Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Akcelerace zpracování HDR rastrového obrazu na GPU

Diplomová práce

Autor: Bc. Miroslav Kořínek

Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Bruno Ježek, Ph.D.

Hradec Králové 29. Dubna 2016

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

vlastnoruční podpis

V Hradci Králové dne 30.4.2016 Miroslav Kořínek

Poděkování:

Děkuji vedoucímu diplomové práce Ing. Brunovi Ježkovi, Ph.D. za metodické vedení práce a odbornou pomoc.

Anotace

Práce se zabývá podrobným rozebráním problematiky obrazu s vysokým dynamickým rozsahem. Dále se zabývá implementací dostupných metod pro zpracování obrazů HDR v reálném čase s využitím grafického hardware a jejich následným otestováním na vybraném datovém souboru.

Annotation

Title: GPU accelerated HDR image processing

This thesis is aimed at images with high dynamic range and evaluating different methods for creating HDR images and rendering them in realtime with graphics hardware on conventional displays. These methods will be implemented and tested on different data files.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Okolní svět je pozorován na základě odraženého světla, které je vnímáno lidským zrakem. Rozsah intenzit jasu, který je schopné lidské oko zachytit je dán zrakovou křivkou pro vnímání světelných podnětů. Vždy je ale možné pozorovat pouze určitou část rozsahu intenzit světla v reálné scéně. Prahová hodnota, určující viditelný rozsah, se mění v závislosti na předešlých podmínkách osvětlení. Například jestliže člověk vstoupí do temné místnosti z osvětleného prostoru, není nejprve schopen vnímat žádný světelný podnět, postupně se však zrakové vnímání zlepšuje, oko se adaptuje a umožní rozpoznat i velice nízké intenzity. Rozsah intenzit, které dokáže zachytit běžný fotoaparát, je ještě více limitován. Technologie obrazu s vysokým dynamickým rozsahem jasových hodnot neboli HDR (High dynamic range) představuje způsob napodobení funkcí adaptace lidského oka. S využitím speciálních postupů a zařízení dokáže digitálně zachytit celý světelný rozsah jakékoliv scény.

V dnešní době je technologie HDR stále více rozšířená a to zejména u digitálních fotografií. Protože se jedná o statické obrazy, neklade se takový důraz na rychlost jejich zpracování. Tato práce ale nabízí postupy podporující zpracování v reálném čase s důrazem na využití běžně dostupných hardwarových zařízení. To umožňuje zobrazovat HDR data prakticky okamžitě a otevírá nové možnosti využití takových dat např. v podobě HDR videa nebo zobrazování dynamické scény ve vysokém dynamickém rozsahu v reálném čase.

Práce je rozdělena celkem do pěti kapitol. První kapitola popisuje samotný problém reprezentace jasových hodnot reálných scén v digitální technologii a formáty obrazů. Jsou uvedeny formáty pro ukládání obrazů s nízkým (LDR) i vysokým dynamickým rozsahem (HDR). Druhá kapitola se zabývá mapováním tonality HDR obrazů a uvádí různá dostupná řešení tohoto problému jak na globální tak i na lokální úrovni. Třetí kapitola je zaměřena na zpracování HDR dat v reálném čase. Obsahem čtvrté kapitoly je návrh řešení a implementace jednotlivých metod, uvedených v předchozích kapitolách, včetně popisu

jednotlivých aplikací a jejich grafického rozhraní. V páté kapitole je provedeno srovnání výsledků uvedených metod na různých datových souborech.

Cíl práce

Cílem práce je prozkoumat, implementovat a otestovat metody pro zpracování sekvence obrazů s vysokým dynamickým rozsahem v reálném čase. Budou popsány vlastnosti HDR obrazů, jejich získávání a způsoby jejich datové reprezentace. Popsané metody budou implementovány, testovány a zhodnoceny na vytvořených datových souborech a bude popsáno jejich možné využití v praxi. Pro zobrazení HDR obrazů na LDR zařízení budou použity metody mapování tónů s důrazem na co nejkratší čas zpracování za použití výkonu grafických karet.

1 Technologie obrazu HDR

V následující kapitole je popsán úvod do problematiky vysokého dynamického rozsahu, jak takový rozsah zachytit, uchovat a následně zobrazit na běžně dostupných monitorech.

1.1 Dynamický rozsah

Dynamický rozsah v digitální fotografii je definován jako poměr mezi hodnotou jasu nejtmavšího a nejsvětlejšího pixelu v obraze. Hodnota dynamického rozsahu se uvádí jako poměr odpovídajících jasů pixelů, například 1 : 100. Protože se ale na snímku často vyskytují i pixely s nulovou hodnotou jasu, musí se pro správné určení dynamického rozsahu vycházet z minimální a maximální hodnoty jasu celého snímku, a nejmenšího kontrastu. Ten je dán poměrem minimální hodnoty snímku, a hodnoty, která se liší o nejmenší rozpoznatelnou hodnotu jasu. Například pokud je minimální hodnota jasu snímku a jiné místo s nejmenším nárůstem jasu má hodnotu 0,15, bude rozdílem určena základní jednotka kontrastu snímku, tj. 0,15. Pokud je maximální hodnota jasu na snímku dvěstěkrát vyšší než je základní kontrast (pro uvedený příklad by bylo maximum 30), lze říci, že dynamický rozsah snímku je 1 : 200.

1.1.1 Expoziční hodnota

Kontrastní poměr lze také vyjádřit ve stupních expozice (EV), které snímek zachycuje. Převod mezi expoziční hodnotou a kontrastním poměrem vypadá následovně:

$$2^{(\mathrm{EV})} = \text{kontrastní poměr} \tag{1}$$

EV hodnota rovnající se nule je definována mezinárodní standardizační organizací (ISO) jako expoziční hodnota při cloně f/1 a expozičním čase 1 sekunda. Změna expozice o jeden stupeň odpovídá redukci množství světla na polovinu, respektive navýšení na dvojnásobek. Na fotoaparátech bývá často symbol EV +/- jako ovládací prvek pro změnu této hodnoty.

1.1.2 Reálný dynamický rozsah

Při fotografování reálných scén dopadají na senzor fotoaparátu paprsky světla. Dynamický rozsah senzoru je však oproti lidskému oku velmi limitován. Lidské oko umožňuje globální i lokální adaptaci. Christian Bloch [3] ve své knize uvádí, že lidské oko dokáže zobrazit kontrastní poměr až 1 : 10 000. Dokáže se navíc přizpůsobit globálním změnám světla a rozšířit vnímatelný rozsah až na 1 : 1 000 000 000. Globální změny probíhají například při přechodu z tmavé místnosti do otevřeného prostředí za slunečného počasí. Nedokáže ale takovéto extrémní rozdíly pozorovat najednou. Oči většinou potřebují nějakou dobu, než se adaptují na změnu jasu.

1.1.3 Dynamický rozsah digitální fotografie

Digitální fotoaparát, respektive snímací senzor, pracuje podobně. Množství světla, které dopadne na senzor, lze ovlivnit pomocí clony, délky expozice a citlivostí snímače. Pomocí těchto proměnných je možné napodobit globální adaptaci lidského oka. S lokální adaptací už si ale většinou snímač nedokáže poradit. Oko využívá lokální adaptaci pro rozpoznávání různých částí zorného pole a podle intenzity světla dokáže nastavit různou citlivost svých receptorů. Správnou viditelnost zajišťují různé receptorové buňky v sítnici, které se stávají citlivějšími v závislosti na intenzitě světla ve scéně. Tento druh adaptace se dá v běžné fotografii jen těžko napodobit.

Jelikož dynamický rozsah reálných scén může být až 1 : 1 000 000 000 nebo i více, při fotografování je nutné zvolit, která část celého rozsahu bude zachycena na fotografii. Vybraná část rozsahu je označena jako okno expozice. Pomocí nastavení snímače na fotoaparátu, lze oknem expozice pohybovat v dynamickém rozsahu snímané scény. Šířku okna ale ve většině případů nelze měnit. Je limitována nejen technickými parametry snímače, ale v praxi i výstupním formátem pro ukládání snímku. Protože výsledný obraz nenese celý dynamický rozsah fotografované scény, označuje se formát jako LDR (Low dynamic range).

1.1.4 Vysoký dynamický rozsah

Technologie HDR (High dynamic range) se zabývá rozšířením okna expozice na pokud možno celý dynamický rozsah snímané scény. K získání takového obrazu existují různé techniky. Například speciální technická zařízení pro HDR snímání, které umožňují přímé zachycení HDR snímku. Druhou možností je technika skládání HDR obrazu pomocí sady LDR snímků. Nasnímaná sada snímků stejné scény pořízená s různými expozičními hodnotami je následně různými metodami spojena do jednoho výsledného HDR obrazu tak, že se z každého snímku použijí pouze jeho správně exponované části.

1.2 Vytváření HDR

1.2.1 Přímé snímání

Snímání obrazu s vysokým dynamickým rozsahem je možné provést přímým snímáním pomocí speciálního snímače s vysokým rozsahem. Princip spočívá v tom, že optický člen kamery rozdělí vstupní paprsky světla pomoci zrcadlového jehlanu na několik svazků. Každý ze svazků světla pak dopadá na běžný 8bitový senzor. Pomocí různých filtrů se docílí rozdílných časů expozice na každém senzoru. [1] Při použití trojbokého jehlanu lze získat poměr časů expozice například 1 : 2 : 4, tedy tři snímky s rozdílem jeden expoziční stupeň.

1.2.2 Fotografování série expozic

Získat HDR obraz je možné i nepřímo. Jednu z metod popisují autoři Debevec a Malik ve svém článku. [5] Základem metody je skutečnost, že vyfotografování jedné expozice nedokáže vždy pokrýt celý dynamický rozsah snímané scény. Příkladem může být scéna vycházejícího slunce.

Obrázek 1 Dva snímky stejné scény s rozdílnými časy expozic. Na každém snímku je zachycena jiná část dynamického rozsahu.

Zachycení všech detailů na jedné fotografii není možné. Na to běžný formát nestačí. Řešením je vyfotografovat sérii fotografií stejné scenérie. Každá fotografie bude mít jinou expoziční hodnotu. Fotoaparát by měl být nastavený tak, aby první fotografii se scénou co nejvíce podexponoval a zachytil tak detaily ve tmavých oblastech. Na fotografii budou světlé oblasti přepálené, protože rozsah světlých hodnot přesáhl maximální hodnotu. Na dalších fotografiích by měl mít fotoaparát nastavenou expoziční hodnotu vždy o jeden až dva stupně víc. Výsledkem je série expozic s rozdílem 1 - 2 expoziční hodnoty a na každém snímku je zachycená jiná část dynamického rozsahu scény. Jednotlivé části rozsahu by se měli vzájemně překrývat. Při nedodržení maximálního rozdílu expoziční hodnoty mezi snímky nedojde k průměrování hodnot z více než jednoho snímku a na výsledném obrazu bude nežádoucí šum.

1.2.2.1 Nastavení fotoaparátu

Aby výsledek skládání expozic byl co nejpřesnější, je potřeba správně nastavit fotoaparát. Základem je vypnutí všech speciálních funkcí pro vylepšení obrazu, které fotoaparát podporuje. Ne vždy je jasné, jak by tyto funkce ovlivňovaly výsledný snímek v různých stupních expozice. Dále je potřeba nastavit fotoaparát na manuální režim focení, kde jedinou proměnnou složkou bude čas. Změna clony by měnila hloubku ostrosti fotografie a změna citlivosti ISO zase šum. To by vedlo ke vzniku artefaktů v obraze. Některé fotoaparáty podporují funkci auto - bracketing, která automaticky vyfotí sérii tří snímků s rozdílnými expozicemi. Je potřeba si dát pozor, jestli tato

funkce mění pouze hodnotu času. Často totiž mění všechny tři složky. Další důležitou podmínkou je, aby byla poloha fotoaparátu i směr snímání fixován například stativem, a při snímání nedošlo ke změně ohniskové vzdálenosti objektivu. Malé změny polohy snímače je možné později eliminovat zarovnáním jednotlivých obrazů, ale stabilní uchycení situaci zjednodušuje.

1.2.2.2 Spojování obrazů

Získané obrazy pořízené za různých expozičních podmínek je možné sestavit do výsledného HDR obrazu pomocí schéma osvícení [23] . Hodnotu každého pixelu v obraze HDR lze určit vydělením hodnot pixelů expozičním časem ze všech vstupních LDR obrazů [5] . Což je možné pouze za předpokladu, že odezva senzoru digitálního fotoaparátu je dokonale lineární. HDR obraz pak vznikne zprůměrováním hodnot odpovídajících pixelů při vyloučení přeexponovaných a podexponovaných pixelů podle rovnice (3),

$$\mathrm{L_i} = \sum_{\mathrm{j=1}}^{\mathrm{j=N}} \frac{\mathrm{Z_{ij}w(Z_{ij})}}{\Delta \mathrm{t_j}} / \sum_{\mathrm{j=1}}^{\mathrm{j=N}} \mathrm{w(Z_{ij})} \tag{2}$$

kde L_i je hodnota pixelu v HDR obrazu, N je počet snímků vstupujících do výpočtu, Z_{ij} je hodnota pixelu ij-té fotografie, t_j je expoziční čas j-té fotografie,$w(Z_{ij})$ je váha pixelu i v j-tém snímku. Nastavení váhy jednotlivých jasových hodnot je možné řešit pomocí jednoduché váhové funkce určené vztahem (3), který je založený na předpokladu, že středně exponované pixely mají mít větší váhu než pixely blízko obou konců rozsahu.

$$\begin{aligned} \mathrm{w(z)} &= \mathrm{z - Z_{min}} \text{ pro } \mathrm{z} \leq \frac{1}{2}(\mathrm{Z_{min} + Z_{max}}) \\ \mathrm{w(z)} &= \mathrm{Z_{max} - z} \text{ pro } \mathrm{z} \geq \frac{1}{2}(\mathrm{Z_{min} + Z_{max}}) \end{aligned} \tag{3}$$

Z_{min} a Z_{max} jsou hraniční hodnoty vstupních pixelů a z je hodnota aktuálního pixelu. Při zpracování barevného tří-kanálového obrazu se zpracovávají jednotlivé kanály odděleně.

1.2.3 Pseudo HDR fotografie

Technika pseudo HDR vychází pouze z jednoho LDR snímku. Vytvoříme dvě nebo více kopií tohoto snímku a každou kopii ručně zesvětlíme, respektive ztmavíme, abychom napodobili odpovídající posun expozičních hodnot +/- 2EV. Z takto vytvořených snímků můžeme složit nepravý obraz HDR. Problém nastává při technice mapování tónů, která je detailněji popsaná v podkapitole 1.4. Výsledkem mapování tónů je mimo jiné právě zvýraznění detailů ve tmavých částech fotografie. Při mapování pseudo HDR, ale tyto detaily úplně chybí, protože se v původním LDR snímku nevyskytovali a v obraze se poté v těchto oblastech zobrazí pouze šum.

Obrázek 2 Porovnání šumu v obraze HDR a Pseudo-HDR [13]

1.3 Reprezentace a uchování HDR dat

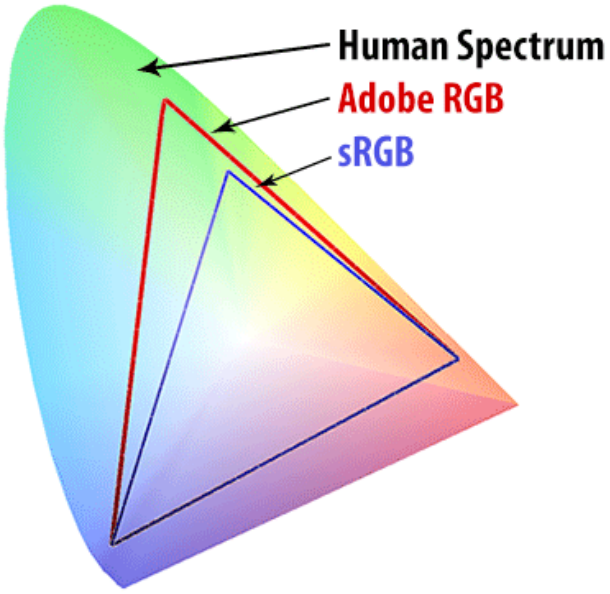
HDR i LDR obraz získaný z digitálního snímače je reprezentován rastrem, který jednotlivé obrazové elementy označované jako pixely uspořádává do pravidelné mřížky. V následující kapitole budou popsány obrazové rastrové formáty ukládající pro každý pixel jeho jas a v případě barevného obrazu i jeho barevný tón.

Barevný model

K uchování informací o barvě se nejčastěji používá míchání základních barev podobně, jako tomu je v reálném světě, kde lidské oko zobrazuje barvy na základě míchání záření světla o různých vlnových délkách. Nejčastějším barevným modelem, využívajícím míchání barev, je model RGB. Využívá aditivní míchání barev, kde se jednotlivé složky sčítají. Pracuje se třemi základními barvami:

červená, zelená, modrá. Ostatní barvy jsou dány různou sytostí těchto barev a jejich následným složením. Rozsah barev lze pozorovat na obrázku (3).

Alternativou k tomuto modelu je model HSV (Hue, Saturation, Value). Má tři základní parametry: odstín, sytost a jas. Používání modelu HSV v technologii HDR přináší výhodu oproti RGB modelu, která spočívá v oddělené jasové složce od samotné barvy. Při mapování tonality obrazu, která je podrobněji popsána v podkapitole 1.4, stačí pouze měnit jasovou složku, na rozdíl od RGB modelu, kde je nutné měnit hodnotu všech tří základních barev.



Obrázek 3 Barevný rozsah při míchání RGB barev [20]

Kódování hodnot rastru

Barevná hloubka souvisí se zvoleným kódováním obrazu. Jedná se o počet bitů na jeden pixel, neboli kolika bity bude reprezentovaná barva a jas každého pixelu. Barevný model RGB ve standardu sRGB (Obrázek 3) využívá nejčastěji k uložení jednoho pixelu 24 bitů. Tyto bity jsou rovnoměrně rozděleny pro každou základní barvu. Na každou barvu spadá 8bitů, to je 256 různých hodnot. Ve výsledku lze uložit 16 777 216 různých barev. Pro změnu jasu v RGB modelu se musí změnit všechny tři základní složky barevného modelu o stejnou hodnotu, jinak by došlo ke změně odstínu barvy. Z toho vyplývá, že dynamický rozsah klasického snímku je ještě menší než 1 : 256. Kromě nízkého dynamického rozsahu je problém i

celočíselná reprezentace hodnot. Neexistuje zde pixel, který by měl hodnotu mezi číslem 1 a 2. Hodnota takového pixelu musí být v celočíselném formátu zaokrouhlena. Řešením je uložení hodnot jednotlivých barevných kanálů pomocí kódování hodnot v plovoucí řádové čárce a s větší bitovou hloubkou. Volba kódování závisí na zvoleném formátu souboru. Každý formát podporuje různé barevné modely i kódování.

1.3.1 Formát JPEG

Jedním z nejpoužívanějších formátů pro ukládání LDR snímků je formát JPEG. Formát používá naprostá většina fotoaparátů. Pro ukládání rastru používá barevný prostor RGB ve standardu sRGB se 24 bitovou barevnou hloubkou. Využívá ztrátovou kompresi DCT. Kvůli ztrátovosti a omezené barevné hloubce není JPEG formát pro HDR obrazy příliš vhodný.

1.3.2 Formát radiance

Řešení nízké barevné hloubky nabízí formát radiance s příponou hdr, používající barevný model RGB a kódování RGBE. Je to 32 bitový formát, který stále používá 8 bitů na barevný kanál. Navíc obsahuje dalších 8 bitů pro uložení exponentu. Tento exponent v sobě uchovává jasovou informaci. Skutečná hodnota pixelu je vypočítána podle vzorce (4).

$$\frac{(\mathrm{R}, \mathrm{G}, \mathrm{B})}{255} \times 2^{(\mathrm{E}-128)} \tag{4}$$

Exponent umožňuje opravdu velké rozdíly jasu. Na příkladu je vidět, jak výrazně ovlivňuje výslednou hodnotu:

$$\begin{aligned} \frac{(100;\ 100;\ 130)}{255} \times 2^{(150-128)} &= (1640000;\ 16400000;\ 2140000) \\ \frac{(100;\ 100;\ 130)}{255} \times 2^{(115-128)} &= (0{,}0000479;\ 0{,}0000479;\ 0{,}0000622) \end{aligned} \tag{5}$$

Výhodou formátu je i jednoduchá implementace a na převody hodnot nejsou potřeba speciální knihovny.

Formát dokáže zachytit až 253 expozičních stupňů. Velký dynamický rozsah není většinou v praxi využit a kódování exponentu obsahuje velké množství nulových bitů. Pro snížení datové náročnosti je použita RLE komprese dat, kterou formát radiance podporuje.

1.3.3 Formát OpenEXR

Formát OpenEXR je novější formát než formát radiance. Vyvinula ho společnost Industrial Light&Magic (ILM) v roce 2000. Je založený na kódování Half RGB využívající speciální datový typ Half, který zaokrouhluje 32 bitové číslo s pohyblivou řádovou čárkou pro každý barevný kanál na 16 bitů. Celkem tedy potřebujeme 48 bitů na pixel. 16 bitové číslo je rozděleno na jeden znaménkový bit, 10 bitů mantisy a 5 bitů exponent. Základ čísla umožňuje uložit $2^{10} = 1024$ hodnot. Při vynásobení všech tří barevných složek dostaneme více než 1 miliardu barev nezávislých na expozici. O jasovou složku se pak stará exponent, který nám poskytuje $2^5 = 32$ expozičních hodnot. Oproti zmíněnému JPEG formátu nabízí opět mnohem přesnější reprezentování hodnot pixelů.

1.3.4 Formát Tiff

Formát TIFF (Tagged Image FileFormat) je jedinečný v tom, že obrazová data jsou zabalena do popisných značek (tagů), díky kterým se tento formát stává univerzálním. Značky umožňují zadat například počet bitů na pixel (1 až 32 bitů) nebo použitý barevný prostor. Jeho hlavní výhoda je ale zároveň i jeho hlavní nevýhoda. Při načítání fotografie s příponou tiff by měla aplikace počítat se všemi funkcemi, jaké formát podporuje. Dopředu totiž není jasné, jaké metody budou v konkrétním souboru použity. Podporovaných funkcí je řada a často nejsou všechny algoritmy pro kompresi dat implementovány. Z tohoto důvodu není formát TIFF příliš využívaný, ale k uchování HDR obrazů ho lze využít.

1.3.5 RAW

Formáty typu RAW se používají jako výstupní formáty u kvalitnějších digitálních fotoaparátů. Existuje mnoho variant těchto formátů, protože každý výrobce fotoaparátů má svůj vlastní typ formátu RAW a není tedy možná nějaká

jednoduchá jednotná konverze. Fotoaparáty od společnosti Canon používají formát CR2. K extrahování hodnot jednotlivých pixelů z formátu CR2, je ale zapotřebí detailně znát specifikaci formátu, které se navíc liší pro jednotlivé modely fotoaparátů. Pro extrakci dat je možné použít některá z řešení od Davida Coffina [4] , který na svých webových stránkách poskytuje volně dostupné dekodéry pro nejznámější formáty RAW.

Formát RAW dokáže zachytit větší dynamický rozsah než 24bitové formáty. Zachycený dynamický rozsah ale není dost velký natolik, aby se formáty daly označovat jako pravé HDR. Nicméně jejich účelem je uchovávat obrazová data v původní nasnímané formě. Bez žádné komprese dat nebo úprav barev.

1.3.6 HDRJpeg

Ke čtení HDR obrazů je většinou zapotřebí speciální software, který daný formát podporuje. Formát HDRJpeg tuto nutnost odstraňuje. Proces ukládání formátu probíhá tak, že se HDR obraz transformuje na obraz LDR a k němu se vytvoří druhý dodatečný snímek, obsahující pouze dodatečné informace, potřebné k jeho rekonstrukci. Dodatečný snímek je uložený v metadatech, takže standardní aplikace tento atribut ignorují, a zobrazují pouze první mapovaný LDR obraz. Formát dokonce není závislý pouze na jedné metodě, ale autoři uvádějí, že technik mapování tónů je pro tento formát vhodných více. [18]

1.4 Zobrazování HDR obrazů

Schéma osvícení obrazu [23] reprezentuje každý pixel jako tří-složkový vektor RGB složek, kde každá hodnota je uložena jako číslo v intervalu [0 - ∞). Výslednou barvu není možné přímo zobrazit na běžných, výstupních zařízení s limitovaným rozsahem. Pro zobrazení HDR obrazu na běžných monitorech je potřeba vstupní HDR obraz transformovat zpět na obraz LDR. Pro správné zobrazení takového obrazu se využívá mapování tónů. Různé techniky byly prezentovány v mé bakalářské práci [12] . Základní technikou mapování je lineární transformace, kde se pouze lineárně smrští libovolný rozsah hodnot na hodnoty v intervalu LDR. Byly ale uvedeny i složitější techniky, které například dokáží z charakteristik

obrazu určit jeho celkovou světlost a při transformaci upravit křivku tak, aby nebyla zcela lineární a tím posílit tmavé nebo naopak světlé oblasti v obraze. Každá technika podávala různé vizuální výsledky a vyžadovala odlišný čas zpracování. Protože výpočet probíhal pouze na procesoru počítače, zpravidla bylo i pro relativně malé obrazy zapotřebí až desítky sekund pro dokončení transformace a vykreslení na monitor. Pro vykreslení obrazů v reálném čase je proto nutné zvolit odlišný způsob zpracování jednotlivých pixelů, jako je například paralelní zpracování, kterým se tato práce zabývá v kapitole 2, kde jsou vhodné metody transformace HDR dat na data LDR vybrány a popsány.

2 Zobrazování HDR dat v reálném čase

Při renderování HDR dat je nutné zajistit několik výpočetně náročných operací, než se obraz úspěšně vykreslí na obrazovce. Pokud všechny operace vykonává CPU, budou potřeba i při běžném rozlišení snímku desítky vteřin k dokončení celého procesu. Důvodem je sériové zpracování jednotlivých pixelů, kterých je například při full HD rozlišení přibližně 2 miliony. A při vykreslování obrazu je nutné tyto pixely projít několikrát. Navíc se nejedná jako v případě 8bitové hloubky o celočíselné operace, protože se pracuje s čísly s plovoucí řádovou čárkou.

Pokud by bylo dosaženo zpracování a vykreslení obrazů ve frekvenci alespoň 24 snímků/s, výsledkem by byl plynulý obraz a HDR technologie by se dala využít i pro procesy probíhající v reálném čase. To ale znamená, že na zpracování jednoho snímku spadá zhruba 41 milisekund a to je při sériovém zpracování opravdu velmi málo. Drobné zlepšení nabízejí více-jádrové procesory. Obraz lze poté rozdělit na bloky a poté je paralelně zpracovat. Aktuální počty jader jsou ale na zpracování velkých obrazů stále malé. Mnohem lepší řešení nabízí grafické karty, které jsou na paralelní zpracování lépe připraveny.

2.1 Paralelní výpočty na GPU

Průběh výpočtů na grafických kartách probíhá jinak, než jak je tomu na CPU, kde je architektura uzpůsobená tak, aby efektivně zpracovávala dlouhou sekvenci instrukcí za sebou [10] . Odlišný způsob zpracování dat na GPU vyniká ve zpracování objemových dat paralelně. Příčinou je architektura grafických karet, které jsou složené z několika výpočetních jednotek, kterých se na kartě nachází desítky až stovky. Každá z těchto jednotek vykonává vlastní vlákno procesu. Výpočetní rychlost jednoho vlákna je nižší, než vlákno zpracovávané na CPU, ale díky množství těchto jednotek dokáží mnohem rychleji zpracovat velké objemy datových polí, kde na každý prvek z datového pole se aplikují stejné instrukce. Příkladem může být právě zpracování pixelů v obraze, kde se každý pixel zpracuje podle stejného algoritmu.

Vývoj grafických karet se snaží o efektivní implementaci vizualizačního řetězce. Zobrazení základních primitivních geometrických objektů, jako jsou třeba trojúhelníky, je rozděleno do několika navazujících kroků, které jsou na grafické kartě efektivně implementovány. Navíc některé kroky jsou prováděny na programovatelných jádrech, označované shadery, které mohou být řízeny vlastními shaderovými programy, zkráceně také shadery. Protože shadery zastupují různé funkce vizualizačního řetězce, obsahují programový kód pro zpracování různých entit. Vertex shader řeší zpracování vrcholů, především jejich transformaci. Pro každý vrchol, který je potřeba vykreslit se spustí jeden vertexový shaderový program. Fragment shader naproti tomu pracuje s výsledným rasterizovaným 2D obrazem a kód shaderu je aplikován pro každý jednotlivý pixel výstupního obrazu. Protože se počet vrcholů nebo pixelů může v různých scénách lišit a zpracování scény může vyžadovat jiný podíl výpočetních jednotek pro provádění vertexových nebo fragmentových operací, jsou shadery unifikované. Pokud je vykreslovaná scéna náročná na počet vykreslovaných primitiv, je alokováno více výpočetních jednotek programům vertex shaderu. Pokud je naopak ve scéně primitiv málo, ale vykresluje se ve vysokém rozlišení, přiřadí se jednotky fragment shaderu. Jelikož je ale jednotek málo na to, aby mohly najednou paralelně zpracovat například výstup do HD rozlišení, které obsahuje miliony pixelů, probíhá na GPU virtualizace, která zajistí efektivní zpracování vláken, jejichž počet několikanásobně převyšuje počet výpočetních jednotek na kartě.

2.2 OpenGL

Vizualizační řetězec (pipeline) využívající programovatelné shadery je označován jako programovatelný vizualizační řetězec. Jeho použití a programování grafických karet je standardizováno grafickou knihovnu OpenGL. Jedná se o multiplatformní rozhraní pro tvorbu aplikací počítačové grafiky, které se používá k vykreslení základních grafických primitiv. Knihovna dále podporuje různé pokročile grafické funkce, jako jsou například míchání barev, osvětlení, stínování, texturování ploch apod. Od verze OpenGL 2.0 je součástí i programování shaderových programů. Knihovna OpenGL definuje jazyk GLSL, který vychází se syntaxe jazyka C a používá se pro všechny typy shaderů.

Pro uložení rastrového obrazu knihovna OpenGL podporuje řadu typů textur, které se liší způsobem reprezentace barvy nebo intenzity pixelů. Pro HDR technologii je nejdůležitějším parametrem bitová hloubka. Základním způsobem reprezentace je 8 bitová hloubka pro každý barevný kanál, kterou lze rozšířit na 16 (vhodné pro formát half) nebo až na 32 bitů. Grafické karty ale jednotlivé barevné hodnoty zpracovávají vždy v plovoucí řádové aritmetice a jsou tak připraveny na renderování ve vysokém dynamickém rozsahu. Je ale potřeba mít na paměti, že OpenGL vždy provede transformaci vstupních hodnot textury na rozsah <0 ; 1>.

Výstupem z fragment shaderu je 2D obraz, uložený v obrazovém bufferu (framebufferu) [9] , který je ve výchozím nastavení přímo vykreslen na obrazovku. Výstup z fragment shaderu lze uložit i do obrazového bufferu reprezentovaného pomocí textury. Tato technika se nazývá vykreslování do textury. V podstatě se výsledný obraz uloží do nově vytvořené textury a lze ji následně použít jako klasickou texturu pro další krok vykreslovacího řetězce a tedy i jako vstup do fragment shaderu. I zde lze použít 32bitovou hloubku, takže neztratíme dynamický rozsah.

2.3 Mapování tónů na GPU

Při zobrazení HDR obrazu na LDR výstupním zařízení je výpočetně nejnáročnější proces mapování tónů. Pomocí paralelního zpracování pixelů ve fragmentovém procesoru lze ale docílit rychlejšího zpracování. Protože se při zobrazení rastrového obrazu jedná v podstatě o vykreslení 2D objektu s mapovanou texturou, není pro zpracování HDR dat vertex shader nijak významný. Pro vykreslení snímku stačí vykreslit obdélník a bez nutnosti úpravy programu vertex shaderového programu. Samotné zpracování pixelů HDR obrazu proběhne až ve fragment shaderu.

Program napsaný pro fragment shader, se vykoná pro každý pixel obrazu zvlášť. Aby ale došlo k úspěšné transformaci jasových hodnot na LDR rozsah monitoru, je potřeba do shaderu dodat informace o globálních vlastnostech obrazu a to i v případě lokálních technik. Jednou z možností je vypočítat tyto parametry na CPU před tím, než je textura odeslána na GPU. Vzhledem k tomu, že u

některých výpočtů, např. určení celkové průměrné hodnoty jasu v obraze, je nutné projít úplně všechny pixely, jen stěží bychom dosáhli požadovaného časového limitu na jeden snímek.

Druhou možností je vypočítat tuto hodnotu ve fragment shaderu v jednom průchodu vizualizačním řetězcem a výsledek uložit do cílového framebufferu o velikosti 1x1 pixel. Program ve fragment shaderu projde celou vstupní texturu a zjistí požadované informace, které uloží do výstupního pixelu. Výpočet je sice přenesen na GPU a zdánlivě urychlen. Tím, že je ale celý průchod implementován jako sekvenční průchod prováděný v jednom shaderu, není GPU paralelismus vůbec využit.

Třetím řešením je pyramidové prohledávání obrazu [17] . Při využití techniky vykreslování z textury do textury pyramidovým způsobem, kdy při každém dalším kroku je šířka i výška textury zredukována na třetinu s tím, že v shaderu se pro dané okolí pixelu o velikost 3x3 vypočte aritmetický průměr hodnot a uloží se na výstupní pixel. Tato operace se opakuje tak dlouho, dokud není textura natolik malá, že se výpočet vyplatí provést v jednom shaderu pro celý zbytek textury (Obrázek 12)

Požadavkem při zpracování barevného obrazu s vysokým dynamickým rozsahem je také zachování původních barev. Pokud je obraz uložen ve formátu, kde je jasová složka oddělené od barev, aplikuje se mapování tónů pouze na jasovou složku a sytost barev by měla zůstat zachována. V případě RGB modelu, který OpenGL podporuje jako výchozí barevný formát, se musí obraz před samotným mapováním vhodně dekomponovat. Dekompozice obrazu spočívá v oddělení jasové složky od barev tak, že se hodnotou jasu vydělí všechny barevné kanály zvlášť. Samotná jasová složka se vypočte dle vzorce (6). A protože je možné zvolit výstupní formát textury jako RGBA, může se využít volný alfa kanál právě pro uložení jasové složky a dekomponovaných barevných kanálů.

$$L = R * 0.2989 + G * 0.5866 + B * 0.1145 \quad (6)$$

Zpětný proces probíhá jednoduchým vynásobením jasové hodnoty s barevným kanálem.

$$C_{RGB} = C_{RGB}/L \tag{7}$$

Při tomto procesu může dojít k přetečení LDR rozsahu, proto je nutné hodnoty ořezat. To může způsobit změnu charakteru barev, a proto se do této transformace přidává možnost úpravy gama křivky tak, jak je to znázorněno v rovnici (8), kde parametr *G* udává hodnotu gama korekce.

$$L = C^{G} * L_{compressed} \tag{8}$$

2.3.1 SimpleSpatial Tone MappingOperator

První operátor, který je vhodný pro paralelní zpracování, byl prezentován autory Pattanaik a kol. [14] a vychází z předpokladu, že v obraze se nacházejí různě osvětlené oblasti. Každá oblast je samostatně zpracovávaná vzhledem k pozadí obrazu. Oblast je tedy buď tmavší, nebo světlejší než pozadí. Výpočet pro jednotlivý pixel je vyjádřen vztahem:

$$YD(x, y) = Y(x, y) \,/\, [Y(x, y) + GC] \tag{9}$$

Úprava zajistí výsledné hodnoty v rozsahu <0;1>. Hodnota *GC* je globální kontrastní faktor, určený rovnicí (10), kde *c* je uživatelsky zvolený faktor a *YA* je průměrná hodnota jasu v celém obrazu.

$$GC = c\, YA \tag{10}$$

Pro většinu obrazů je dle autorů vhodné zvolit výchozí hodnotu parametru *c* jako 0,15.

2.3.2 Novel histogram adjustment

V mé bakalářské práci [12] byla popsána a testována metoda Haleq od autorů Duan a kol. [6] Metoda eliminovala většinu artefaktů, které mohou při transformaci dat vzniknout, a celkový vizuální výsledek byl velice dobrý. Daná transformace může být aplikována buď na celý obraz, nebo lze obraz rozdělit do stejně velkých bloků a transformaci aplikovat na ně.

Základní vzorec číslo (11) pro redukci dynamického rozsahu využívá logaritmické funkce, která zvýší kontrast a jas v nízkých hodnotách jasu snímku, zatímco vyšší jasové hodnoty jsou potlačeny.

$$\mathrm{D(I)} = (\mathrm{D_{max}} - \mathrm{D_{min}}) \frac{\log(I + \tau) - \log(I_{min} + \tau)}{\log(I_{max} + \tau) - \log(I_{min} + \tau)} + \mathrm{D_{min}} \qquad (11)$$

I_{min} a I_{max} jsou minimální a maximální jas scény, D_{max} a D_{min} jsou maximum a minimum jasu, které aktuálně používané zobrazovací zařízení dokáže zobrazit (většinou se jedná o hodnoty v intervalu [0 - 255]) a τ ovlivňuje celkovou světlost obrazu. Větší hodnota τ obraz ztmaví a menší hodnota τ obraz naopak zesvětlí. Pro správné a automatické určení této hodnoty je nutné vypočíst hodnotu I_{ave}, která se vypočítá podle rovnice (12),

$$\mathrm{I_{ave}} = \frac{1}{\mathrm{N}} \exp\left(\sum_{\mathrm{x,y}} \log(\delta + L(\mathrm{x,y})) \right) \qquad (12)$$

kde L je jas pixelu o souřadnicích x a y, N je celkový počet pixelů a δ je konstantní malá hodnota, zajišťující, aby do logaritmu vstupovala vždy hodnota větší než nula. K vypočítání hodnoty k na škále [0 - 1] byla zvolena rovnice

$$\mathrm{k} = \mathrm{A} * \mathrm{B}^{(2\log \mathrm{I_{ave}} - \log \mathrm{I_{min}} - \log \mathrm{I_{max}})/(\log \mathrm{I_{max}} - \log \mathrm{I_{min}})} \qquad (13)$$

kde hodnoty *A* a *B* jsou konstanty a nabývají hodnot 0,4 a 2, aby výsledná hodnota *k* byla v rozmezí od 0,2 až 0,8. Pokud průměrný jas obrazu bude spíše v dolní polovině rozsahu od minimálního do maximálního jasu, obraz by měl být tmavého charakteru a měla by mu být tedy přiřazena menší klíčová hodnota. V opačném případě je přiřazena hodnota větší.

Pro určení hodnoty τ takové, aby hodnota I_{ave} byla mapovaná na vypočítanou *k* hodnotu, se využívá rovnice (14).

$$\mathrm{k} = \frac{\log(I_{ave} + \tau) - \log(I_{min} + \tau)}{\log(I_{max} + \tau) - \log(I_{min} + \tau)} \tag{14}$$

Pro vypočtení výše uvedené rovnice se osvědčilo použít Newtonovu metodu výpočtu kořenů rovnice. Po použití správné hodnoty τ se sice celkový jas obrazu správně transformuje, je ale stále ochuzen o detaily ve tmavých oblastech. Důvodem je lineární mapování výsledných hodnot funkce *D(I)*. V našem případě je rozsah *D(I)* rozdělen na *N* = 256 stejných intervalů pomocí dělících bodů l_n a pixely, které spadají vždy mezi tyto dva dělící body, jsou následně mapovány na jednu celočíselnou hodnotu *D* v rozsahu od 0 až 255. Hodnoty jasů pak mají různou hustotu distribuce napříč všemi intervaly. Abychom co nejlépe využili omezený počet zobrazitelných jasů, upravíme hustotu distribuce v intervalech pomocí ekvalizace histogramu [2] . Přepočet intervalů pomocí ekvalizace ale není pro paralelní zpracování dat vhodný z důvodu velkých polí dat, ve kterých se prvky prohazují. Data ale lze zobrazit i při absenci ekvalizace za cenu mírně horších vizuálních výsledků.

3 Využití zpracování HDR dat v reálném čase

Teoretické využití má HDR technologie v mnoha odvětvích. Může se jednat jen o korekční funkci s cílem vytvořit vzhledově lepší fotografie nebo videa. Větší dynamický rozsah umožňuje dosáhnout mnohem lepších výsledků při post-processingu, protože nedochází k zaokrouhlování a ořezávání hodnot při výpočtech s neceločíselnými hodnotami. Lze nalézt i důležitější uplatnění, například v medicíně nebo průmyslu, kde snímána informace je reprezentována velkým rozsahem, např. snímky z rentgenu či dalších vyšetřovacích modalit, nebo kdy při snímání dochází k extrémním změnám světelných podmínek, např. podmínky při sváření. Pokud jsou data v HDR formátu a je k dispozici zpracování obrazu v reálném čase, je možné si prohlédnout celý dynamický rozsah scény a pozorovat detaily, které by v nízkém dynamickém rozsahu zůstali skryté.

3.1 Problém snímání

Největším problémem HDR dat je jejich získání. V dnešní době sice již existují videokamery schopné zachytit vyšší dynamický rozsah v reálném čase. Nevýhodou je ale vysoká pořizovací cena. Tato práce se spíše zaměřuje na prostředky, které jsou běžně dostupné, v dnešní době dokonce i ve většině domácností.

3.1.1 Paralelní snímání

Jednou z možností, jak HDR snímky získat, je paralelní snímání. Ke snímání dané scény je využito více záznamových zařízení, která jsou nasměrovaná stejným směrem. Každé záznamové zařízení má nastavené jiné hodnoty expozice tak, jak je naznačeno v kapitole 1 a výstup z každého zařízení je jinak nasvícené video, které je ale stále ve formátu s nízkým dynamickým rozsahem. Aby bylo možné hovořit o videosekvenci s vysokým dynamickým rozsahem, musí se z každého videa vyextrahovat každý jednotlivý snímek a provést automatické zarovnání, protože každá kamera zabírala scénu z jiného úhlu. Poté ze snímků se stejným časem složit snímky HDR a vložit do nové videosekvence. Vzhledem k odlišnostem u různých videokamer se doporučuje provést snímání se stejnými typy kamer.[21]

3.1.2 Časosběrné video

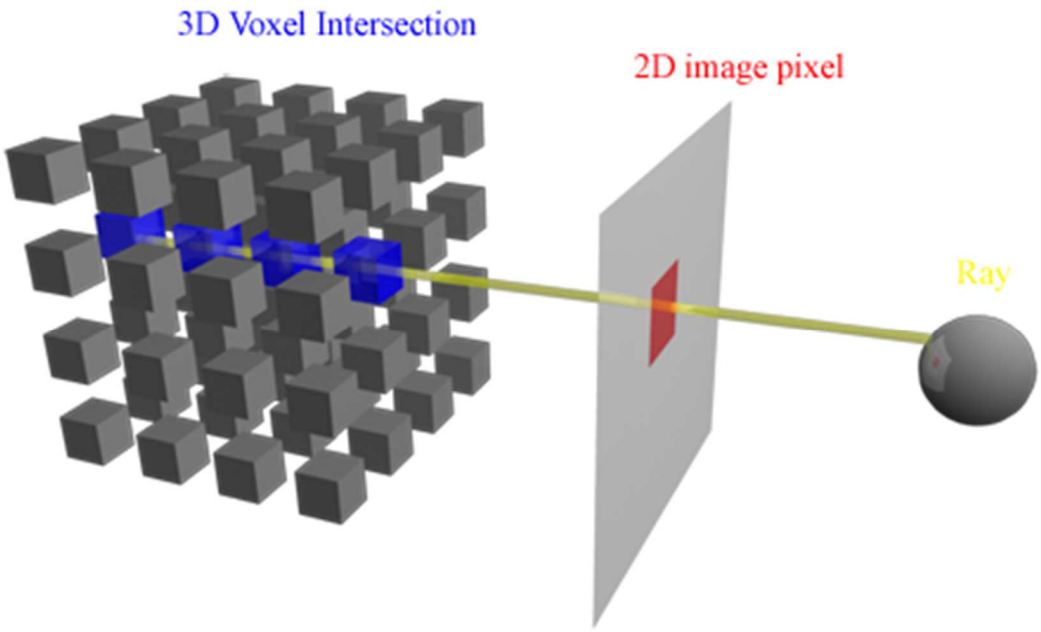
Alternativou k výše zmíněnému postupu je časosběrné video. O snímání se sice nedá říct, že probíhá v reálném čase, nicméně výsledné video může být užitečné, například pro pozorování dlouhotrvající jevů nebo v medicíně, kde se analýza může provést až zpětně z vytvořeného videa.

K vytvoření časosběrného HDR videa stačí pouze jedno záznamové zařízení, ale může jich být použito více. V případě, že se využívá pouze jeden fotoaparát, který je schopný zachytit pouze nízký dynamický rozsah, musí poskytnout pro jeden snímek HDR videa minimálně tři snímky LDR. Z nich se složí obrazy HDR a při jejich vykreslování za sebou v dostatečně rychlém intervalu, vzniká video HDR.

I když dnes fotoaparáty dokáží snímat poměrně velké množství snímků za vteřinu, potřebují nějaký čas pro přizpůsobení se změně expozice. Často je tak potřeba i několik vteřin k vyfotografování 3 snímků s kompenzací expozice -2, 0 a +2 EV. Z toho důvodu není tato metoda vhodná pro snímání pohyblivých scén nebo rychle se pohybujících objektů ve scéně.

3.2 Využití v medicíně

Teoretické využití by mohly mít HDR technologie, aplikované na medicínská data. Metoda, jak vizualizovat nasnímané 3D diskrétní medicínská data, se nazývá volume-rendering. Data jsou reprezentována voxely, které jsou většinou uspořádány do pravidelné 3D mřížky, kterou si lze představit jako 2D řezy objektem, pořízené například pomocí magnetické rezonance nebo počítačového tomografu. Každý bod v mřížce (voxel) je reprezentován pomocí jedné hodnoty v určitém rozsahu. Rozsah hodnot jednotlivých voxelů často bývá v rozsahu <0;255>, takže jednotlivé řezy lze snadno vykreslit jako šedo-tónový obraz. V praxi je však vyžadováno zobrazení celého 3D objektu najednou jako průmětu všech voxelových hodnot do výsledného obrazu. Jednou ze zobrazovacích metod volume ray casting, při které se z pozice kamery vysílají paprsky skrz voxely, jejichž hodnoty se kumulují. Jakmile paprsek projde skrz celý 3D objekt, je jeho hodnota upravena pro rozsah monitoru, většinou pomocí aritmetického průměru.



Obrázek 4 Ukázka Volumeraycasing u voxelových dat [25]

Jelikož se při kumulaci hodnot voxelů překročí výsledná hodnota dynamický rozsah zobrazitelný na monitoru, nabízí se zde možnost aplikovat techniky mapování tónů na výsledné hodnoty paprsků. V rámci této práce byla tato teorie implementována a v kapitole 5 si lze prohlédnout výsledky.

3.3 Další využití

Ve dvacátém století vznikl projekt EyeTap Digital EyeGlass [11] , který měl pomoci uživatelům s různými specifickými úkoly v běžném životě, anebo dokonce pomoci pracovníkům v různých odvětvích průmyslu, například pro pozorování materiálu při sváření apod. Zřízení EyeTap slouží jako kamera a displej zároveň a umožňuje uživateli lépe vidět v každodenním životě a zároveň ho lze využít jako přenosný počítač. V kontextu HDR dat by mohlo toto zařízení umožnit uživateli sledovat scény s takovým dynamickým rozsahem, s jakým si lidské oko není schopné poradit. Nebo jen přizpůsobit světelné podmínky tak, aby zařízení zvýšilo komfort uživatele.

4 Implementace řešení

Cílem implementace bylo vytvořit komplexní řešení pro zpracování HDR dat, zahrnující jak snímání, tak i výsledné zobrazení na LDR zařízení. Z výše uvedených metod byly vybrány vhodné postupy a byl kladen důraz na využití běžně dostupných hardwarových zařízení i softwarových technologií. Veškeré použité technologie jsou volně dostupné na internetu a jako programovací jazyk byl ve všech programech zvolen jazyk JAVA.

Následující kapitola popisuje implementaci postupu zpracování dat, jak z pohledu objektové struktury aplikace, tak i z pohledu vytvořeného uživatelské prostředí. Jsou zde podrobně vysvětleny navržené třídy, jejich vzájemná komunikace mezi sebou i s externími knihovnami včetně nastavení všech parametrů.

4.1 Snímání HDR dat na smart zařízení

První etapou při zpracování HDR dat je jejich získání. Protože běžné fotoaparáty, včetně těch, vestavěných v mobilních zařízeních nepodporují vyšší dynamický rozsah, než je 8bitová hloubka na jeden barevný kanál, byla v implementaci použita metoda získávání HDR obrazů, která je popsána v podkapitole 1.2.2. Při snímání je nutné zajistit nastavení řady parametrů. Běžné fotoaparáty většinou neumožňují vytváření vlastních programů, které by fotoaparát řídili, bylo pro snímání zvoleno mobilní smart zařízení, které podporuje operační systém android. Pokud by ale mělo být zvoleno právě jako záznamové zařízení HDR dat, je důležité vybírat ne jen podle vlastností vestavěného fotoaparátu, jako je například rozlišení či velikost clonového čísla. OS android umožňuje komunikaci s HW zařízeními pomocí vestavěných funkcí a tříd, ale ne všechny jsou u různých typů mobilních zařízení podporovány. Z těchto důvodů byl pro implementaci vybrán mobilní telefon od značky Samsung vyšší řady, konkrétně Galaxy Note 4, který obsahuje zabudovaný 16 megapixelový snímač a podporuje většinou funkcí Android rozhraní Camera API, včetně funkce

kompenzace expozice, která je pro snímání pomocí techniky skládání obrazu klíčová.

4.1.1 Aplikace HDR_Timelapse

Program, implementovaný pro snímání HDR dat se nazývá HDR Timelapse a jeho zdrojové kódy jsou přiloženy v příloze. Je napsaný v jazyce JAVA a deploy se provádí pomocí souboru APK přímo do zařízení. Aplikace byla vyvíjena proti Android rozhraní ve verzi 21 (Android 5.0 Lollipop), ale je zpětně kompatibilní i se staršími zařízeními podporujícími pouze verzi 17 a výš.

Android Camera api

Základním rozhraním pro využití kamery na Android zařízení je Camera API. V android verzi 21 je již toto rozhraní označeno jako DEPRECATED a mělo by být nahrazeno rozhraním Camera2 API. To ovšem v době implementace nebylo telefony plně podporováno, a proto bylo k implementaci využito původní rozhraní Camera API.

Proces práce s tímto rozhraním při snímání obrazů probíhá následovně. Nejdříve si aplikace vyžádá přístup ke kameře a dostane instanci třídy Camera. Nyní je možné s kamerou manipulovat. Základní nastavení probíhá pomocí pomocné třídy Camera.Parameters. Jedná se o databázi parametrů typu klíč hodnota, ke kterým lze přistupovat pomocí jejich getterů a setterů. Po úpravě parametrů se musí objekt předat zpět kameře, aby se nastavení projevilo. Pro snímání HDR dat jsou důležité následující:

- setPictureSize
- setExposureCompensation
- getExposureCompensationStep
- setAutoExposureLock
- setFlashMode

Pomocí nastavení PICTURE_SIZE lze měnit rozlišení výstupního obrazu. Nelze ovšem zvolit jakoukoliv hodnotu. Každé zařízení může podporovat různé rozlišení,

z toho důvodu je vhodné nejprve vypsat dostupná rozlišení pomocí volání *getSupportedPictureSizes()* a vybrat vhodné rozlišení ručně.

Volání *setExposureCompensation()* je nejdůležitější z hlediska vytváření HDR. Slouží k upravení expoziční hodnoty (EV) na jinou, než jakou vybrala kamera automaticky. Před jeho voláním je opět důležité zjistit možnosti daného hardwarového zařízení. První důležitou hodnotou je EXPOSURE_COMPENSATION_STEP, která udává velikost kroku expozice. Pokud by se volání kompenzace expozice zavolalo s parametrem -6 a velikost kroku bude 0,33333, bude výsledná kompenzace -2EV. Dalšími důležitými hodnotami jsou MIN_EXPOSURE_COMPENSATION a MAX_EXPOSURE_COMPENSATION, které udávají celočíselný rozsah hodnot, které lze vložit jako parametr pro volání kompenzace expozice.

Nastavení AUTO_EXPOSURE_LOCK slouží k uzamčení automatického výběru expozice. Je nutné tuto funkci vypnout vždy před začátkem snímání HDR obrazu, jinak nemusí dojít ke korektnímu rozložení LDR hodnot v jednotlivém snímku. To způsobí přesvícení respektive podsvícení obrazu a ztrátu dynamického rozsahu.

Poslední funkcí je FLASH_MODE, kde se doporučuje vložit hodnotu FLASH_MODE_OFF, aby nedocházelo ke spuštění blesku při nízkém osvětlení. Automatický blesk totiž bývá často nastaven jako výchozí hodnota.

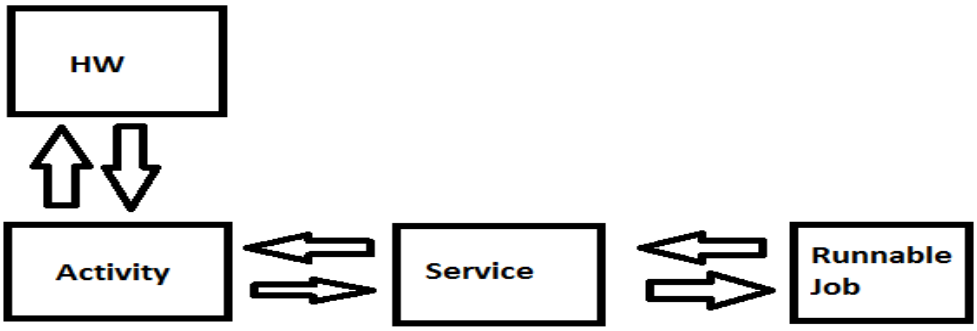
Ostatní nastavení, jako hodnota clony nebo čas expozice nelze bohužel ke snímání HDR obrazů využít. Software kamery totiž vždy při snímání dopočítá ostatní hodnoty tak, aby došlo ke správnému osvětlení obrazu. Jinou možnost než nastavení kompenzace expozice tedy Camera API nenabízí.

Novější rozhraní Camera2 API již takto limitováno není. Komunikace s hardwarem kamery je mnohem více otevřena a ovládat lze již prakticky cokoliv, co lze nastavit například na zrcadlovkách. Je dokonce umožněn přístup k RAW datům kamery, které je možné přímo streamovat na display nebo uložit ve formě snímku. Takže ve výstupních obrazech nedojde ke zkreslení vlivem komprese jpeg. Další vylepšenou funkcí je i například HighSpeedCaptureSession, která by měla umožňovat snímání obrazů ve vysoké rychlosti. Ze strany operačního systému

Android je rozhraní plně funkční, ze strany výrobců mobilních telefonů tomu tak bohužel není. Na trhu existuje zatím pouze relativně málo modelů, které novější rozhraní alespoň zčásti podporují.

Architektura

Aby aplikace mohla využít funkce fotoaparátu, potřebuje potřebné oprávnění, které se musí umístit do manifest souboru. Konkrétně se jedná o oprávnění získat instanci objektu kamery a využití automatického ostření. Dalším z oprávnění, které aplikace vyžaduje je zápis na externí úložiště neboli paměťovou kartu. S těmito oprávněními musí uživatel zařízení před instalací souhlasit, jinak nebude možné aplikaci do zařízení nainstalovat.



Obrázek 5 Struktura komunikace komponent aplikace HDR Timelapse

Samotná aplikace se skládá z jedné aktivity, jedné služby a pracovního vlákna, které se stará o samotné snímání obrazů. Po startu aplikace se nastartuje aktivita CamActivity, která si ihned od operačního systému vyžádá přístup ke kameře. Zařízení může disponovat více kamerami, jako výchozí se ale bere kamera s indexem 0, která je ve většině zařízení umístěna vzadu. Jelikož kameru může v tuto chvíli využívat jiná aplikace a přístup k ní může mít naráz pouze jedna, nemusí se tento krok zdařit. V takovém případě vypíše aplikace chybu, že kamera nebyla nalezena. Po úspěšném získání objektu kamery aplikace zaregistruje SurfaceView, na kterém se ihned zobrazuje aktuální náhled kamery. Zde má uživatel možnost volby zaostření pomocí ovládacích prvků a také možnost

kontroly, zda je scéna správně nasvícena a zda na snímcích bude vše zobrazeno. Aplikace po stisknutí tlačítka Test vyfotí testovací snímek, který je možné si prohlédnout v galerii a případně nastavení změnit.

Po startu snímání předá aplikace řízení a objekt kamery službě CameraService. Protože je aplikace navržena tak, aby snímala jednotlivé snímky po dlouhou dobu, není při snímání přítomný náhled na displeji zařízení, takže se před samotným předáním musí zastavit náhled pomocí *Camera.stopPreview()* a odregistrovat SurfaceView pomocí volání *view.setCamera(null)*. Nyní se nastartuje služba CameraService a aktivita se ukončí. Aplikace dále běží i bez grafického rozhraní a po určitém čase se telefon přepne do úsporného režimu, aby šetřil baterii. Služba na pozadí ale běží dále. Pokud je potřeba proces snímání zastavit, je možné kdykoliv znovu spustit aplikaci, která znovu nastartuje aktivitu CamActivity. Ta zjistí, zda je služba spuštěna a pokud ano, vyžádá si od ní objekt kamery, aby převzala řízení, a znovu se na displeji zařízení zobrazí náhled a stav služby, zda běží nebo neběží. Uživatel zde má možnost předat řízení zpět službě nebo službu zastavit.

Služba zde existuje pouze jako kontejner pro vlákno snímacího procesu, které se nastartuje ihned po startu služby. Vlákno je implementováno ve třídě PhotoJob a výhradní řízení nad ním má pouze služba. Pokud chce aktivita měnit běh vlákna, musí využít dostupné veřejné metody služby.

Proces snímání je implementovaný podle metody autorů Debevec a Malik [5] . Implementace byla cílena na nasnímání co nejvíce snímku v co nejkratším čase. Po startu procesu se kameře, kterou mu poskytla služba, nastaví základní parametry jako rozlišení výstupních snímků apod. Jakmile je kamera připravena, vyfotí se první dávka snímků, která se skládá ze tří obrazů, každý s jinou hodnotou expozice. Snímání probíhá tak, že se na objektu kamery zavolá *startPreview()*, aby se nastavila správná hodnota expozice. Tato operace vyžaduje určitý čas, takže není možné ihned po tomto volání snímání provést. Podle testování bylo potřeba ve většině případů alespoň 1 000 milisekund, než se kamera správně přizpůsobila světelným podmínkám. Bohužel rozhraní nepodporuje u této funkce callback, a tak je nutné čekání zajistit ručně. Alternativou je vynutit kameru aby provedla

autofocus a pokračovat až v metodě AutoFocusCallback. Při zaostření se totiž také určuje správná hodnota expozice a v daném callbacku je již na kameře správná expoziční hodnota nastavena. Nicméně ne vždy se autofocus povede, a protože snímání probíhá v dávkách automaticky bez kontroly uživatele a bez grafického rozhraní, mohli by některé výsledné obrazy být rozmazané a výsledná sekvence obrazů by byla nepoužitelná.

Volání *startPreview()* není možné bez objektu Surface, protože kamera by neměla kam náhled zobrazit. Služba ale pracuje bez grafického rozhraní a není nutné náhled uživateli zobrazovat. Řešením bylo vytvoření prázdného objektu SurfaceTexture, který se kameře předal a kam kamera náhled vykresluje. Uživateli se ale nikde nezobrazuje.

Jakmile je expoziční hodnota správně nastavena, zavolá vlákno na kameře *takePicture()*, která ve formátu jpeg uloží aktuální obraz na externí paměťovou kartu. Ukládání probíhá v jiném vlákně, takže doba ukládání neovlivní další snímání. Dalším krokem je změna expoziční hodnoty. Na testovaném zařízení byla velikost kroku expozice rovna hodnotě 0,5 a minimální a maximální hodnota expozice byla -4 a +4, takže zařízení umožňovalo pohybovat se v rozsahu -2EV až +2EV s krokem 0,5. Tento rozsah není velký, ale HDR obraz lze ze sady snímků vytvořit. Nasnímají se tedy další dva obrazy. Jeden s kompenzací expozice -4 a druhý s +4. Po každé změně nastavení se volá *startPreview()* a čeká 1 000 milisekund, aby se nastavení projevilo.

Snímání dávek je prováděno v cyklu a je ovlivněno dvěma parametry. První parametr určuje interval mezi dávkami a druhý parametr určuje, kolik dávek se vyfotí. Interval mezi dávkami se nedoporučuje dávat nulový. V praxi se totiž systém zahltil a do výsledných obrazů uložil jinou hodnotu rychlosti závěrky (shutter speed), než s jakou byly obrazy skutečně pořízeny. Rychlost závěrky je k rekonstrukci HDR obrazu klíčová a při chybných hodnotách nejde ke korektnímu sestavení HDR obrazu.

Jakmile proběhne snímání poslední dávky, zavolá se na objektu kamery *release()* a tím se kamera opět zpřístupní ostatním aplikacím. Zároveň se i ukončí služba a odstraní se veškeré reference na objekt kamery.

Data získaná při snímání jsou uchovávány jako série snímků ve formátu jpeg. Snímky jsou pojmenované podle data začátku snímání a pro snazší automatické zpracování jsou označeny číselnou sekvencí tak, jak byly nasnímány. Důvodem pro uchování snímků v tomto formátu je, že můžeme snímky libovolně upravovat, než je převedeme do formátu HDR. Například změna rozlišení na LDR snímku je mnohem snazší než na snímku HDR.

Obrázek 6 Snímací okno aplikace HDR Timelapse se všemi ovládacími prvky

Aplikace HDR_Timelapse obsahuje pouze jednu obrazovku, která obsahuje pozadí, na které se vykresluje náhled kamery a čtyři tlačítka, pomocí kterých lze vyfotit testovací obraz, zaostřit, zapnout/vypnout službu pro focení a zjistit informace o počtu zbývajících snímacích dávek ve formě notifikace.

4.2 Konverze snímků a reprezentace HDR obrazu

4.2.1 Skládání obrazů

Vytváření HDR obrazů obstarává třída VideoCreator, obsažená v balíčku tříd v aplikaci HDRonGPU. Ke správnému vytváření obrazů potřebuje třída znát cestu ke složce, kterou následně celou zpracuje. Dalšími volitelnými parametry jsou výška a šířka obrazu, na kterou se snímky zmenší, respektive zvětší, před sestavením HDR obrazu. Po startu zpracování složky třída vždy vytáhne tři související soubory podle jejich indexu a ty předá třídě HDRCreator. Ta v prvním kroku načte hodnoty časů expozic z EXIF dat a následně načte i samotná RGB data.

Samotné vytváření obrazu probíhá podle postupu skládání obrazů, uvedeného v podkapitole 1.2.2.2. Vstupní RGB data jsou vložena do třech jednorozměrných polí, pro každý obraz jedno pole. Data se v cyklu zpracují a v každém kroku se vytáhne z polí pod stejným indexem hodnota, která je přepočítána a vložena do nového pole. Při přepočtu hodnoty podle váhové funkce (3) je důležité zvolit menší krajní hodnotu Z_{max} než je hodnota krajní (255), protože v případě přesvícených pixelů ve všech třech zdrojových obrazech dojde v algoritmu k dělení nulou nebo vzniknou v obraze artefakty v podobě černých pixelů z důvodu příliš malé váhy hodnot. Tyto hodnoty jsou v procesu nahrazeny dočasnou hodnotou a na konci zpracování jsou dočasné hodnoty nahrazeny hodnotami minimálními nebo maximálními, podle toho, jestli se původně jednalo o pixely tmavé (0) nebo světlé (255). Hodnoty minima a maxima jsou vypočteny z hodnot nových, nikoliv původních.

4.2.2 Formát HVF

Nová zpracovaná data jsou předána třídě HVFWriter, která je uloží na disk do souboru ve formátu HVF (Hdr Video Format), který byl vytvořen v rámci této práce. Důvodem volby nového vlastního datového formátu byla příliš velká časová náročnost pro načítání obrazových dat. HDR formáty, které mají za cíl snížit požadavky na datovou náročnost záznamového zařízení, jsou většinou náročné na výpočetní čas při rekonstrukci dat. Což je pro zpracování dat, které je nutné k jejich vykreslení v reálném čase nežádoucí.

HVF formát slouží jako jednoduchý kontejner objemových dat. Skládá se z hlavičky a datové části. Aby se obě části odlišily, jsou první čtyři bajty rezervovány pro celočíselnou hodnotu, která udává velikost hlavičky v bajtech. Tím je zaručena flexibilita formátu do budoucna pro případné přidávání informací do hlavičky. V hlavičce jsou uvedeny dvě hodnoty, oddělené středníkem. První hodnotou je šířka obrazů a druhou hodnotou je výška obrazů. Bez těchto hodnot není možné obrazy ze souboru HVF zrekonstruovat. V datové části už jsou uložena pouze obrazová data snímků za sebou, tak jak byly obrazy zpracovány. Každý obraz je složen z RGB hodnot, kde na jeden pixel připadají 3 čísla v plovoucí řádové

čárce formátu FLOAT. Hodnoty jsou v souboru uloženy v pořadí vždy červená, zelená a modrá a při procházení obrazu po řádcích od pixelu v levém horním rohu obrazu. Pořadí bitů je kvůli reprezentaci OpenGL nastaveno na LITTLE_ENDIAN.

4.3 Zobrazení HDR s využitím zpracování na GPU

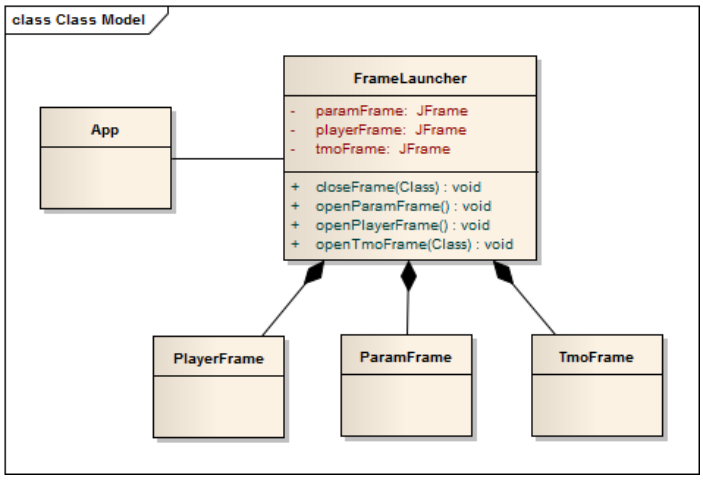
Další z implementovaných částí jsou metody pro zobrazení získaných HDR dat. První z metod využívá pro náročnější výpočty grafickou kartu a paralelní zpracování dat za pomocí shaderových programů, zatímco druhá metoda nechává veškerou výpočetní činnost na procesoru.

4.3.1 Aplikace HDRonGPU

Struktura aplikace je rozdělena do několika Java tříd, kde každá třída má svoji jasně definovanou úlohu. Aplikace dále obsahuje sadu textových souborů, které jsou za běhu načteny a předány grafické kartě jako zdrojový kód pro shaderové programy.

FrameLauncher

Nastavení parametrů aplikace se provádí prostřednictví řady ovládacích prvků, které jsou rozdělné do několika nezávislých oken. Nezávislost oken a korektní vláknové spouštění zajišťuje třída FrameLauncher.



Obrázek 7 Diagram tříd modulu pro nezávislé zobrazování oken aplikace

FrameLauncher zajistí spuštění oken PlayerFrame a ParamsFrame ve správném vlákně (Obrázek 7). Nabízí i možnost spustit další okno pro operátor mapování

tónů, které se pro každý operátor může lišit. Z toho důvodu metoda přijímá povinný parametr frameClass a pomocí reflexe vytvoří instanci předané třídy a okno spustí. V případě, že je již nějaké okno pro jiný operátor spuštěno, třída zajistí správně ukončení okna před vytvořením nového. FrameLauncher si drží reference na všechna spuštěná okna a proto se pro jejich zavření nebo znovuotevření vždy využívají pouze metody vytvořené přímo v této třídě.

ParamsFrame

ParamsFrame (Obrázek 14) je základní okno aplikace, které se zobrazí po startu a zajišťuje základní operace nad výstupním obrazem, jako například:

- Gama korekce
- Globální jas
- Síla operátoru
- Šířka okna

Gama korekce probíhá před finální transformací na LDR rozsah a je implementována podle vzorce (8). Globální jas je desetinné číslo v rozsahu od -1 do1 a je přičteno k transformované hodnotě jasu před gama korekcí. Výchozí hodnota je 0, takže po startu není obraz nijak jasově upraven. Síla operátoru je implementována jako lineární interpolace mezi původní, lineárně transformovanou jasovou hodnotou a novou hodnotou, transformovanou pomocí zvoleného operátoru. Aplikace umožňuje měnit dělící bod interpolace a tím ovlivnit, jak hodně bude výsledný obraz operátorem ovlivněn. Šířka okna slouží pro určení dělící linie v obraze, kde se ještě zobrazuje původní verze obrazu, a kde transformovaná verze obrazu. Účel této funkce je pro porovnání obrazu před a po transformaci.

Okna operátorů mapování tónů

Ovládací prvky v okně jsou závislé na zvoleném operátoru mapování tónů. Při volbě operátoru SimpleSpatial TMO se zobrazí následující možnosti volby (Obrázek 15):

- Klíčová hodnota
- Verze operátoru
- Šířka lokálního okolí

Klíčová hodnota ovlivňuje celkovou světlost obrazu a je zadána uživatelem. To se v praxi ukázalo jako největší nevýhoda operátoru, protože jak se snímaná scéna dynamicky mění, je někdy zapotřebí upravit i klíčovou hodnotu. Z uživatelského hlediska je ale nemožné změnu provést pro každý snímek při zobrazování snímků ve frekvenci 30fps.

Verze operátoru poskytuje tři volby. První je globální verze operátoru, která provede transformaci podle vzorce (9). Druhá verze je již lokální úpravou, která k výpočtu využívá aritmetický průměr okolních hodnot. Poslední verze upravuje předchozí výpočet, a místo aritmetického průměru vypočítá průměrnou hodnotu okolních pixelů pomocí mediálního filtru. Výpočet střední hodnoty ale vyžaduje seřazení prvků v poli a taková operace se musí provést pro každý pixel. GLSL navíc nemá implementovaný žádný řadící algoritmus podporující paralelismus, takže volba šířky filtru nesměla být příliš velká, jinak docházelo k příliš dlouhému času při zpracování jednoho snímku. Parametr šířka lokálního okolí určuje počet pixelů šířka x výška při výpočtu aritmetického průměru. U mediálního filtru se tato hodnota nezohledňuje, protože GLSL nepodporuje pole s dynamicky alokovanou velikostí.

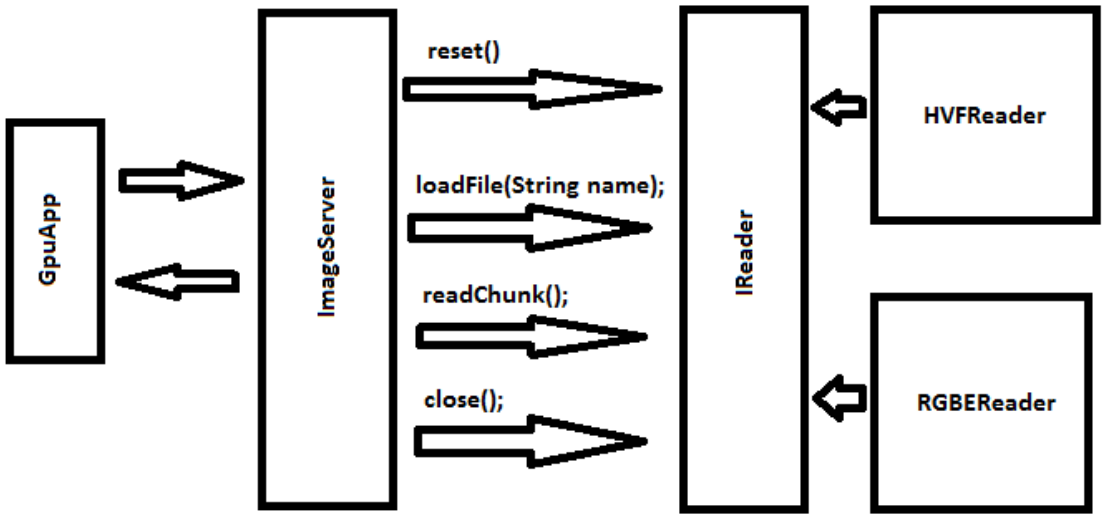
PlayerFrame

Posledním z oken aplikace, které slouží k zobrazení ovládacích prvků je PlayerFrame (Obrázek 14). Okno obsluhuje běh snímků, konkrétně lze nastavit frekvenci změny snímků nebo běh pozastavit a opětovně spustit.

Zobrazovací okno

Okno, které se stará o zobrazení samotných snímků, je implementováno ve třídě GpuWindow. Třída jako jediná nevyužívá Swing, ale ke svému zobrazení využije přímo knihovnu OpenGL. Data k zobrazení ji poskytuje třída ImageServer, jejíž konstruktor přijímá parametr v podobě řetězce, určujícího cestu k souboru.

Dokáže rozeznat a načíst dva typy souborů. Prvním typem je soubor s objemovými daty s koncovkou hvf a druhý implementovaný datový formát je s koncovkou hdr a jedná se o formát popsaný v kapitole 1.



Obrázek 8 Struktura komunikace tříd při načítání dat z datových souborů

Obrázek 8 popisuje komunikaci tříd při načítání jednotlivých obrazů. Vytvořené řešení je cílené na co nejrychlejší přenos dat z pevného disku do paměti grafické karty.

Přenos dat

Jedním z možných řešení, jak přenést obrazová data na grafickou kartu, bylo načíst zdrojový HVF soubor přímo do operační paměti a následně z něj odesílat jednotlivé snímky pomocí OpenGL funkce glTexImage2D na GPU. Proces načtení do operační paměti by se mohl provést před začátkem vykreslování všech snímků a při samotném vykreslování jednotlivých snímků už by proces nezpomaloval diskové operace. Toto řešení se ale ukázalo nepoužitelné pro takto objemná data. I video o délce několika jednotek vteřin v HD rozlišení dosahuje velikosti v řádech tisíců megabajtů a do operační paměti počítače se často nevejde. Byl proto zvolen jiný přístup, a do paměti se načte vždy jen konkrétní obraz, který se má vykreslit. Jazyk JAVA navíc umožňuje mapování souborů, uložených na pevném disku, přímo

do paměti (Obrázek 9). Systém takto zpřístupní objekt bufferu pro přístup k datům stejně, jako kdyby byly data v operační paměti. Pokud ale aplikace zažádá o přístup k těmto datům, budou data přečteny z pevného disku, nikoli z operační paměti.

```
fis = new FileInputStream(path);
channel = fis.getChannel();

MappedByteBuffer headerSizeBuffer = channel.map(FileChannel.MapMode.READ_ONLY, 0, 4);
```

Obrázek 9 Ukázka zdrojového kódu pro mapování dat do operační paměti

Pro načtení jednoho snímku ze souboru se volá na třídě HVFReader metoda *readChunk()*, která vrací objekt FloatBuffer. Ten se přímo předá knihovně OpenGL a ta už data přenese do textury. Metoda *readChunk()* nepřijímá žádné parametry. Správné rozměry obrazu načte z hlavičky souboru a vždy načte další snímek tak, jak jsou snímky v souboru uloženy. Pokud je potřeba změnit pozici načítání, je nutné zavolat metodu *reset()*, která vynuluje index a soubor se bude načítat znovu od prvního obrazu. Pokud se už v souboru nenachází žádné další soubory, rozhodne ImageServer o další akci. Pokud běží v režimu REPEAT, bude čtení pokračovat znovu od prvního snímku. Pokud ne, program zavře všechna otevřená okna a ukončí se.

První obraz se na GPU přenese přes funkci glTexImage2D, kde OpenGL alokuje pro předaná data místo v paměti ve formě textury a samotná data do místa vloží. Proces alokace je časově náročný a proto OpenGL umožňuje aktualizovat danou texturu pomocí funkce glTexSubImage2D, kde se původní data nahradí daty novými a nedochází k alokaci nové části paměti a dealokaci staré tak, jak by tomu bylo v případě použití glTexImage2D v cyklu za sebou.

4.3.2 Shaderové programy

Samotnou vizualizaci dat obstarávají shaderové programy, které jsou umístěné v balíčku cz/mk/slideshow/glsl/shaders. Každý program je reprezentovaný jedním txt souborem. Kompilaci i nalinkování programů zajišťuje třída ShaderProgram. Ta zajištuje dodání jednoho vertex shaderu a jedno fragment shaderu do vizualizačního řetězce. Třída poskytuje i další metody pro práci s programem jako např. dodání hodnot do uniform proměnných apod.

Dekompoziční shader

Shader provádí dekompozici obrazu popsanou ve vzorci (6). Protože se program spouští pro každý pixel vstupního snímku, musí nejprve získat správný korespondující pixel ze vstupní textury. Z vertex shaderu byla do programu předána interpolovaná pozice. A protože aplikace vykresluje jednoduchý čtverec o souřadnicích [-1;-1] [-1; 1] [1;-1] [1; 1], můžeme interpolované souřadnice čtverce přepočíst na souřadnice do textury.

```
14      vec2 pos = (pozice * 0.5) + 0.5; //uprava pro rozsah 0-1
15      vec4 texColor = texture2D(texture1, vec2(pos.x, -pos.y));
```

Obrázek 10 Transformace souřadnic čtverce na souřadnice textury

Získání jasové hodnoty z hodnot RGB zajistí metoda getLuminance, ukázaná na následujícím obrázku:

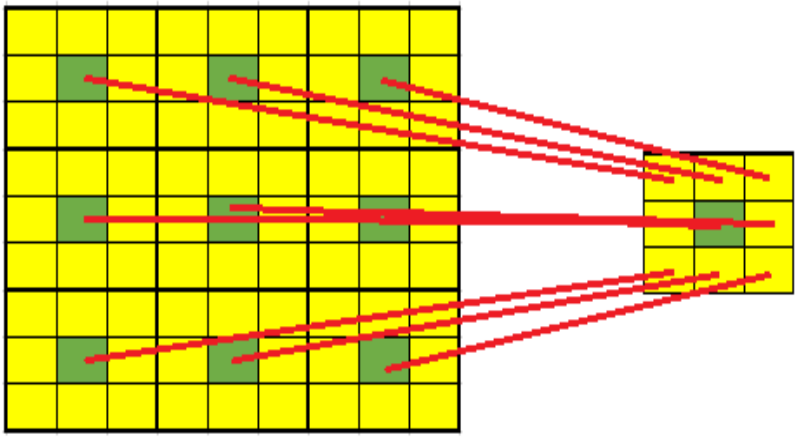
```
 8 float getLuminance(float red, float green, float blue) {
 9     return red * 0.2989 + green * 0.5866 + blue * 0.1145;
10 }
```

Obrázek 11 Implementace dekomponování jasové složky od barevných kanálů

Výsledná hodnota je uložena do volného atributu W čtyř-složkového vektoru texColor, kde jsou již první tři atributy vyplněny hodnotami RGB a který je následně označen jako výstupní hodnota daného pixelu.

Pyramid shader

Pyramid shader má za úkol snížit rozlišení vstupního obrazu pro získání globálních informací, které nelze efektivně získat paralelním zpracováním. Protože program proběhne pro každý pixel ve výstupním obraze, je výstupní obraz tvořen třikrát menší texturou, než byla textura vstupní. Lineární interpolací souřadnic podle postupu, uvedeného na obrázku (10), se získá pixel přesně uprostřed lokálního okolí o rozměrech 3 x 3 pixely.



Obrázek 12 Ukázka pyramidového hledání informací v obraze

Z lokálního okolí se poté vypočtou požadované informace, které se zprůměrují nebo jiným požadovaným způsobem sjednotí a vloží do výstupní zmenšené textury.

Preprocessshader

Program má za úkol zjistit globální informace ze zmenšené textury, kterou mu připravil Pyramid shader. Algoritmus pomocí vnořených cyklů vypočítá z textury celkový průměrný jas obrazu a celkový logaritmický jas obrazu. Z vypočtených hodnot je dále řešena rovnice (13) pro proměnou K a následně se hledají numerickou metodou kořeny rovnice (14) podle Newtonovy iterační metody tečen.

SimpleSpatial TMOshader

První ze dvou implementovaných metod mapování tónů je SimpleSpatial TMO. Jedná se o finální shader, jehož výsledek je mapován přímo na výstup do okna obrazovky (GpuWindow). Okno má rozlišení stejné, jako vstupní obrazy, aby nemuselo docházet k přepočtu. Kód shaderu se tedy provede pro každý bod obrazu. Vstupem do shaderu jsou tři textury a 7 dalších proměnných pro ovládání procesu mapování. První vstupní texturou je původní originální obraz, jehož šířku vykreslení lze ovlivňovat vstupním parametrem. Druhou texturou je dekomponovaný vstupní obraz a třetí textura obsahuje globální informace, zajištěné v Preprocess shaderu.

Z dekomponované textury program načte hodnotu jasu, kterou zpracuje podle vzorce (9) nebo výpočet případně upraví podle lokálního okolí pomocí průměru nebo mediánu. Protože GLSL nepodporuje pole s dynamicky alokovanou velikostí, je velikost pole při výpočtu mediánu nastavena na pevnou hodnotu 25 prvků, která odpovídá čtvercovému okolí o hraně 5 pixelů. Kvůli absenci rekurze v GLSL, byl k seřazení pole pro zjištění střední hodnoty implementován pouze jednoduchý bubble sort.

Po transformaci provede shader přinásobení jasu zpět k barevným kanálům a případné další korekce obrazu podle vstupních parametrů.

Histogram adjustment shader

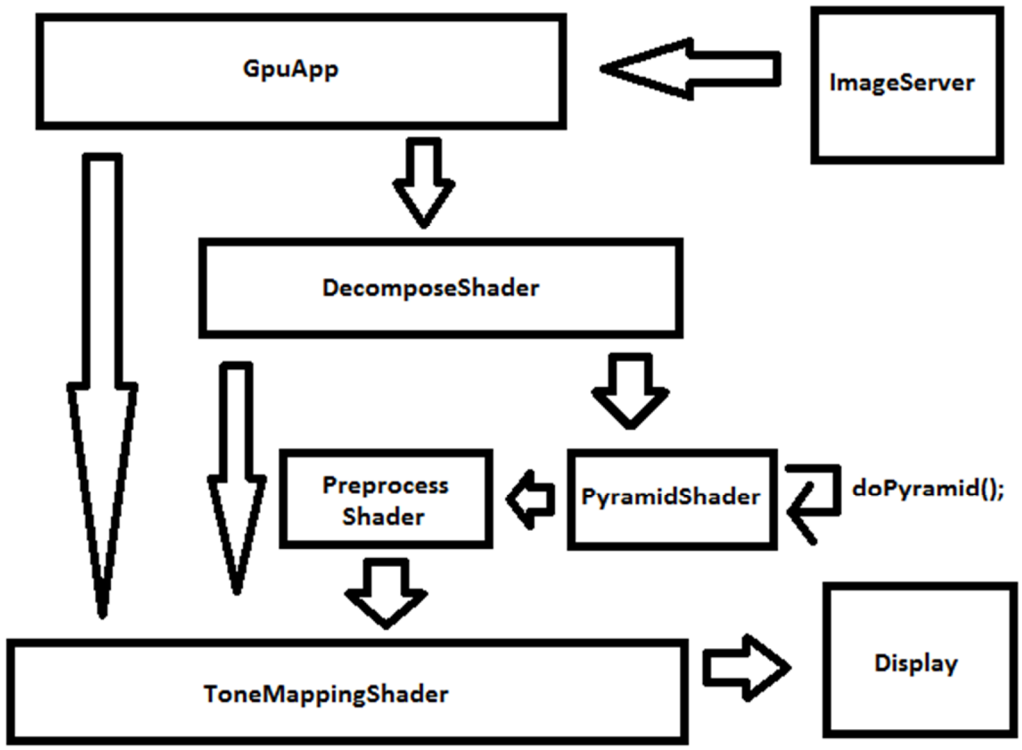
Druhou implementovanou metodou pro mapování tónů je Novel Histogram Adjustment. Metoda nebyla na GPU implementována celá, protože ne všechny její části dobře podporují paralelní zpracování. Vstupní textury jsou stejné jako u předešlého shaderu. Z dekomponované textury se opět načte jasová složka, která je tentokrát transformována podle vzorce (11). Protože knihovna OpenGL transformovala hodnoty textury na rozsah 0-1 a stejný rozsah požaduje i pro výstupní hodnoty, jsou parametry vzorce D_{min} i I_{min} nastaveny na 0 a hodnoty D_{max} a I_{max} nastaveny na 1. Poslední hodnotou, potřebnou pro výpočet je parametr τ, který ale již byl vypočten v Preprocess shaderu a je dostupný ve vstupní textuře. Proces vyrovnání histogramu včetně jeho lokální verze nebyl na GPU implementován z důvodu nutnosti využití polí a jejich řazení. Tyto techniky nejsou vhodné pro paralelní zpracování a při přenechání výpočtu pouze na jedné shaderové jednotce by nebyl výpočet efektivní.

4.3.3 Linearmapping shader

Lineární mapování HDR rozsahu zajistí samotná knihovna OpenGL, která automaticky transformuje vstupní data v podobě textury na rozsah čísel s plovoucí řádovou čárkou 0-1. Program tedy mapuje vstupní data přímo na výstup shaderu.

4.3.4 Proces vykreslení

Pro úspěšné vykreslení snímků je zapotřebí kombinace všech implementovaných shaderových programů. V rámci jednoho průchodu je možné využít pouze jeden shaderový program a proto před zobrazením obrazu na monitor je nutné takovýchto průchodů více za pomoci vykreslování do textury. V aplikaci se o tuto činnost stará třída FrameBufferObject, která v konstruktoru přijímá jeho šířku a výšku v pixelech. Třída připraví novou texturu, do které bude možné zapisovat a příkazem glBindFramebufferEXT, kam se předá ID framebufferu, knihovna OpenGL nastaví cíl renderování na požadovanou texturu. Samotný proces vykreslení si lze prohlédnout na obrázku (13).

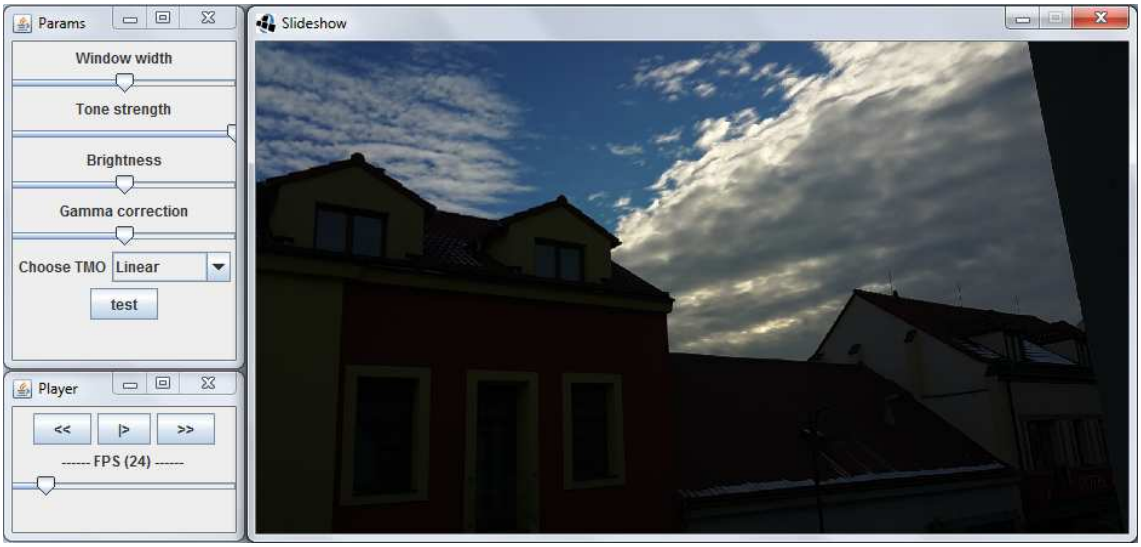


Obrázek 13 GPU Pipeline upravená pro HDR data

Běžící aplikace si vyžádá od třídy ImageServer aktuální snímek, který se má vykreslit. Ve formě textury je pošle do dekompozičního shaderu, který dekomponuje z RGB hodnot jasovou složku a nové hodnoty předá opět ve formě textury do pyramidového shaderu. Ten texturu zredukuje na třetinu a hodnoty zprůměruje. Proces redukce může být zavolán vícekrát za sebou, aby došlo

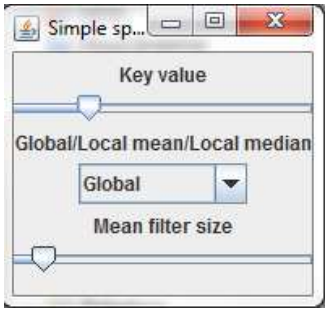
k vícenásobné redukci textury. Ve chvíli, kdy je textura dostatečně malá pro sériové zpracování na GPU, jsou data předána do Preprocess shaderu, který vypočte globální informace o obraze. V posledním kroku jsou všechny tři textury předány do ToneMapping shaderu. O jeho konkrétní implementaci rozhoduje volba uživatele a výstup z tohoto shaderu už je mapován přímo na okno obrazovky. Po úspěšném vykreslení se vyžádá další obraz od objektu ImageServer a celý proces probíhá znovu od začátku.

4.3.5 Dialogy aplikace



Obrázek 14 Dialogy aplikace HDRonGPU a HDR_3D

Aplikace obsahuje základní okno (Slideshow), kam se vykreslují snímky. Okno má stejnou velikost, jako zobrazované snímky. Rychlost změny snímků lze určovat pomocí ovládacích prvků v okně Player. Přehrávání lze i úplně zastavit prostřednictvím tlačítka play/stop. Hlavním oknem pro nastavení parametrů zpracování dat je okno Params. Je zde možné použít základní grafické úpravy obrazů a zvolit operátor mapování tónů. Při výběru jakéhokoliv jiného operátoru než lineárního, se zobrazí nové okno s parametry specifickými pro vybraný operátor.



Obrázek 15 Ovládací prvky operátoru Simplespatial TMO. Lze měnit klíčovou hodnotu, typ hledání okolí a šířku okolí pro aritmetický průměr.

4.4 Zobrazení HDR dat s využitím zpracování na CPU

4.4.1 Aplikace HDRonCPU

Pro srovnání rychlostí grafických výpočtů byla vytvořena ještě druhá aplikace s názvem HDRonCPU, která pro stejné zpracování HDR dat nevyužívá grafickou kartu, ale všechny výpočty zajišťuje procesor počítače. Aplikace nabízí kompletní prostředky pro zpracování HDR obrazů, od načtení až po jejich vykreslení (Obrázek 17).

Podporované formáty

Aplikace podporuje čtyři základní formáty JPEG, PNG, RGBE a OpenEXR. O načtení obrazových hodnot z datového souboru se stará obecná servisní třída FileLoaderFactory, která na základě předané koncovky souboru vrátí singleton třídu implementující rozhraní IFileLoader. Každý datový formát, který je aplikací podporován, má v balíčku servisních tříd vlastní třídu, implementující výše zmíněné rozhraní, které zajišťuje základní operace se souborem.

Reprezentace obrazu

HDR obraz je v aplikaci zastoupen třídou HDRImage, která v sobě uchovává jak zdrojová HDR data tak i transformovaná LDR data. HDR data jsou tvořena poli, obsahující čísla s plovoucí řádovou čárkou. Kromě tří polí, kde každé pole představuje jednu složku barevného prostoru RGB, je zde ještě čtvrté pole, uchovávající jas každého pixelu. LDR data jsou tvořena obdobným způsobem.

Rozdíl je v datovém typu čísel v poli. Zde jsou čísla s plovoucí řádovou čárkou nahrazena celočíselnými hodnotami.

Po vytvoření instance třídy HDRImage je nutné zavolat metodu initImage, která na základě vstupních dat určí počet kanálů a počet pixelů a vhodně uspořádá data v objektu. Jakmile je proces mapování tónů dokončen a jsou k dispozici transformovaná LDR data, předá se objekt k vykreslení do okna aplikace. K samotnému vykreslení slouží třída MKCanvas, která využívá vykreslovací metody komponenty JComponent z balíčku java.awt.

4.4.2 Mapování tónů

Operátory pro mapování tónů jsou umístěny v balíčku toneMappers, kde je každý operátor zastoupen jednou nebo více tříd podle složitosti. Pro použití některého z operátorů se musí předat třídě HDRImage vytvořená instance třídy operátoru. Při následném vykreslování obrazu do okna aplikace zajistí třída HDRImage jeho použití.

Lineární mapování

Lineární mapování zajišťuje třída BasicToneMapper. Její konstruktor přijímá parametry, určující šířku okna expozice, pomocí kterých se lze v dynamickém rozsahu snímku libovolně pohybovat. Před transformací hodnot se vypočte ovlivňující globální faktor, který se aplikuje na každý pixel.

```
float factor =  255 / (endPosition - startPosition);
for (int i = 0; i < width*height; i++) {
    newLuminance[i] = (originalLuminance[i] - startPosition) * factor;
}
```

Obrázek 16 Ukázka kódu lineární transformace v jazyce JAVA

SimpleSpatial TMO

V SimpleSpatial Tone MappingOperatoru hraje klíčovou hodnotu uživatelský parametr C, který se předá do konstruktoru třídu PattanaikToneMapper. Při mapování tónů třída nejprve zjistí průměrnou hodnotu obrazu a pomocí následujícího algoritmu provede samotné mapování. Aplikace podporuje i lokální

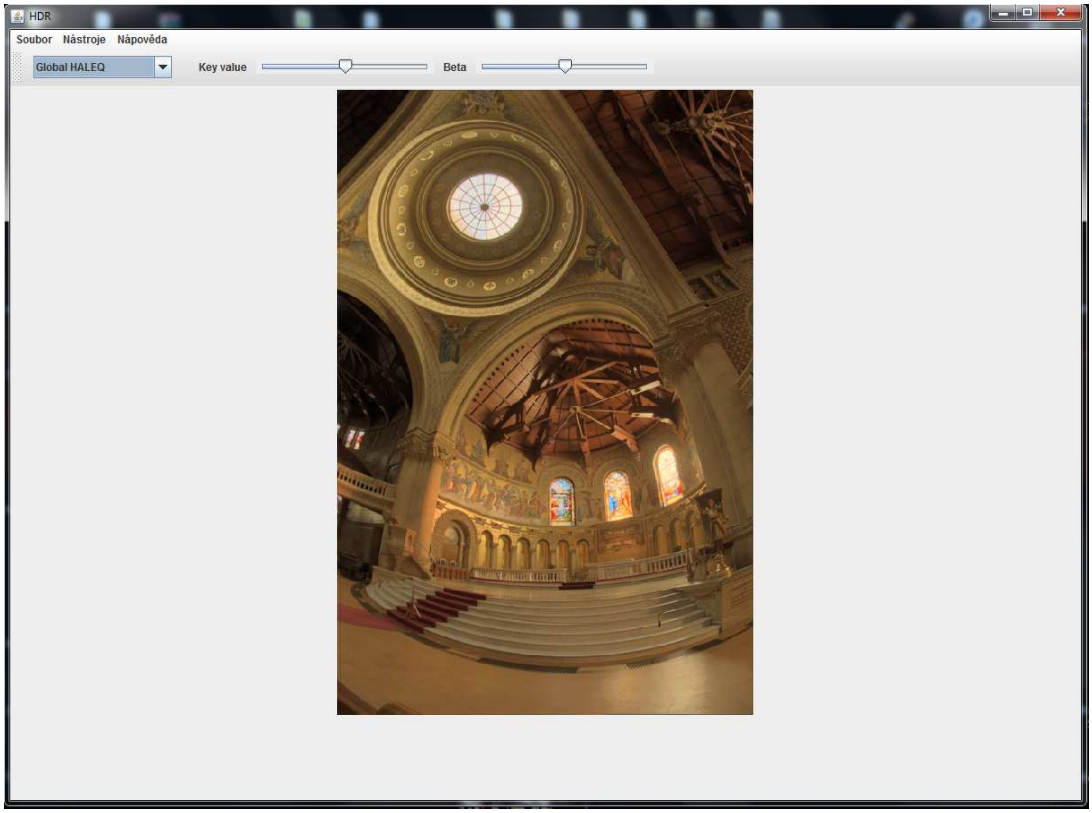
verzi tohoto operátoru a je implementována ve třídě LocalPattanaikToneMapper. K vyhledání správné hodnoty okolí se využívá pouze medián.

Novel Histogram Adjustment

Mapovací funkci operátoru Novel Histogram Adjustment zajišťuje třída LocalHaleqToneMapper, která přijímá dva uživatelské parametry, ovlivňující ostrost a vliv vyrovnání histogramu na každý blok. V prvním kroku mapování je z minimální, maximální a průměrné hodnoty určena klíčová hodnota k, pro kterou se vypočte pomocí implementace Newtonovy metody parametr τ do rovnice (14). Ve druhém kroku se podle rovnice zpracují všechny pixely obrazu a výsledek je uložen do výstupního pole.

4.4.3 Dialogy aplikace

Po spuštění aplikace se zobrazí hlavní ovládací okno, sloužící k zobrazování HDR obrazů na běžném monitoru. Po načtení obrazu do okna si uživatel může zvolit, jaký algoritmus mapování bude na obraz aplikován. Vybírat může z roletky mapovacích funkcí. Po výběru dané funkce se ihned automaticky upraví ovládací prvky daného algoritmu vedle roletky.



Obrázek 17 Hlavní okno aplikace HDRonCPU, které nabízí volbu výběru operátoru a následné zobrazení obrazu

4.5 Zobrazení medicínských dat

4.5.1 Aplikace HDR_3D

Další z implementovaných částí jsou metody pro zobrazení volumetrických dat. Aplikace je napsaná v jazyce JAVA a je v příloze umístěna jako spustitelný JAR soubor.

Načítání dat

Aplikace umí načítat soubory s volumetrickými daty s koncovkou img. Údaje o struktuře si aplikace zjistí ze stejnojmenného souboru s koncovkou hdr, ve kterém je umístěna hlavička s informacemi o rozměrech, dimenzích apod. Jelikož se nejedná o datově velké soubory, jsou po zavolání funkce *readFile()* načteny přímo do operační paměti. Aplikace podporuje zobrazování vrstev pouze z jednoho směru. Pro získání jedné vrstvy se volá metoda *readChunk()*, která vrátí hodnoty

pixelů pro danou vrstvu. Každý pixel může nabývat hodnot 0-256. Metoda dále nastaví čtecí pozici v souboru na další obraz, takže při opětovném zavolání *readChunk()* se vždy vrátí obraz následující.

Skládání obrazů

O rozšíření dynamického rozsahu, které je popsané v kapitole KAPITOLA, se stará aditivní shader. Vstupem do shaderového programu jsou vždy pouze dvě textury, jejichž pixely jsou následně sečteny. Protože jsou hodnoty přepočteny na rozsah 0-1 a výstup je také očekáván v rozsahu hodnot 0-1, došlo by při součtu k přetečení hodnot a jejich následnému ořezání. Z toho důvodu je nejprve proveden přepočet na původní rozsah 0-255 a až poté proveden součet a zpětný přepočet na rozsah 0-1.

```
vec4 texColor1_255 = texColor1 * imgIndex*255;

vec4 texColor2_255 = texColor2 * 255;

outColor = vec4((texColor1_255.rgb + texColor2_255.rgb) / ((imgIndex+1)*255), 1);
```

Obrázek 18 Implementace sčítání voxelových vrstev

Při každém následujícím kroku se k nově vzniklé textuře přidá jedna nová a proces sečtení proběhne znovu. Program bere ohled na počet předcházejících kroků, aby nedošlo k přetečení hodnot, prostřednictvím parametru imgIndex.

Mapování tónů a vykreslování

Aplikace podporuje pouze SimpleSpatial TMO shader a jeho implementace je stejná, jako v aplikaci HDRonGPU. Protože se jedná o šedo-tónový obraz, odpadá zde nutnost dekompozice jasové složky a gama korekce barev. Výstupní hodnotu z rovnice (8) lze tedy použít přímo jako hodnotu všech jednotlivých barevných kanálů RGB. Dialogy aplikace HDR_3D jsou stejné, jako u aplikace HDRonGPU (Obrázek 14).

5 Výsledky

5.1 Snímání HDR dat

V rámci testování byla vytvořena sada obrazů pomocí aplikace HDR Timelapse, které snímala scénu s vysokým dynamickým rozsahem po dobu jedné hodiny. Dávky snímků byly pořízeny v intervalu 8 vteřin, kde každá dávka obsahuje jeden správně nasvícený snímek, jeden s kompenzací expozice -2EV a jeden s kompenzací expozice +2EV. Celkem bylo nasnímáno 1284 obrazů v rozlišení 1920 x 1080 pixelů a celková velikost všech snímků byla568 megabajtů.

Časové rozmezí snímků v jedné dávce bylo velké 3-4 vteřiny. Mobilní zařízení nebylo schopné rychleji přizpůsobit změnu expozice, a protože z jedné dávky snímků byl vytvořen jeden HDR obraz, vznikaly v obraze nechtěné artefakty v podobě takzvaných „duchů“ v případě, když se ve scéně vyskytly rychleji se pohybující objekty.

Obrázek 19 „Duchové“ v obraze při složení třech snímků scény s pohybujícími se objekty

Dále bylo testováno, kolik času je potřeba k vytvoření HDR obrazu ze zdrojových snímků v různých rozlišeních a jaká bude výsledná velikost. Výsledky měření jsou uvedeny v tabulce (1)

Rozlišení	rychlost (ms)	velikost (kb)	
		1 snímek	Video(17 vteřin/24 fps)
1920x1080	2 416	24 301	9 914 401
1366x768	680	12 295	5 015 953
720x405	350	3 418	1 394 213

Tabulka 1 Naměřené časy vytváření HDR obrazů ze třech snímků ve formát Jpeg

5.2 HDR rendering CPU

Rychlost zpracování HDR dat na procesoru počítače bylo testováno na třech stejných snímcích s různým rozlišením. Testovací počítač obsahoval procesor AMD Phenom II P820 Triple-Core 1.8 GHz. Dokonce ani při nižších rozlišení obrazu nebyl schopen procesor zpracovat data dostatečně rychle. Při započtení času, potřebného k načtení obrazů z pevného disku a dalších operací, není procesor schopný vykreslit více než jeden snímek za vteřinu při rozlišení 720x405 (Tabulka 2). To je k 24fps, potřebných pro video, hodně vzdálené.

HDR map haleq	
Rozlišení	rychlost (ms)
1920x1080	5 129
1366x768	1 962
720x405	700

Tabulka 2 Porovnání rychlostí zpracování HDR dat prostřednictvím Haleq operátoru na CPU

5.3 HDR rendering GPU

Na stejných snímcích bylo testováno vykreslování i na grafické kartě. Testovací grafická karta AMD Mobility Radeon HD 5000 Series má následující parametry:

- Frekvence jádra 750MHz
- Frekvence pamětí 800MHz
- Velikost dedikované paměti 512MB
- 80 shaderových jednotek

Testování bylo zaměřeno jednak na rychlost zpracování s ohledem na požadavek zobrazení videa a na kvalitu výsledného obrazu z pohledu transformace HDR na LDR obraz.

5.3.1 Rychlost

Díky paralelnímu zpracování je rychlost oproti zpracování na procesoru počítače výrazně rychlejší. Větší vliv na rychlost nemá ani rozlišení obrazů, dokonce i v HD rozlišení stíhá grafická karta vykreslovat snímky ve frekvenci až 300 fps. Měření uvedená v tabulce (3) obsahují čas přenesení textury na grafickou kartu a její následné zpracování pomocí zvoleného operátoru a vykreslení na obrazovku. Uvedené časy nezahrnují, ale diskové operace, které jsou uvedeny v tabulce (4).

Rozlišení	GPU (ms)		CPU (ms)
	Haleq	Simplespatial	Haleq
1920x1080	2,964	2,570	5 129
1366x768	2,578	2,444	1 962
720x405	2,271	2,208	700

Tabulka 3 Naměřené časy, potřebné k transformaci HDR rozsahu na rozsah LDR

Každý obraz před vykreslením musí být načten z pevného disku. Tato operace bohužel trvá nejdéle a značně limituje výkon grafické karty. Hodnoty v tabulce (4) byly naměřeny na pevném disku s rychlostí otáček 5400RPM. Cílové frekvence 24 fps je možné dosáhnout pouze se snímky s nižším rozlišením nebo za použití rychlejších disků jako jsou například disky SSD.

Rozlišení	rychlost (ms)	fps
1920x1080	320,7	3,1
1366x768	176,6	5,7
720x405	49,1	20,4

Tabulka 4 Rychlost načítání jednoho HDR snímku ve formátu HVF

5.3.2 Vizuální kvalita

HDR snímek na obrázku (20) je jedním ze sady nasnímaných testovacích obrazů a obsahuje nejmenší kontrastní poměr o velikost 0,000003814697266 a celkový dynamický rozsah obrazu je 1 : 823 835 392. Testovací obrazy jsou po převedení na HDR snímky a po aplikaci lineární transformace zobrazeny velice tmavě. Při

klasickém zesvětlení pomocí přičtení konstantní hodnoty ke všem pixelům obrazu vzniká nehezký vybledlý obraz, protože přičítaná hodnota nebere ohled na původní hodnotu jasu. Aby byl obraz správně osvětlen ve všech oblastech, je nutné využít některé z technik mapování tónů.

Obrázek 20 Vlevo je obraz mírně ztmavený tak, aby byly dobře pozorovatelné detaily na oblez. Vpravo je obraz naopak zesvětlen, aby byly vidět detaily budov.

Na obrázku (21) je možné pozorovat výsledek mapování po aplikaci operátoru HALEQ, popsaného v kapitole 2.

Obrázek 21 Ukázka výsledku mapování Haleq operátoru vpravo, vlevo původní obraz

Operátor zvolil správné hodnoty jasů pro všechny oblasti obrazu. Navíc nabízí výhodu pro zpracování v reálném čase, protože nepotřebuje zadat žádné uživatelské parametry a vždy korektně určí správnou celkovou světlost obrazu a nedochází k problikávání obrazu. Nicméně lze pozorovat barevný posun, který se nepodařilo kompenzovat ani pomocí gama korekce.

Druhým z testovaných operátorů je SimpleSpatial TMO a výsledek transformace, při použití globální verze, je možné pozorovat na obrázku (22).

Obrázek 22 Ukázka výsledku mapování Simple Spatial TMO (globální verze) vpravo, vlevo původní obraz

I druhý operátor provedl správné osvícení všech oblastí obrazu a to dokonce při zachování správných barev. Bohužel je ale nutné zadat uživatelský parametr v podobě klíčové hodnoty, která se ale pro každý obraz může lišit. Ve výsledku tak dostáváme při zobrazování snímků za sebou mírné problikávání, protože není možné, aby uživatel stihl změnit parametr, který lze navíc určit pouze podle subjektivního dojmu, nikoli pomocí nějakého výpočtu. Řešením je využití jasových hodnot předešlých obrazů pro výpočet aktuálního obrazu, které zajistí plynulejší přechod.

Obrázek 23 Ukázka výsledku mapování Simple Spatial TMO (lokální medián verze) vpravo, vlevo původní obraz

Operátor SimpleSpatial TMO podporuje také lokální variantu. Výsledek při použití mediánu je zobrazen na obrázku (23). Protože vždy k výpočtu daného pixelu využívá hodnot pixelů v lokálním okolí, dojde nejen ke správnému osvětlení všech detailů v obraze, ale dokonce k jejich zvýraznění. Výsledkem je vizuálně lepší obraz za cenu pomalejšího zpracování.

Obrázek 24 Ukázka výsledku mapování Simple spatial TMO (lokální verze s průměrem) vpravo, vlevo původní obraz

V rámci této práce byla ještě implementována možnost hledání správné hodnoty nikoli podle mediánu, ale pomocí aritmetického průměru (Obrázek 24). K výpočtu není nutné řazení polí, takže je možné zohlednit širší okolí pixelu a více zvýraznit detaily.

Verze operátoru	Rychlost zpracování (ms)
Globální	3
Lokální 5x5 (medián)	35
Lokální 21x21 (průměr)	27

Tabulka 5 Rozdíly časů, potřebných pro transformaci HDR dat, v závislosti na zvoleném typu operátoru

Řazení polí není na grafických kartách efektivní a z toho důvodu není výhodné medián pro výpočet použít. Na druhou stranu by ale mohl podávat přesnější informace o okolí než aritmetický průměr.

Celkové vizuální srovnání operátorů je možné si prohlédnout v tabulce 6 a jejich časovou náročnost v tabulce 5.

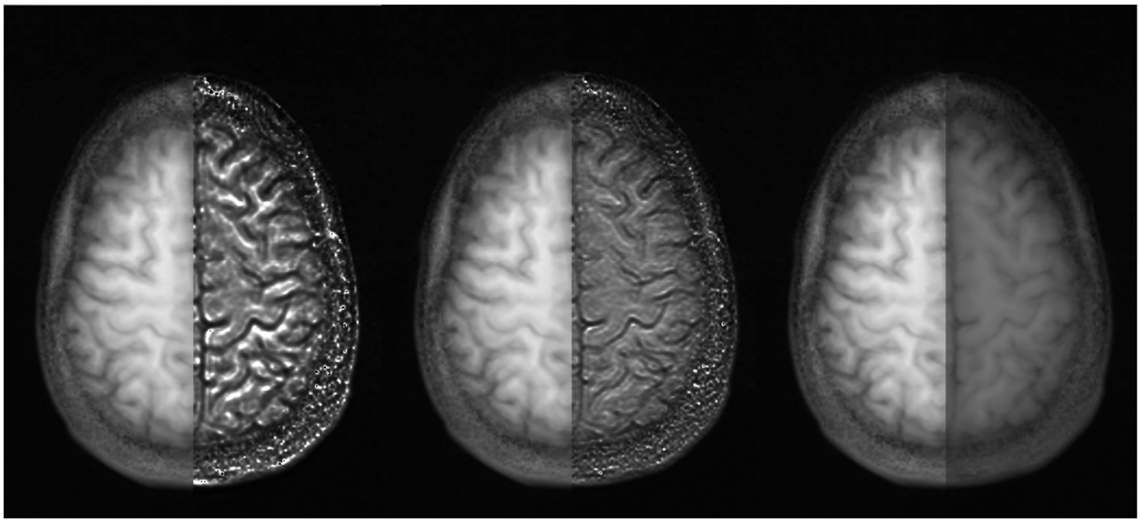
Operátor	Detaily stínů	Blikání obrazů	Volba parametrů	Věrnost barev
Žádný	Nízké	Ne	Ne	Vysoká
Haleq	Vysoké	Ne	Ne	Nízká
Simplespatial	Vysoké	Ano	Ano	Vysoká

Tabulka 6 Srovnání operátorů podle vizuální kvality

5.4 Zobrazení medicínských dat

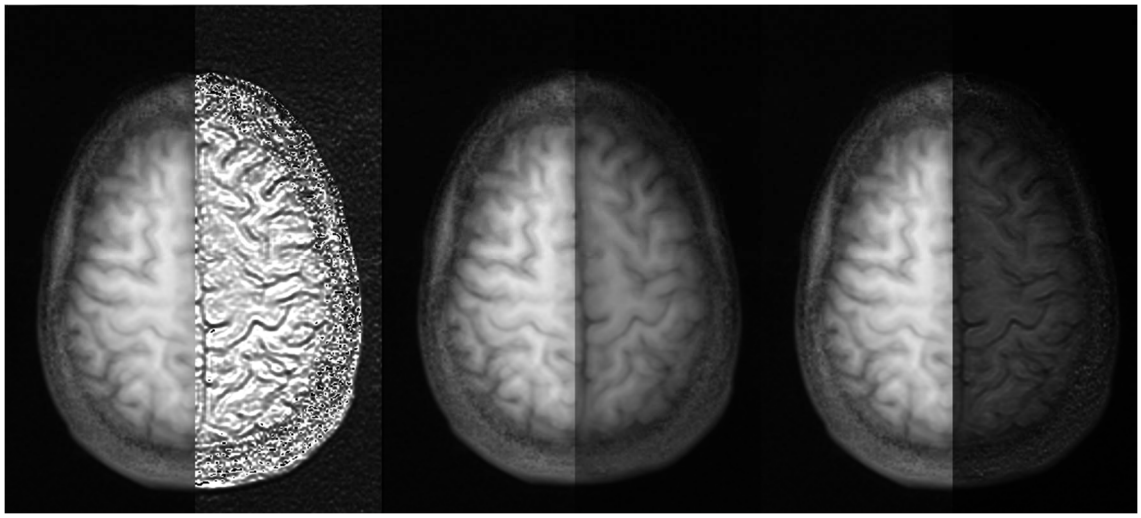
Technologie HDR byly otestovány také na datech z magnetické rezonance. Testovací data jsou složena z voxelové mřížky o velikosti 176 x 236 x 187. Na obrázku 24 je zobrazen řez těchto dat, který je složen ze dvaceti vrstev. Hodnota

každého pixelu může nabývat hodnot 0-255, teoretický dynamický rozsah může být při kumulaci vrstev až 1 : 5 100. Běžný monitor dokáže ale zobrazit u šedotónového obrazu pouze rozsah 1 : 255.



Obrázek 25 Vlevo je šířka okolí nastavena na 13x13, uprostřed na 7x7 a vpravo na 1x1, obraz je vždy rozdělen pro porovnání s transofmací pomocí průměrování

Jako metoda transformace dynamického rozsahu byl zvolen operátor SimpleSpatial TMO, protože umožňuje pomocí uživatelských parametrů měnit výsledek transformace. Na obrázku 24 je obraz vždy rozdělen na polovinu a dynamický rozsah levé poloviny je pro srovnání pouze zprůměrován a na pravou polovinu byl aplikován operátor mapování tónů. Šířka okolí lokální verze operátoru slouží jako zvýrazňovač detailů, a čím větší okolí je, tím více jsou detaily zvýrazněny. Naopak při šířce okolí o velikost 1x1 pixel se jedná o globální variantu, která podává srovnatelné výsledky s obrazem zprůměrovaným.



Obrázek 26 Vlevo je klíčová hodnota nastavena na 0,1; uprostřed na 0,5 a vpravo na hodnotu 1

Obrázek 25 ukazuje vliv klíčové hodnoty (key value) na výsledný obraz. Čím menší hodnota je, tím více se zvýrazní informace. Při příliš nízké hodnotě může dojít k přetečení hodnot a přepálení obrazu.

Závěr

Cílem diplomové práce bylo prozkoumat, implementovat a otestovat metody pro zpracování sekvence obrazů s vysokým dynamickým rozsahem v reálném čase. Zahrnuta byla problematika snímání, reprezentace a zobrazení obrazu s vysokým dynamickým rozsahem s důrazem na co nejkratší čas zpracování dat. Proto bylo řešení využívající technologii programovatelných grafických karet, umožňující rychlé zpracování datově rozsáhlých souborů. Vybrané metody pro zpracování obrazů HDR byly implementovány, otestovány a porovnány na testovacích datových souborech. Výsledkem je sada aplikací, umožňující celý proces zpracování HDR dat, od jejich snímání až po jejich zobrazování. Aplikace HDR_Timelapse umožňuje snímat HDR obrazy na zařízeních s operačním systémem Android a umožňuje snímat nejen sadu LDR obrazů, ze kterých vznikne jeden snímek HDR, ale dokáže snímat obrazy opakovaně a vytvářet sady HDR obrazů, které aplikace HDRonGPU dokáže zobrazovat ve zvolené frekvenci a vytváří tak HDR video.

Pro zobrazení HDR obrazů byly implementovány globální a lokální metody, založené například na lineární transformaci nebo vlastnostech lidského oka. Všechny implementované operátory pro mapování tónů jsou využitelné pro zpracování v reálném čase. Každý má ale své nevýhody. Lineární mapování je závislé na rozložení jasových hodnot v obraze a v případě obrazu s nevyrovnaným histogramem dojde ke špatně osvětlenému výsledku. S tímto problémem si dokáže poradit operátor Simplespatial TMO, ale vyžaduje zadání uživatelských parametrů, které se pro každý obraz individuálně liší a při zpracování v reálném čase je nemožné tyto parametry včas měnit. Operátor podporuje i lokální verzi, která úspěšně funguje jako zvýrazňovač detailů. Posledním operátorem byl Haleq TMO, který řeší všechny předešlé nedostatky. Ke správnému osvětlení obrazu nevyžaduje zadání uživatelských parametrů a proto je pro zpracování dat nejvhodnější. Bohužel byla implementována pouze jeho globální varianta, protože lokální nepodporuje paralelní zpracování a tudíž by její výpočet nebyl na GPU efektivní. Volba zpracování dat na grafických kartách se ukázala jako velice

vhodná, při rozumném rozlišení se podařilo úspěšně přehrávat snímky v požadované frekvenci 24fps. Při vyšších rozlišeních jako je například HD už se data nestíhala načítat z pevného disku a frekvence rapidně klesala. Řešením by byla volba disků, cílených na rychlost čtení jako jsou například disky SSD.

Aplikováním globálních mapovacích technik na medicinská data bylo dosaženo srovnatelného výsledku, jako při použití běžného aritmetického průměru. Určité zlepšení je možné pozorovat při aplikaci lokálních technik, kde dojde ke zvýraznění detailů. A díky nelineární transformaci by mohl pomoci v odhalení objektů, složených pouze z velice nízkých intenzit.

Mezi vhodné rozšíření práce patří implementace shaderových programů ne na desktopových grafických kartách ale přímo na grafických kartách mobilních zařízení. Při využití novějšího rozhraní operačního systému Android pro kontrolu fotoaparátu CameraApi2, který by umožnil rychlé snímání obrazu nebo dokonce přímé streamování HDR dat, by bylo možné uživateli přímo zobrazit obraz, který kamera vidí, na displej zařízení bez větší prodlevy.

Seznam použité literatury

[1] AGGARWAL, Manoj a Narendra AHUJA. Split Aperture Imaging for High Dynamic Range. *Split Aperture Imaging for High Dynamic Range* [online]. 2001 [cit. 2014-02-17]. Dostupné z: http://vision.ai.illinois.edu/publications/download2004j2.pdf

[2] Ekvalizace histogramu. BARVA, Martin. *CVUT* [online]. 2005 [cit. 2014-04-17]. Dostupné z: http://cmp.felk.cvut.cz/~perdom1/vyuka/dzo/cv2/cv2_histeq/cv2_histeq.html

[3] BLOCH, Christian. HDRI pro fotografy a počítačové grafiky: High Dynamics Range Imaging = zobrazení vysokého dynamického rozsahu. Vyd. 1. Brno: Zoner Press, 2008. ISBN 978-80-7413-001-4.

[4] COFFIN, David. Decoding raw digital photos in Linux. *Decoding raw digital photos in Linux* [online]. [cit. 2013-09-21]. Dostupné z: http://www.cybercom.net/~dcoffin/dcraw/

[5] DEBEVEC, Paul E. a Jitendra MALIK. Recovering High Dynamic Range Radiance Maps from Photographs. *Recovering High Dynamic Range Radiance Maps from Photographs* [online]. 1997 [cit. 2013-07-10]. Dostupné z: http://www.debevec.org/Research/HDR/debevec-siggraph97.pdf

[6] DUAN, Jiang, Marco BRESSAN, Chris DANCE a Guoping QIU. Tone-mapping high dynamic range images by novel histogram adjustment. *Tone-mapping high dynamic range images by novel histogram adjustment* [online]. 2010 [cit. 2013-12-04]. Dostupné z: http://ima.ac.uk/papers/duan2010.pdf

[7] DURAND, Fredo a Julie DORSEY. Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. *Fast Bilateral Filtering for the Display of High-Dynamic-Range Images* [online]. [cit. 2014-03-19]. Dostupné z: http://people.csail.mit.edu/fredo/PUBLI/Siggraph2002/DurandBilateral.pdf

[8] FATTAL, Raanan, Dani LISCHINSKI a Michael WERMAN. Gradient Domain High Dynamic Range Compression. *Gradient Domain High Dynamic Range Compression* [online]. [cit. 2014-03-19]. Dostupné z: http://www.cs.huji.ac.il/~danix/hdr/hdrc.pdf

[9] GREEN, Simon. *The OpenGL Framebuffer* [online]. In: . San Francisco, 2005 [cit. 2016-04-29]. Dostupné z: http://http.download.nvidia.com/developer/presentations/2005/GDC/OpenGL_Day/OpenGL_FrameBuffer_Object.pdf

[10] HRUBÝ, Petr. *Obecné výpočty na GPU* [online]. 2008 [cit. 2016-04-29]. Dostupné z: http://wh.cs.vsb.cz/mil051/images/d/d7/PAP_Vyu%C5%BEit%C3%AD_grafick%C3%BDch_karet_pro_obecn%C3%A9_v%C3%BDpo%C4%8Dty_(GPGPU)_(Petr_Hrub%C3%BD).pdf

[11] CHUN HING LO, Raymond a Steve MANN. *High Dynamic Range (HDR) Video Image Processing For Digital Glass* [online]. University of Toronto, 2012 [cit. 2016-04-29]. Dostupné z: http://www.eyetap.org/papers/docs/hdr_digital_glass_p1477.pdf

[12] KOŘÍNEK, Miroslav. Zpracování obrazu HDR: Snímání a zpracování obrazu HDR. Hradec Králové, 2014. UHK.

[13] LÁNĚK, Jan. *Obraz s vysokým dynamickým rozsahem* [online]. Univerzita Pardubice, 2011 [cit. 2014-03-17]. http://hdl.handle.net/10195/41989. Dostupné z: http://hdl.handle.net/10195/41989. Bakalářská práce. Univerzita Pardubice.

[14] PATTANAIK, Sumanta a K.K. BISWAS. A Simple Spatial Tone Mapping Operator for High Dynamic Range Images. *A Simple Spatial Tone Mapping Operator for High Dynamic Range Images* [online]. 2004 [cit. 2014-01-04]. Dostupné z: http://www.researchgate.net/publication/228411975_Fast_Algorithm_for_Completion_of_Images_with_Natural_Scenes/file/9c960517bc9606657f.pdf

[15] REINHARD, Erik. *High dynamic range imaging: acquisition, display, and image-based lighting*. Amsterdam ; Boston: Morgan Kaufmann Publishers, c2006. Morgan Kaufmann series in computer graphics and geometric modeling. ISBN 01-237-0475-8.

[16] REINHARD, Erik, Michael STARK, Peter SHIRLEY a James FERWERDA. Photographic Tone Reproduction for Digital Images. *Photographic Tone Reproduction for Digital Images* [online]. [cit. 2014-02-01]. Dostupné z: http://www.cs.utah.edu/~reinhard/cdrom/tonemap.pdf

[17] ŠŤOVÍK, Petr. Analýza obrazu v reálném čase: Detekce prostorové souřadnice předmětu s užitím GPU. Hradec Králové, 2009. UHK.

[18] WARD, Greg. *JPEG-HDR: A Backwards-Compatible, High Dynamic Range Extension to JPEG* [online]. 2005 [cit. 2016-04-29]. Dostupné z: http://www.anyhere.com/gward/papers/cic05.pdf

[19] *Temporally Coherent Local Tone Mapping of HDR Video* [online]. Asia: ACM SIGGRAPH, 2014, 2014 [cit. 2016-01-01]. Dostupné z: https://www.disneyresearch.com/publication/hdr-video/

[20] *AdobeRGB vs. sRGB* [online]. Los Angeles: fstoppers.com, 2013 [cit. 2016-04-29]. Dostupné z: https://fstoppers.com/pictures/adobergb-vs-srgb-3167

[21] *High-dynamic range video acquisition with a multiview camera* [online]. BELLINGHAM: 1000 20TH ST, PO BOX 10, BELLINGHAM, WA 98227-0010 USA, 2012, 2012 [cit. 2016-01-01]. Dostupné z: http://apps.webofknowledge.com/full_record.do?product=UA&search_mode=GeneralSearch&qid=21&SID=T2sBSsaSSbw95bgZ4IO&page=1&doc=1

[22] *Art-directable Continuous Dynamic Range video* [online]. OXFORD, ENGLAND: THE BOULEVARD, LANGFORD LANE, KIDLINGTON, OXFORD OX5 1GB, ENGLAND, 2015, 2015 [cit. 2016-01-01]. Dostupné z: http://apps.webofknowledge.com/full_record.do?product=UA&search_mode=GeneralSearch&qid=17&SID=T2sBSsaSSbw95bgZ4IO&page=1&doc=1

[23] Fotografie s vysokým dynamickým rozsahem. *Mendelova univerzita v Brně* [online]. Brno: Mendelova univerzita, 1999 [cit. 2016-04-29]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=18432

[24] *AN IMPROVED HDR IMAGE SYNTHESIS ALGORITHM* [online]. NEW YORK: 345 E 47TH ST, NEW YORK, NY 10017 USA, 2009, 2009 [cit. 2016-01-01]. Dostupné z: http://apps.webofknowledge.com/full_record.do?product=UA&search_mode=GeneralSearch&qid=6&SID=T2sBSsaSSbw95bgZ4IO&page=1&doc=2

[25] *Multi-Texturas* [online]. Mario Camillo [cit. 2016-04-29]. Dostupné z: http://www.dca.fee.unicamp.br/courses/IA369E/2s2010/projects/camillo_curado/algoritmos.html

Přílohy

1) Obrazový materiál

Obrázek 27 Porovnání LDR a HDR obrazu [12]

Obsah CD

1. Text diplomové práce

2. Zdrojový HVF soubor, obsahující sekvence HDR obrazů

3. Obraz kaple v RGBE formátu

4. Zdrojové kódy aplikace HDRonCPU

5. Zdrojové kódy aplikace HDRonGPU

6. Zdrojové kódy aplikace HDR_Timelapse

7. Ukázky výstupů aplikací

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Kořínek Miroslav	1. Máje 555, Rokytnice v Orlických horách	I1463

TÉMA ČESKY:

Akcelerace zpracování HDR rastrového obrazu na GPU

TÉMA ANGLICKY:

GPU accelerated HDR image processing

VEDOUCÍ PRÁCE:

Ing. Bruno Ježek, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl práce: Prozkoumat, implementovat a otestovat metody pro zpracování sekvence obrazů s vysokým dynamickým rozsahem v reálném čase

Osnova:
1. Vytvořit přehled softwarových prostředků pro získání, uchování a zpracování HDR videa
2. Zaměřit se na mapování tónů sekvence HDR obrazů na rozsah monitoru s pomocí GPU, kódování digitálního videa a optimalizaci
3. Prozkoumat možnosti využití videa s vysokým dynamickým rozsahem
4. Navrhnout metodiku testování včetně datových souborů pro testování
5. Provést testování a srovnání s dostupnými řešeními
6. Zhodnotit dosažené výsledky

SEZNAM DOPORUČENÉ LITERATURY:

1) Bloch Christian, HDRI pro fotografy a počítačové grafiky, 2008 Zoner Press, ISBN 978-80-7413-001-4
2) Moeslund, Thomas B., Introduction to Video and Image Processing, ebook, ISBN 978-1-4471-2503-7
3) Yao Wang, Jorn Ostermann, Ya-Qin Zhang, Video Processing and Communications, 2001 Prentice Hall, ISBN-10: 0130175471

Podpis studenta: ..... **Datum:** 5.3.2015

Podpis vedoucího práce: ..... **Datum:** 5.3.2015