hodnotu pixelu a osa y hodnotu novou. Tímto nástrojem tedy budeme měnit hodnoty pixelů obrázku na základě nakreslené křivky. Na začátku je vidět linka, která směřuje pod úhlem 45° z levého dolního rohu do pravého horního. To znamená, že v každém jejím bodě je stejná hodnota na ose x, jako na ose y. Nebyla tedy zatím provedena žádná změna. Matoucí je, kterým směrem hodnoty osy x a y rostou. Standardně je počátek v levém dolním rohu, je to hodnota 0, tedy černá nebo nepřítomnost barvy kanálu. V rozšířeném nastavení nástroje však můžeme přepnout do módu, kdy je v levém dolním rohu nejvyšší hodnota, tedy 255, a počátek v rohu vpravo nahoře. Toto zobrazení používá Margulis v celé knize, my však pro názornější představení nástroje použijeme jednodušší původní zobrazení. Kde v našem zobrazení je bílá (plný kanál) a kde černá (prázdný kanál) poznáme podle gradientů vlevo od mřížky a pod ní.





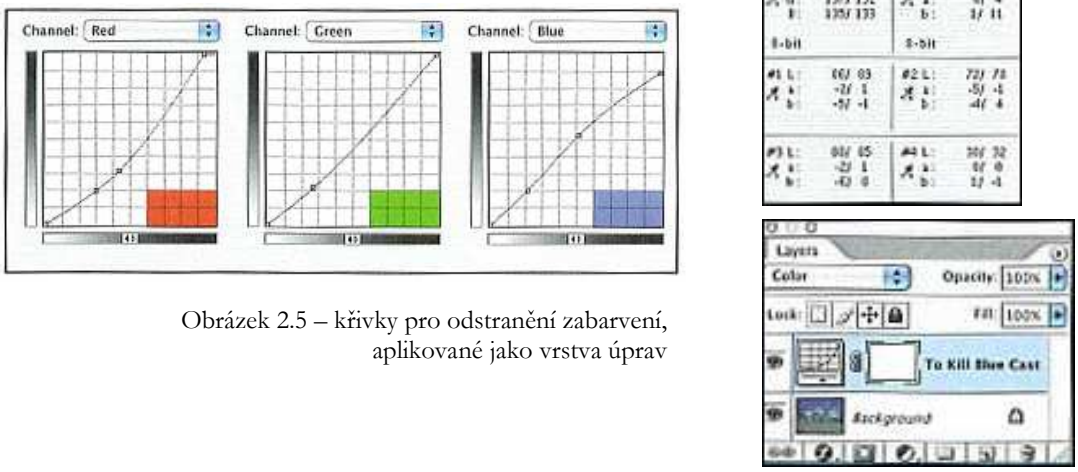
Obrázek 2.4 – nástroj Curves (křivky)

Nyní přijdou k užitku naše body na Info-paletě. Nastavíme si pro jednoduchost kompozitní RGB křivku. Levý dolní znamená hodnotu 0, tedy černou. Pokud jím přetáhneme kousek doprava, změníme na 0 i pixely s hodnotou 1, 2, 3 atd. Zároveň se ve středu linie, tedy ve středních tónech, o něco snížily hodnoty, z 80 se stalo 78. Celkový obrázek tedy sníží jas. Pokud jsme v obrázku ale měli pixely s hodnotou 3 vedle pixelu s hodnotou 2 a oba jsou nyní 0, ztratili jsme původní informaci o jejich přechodu, detail. Proto jsme také nejdříve hledali nejčernější významový bod obrázku, abychom měli jistotu, že pokud změníme jeho hodnotu na 0, nevynulujeme tím žádné další významné pixely. Na Info-paletě vidíme, jak náš nejčernější bod mění všechny své tři hodnoty. Vrátíme tedy bod na původní polohu vlevo dole a změníme křivku. Z rozbalovacího menu na kanál R. Při pohybu týmž bodem vidíme že se mění pouze jeho červené složka. V každém kanálu tak posuneme levý dolní roh doprava tak, abychom v Info-paletě viděli u našeho nejčernějšího bodu hodnoty velmi blízké 2R 2G 2B. Je vhodné si nechat rezervu pro případ, že jsme přes naše důkladné hledání stejně nějaký tmavější bod neminuli. Totéž provedeme pro nejbělejší bod – pohneme v každém kanále s bodem v pravém horním rohu doleva, než dosáhneme 253R 253G 253B. Zaškrtávacím tlačítkem Preview přepínáme mezi původním a změněným obrázkem, a při porovnání tak už můžeme vidět chyby původního obrázku snadněji. Stejně jako jsme upravili černou a bílou, se snadno upravuje také šedá, ta je také neutrální, tzn. má hodnoty ve všech kanálech vždy stejné. Na tomto obrázku si však nemůžeme být jisti, které objekty byly ve skutečnosti šedé. Potvrdíme tak pouze vyrovnání bílé a černé klikem na OK.

Data obrázku se nyní změnila. Na Info-paletě již vidíme pouze nové hodnoty a ne ty, které měl obrázek před korekcí. Na ten se můžeme vrátit, díky tomu, že si Photoshop uchovává historii. V ní jsou však pouze data obrázku, nemůžeme vidět, jakým způsobem jsme obrázek upravovali, jaký filtr jsme použili, natož kam jsme posunuli krajní body křivek. Pokud bychom nyní obrázek uložili a restartovali Photoshop, historie by byla vymazána a původní obrázek bychom již nemohli získat. Provedli jsme *destruktivní* barevnou korekci, která přes svůj název je ve Photoshopu regulérním a používaným

postupem. Všechny filtry, které vybereme z menu stejně, jako jsme vybrali křivky, jsou aplikovány destruktivně.

Můžeme také použít jiný přístup. Photoshop pro každý obrázek vytváří zásobník vrstev. Vrchní vrstvy vždy překrývají spodní. Do tohoto zásobníku vrstev můžeme přidat kromě obrázku také celý nástroj se všemi jeho parametry. V tom případě hovoříme o vrstvě úprav. Vrátíme se tedy tlačítkem Edit->Back na původní obrázek. V pravém panelu Photoshopu vidíme záložku Layers.V její dolní části je ikonka pro přidání nové vrstvy úprav. Z rozbalovacího menu vybereme opět Curves. Křivky byly přidány nad aktuální vrstvu. Parametry nástroje nyní nenastavujeme v samostatném okně, ale v záložce Adjustments v pravém panelu nad záložkou Layers. Zde provedeme tutéž úpravu, jako předtím. Pro odstranění modrého zabarvení použil Margulis tyto křivky:



Obrázek 2.5 – křivky pro odstranění zabarvení, aplikované jako vrstva úprav

Zafixoval k tomu nejdříve 4 body, které měly zůstat neutrální, nejspíše na hotelu a na slunečníku. Na Info-paletě si přednastavil, že jejich hodnoty mají být zobrazeny v barevném prostoru LAB. Ten má jednu jasovou složku L, udávanou v procentech, tedy s hodnotami 0-100, a dvě barevné složky A a B, obě od -128 do 127. Pro naše účely si postačí pamatovat, že pokud obě složky A i B jsou rovny 0, pixel je neutrální, žádná barva nedominuje. Prostor je tak vhodný pro vyrovnávání barev, v RGB jsme museli mít všechny tři hodnoty stejné, ale nevěděli jsme jak vysoké. Při vyrovnávání se tak měnil jas, v LAB jas (L kanál) zůstává nezměněn. První zafixovaný bod má hodnoty A(-2) B(-5), tedy zabarvení do azurova, kterého se snažíme zbavit. Hledáme takové křivky, při kterých všechny fixované body budou co nejblíže nule. Jak jsme ale naznačili, to, že zafixované body jsou ve skutečnosti neutrální a že hotel nemá např. jen lehce azurový nátěr, není ihned prokazatelné. Zde však z Info-palety vyčteme, že všechny body mají totéž zabarvení, a jelikož jsou na různých objektech, je pravděpodobnost velká, že zabarvení bylo způsobeno chybou fotoaparátu.

Zajímavá je modrá křivka. Zkoumejme pravý horní roh, který je posunutý o něco dolů. Nezapomeňme, že nyní je 0, tedy počátek soustavy souřadnic, vpravo nahoře. To znamená, že hodnotu modrého kanálu těch pixelů, které ji mají poblíž nuly, o něco zvýšíme. Tam, kde modrá zcela chybí ji tak přidáme. Do stínů, a také např. do stromů, které jsou zelenožluté a tedy mají velké hodnoty kanálu R a G, ale malé u B. Bod ve středu křivky je posunut o trochu nahoru, proto ve středních tónech modrou ubíráme. Tam spadá většina obrázku a proto také Margulis pojmenoval naši vrstvu To kill a blue cast.

Režimy prolnutí

Po vyrovnání barvy se v některých částech obrazu ztratil detail. Křivky totiž nemění jen barvu, ale také kontrast obrázku. Místa, která jsou na křivce ve strmé části, budou mít po korekci větší rozdíly mezi sousedními pixely, tedy viditelnější přechody, tedy větší kontrast. Tam, kde křivka klesá pozvolně, v úhlu menším než 45°, bude také kontrast menší než původně. To se stalo v našem případě, všimněme si levé části červené a zelené křivky a pravé části modré. Tedy všude tam, kde je mnoho červené a zelené, což jsou stromy a všechny objekty bílé barvy, tedy ty nejdůležitější na snímku.

Photoshop nabízí možnost použít z křivek pouze jejich barevnou informaci a neaplikovat změny v kontrastu. Musíme k tomu změnit režim krytí mezi naší vrchní vrstvou, tedy obrázkem po úpravě křivkami, a původním obrázkem na spodní vrstvě. Na obr. 2.5 vidíme ihned pod záložkou Layers rozbalovací menu, kde Margulis vybral režim Color. To je právě zmíněný režim krytí, který změní pouze barvu. Standardně je pro každou novou vrstvu nastaven režim Normal, který zkrátka prolne obrázek z horní vrstvy s tím z dolní bez použití speciálního algoritmu. Vedle režimu prolnutí nastavujeme také procento krytí (Opacity). Zde upravujeme, v jakém poměru se mají použít data z vrchní k datům ze spodní vrstvy. V našem případě jsme ponechali 100%, čili kompozitní obrázek bude používat pouze barevnou informaci vrchní vrstvy, která byla upravena křivkami, jasová informace se v režimu Color bere vždy ze spodní vrstvy a krytí ji neovlivní. Režimy krytí se pro barevnou korekci používají pouze v obrázku, softwary pro barevnou korekci videa nutí uživatele, aplikovat veškeré úpravy najednou v jedné vrstvě.

Další fází bude odstranění šumu z oblohy. Přiblížíme-li obrázek kolečkem myši za stisku klávesy Alt, uvidíme jej zřetelněji. Abychom se šumu zbavili, musíme zjistit, ve kterém kanále je nejvíce chyb. V záložce Channels si jednotlivé kanály zobrazíme a uvidíme, že v červeném kanále je šumu zdaleka nejvíce. Problém je obecně známý, fotoaparáty produkují v obloze nejvíce chyb právě zde. Navíc je to právě kanál, který je pro detail rozhodující. Obloha je modrá, pixely tak budou mít v modrém kanále vysoké hodnoty, v zeleném střední. Kdybychom mohli nějak modrý i červený kanál přiblížit zelenému, detail by byl vyrovnaný a zároveň bychom odstranili šum. Druhá stránka věci je, že bychom celý obrázek barevně posunuli, obloha by pak byla skoro šedá, zelený, modrý a červený kanál by si byly blízko. Chceme opak toho, co jsme udělali v předchozím kroku, zachovat informaci o změně jasu a zahodit informaci o barvě. Nabízí se tak nová vrstva a speciální režim Luminosity. Vytvoříme novou prázdnou vrstvu a v menu vybereme Image->Apply Image... Tento nástroj slouží ke kopírování určitého kanálu nebo vrstvy. Jako zdrojový vybereme zelený kanál z nejspodnější vrstvy, jako cílový všechny kanály (RGB kompozit) nově vytvořené vrstvy. Po potvrzení vidíme na obrazovce pouze zelený kanál obrázku původního. Je to tím, že nová vrstva byla nastavena na režim Normal, a poté, co jsme do ní zkopírovali zelený kanál, překryla původní obrázek ve spodní vrstvě. Změníme tedy mód na Luminosity a vidíme opět vše v barvě. Možná jsme nepostřehli změnu, po kliku na ikonku oka vlevo od nové vrstvy ji můžeme na okamžik vypnout, tj. zobrazit obrázek, jak vypadal před jejím použitím. Opakovaným klikáním na ikonu oka snadno oba výsledky porovnáme, a odstranění šumu tak bude zjevné.

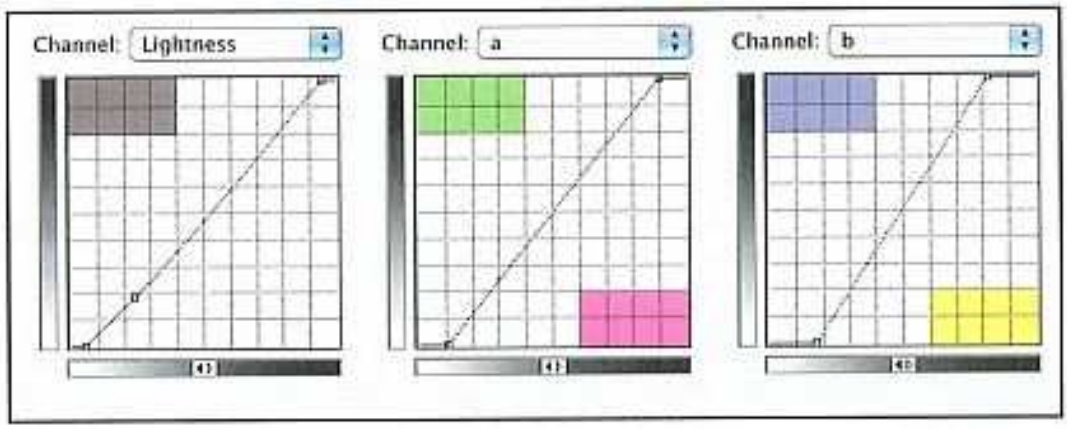
LAB křivky

Nastavili jsme tedy správně bílou a černou a eliminovali jsme šum i zabarvení, všechny objekty na obrázku by nyní měly mít stejný odstín, jako ve skutečnosti. Přesto jsou barvy poměrně mdlé. Nyní dodáme barvám sytost a podobné barvy oddělíme více od sebe. Je to další akce ke které je vhodný barevný prostor LAB. Zatím jsme jej poznali

z Info-palety, kde jsme si nechali zobrazit, jak by v tomto prostoru vypadaly hodnoty pixelu. Data obrázku však stále zůstávala v prostoru RGB. Nyní ale potřebujeme obrázek v prostoru LAB upravovat a to už Photoshop neumožňuje za současného uchování originálních dat. Obrázek je nutno nejdříve do prostoru LAB překonvertovat. Každý pixel obrázku bude přepočítán do nových souřadnic. Před konverzí je tak nutno potvrdit úpravy ve všech vrstvách, vypočítat jeden RGB obrázek, který se teprve bude konvertovat. Při tom ale ztrácíme informaci o úpravách, které jsme na vrstvách provedli a výsledek je tak stejný, jako bychom používali pouze destruktivní barevné korekce. To je velké omezení Photoshopu. Pokud kdykoliv v průběhu postupu korekce budeme potřebovat provést úpravy v jiném barevném prostoru, už si nemůžeme celý tento postup od začátku do konce zaznamenat. Po konverzi se nemůžeme vrátit k původním RGB křivkám a pohnout s některým bodem. Dikobraz musí tento problém řešit, neboť ve videu celý postup zaznamenaný mít musíme, data se zde přepočítávají pouze jednou, na konci práce.

Zcela konkrétně pro náš obrázek: Klikneme na nejspodnější vrstvu pravým tlačítkem a vybereme Flatten Image. Data obrázku tak byla přepočítána z všech vrstev a zbyla pouze jediná. Nyní změníme barevný prostor Image->Mode->LAB Color. Data byla přepočítána znovu, tentokrát už do nových souřadnic. V záhlaví obrázku vidíme také vidíme LAB, jako aktuální barevný prostor. Otevřeme křivky, stačí přes menu, nebudeme používat režimy prolnutí, a tak můžeme provést destruktivní úpravu. Na výběr máme nyní křivky L, A a B. Také v menu jsme si všimli, že některé nástroje již nemůžeme použít. Photoshop totiž odvozuje jak položky menu, tak jednotlivé nástroje z barevného prostoru aktuálního obrázku. Pokud máme otevřené dva obrázky, jeden v RGB, druhý v LAB, bude se po překliknutí z jednoho na druhý měnit hlavní menu celého programu a také význam jeho položek, což je chyba v designu a člověk, který na Photoshop není zvyklý, je tak zmaten.

Křivky nastavíme podle obrázku. Fungují také jinak, než v RGB. V L křivce máme celý rozsah jasů obrázku, objekty v ní bývají od sebe odděleny a my je tak můžeme snadno zesvětlit, ztmavit, nebo jim měnit kontrast podle strmosti křivky. Již bylo řečeno, že A a B křivky mají hodnoty v rozsahu od -128 do 127, uprostřed křivky je tedy 0, neutrální barva. Protože jsme v předchozím kroku při RGB křivkách neutrální barvy správně nastavili, nechceme s nimi hýbat. A i B křivka tak musí procházet přesně středem. Také nyní není na jedné straně křivky barva plná a na druhé prázdná, jak by tomu naznačovaly gradienty vedle mřížky. Na obou krajích, vlevo dole i vpravo nahoře jsou lokalizovány ty nejzářivější barvy, kanál A jde od zelené k purpurové, kanál B od modré ke žluté. Většina obrázku je vždy koncentrována kolem středu křivky, bodu 0. Velmi sytých barev je na fotografiích málo. Pokud pohneme oběma konci křivky o stejnou hodnotu, zachováme neutrální 0 a prozáříme barvy, které se od ní liší. Pokud např. měly stromy barvu trochu do zelena, budou zelenější.



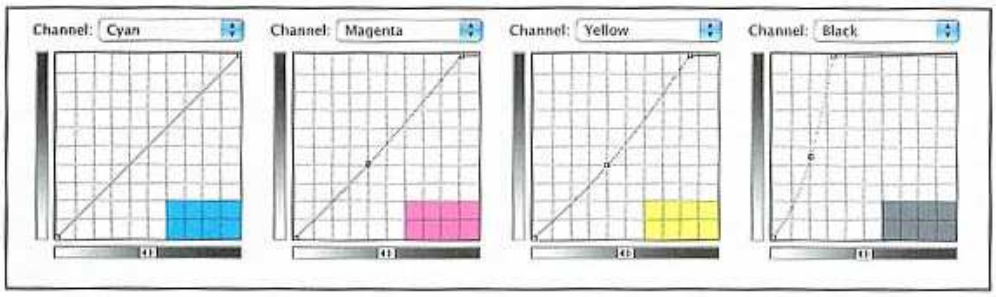
Obrázek 2.6 – LAB křivky pro zvýšení sytosti barev

Rozmazání a ostření

Z oblohy jsme již odstranili šum, Luminosity mód nám zajistil, že rozdíly v jase byly eliminovány. Barva však byla ponechána a v ní rozdíly přetrvaly. Jak odstranit také barevný šum? Nejjednodušší je použití filtru pro rozmazání. Chceme ale rozmazat pouze oblohu, ostatní části fotografie musí zůstat zachovalé. Postupujeme takto : Zduplikujeme vrstvu a obrázek rozmažeme např. filtrem Gaussian Blur z nabídky Filter. Filtry pro rozmazání zprůměrují pro každý pixel hodnoty v jeho okolí vynásobené určitou váhovou funkcí, největší váhu budou mít nejbližší body. Tím jsme ale rozmazali celý obrázek. Nyní chceme, aby bylo rozmazání aplikováno pouze na oblohu. Protože je jednolitě modrá a žádný jiný objekt fotografie modrý není, můžeme zkrátka nastavit, že horní rozmazaná vrstva se má s dolní prolnout pouze tehdy, pokud v dolní vrstvě je pixel modré barvy. Poklepáním na horní vrstvu se zobrazí okno s jejími vlastnostmi a v dolní části Blend If nastavíme v posuvníku B kanálu (modrá-žlutá) rozsah barev pouze do jeho levé části.

Po zploštění obrázku do jediné vrstvy (Flatten Image) přejdeme k ostření. Ostření opět bere v potaz okolní body každého pixelu. Cílem je zvýraznit hrany, zesvětlit na nich světlé části a ztmavit tmavé. Také je přidána kontura, obrys objektu. Opět duplikujeme vrstvu, a pak na ni aplikujeme ostřící filtr, standardně Filter->Sharpen->Unsharp Masking... Než hledat hned na začátku nejlepší možné parametry filtru, je lepší přeostřit a posléze jen upravovat procento krytí vrstvy. Jelikož v LAB jsou problémy s režimy krytí, které chceme použít pro oddělení ostření světel a stínů, je vhodné překonvertovat obrázek do prostoru CMYK. Ten potřebujeme jako výstupní formát pro tisk, proto poslední úpravy provedeme právě tam. Přepneme tedy Image->Mode->CMYK Color. Nejvrchnější přeostřenou vrstvu opět zduplikujeme. Chceme ostřit tmavá místa více, než světlá. U vrchní vrstvy tak nastavíme režim prolnutí Lighten s krytím např. 50%, u spodní Darken s krytím 90%. V režimu Darken je vrchní vrstva prolnuta pouze pokud je pixel tmavší, než ve spodní vrstvě, u Lighten naopak.

Posledním krokem už je jenom zkontrolovat, zda je obrázek vskutku připravený pro tisk, zda jsou světla vyrovnaná, zda jsou střední tóny bez zabarvení, ale hlavně, zda nejtmavší body nepřekračují maximální množství inkoustu, který je papír ještě schopen snést. Součet všech složek CMYK by neměl přesáhnout 300%. Margulis vytvoří níže uvedenými křivkami černý bod o hodnotách 80C 70M 70Y 85K, což dává celkově 305% a je ještě v tolerovatelné mezi. Nedivme se, že hodnota jednotlivých složek není vyrovnaná, typický azurový pigment není tak čistý, jako ostatní barvy a pro dosažení neutrálního odstínu ho musí být vždy o něco více než složek M a Y. Složka K je černý inkoust, který barvu neovlivní.



Obrázek 2.7 – CMYK křivky pro poslední úpravy před tiskem

Obrázek 2.8 – Vlevo původní obrázek, vpravo obrázek po provedení všech korekcí
(oba byly naskenovány z [Margulis], jsou proto v nízkém rozlišení a slouží spíše k ilustraci rozdílů)

2.1.2 Adobe Photoshop Lightroom

Na rozdíl od Photoshopu, průmyslového standardu, který integruje všechny nástroje pro zpracování obrazu, je Lightroom jednodušší a byl navržen s cílem usnadnit práci profesionálnímu fotografovi. Umožňuje správu velkého množství fotografií, jejich vyhledávání, hodnocení, komentování či prezentování. Photoshop při otevírání obrázků zobrazoval např. jen malý náhled, ze kterého fotograf těžko určil nejkvalitnější fotografii ze série, a musel se proto pro prohlížení fotografií používat externí program. I pro fotografa zůstává samozřejmě Photoshop nezastupitelný a i Lightroom počítá s používáním obou programů naráz. Nás však Lightroom zajímá tím, že umožňuje také základní barevné korekce.

Obrázek 2.9 – Lightroom - Katalog fotografií

Jako příklad budeme upravovat tentýž soubor, jako ve Photoshopu. Nejdříve soubor naimportujeme z CD. Lightroom spravuje fotografie formou katalogu. Pouze odkazuje na místo, kde jsou soubory skutečně umístěny. To může být i externí zařízení jako CD, USB-disk nebo karta fotoaparátu. Na disku se vytvoří pouze náhled souboru a jeho metadata. Pokud zrovna není paměťové zařízení připojeno, Lightroom vydá varování, že se díváme pouze na náhled a nebude nám např. také umožněno provést barevnou korekci.

Ta se provádí v modulu Develop. Všechny použitelné nástroje vidíme pod sebou na panelu v pravé části. Jsou zde posuvníky pro jas, kontrast, křivky, posuvníky pro korekci černobílé fotografie, odstranění šumu, ostření. Vše je integrované, nemůžeme změnit pořadí, v jakém budeme nástroje aplikovat, můžeme je maximálně pro úschovu místa na obrazovce skrýt. Jsme tak omezeni pevně daným postupem úprav, přičemž každá může být provedena pouze jednou. Všechny úpravy jsou ale nedestruktivní a můžeme tak vždy na přeskáčku měnit parametry jakéhokoliv nástroje na panelu. K dispozici jsou tlačítka pro přepínáním mezi původním obrázkem a obrázkem po korekci. Oba si lze také zobrazit vedle sebe.

Obrázek 2.10 – Lightroom - Modul Develop pro barevné korekce

Vpravo nahoře vidíme histogram s RGB kanály, po Info-paletě další používaný analyzační nástroj. Souřadnice x je hodnota kanálu, souřadnice y procento pixelů obrázku s touto hodnotou. Z barvy plochy histogramu musíme správně přečíst kanál, ke kterému se souřadnice vztahuje. U červené, zelené nebo modré je to triviálně kanál R, G a B. Žlutá platí pro oba kanály R a G, purpurová pro R a B a azurová pro G a B. Šedá plocha histogramu znamená, že pixel s touto hodnotou je v míře dané souřadnicí y zastoupen jak v kanálu R, G, tak i B. Histogram budeme muset použít jako vodítko pro vyvážení bílé a černé. Info-paleta z Photoshopu je zde redukována pouze na zobrazení R, G a B hodnot pixelu, pod kurzorem. Není možné zafixovat si důležité body. Nemůžeme ani překonvertovat obrázek do jiného barevného prostoru, ani v jiném barevném prostoru odečítat hodnoty pixelu.

Ve Photoshopu jsme vyrovnávali neutrální barvu i rozsah jasů pomocí křivek. V Lightroomu to nelze provést, k dispozici je jen jedna kompozitní křivka, a ani v ní nelze

pohnout s krajními body, slouží pouze pro úpravu kontrastu. Pro vyrovnání neutrální barvy vybereme v Lightroomu nástroj White Balance, připomínající kapátko z Photoshopu. Klikneme jím na pixel, o kterém víme, že má být neutrální. Podle toho, jak daleko od neutrální barvy skutečně je, odhadne Lightroom automaticky, do jaké barvy je zabarven celý obrázek, a toto zabarvení eliminuje. Pro nastavení rozsahu jasů pak Lightroom poskytuje své vlastní posuvníky. Cílem je roztáhnout histogram od jednoho konce osy x do druhého. Zároveň nesmí být při těchto krajích pixelů příliš moc, neboť to znamená ztrátu detailu. Exposure zvětšuje expozici, tedy zesvětluje, posouvá histogramem doprava, Recovery vrací přesvětlené pixely zpět do reálného rozsahu jasů, podobně pro stíny fungují posuvníky Blacks a Fill light. Na histogramu vidíme v horní části dvě ikony s trojúhelníkem. Tato přepínací tlačítka zobrazují body, které se po korekci dostaly mimo povolený jasový rozsah, varovnou svítivou barvou. Při vyrovnávání černé a bílé je tak dobré je mít na očích a snažit se, aby těchto bodů bylo co nejméně. Vyrovnání černé, bílé a neutrálních barev jsme tak snadno provedli jinými nástroji, než ve Photoshopu.

Druhým krokem bylo odstranění šumu z oblohy. Nemůžeme nyní již provést stejný trik, jako ve Photoshopu, nemáme možnost podívat se na konkrétní kanál, natož ho kopírovat, nemáme vrstvy a nemáme režimy prolnutí. Můžeme však lokalizovat oblohu pomocí masky. Maska je vždy spojena s určitou barevnou korekcí a představuje vybranou část obrázku, na kterou bude tato korekce použita. Masku nakreslíme ručně pomocí štětce. Omezením Lightroomu je, že k masce nabízí jen několik základních posuvníků, ne tedy všechny, které jsou v modulu Develop k dispozici pro celý obrázek. Nejblíže našemu snažení by byl posuvník Sharpness, který v negativních hodnotách rozmazává. Kdybychom prováděli redukci šumu na celém obrázku, nabízí nám k tomu v nástroji Detail přímo speciální nástroj Noise Reduction s oddělenými posuvníky pro šum jasový i barevný.

Pro zvýšení sytosti barev nabízí Lightroom jako alternativu ke křivkám v prostoru LAB posuvníky Saturation a Vibrance. Saturation funguje podle standardizovaného modelu HSV, tedy odstín, sytost a jas čistě matematicky. Pro fotografie se tak vždy nehodí. Posuvník Vibrance oproti tomu zachovává i při zvýšené sytosti přírodní barvy, a je tedy vhodnější. Pro ostření má Lightroom nástroj, který kopíruje ten z Photoshopu. Můžeme ale opět ostřit jen jednou, a to celý obrázek. Ostřit jinak světla a jinak stíny nemůžeme.

Výsledkem tedy bude nepochybně lepší obrázek, než byl původní, ale těžko srovnatelný s obrázkem, upraveným ve Photoshopu. Zatímco tam jsme připravili obrázek ve formátu CMYK pro tisk, v Lightroomu jsme neopustili prostor RGB. Výhodou však je, že máme zaznamenané všechny parametry všech nástrojů, a můžeme tak kterýkoliv kdykoliv měnit nebo nastavení uložit a použít na jakoukoliv jinou fotografii. V praxi jsou nedestruktivní úpravy výhodnější. Zákazník dodá fotografie, kolorista je opraví, zákazník si je prohlédne a v případně nutnosti s koloristou provede opravy barevných korekcí. Ve Photoshopu to není možné, máme jen data výstupního souboru.

2.2 Barevné korekce ve videu

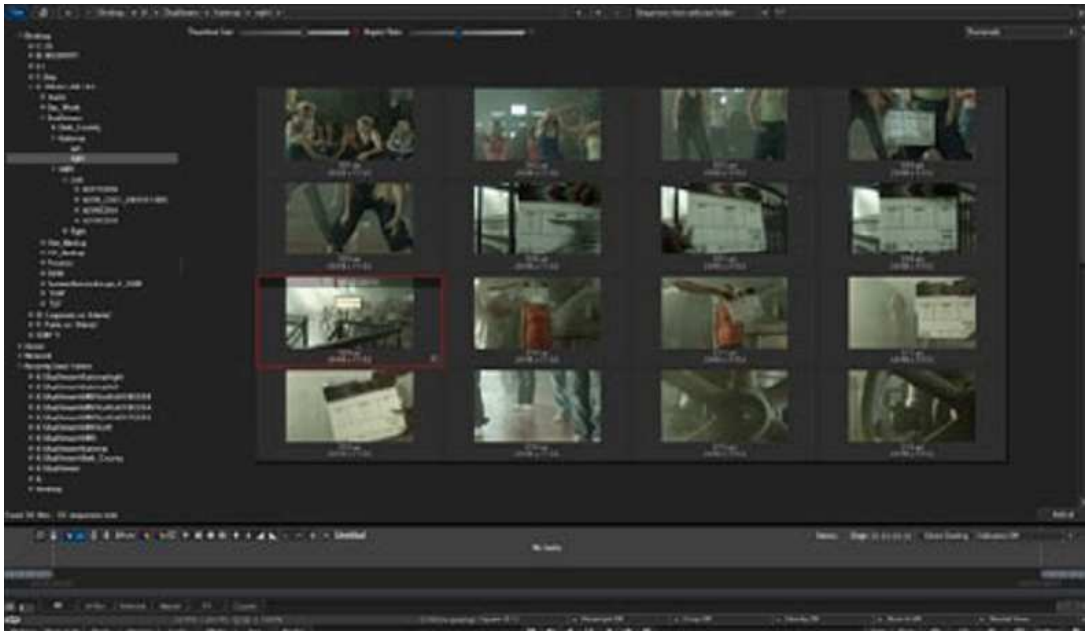
Pokud bychom na filmový projekt chtěli nahlížet vodopádovým modelem, byly by nejdříve natočeny všechny záběry dle technického scénáře, poté by film poputoval do střižny a následně do postprodukce. Vše je ale nelineární proces, na projektu se pracuje simultánně na různých místech. Film neupravuje jeden kolorista, ale celý tým. Za výslednou náladu filmu je však zodpovědný režisér, proto prvotní představu o barvě stanoví právě on již ve fázi natáčení.

Ve videu jsou barevné korekce ještě potřebnější než v obrázku, neboť divák má vždy srovnání s předchozím záběrem a pokud je mezi oběma jen jemná změna barvy, oko ji postřehne. Barva nesmí být prvkem, který ruší kontinuitu záběrů, které k sobě významově patří. U obrázku takové srovnání nemáme a laik si tak drobné chyby nevšimne. Základní barevné korekce lze provádět již ve střihacích softwarech jako Avid či Adobe Premiere. My se však zaměříme na profesionální postprodukční softwary, určené pouze k úpravě barvy, Iridas Speedgrade a Autodesk Lustre. Další používané softwary pro barevnou korekci videa, jako Quantel Pablo a Synthetic Aperture Color Finesse, fungují na obdobných principech, a jejich rozhraní proto detailně zkoumat nebudeme.

2.2.1 Iridas Speedgrade

Představme si, že obr. 2.1, který jsme upravovali, byl jeden snímek celého videa z pláže. Do postprodukce nyní dorazil sestříhaný projekt. Spustíme tedy Speedgrade. Největší část obrazovky zabírá Browser, správce souborů s náhledy, pod ním je časová osa. Procházíme disk a nalezneme složku s naším projektem. Celý projekt sestává zatím z dlouhého video souboru a dvou textových souborů : EDL souboru, a LOOK souboru. EDL soubor obsahuje informace o tom, jak má být ono dlouhé video sestříháno, LOOK soubor informace o parametrech prvotní korekce. Ta byla provedena v programu Speedgrade OnSet již v produkci, nebo dokonce preprodukci a od ní pak kolorista v postprodukci začíná svou práci.

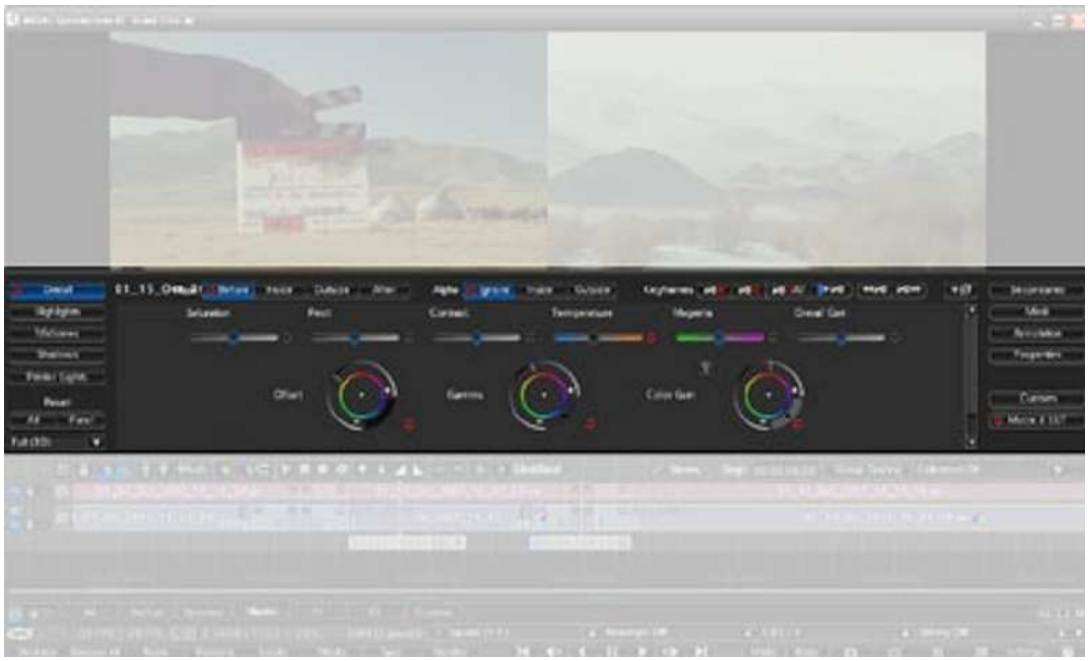
V Browseru tedy nejdříve přetáhneme video soubor na časovou osu. Přes nově vytvořenou vrstvu na časové ose přetáhneme EDL soubor, čímž původní dlouhý soubor rozstříháme na jednotlivé záběry. EDL soubor v Browseru dále přetáhneme na LOOK soubor. Nad stopou s videem se tak přidá nová stopa pro barevné korekce, stejně rozstříhaná, jako stopa videa pod ní. Nad každým záběrem tak máme možnost podle libosti změnit původní korekci z LOOK souboru.



Obrázek 2.11 – Speedgrade - Browser s náhledy na videa, pod ním časová osa

Primární korekce

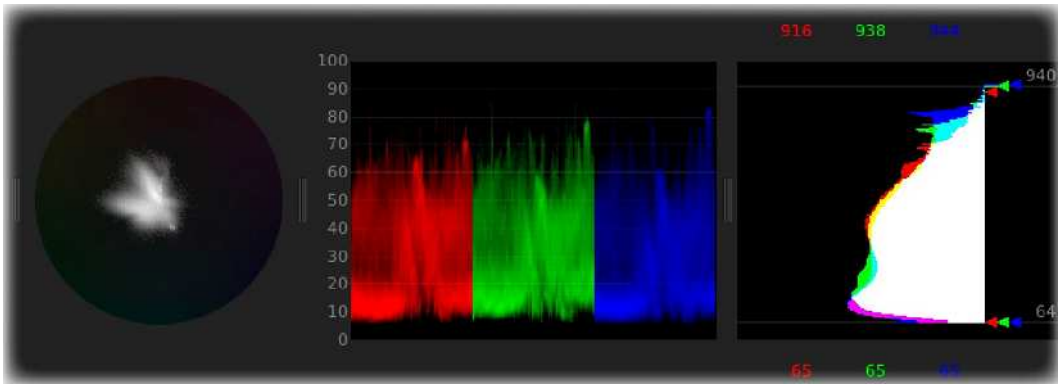
Na časové ose máme nyní již celý film a nepotřebujeme dále prohlížet disk, uzavřeme tedy Browser, a zobrazí se nám tak viewport, který Browser dosud zakrýval. Viewport zobrazuje snímek, na který je nastaven ukazatel časové osy. V kontextovém menu nastavíme, aby viewport zobrazoval zároveň vedle sebe původní i korekcí upravený snímek. Na časové ose vybereme první záběr a po kliknutí na ikonu korekcí (tři prolnuté RGB barvy) se nám otevře panel pro samotné úpravy.



Obrázek 2.12 – Speedgrade - Panel pro barevné korekce

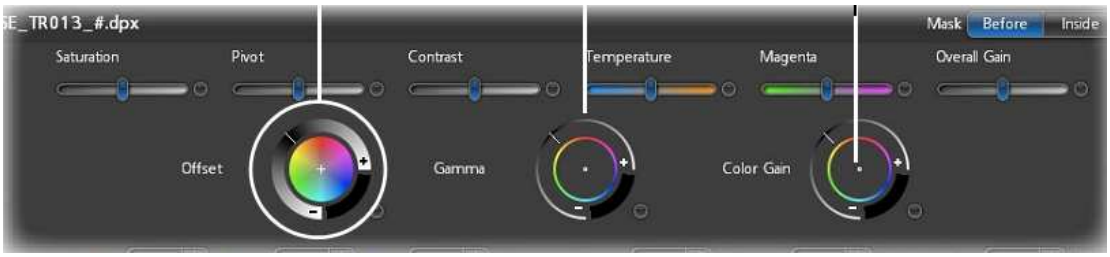
Jednotlivé položky menu na levé a pravé straně panelu mění nástroje v jeho centrální části. Jako vizuální informace pro to, u které položky již byla provedena korekce, je vedle ní zobrazen červený kruh. Podobně jako v Lightroomu jsou všechny nástroje integrované na témže panelu, v menu Custom však lze ještě nabídku nástrojů rozšiřovat pomocí pluginů.

První krok je tentýž jako v obrázku: Nastavení černé a bílé, vyrovnání neutrální. Ve videu se však liší nástroje pro analýzu snímku. Info-paleta se nepoužívá. Místo ní Speedgrade nabízí histogram, ale hlavně nástroje Vectorscope a Waveform. Bližší vysvětlení obou těchto nástrojů ponechme stranou, pro naše účely postačí konstatovat, že slouží k vyrovnání barev snímku, profesionální kolorista je na ně zvyklý a není bez nich schopen pracovat. My budeme barvu kontrolovat již známým histogramem.



Obrázek 2.13 – Analyzační nástroje ve videu : zleva Vectorscope, Waveform a histogram

Jsme v nabídce Overall, čili budeme měnit barvu na celém obraze. Vidíme, že centrální část zabírají barevné kruhy Offset, Gamma a Color Gain. Barevné kruhy kopírují rozhraní hardwarové konzole, kterou kolorista video většinou upravuje. Kolorista se díky ní může dívat na obrázek a poslepu hýbat s trackbally. Každý barevný kruh má dvě části. V jeho středu měníme barevný tón, posuvníkem na jeho okraji intenzitu. U kruhu je také červené tlačítko Reset pro nastavení na původní hodnoty. Kruh Offset upravuje černý bod. Pokud na jednom z analyzačních nástrojů zjistíme, že nejtmavější body mají zabarvení do žluta, v barevném kruhu Offset posuneme střed směrem ke komplementární modré. Poté posuvníkem na okraji nastavíme, aby nejmenší možnou hodnotu jasu měl skutečně nejtmavější významový bod. Poté budeme stejným způsobem nastavovat bílou pomocí kruhu Color Gain. Kruh Gamma slouží k vyrovnání neutrální šedé a k nastavení kontrastu ve středních tónech. Zde máme zabarvení do azurova, střed kruhu tak musíme posunout směrem k červené. Upravili jsme tak jasový rozsah na celém obrázku. Barevnými kruhy však můžeme obrázek upravovat i v jeho jednotlivých jasových rozsazích, ve světlech, středních tónech i stínech. Příslušný bychom vybrali v nabídce vlevo, v našem případě to není potřeba.



Obrázek 2.14 – Barevné kruhy Offset, Gamma a Color Gain

Sekundární korekce

Druhým krokem je odstranění šumu z oblohy. Zde se jedná o *sekundární korekci*, tzn. v terminologii videa, že bude provedena pouze na části obrazu, ne na celém jako korekce primární. Sekundární korekci je možné provádět buď na vybraném objektu, který obkreslíme pomocí vektorové masky, nebo na všech pixelech určité barvy (té říkáme barva klíčová a celá korekce se pak nazývá klíčování). U oblohy bychom mohli postupovat oběma způsoby, klíčování by bylo vhodné protože všechny modré pixely jsou pouze a jenom na obloze. Překlikli bychom do panelu Secondaries. V levé části bychom nastavili barevný rozsah v prostoru HSV, kde H znamená odstín, S sytost a V jas. Na krajích všech tří posuvníků se nastavuje také mez tolerance, kde se bude korigovaný obrázek interpolovat s původním. Pro takto nastavený rozsah bychom stejně jako u primárních korekcí měnili Offset a Color Gain.

Obrázek 2.15 – Panel pro sekundární korekce, vlevo posuvníky pro nastavení barevného rozsahu, vpravo barevné kruhy a další posuvníky pro samotnou změnu barvy

My však použijeme vektorovou masku, abychom si také vysvětlili princip klíčových snímků. Mate zde trochu názvosloví, Speedgrade má na panelu jednu položku s názvem Secondaries, pod kterou se však skrývá pouze klíčování. Maska je taktéž z definice sekundární korekce, přesto je ve vlastní sekci Mask. V ní tedy vybereme jeden z tvarů a obkreslíme oblohu. Když jsme v Lightroomu jsme masku kreslili štětcem, jednalo se o bitmapu Ta je ve videu nevýhodná, místo ní nyní přidáváme vektorový objekt. Nejsme omezeni na základní geometrické tvary jako elipsa nebo obdélník, můžeme na objekt vždy přidávat další body a měnit jeho tak jeho tvar. Také můžeme přidat takových objektů hned několik. Po obkreslení oblohy je vhodné zjemnit přechod s nekorigovaným snímkem roztáhnutím vnějšího okraje masky.

Obrázek 2.16 – Izolování objektů pomocí vektorové masky

Takto přesně vybranou masku však lze použít právě jen na aktuálním obraze, pokud se kamera např. pohne směrem dolů, rozmazali bychom také budovy, neboť i ty by nyní do oblasti masky spadaly. Chceme mít v každém momentě po dobu celého záběru masku na správném místě. To nemusíme hned řešit tak, že budeme v každém snímku masku kreslit znovu (standardní frekvence ve videu je 25 snímků za sekundu, proto by to ani nebylo reálné). Stačí ji nakreslit jednou na začátku záběru, nastavit, jak má vypadat na konci záběru a ve všech snímcích mezi se bude pozice i případně tvar masky interpolovat. Nakreslili jsme tedy masku v čase 0s. Kliknutím na ikonu klíče přidáme klíčový snímek, kterým masku v tomto čase zafixujeme. Nyní přesuneme hlavní ukazatel na konec záběru. Maska zůstala pořád zobrazena na témže místě. Přesuneme ji opět na oblohu, okrajem vektorového objektu může klidně zasahovat mimo viewport. Opět přidáme klíčový snímek. Tím jsme zajistili správný pohyb masky v záběru. Stejně jako jsme přidávali klíčové snímky pro masku, lze také přidat klíčové snímky na nastavení primární i sekundární korekce. Na začátku záběru tak můžeme např. nastavit jiné zabarvení, než na konci. Přechod mezi nimi bude plynulý.

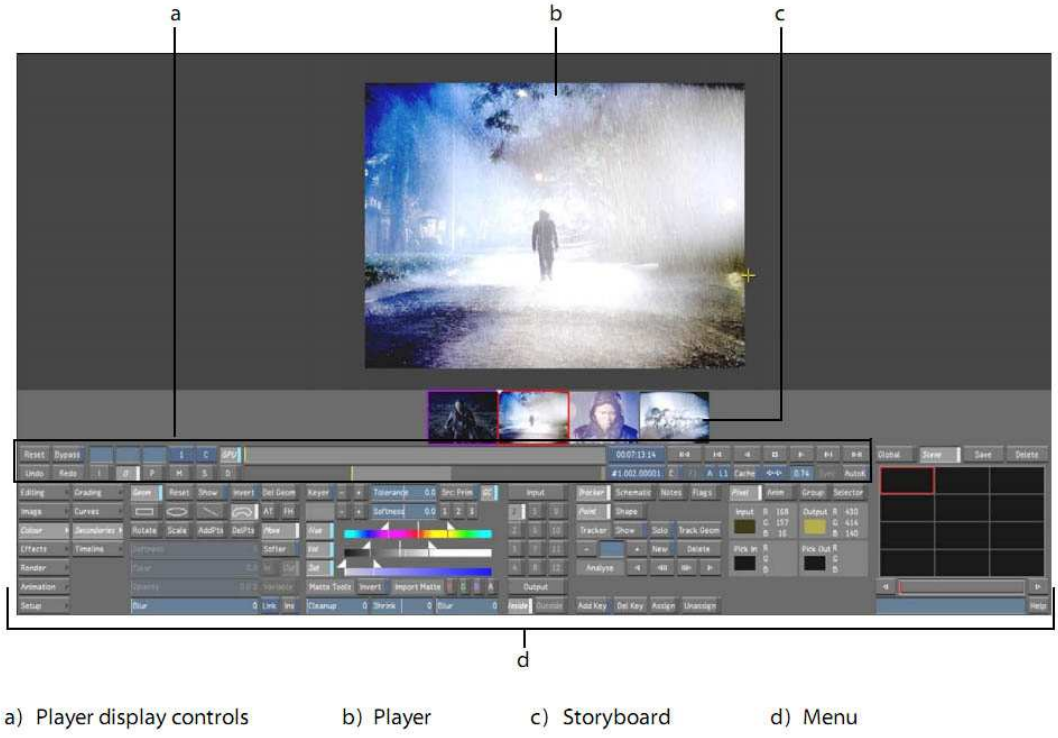
Masku, kterou jsme nakreslili na oblohou, chceme rozmazat. Speciální efekty, jako právě Gaussian Blur nabízí Speedgrade pod položkou Custom. Přidáváme je nad sebe do zásobníku, stejně jako vrstvy ve Photoshopu, standardní položka Base znamená výstup vrstvy po primární a sekundární korekci. Efekty tak tvoří jakousi třetí složku.

Speedgrade umožňuje měnit také měnit procento krytí, různé režimy prolnutí však nenabízí. Ve filtru Gaussian Blur nastavíme, že má být použit jen na masku. Nad něj do zásobníku přidáme ještě efekt Sharpen, který nyní použijeme na celý obraz. Tím by byl záběr upraven, a to stejně jako v Lightroomu bez toho, abychom změnili barevný prostor.

Nyní je nutno ještě tento záběr barevně srovnat se záběrem následujícím. K tomu není nutno po úpravách nového záběru vždy vracet ukazatel na záběr předchozí. Pohodlnější je, přidat si na časovou osu dva ukazatele (celkem je jich možné mít na časové ose až devět) a viewport nakonfigurovat tak, aby je vedle sebe zobrazil. Náhled tak budeme mít vždy na několik záběrů naráz. Poté, co takto upravíme všechny záběry filmu, vyrenderujeme výsledné video.

2.2.2 Autodesk Lustre

Rozhraní Lustru je již na první pohled odlišné od Speedgradu především tím, že postrádá časovou osu. V horní části je opět zobrazen viewport, pod ním tzv. Storyboard, pás s náhledy na první snímek jednotlivých záběrů výsledného videa, který slouží jako jednoduchá náhrada časové osy. Pod ním je panel, jímž se celý program ovládá. Zde však nastává zmatek. Naráz je zobrazeno velké množství tlačítek, které nejsou nijak vizuálně rozdělené do kompaktních bloků. Šetření místem na obrazovce je dotáhnuto do krajností. Tlačítka jsou pojmenována zkratkovitě a není jasné, zda se jedná o tlačítka standardní, zaškrtávací, přepínací nebo o položky menu. Všimněme si dvou sloupců tlačítek v levé dolní části. Položky sloupce druhého tvoří podmenu vybrané položky sloupce prvního. Po vybrání položky ve druhém sloupci měníme aktuální nástroj, tedy i rozvržení centrální části panelu. To nemusí být vždy jen nástroje pro barevnou korekci, ale také pro načítání souborů, střih, rendering. Jednou položkou je také časová osa. Připraveni tedy o možnost pracovat s více vrstvami nejsme, Lustre ale podporuje uživatele v používání Storyboardu, který je vidět vždy. Vlevo jsou tedy položky menu. To ale opět uživatel neví, tlačítka vypadají jako jakákoli jiná a trojúhelník v jejich pravé části vnímá spíše tak, že po jejich kliku by se mohlo zobrazit kontextové menu. Lustre má tedy velmi špatnou přístupnost. Větší váhu přikládal tomu, využít daný prostor maximálně, aby uživateli nabídl co největší počet nastavení na jedno kliknutí.

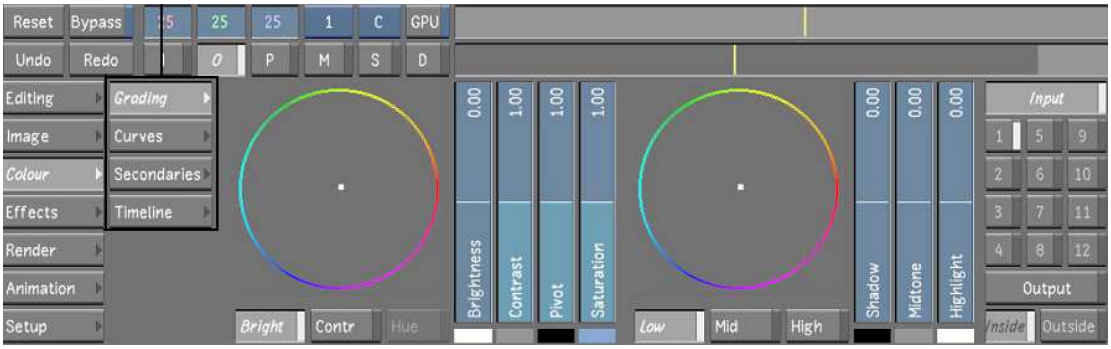


Obrázek 2.17 – Lustre - Uživatelské rozhraní

Nejdříve musíme do programu naimportovat naše vstupní soubory. Po kliku na Editing->Browse se horní část obrazovky změní na prohlížeč souborů, v jehož levé části je Shot Bin, který zobrazuje naimportované záběry. Lustre opět pracuje s odkazy, soubor se po přetáhnutí do Shot Binu nebude kopírovat. Vyhledáme tak naše video a přetáhneme jej sem. Video nyní chceme rozstříhat podle EDL souboru na jednotlivé záběry. V menu vybereme Edit->Assemble. Zde jsou zobrazeny EDL soubory z určitého pevně

stanoveného adresáře. Vybereme použitou frekvenci snímků a EDL soubor načteme. V něm jsou informace o tom, ke kterému video souboru je přiřazen. Pokud jej program nalezne v Shot Binu, přidá ho do Storyboardu (a tedy také na časovou osu) již rozstříhaný.

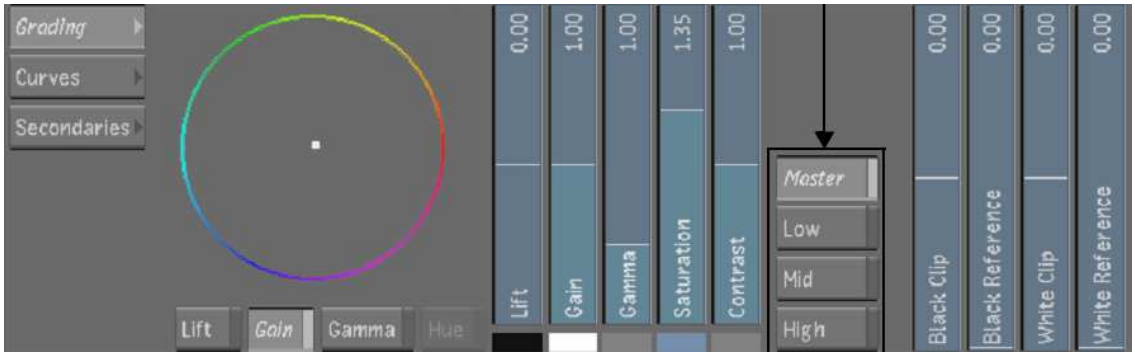
Přejděme tedy nyní k samotným barevným korekcím. V menu vybereme Colour->Grading. V jeho pravé části vybereme nejdříve, kde se má korekce aplikovat. Input znamená primární korekci přímo na datech vstupního souboru, Output znamená primární korekci před závěrečným renderingem videa. Všechna čísla mezi tím jsou položky, do kterých v jiném panelu vkládáme barevné rozsahy nebo masky. Pokud je číslu nějaký rozsah přiřazen, je zobrazeno bílou barvou, můžeme jej vybrat a provést na tomtéž panelu Grading sekundární korekci. Opět je to chování, které uživatel neočekává. Spíše se bude domnívat, že jednotlivá čísla znamenají stejně jako ve Photoshopu vrstvy. Porovnáme-li způsob provádění sekundárních korekcí se Speedgradem, je výhodou, že do menu Grading musíme překliknout jak pro klíčování, tak pro úpravu masky. Ve Speedgradu se takto muselo přepnout pouze pro masky a klíčování bylo prováděny vlastními barevnými kruhy Offset a Color Gain (kruh Gamma chyběl) v tomtéž menu, kde jsme vybírali barevný rozsah.



Obrázek 2.18 – Panel Grading v logaritmickém módu

Vraťme se k menu Grading. Tlačítka pod oběma barevnými kruhy jsou přepínací, celkem tedy máme k dispozici barevných kruhů pět. Barevné kruhy mají podobné rozhraní, jako ve Speedgradu, jen jsou od nich odděleny posuvníky. Není to opět příliš dobré řešení neboť kruh s některými posuvníky souvisí, s některými ne. Tak např. posuvníkem Brightness měníme celkový jas, v barevném kruhu Bright proporcionální zastoupení barev při tomto jasu. V pravé části měníme posuvníky shadow, midtone a highlight jas pro jednotlivé rozsahy, barevnými kruhy Low, Mid, High jim příslušné barvy. Low je proporcionální zastoupení barev k hodnotě posuvníku Shadow, nekonzistence opět vede ke zmatení. Barevným kruhem Contr měníme ve kterých barevných kanálech má být kontrastu více, na úkor kanálů ostatních, opět za konstantní hodnoty kontrastu, udané posuvníkem.

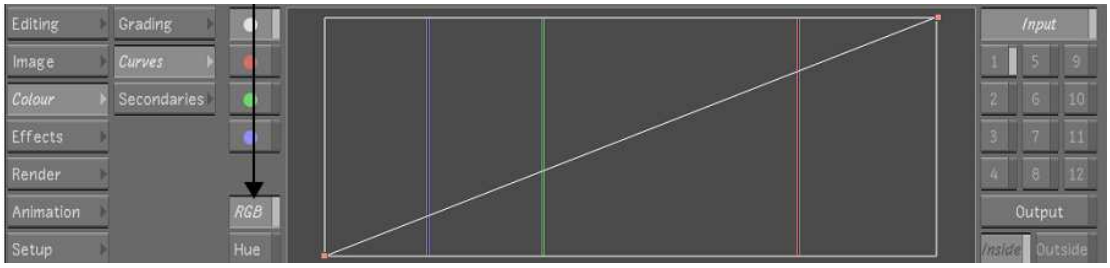
Barevné kruhy jsou tedy jiné, než u Speedgradu, tam jsme pro každý jasový rozsah měnili jak černý (Offset), tak bílý bod (Color Gain). Zde měníme pro každý rozsah jen jeho jas. Překvapivé je, že Lustre nám na panelu Grading může model ze Speedgradu přesto nabídnout (obr. 2.19).



Obrázek 2.19 – Panel Grading v lineárním módu

To, jaké barevné kruhy se v menu Grading zobrazí, záleží pouze na módu, který se mění v menu s vlastnostmi projektu. Logaritmický mód, ve kterém měníme jas, se používá pro projekty, které vycházejí z naskenovaném filmu. Pro digitální video se pracuje většinou v lineárním módu, tedy jako ve Speedgradu s kruhy pro nastavení černého a bílého bodu. Mód projektu by měl být nastaven na jeho začátku, a poté už by se během celé korekce neměl měnit. Opět je ale design celkem nepřívětivý, neboť změna parametru na jednom místě zcela ovlivní rozhraní jiného nástroje, skrytého kdesi v menu, aniž by o tom byl uživatel informován.

My však nebudeme video upravovat barevnými kruhy, neboť Lustre nabízí oproti Speedgradu i mnohem flexibilnější křivky.

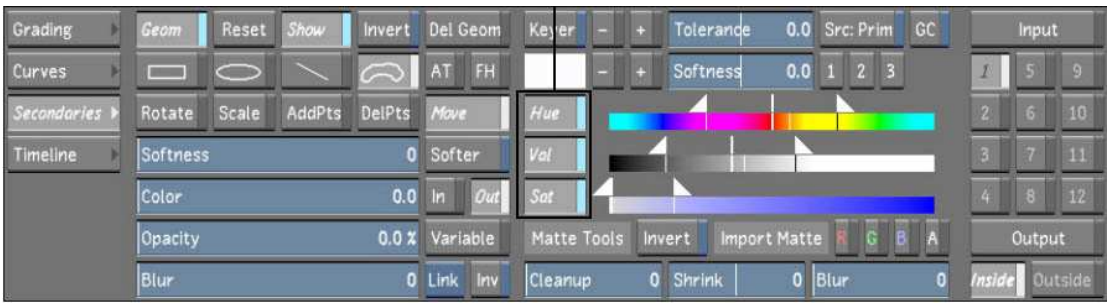


Obrázek 2.20 – Panel Curves

Na výběr máme mezi křivkami v prostorech RGB a HSV (odstín, sytost, jas). Klikneme na RGB, máme k dispozici tři křivky pro každý kanál a jednu kompozitní. My bychom tedy pro korekci obrázku pláže vybrali nejdříve Input, neboť chceme křivkami provést primární korekci na vstupním souboru, a nastavili křivkami bílý a černý bod. Ke kontrole rozmístění barev bychom opět použili histogram, Waveform nebo Vectorscope. Mezi originálem a korigovaným obrázkem bychom přepínali tlačítkem Bypass v horní části panelu.

Nyní rozmazání oblohy. Přejdeme do Colors->Secondaries (obr. 2.21). Zde nastavujeme jak masky, tak klíče. V tomto případě ohraničíme oblohu pomocí klíčování. Klikneme na tlačítko 1, na toto umístění budeme chtít přidat nový klíč. Barva se nastavuje pomocí posuvníků Hue pro odstín, Val pro jas a Sat pro sytost. Nemusíme ji nastavovat ručně, stačí kliknout na ikonku barvy nad tlačítkem Hue, tím se z kurzoru stane kapátko a kliknutím na pixel na viewportu nastavíme barvu klíče automaticky. Je výhodné takto navzorkovat více pixelů, při stisku klávesy Alt klikneme na oblohu, přejedeme po ní a tlačítko myši pustíme. Tak nastavíme na posuvnících také správné rozsahy Softness a Tolerance okolo centrální barvy. Barvy v rozsahu Tolerance budou plně naklíčované, ty

spadající do širšího rozsahu Softness budou po sekundární korekci prolnuty s originálním obrázkem. Pro zobrazení naklíčované oblasti klikneme na tlačítko M v horní části panelu, zpět se vracíme tlačítkem O, output. Prohlédneme si masku na celém záběru, posuneme ukazatelem časové pozice od začátku do konce. Když budeme mít jistotu, že je v celém záběru naklíčována vždy pouze obloha, můžeme přistoupit ke korekci.

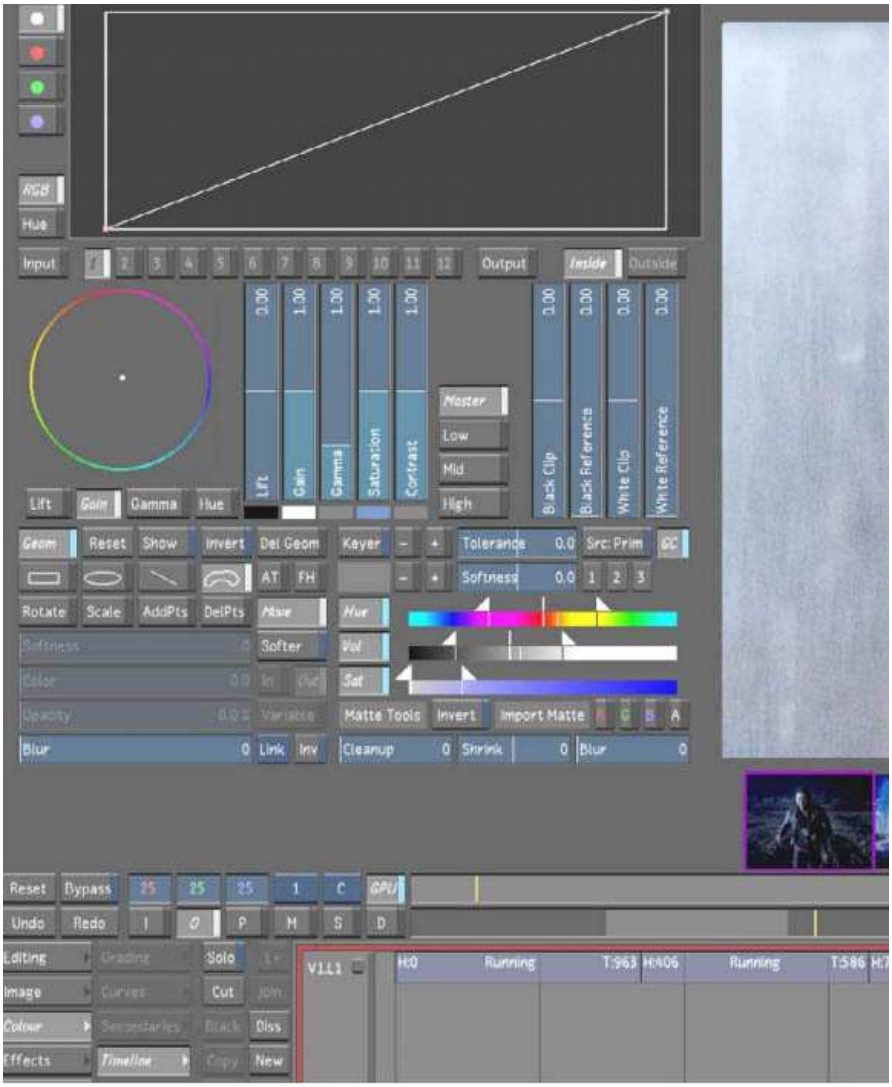


Obrázek 2.21 – Panel Secondaries pro výběr masky či barevného rozsahu

Rozmazání je zde opět jedním speciálním efektem, který se neřadí do kategorie primárních či sekundárních korekcí. Efekty jsou prováděny vždy po poslední sekundární korekci na pozici čísla 12 a finální primární korekcí pod Output. Klikneme na Effects->Plugin Setup a do zásobníku efektů přidáme BlurPlugin. Nastavíme jeho parametry a přiřadíme k pozici 1, na kterou jsme uložili náš klíč. Obloha je tak rozmazána v každém snímku záběru.

Nyní srovnání barvy napříč záběry. Možnosti jsou podobné jako ve Speedgradu. Lustre má však na časové ose vždy právě dva ukazatele pozice, A a B. Nastavíme je na sousední záběry a pomocí F5 měníme náhled viewportu jen na první, nebo na oba ukazatele zároveň, zobrazeny mohou být buď na split-screen nebo vedle sebe. Celkem si můžeme zobrazit až šestnáct záběrů. První dva jsou vždy pozice ukazatelů A a B, zbylých čtrnáct první snímky ve Storyboardu vybraných záběrů.

Viděli jsme tedy, že navigace ve Storyboardu přes svojí jednoduchost pro korekci dostačuje. Rozhraní s vždy viditelnou časovou osou je však natolik standardní, že jej i Lustre nabízí jako volbu. Pomocí F7 budou panely pro barevné korekce do prostoru viewportu a ve spodní části zůstane jen časová osa (obr. 2.22). Omezujeme tím však zároveň video a např. náhledy na dva snímky vedle sebe už tak budou muset být zmenšeny. Jen s časovou osou můžeme používat více stop, což je ve videu užíváno ne pro prolínání se stopami nižšími, ale především pro různé verze korekce, které jsou pak představeny člověku, zodpovědnému za finální podobu.



Obrázek 2.22 – Panely pro barevné korekce mohou být po vykotvení z menu do prostoru viewportu zobrazeny zároveň s časovou osou

Porovnáme-li Lustre se Speedgradem, konstatujeme, že oba jsou reálně použitelné softwary s týmiž základními funkcemi. Speedgrade nabízí uživatelsky příjemnější prostředí. U Lustru jsou pak nezastupitelnou výhodou křivky. Oba softwary jsou považovány za průmyslový standard. Z referencí Lustru uveďme filmy Babel či Valčík s Bašírem. Speedgradem byly upravovány např. filmy Cesta do středu země, Milionář z chatrče či Mládnoucí Benjamin Button.

3. Realizační část

Úkolem této práce bylo především navrhnout pro aplikaci Dikobraz grafické uživatelské rozhraní. Jak zkombinovat standardní nástroje Photoshopu s jednoduchostí a přehledností časové osy Speedgradu a s kompaktností Lustru?

Základní myšlenkou programu bylo, že bychom mohli upravit video nebo obrázek jak tím způsobem, který je standardní ve videu, tak tím v obrázku. Pro zopakování : hlavním rozdílem je kromě odlišných nástrojů to, že tyto ve Photoshopu jsou od sebe oddělené, zatímco ve video softwarech integrované. Kolorista videa tak jednou vrstvou provede veškeré úpravy a více vrstev použije, pokud chce vytvořit více verzí, více možností korekce. V obrázku máme buď každý nástroj na samostatné vrstvě, nebo vrstvy vůbec nepoužijeme a nástroj aplikujeme destruktivně. Photoshop si to může dovolit, obrázek je ihned rychle přerenderován a další úprava tak bude provedena ihned už na takto přepočítaném obrázku. To ale ve videu možné není. Představme si, že uděláme deset úprav. Museli bychom celou časovou osu počítat desetkrát, nejen aktuálně viditelný, ale každý jednotlivý snímek. Prodleva by tak zamezila rozumné práci. Nemohli bychom ani změnit parametry nástroje, který jsme použili o několik kroků dříve.

Je nutné zaznamenat celý proces od zdrojového souboru do výsledku, každý použitý nástroj a jeho parametry. Okamžitě budeme přepočítávat pouze náhled, na který se uživatel právě dívá. Rendering celého videa provedeme teprve na konci práce, když jsme po libovolném množství aplikovaných korekcích s výsledkem spokojeni. Tak pracuje i Speedgrade i Lustre, nevýhodou je, že oba aplikují jednotlivé nástroje v jakémsi pevném sledu, který není zjevný a uživatel jej jinak než z manuálu nezjistí. Dikobraz musí být flexibilnější.

Photoshop možnost zaznamenání parametrů nástrojů nabízí prostřednictvím vrstev úprav také. Každý nástroj si mohu přidat jako vrstvu úprav a poté se i třeba vrátit a upravit nástroj o několik vrstev níž. Vrstvy úprav ve Photoshopu ale selhávají při přepínání mezi barevnými prostory. Photoshop vyžaduje před konverzí zploštit obrázek do jediné vrstvy. Poté tak ztrácíme možnost navrátit se k nástrojům, použitým v předchozím barevném prostoru. Photoshop totiž vyžaduje, aby celý zásobník vrstev musí být nastaven na jeden barevný prostor. Nelze mít křivky v RGB a nad nimi křivky v LAB. GUI nás vede k tomu, že v barevném prostoru je obrázek, což teprve ovlivní barevný prostor nástrojů. U některých nástrojů je naopak barevný prostor tak pevně daný, že se nedá změnit ani konverzí do jiného barevného prostoru, např. Selektivní barva má posuvníky vždy pouze pro barvy CMYK.

3.1 Konverze jako samostatný nástroj

Tohoto omezení se chceme zbavit a nabídnout možnost použít libovolný nástroj v libovolném barevném prostoru. Potřebujeme také mít zaznamenány i všechny použité nástroje před konverzí do jiného barevného prostoru. Cestou k tomu by bylo vytvořit konverzi jako samostatnou vrstvu úprav. Nástroj by měl pouze zaznamenáno, do jakého barevného prostoru konvertovat a všechny nástroje ve vyšších vrstvách by pak už měnily parametry v jeho souřadnicích.

Dejme tomu, že bychom upravovali obrázek pouze nedestruktivně. Při ukládání každého jednotlivého kroku jako samostatné vrstvy úprav by ale počet vrstev velmi rychle narostl tak, že by se nedaly přehledně zobrazit.

Příklad:

7. konverze do RGB
6. ostření L kanálu
5. rozšíření barevného rozsahu křivkami
4. konverze do LAB
3. nastavení správného odstínu barev
2. konverze do CMYK
1. křivky pro vyrovnání světel, stínů

Příklad ukazuje jeden z nejjednodušších postupů pro úpravu obrázku a přesto by byl zaznamenán na 7 vrstev. Kdybychom je chtěli naráz zobrazit všechny, narazili bychom na problém s prostorem na obrazovce. Při rozlišení 1280x800 pixelů by časová osa o velikosti řádku 30 pixelů, po odečtení prostoru pro hlavičku okna, panel nástrojů, stavový řádek a Start lištu, zabírala téměř polovinu zbývající plochy obrazovky a na viewport by pak připadlo málo místa. Lišta Speedgrade zabírá mezi něco třetinou a čtvrtinou. Úkolem tedy bylo zobrazovat aplikované změny kompaktněji.

3.2 Model vrstvy

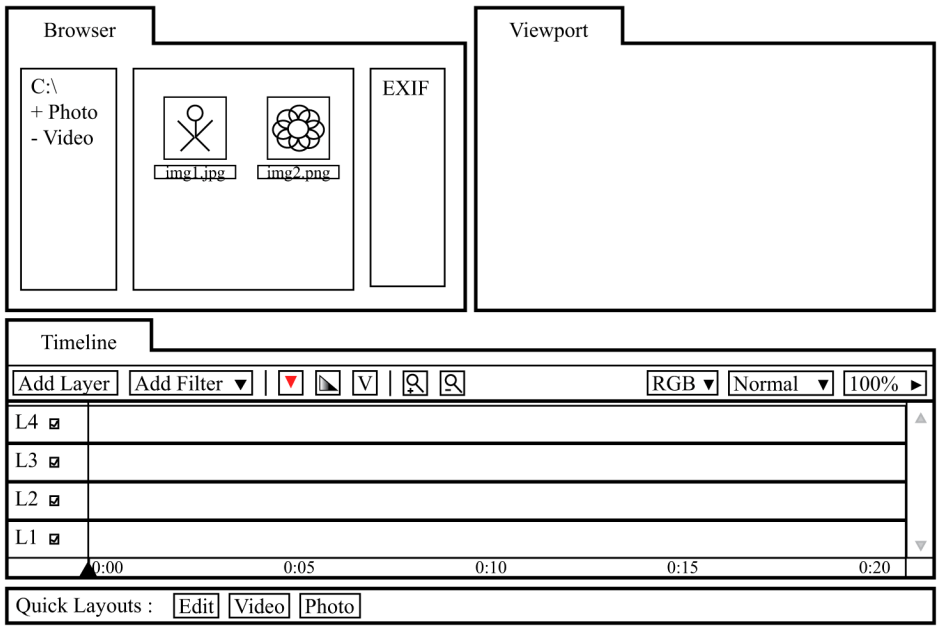
Nadefinujme si nejprve názvosloví, neboť jej nemůžeme převzít ani z Photoshopu, ani Speedgradu. Jednotlivé řádky časové osy se nazývají stopy. Do stop se kladou vrstvy. Vrstva může odkazovat na nějaký zdrojový soubor, tj. obrázek nebo video, nebo také nemusí, potom slouží pouze pro korekci. Filtr je nástroj pro přepočítání dat např. křivky, selektivní barva, odstín-sytost. Potud je vše podobné.

Vrstva úprav Photoshopu byla vrstva jednoho konkrétního filtru, v podstatě jen jeho jiné zobrazení. My budeme ale vrstvu chápat jako kontejner pro jednotlivé filtry. Do vrstvy budeme klást vybrané filtry za sebe. Vrstva bude mít vstup a výstup. Vstupem bude buď soubor z disku nebo výstup nižší vrstvy, výstupem data přepočítaná všemi filtry ve vrstvě. Každý filtr ale mění své parametry podle barevného prostoru, ve kterém se obrázek nebo video právě nachází. Předpokládáme, že úprav v jednom barevném prostoru bude více a je tedy nasnadě ponechat barevný prostor jako vlastnost vrstvy, která se propaguje do všech jejích filtrů. Pokud je vrstva na stopě 2 v jiném barevném prostoru, než vrstva na stopě 1 pod ní, provede se na vstupních datech vrstvy na stopě 2 konverze do jejího barevného prostoru.

Uvažovali jsme ještě o další možnosti šetření místa, a to ukládání vrstev do složky na jedné stopě, po vzoru softwaru Avid pro střih. Výhodou by bylo, že by si pak mohl uživatel mohl jednotlivé úpravy lépe hierarchizovat a věnovat např. jednu stopu pouze úpravám obličeje, další pouze úpravám pozadí, nehledě na to, kolik vrstev by to obnášelo. Složka by se dala zavřít a výška stopy v GUI by se tak měnila podle toho, zda by byla složka otevřená nebo zavřená. Nevýhodou by bylo, že při zavřené složce by se velmi špatně alespoň symbolicky zobrazovalo, co obsahuje, protože vrstev a filtrů v jedné takové složce by mohlo být narozdíl od Avidu mnoho. Při jedné otevřené složce by se zvětšila výška celé její stopy, uživatel by si tak musel hlídat, aby před úpravou jiných částí stopy měl složky vždy zavřené, jinak by časová osa ztratila na přehlednosti. Na jednom místě bychom měnili obrázek několika vrstvami ve složce, na druhém vrstvami v různých stopách. Nový stupeň v hierarchii by tak přispěl také ke zmatení. Proto jsme od této varianty upustili.

3.3 Korekce obrázku

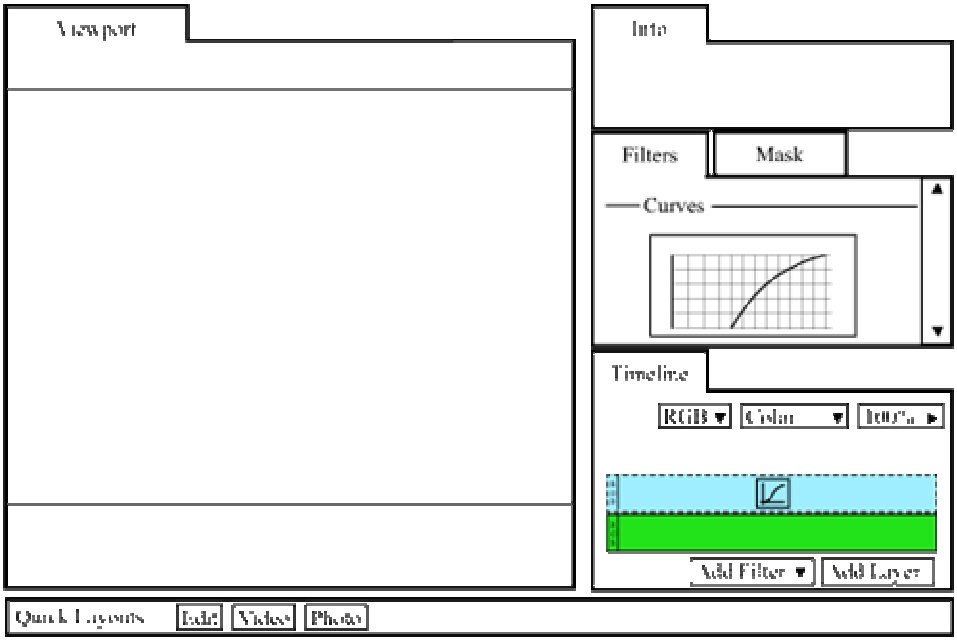
Po ujasnění nejdůležitější změny, kterou Dikobraz staví proti tradičnímu zobrazení informace ve střihacích softwarech, přistupme k návrhu uživatelského rozhraní celého programu. Představíme si, jak by se korekce obrázku z teoretické části prováděla v Dikobrazu. Po nastartování programu je v horní části programu zobrazen prohlížeč souborů, v dolní části časová osa. Budeme chtít upravovat obrázek, stačí tedy na něj v prohlížeči poklepat.



Obrázek 3.1 – Rozhraní pro střih, obrazovka po startu Dikobrazu

Rozvržení plochy se nyní změnilo a připomíná GUI Photoshopu. Obrázek je v centrální části roztáhnutý přes celou výšku obrazovky, časová osa na panelu vpravo. V první stopě časové osy nyní máme jedinou vrstvu, reprezentující obrázek ze zdrojového souboru.

Nad ní nyní přidáme tlačítkem Add Layer filtrovací vrstvu. Tu vybereme a přidáme na ní filtr práh Add Filter->Threshold. Na filtrovací vrstvě se nyní objevila nová ikona, reprezentující filtr. V záložce Filters nad časovou osou je nyní posuvník pro vybrání prahu. Posuneme jej na kraje a zafixujeme stejně jako ve Photoshopu nejbělejší a nejčernější body, abychom je měli na Info-paletě vždy na očích. Poté již filtr Threshold není potřeba a odstraníme z vrstvy pravým klikem na jeho ikonu a výběrem Remove. Nyní přidáme křivky, Add Filter->Curves. Na vrstvě je nyní nová ikona, jiná než v předchozím případě. Poklepeme na ní, křivky se otevřely v novém okně. Mohli bychom je také upravovat v záložce Filters, nové samostatné okno pro každý filtr je druhá možnost. Nastavíme nejdříve správnou bílou a černou, poté odstraníme zabarvení. Pomocí zaškrtávacího tlačítka Preview můžeme porovnat korigovaný obrázek s originálem. Křivky uzavřeme a z menu v horním panelu časové osy nastavíme vrstvě režim prolnutí Color.



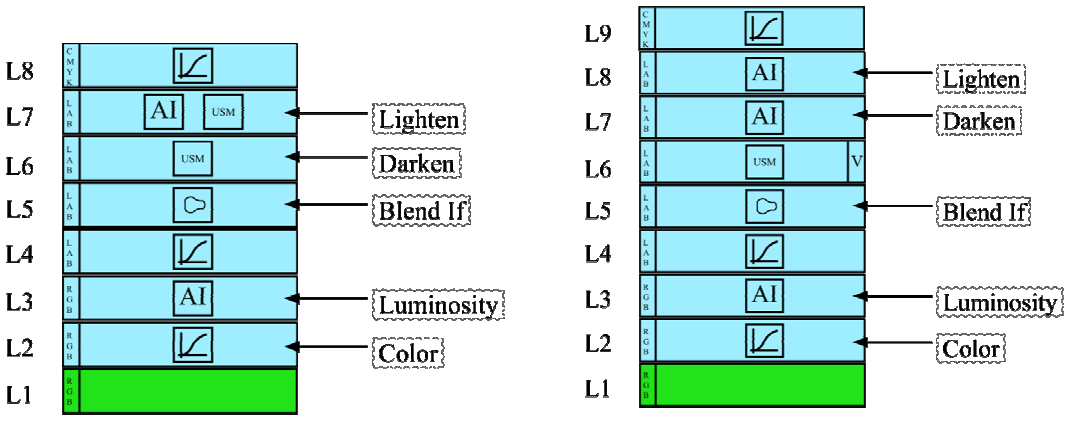
Obrázek 3.2 – Rozhraní pro úpravu obrázku

Nyní odstraňujeme šum z oblohy. Apply Image je v Dikobrazu samostatný filtr, nemůžeme ho ovšem přidat do téže vrstvy, jako křivky, neboť nechceme, aby se pak prolínal v režimu Color. Naopak, chceme prolínat v režimu Luminosity a jako vstup filtru chceme křivkami upravený obrázek (tedy po prolnutí v režimu Color). Přidáme tedy nejdříve novou vrstvu a do ní teprve filtr. Jako zdroj nastavíme zelený kanál základní vrstvy. Poté můžeme změnit režim prolnutí.

Dalším krokem je zvýšení sytosti barev v prostoru LAB. Barevný prostor je ale vlastnost celé vrstvy, a my tak opět potřebujeme přidat novou. Poté z rozbalovacího menu v levé části vrstvy vybereme LAB. Na vrstvu pak přidáme křivky a roztáhneme je kolem středu, stejně jako ve Photoshopu.

Pokud bychom přidali do téže vrstvy filtr pro rozmazání a ve vlastnostech vrstvy nastavili Blend If na modré plochy, aplikovaly by se i LAB křivky pouze v těchto místech. Filtr tedy přidáme do nové vrstvy, již s číslem 5. U dalšího kroku, tedy ostření, potřebujeme jinak ostřit světla a stíny, tedy nejdříve přeostřit a mít dvě vrstvy s různými režimy krytí. Jednodušší je do vrstev 6 a 7 přidat filtr rozmazání na maximum, u vrstvy 6 nastavit režim Darken s 90% krytí, u vrchní Lighten s 50%. Vrstva 7 by si tak ale jako svůj vstup brala již ostřenou vrstvu 6. Musíme tak na vrstvu 7 před ostření přidat filtr Apply Image, kterým do ní zkopírujeme výstup vrstvy 5. Také bychom mohli volit složitější variantu. Přeostřili bychom jen vrstvu 6, nenastavili ji žádné krytí, ale pouze bychom ji označili, jako virtuální. Virtuální vrstva se pouze vypočítá, ve vyšších vrstvách se však nepoužije jako zdroj. Zdrojem vrstvy 7, je nyní standardně vrstva 5, potom přidáme jen Apply Image z vrstvy 6 a Darken 90%, u vrstvy 8 taktéž jen s Lighten 50%. Vrstva 7 nyní sama prolíná do vrstvy 5 a vrstva 8 do vrstvy 7. Obě varianty jsou možné.

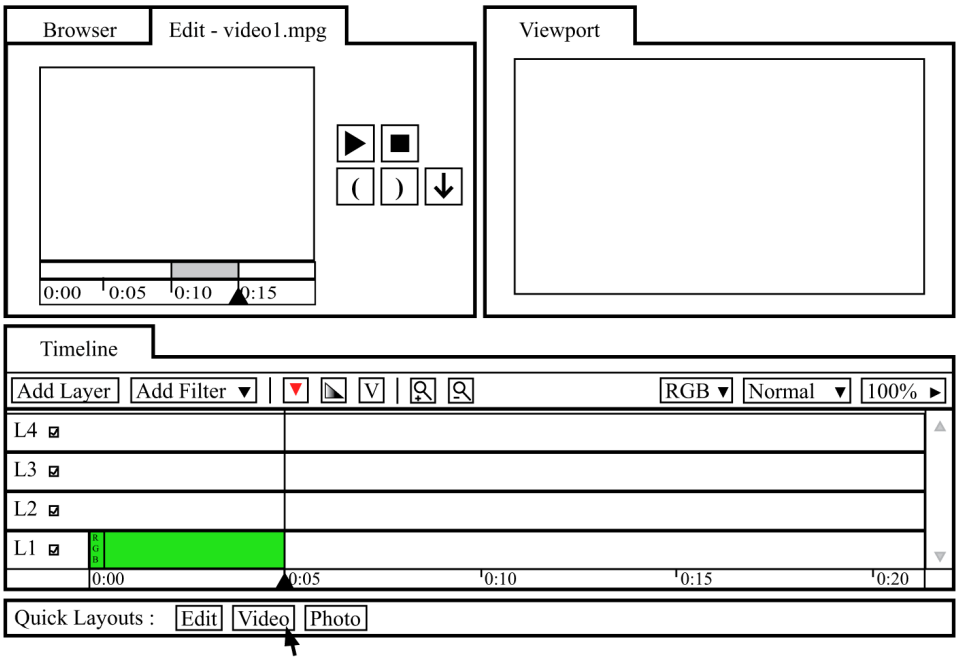
V příkladu k Photoshopu jsme prováděli ostření ve CMYK jen proto, že měl problémy s režimy prolnutí v LAB. V Dikobrazu si můžeme dovolit zůstat v režimu LAB a na CMYK změnit až poslední vrstvu, kam bychom přidali křivky pro závěrečnou korekci.



Obrázek 3.3 – Možnosti pro úpravu obrázku 2.1 v Dikobrazu

3.4 Korekce videa

Co by bylo jinak, pokud bychom upravovali video? Po poklepání na video v prohlížeči by se nezměnilo rozvržení plochy a časová osa by zůstala na témže místě. Pouze by se na místě prohlížeče otevřelo nové okno Edit pro střih vybraného video souboru. V něm bychom označili úsek, na kterém je zachycena naše pláž a po kliku na příslušné tlačítko bychom přidali na časovou osu novou vrstvu. Na obrazovce nyní máme v levé části náhled na konkrétní soubor v pravém na aktuální snímek našeho projektu, který měníme posouváním ukazatele na časové ose.



Obrázek 3.4 – Střih video souboru

Nyní stanovme, že na časovou osu již žádné další záběry přidávat nechceme. Přepneme tedy do rozhraní obrazovky pro korekce videa klikem na Video ve spodním panelu. Okna Edit a Browser již nejsou potřeba a byla skryta. Narozdíl od obrázku je v rozhraní pro video časová osa stále roztažena přes celou šířku obrazovky a v levé horní části je zobrazen místo Info-palety analyzační nástroj Waveform. Je vhodné podotknout, že rozvržení plochy je možné kdykoliv libovolně měnit nezávisle na těch předdefinovaných. Díky technologii Qt jsou okna vykotvitelná a uživatelem přizpůsobitelná, vyhneme se tak výše zmíněné neforemnosti Lustru.

Nad základní vrstvu bychom potom mohli přidávat stejné vrstvy jako v předchozím případě při korekce obrázku. I tam jsme terminologií videa aplikovali primární, a při rozmazání oblohy také sekundární korekce. Mohli bychom ale také postupovat způsobem, běžným ve videu. Nad zdrojovou vrstvu bychom do nové vrstvy přidali jen filtr Video Multitool, který je podobný panelu ze Speedgradu a obsahuje barevné kruhy pro nastavení černého, bílého bodu a gammy ve středních tónech, a to jak celkově, tak pro jednotlivé jasové rozsahy. Tak bychom měli vyřešenu korekci primární. Sekundární by byla v nové vrstvě opět s Video Multitoolem, která by jen měla nastavenou masku. Poté bychom do nové vrstvy přidávali filtry ostření a rozmazání.

3.5 GUI Dikobrazu

Photoshop funguje tak, že všechna okna s nástroji odvozují svůj obsah podle právě vybraného okna s obrázkem. V jednom okně programu může být otevřeno několik různých obrázků. V softwarech pro střih a korekci videa však nepracujeme na konkrétním souboru obrázku či videa, nýbrž vždy na projektu, který aplikace vytváří. V projektu jsou všechna okna provázaná navzájem a není jediné centrální. V témže okně s programem tak několik projektů naráz otevřít nemůžeme. Místo toho je při založení nového projektu aktuální projekt uložen a uzavřen a nový projekt je otevřen v témže okně. Pro naši aplikaci se tato standardní metoda organizace uživatelského rozhraní také hodí, neboť je kompatibilní i s případem užití, kdy chce uživatel provádět korekce mnoha obrázků najednou. Ty mohou být v témže projektu přidány na časovou osu a zde upravovány navzájem různými korekcemi.

Předpokládáme dále, že vstupem bude profesionálnímu koloristovi již sestříhaný projekt ve formě EDL souboru. Alespoň základní možnosti střihu jsou však žádoucí pro malé projekty, které by tak mohly být celé realizovány pouze pomocí Dikobrazu. Střih nebo výběr EDL souboru či dříve uloženého projektu je také prvním krokem uživatele po nastartování programu. Při střihu je potřeba kromě časové osy vidět také prohlížeč se soubory na disku, přehrávač vybraného videa z disku a přehrávač hotového projektu. Při úpravě videa již prohlížeč disku nepotřebujeme, při střihu oproti tomu nepotřebujeme např. Info-paletu. K rozhraní pro úpravu obrázku a videa tak přibylo rozhraní pro střih.

Mezi jednotlivými rozhraními budeme chtít snadno přepínat. Nevíme přesně, kdy uživatel dokončil střih a chce přejít ke korekci videa, nebo kdy bude chtít právě vložený obrázek upravovat jako součást videa a ne samostatně. Proto bude standardní a vždy viditelnou součástí rozhraní panel Quick Layouts s tlačítky pro tato přednastavená rozvržení, popřípadě pro další, uživatelem přidaná. Photoshop používá pro změnu rozhraní nenápadné rozbalovací menu v pravé horní části obrazovky. My jsme zvolili panel, protože je v našem případě přepínání nutné k reálné práci a musí se uživateli nabízet na jedno kliknutí.

Pro střih potřebujeme tři okna : prohlížeč, viewport a časovou osu. Prohlížeč bude zobrazovat náhledy na soubory na disku a také informace o právě vybraném souboru a

další metadata. U obrázků jsou to např. EXIF informace z fotoaparátu, u videa informace o lokaci, čase, autorovi, apod. Z okna Browser lze přetáhnout obrázky nebo video na časovou osu. Pokud v okně Browser jen poklepeme na obrázek, lze předpokládat, že uživatel chce upravovat pouze jej. Přepne se tak ihned do rozvržení plochy pro úpravu obrázku. Pokud poklepeme na video soubor, chceme na něm provést střih, otevře se tak přes Browser okno Edit. V levé části máme náhled do vybraného souboru, v pravé viewport, náhled na náš projekt. V okně Edit vybíráme vybíráme úsek videa, který chceme přidat na časovou plochu jako vrstvu. Je narozdíl od viewportu jedinečné, pokud by se uživatel vrátil do prohlížeče a vybral nové video, otevřelo by se ve stávajícím okně Edit.

Viewport zobrazuje snímek na pozici ukazatele časové osy. Jelikož časová osa může mít několik ukazatelů na různé pozice, bude vlastností viewportu také to, se kterým ukazatelem je spojen a výstup které stopy ukazuje. Viewportů můžeme mít také libovolný počet, každý si můžeme nakonfigurovat na jiný ukazatel, nebo také na týž, pouze na výstup jiné vrstvy. Tak bude moci uživatel přepínáním mezi viewporty kontrolovat stav před korekcí a po korekci, či zda ladí barvy v různých časech.

V rozhraní pro úpravu videa věnujeme opět velkou část obrazovky časové ose a viewportu. Zbylé místo zabírají nástroje pro barevné korekce videa. Vpravo nahoře jsou umístěny analyzační nástroje Waveform, Vectorscope a Info-paleta. Záložka Filters bude zobrazovat v seznamu pod sebou parametry všech filtrů, které jsou aplikovány na právě vybrané vrstvě, podobně jako v aplikaci Adobe Photoshop Lightroom. Parametry konkrétního filtru bude možné měnit také v samostatném okně, které se otevře po dvojkliku na jeho ikonu a nebude již na právě vybrané vrstvě závislé.

Rozhraní pro úpravu obrázku se liší jen tím, že časová osa je v pravém panelu a nejvíce místa zabírá viewport. Aktivním analyzačním nástrojem bude Info-paleta. Poměr stran obrázku navíc už není omezen na 4:3 nebo 16:9, jako ve videu, a často je potřebné upravovat obrázek focený na výšku. Toto rozhraní je tak výhodnější a uživatel Photoshopu je na něj zvyklý.

3.5.1 Časová osa

Časová osa definuje předpis, jakým způsobem mají být vstupní data modifikována. Obsahuje tlačítka pro přidání a odebrání jak vrstev tak filtrů. Má standardně dvě stopy, lze jich ale přidat libovolné množství. Do stop pak jednotlivé vrstvy klademe. Existují dva typy vrstev, datové a filtrovací.

Datové vrstvy jsou odkazem na konkrétní soubor videa či obrázku. Jsou vždy ponechány ve vstupním barevném prostoru, tedy standardně v RGB, a nemohou do nich být přidávány filtry. Bude tak zajištěno, že bude vždy k dispozici originální snímek jako výstup datové vrstvy a budeme se na něj moci ve vyšších vrstvách odkazovat, např. kopírovat jeho kanály či ho použít jako masku. U videa budou mít datové vrstvy omezený rozměr, vrstvu nepůjde roztáhnout na delší čas, než jaký video zabírá.

Filtrovací vrstva již toto omezení mít nebude. Do ní bude uživatel moci vkládat filtry a klíčové snímky. Jednotlivé filtry budou na vrstvě reprezentovány vlastní ikonou, po dvojkliku na ní se otevře nové okno s parametry filtru. Takových oken si budeme moci otevřít více a měnit tak snímek více nástroji naráz. Jako vstup si filtrovací vrstva bere v každém čase snímek z vrstvy na stopě pod ní, pokud zde žádná vrstva není vyzkouší další nižší stopu. Můžeme také manuálně nastavit, ze které stopy se má vstup brát pomocí filtru Apply Image. Parametrem filtrovací vrstvy je její barevný prostor, uživatel jej může podle libosti měnit. Pokud má vrstva nastaven jiný barevný prostor než má její vstup, provede se nejprve konverze. Snímek je pak postupně přepočítán všemi filtry a výstup posledního je zároveň výstupem vrstvy. Vrstvu si můžeme označit jako virtuální, tzn. vrstvu, která

nebude mít výstup. Pouze se vypočítá, ale pokud vyšší vrstva hledá svůj vstup, virtuální vrstvu ignoruje a přejde do nižší stopy. Virtuální vrstvy bude ale opět možné použít např. jako masku. Režimy prolnutí a procento krytí se mění v panelu časové osy pro právě vybranou vrstvu. Uvažovali jsme, zda by nebylo vhodné tyto parametry zobrazit a měnit přímo na každé vrstvě, ale ztratili bychom tím na přehlednosti.

Budeme moci měnit délku jednotlivých vrstev a přesouvat je. Vrstvy budou přitom těsně přilínat na sousední a to jak v horizontálním, tak ve vertikálním směru. Vždy při provádění korekce totiž musí nová vrstva začínat a končit přesně na tomtéž snímku, jako vrstva pod ní. To je zohledněno i při samotném přidávání nových vrstev. Vždy nejdříve označíme vrstvu a klikem na Add Layer přidáme novou, přesně nad ní na správnou pozici.

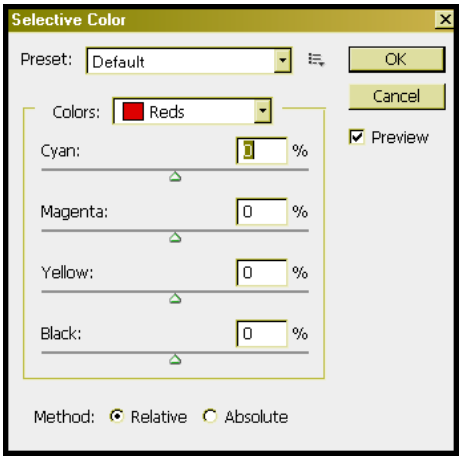
Součástí časové osy budou i ukazatele pozice, podle kterých se mění náhledy ve viewportech. Takových ukazatelů si bude možno přidat libovolné množství. Jeden ukazatel bude vždy hlavní a na jeho pozici se budou do vybrané vrstvy přidávat klíčové snímky. Časová osa bude umožňovat změnu měřítka.

3.5.2 Návrh filtrů

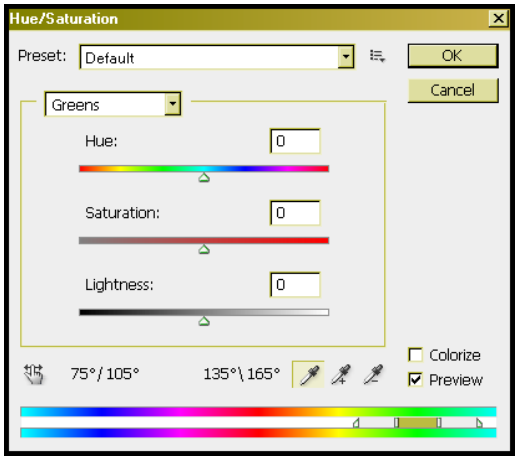
Filtry budou mít zaškrtávací tlačítko Preview. To aktivuje filtr pro zobrazení ve viewportu. Narozdíl od Photoshopu ale neslouží jen pro aktuální otevřené okno s filtrem, ale je to zkrátka další parametr filtru, který se po jeho zavření a znovuotevření uchovává. Pokud je zaškrtnuté a viewport je nastaven na vrstvu s tímto filtrem, zobrazí výstup této vrstvy. Pokud zaškrtnuté není, zobrazí obrázek, který byl přepočítán pouze ostatními aktivními filtry na vrstvě. Kdybychom za našimi křivkami měli na vrstvě další filtry a chtěli zobrazit pouze výstup křivek, zkrátka bychom u těchto zbylých filtrů tlačítko Preview odškrtli. Pokud je filtr neaktivní, má jeho ikona na časové ose šrafované pozadí. Tak má uživatel i po zavření okna s parametry filtru kontrolu, co vlastně viewport zobrazuje.

Selective Color

Filtr bude obsahovat posuvníky pro měnění jednotlivých kanálů barevném prostoru vrstvy. Narozdíl od Photoshopu (obr. 3.5) tak nebude uživatel omezen jen na CMYK. Vedle každého posuvníku bude tlačítko pro jeho nastavení do základní polohy. Selektivní barva, která pak bude těmito posuvníky měněna, bude nastavitelná jak v rozbalovacím menu, tak na barevném posuvníku ve spodní části filtru, podobně, jako ve Photoshopu u nástroje Hue/Saturation (obr. 3.6). Tím bude uživatel mít možnost rozsah selektivní barvy podle přání měnit.



Obrázek 3.5 – nástroj Selective Color ve Photoshopu



Obrázek 3.6 – nástroj Hue/Saturation ve Photoshopu

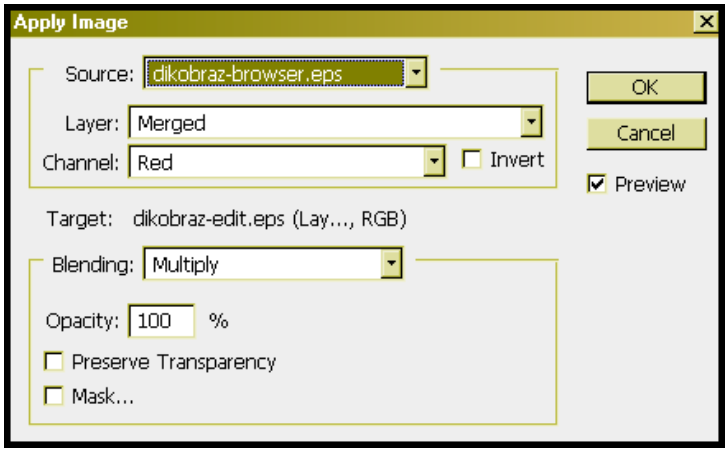
Curves

Křivky v Dikobrazu se budou inspirovat rozhraním křivek z Photoshopu (obr. 2.4) s několika vylepšeními. Jejich barevný prostor se bude odvíjet od toho, na který je nastavena vrstva. Pro každou křivku by narozdíl od Photoshopu měly být vodící gradienty po stranách mřížky v příslušné barvě. V R křivce např. od černé do červené, hlavně ale v A křivce LAB prostoru od zelené do purpurové. Zde totiž i zkušený uživatel zápasil s tím, která barva je na které straně od středu. Body na křivce můžeme číselně zobrazit jako skutečnou hodnotou pixelu nebo v procentech tiskařského pigmentu. Pak jsou ovšem barvy na opačných koncích os. Se správnými barevnými gradienty bude situace přehlednější.

Body na křivce reagují na šipky na klávesnici. Jemné změny pozice bodů jsou nejdůležitější opět právě v LAB prostoru. Zde mohou být body kolem středu velmi blízko sebe, klidně vzdáleny pouhý pixel. Naše křivky by situaci měly zpřehlednit a umožňovat přiblížení, narozdíl od časové osy budou plně dostačovat dvě měřítka..

Apply Image

Nástroj pro zkopírování snímku nemůže být aplikován destruktivně, jako ve Photoshopu (obr. 3.7), ale bude součástí naší základní sady filtrů. Odkazuje se na výstup nějaké nižší vrstvy, který lze prolnout s aktuálním obrázkem. Podle jejího barevného prostoru nabídne také, které kanály bude možno zkopírovat. Pokud bude vybrán v rozbalovacím menu pro zdroj kanál, bude moci také být v druhém rozbalovacím menu také vybrán kanál cílový na aktuální vrstvě. Opět budou přitom nabídnuty kanály jejího barevného prostoru. Budeme tak moci např. zkopírovat R kanál do A kanálu, k čemuž ve Photoshopu bylo nutné zduplikovat obrázek.



Obrázek 3.7 – nástroj Apply Image ve Photoshopu

Video Multitool

Samostatným filtrem bude i speciální nástroj pro korekci videa, který bude kopírovat rozložení konzole, na které je video v praxi upravováno. Můžeme si jej představit jako panel pro korekce ve Speedgradu (obr. 2.12, 2.14). K dispozici budou barevné kruhy Offset, Gamma, Gain, a to jak pro celý obrázek, tak pro jednotlivé jasové rozsahy stínů, světel a středních tónů. Ostatní nabídky na pravé straně panelu Speedgradu nebudeme potřebovat. Maska, jak vektorová, tak pixelová bude přidávána ve speciální záložce Mask a bude to vlastnost vrstvy, ne filtru. Speciální zásobník pro efekty nebude zapotřebí, neboť v naší terminologii budou totéž, co filtr.

3.6 Implementace

3.6.1 Pohyb vrstev

Nejdříve byla navržena vrstva tak, že sama odchytávala události pro změnu své šířky a posouvání. Potřebovali jsme ale zobrazovat vodící linky pro přilnutí k dalším vrstvám. Musela by se tak na pokaždé dotazovat časové osy, zda se v blízkosti aktuální souřadnice svého pravého nebo levého okraje nenachází v libovolné stopě nějaká další vrstva. Zároveň po by musela do časové osy vysílat signál, pokud byla vybrána. Nebylo to vhodné, protože jejím rodičovským prvkem není časová osa není, nýbrž stopa. V hierarchii by tak musela přeskakovat jednu úroveň.

Na události pro vrstvy tak nebude reagovat vrstva sama, ale objekt LayersController. Ten bude součástí objektu Timeline, bude mít přehled o všech vrstvách a provádět s nimi operace. Obstará posouvání vrstev, měnění jejich velikosti, bude pro ně hledat místa vhodná k přilnutí i udržovat seznam vrstev právě vybraných a přidávat do nich nové klíčové snímky a filtry. Podobně TracksController pak bude udržovat seznam všech stop a přidávat nové vrstvy. Datové vrstvy do aktuálně vybrané stopy na pozici ukazatele, filtrovací vrstvu vždy do stopy nad aktuálně vybranou vrstvu s týmiž pozicemi levého a pravého okraje. Bude také zajišťovat počítání obrázku na pozici ukazatele. To probíhá tak, že se dotazuje všech stop, zda mají na pozici ukazatele nějakou vrstvu a pokud ano, předá této vrstvě ke zpracování výstup nižší vrstvy.

Vrstva bude uchovávat svou přesnou pozici a rozměr. Jednotkou k tomu budou snímky. Zároveň bude obsahovat jako lokální parametr měřítko časové osy, kolik snímků připadá na jeden pixel. Vrstvy pak sami přepočítají, na jaké pozici se mají zobrazit.

3.6.2 Filtr

Každý filtr bude samostatnou třídou, odvozenou od společného rozhraní Filter. Každý filtr má např. ikonu, název filtru formou řetězce, či svůj typ, reprezentovaný celočíselnou proměnnou, pod kterou bude filtr v Dikobrazu zaregistrovaný. Dále budou mít jednotlivé filtry vlastní parametry.

Objekty filtrů budou přidávány přímo filtrovací vrstvou v metodě addFilter. Ta má jako parametr typ filtru, podle kterého by musela nejdříve nalézt správnou třídu. Jelikož ale by takové rozhodování nemělo být starostí vrstvy, přesunuli jsme instancování filtrů do samostatné tovární třídy FilterFactory, která bude dělat pouze a jen tuto činnost. Dojde tak k lepšímu oddělení objektů od sebe Časová osa i vrstvy pak budou pracovat pouze s rozhraním Filter, ne s třídami konkrétních filtrů. Navíc tím bude pro další verze programu umožněno přidávání dynamických filtrů pouze jejich registrací do této tovární třídy.

3.6.3 Vztah klíčového snímku a filtru

Vrstva bude mít určitý počet klíčových snímků n a určitý počet filtrů m. Jelikož každý filtr bude mít různé parametry v jednotlivých klíčových snímcích, musíme datově situaci reprezentovat tak, že pro každý klíčový snímek vytvoříme zvláštní seznam m filtrů. Vznikne tak dvojrozměrné pole. Pro potřeby prezentační vrstvy, která klíčové snímky a filtry dynamicky přidává a odebírá, potřebujeme na jeho místě dvojrozměrný spojový seznam QList<QList<Filter*> > keyfr_filters. Index ve vnějším seznamu bude reprezentovat klíčový snímek, index v seznamu vnitřním pak ke které ikoně na filtrovací vrstvě se filtr váže.

Mimo tento seznam si bude vrstva udržovat seznam grafických objektů, reprezentující ikony filtrů QList<QWidget*> filterIcons a keyframů QList<QWidget*> keyfrIcons. Obě tedy budou čistě vizuální reprezentace bez vlastní dat. Vrstva bude odchytávat události pro tyto grafické objekty a podle jejich indexu najde požadovaný objekt Filter z vlastních dat keyfr_filters. U klíčových snímků bude vrstva udržovat ještě další seznam QList<int> keyfrPos s přesnou pozicí klíčového snímku. Opět budou přes index odpovídat seznamu ikon keyfrIcons.

3.7 Memory management bloků obrázku

Obrázek nebo snímek videa bude rozdělen na bloky, které se budou počítat samostatně. Pokud bude viewport přiblížený pouze na určitý výsek obrázku, bude možno program urychlit a počítat pouze ty bloky, které jsou vidět.

Jednotlivé vypočtené bloky budeme ukládat do paměti. Pokud však bude docházet, potřebujeme ji uvolnit, tj. vymazat z ní ty bloky, které s největší pravděpodobností nebudou v nejbližší době potřeba. Pokud se například ve viewport posuneme do jiné části obrázku, je vhodné udržet tu předchozí v paměti, neboť v se dalším kroku uživatel může snadno přesunout zpět, nebo obrázek oddálit. Stejně tak je pravděpodobné, že uživatel zkontroluje své korekce vypnutím některé z nižších vrstev.

Měli jsme tedy za úkol vytvořit garbage collector. Otázkou je, jak najít nejméně potřebné bloky a jestli není nakonec rychlejší je nehledat a paměť uvolňovat náhodně. Cílem je vždy co nejrychlejší počítání.

3.7.1 Průběh počítání obrázku

V našem simulátoru jsme korekcí obrázku pláže v Dikobrazu (obr. 3.3 vpravo). Celkem máme devět vrstev, které na sobě navzájem závisejí. Vrstva 9 potřebuje ke své činnosti vrstvu 8, vrstva 8 potřebuje vrstvy 7 a 6, vrstva 7 potřebuje vrstvy 6 a 5. Vrstva nezávisí nutně na té, která je právě pod ní a může navíc záviset na více vrstvách.

Cílovým obrázkem je výstup vrstvy 9. Zavoláme tedy metodu pro počítání teto vrstvy. V ní nejprve prohlédneme, na kterých vrstvách závisí. Pokud jsou tyto nižší vrstvy v paměti, můžeme provést výpočet. Pokud ne, což je náš případ, zavoláme tutéž metodu pro vypočítání vrstvy 9. Používáme tedy rekurzi. Vrstva 8 opět hledá v paměti vrstvy, na kterých závisí, tedy vrstvy 7 a 6. Nenalezne je, opět si je tedy rekurzí vypočítá. Takto se dostaneme až na základní vrstvu 1, kterou již máme paměti máme a rekurze se tak zastaví. Ve zpětném chodu rekurze tedy již můžeme počítat vyšší vrstvy. Vrstvu 1 potřebovala vrstva 2, provede se její výpočet a uloží se do paměti. Nyní se rekurze vrátila do metody, která počítala vrstvu 3, opět se uloží do paměti atd.. Na konci budeme mít vypočtenu vrstvu 9.

3.7.2 Vrstvy, které počítají průměr

Tak bychom tedy počítali v každé vrstvě celý obrázek. V našem případě je to ale zbytečné, pokud je viewport přiblížený pouze na jeho část. Obrázek je pak rozdělit na bloky konstantní velikosti a počítat pouze viditelné bloky. Ty budeme také ukládat do paměti a posléze z paměti odstraňovat. Výše uvedená rekurzivní metoda tak nyní platit ne pro vrstvy, nýbrž pro každý jednotlivý blok. Na vrstvě 9 budeme pro každý blok, který je na viewportu rekurzivně sestupovat vrstvami a počítat na sobě závislé bloky.

Za takové situace bychom tedy v každé vrstvě počítali jen bloky na stejných souřadnicích jako bloky vrstvy 9, které jsou na viewportu vidět. Filtry rozmazat či ostřit však fungují tak, že ke své činnosti potřebují ze zdrojové vrstvy nejen bloky na téže souřadnici, ale také v nějakém okolí. Uvažujme jednorozměrný obrázek, v následujícím příkladě je ve sloupcích x-ová souřadnice bloku a v řádcích číslo vrstvy. X znamená, že je blok nutné počítat, O nikoliv :

```
  123456
3 00XX00
2 0XXXX0
1 0XXXX0
```

Ve vrstvě 3 je filtr ostření nastavený na radius 1. Kdybychom tak měli viewport přiblížený pouze na bloky 3 a 4 vrstvy 3, potřebovali bychom pro jejich výpočet také bloky 2, 3, 4, 5 vrstvy 2. Ta již ostřící není, proto potřebuje z vrstvy 1 opět pouze bloky 2, 3, 4, 5. Obráceně chápáno, na bloku 3 vrstvy 2 závisí bloky 3 a 4 vrstvy 3, a je proto důležitější, než např. blok 3 vrstvy 1. Pokud by bylo více ostření nad sebou, můžeme i velmi přiblížený viewportu potřebovat mnoho bloků z nižších vrstev.

3.7.3 Odstraňování bloků z paměti

Představme si, že jsme vypočetli pět bloků vrstvy 9 a v průběhu rekurze, která má počítat šestý blok, dojde paměť a je potřeba některé bloky z paměti odstranit. Určitě neodstraníme vypočtené bloky vrstvy 9, neboť právě ty chceme posléze zobrazovat.

Naskýtá se nyní několik variant, které bloky odstranit :
1. ty, které jsme do paměti přidali jako první
2. ty, které jsou nejméně náročné na výpočet
3. ty, na kterých závisí co nejméně bloků z vrchních vrstev

Nejrozumnější by bylo všechny způsoby zkombinovat. Ze všech parametrů tak vypočítáme pro každý blok jedinou proměnnou idx, podle které budeme udržovat na paměti bloků seřazenou haldu. Nejméně důležité bloky s nejmenším indexem tak budou moci být odstraněny v O(1).

Každá vrstva má nastavenu náročnost výpočtu jejího jednoho bloku. Náročnosti se různí, vrstva s křivkami bude mít jinou náročnost, než vrstva se třemi dalšími filtry. U každého bloku si budeme také uchovávat počet bloků z vyšších vrstev, které na něm závisí (proměnná citac). A konečně, paměť bude udržovat svůj časovač, inkrementovaný po každém vložení bloku. Při vložení bloku do paměti do něj uložíme její aktuální čas.

Vše tedy nyní můžeme zkombinovat :
idx = narocnost * (citac + 1) * sqrt (cas_vlozeni / aktualni_cas)

Proměnnou citac zvyšujeme o 1, aby nenuloval index v případě, že bude 0 (tzn. když žádný další blok na aktuálním bloku nezávisí). Odmocnina je ve vzorci proto, abychom snížili váhu času oproti náročnosti a hodnotě čítače. To se ukázalo po testu jako optimální jak oproti ponechání vzorce bez odmocniny, nebo snížení váhy ještě více třetí či pátou odmocninou.

Paměť neuvolňujeme po jednom prvku, ale hromadně při garbage kolekci. Jak bude paměť docházet uvolníme naráz celou polovinu. Halda tak bude muset být po každém odstranění prvku přeřazena, což ovšem nevadí, neboť tak činí velmi rychle v O(log n).

3.7.4 Faktická náročnost

Pokud ale v průběhu dalšího počítání narazíme např. ve vrstvě 5 na blok, který ke své činnosti potřebuje blok, který jsme již odstranili z paměti (např. z vrstvy 4), musí se tento vypočítat a uložit do paměti znova. Pokud by byly zároveň odstraněny i všechny bloky, na kterých rekurzivně závisí tento blok z vrstvy 4, museli bychom pro náš původní cíl, tedy výpočet bloku ve vrstvě 5 zároveň vypočítat navzájem závislé bloky na nižších vrstvách a mohli bychom se takto dostat opět až na vrstvu 1. Jde tedy také o to, aby takové opětovné počítání alespoň dlouho netrvalo, když už k němu má dojít. Aby např. při uvolnění bloku z vrstvy 4, zůstal v paměti blok na vrstvě 3, ze kterého by se poté počítání bloku vrstvy 5 mohlo vyjít.

3.7.5 Algoritmus s faktickou náročností

Výše uvedený vzorec pro uvolňování z paměti můžeme vylepšit nahrazením náročnosti za faktickou náročnost, kterou blok potřebuje pro svůj výpočet. Máme opět

celkem devět vrstev, pro jednoduchost každou o náročnosti 100. Máme v paměti blok z vrstvy 5, také s náročností 100, a došla paměť, odstranili jsme z paměti např. potřebný blok z vrstvy 4 s náročností 100. Referenční blok z vrstvy 3 v paměti zůstal. Potom faktická náročnost daného bloku z vrstvy 5 bude 200. Tento blok tedy bude důležitější, než např. blok ve vrstvě 6 s faktickou náročností 100. Když tedy budeme hledat další prvek k odstranění, odstraníme ten s nižší faktickou náročností, třeba právě z vrstvy 6, čímž nám faktická náročnost bloku na vrstvě 7 vzroste na 200. Což je ale méně než 300, kdybychom odstranili blok 5 museli pro blok vrstvy 6 vycházet až z vrstvy 3. Vždy se bude nějaký prvek velmi blízko kýženého udržovat v paměti.

idx = fakticka_narocnost * (citac + 1) * sqrt (cas_vlozeni / aktualni_cas)

Proměnná fakticka_narocnost se tedy opět bude počítat rekurzivně a dotazovat se, zda jsou závislé bloky v paměti, pokud ne fakticka_narocnost se zvýší o proměnnou narocnost dané vrstvy. Po každém odstranění prvku z paměti se tak musí před řazením haldy přepočítat i indexy bloků z vyšších vrstev, které na aktuálním závisí, protože se u nich změnila faktická náročnost. Prodluží to tedy čas výpočtu, proto v naší implementaci trvá tento algoritmus kvůli přepočítávání mnoha indexů zdaleka nejdéle. Cílem ale bylo zjistit, kolik přitom ušetřím na počítání samotných bloků obrázku. Díky tomuto algoritmu bychom měli v paměti zachránit ty důležitější, celkový proces by se tedy teoreticky měl urychlit, což bylo nutno potvrdit mnoha testy.

3.7.6 Průběh testování

Vytvořili jsme si program, který výše uvedené přidávání a odstraňování bloků do paměti simuluje. Jeho cílem bylo porovnat celkové náročnosti počítání při použití tří algoritmů pro memory management.

1. bez plánovače
2. s použitím náročnosti, čítače a stáří
3. s použitím faktické náročnosti, čítače a stáří

Je samozřejmé, že nestačí provést test pouze na jednom obrázku. Za určitých podmínek se může plánovač chovat jinak, než ve většině případů. Je nutno počítat s mnoha operacemi, výsledná tabulka by měla porovnat tři výše uvedené algoritmy na jedné konstantní sekvenci operací. Tyto operace reprezentují práci uživatele. Jsou to změny, po kterých je nutno přepočítat obrázek :

1. posun viewportu
2. zoom viewportu
3. změna parametrů některé vrstvy
4. zapnutí nebo vypnutí některé vrstvy

Na začátku počítání tabulky tedy náhodně vygenerujeme nejprve sekvenci těchto operací a tu pak necháme počítat všemi algoritmy. Výstupem jednoho takového počítání sekvence bude číslo, které udá, jak se doba počítání liší od doby počítání, kdyby sekvence proběhla bez odstraňování bloků z paměti. Teoreticky bychom tedy měli k dispozici nekonečnou paměť. Pro každý algoritmus tak vypočítáme

out = cas_pocitani_sekvence / cas_pri_nekonecne_pameti

Z tohoto čísla lze vyčíst, kterým algoritmem se ideálnímu výsledku blížíme nejvíce. Čím menší číslo, tím rychlejší algoritmus. Ještě zbývá jeden parametr, který hraje roli, a totiž velikost paměti. Pokud máme paměti málo, budeme muset uvolňovat velmi často a plánovač se může projevit zcela jinak.

3.7.7 Testovací příklad

Vraťme se k našemu příkladu z Margulise. Rozměr obrázku bude 8x8 bloků, viewport v levém horním rohu přiblížený s velikostí 5x5 bloků. Máme 10 vrstev, celkem tedy 8x8x10 = 640 bloků. Pokud by paměť měla místo alespoň pro 640 bloků, nemuseli bychom z ní nikdy bloky uvolňovat a náš plánovač bychom neotestovali. Proto si rozškálujeme velikost paměť na 100, 200, 300 a 400 bloků. Pro každou velikost paměti vypočteme sekvenci třemi algoritmy. Celkem tedy budeme provádět minimálně 12 počítání sekvence. Abychom obdrželi v rozumném čase výsledek, stanovili jsme velikost sekvence na konstantních 200 operací.

Uživatel si bude také moci zvolit, kolik různých sekvencí chce počítat. Výstupem bude tabulka s výše uvedenými podíly out pro jednotlivé algoritmy a velikosti paměti. Tabulka tak nabídne lepší srovnání pro více testovacích případů, pokud bude obsahovat výsledky počítání vícera sekvencí. Je však pak třeba počítat s delším časem. Vypočítání jedné sekvence přes všechny algoritmy a náročnosti paměti trvá na počítači Intel Core2 Duo 1,5 GHz s 2 GB RAM něco přes minutu. Kromě počtu počítaných sekvencí bude uživatel vyzván také k vybrání operací, které se mají provádět. Bylo totiž zajímavé zjistit, zda algoritmus pro faktickou náročnost nefunguje lépe např. pro posouvání viewportu, než pro měnění parametrů vrstvy.

3.7.8 Výsledek Testování

Jako výsledek počítání tří sekvencí jsme obdrželi následující tabulku. Pro každou velikost paměti (číslo na začátku) máme v řádcích algoritmy, ve sloupcích jednotlivé sekvence.

```
100 : bez planovace                 6.04 4.57 3.98
      podle narocnosti              2.64 3.09 2.98
      podle fakticke narocnosti     2.58 3.04 2.86
200 : bez planovace                 2.31 2.12 2.03
      podle narocnosti              1.30 1.53 1.46
      podle fakticke narocnosti     1.26 1.45 1.50
300 : bez planovace                 1.36 1.38 1.37
      podle narocnosti              1.08 1.19 1.16
      podle fakticke narocnosti     1.08 1.17 1.16
400 : bez planovace                 1.10 1.11 1.16
      podle narocnosti              1.04 1.04 1.06
      podle fakticke narocnosti     1.03 1.05 1.05
```

Je vidět, že použití plánovače je vhodné. Algoritmus s faktickou náročností přinesl také zlepšení, ne ovšem takové, jak jsme očekávali. Nejvíce se projeví, pokud je k dispozici málo paměti a musí tak být provedeno co nejvíce odstraňování bloků.

4. Závěr

Lidé, kteří dlouhodobě pracují s danou technologií, často přestanou vnímat omezení, která jim klade. Působí zvyk na osvědčená řešení a úžasná lidská adaptabilita, kterou bez fantazie nelze překonat a nastínit lepší alternativy. Zaslouží tak zmínit, že odrazovým můstkem byl nápad vedoucího práce na propojení barevných korekcí ve videu a v obrázku, jedné a téže činnosti, která byla v praxi rozdělena.

Cílem práce byl pak především návrh uživatelského rozhraní pro program, který by to umožňoval. Oba způsoby korekce jsme nejdříve představili v referenčních aplikacích a zároveň poznamenali, jak se liší. Nejdůležitější změnou, kterou jsme v návrhu museli oproti tradici provést, bylo přidávání filtrů do vrstvy. Uživatel má nyní možnost zvolit, kterými filtry chce obrázek či video upravovat, které budou aplikovány dříve a které až později, není svázán jejich pevně daným sledem. Díky tomu jsou filtry registrované v programu škálovatelné a v budoucnu nebude problém filtry přidávat dynamicky formou pluginů. V základní sadě filtrů jsme pak navrhli jejich vylepšení oproti referenčním aplikacím. Dále bylo vyřešeno nedestruktivní přepínání mezi barevnými prostory na vstupu každé vrstvy.

Byly navrženy tři základní rozvržení plochy: pro střih, úpravu videa a úpravu obrázku. Na konkrétním příkladě jsme si ukázali celý postup práce od původního snímku k výsledku a ujistili se tak, že program bude použitelný i v praxi.

Otestovali jsme se také, jak by měl program nejlépe hospodařit s pamětí při přepočítávání obrázků. Zjistili jsme, že teoreticky lepší algoritmus, který zohledňoval faktickou náročnost při počítání každého jednotlivého bloku obrázku, snižoval celkový čas počítání jen minimálně. V budoucnu tak budeme tak vždy zohledňovat náročnost počítání pouze aktuálně odstraňovaného bloku.

Pro autora pak byla práce především vhodná příležitost k tomu, naučit srozumitelně vyjadřovat, stavět mosty mezi myšlenkami a více vnímat barvy v běžném životě.

Přehled zkratek

EDL – Edit Decision List
GUI – Graphical User Interface
HSV – Hue Saturation Value
EXIF - Exchangeable image file format

Literatura

[Margulis] MARGULIS, D. Professional Photoshop : The Classic Guide to Color Correction. 5th edition, Berkeley (California): Peachpit Press, 2007. ISBN 0-321-44017-X.

[Hullfish] HULLFISH, S. The Art and Technique of Digital Color Correction, Burlington: Focal Press. 2008. ISBN 978-0-240-80990-8.

[Prata] PRATA, S. Mistrovství v C++. 1. vydání. Praha: Computer Press, 2001. ISBN 80-7226-339-0.

[Blanchette] BLANCHETTE, J. – SUMMERFIELD, M. C++ GUI Programming with Qt4. 1st printing. Stoughton (Massachusetts): Courier, 2006. ISBN 0-13-187249-4.

[SpMan] Speedgrade 2009 User Guide [online] . [cit. 20010-05-01] http://doc.iridas.com/iri49ges/d/d7/SpeedGrade_User_Guide_2009.pdf

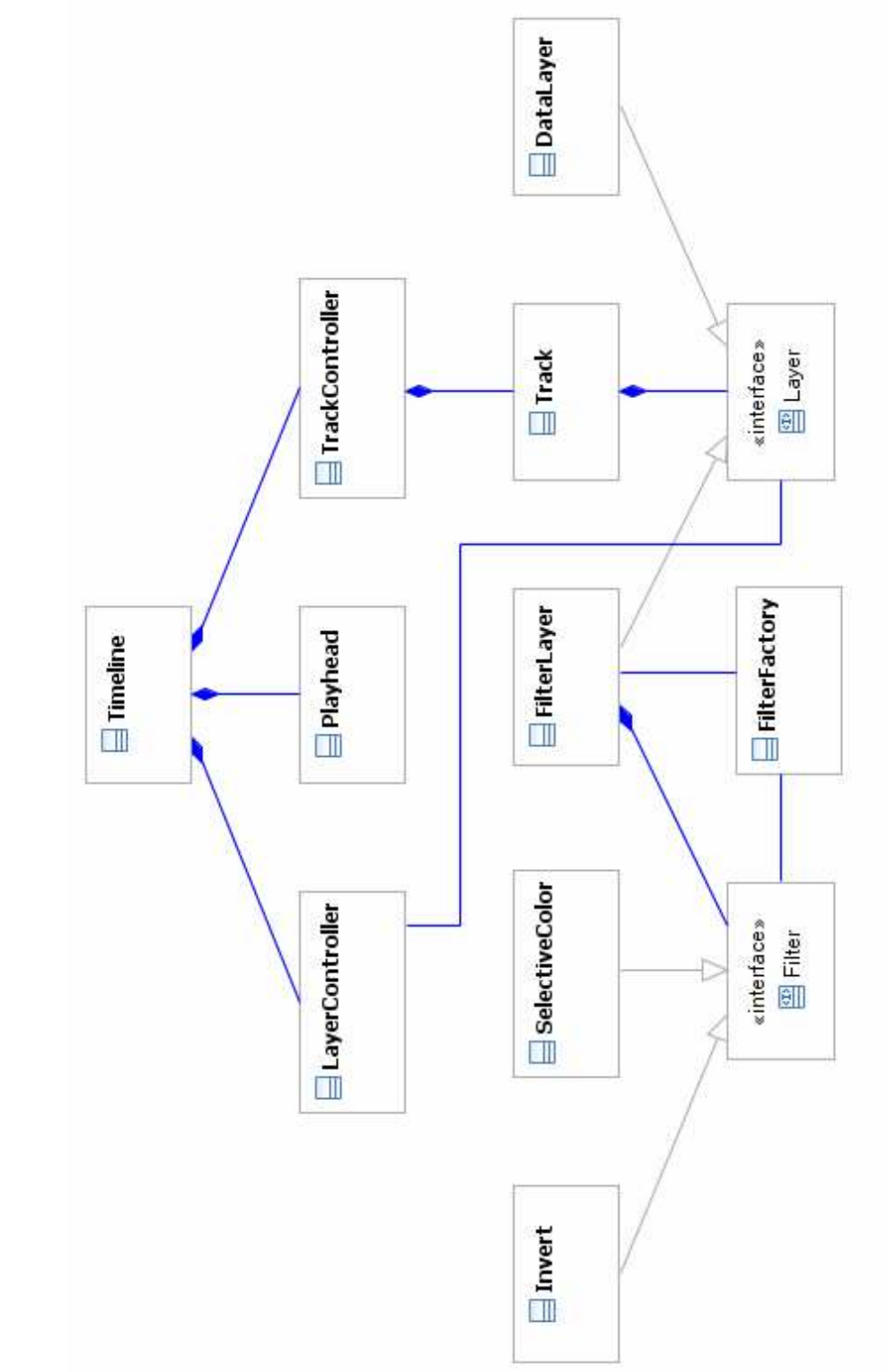
[LuMan] Autodesk Lustre 2009 User Guide [online]. [cit. 2010-05-01] http://download.autodesk.com/us/systemdocs/pdf/lustre2009_user_guide.pdf

[LrTut] Adobe TV | Learn Lightroom 2.0 [online]. [cit. 2010-05-01] http://tv.adobe.com/show/learn-lightroom-20/

[LrMan] Using Adobe Photoshop Lightroom 2 [online]. [cit. 2010-05-01] http://help.adobe.com/en_US/Lightroom/2.0/lightroom_2_help.pdf

Přílohy

Příloha A : UML Diagram



Příloha B : Některé zajímavé části kódu

Dikobraz

```
/* counts the image through all the layers
 * framePos - position of the playhead from the start of the project
*/
ComposedImage TracksController::getComposedImage(int framePos)
{
      ComposedImage cImage;
      int pixelPos = framePos / timeline->getFramesPerPixel();
      qDebug() << "pixelpos " << pixelPos;
      cImage.original = new QImage();
      cImage.corrected = new QImage();
      QList<Track*>::iterator i;
      for (i = tracks.begin(); i != tracks.end(); i++)
      {
            QWidget* layerWg = (*i)->childAt(pixelPos, 1);
            Layer* layer = qobject_cast<Layer*>(layerWg);
            if (layer == 0)
            {
                  qDebug() << "layer not found";
            }
            else {
                  if (qobject_cast<DataLayer*>(layer))
                  {
                        layer->processImage(cImage.original);
                  }
                  layer->processImage(cImage.corrected);
            }
      }
      return cImage;
}

/* moves the layer, possibly to another track
 * layer - layer to be moved
 * layersAreaPos - mouse position in the layersArea coordinates */
void LayersController::moveLayer(Layer *layer, QPoint layersAreaPos)
{
      int moveX = layersAreaPos.x() - dragStartPosInLayer.x();

      int framesPerPixel = timeline->getFramesPerPixel();
      int newFrameStart = layer->getProjectFrameStart()
            + moveX * framesPerPixel;
      int newFrameEnd = layer->getProjectFrameEnd()
            + moveX * framesPerPixel;
      QList<int>::iterator possibleLeftSnap = qLowerBound(
                  snappingFrames.begin(),
                  snappingFrames.end(),
                  newFrameStart);
      QList<int>::iterator possibleRightSnap = qLowerBound(
                  snappingFrames.begin(),
                  snappingFrames.end(),
                  newFrameEnd);
      // borders for snapping
      int leftSnapLow = layer->getProjectFrameStart() +
                  (moveX - SNAPPING_DISTANCE) * framesPerPixel;
      int leftSnapHigh = layer->getProjectFrameStart() +
                  (moveX + SNAPPING_DISTANCE) * framesPerPixel;
```

```
	int rightSnapLow = layer->getProjectFrameEnd() +
			(moveX - SNAPPING_DISTANCE) * framesPerPixel;
	int rightSnapHigh = layer->getProjectFrameEnd() +
			(moveX + SNAPPING_DISTANCE) * framesPerPixel;

	QPoint timelinePos = layer->mapTo(timeline, layersAreaPos);
	Track* pointedTrack = qobject_cast<Track*>(
		timeline->childAt(timelinePos));

	// moving to another track
	if (pointedTrack != 0 && pointedTrack != layer-
>getParentTrack())
	{
		bool moveLayer = true;
		for (int i = layer->x() + moveX; i < layer->x() + layer-
>width() + moveX; i++)
		{
			if (pointedTrack->childAt(i, 1) != 0) {
				moveLayer = false;
				break;
			}
		}
		if (moveLayer == true) {
			layer->setParent(pointedTrack);
			layer->getParentTrack()->removeLayer(layer);
			layer->setParentTrack(pointedTrack);
			layer->getParentTrack()->addLayer(layer);
			layer->show();
		}
	}

	// different layers cannot be overlapped
	for (int i = layer->x() + moveX; i < layer->x() + layer->width()
+ moveX; i++)
	{
		if (layer->getParentTrack()->childAt(i, 1) != 0 &&
			layer->getParentTrack()->childAt(i, 1) != layer)
		{
			return;
		}
	}

	// layer cannot be moved behind the borders of the timeline
	if (newFrameStart < snappingFrames.first() ||
		newFrameEnd > snappingFrames.last())
	{
		return;
	}

	// finds out possible snapping position
	if (*possibleLeftSnap != snappingFrames.first() &&
		*(possibleLeftSnap - 1) > leftSnapLow)
	{
		layer->moveToProjectFrame(*(possibleLeftSnap - 1));
		showSnapLine(layer->getProjectFrameStart());
	}
	else if (*possibleLeftSnap < leftSnapHigh)
	{
		layer->moveToProjectFrame(*(possibleLeftSnap));
		showSnapLine(layer->getProjectFrameStart());
	}
```

```
	else if (*(possibleRightSnap - 1) > rightSnapLow)
	{
		layer->moveToProjectFrame(
			*(possibleRightSnap - 1) -
			(layer->getProjectFrameEnd() - layer-
>getProjectFrameStart()) - 1);
		showSnapLine(layer->getProjectFrameEnd());
	}
	else if (*(possibleRightSnap - 1) != snappingFrames.last() &&
		*possibleRightSnap < rightSnapHigh)
	{
		layer->moveToProjectFrame(
			*possibleRightSnap -
			(layer->getProjectFrameEnd() - layer-
>getProjectFrameStart()) - 1);
		showSnapLine(layer->getProjectFrameEnd());
	}
	else {

		layer->moveToProjectFrame(newFrameStart);
		hideSnapLine();
	}

	timeline->updateViewport();
	timeline->setModified(true);
}
```

Simulator

```
/* simuluje proces pri pocitani bloku
 * rekurzivne vola pocitani pro zavisle
 * bloky, ktere jeste nejsou v pameti */
void rekurze_pocitani_bloku(blok* pocitany_blok)
{
	int vrstva = pocitany_blok->vrat_cislo_vrstvy();
	int x = pocitany_blok->vrat_x();
	int y = pocitany_blok->vrat_y();
	fprintf(log_soubor, "rekurze vr%i[%i,%i] start\n", vrstva, x,
y);
	if (pamet.obsahuje(pocitany_blok)) {
		fprintf(log_soubor, "rekurze vr%i[%i,%i] end, jiz je v
pameti\n", vrstva, x, y);
		return;
	}

	// nejdrive musi byt v pameti vsechny bloky,
	// na kterych pocitany zavisi -> rekurze
	set<blok*> zavislosti = pocitany_blok->vrat_zavislosti_dolu();
	set<blok*>::iterator i;
	for (i = zavislosti.begin(); i != zavislosti.end(); i++) {
		rekurze_pocitani_bloku(*i);
	}

	// zde by se v praxi provedl vypocet aktualniho bloku,

	// simulator ho jen prida do pameti
	pamet.vloz(pocitany_blok);

	// pripocti narocnost pocitani, tedy dobu vypoctu
	if (nekonecna_pamet.obsahuje(pocitany_blok) == false) {
		nekonecna_pamet.vloz(pocitany_blok);
```

```
            min_narocnost_sekvence += pocitany_blok->vrat_vrstvu()-
>vrat_narocnost();
      }
      narocnost_pocitani += pocitany_blok->vrat_vrstvu()-
>vrat_narocnost();

      fprintf(log_soubor, "rekurze vr%i[%i,%i] end\n", vrstva, x, y);

}

void blok::prepocitej_faktickou_narocnost() {
      fakticka_narocnost = pvrstva->vrat_narocnost();
      set<blok*>::iterator i;
      queue<blok*> fr;
      blok* cblok;
      set<blok*> zavislosti;
      // bloky z nizsich vrstev, soucet jejichz narocnosti
      // bude nase fakticka narocnost
      set<blok*> bloky_k_fakticke;

      fr.push(this);
      while (fr.empty() == false) {
            cblok = fr.front();
            zavislosti = cblok->vrat_zavislosti_dolu();
            fr.pop();
            for (i = zavislosti.begin(); i != zavislosti.end(); i++) {
                  if (pamet.obsahuje(*i) == false) {
                        fr.push(*i);
                        bloky_k_fakticke.insert(*i);
                  }
            }
      }
      // musime udelat zvlast, nebot jeden blok muze odkazovat
      // na vrstvy, ktere samy na sebe budou zavisle taky
      for (i = bloky_k_fakticke.begin(); i != bloky_k_fakticke.end();
i++) {
      fakticka_narocnost += (*i)->vrat_vrstvu()->vrat_narocnost();
      }

}
```

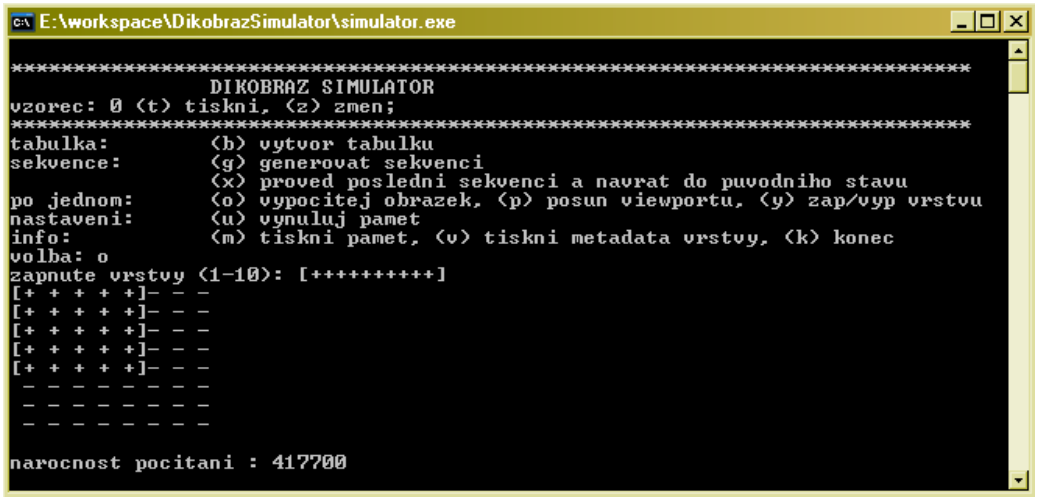
Příloha C : Uživatelská příručka Simulátoru

Program Simulátor má textové rozhraní a ovládá se intuitivně. Pro spuštění určité funkce stiskneme a potvrdíme znak v závorce. Hlavní funkcí je vytvoření tabulky s porovnáním jednotlivých algoritmů. Po stisku ‘b‘ navolíme, kolik různých sekvencí chceme do tabulky zahrnout, pro každou sekvenci se provede 12 počítání. Pokud tedy zvolíme více, než jen jednu, je třeba počítat s tím, že na výslednou podobu tabulky budeme dlouho čekat. Poté zvolíme, jaké chceme v sekvenci povolit operace.

Nyní vidíme počítání jednotlivých obrázků. V horní části je reprezentace stavu vrstev, ‘+‘ na prvním místě znamená, že vrstva 1 je zapnuta, ‘-‘, že je vypnuta, znaménka na druhé pozici reprezentují stav vrstvy 2 atd. Pod tím máme zobrazenu reprezentaci bloků nejvyšší zapnuté vrstvy. ‘-‘ znamená, že blok není v paměti, ‘+‘, že v paměti je. symboly ‘ [‘ a ‘]‘ označují, na které bloky je zrovna přiblížen viewport. Je logické, že všechny bloky na viewportu musí být po vypočítání v paměti, protože to je naším cílem.

Výsledky se vytvářejí v logovacím adresáři s datem startu programu :

popis.txt ..... popis vstupních dat, provázanost vrstev, jejich náročnosti
sekv01.txt ..... sekvence operací, nejdřív vygenerován tento soubor, poté opakovaně načítán pro vykonání různými algoritmy
p00_sekv01_0000.txt .. kontrolní průběh celého jednoho počítání obrázku, jak byly do paměti jednotlivé bloky přidávány, jak byla paměť uvolňována, na konci také stav paměti po počítání a jeho náročnost
tabulka01.txt ..... tabulka s porovnáním algoritmů, výstup programu



Příloha D : Uživatelská příručka aplikace Dikobraz

V současné verzi aplikace Dikobraz uvidí uživatel po startu Info-paletu a časovou osu. Časové ose lze měnit měřítko pomocí posuvníku vlevo nahoře, nebo pomocí tlačítek Zoom In a Zoom Out. Klikem na popisek stopy vlevo bude stopa vybrána. Nové stopy se pak přidávají pomocí tlačítka Add Track. Na časové ose pohybujeme ukazatelem za stisku levého tlačítka, nebo klikem do oblasti s časy.

Do vybrané stopy na pozici ukazatele bude po kliku na tlačítko Add File... a vybrání souboru z disku vložena nová datová vrstva. Klikem na novou vrstvu ji vybereme, klikem za současného držení klávesy Shift je přidána do vybraných. klikem mimo vrstvu vybrání zrušíme. Vrstvu odstraníme pravým klikem a vybráním Remove nebo stiskem klávesy Delete.

Vrstvy je možno za stisku levého tlačítka posunovat nebo měnit jejich velikost. Pokud tlačítko stiskneme na kraji vrstvy, měníme velikost, jinak posunujeme. Vrstvu lze posunovat i napříč stopami, není však možné vrstvu přemístit na pozici jiné vrstvy. Jak při posunování, tak při měnění velikosti jsou zobrazovány vodící linky k přilnutí k dalším vrstvám na časové ose.

Pokud je ukazatel na pozici s datovou vrstvou, je obrázek zobrazen na viewportu. Přejížděním myši přes obrázek se na Info-paletě zobrazují hodnoty jednotlivých pixelů, vlevo po korekci, vpravo hodnoty originálu.

Nová filtrovací vrstva se vždy přidává do stopy nad právě vybranou vrstvu. Tlačítko Add Layer je tak zpřístupněno jen když je nějaká vrstva vybrána. Pokud máme vybráno více vrstev, je nová filtrovací vrstva roztáhnuta přes všechny vybrané vrstvy. Do filtrovací vrstvy lze přidávat filtry pomocí tlačítka Add Filter a klíčové snímky pomocí tlačítka K. Klíčové snímky se ukládají na pozici ukazatele a jsou znázorněny červeným trojúhelníkem v horní části vrstvy.

Na vrstvě je filtr reprezentován ikonou. Každý typ filtru ji má jedinečnou. Po kliku na ní se otevře okno s nastavením parametrů filtru. Každý filtr má tlačítko Preview, kterým je možno filtr dočasně deaktivovat a na viewportu nezobrazovat jeho výstup.