VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NEFOTOREALISTICKÉ ZOBRAZOVÁNÍ

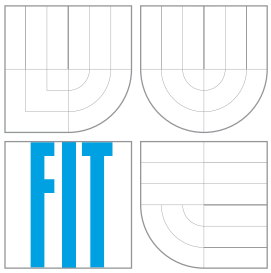
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAROSLAV ŽILÁK

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NEFOTOREALISTICKÉ ZOBRAZOVÁNÍ

NON-PHOTOREALISTIC RENDERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAROSLAV ŽILÁK
AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL ŽÁK
SUPERVISOR

BRNO 2010

Abstrakt

Ústředním cílem této práce je kromě úvodu do problematiky nefotorealistického zobrazování, poskytnutí návrhu a možného řešení realizace systému, který toto zobrazování umožňuje ve formě obrazového post-processingu. Popisuje řešení dostatečně flexibilní na to, aby umožňovalo uživateli kromě již vestavěných filtrů, tvořit filtry vlastní a rekurzivně je využívat při tvorbě dalších, nových filtrů, pro tento typ zobrazování. Její součástí je tak i jakýsi manuál, který popisuje základní syntaxi a možnosti tohoto systému.

Abstract

The main purpose of this thesis besides providing the reader with basics of nonphotorealistic rendering, is concept and possible solution for system realization, which is capable of this rendering using image post-processing. This thesis describes solution that is sufficiently flexible to support use of integrated filters along with the creation of new ones that can be recursively used to create another filters for this type of rendering. Thus part of this thesis is some kind of manual which provide basic syntax description and possibilities of use.

Klíčová slova

Nefotorelistické zobrazování, digitální obraz, filtrace obrazu, spracování obrazu, filtry, simulace, náhodnost, post-processing

Keywords

Non-photorealistic rendering, digital image, image filtering, image processing, filters, simulation, randomness, post-processing

Citace

Jaroslav Žilák: Nefotorealistické zobrazování, bakalářská práce, Brno, FIT VUT v Brně, 2010

Nefotorealistické zobrazování

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Pavla Žáka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jaroslav Žilák
17. mája 2010

Poděkování

Týmto sa chcem poďakovať pánovi Ing. Pavlovi Žákovi, vedúcemu tejto práce, za jeho odbornú pomoc, cenné rady a venovaný čas.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Obraz pre človeka predstavuje najľahšie spracovateľný zdroj veľkého množstva informácii istého zamerania. Slúži ako komunikačný prostriedok, estetickým účelom, prípadne ako forma uloženia, pre nás cenných dát. Vo všeobecnosti je snaha o čo najväčší počet obrazom obsiahnutých informácii. Tento počet priamo súvisí s detailom obrazu, avšak obraz s prehnane veľkým počtom informácii môže pre pozorovateľa pôsobiť mätúco, ba až chaoticky. Obraz tým stratí svoj pôvodne myslený zámer. Inými slovami, fotorealisticky verné zobrazenie nie je vždy vhodné. Do hry teda vstupuje zobrazovanie nefotorealistické, ktoré slúži najmä pre zvýraznenie, resp. extrakciu podstatných dát obrazu. Výsledkom je jeho zjednodušená podoba zanedbávajúca detail, no zachovávajúca pôvodnú myšlienku obrazu. Samostatnou kapitolou nefotorelistického zabrazovania je potom napodobňovanie umeleckej tvorby obrazu, v rámci ktorej sa využíva simulácia troch základných maliarskych nástrojov a kresliacich techník.

Jadrom tejto práce je systém, ktorý toto zobrazenie umožňuje s využitím filtrácie, postprocesingu. Svojou realizáciou sa pritom snaží prekonať iné, podobné aplikácie, najmä čo sa týka rýchlosti filtrácie, fexibility a kvality výsledného obrazu. Filtrácia zahŕňa okrem všeobecne známych filtrov aj filtre realizujúce simuláciu umeleckej tvorby a kresliacich techník. Celá filtrácia je pritom riadená prostredníctvom skriptov a ich následnou interpretáciou. Súčasťou práce je teda okrem jednotlivých filtrov aj návrh a popis implementácie interpretera a ním relizovaného skriptovacieho jazyka.

Práca obsahuje celkom 8 kapitol. Druhá kapitola vysvetľuje základné princípy počítačovej grafiky a reprezentáciu obrazu počítačom ako takého z hľadiska dát. Jej účelom je čitateľovi poskytnúť potrebný teoretický základ, o ktorý sa opiera zbytok tejto práce. Tretia kapitola potom pojednáva o základných a všeobecne známych filtroch a o filtrácii vôbec. Popisuje princíp a využitie týchto filtrov spolu s názornými príkladmi ich výstupu. Štvrtá, posledná kapitola teoretickej časti, sa zaoberá samotným nefotorealistický zobrazovaním. Bližšie popisuje čo všetko toto zobrazovanie obnáša a kde ho možno uplatniť. Piata kapitola popisuje návrh prezentovaného systému, ktorý popisuje z hľadiska jeho štruktúry, rovnako ako aj jeho základné vlastnosti. Šiesta kapitola potom detailne popisuje samotnú implementáciu tohto systému. Popisuje syntax a sémantiku jazyka, ktorý tento systém pre svoju činnosť využíva. Siedma, predposledná kapitola, prezentuje výsledky vo forme porovnania s obdobnými aplikáciami, najmä z pohľadu realizácie filtrov a filtrácie. Zameriava sa hlavne na ich rýchlosť, flexibilitu a kvalitu prevedenia, resp. výsledný estetický dojem. Posledná ôsma kapitola je záver, v ktorom sú stručne zhrnuté dosiahnuté výsledky a návrhy pre možné rozšírenia.

Kapitola 2

Základy počítačovej grafiky a reprezentácie obrazu

2.1 Počítačová grafika

Počítačová grafika označuje veľmi širokú oblasť informačnej techniky. Týmto termínom označujeme všetko čo súvisí s obrazom, jeho tvorbou a obrazovým spracovaním. Z hľadiska dát ju delíme na dve základne kategórie: vektorovú a rastrovú grafiku. Prvá, vektorová grafika, popisuje obraz vo forme základných objektových entít. Týmito entitami rozumieme objekty ako sú bod, krivka, kružnica, pre dvojrozmerný priestor, hranol, kocka, ... pre priestor trojrozmerný, atď. Reprezentáciu obrazu v takto uloženej forme však nie je možné ihneď zobrazovať, najmä z dôvodu, že väčšina zobrazovacích systémov nedokáže spracovať vektorové objekty. Nutnosťou je teda rasterizácia, ktorá prevedie vektorovú reprezentáciu obrazu na už zobraziteľnú rastrovú, čím sa dostávame k rasterizačnej grafike. Resterizačná grafika, ktorá využíva popis obrazu vo forme matice o pevne danej šírke a výške, tiež označovanej ako rozlíšenie, je dnes najpoužívanejšou metódou reprezentácie obrazu. Keďže je matica matematickou entitou, s výhodou ju možno použiť pre najrôznejšie matematické a iné operácie, čo je aj jadrom tejto práce a základným princípom prezentovaného nefotorealistického zobrazovania [1].

V rámci matematických operácii, budeme aspoň v tejto práci rozumieť pod pojmom základné aritmetické úkony nasledujúce operácie: sčítanie, odčítanie, násobenie, delenie a operáciu modulo. Pod pojmom iné zase bitové operácie, predovšetkým operácie boolovej algebry, bitové posuny, bitové rotácie a ďalšie zložitejšie operácie v podobe filtrov.

2.2 Digitálny Obraz

Ako už bolo v úvode naznačené, obraz je v počítačovom svete reprezentovaný vo forme dvojrozmernej matice, obrazových elementov, tiež nazývaných pixely. Matematicky vyjadrené, ide o funkciu: $f(x, y)$, kde x a y vyjadrujú koordináty, pozíciu v rámci obrazu a funkčná hodnota v týchto koordinátoch vyjadruje intenzitu farby obrazu v danom mieste. Súhrnne, teda označujeme za digitálny obraz taký obraz, ktorý je rozdelený, rasterizovaný na konečný počet pixelov usporiadaných do rastra, matice, pričom každý pixel nadobúda práve jednu úroveň intenzity z konečnej množiny úrovni intenzít [7].

Vznik digitálneho obrazu má niekoľko podôb, najčastejšou je snímanie obrazu prostredníctvom senzorových zariadení, akými sú: kamera, fotoaparát, skenery a podobne. Každé

z týchto zariadení pracuje na rovnakom princípe v nasledujúcich troch krokoch:

- vzorkovanie – prevod intenzity svetla odrážaného od snímaného objektu na napätie v určitých časových okamihoch. Obyčajne každý pixel snímajú tri snímače súčasne. Jeden pre každú farbu základného RGB farebného modelu, ktorý je bližšie vysvetlený v nasledujúcej kapitole. Počet snímačov pritom určuje rozlíšenie výsledného obrazu. Napríklad ak hovoríme o dnes už štandardnom rozlíšení *full HD*, ktoré definuje natívne rozlíšenie 1920×1080 musí mať snímacie zariadenie celkovo cca. $6,22$ milióna snímačov.

- kvantovanie – rozdelenie výstupného napätia snímačov do jedného z presne definovaných rozsahov. Ich počet určuje celkový počet rozlíšiteľných farieb, označovaný ako hĺbka farieb. Vyjadrovaná je počtom bitov na farbu. Štandardnými hodnotami sú 8, prípadne až 16 bitov na farbu.

- digitalizácia – prevod na digitálne dáta, ktorý je častokrát súčasťou kvantovania.

Druhou možnosťou je tvorba obrazu priamo v pamäti počítača, ktorá obyčajne zahŕňa tvorbu modelov zložených zo základných geometrických entít a následný výpočet ich obrazu procesom rasterizácie. Uvedená metóda je nad rámec tejto práce a nebudeme sa ňou ďalej zaoberať.

2.3 Farebné modely

Neodmysliteľnou súčasťou počítačovej grafiky a z nej vyplývajúcich postupov je pochopenie farebných modelov. Farebný model charakterizujeme ako sústavu základných farieb, ktorých kombináciou sme schopní interpretovať ľubovoľnú farbu z viditeľného spektra. Najpoužívanejšími modelmi sú: aditívny model RGB(z angl. Red, Green, Blue), subtraktívny model CMY(z angl. Cyan, Magenta, Yellow) a modely HSL(z angl. Hue, Saturation, Lightness), HVS(z angl. Hue, Value, Saturation). Počet použitých farieb u týchto modelov vyplýva z fyziológie ľudského oka. Z rovnakej fyziológie ďalej plynie aj to, že ľudské oko vníma intenzitu každej farby inak. U RGB modelu, by sme celkovú intenzitu počítali zo vzťahu 2.1 [7].

$$I = 0,597G + 0,112B + 0,281R \tag{2.1}$$

Samozrejmosťou je, že fyziológia očí je u každého z nás trochu odlišná, uvedený vzťah teda nemožno chápať ako normu, ale skôr ako akýsi globálne uznávaný priemer. Rovnako ako u RGB modelu existujú rovnice pre výpočet celkovej intenzity aj pre ostatné modely, ktoré však nebudú predmetom tejto práce.

V počítačovej grafike sa použitie modelu a hodnoty intenzít jednotlivých jeho farieb, prejavujú najmä formou ukladania dát obrazu v pamäti počítača. Typickým príkladom rozlišovania intenzity formátom je ukladanie dát vo formáte RGB pri ôsmych bitoch na každý prvok rasterizačnej matice, pixelu. V uvedenom prípade budú pre červenú a zelenú farbu vyhradené tri bity zatiaľ čo pre modrú len dva. Vplyv počtu farieb zase vidieť pri porovnaní RGB modelu s modelom GRAY(odtiene sivej), pričom sú oba v režime ôsmych bitov na kanál(farbu). Pixel uložený vo farebnom modeli RGB bude zaberať 24 bitov zatiaľ čo pri GRAY modeli len 8. V praxi to znamená, že obraz nie je maticou hodnôt, ale skôr maticou vektorov, podľa čoho je nutné prispôsobiť aj spôsob práce s obrazovou informáciou.

Špeciálnou variantou obrazovej reprezentácie, sú obrazy využívajúce palety farieb. Obraz uložený v takejto forme nie je maticou intenzít ako to bolo u predošlých variant, ale

maticou indexov, ktoré odkazujú do zoznamu použitých farieb, už zmienenej palety. Pamäťová náročnosť, teda nezávisí na bitovej hĺbke farieb, ale na počte použitých farieb, resp. veľkosti palety. Formát palety je pritom identický s formátom pixelu pri použití RGB modelu. Výhodou tohto riešenia je v prvom rade značné zníženie pamäťových nárokov, čo samozrejme platí za predpokladu minimalizácie počtu použitých farieb, najčastejšie 256. Na druhej strane však dochádza k predĺženiu doby spracovávania, čo vyplýva z nutnosti pristupovať do pamäte kvôli každému pixelu dvakrát. Prvýkrát, pre načítanie indexu z pixelu obrazu a druhýkrát, pre načítanie odpovedajúcej farby z palety. Uvedená metóda sa tiež zvykne označovať pojmom indexová organizácia obrazu.

Kapitola 3

Filtrácia obrazu a obrazové filtre

Filtrácia obrazu spadá do kategórie obrazového spracovania, presnejšie do priamej manipulácie s obrazom na úrovni jeho jednotlivých pixelov. Delíme ju na dve základné kategórie, ktorými sú: transformácia intenzity pixelov a plošná filtrácia. Filtrácia v podobe transformácie intenzity pixelov, ich RGB hodnôt, operuje vždy len s jednotlivými pixelmi obrazu. Spadajú sem techniky ako sú: prahovanie(ang. trasholding), prevod do odtieňov sivej(ang. grayscale), úpravy kontrastu, jasu, farebného vyváženia a množstvo ďalších. Naproti tomu plošná filtrácie mení hodnotu pixelu na základe jeho okolia. Do tejto kategórie filtrov zaraďujeme: rozmazávanie obrazu, zaostrovanie obrazu, hranovú detekciu a pod [2].

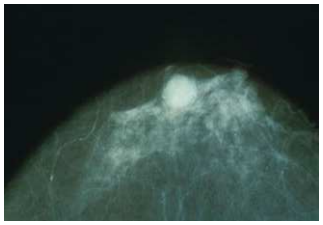
Praktické použitie filtrov sa líši od obrazu k obrazu. Tam kde je jeden filter najlepší pre úpravu fotografií, nemusí a častokrát ani nie je najlepším pre úpravu napríklad satelitných snímkov. Nasleduje niekoľko najčastejšie používaných filtrov spolu s krátkym popisom ich činnosti a využitia.

3.1 Negatív

Negatív, alebo negácia obrazu patrí medzi filtre operujúce nad jednotlivými pixemi obrazu. V rámci prechodu obrazom tento filter prepočítava jednotlivé hodnoty RGB zložiek podľa vzťahu 3.1,

$$f(x, y) = I_{max} - f(x, y) \tag{3.1}$$

kde I_{max} je maximálna hodnota intenzity farby. Výsledný obraz je ekvivalentom fotografického negatívu. Uplatnenie tento filter nachádza napríklad pri röntgenových snímkach, kde umožňuje zvýšiť kontrast mäkkých tkanív.



Obrázok 3.1: Originálny snímok, mamograf

Obrázok 3.2: Upravený snímok, prevodom na negatív

Ako vidieť, z negatívu snímku je omnoho jednoduchšie analyzovať zachytené mäkké tkanivo a jeho prípadné anomálie.

3.2 Gamma filter

Gamma filtre označujú filtre, ktoré prepočítavajú hodnoty intenzít pixelov podľa vzťahu 3.2,

$$s = c \cdot r^{\gamma} \tag{3.2}$$

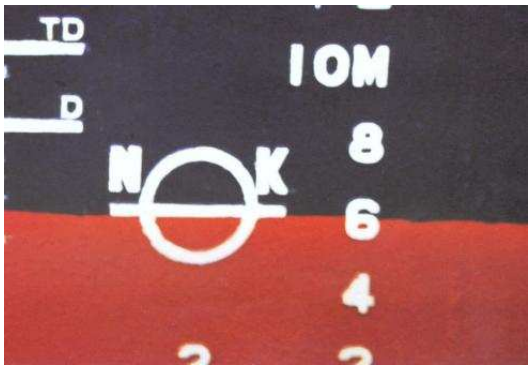
kde c a γ sú kladné konštanty. Využívajú sa pre korekciu obrazu, ktorého skreslenie vzniká pri jeho snímaní. Toto skreslenie sa prejavuje stmavnutím, alebo zosvetlením zosnímaného obrazu oproti snímanej skutočnosti. Pri bližšom pohľade na matematický zápis prepočtu zistíme, že gamma filter v sebe spája logaritmický a exponencionálny filter. To, aké výsledky bude produkovať, ovplyvňuje konštanta γ podľa ktorej je odvodený názov tohto filtra. Pre hodnoty $\gamma \in \langle 0,0;\ 1,0)$ sa bude tento filter chovať ako logaritmický filter t.j. zvýrazní nízke intezity, pridá im na hodnote a minimalizuje rozdiely medzi vysokými intenzitami, navýšením menších z nich. Vo výsledku sa to prejaví zosvetlením obrazu. Pre hodnoty $\gamma \in (1,0;\ +\infty)$ to bude naopak. Rozdiely medzi nízkymi hodnotami sa zmenšia, čím sa celkovo zmenšia všetky nízke hodnoty a zároveň sa zvýraznia hodnoty vyššie, znížením nižších z nich. Pre $\gamma = 1,0$ sa filter neuplatňuje.

Obrázok 3.3: Pôvodný, tmavý obraz $\gamma = 1.0$

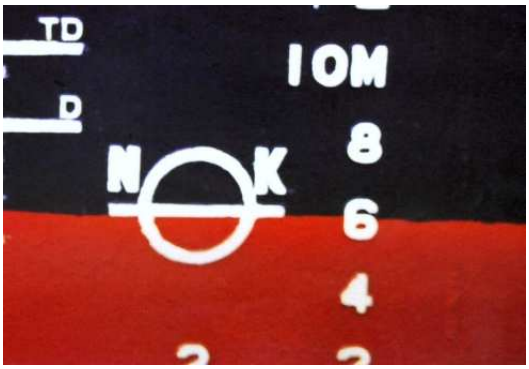
Obrázok 3.4: Obraz po gamma korekcií $\gamma = 0,65$

3.3 Filtre rozťahujúce histogram

Histogram chápeme ako mapovanie početnosti udalostí, hodnôt, alebo pre náš účel podstatných intenzít, ktoré spadajú do jednej z vopred definovaných kategórii, resp. rozsahu. Predpokladajme, že máme napríklad obraz s ôsmymi bitmi na pixel. Celkový počet zobraziteľných farieb je potom 2^8 čo predstavuje celkom 256 farieb. Ďalej majme rovnaký počet kategórii, kde každá kategória označuje práve jednu farbu. Potom pri prechode celým obrazom a počítaním výskytu jednotlivých farieb získame histogram tohto obrazu. Ak tento histogram obsahuje kategórie s nulovým výskytom(žiadna farba tohto typu sa v obraze nenachádza) a prvá nenulová kategória nie je prvou kategóriou histogramu, alebo posledná nenulová kategória nie je poslednou kategóriu histogramu, hovoríme o obraze so zúženým histogramom. Iba v takomto prípade, má zmysel uplatňovať filter, ktorý tento histogram rozťahuje. Toto rozťahovanie pritom prebieha prepočtom jednotlivých farieb každého pixelu na základe hraničných kategórií zaznamenaného histogramu, čo sa v konečnom dôsledku odzrkadlí v kontraste výsledného obrazu.



Obrázok 3.5: Pôvodný obraz so zníženým kontrastom



Obrázok 3.6: Upravený obraz fitrom roztiahnutia histogramu

3.4 Rozmazanie – Blur

Filter rozmazania patrí do kategórie plošných filtrov, ktorý sa tiež označuje ako filter dolnej priepuste. Tento termín je prebraný z oblasti spracovania signálu, kde dolno-priepustný filter slúži pre filtráciu vysokých frekvencii. Obdobne je tomu aj u obrazového spracovania, s tým rozdielom, že to čo v oblasti signálov predstavuje vysoké frekvencie, sa v obraze prejavuje ako šum. Šum je súčasťou každého obrazu a vzniká nedokonalosťou samotného snímania. Môže byť náhodný, periodický, alebo kombináciou oboch súčasne. To, čo sa v oblasti signálov a elektrotechniky javí ako obmedzenie frekvenčného spektra, sa u obrazových informácií javí ako rozmazanie. Dolno-priepustné filtre nachádzajú široké uplatnenie najmä pre svoju schopnosť potláčať obrazový šum i keď za cenu straty detailu. Pri vhodne nastavenej matici okolia, však dokáže kvalitu obrazu výrazne vylepšiť bez pozorovateľnej straty na detailoch. V princípe ide o priemerovanie hodnoty pixelu obrazu s hodnotami pixelov jeho okolia, pričom jednotlivým pixelom tohto okolia je možné priradiť rôznu váhu, resp. určiť v akej miere sa ich hodnota vo výsledku prejaví.

Obrázok 3.7: Pôvodný zašumený obraz

Obrázok 3.8: Po úprave filtrom rozmazania

3.5 Detekcia hrán – Edge detection

Tento druh filtra patrí, rovnako ako filter predošlý, do kategórie plošných filtrov. V počítačovej grafike nachádza uplatnenie predovšetkým pre detekciu objektov, zvýraznenie ich kontúr, alebo pre zjednodušenie obrazu, výsledkom, ktorého je obraz, v ktorom sú len kontúry snímaných objektov. Princíp filtra je nasledujúci: Prechodom obrazu vypočíta pre každý pixel diferencionálny koeficient, ktorý sa rovná aritmetickému priemeru rozdielov intenzít filtrovaného pixelu a pixelov okolia. Následne podľa tohto koeficientu prepočíta intenzitu každej zložky filtrovaného pixelu. Aj pri tomto filtri určuje výsledok veľkosť a váhové koeficienty matice okolia.

Obrázok 3.9: Originálny obraz

Obrázok 3.10: Obraz po detekcii hrán

3.6 Zaostrenie obrazu – Sharpening

Filter zaostrenia je protikladom k filtru rozmazania. Jeho účelom je v rozmazanom, málo detailnom obraze zvýrazniť inak nevýrazné detaily. Pracuje pritom obdobne ako filter detekcie hrán, s tým rozdielom, že diferencionálny koeficient každého filtrovaného pixelu je navýšený o hodnotu 1. Vo výsledku sa toto navýšenie prejaví zvýraznením rozdielov medzi pixelom a jeho okolím, v podobe zníženia, alebo zvýšenia intenzít farebných zložiek pixelu, pričom sa zachovávajú farby súvislých plôch obrazu.

Obrázok 3.11: Originálny obraz

Obrázok 3.12: Po zaostrení

3.7 Ďalšie z bežne používaných filtrov

Okrem už uvedených filtrov, sa za bežne používané filtre považujú filtre: monochromatický filter, filter prevodu do odtieňov sivej a tresholding, tiež známy pod názvom prahovanie. Všetky tieto filtre patria do kategórie filtrov pracujúcich s intenzitou jednotlivých pixelov obrazu. Ich primárnou úlohou je zjednodušenie obrazu znížením počtu jeho farieb.

Obrázok 3.13: Monochromatický filter

Obrázok 3.14: Filter odtieňov sivej

Obrázok 3.15: Treshold filter

Kapitola 4

Nefotorealistické zobrazovanie

Cieľom fotorealistického zobrazovania je podávanie čo najvierohodnejšieho výsledku, s kladením dôrazu na fyzikálne zákony, predovšetkým z oblasti optickej fyziky. V úvahu sa pri zobrazovaní, teda berú aspekty ako lom svetla, jeho odraz, priehľadnosť objektov, pohlcovanie svetla a iné. Výsledkom sú detailne prepracované obrazy resp. scény, ktoré ľudské oko častokrát nedokáže rozlíšiť od skutočných záberov reálneho sveta.

Nefotorealistické zobrazovanie, anglicky Non-photorealistic rendering, skrátene NPR je presným opakom fotorelistického. Snaží sa obraz pretvárať do jeho zjednodušenej podoby. Dôvodom je obyčajne potreba aby obraz poskytoval len podstatné informácie, alebo aby svojim zjavom navodzoval dojem, že ide o ľudskú umeleckú prácu. Častým prípadom využitia tohto zobrazenia je napríklad pre účely priemyslu, kde je zbytočné zobrazovať detaily a vlastnosti predmetov ako je farba, svetelná reflektivita, vrhané tiene. Dôležité sú v tomto prípade len základné obrysy objektov, ktoré sú schopné zachytiť ich tvar.

O účelovosti však už nemožno hovoriť v druhom, už spomínanom využití nefotorealistického zobrazovania. Zatiaľ čo využitie pre priemysel je pomerne jednoducho realizovateľné, pre umelecký smer ide o niečo viac ako len prostú detekciu hrán a prevod do odtieňov sivej farby. Obrazy tvorené ľudskou rukou sú charakteristické svojou náhodnosťou, nejednoznačnosťou a vo všeobecnosti voľnosťou samotnej tvorby. Svoju úlohu tu zohráva náhodnosť, simulácia a rôzne simulačné techniky. V praxi ide o opätovnú snahu dodať zjednodušenému obrazu dávku reálnosti. Opäť sa uplatňujú fyzikálne zákony tentokrát, ale už na úrovni materiálov: drsnosť povrchu, schopnosť podkladu vpiť rôzne tekutiny,... a pôsobenia rôznych síl: vplyv tlaku na množstvo uvolnenej farby zo štetca, vplyv rýchlosti ťahu štetca na hrúbku výslednej čiary a pod. Výsledné aplikovanie týchto metód je tak v mnohých ohľadoch časovo oveľa náročnejšie ako fotorealistické zobrazovanie.

Súčasťou každej umeleckej tvorby sú tri základné nástroje, prostredníctvom ktorých vzniká samotný výsledný obraz. Ide o médium, aplikátor a podklad. Pod jednotlivými pojmami budeme rozumieť nasledujúce:

- Médium – predstavuje nanášanú látku, najčastejšie farbu, grafit, alebo iný prostriedok schopný zanechávať trvalú vizuálnu stopu na podklade.
- Podklad – Podkladom rozumieme plátno, papier, alebo iný objekt, ktorý je nosičom obrazu.
- Aplikátor – predstavuje nástroj, ktorým médium nanášame na podklad.

Kľúčom k úspešnému nefotorealistickému zobrazovaniu je simulácia všetkých troch spomenutých nástrojov [4].

4.1 Real-time processing a Post-processing

Prácu s obrazom delíme podľa doby nad ním vykonávaných operácii na real-time processing a post-processing. Real-time processing vyjadruje prácu s obrazom pri jeho vzniku, teda v dobe snímania objektu, prevodu z vektorového vyjadrenia do rastrového a pod. Ide o náročný proces, kedy sa kladie dôraz na čas potrebný pre spracovanie jedného snímku. Vhodným príkladom môže byť snímanie farebnou kamerou, pri ktorej predpokladajme minimálnu snímkovú frekvenciu 25 snímkov za sekundu, s rozlíšením 800×600 pixelov. Jednoduchou matematikou dospejeme k záveru, že pre real-time processing sa musia všetky obrazové operácie stihnúť vykonať najneskôr za 4 stotiny sekundy, pričom každá takáto operácia zahŕňa prechod celým obrazom, čo odpovedá 1 920 000 operáciám procesora(vrátane prístupu do pamäte). V konečnom dôsledku, teda musíme v rámci real-time processingu vykonávať minimálne 48 miliónov operácii za sekundu. V praxi sa však využívajú zariadenia snímajúce obraz s väčšou snímkovou frekvenciou a pri vyššom rozlíšení, z čoho plynie aj potreba vykonávať väčší počet operácii za sekundu. Z uvedeného príkladu vyplýva, že real-time processing zahŕňa len jednoduché spracovanie obrazu, filtráciu a tak uplatnenie nefotorealistického zobrazovania je značne obmedzené.

Táto práca je však o post-processingu, ktorý zahŕňa prácu so statickým, časovo nemenným obrazom. Odpadá teda časový limit spracovania a otvára sa možnosť využitia zložitejších postupov pre nefotorealistické zobrazovanie.

4.2 Základné simulačné princípy – simulácia náhodnosti

Snahou simulácie je v rámci nefotorealistického zobrazovania priblížiť sa výsledkom čo najviac simulovanej technike kreslenia. Každá kresliaca technika má svoje špecifiká. Olejomaľba je charakteristická zdanlivo náhodnými ťahmi štetca, ktoré svoj význam ukazujú až pri pohľade na obraz ako celok, aquarel je charakteristický ich prekrývaním a splývaním, zatiaľ čo perokresba využíva len jeden typ farby s tieňovaním pomocou nepravidelného šráfovania. Kresliacich metód je veľké množstvo, no ich spoločným znakom je predovšetkým náhodnosť. Každý ručne kreslený obraz bez ohľadu na techniku je unikát, množinou nepresných a rôznymi náhodnými aspektmi ovplyvnených čiar, škvŕn, ťahov štetca, ktoré vo výsledku tvoria výsledný obraz.

Keďže je počítač deterministický stroj a pracuje len s presnými binárnymi číslami, predstavuje náhodnosť pri simuláciách istý problém. Riešením môže byť využitie hardwarového vybavenia: meranie prístupových dôb disku, výstupného napätia zdroja, a pod., prípadne špecializovaných zariadení, ako sú generátory šumu. I keď toto riešenie poskytuje skutočne náhodné hodnoty, ide o veľmi pomalú a z tohto dôvodu nevhodnú metódu pre tento účel. Alternatívnym a často využívaným riešením je použitie algoritmických, tzv. kongruentných generátorov. Tieto sú z hľadiska rýchlosti rádovo rýchlejšie než hardwarové vybavenie, no cenou za ich rýchlosť je strata náhodnosti. Výstupom týchto algoritmov, sú len zdanlivo náhodné hodnoty, ktoré sa s určitou periódou opakujú. Nejde teda o náhodnosť v pravom slova zmysle, ale o tzv. pseudonáhodnosť. Kongruentných generátorov existuje veľké množstvo. Spomedzi všetkých za zmienku stoja:

- Lahmerov lineárny generátor, ktorý generuje hodnoty podľa vzťahu 4.1,

$$x_i = (ax_{i-1} + c) \bmod m \tag{4.1}$$

  kde a je násobiteľ, c je inkrement a m je modulus. Vhodným dosadením jednotlivých

hodnôt je možné dosiahnuť dobré výsledky v podobe generovania čísel s periódou m [3].

- Tausworthov generátor využívajúci bitové operácie vo vzťahu 4.2,

$$a_k = (c_p a_{k-p} + c_{p-1} a_{k-p+1} + c_{p-2} a_{k-p+2} + \cdots + c_1 a_{k-1}) \text{mod} 2 \tag{4.2}$$

  kde všetky premenné nadobúdajú hodnoty v rozsahu 0 až 1. Pri vhodne zvolených parametroch p, c a počiatočnej hodnote(nastavení bitov) je možné dosiahnuť periódu dĺžky až 2^{521} [3].

- Witchmann-Hillov generátor, ktorý kombinuje tri lineárne generátory podľa vzťahov 4.3.

$$x_i = m x_{i-1} \bmod a \tag{4.3a}$$
$$y_i = n y_{i-1} \bmod b \tag{4.3b}$$
$$z_i = o z_{i-1} \bmod c \tag{4.3c}$$

  Výslednú hodnotu potom počítame podľa vzťahu 4.4.

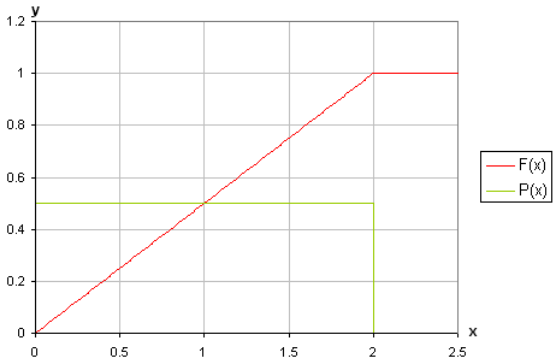
$$u_i = \left[\frac{x_i}{a} + \frac{y_i}{b} + \frac{z_i}{c} \right] \bmod 1 \tag{4.4}$$

  Po dosadení: $a = 30269$, $b = 30307$, $c = 30323$, $m = 171$, $n = 172$ a $o = 170$, má výsledný generátor dĺžku periódy cca. 2^{42} [3].

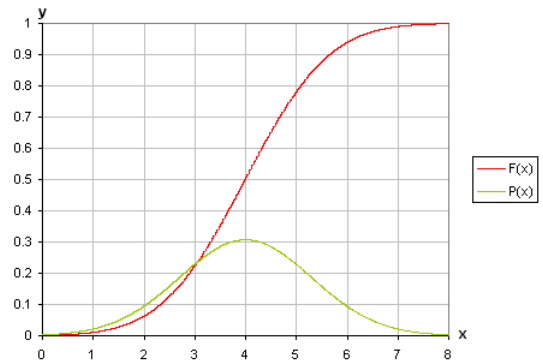
Všetky vyššie zmienené generátory pseudonáhodných čísel generujú hodnoty s rovnomerným pravdepodobnostným rozložením (obr.4.1). Pre praktické použitie sú však nutné ďalšie rozloženia, menovite najmä normálne, Gaussovo, (obr.4.2) a exponencionálne rozloženie(obr.4.3). Tie je možné získať z rovnomerného rozloženia jednou z nasledujúcich metód transformácie:

- Vylučovaním – generovanie hodnôt pre osi x a y a následné porovnávanie či takto vygenerovaný bod spadá do plochy vymedzenej osou x a funkciou hustoty $P(x)$ cieľového rozloženia. Časovo ide o pomerne náročnú metódu, ktorej náročnosť rastie s maximom funkcie hustoty a pri neobmedzených rozloženiach, ju nemožno použiť vôbec.

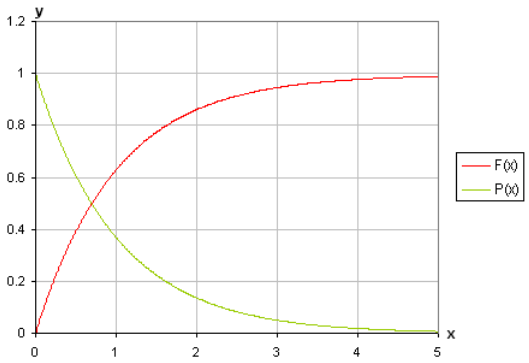
- Inverznou funkciou – generované hodnoty s rovnomerným rozložením sa prepočítavajú podľa inverznej distribučnej funkcie $F(x)^{-1}$ želaného rozloženia. Časovo nenáročná metóda, ktorú však nemožno uplatniť pri niektorých rozloženiach z dôvodu netriviálnej inverznej funkcie (napr. Gaussove rozloženie). V takomto prípade možno uplatniť metódu vylučovania za predpokladu splnenia podmienok jej uplatnenia(viď. Metóda transformácie vylučovaním)



Obrázok 4.1: Rovnomerné rozloženie



Obrázok 4.2: Gaussovo rozložnie



Obrázok 4.3: Exponencionálne rozloženie

Kapitola 5

Návrh

Cieľom tohto projektu bolo navrhnúť a vytvoriť systém, ktorý by umožňoval nefotorealistické zobrazovanie s dôrazom na flexibilitu a jednoduchosť jeho použitia. Návrh teda definuje nasledujúce problémy, ktorých implementačné riešenie nasleduje v ďalšej kapitole:

1. Samotný systém a jeho rozdelenie z hľadiska funkčných častí
2. Popis činnosti týchto častí, najmä:
   a. interpretera,
   b. filtrov realizujúcich nefotorealistické zobrazovanie a
   c. simulačné metódy, konkrétne simulácia náhodnosti.

5.1 Návrh systému

Celá aplikácia sa delí na tri časti:

- interpreter – hlavná časť aplikácie,
- filtre – množina operácií nad obrázkami(filtre), zahŕňajúc matematické funkcie a kongruentné generátory,
- prehliadače obrázkov – celkovo dva programy, jeden pre architektúru *Linux* a druhý *WinAPI*, využívajúce pre svoju činnosť *OpenGL* a *DevIL* knižnice.

Písaná je v jazyku *C* z dôvodu zachovania čo najvyššej rýchlosti a zároveň spravovateľnosti kódu. Priložený zdrojový kód je preložiteľný na väčšine operačných systémov typu *Linux* a *Windows*. Pre *Linux* je pripravený makefile, pre *Windows* projektový súbor programu *IDE Code:Blocks*, ktorý je freewarom a je dostupný na stránke: http://www.codeblocks.org.

Pre kódovanie a dekódovanie jednotlivých obrazových formátov je využitá už zmienená knižnica *DevIL*, ktorá je časťou balíka *OpenCV*. Ide o open-source s možnosťou bezplatného stiahnutia z tejto stránky: http://openil.sourceforge.net.
Zobrazovanie zabezpečuje knižnica *OpenGL*, ktorá je súčasťou základnej softwarovej výbavy každej počítačovej zostavy.

5.2 Interpreter

Interpreter, ako hlavná časť aplikácie prekladá a zároveň vykonáva parametrom predané skripty. Tie sú v textovom formáte a pre ich tvorbu je možné použiť ľubovoľný textový editor. Použitá syntax je case-sensitive a svojou podobou sa blíži syntaxi jazyka *C*. Implementuje jej niektoré syntaktické formy v podobe:

- riadkových a blokových poznámok,
- zápisu priradenia hodnoty do premennej,
- využívania aritmeticko-logických operátorov a
- zápisu volania filtrov v tvare: `filter(parameter1, parameter2, ...);`.

Rozdielom oproti tejto syntaxi je využitie zápisu volania funkcie aj pre definície a tvorbu premnných niektorých dátových typov, rovnako ako aj obohatenie aritmeticko-logických operátorov o operátory rotácie.

Aplikácia podporuje celkom päť dátových typov, spolu s rozšírením o preddefinované konštanty. Sú to dátové typy: celé číslo, desatinné číslo, textový reťazec, matica a obraz. Zmienené preddefinované konštanty slúžia najmä pre určenie matíc okolia a spôsob načítania obrazu. Každý uvedený dátový typ má svoju množinu povolených operácií, ktorých tabuľka nasleduje. Vzhľadom na to, že konštanty nepodporujú žiadnu operáciu, nie je tento „dátový typ“ v tabuľke uvedený:

Dátový typ	Operátor															
	+	-	*	/	%	&	|	^	!	--	++	>>	<<	>@	<@	Filtrácia
Integer	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	×
Float	√	√	√	√	√	√	√	√	√	√	√	√	√	×	×	×
String	√	√	×	×	×	×	×	×	√	√	√	√	√	√	√	×
Matrix	√	√	√	√	√	√	√	√	√	√	√	√	√	×	×	×
Image	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

Tabuľka 5.1: Prehľad povolených operátorov jednotivých dátových typov

Implementovaný jazyk je netypový a teda dátový typ sa pri deklaráciach a definíciach premenných neuvádza. Pri operáciách s premennými rôzneho dátového typu dochádza k pretypovaniu podľa nasledujúcich pravdiel:

1. Pri operácii zahŕňajúcej celé číslo a číslo s plávajúcou desatinnou čiarkou, bude celé číslo pretypované na číslo s plávajúcou desatinnou čiarkou.
2. Pri operácii čísla, či už celého, alebo destinného a textového reťazca bude toto číslo prevedené na jeho textovú podobu.
3. S maticami a obrazmi je možné vykonávať niektoré aritmeticko-logické operácie a to len v prípade, že ich operandom je číslo.

Súčasťou návrhu je aj pomerne rozsiahla množina funkcií implementujúcich abstraktné dátové typy. Ide predovšetkým o obojsmerne viazané zoznamy s rozšírením o ukazateľ na aktuálny prvok, hashovaciu tabuľku spolu s hashovacou funkciou, LIFO fronty a textové reťazce s programovo neobmedzenou dĺžkou s implementovanými matematickými funkciami

ako sú rotácie, inkrementácia, dekrementácia, sčítanie, odčítanie a ďalšie. Každá skupina funkcií daného abstraktného dátového typu je pritom vytvorená ako samostatný zdrojový súbor pre zaručenie znovupoužiteľnosti a prehľadnosti výsledného programu.

5.3 Návrh filtrov

Navrhnutá aplikácia podporuje množinu základných filtrov, ktoré sa snaží optimalizovať a predstavujú ju filtre: negatív, grayscale, monochroma, treshold, gamma filter, filter rozmazávania, filter zaostrovania a filter detekcie hrán. Okrem nej potom implementuje niektoré špecifickejšie filtre, menovite: sépiu, pixelizáciu, medián spolu s jeho modifikovanou verziou, filter explózia, filter textúry kameňa a olejomaľbu. Z hľadiska parametrov podporuje nastavenie veľkého množstva číslených parametrov filtrácie, rôzne preddefinované konštanty a u niektorých filtroch možnosť uplatnenia užívateľom zadanej matice okolia. Podrobný popis jednotlivých filtrov, ktoré neboli spomenuté v teoretickej časti je uvedený v kapitole implentácia, podkapitola filtre.

5.4 Simulácia náhodnosti

V rámci filtrácie je súčasťou filtrov aj množina kongruentných generátorov, ktorých primárnou úlohou je poskytovať pseudo-náhodne genrované čísla v rozsahu $\langle 0,0;1,0\rangle$ a náhodne generovaný index s predaným rozsahom. Ďalším rozšírením tejto časti je implementácia prevodu základného prevdepodobnostného rozloženia, ktoré generuje pseudo-náhodný generátor, na exponencionálne a kvázi normálne rozloženie.

Kapitola 6

Implementácia

6.1 Lexikálny analyzátor

Lexikálny analyzátor pracuje zaužívaným spôsobom stavového automatu. Oproti tomuto riešeniu je však rozšírený o slovník príkazov, ktorý vstupuje do funkčnosti v momente spracovávania názvov premenných. Ide o pole polí štruktúr, ktoré definujú textový zápis príkazu a jemu odpovedajúci lexém. Každé takéto pole definuje skupinu príkazov, začínajúcich rovnakým znakom a je ukončené zarážkou v podobe nulového ukazateľa na textový reťazec - `NULL`. Prehľadávanie tohto poľa polí sa teda obmedzuje podľa zaznamenaného prvého znaku, čím sa celé prehľadávanie značne urýchľuje. Toto riešenie umožňuje jednoduché pridávanie a mazanie príkazov, bez nutnosti zasahovania do kódu samotného lexikálneho analyzátora. V dobe písania tohto textu boli definované nasledujúce textové reťazce, ktorých význam je predmetom nasledujúcich kapitol: `ADD`, `ADDITION`, `AND`, `BLUR`, `CUSTOM`, `CLIP`, `DEC`, `DECREMENT`, `DIV`, `DIVISION`, `DISPLAY`, `DEFAULT`, `DEF`, `DEFINE`, `EDGE`, `EDGE_DETECTION`, `EXPLO`, `EXPLOSION`, `EMOOR`, `EXT_MOOR`, `EXTENDED_MOOR`, `EXIT`, `GAM`, `GAMMA`, `GREY`, `GREYSCALE`, `GRAY`, `GRAYSCALE`, `HRZ`, `HORIZONTAL`, `INC`, `INCREMENT`, `IN`, `INPUT`, `LRT`, `LROTATE`, `LSH`, `LSHIFT`, `MUL`, `MULTIPLY`, `MOD`, `MODULO`, `MED`, `MEDIAN`, `MTX`, `MATRIX`, `MONO`, `MONOCHROMA`, `MOOR`, `NOT`, `NEGATION`, `NEG`, `NEGATIVE`, `NEUMANN`, `OR`, `OIL`, `OILPAINT`, `OUT`, `OUTPUT`, `PIX`, `PIXELIZE`, `PMODE`, `PROCESSING_MODE`, `QUIT`, `RRT`, `RROTATE`, `RSH`, `RSHIFT`, `REPEAT`, `SUB`, `SUBTRACT`, `SEPIA`, `SMEDIAN`, `SQUARE_MEDIAN`, `SHARP`, `SHARPEN`, `SRC`, `SOURCE`, `STRETCH`, `STRETCH_HISTO`, `STRETCH_HISTOGRAM`, `TRESH`, `TRESHOLD`, `TEX_STONE`, `TEXTURE_STONE`, `VRT`, `VERTICAL` a `XOR`.

Čo sa povolených lexémov a ich tvaru týka, podporované sú okrem príkazov, premenných a číselných konštánt:

- blokové a riadkové poznámky obdobne ako je tomu v jazyku $C/C++$:
  riadkové poznámky „//“,
  blokové poznámky: „/*“ začiatok poznámky, „*/“ jej koniec,

- základné matematické operátory:
  sčítanie „+”,
  odčítanie „−“,
  násobenie „*“,
  delenie „/”,
  modulo „%”,
  operácia priradenia „=”,

- rozšírené matematické operátory:
  inkrementácia „++”,
  dekrementácia „--“,

- zátvorky:
  okrúhle zátvorky „(”, „)”,
  hranaté zátvorky „[“, „]”,
  množinové zátvorky „{”, „}”,

- odvodené matematické operátory:
  priradenie súčtu pravej a ľavej strany „+=”,
  priradenie rozdielu pravej a ľavej strany „-=”,
  priradenie násobku pravej a ľavej strany „*=”,
  priradenie podielu pravej a ľavej strany „/=”,
  priradenie modula pravej a ľavej strany „%=”,

- základné bitové operátory:
  logický súčin „&”,
  logický súčet „|”,
  výlučný logický súčet „^”,
  negácia „!”,

- operátory bitových posunov a rotácii:
  bitový posun vpravo „>>”,
  bitový posun vľavo „<<”,
  bitová rotácia vpravo „>@”,
  bitová rotácia vľavo „<@”,

- odvodené bitové operátory:
  priradenie logického súčinu pravej a ľavej strany „&=”,
  priradenie logického súčtu pravej a ľavej strany „|=”,
  priradenie výlučného logického súčtu pravej a ľavej strany „^=”,
  priradenie negácie pravej strany „!=”,

- odvodené operátory bitových posunov a rotácii:
  priradenie bitového posunu vpravo, ľavej strany o počet bitov určených pravou stranou, „>>=”,
  priradenie bitového posunu vľavo, ľavej strany o počet bitov určených pravou stranou „<<=”,
  priradenie bitovej rotácie vpravo, ľavej strany o počet bitov určených pravou stranou „>@=” a konečne
  priradenie bitovej rotácie vľavo, ľavej strany o počet bitov určených pravou stranou „<@=”.

Dátové typy majú nasledujúcu formu zápisu:

- celočíselné čísla – môžu byť zapísané okrem decimálnej podoby, t.j. číselnými znakmi „0” až „9” aj v binárnej, osmičkovej, alebo hexadecimálnej podobe. Posledné tri zmienené formy zápisu majú každá svoj špecifický prefix. Pre binárny zápis to je „0b“, pre osmičkový „0o“ a pre hexadecimálny „0x“. Po tomto prefixe nasleduje samotné číselné vyjadrenie v rozsahu povolených znakov pre danú sústavu.

- Čísla s plávajúcou desatinnou čiarkou – je možné zadávať buď zaužívanou formou, kedy je celá časť čísla oddelená od desatinnej bodkou, alebo v exponencionálnom tvare. V rámci zaužívaného spôsobu je možné použiť skrátenú formu zápisu, kedy sa pre čísla menšie ako $1,0$ nepíše pred desatinnou bodkou znak „0“. Exponencionálny tvar je možné zadávať v jednom z následujúcich formátov:

  `1e23` `0.23e4`, `0.23e-4`, `0.23e+4`, `0.e4`, `.e4`,

  výsledná hodnota v posledných dvoch príkladoch je vždy nulová.

- Textové reťazce – vymedzujú symboly dvojitých úvodzoviek „`"`“, podobne ako u iných programovacích jazykoch. Odlišnosťou však je vylúčenie escape sekvencií ako sú napríklad tabulátory, ukončenie riadku, vertikálne tabulátory a pod. Dôvodom je najmä obmedzené využitie tohto dátového typu. Textové reťazce sa v tejto práci využívajú len pre selekciu vstupného a výstupného názvu súboru v ktorých sa uvedené znaky vyskytovať nemôžu.

- Matice – musia byť vždy štvorcového tvaru a v nasledujúcom formáte zápisu:

```
size  = 3;
   a  = MATRIX(size,  1, 2, 3
                      4, 5, 6
                      7, 8, 9);
```

- Obraz – je jediným dátovým typom, ktorý nemá svoj zápis. Jediným spôsobom definície premennej tohto typu je s využitím príkazu `INPUT`, alebo jeho skrátenou alternetívou `IN`.

6.2 Preinterpreter

Preinterpreter by sme mohli v tejto práci chápať tiež ako optimalizátor, i keď optimalizácia nie je jeho primárnym účelom. V rámci optimalizácie ide o optimalizáciu parametrom predaného skriptu, ktorého zdrojový kód sa optimalizuje v zmysle odstránenia redundantných operátorov a unifikáciu celočíselných čísel. Redundantnými operátormi sú najmä: násobné unárne znamienka, implicitne určené kladné polarity čísel, násobné negácie, inkrementácie spojené s dekrementáciami a ich násobné varianty. Unifikáciou celočíselných čísel zas rozumieme prevod všetkých binárne, osmičkovo a hexadecimálne vyjadrených čísel do ich desiatkového ekvivalentu. Ohľadom primárneho účelu preinterpretera, ide hlavne o spracovávanie užívateľom definovaných úsekov kódu v podobe definícií. Toto spracovanie zahŕňa ich syntaktickú analýzu, pridávanie ich symbolického pomenovania do hashovacej tabuľky spolu s ich telom a následné nahradenie všetkých výskytov tohto pomenovania telom príslušnej definície v zdrojovom kóde skriptu. Výsledný optimalizovaný zdrojový súbor je potom uložený ako dočasný súbor s názvom „tmp.tmp“, ktorý už definície neobsahuje. Tento súbor je naviac obohatený o špeciálne lexémy, ktoré nie sú užívateľovi za normálnych okolnosti dostupné. Ich hlavnou funkciou je identifikácia čiastočne interpretovaných príkazov(máme na mysli optimalizované násobné inkrementácie a dekrementácie) a čísel riadku pre neskoršie správne detekovanie miesta chyby v zdrojovom kóde.

6.3 Syntaktický analyzátor

Syntaktický analyzátor zabezpečuje syntaktickú analýzu, ktorá zahrňuje kontrolu poradia a počtu lexémov pre jednotlivé operácie a funkcie resp. filtre. Jeho úlohou je vytvoriť z lexikálnym analyzátorom overených lexémov, predaných ako optimalizovaný súbor, výstup preinterpretera, jednoducho spracovateľné matematické rovnice. Pre tento účel sa ako najvhodnejšou variantou ukázalo byť použitie zápisu týchto rovníc v poľskej notácii. Táto je vhodná predovšetkým kvôli svojej schopnosti určovať prioritu operácii ich poradím rovnako ako aj tvarom výsledného zápisu, ktorý je analógiou k syntaxi filtrov. Tým je možné ich spracovávanie v rámci jedinej funkcie unifikovať. Pri potrebe zmeny priorít, sú k dispozícií zátvorky, ktorých použitie je obmedzené len z hľadiska ich párovania. To je testované za pomoci jednej LIFO fronty. Priorita jednotlivých typov zátvoriek je pritom rovnaká a vo výslednej rovnici sú tak všetky nahradené jediným typom zátvoriek, ktoré sú po spracovaní celej rovnice z tejto rovnice nakoniec odstránené (vyplýva z faktu, že zmena priority sa prejavuje len zmenou poradia v rámci rovnice). Ohľadne tvaru vygenerovaných rovníc, ide o obojsmerne viazané zoznamy, kde každú rovnicu tvoria hneď dva. Prvý je pre pravú stranu rovnice, kde je prípustný len jeden z nasledujúcich lexémov:

- premenná – výsledkom je potom rovnica priradenia
- `OUT`, alebo `OUTPUT`, čiže lexémy výstupu – uloženie výsledku do určeného súboru. Ak užívateľ súbor neurčil, uloží sa výsledok do implicitne definovaného súboru s názvom „OutputImage.bmp“.
- `DISPLAY`, lexém zobrazenia – zobrazenie výsledku pravej strany rovnice v konzolovom okne, prípadne ak je výsledkom obraz v samostatnom okne prehliadača.
- `PMODE`, alebo `PROCESSING_MODE` – zmena prístupu apikácie pri kombinovaní rozmerovo rozličných obrazov.
- `EXIT`, alebo jeho synonymum `QUIT` – predčasné ukončenie interpretácie skriptu.

Pravú stranu potom tvoria ostatné podporované lexémy, z čoho je zrejmé, že násobné priradzovanie, zobrazovanie, výstup a ich vzájomné kombinácie v rámci jednej rovnice nie sú prípustné. Základný tvar každej spracovanej operácie a filtra je nasledovný:

$$\texttt{Op}(\texttt{Param}_1, \texttt{Param}_2, \ldots \texttt{Param}_n),$$

kde `Op` predstavuje operátor a $\texttt{Param}_1$ až $\texttt{Param}_n$ predané operandy(pre filtre parametre). Tieto operandy pritom môžu predstavovať ďalšie, zanorené operácie. Je nutné čitateľovi pripomenúť, že počet operandov sa pre jednotlivé operátory líši. Prakticky ich z hľadiska počtu operandov delíme na tri základné skupiny:

1. operátory unárne, teda len s jedným povoleným operandom,
2. operátory binárne, s operandami minimálne dvoma, bez horného obmedzenia ich celkového počtu a
3. špecializované operátory v podobe filtrov, u ktorých sa počet parametrov líši v závislosti na konkrétnom filtri.

Ukončenie rovnice predstavuje lexém bodkočiarky „`;`“, po detekcii, ktorého sa vytvorená rovnica finálne upravuje. Táto úprava zahŕňa pre ľavú stranu rovnice: uzavretie operátorov doplnením patričného počtu zátvoriek na jej koniec, náhradu všetkých symbolicky vyjadrených operácií ich príkazovou formou a spracovanie unárnych operátorov, ktoré boli až do tohto kroku považované za súčasť operandov. Pre pravú stranu finalizácia znamená spracovanie odvodených matematických operátorov doplnením príslušnej operácie a operandu na začiatok ľavej strany výslednej rovnice.

6.4 Interpreter

Po spracovaní rovnice zo zdrojového skriptu syntaktickým analyzátorom je volaný interpreter, ktorý má za úlohu realizovať touto rovnicou zapísanú postupnosť operácií. Proces spracovávania zahŕňa ukladanie užívateľských premenných do hashovacej tabuľky a súčasne do LIFO fronty, pri ktorom prebieha kontrola ich existenice. Zároveň tento proces obsahuje tvorbu obojsmerne viazaného zoznamu operátorov, v ktorom sa na tieto LIFO fronty odkazuje. Samotný proces spracovania začína pri detekcii pravej okrúhlej zátvorky „`)`“, ktorá označuje koniec práve spracovávaného operátora. Po jej detekcii sa vyprázdňuje fronta parametrov počas ktorého sa tento operátor uplatňuje(vykonáva sa užívateľom zadaná operácia). V dobe vyprázdňovania fronty tiež prebieha typová kontrola a ak je to potrebné a u daného parametra možné, prebieha v tomto kroku aj jeho typová konverzia. Možno atypickým rozšírením je úprava parametrom predanej hodnoty, ktorá sa uplatňuje pri opácií typu filter a ktorá presahuje jej povolený rozsah. Táto úprava zahŕňa obmedzenie tejto hodnoty jej prevodom na povolenú hraničnú hodnotu. Pre parametre vyjadrujúce intenzitu farby ide o rozsah dátového typu `char`, ktorý je závislý na aktuálnej architektúre, na ktorej program beží a pre percentuálne vyjadrenie ide o rozsah $0,0 \div 100,0$ percent. V oboch prípadoch je na tento prevod užívateľ patrične upozornený. Výsledok operácie je potom uložený ako parameter nadradeného operátora.

Ako už bolo v kapitole návrhu zmienené, podporovaných je celkom päť dátových typov, pričom každý typ má obmedzený repertoár povolených operácii. Pre dátový typ celočíselné číslo sú prípustné všetky aritmetické operácie, bitové operácie, bitové posuny a rotácie. Obdobne je tomu u čísel s plávajúcou desatinnou čiarkou. Rozdiel je len v podpore bitových operácii, menovite bitové rotácie. Za zmienku stojí implementácia operácie modula a podporovaných bitových operácii. Modulo sa počíta ako rozdiel čísla z ktorého modulo počítame a súčinu celočíselného podielu tohto čísla s deliteľom vynásobeného jeho hodnotou. Symbolicky môžeme tento výrok zapísať ako 6.1.

$$a \bmod b = a - \ \mathrm{round}\left(\frac{a}{b}\right) \cdot b. \tag{6.1}$$

Pre bitové operácie nad číslami s pohyblivou desatinnou čiarkou je využitý prevod tohto čísla na pole celočíselných čísel za pomoci variantných záznamov, tiež známych pod pomenovaním únie. V ďalšom kroku sa potom k takto upravenej hodnote pristupuje ako pri celočíselných číslach. Tretím dátovým typom sú textové reťazce. Ide o najviac obmedzený dátový typ z hľadiska dostupných operátorov. Zo všetkých implementovaných sú u textových reťazcov podporované nasledujúce:

- sčítanie – pre textové reťazce predstuje konkatenáciu,
- odčítanie – odstránenie prvého výskytu odčítaného reťazca,

- inkrementácia – navýšenie čísleného postfixu o jednotku, alebo jeho vytvorenie,
- dekrementácia – odstránenie jedného znaku z konca dekrementovaného reťazca
- rotácia – rotácia sa uplatňuje po znakoch, nie po bitoch ako je tomu u číslených dátových typoch
- posun – pri posune doľava ide o odstraňovanie zadaného počtu znakov zo začiatku textového reťazca, pri posune vpravo sa pred reťazec dopĺňajú prázdne znaky, medzery " ".
- negácia – zmena každého znaku systémom negácie poradového čísla v abecede.

Posledné dva dátové typy, ktorými sú matice a obrazy, sú z hľadiska povolených bitových a metematických operácií takmer identické. Jediným rozdielom je podpora rozdielných rozmerov u obrazových operandoch. Špeciálnym typom operácií sú filtre, ktoré vyžadujú zmiešané operandy, čo sa ich dátového typu týka a pojednávané sú v nasledujúcej kapitole.

6.5 Filtre

Základný prehľad najčastejšie používaných filtrov je popísaný v teoretickej časti v kapitole o filtrácii obrazu. Okrem týchto filtrov je v rámci tejto práce implementovaných ďalších 8 filtrov. Celkový počet implementovaných filtrov je teda 17. Niektoré z týchto filtrov možno rozdeliť podľa klasických kritérií, iné nie. Dôvodom náročnosti ich zaradenia je fakt, že pracujú s náhodnosťou v rámci okolia resp. u nich nemožno o okolí hovoriť vôbec. Nasledujúca kapitola by mala slúžiť ako prehľad týchto filtrov spolu s ich popisom činnosti a možným využitím.

6.5.1 Matematický filter

Matematický filter by podľa konvenčného rozdelenia spadal pod filtre operujúce s jednotlivými pixelmi. Ako už jeho názov naznačuje, spája v sebe základné aritmeticko-logické operácie jednotlivých pixelov obrazu a predanej konštanty. Výsledok aplikácie tohto filtra je závislý na použitej operácii. Pre sčítanie a odčítanie funguje ako jasový filter, pre logický súčin zase ako bit-plane filter. Jeho flexibilita je v porovnaní s ostatnými filtrami neporovnateľne väčšia, čo v praktickom dôsledku znamená, že sme schopní s jeho pomocou vytvoriť ekvivalent ľubovoľného základného filtra. Ako praktický príklad predpokladajme nasledujúci krátky skript v tvare:

```
a = INPUT("Image.jpg");
a =  (a & 0xC0) |
    ((a & 0xC0) >> 2) |
    ((a & 0xC0) >> 4) |
    ((a & 0xC0) >> 6);
```

Po jeho interpretácii zistíme, že svojou činnosťou funguje ako filter znižujúci počet farieb obrazu a to z pôvodných 2^{24} na 2^6. Ďalším rozšírením flexibility tohto už aj tak dosť univerzálneho filtra je jeho schopnosť spracovávať obraz nielen na základe predanej konštanty, ale aj podľa sekundárneho obrazu. Vo výsledku sa toto rozšírenie prejavuje schopnosťou filtra obrazy spájať, vzájomne ich maskovať, odčítať, a vykonávať množstvo

ďalších užitočných efektov. Problémy spojené s rozdielnymi rozmermi oboch obrazov sú riešené úpravou sekundárneho obrazu. Spôsob tejto modifikácie je nastaviteľný funkciou `PROCESSING_MODE`, alebo jeho skrátenou podobou `P_MODE`, ktorý v súčasnosti podporuje tri módy. Prvým je mód `STRETCH`, mód roztiahnutia, kedy sa rozmer sekundárneho obrazu prispôsobí, roztiahne do odpovedajúcich rozmerov primárneho obrazu. Druhým módom je `CLIP` mód. V tomto móde dochádza k orezaniu sekundárneho obrazu podľa rozmerov primárneho. Posledným módom spracovania obrazu je mód `REPEAT`, kedy sa obraz kopíruje v x-ovej a y-ovej osi dovtedy, kým nevyplní celú plochu ohraničenú rozmermi primárneho obrazu. Za predpokladu, že je sekundárny obraz väčší ako primárny, dôjde k jeho orezaniu.

Obrázok 6.1: Originálny obraz

Obrázok 6.2: Po interpretácií skriptu

6.5.2 Sépia

Filter sépia je ďalším filtrom z množiny filtrov pracujúcich s jednotlivými pixelmi obrazu. Jeho účel je z estetického hľadiska navodiť dojem starej, zažltnutej fotografie a z hľadiska praktického znížiť celkový počet farieb obrazu. Implementačne ide o modifikovaný grayscale filter. Touto modifikáciou je dodatočný prepočet na určenú farbu z celkovej intenzity farby pixela. Rozšírením oproti iným systémom je meniteľnosť odtieňa výsledného obrazu bez akýchkoľvek obmedzení. Výsledkom je filter, ktorý okrem sépie dokáže plniť úlohu ľubovoľného farebného filtra (nočný filter, filter západ slnka, ...).

Obrázok 6.3: Originálny obraz

Obrázok 6.4: S filtrom Sépia

6.5.3 Pixelizácia

Pixelizácia je typickým predstaviteľom filtrácie jednotlivých pixelov. Zvláštnosťou oproti iným filtrom tohto typu je prechod filtrovaným obrazom. Zatiaľ čo bežné filtre prechádzajú obraz postupne, t.j. po riadkoch pixelizácie využíva prechod obrazom po segmentoch. Segmentami pritom rozumieme štvorcovú oblasť, submaticu obrazu. Princíp filtrácie zahŕňa výpočet priemernej hodnoty jednotlivých zložiek farebného modelu v rámci segmentu a následnú zmenu zložiek farby každého z pixelov v tomto segmente na vypočítanú priemernú hodnotu. Využiteľnosť tohto filtra je najmä v zjednodušovaní obrazu znížením jeho detailov. V praxi sa však používa aj pre dobre známe cenzúrovanie, ktoré často vidieť v rôznych televíznych reláciách.

Obrázok 6.5: Originálny obraz

Obrázok 6.6: Po pixelizácií

6.5.4 Medián

Medián filter by sme mohli nazvať tiež štatistickým filtrom, z dôvodu využitia štatistických postupov pri rozhodovaní o výpočte hodnôt intenzity farieb filtrovaného pixelu. Ako už z názvu vyplýva, touto štatistickou funkciou je funkcia medián, ktorá za priemernú hodnotu považuje prostrednú hodnotu v vzostupne, alebo zostupne usporiadanom zozname všetkých hodnôt okolia pixelu. Využíva sa rovnako ako filter rozmazávania, teda pre elimináciu, resp. potlačenie nežiadúceho šumu v obraze. Narozdiel od dolno-priepustnej filtrácie však mediánový filter zachováva ostré ohraničenie súvislých plôch obrazu. Vylepšením oproti klasickej metóde je využitie histogramu namiesto usporiadaného zoznamu. Stredná hodnota teda nie je prostrednou hodnotou usporiadaného zoznamu, ale hodnotou s najväčším zastúpením, ktorú možno určiť už v priebehu prechodu obrazom. Odpadá tým radiaci algoritmus, čo sa v konečnom dôsledku pomerne výrazne odzrkadľuje v rýchlosti filtra.

Obrázok 6.7: Originálny obraz

Obrázok 6.8: Prefiltrovaný Medián filtrom

6.5.5 Modifikovaný medián – S medián

Principiálne ide o identický filter s klasickým mediánovým filtrom, no narozdiel od neho nevyužíva pre výpočet mediánovej hodnoty štvorcovú maticu, ale len jej frakciu. Tá je tvorená len jedným riadkom a stĺpcom, presnejšie riadkom a stĺpcom, ktorých stred je práve filtrovaný pixel. Výhodou tejto implementácie je značné urýchlenie celého výpočtu, čo plynie z charakteru závislosti náročnosti výpočtu od veľkosti matice okolia: pre klasickú metódu mediánového filtra: $x = a^2$, modifikovaná verzia: $x = 2 \cdot a - 1$. Ako vidieť ide o porovnávanie exponencionálnej závislosti s lineárnou, z čoho jasne plynie efektivita optimalizácie. Ďalší rozdiel oproti klasickému mediánovému filtru sa prejavuje pri opakovanom aplikovaní filtra, alebo zvýšených hodnotách jeho parametrov, v podobe horizontálnych a vertikálnych čiar, ktoré nápadne pripomínajú ťahy štetcom. Ide o zámerný efekt, ktorý v oblasti nefotorealistického zobrazovania určite má svoje uplatnenie.

Obrázok 6.9: Originálny obraz

Obrázok 6.10: Filtrácia S Medián filtrom

6.5.6 Explózia

Tento filter nespadá do žiadnej z bežne používaných kategórii. Dôvodom je princíp na akom pracuje, ktorý popisujeme ako chaotický výber pixelov v obraze a ich zámenu s náhodne vybraním pixelom z ich okolia. Implementácia tohto filtra zahŕňa možnosť určiť rozmery tohto okolia rovnako ako aj mieru vplyvu náhodne vybraného pixelu na pôvodný pixel.

Uplatnenie tento filter nachádza všade tam, kde je potrebné rozrušiť kontúry objektov vo filtrovanom obraze.

Obrázok 6.11: Originálny obraz

Obrázok 6.12: Obraz po aplikácií explózie

6.5.7 Textúra kameňa

Filter textúry kameňa, resp. kamenného povrchu zaisťuje simuláciu podkladu, jedného zo základných nástrojov simulovaných v nefotorealistickom zobrazovaní. Pracuje spôsobom náhodne tvorených vertikálnych čiar, či skôr skupín bodov, systémom zvyšovania, alebo znižovania jasu filtrovaných pixelov. Parametre tohto filtra určujú úroveň zmeny jasu, rovnako ako aj granularitu, pre tento filter chápanú ako rozptyl bodov výslednej textúry. Filter možno zaradiť do skupiny filtrov filtrujúcich jednotlivé pixely obrazu.

Obrázok 6.13: Originálny obraz

Obrázok 6.14: S pridanou textúrou kameňa

6.5.8 Olejomaľba

Posledným zmieneným filtrom je filter olejomaľby. Ide o pomerne komplexný filter imitujúci kreslenie olejovými farbami technikou pointilizmu(bodovaním). Využíva pritom indexový kongruentý generátor a generátor pseudo-normálneho rozloženia, ktorým sa bližšie venuje nasledujúca podkapitola. Ohľadne parametrov má tento filter len jeden, ktorý určuje veľkosť použitého štetca. Účel filtra je zrejmý – simulovať kresliace nástroje médium a aplikátor.

Obrázok 6.15: Originálny obraz

Obrázok 6.16: V podobe olejomaľby

6.5.9 Kongruentné generátory

V rámci filtrov sú implementované celkom dva kongruentné generátory. Prvý je už v kapitole 4.2 spomínaný lineárny, ktorého premenné sú nastavené na konštantné hodnoty: $c = 1$ a $a = 69069$. Premenná m vyplýva z architektúry na ktorej aplikácia práve beží a odvodzuje sa podľa vzťahu 6.2,

$$m = 2^n - 1 \tag{6.2}$$

kde n je počet bitov dátového typu `integer`.

Druhým generátorom je generátor indexov. Ide o upravenú verziu lineárneho kongruentného generátora, ktorý generuje čísla v zadanom rozsahu, pričom každé číslo sa v rámci vygenerovanej postupnosti objavuje práve raz. Pre dosiahnutie tohto správania je použité bitové pole, v ktorom každý bit odpovedá jednému indexu. Vo výsledku teda nejde o pamäťovo veľmi náročné riešenie. Pri vygenerovaní už generovaného indexu sa generuje index nový dovtedy, kým sa nenájde doposiaľ nevygenerovaný, alebo sa nevyčerpá počet pokusov udaný druhým parametrom. Po vyčerpaní pokusov sa vyberá najnižší nepoužitý index, čo je dosiahnuté priebežným uchovávaním najnižšieho voľného indexu. Využitie tohto generátora vidieť najmä pri filtroch využívajúcich náhodnosť v rámci pozície vo filtrovanom obraze, kde by generovanie klasickým lineárnym kongruentným generátorom nemuselo pokryť celý filtrovaný obraz, alebo by to trvalo veľmi dlho. Aj v tomto prípade majú konštanty hodnoty: $c = 1$ a $a = 69069$.

6.5.10 Transformácia rozloženia

Celkovo sú okrem rovnomerného, implementované ďalšie dva typy pravdepodobnostného rozloženia. Sú to exponencionálne a normálne, Gaussovo rozloženie. Exponencionálne rozloženie využíva pre transformáciu lineárneho rozloženia metódu inverznej funkcie, ktorá má nasledujúci tvar 6.3.

$$y = \mu - \sigma \cdot \ln(1 - x), \tag{6.3}$$

U normálneho rozloženia je nutné riešiť hneď dva problémy implementácie. Prvým je neobmedzený rozsah generovaných hodnôt, čo vyplýva z faktu, že asymptoty funkcie rozloženia sú v nekonečne a teda nie je možné použiť jednoduchú metódu vylučovania. Druhým je

tvar samotnej funkcie 6.4 a odvodzovanie jej inverznej podoby:

$$P(x) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \tag{6.4}$$

[6]

jej integráciou dostávame distribučnú funkciu Gaussovho rozloženia 6.5,

$$D(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \tag{6.5}$$

[6]

kde *erf* je funckia chyby, ktorá je odvodená od normalizovanej Gaussovej funkcie a ktorej matematické vyjadrenie nasleduje 6.6.

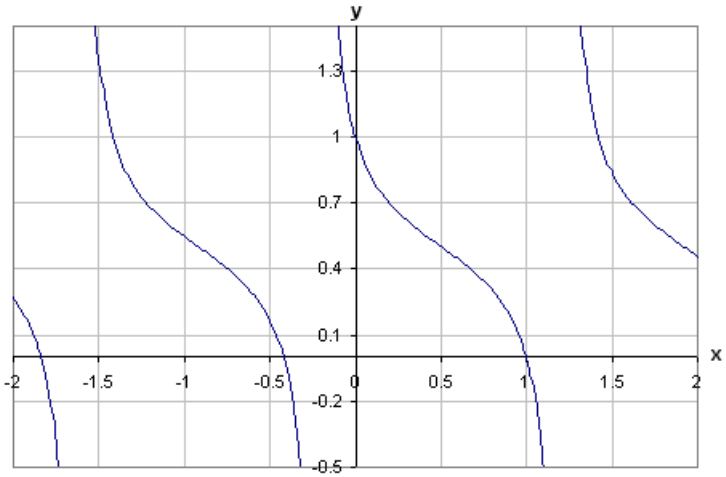
$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \cdot \int_0^z e^{-r^2} \mathrm{d}t, \tag{6.6}$$

[5]

Ako vidieť jedná sa o vyjadrenie obsahujúce integráciu, čo pre programovú implementáciu znamená netriviálny problém. Možným riešením je aproximácia výsledku, čo je však z dôvodu veľkej časovej náročnosti neprijateľné riešenie. Riešením oboch spomenutých problémov je aproximácia distribučnej funkcie rozloženia goniometrickými funkciami, u ktorých je jednoduché obmedziť rozsah funkčných hodnôt. Po sérii pokusov sa ako najvhodnejšou funkciou pre tento účel ukázala byť nasledujúca funkcia 6.7,

$$f(x) = \frac{\tan(2x \cdot \operatorname{atan}(\sigma) + \operatorname{atan}(\sigma^{-1}))^{-1}}{2\sigma} + \frac{1}{2} \tag{6.7}$$

kde σ definuje špicatosť a ktorej priebeh vidno na obrázku 6.17. Z uvedeného obrázku taktiež vidno, že táto funkcia je aproximáciou len polovice nami požadovanej funkcie a zároveň ide o jej inverznú podobu.



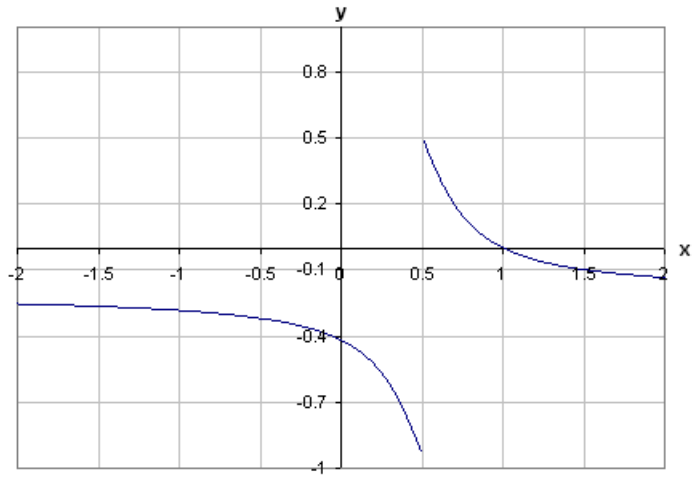
Obrázok 6.17: Aproximácia distribučnej funkcie Gaussovho rozloženia pravdepodobnosti

Vzhľadom k tomu, že priebeh normálneho rozloženia, Gaussova krivka, je bez posunu súmerná podľa osi y, je jednoduché implementovať generovanie čísel aj v jej zápornej polovici. Inverziu funkcie odstránime jej inverziou podľa nasledujúceho postupu:

$$\begin{aligned} x &= \frac{\operatorname{cotan}(2y \cdot \operatorname{atan}(\sigma) + \operatorname{atan}(\sigma^{-1}))}{2\sigma} + \frac{1}{2} \\ \sigma \cdot (2x-1) &= \operatorname{cotan}(2y \cdot \operatorname{atan}(\sigma) + \operatorname{atan}(\sigma^{-1})) \\ \operatorname{atan}\left(\frac{1}{\sigma \cdot (2x-1)}\right) &= 2y \cdot \operatorname{atan}(\sigma) + \operatorname{atan}(\sigma^{-1}) \\ y &= \frac{\operatorname{atan}\left(\frac{1}{\sigma \cdot (2x-1)}\right) - \operatorname{atan}(\sigma^{-1})}{2 \cdot \operatorname{atan}(\sigma)} \end{aligned} \tag{6.8}$$

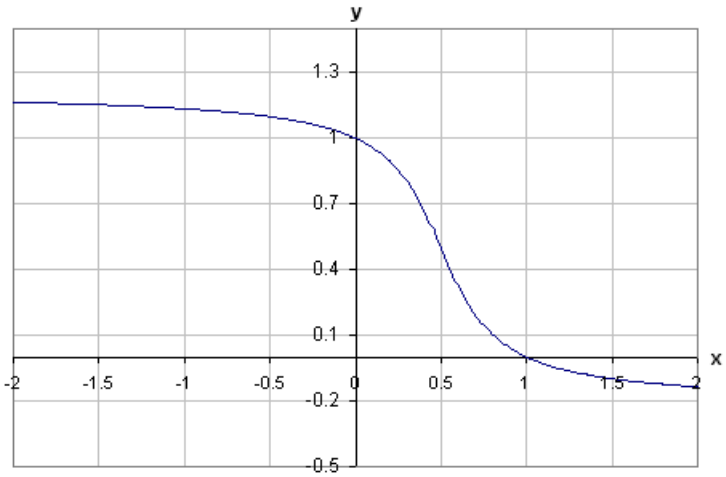
Priebeh invertovanej funkcie 6.8 vidíme na obrázku 6.18, z ktorého tiež vidieť, že ani po invertovaní nami vybraná funkcia ešte stále neodpovedá požadovanému priebehu a preto je nutná jej nasledujúca úprava:

$$y = \frac{\operatorname{atan}(\sigma \cdot (1-2x)) - \operatorname{atan}(\sigma)}{2 \cdot \operatorname{atan}(\sigma)} + 1. \tag{6.9}$$



Obrázok 6.18: Inverzná podoba aproximačnej funkcie

Takto upravenú funkciu 6.9 už môžeme použiť pre transformáciu normálneho rozloženia na želané, pseudonormálne rozloženie. Výhodou tejto funkcie je jej obor funkčných hodnôt, ktoré pre $x \in \langle 0; 1\rangle$ nepresahuje hodnotu 1 a už zmienený koeficient špicatosti σ. Pre naše použitie sa však obmedzíme len na interval $\langle 0; 0,5\rangle$. Priebeh výslednej funkcie je na nasledujúcom obrázku 6.19:



Obrázok 6.19: Výsledná funkcia aproximujúca distribučnú funkciu Gaussovho rozloženia pravdepodobnosti.

Kapitola 7

Testovanie a výsledky testov

Testovanie z hľadiska kvality výstupu je ponechané na čitateľovi. Výsledky v tomto smere dostatočne reprezentujú názorné ukážky aplikácie jednotlivých filtrov, ktoré sú okrem tejto kapitoly obsiahnuté v kapitolách 3 teoretickej časti a podkapitole 6.5 časti praktickej. Predmetom tejto kapitoly je najmä testovanie rýchlosti filtrácie a flexibility konkrétnych filtrov t.j. ich modifikovateľnosti parametrami s využitím metódy porovnávania. Porovnávanými aplikáciami sú voľne dostupné programy umožňujúce post-processing obrazu a implementujúce filtre pre nefotorealistické zobrazovanie. Týmito aplikáciami sú: GIMP 2.6.6 a irfanview 4.25.

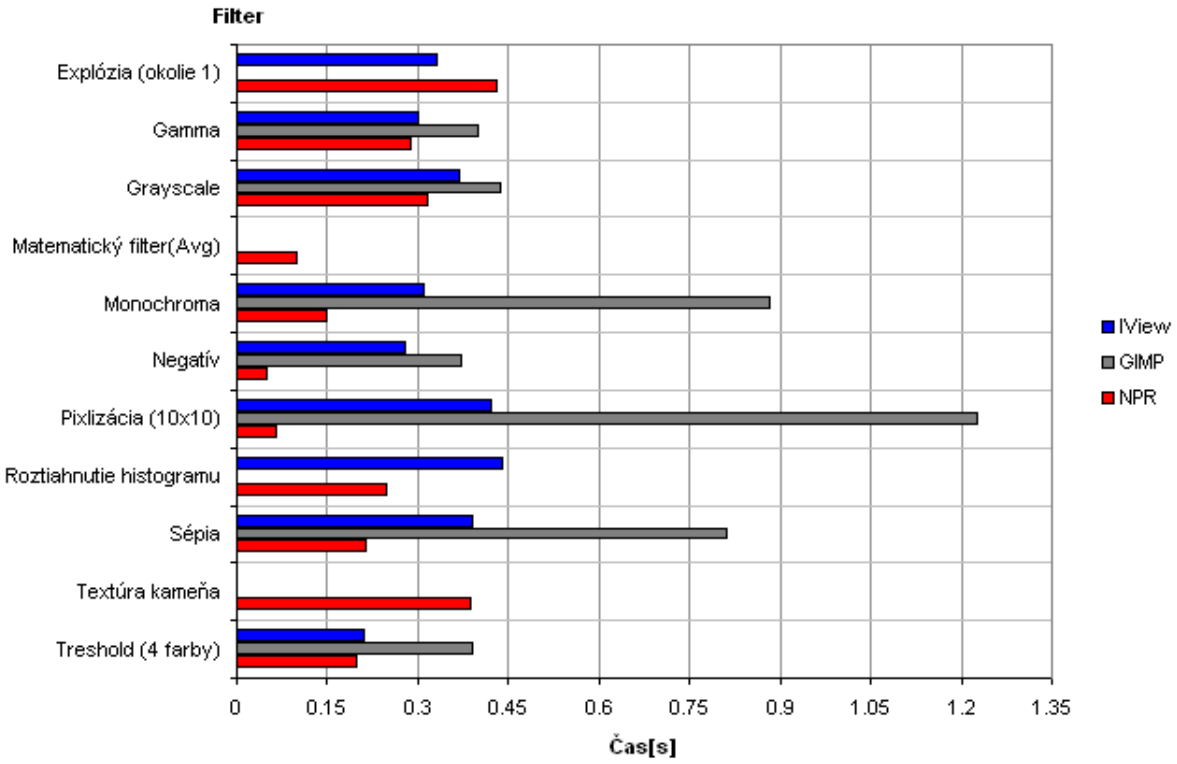
GIMP, alebo GNU Image Manipulation Program je aplikácia vyvinutá pod X11, s primárnym určením pre jednoduchú, ale aj pokročilú úpravu obrazu. Podporuje operačné systémy typu Linux, Windows a MAC.

Druhou testovacou aplikáciou je IrfanView. Ide o aplikáciu určenú pre operačný systém Windows so zameraním na prehliadanie obrazových dát, známu svojou rýchlosťou spracovania obrazu. Súčasťou IrfanView je aj balíček funkcií, do ktorých spadá podpora filtrácie.

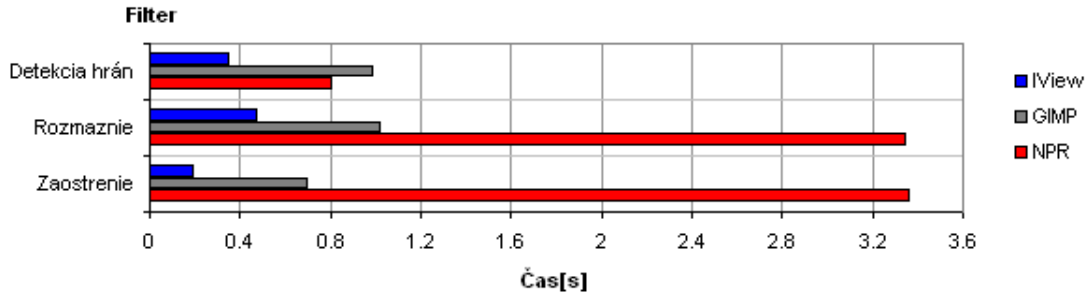
7.1 Rýchlostné testy

Test rýchlosti, ktorého výsledky nasledujú, obsahuje celkom tri časti, v ktorých je postupne filtrovaný obraz medián filtrom, filtrom olejomaľby a nakoniec ostatnými implementovanými filtrami tejto práce.

Test medián filtrom reprezentuje okrem rýchlosti medián filtra aj väčšinu filtrov využívajúcich pre svoj výpočet okolie pixelu. Vyplýva to z princípu filtrov tohto typu a s tým súvisiacej exponencionálnej závislosti rýchlosti filtra od veľkosti použitej matice okolia. Medzi tieto filtre okrem medián filtra patria: filter detekcie hrán, filter zaostrenia a filter rozmazania. Uvedené filtre nie sú testované v širšom rozsahu veľkostí matice okolia, ale len pre jej jednu, pevne stanovenú veľkosť. Čitateľ by si mal teda pri pohľade na výsledky uvedomiť, že daný výsledok nevypovedá o rýchlosti filtra, no na základe konkrétneho čísla a charakteru závislosti rýchlosti od veľkosti matice okolia tohto typu filtra je možné ostatné hodnoty približne odhadnúť. Samostatný test je venovaný modifikovanej verzii medián filtra, ktorého výsledky sú z dôvodu prehľadnosti zahrnuté do výsledkov prvého testu. V druhom teste je testovaný filter olejomaľby, u ktorého je využitá obdobná metodika testovania ako pre medián filter. Posledný test zahŕňa všetky ostatné filtre, ktorých výsledky sú rozdelené do tabuľky podľa základnej kategorizácie filtrov.



Obrázok 7.1: Výsledky testov filtrov operujúcich s jednotlivými pixelmi



Obrázok 7.2: Výsledky testov filtrov využívajúcich okolie pixelu

7.2 Test flexibility

Test flexibility zahŕňa porovnávanie počtu užívateľom nastaviteľných parametrov, rovnako ako aj ich rozsah. Jednotlivé typy filtrov a prehľad ich nastaviteľných parametrov, bez implicitného parametra obrazu je zobrazený v nasledujúcej tabuľke:

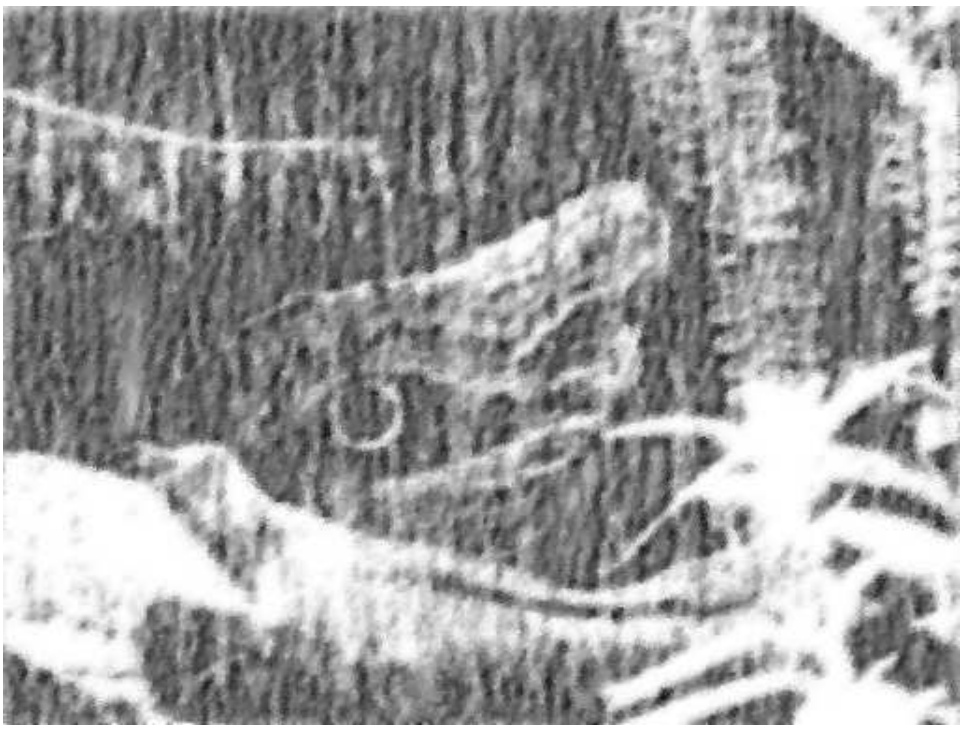
		Aplikácia		
		NPR	IrfanView	GIMP
Filter	Parameter	Rozsah	Rozsah	Rozsah
Grayscale	metóda	-	-	konštanta
Monochroma	úroveň	$0 \div 255$	-	-
Negatív	-	-	-	-
Sépia	úroveň červenej	$0 \div 255$	-	-
	úroveň zelenej	$0 \div 255$	-	-
	úroveň modrej	$0 \div 255$	-	-
Treshold	počet úrovní	$0 \div 255$	$2 \div 255$	$2 \div 256$
Rozmazanie	okolie	$0 \div 255$	-	
	výška okolia	-	-	$0,0 \div 960,0$
	šírka okolia	-	-	$0,0 \div 960,0$
	intenzita	$0,0 \div 100,0$	$1 \div 99$	-
Gamma filter	koeficient	$0,0 \div 100,0$	$0,01 \div 6,99$	graf
Pixelizácia	veľkosť pixelu	$0 \div 2^{32} - 1$	$2 \div 50$	-
	výška pixelu	-	$2 \div 50$	$0,0 \div 960,0$
	šírka pixelu	-	$2 \div 50$	$0,0 \div 960,0$
Explózia	okolie	$0 \div 255$	$1 \div 99$	-
	intenzita	$0,0 \div 100,0$	-	-
Textúra kameňa	granularita	$0 \div 255$	-	konštanta
	intenzita	$0,0 \div 100,0$	-	$0,0 \div 960,0$
Matematický filter	operand	$1 \div n$	-	-
Roztiahnutie histogramu	-	-	-	-
Zaostrenie	okolie	konštanta	-	-
	intezita	$0,0 \div 100,0$	$1 \div 99$	$1 \div 99$
	(vlastná matica)	$x \times y$	-	-
Detekcia hrán	okolie	konštanta	$2 \div 5$	konštanta
	intenzita	$0,0 \div 100,0$	-	-
	(vlastná matica)	$x \times y$	-	-
Medián	okolie	$0 \div 255$	$3 \div 9$	$1 \div 20$
	intenzita	$0,0 \div 100,0$	-	$1 \div 20$
S Medián	okolie	$0 \div 255$	-	-
	intenzita	$0,0 \div 100,0$	-	-
Olejomaľba	veľkosť štetca	$0 \div 255$	$0 \div 255$	$3 \div 50$
	rozptyl	-	-	$1 \div 20$

Tabuľka 7.1: Zoznam parametrov filtrov a ich rozsahu jednotlivých aplikácií

7.3 Príklady výstupu

Nasledujú ukážky výstupu implementovaného systému spolu so zdrojovým skriptom, ktoré by mali čitateľovi ukázať možnosti tohto systému, rovnako ako aj jeho jednoduché a intuitívne použitie.

```
/*** Príklad zloženého filtra imitujúceho kresbu uhlíkom ***/
a = INPUT("Ref2.bmp");
// Príprava šumu s využitím textúry kameňa
b = MONO(TEX_STONE(a - 255, 1, 32), 16);
b *= b;
// Prevod do monochromatického zobrazenia s pridaním pripraveného šumu
a = !EDGE(MONO(TEX_STONE(a, 1, 32) + b, 128), MOOR, 100);
// Vyhladenie ťahov uhlíka
a = SMEDIAN(EXPLO(BLUR(a, 1, 100), 1, 30), 1, 50);
// Výsledný obraz si môžme kedykoľvek zobraziť
DISPLAY(a);
OUTPUT("Out1.bmp", a);
```



Obrázok 7.3: Imitácia kresby uhlíkom

```
/*** Príklad zloženého filtra imitujúceho kresbu sprejom na múr ***/
a = INPUT("Ref2.jpg");
b = INPUT("Brick.bmp");
// Zaostrenie obrazu s využitím vlastnej matice
a = SHARP(a, CUSTOM, 50, MTX(3,
                                    1.0, 0.5, 0.0,
                                    0.0, 1.0, 0.5,
                                    0.5, 0.0, 1.0));
// Imitácia kresby sprejom
a = EXPLO(OIL(a * (a / 64), 3), 2, 30);
// Nastavíme mód práce s obrazom rozdielných rozmerov
PMODE(REPEAT);
// Spajánie obrazov prostým sčítaním
OUTPUT("Out2.png") = a * 0.6 + b * 0.4;
```

Obrázok 7.4: Imitácia kresby sprejom na stenu

```
/*** Príklad zloženého filtra imitujúceho kresbu fixami ***/
/*** s efektom "západu slnka"***/
// Ukážka využitia definície
DEFINE aS6 #(a >> 6)#;
a = INPUT("Ref2.jpg");
// Príprava filrov základných farieb do samostatných obrazov
r = SEPIA(a, 255, 0, 0);
g = SEPIA(a, 0 ,255, 0);
b = SEPIA(a, 0, 0, 255);
// Pripravíme si zjednodušený obraz originálu pre presnejšiu detekciu hrán
edge = STRETCH_HISTO((aS6 << 6) | (aS6 << 4) | (aS6 << 2) | aS6);
// Prevedieme obraz filtrom "západu slnka"
a = a + r - b - g / 4;
// Detekujeme hrany a vymaskujeme nimi upravený obraz
a = a & !MONO(EDGE(edge, HRZ, 100) + EDGE(edge, VRT, 100), 30);
OUTPUT("Out3.png", a);
```

Obrázok 7.5: Imitácia kresby fixami s efektom západ slnka

Ako referenčný obraz bol použitý obraz typu jpg, s 24 bitmi na pixel a s rozlíšením 1024×768 bodov. Celkový počet jedinečných farieb tohto obrazu je 236955 a v danom formáte zaberá 540,22KB. Testy boli vykonané na stroji s nasledujúcimi parametrami: CPU – T5500, $1,66$ GHz, 667 MHz FBS, 2MB L2 cache RAM – $0,99$ GB, DDR2

Kapitola 8

Záver

Výsledkom tejto práce je aplikácia schopná nefotorealistického zobrazovania s využitím nie len základných, ale aj špecializovaných filtrov, ktoré simulujú základné tri maliarske nástroje. Jednotlivé filtre sú pritom implementované s dôrazom na ich flexibilitu, univerzálnosť a hlavne malú časovú náročnosť. Výsledná aplikácia okrem použitia filtrov jednotlivo, umožňuje aj ich vzájomné kombinovanie v podobe sekvenčne spracovávaných skriptov. Tie sú písané formou rovníc, ktorých zápis je silne intuitívny a pre orientáciu a pochopenie ich zápisu stačia základné matematické znalosti. Z hľadiska kompatibility je program preložiteľný na väčšine operačných systémov typu Windows a Linux. V porovnaní s obdobnými aplikáciami, máme na mysli aplikácie schopné filtrovať obraz, je táto aplikácia porovnateľne rýchla, v niektorých prípadoch dokonca rýchlejšia a flexibilnejšia ako porovnávané aplikácie, čo je doložené výsledkami testov v predchádzajúcej kapitole.

V budúcnosti by bolo vhodné dotvoriť k už vytvorenému interpréteru grafické užívateľské rozhranie, ktoré by umožňovalo písanie skriptov systémom výberu z množiny dostupných filtrov jednoduchým kliknutím myšou. Značne by sa tým uľahčila samotná tvorba skriptov pre bežného užívateľa. Ďalším rozšírením by mohlo byť doplnenie aplikácie o podporu práce s 3D modelmi, rovnako ako aj paralelizácia interpretácie resp. výpočtu aplikovaných filtrov.

Literatúra

[1] Foley, J. D.: *Computer graphics*. Addison-Wesley Publishing Company, Inc., druhé vydanie, 1995, ISBN 0-201-84840-6, 1175 s.

[2] González, R. C.; Woods, R. E.: *Digital image processing*. Pearson Education Inc., tretie vydanie, 2008, ISBN 0-13-168728-8, 954 s.

[3] Kent, A.; Williams, J. G.: *Encyclopedia of Microcomputers*. 270 Madison Avenue, New York, USA: Mercel Dekker, Inc., 1995, ISBN 0-8247-2714-2, 416 s.

[4] Strothotte, T.; Schlechtweg, S.: *Non-photorealistic computer graphics*. 340 Pine Street, San Francisco, USA: Morgan Kaufmann Publishers, 2002, ISBN 1-55860-787-0, 470 s.

[5] Weisstein, E. W.: Erf. online.
URL http://mathworld.wolfram.com/Erf.html

[6] Weisstein, E. W.: Normal Distribution. online.
URL http://mathworld.wolfram.com/NormalDistribution.html

[7] Xiang, Z.; Plastock, R. A.: *Schaum's outline of theory and problems of computer graphics*. McGraw-Hill, druhé vydanie, 2000, ISBN 0-201-13447-0, 347 s.