

Algoritmy transformace 2D obrazu

2D image transformation algorithms

Bc. Rastislav Petráš

Diplomová práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Rastislav PETRÁŠ**
Osobní číslo: **A09715**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Algoritmy transformace 2D obrazu**

Zásady pro vypracování:

1. Vytvořte literární rešerši na zadané téma.
2. Seznamte se s běžně používanými algoritmy transformace barev 2D rastrového obrazu. V práci je popište.
3. Seznamte se s běžně používanými algoritmy geometrických transformací 2D rastrového obrazu. V práci je popište.
4. Navrhněte vlastní program, který bude umožňovat provádět vybrané geometrické i barevné transformace 2D rastrových obrazů.
5. Tento program naprogramujte ve zvoleném programovacím jazyce. Odladte jej a ověřte jeho funkčnost na vhodně zvolených příkladech. Program bude umožňovat vstupy a výstupy obecně rozšířených formátů grafických souborů.
6. Vytvořte podrobnou elektronickou dokumentaci k nastudovaným transformačním algoritmům tak, aby se dala využít při výuce předmětu Počítačová grafika.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. BAYER, Jürgen. C# 2005 : Velká kniha řešení. Vydání první. Brno : Computer Press, a.s., 2007. 813 s. ISBN 978-80-251-1620-3.
2. DOBEŠ, Michal. Zpracování obrazu a algoritmy v C#. 1. vydání. Praha : BEN – Technická literatura, 2008. 143 s. ISBN 978-80-7300-233-6, EAN 9788073002336.
3. JOHNSON, James T. The Code Project [online]. 2002 [cit. 2011-01-26]. Image Rotation in .NET. Dostupné z WWW: [http://www.codeproject.com/KB/graphics/rotateimage.aspx].
4. MAREŠ, Amadeo. 1001 tipů a triků pro C#. Vydání první. Brno : Computer Press, a.s., 2008. 360 s. ISBN 978-80-251-2125-2.
5. Microsoft. Craigs Utility Library [online]. 2011 [cit. 2011-01-26]. Dostupné z WWW: [http://cul.codeplex.com/SourceControl/changeset/view/61004707831].
6. NAGEL, Christian, et al. C# 2008 : Programujeme Profesionálně. Vydání první. Brno : Computer Press, a.s., 2009. 1126 s. ISBN 978-80-251-2401-7.
7. Počítačová Grafika [online]. 2010 [cit. 2011-01-25]. TRANSFORMÁCIE V PG . Dostupné z WWW: [http://pg.kpi.fei.tuke.sk/?q=node/6].
8. ŠTUGEL, Juraj. Netgraphics : výuka počítačovej grafiky [online]. 2011 [cit. 2011-01-26]. Dostupné z WWW: [http://www.netgraphics.sk/sk].

Vedoucí diplomové práce:

Ing. Pavel Pokorný, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

24. února 2011

Termín odevzdání diplomové práce:

18. května 2011

Ve Zlíně dne 24. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Táto diplomová práca je zameraná na problematiku spracovania a úpravy rastrovej grafiky v prostredí .NET.

Na začiatku, v teoretickej časti práca objasňuje základné pojmy v počítačovej grafike. Následne sa venuje najčastejšie používaným transformáciám a filtrom grafických rastrových súborov.

Praktická časť obsahuje implementáciu týchto filtrov v programe, ktorý bol vytvorený pre diplomovú prácu. Popisuje aplikáciu a nastavenie každého algoritmu aj s príkladom.

Kľúčové slová: .NET, C#, spracovanie obrazu, filtrácia obrazu, farebné filtre, transformácia 2D obrazu

ABSTRACT

The thesis deals with the question of processing and adjusting of raster graphic in .NET environment.

At the beginning, in the theoretical part, the thesis clarifies the basic expressions in computer graphic. Consequently, it attends to the most frequently used transformations and filters of graphic raster files.

The practical part contains the implementation of these filters in the program which was created for the thesis. It describes the application and set-up of each algorithm with an example, as well.

Key words: .NET, C#, image processing, image filtration, colour filters, image transformations.

Chcel by som sa poďakovať môjmu vedúcemu diplomovej práce Ing. Pavlovi Pokornému, Ph.D., za odborné vedenie, cenné rady pri písaní a čas na konzultácie, ktorý si pre mňa vždy našiel.

Taktiež musím poďakovať mojej rodine a priateľke, že mali so mnou toľko trpezlivosti počas štúdia a pri písaní tejto práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	10
I TEORETICKÁ ČASŤ	11
1 ÚVOD DO POČÍTAČOVEJ GRAFIKY	12
1.1 REPREZENTÁCIA OBRAZOVÝCH DÁT.....	12
1.1.1 VEKTOROVÉ DÁTA	12
1.1.2 RASTROVÉ DÁTA	12
1.2 FARBY.....	12
1.2.1 FARBY.....	12
1.2.2 ADITÍVNE MIEŠANIE FARIEB	13
1.2.3 SUBTRAKTÍVNE MIEŠANIE FARIEB	13
1.3 FAREBNÉ MODELY	14
1.3.1 RGB.....	14
1.3.2 CMY A CMYK.....	14
1.3.3 HSV A HLS	15
1.4 FORMÁTY GRAFICKÝCH SÚBOROV	17
1.4.1 BMP	17
1.4.2 TIFF.....	17
1.4.3 JPEG.....	18
1.4.4 RAW.....	18
1.4.5 GIF 18	
1.4.6 PNG.....	19
2 GEOMETRICKÉ TRANSFORMÁCIE OBRAZU	20
2.1 POSUNUTIE.....	20
2.2 OTOČENIE	20
2.3 ZMENA MIERKY	21
2.4 SKOSENIE.....	21
2.5 ODRAZ	21
2.6 WARPING	22
3 TRANSFORMÁCIE FARIEB OBRAZU	24
3.1 ODTIEŇ ŠEDEJ.....	24
3.2 NEGATÍV	24
3.3 GAMA.....	25
3.4 JAS	26
3.5 KONTRAST.....	27
3.6 INTERPOLÁCIA.....	27
3.7 HISTOGRAM	28
3.7.1 VYROVNANIE HISTOGRAMU	29
4 FILTRÁCIA OBRAZU	31
4.1 ODSTRAŇOVANIE ŠUMU	31

4.1.1	LINEÁRNE METÓDY	31
4.1.2	NELINEÁRNE METÓDY	33
4.2	DETEKCIA HRÁN	33
4.2.1	POMOCOU PRVEJ DERIVÁCIE	34
4.2.2	POMOCOU DRUHEJ DERIVÁCIE	36
5	VÝVOJOVÉ PROSTREDIE .NET A JAZYK C#	38
5.1	ARCHITEKTÚRA .NET	38
5.2	PROGRAMOVACÍ JAZYK C#	40
II	PRAKTICKÁ ČASŤ	41
6	IMPLEMENTÁCIA ALGORITMOV VO VÝVOJOVOM PROSTREDÍ	42
6.1	MOŽNOSTI ÚPRAVY V JAZYKU C#	42
6.1.1	TRIEDA BITMAP A IMAGE	42
6.1.2	TRIEDA COLORMATRIX	42
6.1.3	POUŽITIE UKAZATELOV	44
6.1.4	POUŽITIE POĽA	44
6.1.5	POROVNANIE RÝCHLOSTI	45
6.2	POUŽITÉ ALGORITMY	46
6.3	ALGORITMY PRE SPRACOVANIE FARIEB	46
6.3.1	ODTIEŇ ŠEDEJ	46
6.3.2	NEGATÍV	48
6.3.3	ÚPRAVA GAMY	50
6.3.4	ZMENA JASU	52
6.3.5	ZMENA KONTRASTU	54
6.4	GRAFICKÉ FILTRE	56
6.4.1	KONVOLUČNÁ MASKA	56
6.4.2	VYHLADZOVANIE OBRAZU	56
6.4.3	GAUSSOVO VYHLADZOVANIE OBRAZU	57
6.4.4	OSTRENIE OBRAZU	58
6.4.5	ŠTREDNÉ OSTRENIE OBRAZU	59
6.4.6	VYTLAČENÝ VZOR – EMBOSS	60
6.4.7	DETEKCIA HRÁN	61
7	GRAFICKÝ EDITOR PRE TRANSFORMÁCIE OBRÁZKOV	63
7.1	PROGRAMÁTORSKÝ POHĽAD	63
7.1.1	ŠTRUKTÚRA PROGRAMU	63
7.1.2	TRIEDA <i>IMAGEALGORITHMS</i>	63
7.1.3	TRIEDA <i>CONVOLUTIONMATRIX</i>	64
7.2	UŽÍVATEĽSKÝ POHĽAD	64
7.2.1	SYSTÉMOVÉ POŽIADAVKY	64
7.2.2	INŠTALÁCIA	65
7.2.3	POPIS PROGRAMU	65
ZÁVER	66

ZÁVER V ANGLIČTINE.....	68
ZOZNAM POUŽITEJ LITERATÚRY	68
ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK.....	72
ZOZNAM OBRÁZKOV	73
ZOZNAM TABULIEK	74
ZOZNAM PRÍLOH.....	75

ÚVOD

V dnešnej modernej dobe sa stretávame so spracovaním obrazu na každom kroku. Či už v novinách, časopisoch, letákoch, reklamných tabulách, internetových stránkach. Často aj s upravenými fotkami, ktoré nezachytávajú skutočnú situáciu. Všetky tieto úpravy sú robené pomocou programov pre editáciu grafických dát.

Pri tvorbe týchto programov je hlavné hľadisko jednoduchosť a rýchlosť tak, aby užívateľ a príliš nezaťažovali zložitou. V poslednej dobe sa stretávame s integrovaním programov priamo do fotoaparátov alebo mobilných telefónov. Zväčša ide o jednoduché úpravy, ktoré sa v profesionálnej sfére aplikujú až po odfotení v profesionálnych programoch.

Medzi najbežnejšie používané úpravy medzi širokou verejnosťou patrí hlavne úprava farieb a vyhladzovanie. Transformácie obrazu majú však v počítačovej grafike oveľa väčší pojem napr. geometrické transformácie, transformácie farieb, spracovanie 2D a 3D obrazu. Všetky transformácie sú náročné na výpočtový výkon. To vedie vývojárov k optimalizovaniu softvéru a vyvíjaniu nového a rýchlejšieho hardvéru.

Technológia .NET je v súčasnej dobe najrozšírenejšia platforma, určená pre vývoj a beh aplikácií určených pre platformu Microsoft Windows. Podobne ako u operačných systémoch Windows, stojí aj za .NET firma Microsoft.

Pomocou .NET môžeme vyvíjať aj aplikácie pre web, web stránky, databázové systémy a aplikácie pre mobilné systémy (Windows Mobile).

Táto technológia má aj svoje úskalia. Najväčším z nich je slabý výkon v matematických operáciách.

Táto práca sa po vysvetlení základných transformácií venuje ich implementácií do prostredia .NET pomocou programovacieho jazyka C# tak, aby princíp pochopil aj začiatočník v danej oblasti. Takto získané poznatky môže ďalej použiť na zdokonalenie, prípadne vyhnutie sa chybám v programe a strate času.

I. TEORETICKÁ ČASŤ

1 ÚVOD DO POČÍTAČOVEJ GRAFIKY

1.1 Reprezentácia obrazových dát

1.1.1 Vektorové dáta

Obrazové dáta sú reprezentované vektormi. Dáta obsahujú informácie o objektoch zložených z kružníc, kriviek a jednoduchých telies, ktoré umožňujú ich geometrickú konštrukciu. Pokiaľ je takto uložená napríklad kružnica, súbor neobsahuje informácie o všetkých jednotlivých bodoch, na ktorej leží. Obsahuje informáciu, že sa jedná o kružnicu, potom informácie o jej strede, jedného bodu, ktorý na nej leží a posledný bod určuje rovinu jej konštrukcie. Ďalej sú to informácie o farbe objektu a hrúbke čiary. Vektorové dáta sa používajú hlavne pre technické výkresy v CAD systémoch, kde zmena mierky, alebo prípadná editácia objektov a tvarov nespôsobí skreslenie. Nie je však vhodná pre reprezentáciu náhodných obrazových dát ako sú fotografie.

1.1.2 Rastrové dáta

Obraz je pri rastrovom formáte uložený v matici, kde každý bod matice reprezentuje jeden bod v obraze. Výhodou takéhoto formátu je jednoduchá úprava každého bodu zvlášť. Na tomto princípe pracuje väčšina zobrazovacích zariadení ako sú monitory, tlačiarne, televízory, atď.. Kvalita obrázku je daná predovšetkým počtom bodov a farebnej hĺbke.

1.2 Farby

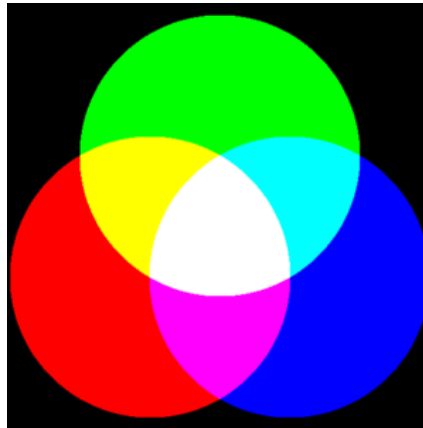
1.2.1 Farby

Oko je vo svojej činnosti obmedzené nielen do veľkosti vnímaných objektov, ale aj do počtu farieb. Veľkosť bodu na obrazovke je závislá na veľkosti a rozlíšení monitora a pohybuje sa rádovo v desatinách milimetrov. Pokiaľ je vedľa seba umiestených niekoľko takýchto bodov a každý ma inú farbu, za normálnych podmienok oko nie je schopné body od seba odlíšiť. Zraková informácia je v mozgu integrovaná a my môžeme vnímať farbu, ktorá na monitore nie je zobrazená. Dôležité pri vnímaní farieb je spôsob, akým je farba vytvorená. Je všeobecne známe, že farby môžeme „miešať“. V farebných modeloch sú preto farby realizované miešaním základných farieb.

Farebné modely sa delia na dve hlavné skupiny: aditívne a subtraktívne. [1]

1.2.2 Aditívne miešanie farieb

Pri aditívnych systémoch je podklad čierny a farby vznikajú pridávaním troch základných farieb a to červenej, zelenej a modrej. V prípade prekrytia všetkých základných farieb pri plnej intenzite vzniká farba biela. V prípade nulovej intenzity vzniká farba podkladu, čiže čierna. Príklad na aditívne miešanie farieb je na obrázku (Obr. 1).

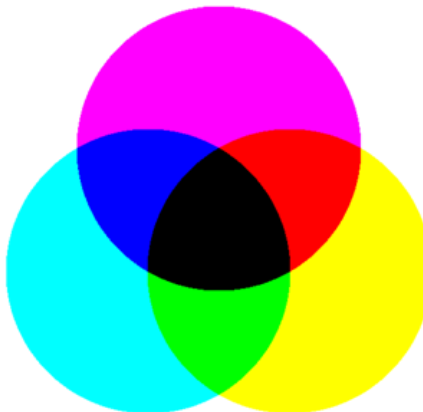


Obr. 1. Aditívny spôsob miešania farieb

Príkladom aditívneho miešania farieb sú televízory a monitory. [1]

1.2.3 Subtraktívne miešanie farieb

Subtraktívne systémy majú farbu podkladu bielu a farby vznikajú odpočítavaním od bielej. Základné tri farby sú: azúrová, purpurová, žltá. Azúrová farba pohlcuje červenú zložku svetla, purpurová zelenú a žltá modrú. V prípade prekrytia všetkých základných farieb pri plnej intenzite vzniká farba čierna. V prípade nulovej intenzity vzniká farba podkladu, čiže biela. Príklad na subtraktívne miešanie farieb je na obrázku (Obr. 2).



Obr. 2. Subtraktívny spôsob miešania farieb

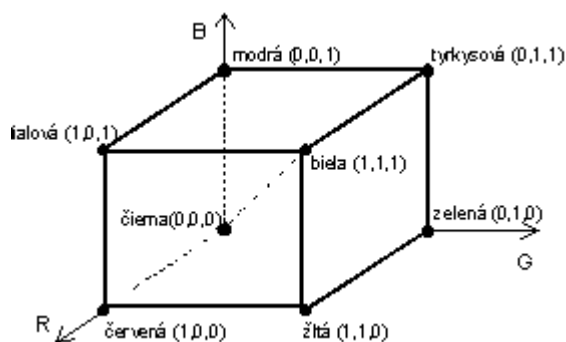
Príkladom subtraktívneho miešania farieb sú tlačiarne. [1]

1.3 Farebné modely

1.3.1 RGB

Ide o najpoužívanejší a najznámejší farebný model. Bol štandardizovaný v roku 1931. Farebný gamut je celý vyprodukovaný aditívnym spôsobom, a to miešaním troch základných farieb, červená (red), zelená (green) a modrá (blue).

Medzi najbežnejšiu implementáciu modelu RGB patrí 24-bitová. Vtedy pripadá na každú farebnú zložku 8 bitov, čo dáva dohromady 256 úrovní každej zložky. Celkový možný počet farieb nám dáva 2^{24} čo je približne 16,7 milióna. Platí, že čím väčšie číslo je, tým je farba svetlejšia. Celý model sa dá jednoducho zobraziť ako jednotková kocka umiestnená v osách r,g,b. Počiatok súradnicového systému $[0,0,0,]$ odpovedá čiernej farbe. Vrchol súradnicového systému v bode $[1,1,1,]$ odpovedá farbe bielej tak, ako je to graficky uvedené na obrázku (Obr. 3). V protiľahlých rohoch ležia farby RGB.

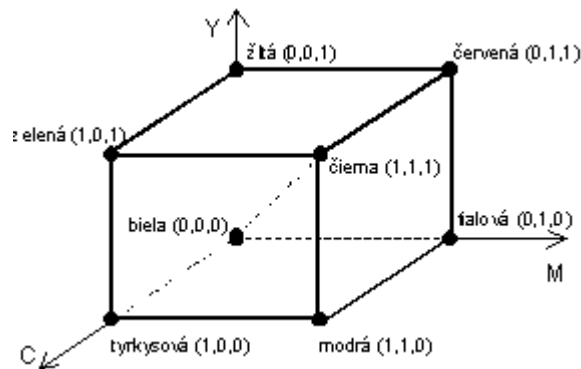


Obr. 3. Geometrické zobrazenie priestoru RGB

V praxi sa môžeme stretnúť s modelom RGBA, kde A je informácia o priehľadnosti. Môže naberať hodnôt od 0 do 1, pričom 0,0 znamená že farebný bod je nepriehľadný a 1,0 že bod je úplne priehľadný. [2]

1.3.2 CMY a CMYK

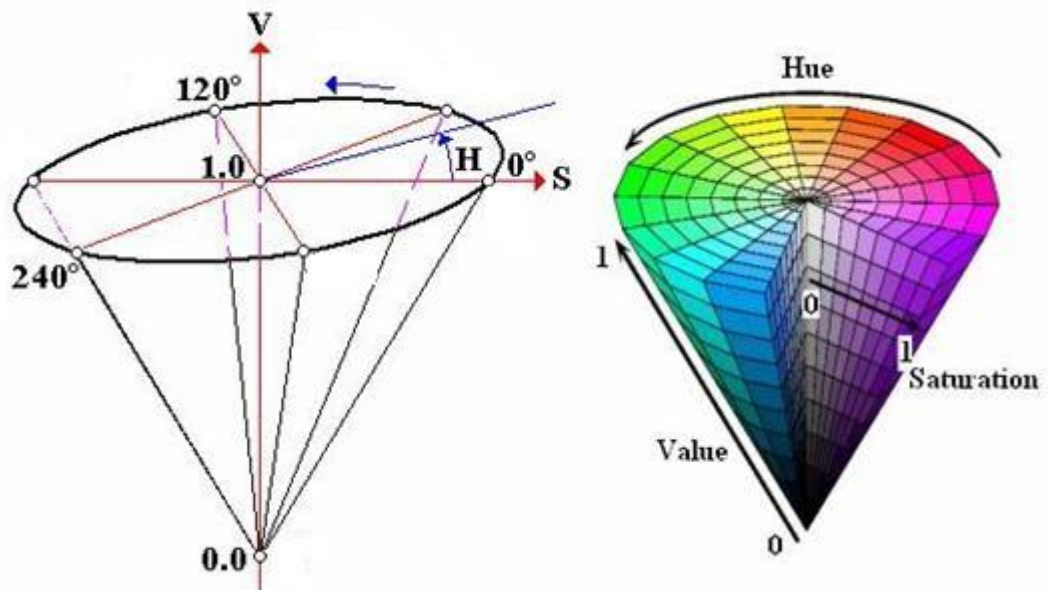
Farebný model sa skladá z farieb Cian (modrozelená), Magenta (purpurová) a Yellow (žltá). Nakoľko z týchto farieb sa nedá namiešať dokonale čierna farba, bola zavedená ďalšia farebná zložka K (black). Farby v tomto modeli vznikajú subtraktívnym spôsobom. Najčastejšie využitie tohto modelu je v tlači. Zobrazenie v jednotkovej kocke je v počiatočných súradniciach $[0,0,0,]$ farba biela a na vrchole $[1,1,1,]$ farba čierna tak, ako to reprezentuje obrázok (Obr. 4). [3]



Obr. 4. Geometrické zobrazenie priestoru CMY

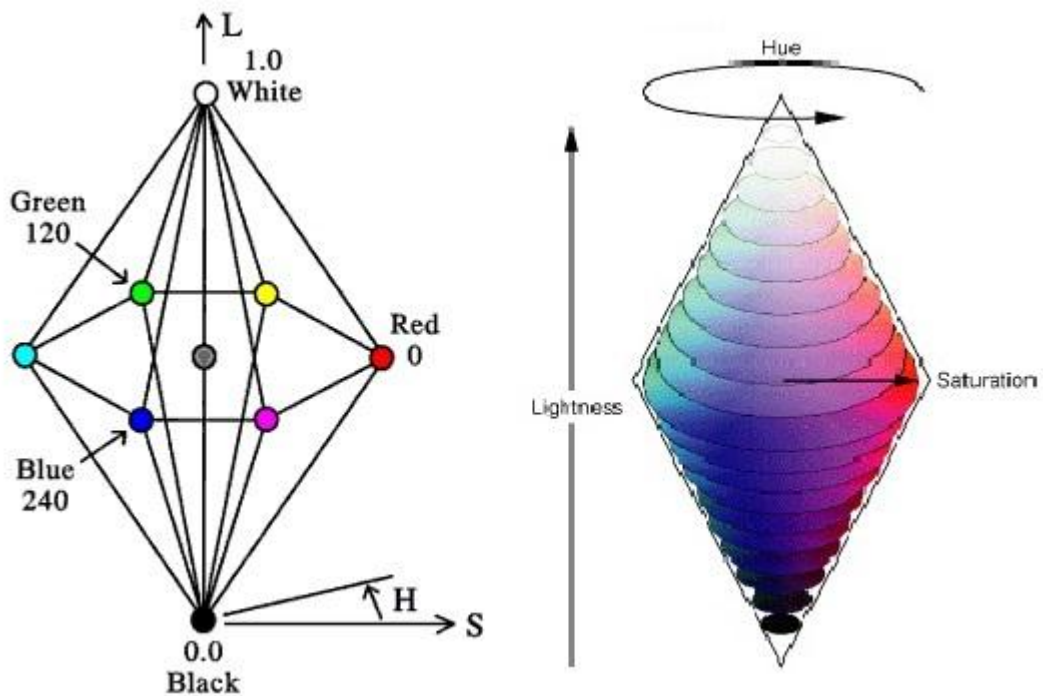
1.3.3 HSV a HLS

Patrí medzi systémy, kde farby nevznikajú miešaním. Skladá sa z farebného tónu – *Hue*, sýtosti – *Saturation* a jasovej hodnoty – *Value*. Farebný tón označuje prevládajúcu spektrálnu farbu, sýtosť určuje prímies iných farieb a jas je daný množstvom bieleho svetla. Používa sa napríklad v štandardnom farebnom editore Windows. Na zobrazenie priestoru sa používa šesťboký ihlan (Obr. 5). Jeho vrchol leží v počiatku sústavy súradníc HSV. Hodnoty súradníc S a V sa menia od 0 do 1, hodnoty uhlu H nadobúdajú hodnôt z intervalu od 0° PO 360° . Vrchol ihlanu reprezentuje čiernu farbu. Ako rastie smerom k podstave, stred podstavy reprezentuje farbu bielu. Sýtosť odpovedá relatívnej vzdialenosti bodu od osy ihlanu. Dominantné farby ležia na plášti, čisté sú na obvodě podstavy. Pri pohybe po obvodě v rovnakej výške od základne sa postupne mení farebný tón, sýtosť a jas zostávajú bez zmeny. Systém HSV má aj nedostatky, ktoré môžu sťažovať prácu s presným určením farby. Jedným z nich je ihlanovitý tvar, ktorý spôsobuje, že vo vodorovnom reze sa musí bod o konštantnej hodnote S pohybovať pri zmene H po dráhe v tvare šesťuholníka, čiže nie po kružnici. Medzi ďalšie nedostatky patrí nesymetria priestoru z hľadiska jas.



Obr. 5. Geometrické zobrazenie priestoru HSV [20]

Všetky spomenuté nedostatky rieši farebný priestor HLS. Skladá sa z farebného tónu – Hue, svetlosti – Lightness a zo sýtosti – Saturation. Priestor HLS je obdobou priestoru HSV, kde bol ihlan nahradený dvojicou kužeľov (Obr. 6). [2]



Obr. 6. Geometrické zobrazenie priestoru HLS [20]

1.4 Formáty grafických súborov

Rôzne formáty rastrových súborov sa vo všeobecnosti delia do dvoch kategórií a to, bez stratovej kompresie ako napríklad BMP, RAW, TIFF, TGA, PNG, MNG a so stratovou kompresiou ako JPEG, GIF.

Výhody bezstratovej kompresie sú hlavne pri ďalších úpravách obrázku, kde nedochádza ku strate kvality. Ich nevýhodou je až niekoľkonásobná veľkosť oproti komprimovaným obrázkom.

Výhoda stratovej kompresie je najmä v malej veľkosti súboru. Ich nevýhodou je nemožnosť obnovenia pôvodného obrázka. Využívajú nedokonalosť ľudského oka, takže bežný človek nerozozná rozdiely medzi stratovou a bezstratovou kompresiou.

1.4.1 BMP

Ide o grafický formát, ktorý je používaný Microsoftom. Je to bezstratový formát, ktorého najväčšou nevýhodou, prečo sa vo veľkej miere nepoužíva, je veľkosť. Maximálna farebná hĺbka je 24-bit. Existujú aj ďalšie modifikácie tohto formátu, ktoré pridávajú väčšiu farebnú hĺbku, alfa kanál, prípadne čiastočnú kompresiu. V praxi je používaný hlavne spoločnosťou Microsoft.

1.4.2 TIFF

Bol definovaný firmou Aldus v roku 1986 ako štandardná metóda pre ukladanie čiernobielych predlôh vytvorenými skenermi. V súčasnosti je TIFF najpoužívanejším štandardným bezstratovým formátom súborov vo väčšine kresliacich programov. Svoje využitie našiel hlavne v skenovaní. Jeho schopnosť rozšírenia a podpora veľkého množstva dátových kompresných schém dovoľuje vývojárom prispôbiť si TIFF formát svojim potrebám. Využíva sa hlavne u obrázkov, ktoré sa majú s istotou ďalej upravovať. Ide taktiež o výhradný formát pre Microsoft Windows GUI. Jeho schopnosť rozšírenia, kedy je schopný ukladať viacnásobné bitmapové predlohy, umožňuje splniť aj tie najnáročnejšie požiadavky pre ukladanie dát. Podporuje farebnú hĺbku až 24 bitov a veľa druhov kompresíí. Univerzálne vlastnosti mu umožňujú jeho použitie v akomkoľvek operačnom prostredí. Najväčšie nevýhody formátu TIFF sú sťažnosti na jeho údajnú neprenositelnosť medzi jednotlivými aplikáciami. Tieto problémy sa väčšinou zvädzajú hlavne na vlastné súbory TIFF. Medzi ďalšie nevýhody je ich veľkosť, čo je ale daňou za kvalitu. [4]

1.4.3 JPEG

Je v súčasnosti najpoužívanejší a najznámejší formát obrázkových súborov. Jeho najväčšie rozšírenie spôsobilo rozširovanie internetu. Ide o stratový formát. Jeho hlavnou výhodou je malá veľkosť. Skupina JPEG vznikla v roku 1986. Princíp kompresného algoritmu je v použití DCT (diskrétna kosínusová transformácia) a kvantizifikácií získaných koeficientov. Celá kompresia a strata údajov je vytvorená tak, aby sa strácali len informácie, ktoré ľudské oko nevie dobre rozlíšiť. To je totiž oveľa citlivejšie na zmenu intenzity farby ako na malú zmenu samotnej farby. Napríklad pri kompresnom pomere 15:1 až 25:1 nedôjde k výraznej strate kvality fotografie.

V oblasti kompresie obrazu sa snažia do popredia dostať algoritmy založené na vlnkovej transformácii (Wavelet Transformation). DWT (Diskrétna vlnková transformácia) dokáže reprezentovať obraz efektívnejšie ako napríklad DCT. DWT rozkladá obraz do tzv. báзовých funkcií, ktoré sú obecné výhodnejšie pre reprezentáciu digitálnych signálov než sínusové alebo kosínusové funkcie. Na tomto základe bol vytvorený štandard JPEG2000. Diskrétnu kosínusovú transformáciu nahradila DWT, ktorá umožňuje dosiahnuť vyššieho stupňa kompresie. [5]

1.4.4 RAW

Formát RAW sa používa pri profesionálnej fotografii. Ide o dáta, ktoré sú zobrazené priamo zo snímača zariadenia. Ich hlavnou výhodou je oveľa širšia možnosť úprav, nedochádza ku stratám – všetky algoritmy na spracovanie a prevod obrazu sú ponechané až na úpravu v počítači, dáta sú ukladané s takou farebnou hĺbkou, s akou sú zo snímané. Oproti formátu TIFF nezaberajú toľko miesta. Nevýhoda je, že každý výrobca používa iný čip, a preto nie každý program je schopný načítať akýkoľvek súbor. Preto je potrebné v niektorých prípadoch použiť softvér originál od výrobcu.

1.4.5 GIF

Je výtvorom CompuServe Inc. a slúži k ukladaniu viacerých bitmapových obrázkov v jednom súbore pre výmenu medzi rôznymi platformami a systémami. Pokiaľ to berieme podľa počtu súborov za dobu jeho existencie, ide pravdepodobne o najrozšírenejší formát pre ukládanie multibitovej grafiky a obrazových dát.

Prevažná väčšina súborov GIF obsahuje 16 alebo 256 farebné obrázky takmer vo fotografickej kvalite. Často sa však do neho ukladajú obrázky v škále šedej.

Najväčšou nevýhodou formátu GIF je, že využíva komprimačný algoritmus LZW, ktorý nie je zadarmo. Medzi ďalšie nevýhody patrí, že podporuje maximálne 8bit farby. [4]

1.4.6 PNG

Vznikol ako alternatívna náhrada formátu GIF, ktorý je patentovo chránený. Vznikol pod záštitou konzorcia W3C a nie je patentovo chránený. Bol navrhnutý s cieľom vytvoriť jednoduchý formát, ktorý sa dá ľahko implementovať, ktorý by bol plne prenositeľný a ktorý dosahuje alebo prekračuje všetky schopnosti formátu GIF. Používa sa hlavne pre jeho bezstratovú kompresiu a zobrazovanie obrázkov na internete. Používa kompresný algoritmus deflate/inflate s posuvným oknom o veľkosti 32 768 bytov. Kompresia Deflate je 100% bezstratová, ktorej autorom je Phil Katz a používa sa aj v utilite pkzip pre archiváciu súborov hlavne pre jej rýchlosť, dobrú zdokumentovateľnosť, je voľne dostupná a podporovaná na veľkom počte operačných platforiem. Jeho hlavné výhody oproti GIF sú možnosť ukladať obrázky v 48 bitovej farebnej hĺbke, alfa kanál pre transparentnú masku, detekcia poškodenia súboru a obrazová gama informácia podporuje automatické jasovo/kontrastové prispôbenie. [6]

2 GEOMETRICKÉ TRANSFORMÁCIE OBRAZU

Geometrické transformácie sa používajú k úprave alebo korekcii obrázkov. Patrí sem posunutie, otočenie, zmena veľkosti, skosenie a v špecifických prípadoch aj zložitejšia metóda nazývaná “warping“. Základné transformácie sú v podstate jednoduché matematické operácie. Avšak pokiaľ potrebujeme napríklad otočiť obrázok s rozlíšením 100x100 okolo iného bodu ako je počiatok súradníc, musíme najskôr posunúť bod okolo ktorého sa bude obrázok otáčať (počítať sto tisíc operácií), otočiť obrázok (počítať sto tisíc operácií) a posunúť bod naspäť (počítať sto tisíc operácií). Keby sme to mali všetko robiť postupne trvalo by to veľmi dlho. Preto sa transformácie skladajú. To prebieha v homogénnych súradniciach, pomocou matíc. V našom prípade budeme preto celú operáciu robiť len raz, čiže spravíme sto tisíc operácií. [7],[8]

2.1 Posunutie

Posunutie znamená premiestnenie objektu z pôvodnej pozície do novej.

Algebraický zápis pre posunutie 2D obrazu je

$$x' = x + t_x \quad (1)$$

$$y' = y + t_y$$

S použitím matíc

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2)$$

2.2 Otočenie

Otočenie je transformácia bodu okolo pevného bodu po kruhovej dráhe. Určuje ju uhol otočenia a stred otočenia.

Algebraický zápis pre otočenie 2D obrazu je

$$x' = x \cos(\beta) - y \sin(\beta) \quad (3)$$

$$y' = x \sin(\beta) + y \cos(\beta)$$

S použitím matic

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4)$$

2.3 Zmena mierky

Pri zmene mierky dochádza k zmene veľkosti objektu a jeho polohy v smere súradnicových osí. Ak je koeficient pomeru novej dĺžky ku starej väčší ako 1, objekt sa predĺži, ak je koeficient menší ako 1, tak sa výsledný objekt skrúti.

Algebraický zápis pre zmenu mierky 2D obrazu je

$$x' = s_x x \quad (5)$$

$$y' = s_y y$$

S použitím matic

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (6)$$

2.4 Skosenie

Pri použití skosenia dochádza k zmene jednej súradnice, pričom druhá ostáva zachovaná.

Algebraický zápis pre skosenie 2D obrazu pre os x je

$$x' = x - y \tan(\beta) \quad (7)$$

$$y' = y$$

S použitím matic

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \tan(\beta) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (8)$$

2.5 Odraz

Zrkadlový odraz, inak nazývaný súmernosť, je v podstate zmenou mierky a to tak, že os podľa ktorej chceme odraz previesť, prehodíme znamienko súradníc.

Algebraický zápis pre odraz 2D obrazu podľa osi x je

$$x' = -x \quad (9)$$

$$y' = y$$

S použitím matíc

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (10)$$

Algebraický zápis pre odraz 2D obrazu podľa osi y je

$$x' = x \quad (11)$$

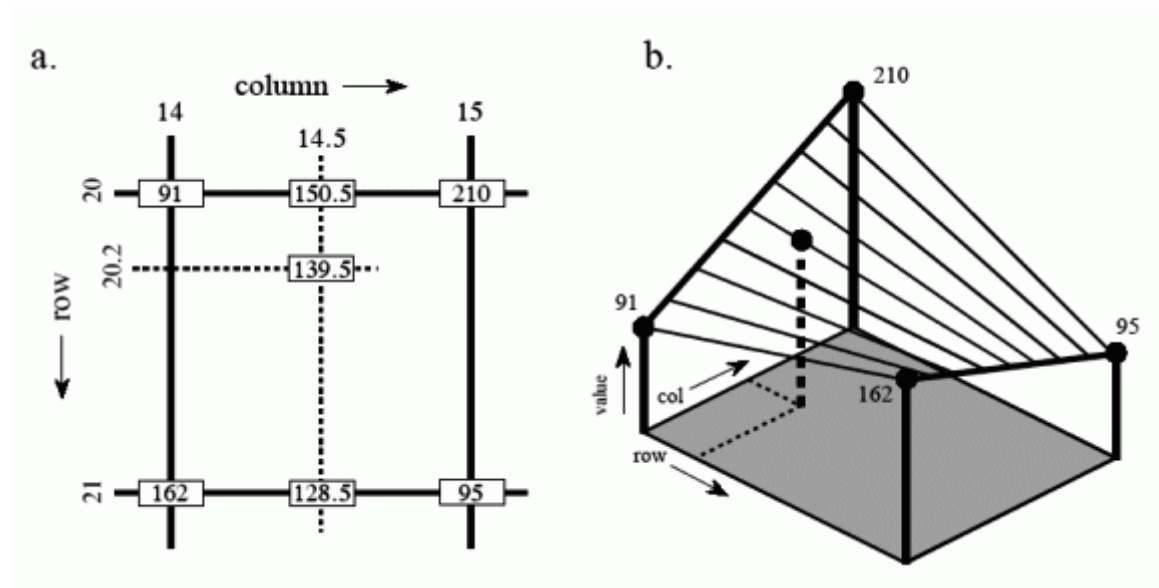
$$y' = -y$$

S použitím matíc

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (12)$$

2.6 Warping

Warping patrí medzi zložitejšie deformácie obrazu. Ide o takzvané pokrútenie alebo zvlhnenie objektu, kde máme zadanú množinu referenčných bodov medzi pôvodným a výstupným obrazom. Algoritmy warpingu môžeme rozdeliť do dvoch kategórií. Jednoučelové algoritmy, ktoré sa zadávajú jednoducho niekoľkými parametrami a majú jediný výstup. Druhá skupina algoritmov zahŕňa obecné metódy, ktoré umožňujú prakticky ľubovoľnú transformáciu. Prvá z nich je založená na položení virtuálnej siete na obraz. Tá svojou zmenou určuje warping obrazu. Sieťový warping je vhodný pre globálne zmeny v obraze a nie je s ním jednoduché robiť malé lokálne operácie. Druhá metóda je založená na vkladanie takzvaných magnetov do obrazu. Zmena ich polohy určuje transformáciu. Sú vhodné pre malé, lokálne transformácie v obraze. [9]



Obr. 7. Princíp fungovania warpingu

3 TRANSFORMÁCIE FARIEB OBRAZU

Transformácie farieb majú kľúčovú úlohu pri tvorbe tieňovaných vizualizácií priestorových dát a to tak, že určujú akými spôsobmi je pôvodná farba modifikovaná. Zlepšujú sa tak vizuálne vlastnosti upravovaného obrázku, ktorý je vhodný pre ďalšie použitie, alebo tlač. Pri transformácií by mal ostať zachovaný odtieň pôvodných farieb. To znamená, že meniť by sa mala len ich intenzita.

3.1 Odtieň šedej

Najjednoduchším spôsobom redukcie farieb obrazu je prepočet na odtiene šedej. Farebné odtiene môžu byť na odtiene šedej prevedené veľmi jednoducho podľa vzorca :

$$I = 0,266 * R + 0,587 * G + 0,114 * B \quad (13)$$

kde: I je výsledná intenzita (úroveň šedej) a R , G , B sú základné farebné zložky pôvodnej farby. Prevod obrazu na úrovne šedej spočíva v postupnom prepočítaní všetkých bodov obrazu podľa vyššie uvedeného vzorca. [10]

3.2 Negatív

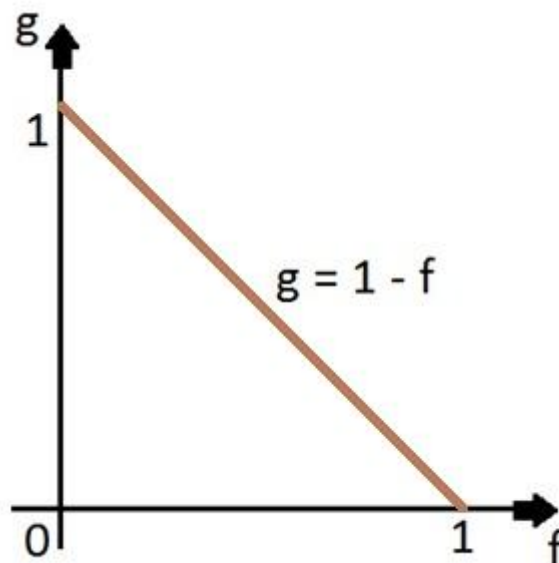
Jedná sa o najjednoduchšiu transformáciu obrazu. Mení sa intenzita jasů každého bodu v obraze podľa vzorca

$$g(x,y) = 1 - f(x,y) \quad (14)$$

pre obraz v stupňoch šedi a

$$g(x,y) = L - 1 - f(x,y) \quad (15)$$

Pre farebný obraz, kde L je počet jasových úrovní. Napríklad u 8 bitového farebného modelu je $L=256$. [10]



Obr. 8. Transformačná krivka hodnôt jasu pre negatív

3.3 Gama

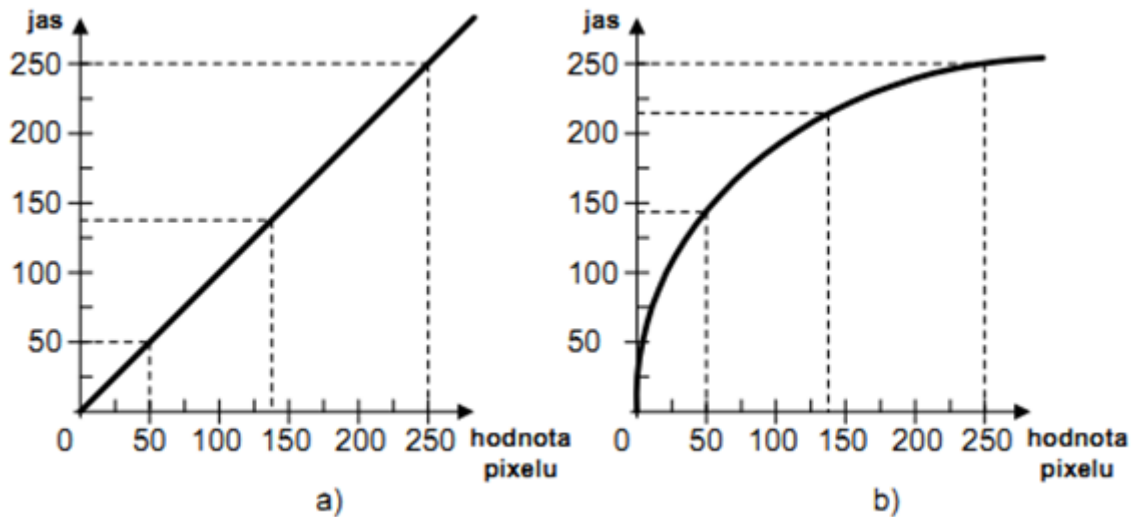
Korekcia gamy je najčastejšie používanou nelineárnou funkciou. V súčasnosti už takmer nepoužívané CRT obrazovky fungujú na princípe toku elektrónov, ktoré vybudia fosfor, ktorý začne žiariť určitou farbou a vysvieti na obrazovke bod. Závislosť medzi jasom (intenzitou) a hodnotou pixelu nie je lineárna. To je spôsobené nelineárnym chovaním fosforu pri vybudení elektrónovým lúčom.

Gama faktor: odchýlka hodnoty pixelu od reálneho jasu pixelu

Gama korekcia: slúži na opravu (odstránenie) nelinearít zobrazovacieho systému, čiže úprava jasu na obrazovke. Využíva sa tiež aj na opravu zlej expozície, kde niektoré miesta sú príliš tmavé alebo naopak príliš svetlé. [1]

Korekcia gamy sa odstraňuje pomocou vzorca:

$$i_n = i^{\frac{1}{\gamma}} \quad (16)$$



Obr. 9. Závislosť hodnoty bodu a jeho jasú:

a) Ideálna

b) Reálna

Hodnota γ závisí na type obrazovky. Väčšina televízorov v súčasnej dobe má túto konštantu už predom nastavenú, a nemusíme sa teda o nič starať. Kvalitnejšie monitory majú zabudované čidlo, vďaka ktorému sa vedú sami kalibrovať, nakoľko hodnota konštanty závisí aj na teplote. Hodnota γ sa vo väčšine prípadov pohybuje v rozmedzí 1,8 až $2,5 \pm 0,3$.

3.4 Jas

Patrí medzi najjednoduchší spôsob zlepšenia vzhľadu zle exponovaného obrázku. Vo väčšine prípadov stačí práve len úprava jasú. Zmeny sa docielia transformáciou pôvodných jasových úrovní na iné. Používajú sa štandardné aritmetické operácie.

Základný vzorec pre výpočet je

$$g(x, y) = T[f(x, y)], \quad (17)$$

kde $g(x, y)$ je výstupný obraz, T je transformácia a $f(x, y)$ je vstupný obraz. Pri zložitejších transformáciách sa výsledný obraz počíta aj z okolia vstupného obrazu ako napríklad ostrenie obrazu pomocou nelineárnej metódy alebo pri vyhladzovaní. [1]

3.5 Kontrast

Používa sa v prípade, pokiaľ je rozdiel medzi rôznymi úrovňami jasů v obraze príliš malý. Obrázok je v tomto prípade nekontrastný a pre nás dôležité dáta sa nachádzajú v príliš úzkom pásme jasových úrovní. Predpokladajme, že minimálna hodnota jasů vyskytujúca sa v obraze je f_1 a maximálna hodnota je f_2 . Hodnoty z ich intervalu je nutné previesť na celý zobraziteľný rozsah hodnôt. Transformáciu realizujeme tak, že od danej hodnoty jasů f odpočítame minimálnu hodnotu nachádzajúcu sa v obraze a vynásobíme koeficientom rozťahnutia $1/(f_2 - f_1)$. Výsledný vzorec je:

$$g = \frac{f - f_1}{f_2 - f_1} \quad (18)$$

Alternatívne, ale menej často používaný je nelineárny vzťah:

$$g = \frac{1}{1 + \left(\frac{m}{f + \varepsilon}\right)^E} \quad (19)$$

kde f sú vstupné hodnoty, hodnota m určuje stred rozťahnutia, hodnota E určuje strmosť krivky a konštanta ε zabraňuje deleniu nulou.[10]

3.6 Interpolácia

Medzi časté požiadavky pri úprave obrázkov je zmena rozlíšení. Obrázok môžeme buď zmenšiť, alebo zväčšiť. Pre každý bod nového obrázku je nutné vypočítať jeho polohu a hodnotu podľa transformačnej funkcie. Existuje viacero metód pre výpočet. Najjednoduchšou je interpolácia najbližším susedom. Hodnota bodu je daná hodnotou bodu v pôvodnom obraze, ktorý je k nemu najbližšie. Táto metóda nie je vždy uspokojivá a spôsobuje stratu detailov. Najčastejšie používanou metódou je bilineárna interpolácia.

Pri bilineárnej interpolácii je výsledná hodnota jasů daná hodnotami jasů a polohy štyroch susedných bodov:

$$f(i, j), f(i + 1, j), f(i, j + 1), f(i + 1, j + 1), \quad (20)$$

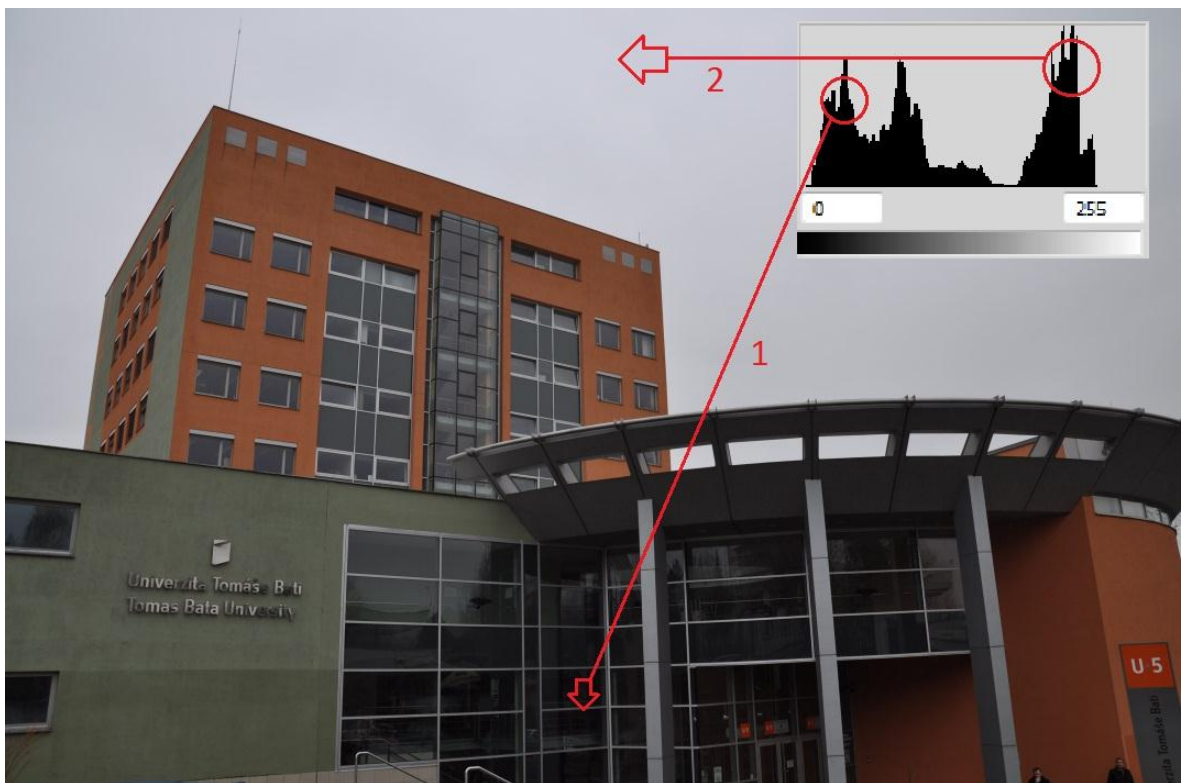
kde (i, j) sú súradnice bodu a f má význam hodnoty jasů. Táto metóda je výpočtovo jednoduchá a vo väčšine prípadov postačuje.

V prípade potreby na oveľa väčšiu kvalitu obrazu, je možné použiť zložitejšiu bikubickú interpoláciu. V tomto prípade sa pre výpočet hodnoty výsledného bodu používa viacej susedných bodov. [1]

3.7 Histogram

Histogram patří mezi nejefektivnější nástroje kontroly kvality a úpravu snímky. Zachytává rozdělení stupňov svetlosti obrazových dát v grafickej forme. Matematicky sa dá histogram vyjadriť ako vektor absolútnej početnosti hodnôt zastúpených v obraze. Stĺpce, ktoré sa nachádzajú nad každou hodnotou, hovoria formou diagramu o tom, ako často sa určitá intenzita pixelu nachádza v obrázku. Keď sa pruhy dotýkajú, histogram naberá tvar pohoria. Čím viac je niektorý odtieň v obrázku zastúpený, tým vyšší je jeho pruh. Pokiaľ niektorý odtieň chýba, je toto miesto prázdne.

Pokiaľ je pôvodný obrázok zložený z jasových zložiek, je histogram jednorozmerný vektor a reprezentuje ho jeden histogram. Pri farebných obrázkoch odpovedá každej farbe vlastný histogram. Tie sa môžu navzájom výrazne líšiť. [10],[11]



Obr. 10. Histogram obrázku

Na obrázku (Obr. 10) môžeme vidieť názornú ukážku rozloženia histogramu, kde v ľavej časti sa nachádzajú tmavé miesta a v pravej časti svetlé miesta.

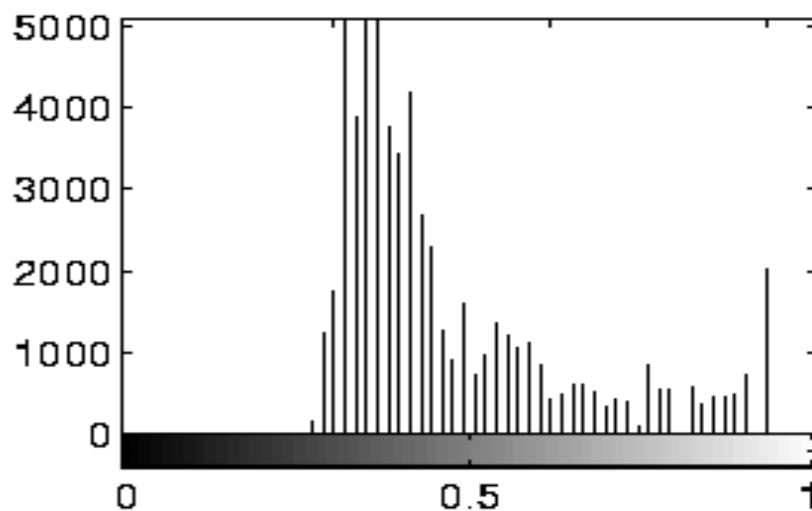
Histogram je neoceniteľným pomocníkom pri analyzovaní fotografií. Môžeme z neho okamžite zistiť, či je obrázok správne exponovaný, informácie o kontraste, o dynamickom rozsahu obrázku a mnoho ďalšieho. Nie je to však vždy ľahké. Pomocou histogramu dokážeme taktiež určiť, či bol obrázok niekedy upravovaný a či bol na uloženie zvolený vhodný formát. Ak použijeme na uloženie obrázku kompresný formát, zmení to aj výsledný histogram.

Základne charakteristiky histogramu

- Pokiaľ je graf šedej stupnice väčšinou na ľavej strane, obrázok je príliš tmavý
- Pokiaľ je graf šedej stupnice väčšinou na pravej strane, obrázok je príliš svetlý
- Pokiaľ graf šedej stupnice nie je dostatočne rozložený, je pravdepodobne potrebné zvýšiť kontrast
- Pokiaľ je graf v úzkej oblasti, fotografia pravdepodobne nemá dostatok detailov
- Pokiaľ je graf rovnomerné rozložený, fotografia má pravdepodobne vyrovnanú kompozíciu a dostatok detailov [13]

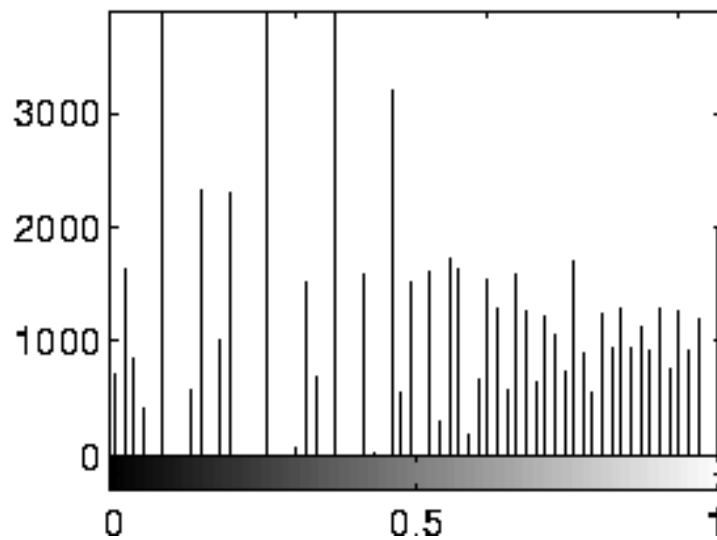
3.7.1 Vyrovnávanie histogramu

Pokiaľ máme nevhodne exponovaný obraz, nerovnomerne rozložené zložky histogramu, snažíme sa ho upraviť tak, aby bolo toto rozloženie rovnomerné (úrovne jasu by sa vyskytovali pravidelne). Metóda, ktorou môžeme tento výsledok doceliť, sa nazýva metóda vyrovnania histogramu (ekvalizácia histogramu).



Obr. 11. Príklad na nevyrovnaný histogram

Tmavé aj svetlé odtiene majú veľmi nízky kontrast. Avšak bežným zvyšovaním kontrastu ho paradoxne ešte viacej znížime. Obráz ako celok má totiž kontrast veľmi vysoký a jeho zvyšovaním posúvame tmavé tóny do ešte tmavších a svetlé do ešte svetlejších. Nepomôže ani zmena jasú, lebo pri znižovaní orezávame tmavé farby a pri zvyšovaní orezávame svetlé farby. Jediná možnosť ako vyriešiť tento problém je už v spomínanej metóde vyrovnania histogramu, čo zabezpečí rovnomerne zastúpenie tmavých, stredných a svetlých farebných odtieňov. Príklad vyrovnaného histogramu vidíme na obrázku (Obr. 12). Týmto dosiahneme kontrastnejší obraz a zvýraznenie detailov. [14]



Obr. 12. Príklad na vyrovnaný histogram

$$g_{(k)} = \frac{q_k - q_0}{x * y} * \sum_{j=k_i}^{i=k} L(i) \quad (21)$$

$L(i)$ predstavuje histogram počtosti jasov i vo vstupnom obrázku (počet bodov s rovnakou intenzitou), $x * y$ je celkový počet bodov v obrázku, $\langle q_0, q_k \rangle$ interval výstupných jasov, a k_i je najnižšia intenzita vstupného obrázku.

4 FILTRÁCIA OBRAZU

Pod pojmom filtrácia obrazu sa rozumie práca s digitálnym obrazom, pri ktorej dochádza k zvýrazneniu alebo potlačeniu určitých informácií. Vo filtrácií sa rozlišuje celá rada metód pre úpravu. Pri úprave obrazu sa využívajú dva hlavné spôsoby a to priestorové a frekvenčné. Priestorové metódy pracujú priamo s obrazom $f(x,y)$, alebo jeho časťou, ktorú transformujú na výstupný obraz $g(x,y)$ podľa vzorca

$$g(x,y) = T[f(x,y)]. \quad (22)$$

Pri frekvenčných metódach sa využíva konvolúcia obrazu.

4.1 Odstraňovanie šumu

Patrí medzi často používanú a užitočnú funkciu pri úpravách obrazu. Šum je definovaný ako nová informácia, ktorá bola k pôvodnej informácii pridaná snímacím zariadením alebo pri transfere dát. Vo väčšine prípadov sa prejavuje v obraze ako zrnenie. Šum rozdeľujeme do dvoch kategórií a to podľa spôsobu, ako bol pridaný do pôvodného obrazu a to na aditívny a multiplikatívny. V praxi existuje viac druhov šumu a vždy je potrebné zvážiť, ktorá metóda sa použije na jeho odstránenie, nakoľko výsledky môžu byť veľmi rozdielne. Hlavným problémom je identifikovať, či ide o šum alebo nie. Každý filter nejakým spôsobom vyhodnotí na základe okolia a veľkosti zmeny hodnoty bodu, či ide o šum. Filtre pracujú buď na princípe konvolúcie alebo lokálnej štatistiky okolia.

4.1.1 Lineárne metódy

4.1.1.1 *Spriemerovanie*

Filtrácia spriemerovaním je jednoduchým príkladom vyhadzovania obrazu pomocou konvolúcie, kde sa na obraz použije konvolučná matica. Ide o najjednoduchšiu metódu vyhladzovania šumu v obraze, pri ktorej sa spoliehame na značnú redundanciu dát v obraze. Vďaka tomu môžeme predpokladať, že susedné body majú rovnaký, prípadne podobný jas. Každému bodu obrazu sa jeho jas nahradí aritmetickým priemerom jasom susedných bodov. To spôsobí, že ostré hrany alebo šum sa v obraze rozmažú, čo je najväčším nedostatkom tejto metódy. Opakovaným použitím filtru dochádza k čoraz väčšiemu rozmazaniu, až nakoniec výsledný obraz dosiahne jednu farbu, ktorá je priemerom všetkých hodnôt. Vo väčšine prípadov sa za susedov považujú body zo štvorcového nepárneho okolia bodu (3x3, 5x5, 7x7,... atď.). [10]

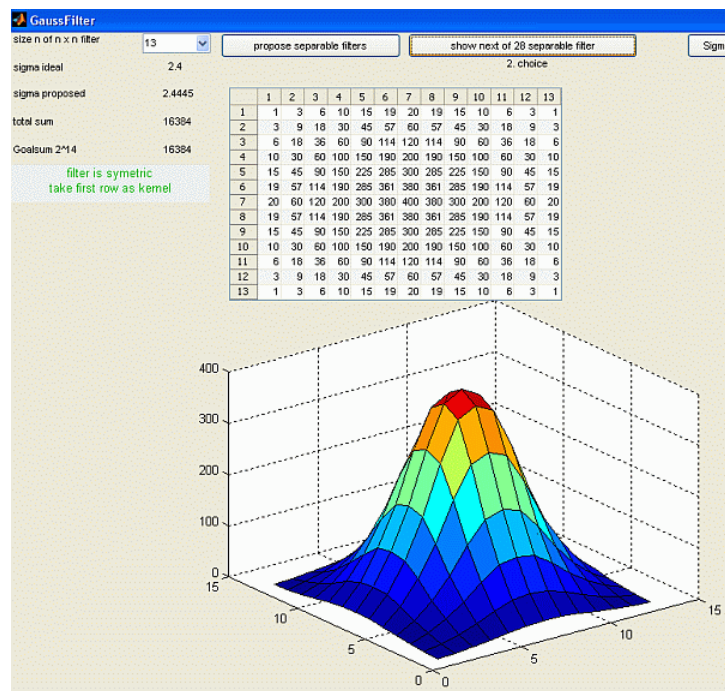
$$g(x, y) = \frac{1}{M} \sum_{(i,j) \in O} f(i, j) \quad (23)$$

Konvolučná maska 3x3 vyzera nasledovne

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (24)$$

4.1.1.2 Gaussové vyhladzovanie

Ide o podobný princíp ako vyhladzovanie pomocou priemerovania. Rozdiel je v konvulčnej matici. Tá ma nastavené koeficienty tak, že tie ktoré sú bližšie k strede majú vyššiu váhu, aby odpovedali gaussovej krivke.



Obr. 13. Príklad na gaussovo rozdelenie

Príklad matice 3x3

$$h = \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (25)$$

Na obrázku (Obr. 11) vidíme tvar hustoty pravdepodobnosti $h(x,y)$ pre dvojrozmerný náhodný vektor s Gaussovým rozdelením.

Dvojrozmerné Gaussovo rozdelenie je dané vzťahom

$$h_i = \frac{\exp\left\{-\frac{(G_j - m)^2}{2\sigma^2}\right\}}{\sigma\sqrt{2\pi}} \quad (26)$$

$$-\infty < G_j < \infty$$

Funkcia hustoty G_j je definovaná pre všetky body v rovine od $-\infty$ do $+\infty$. Od parametru σ závisí veľkosť masky. Od určitej dostatočne malej hodnoty nie sú ostatné hodnoty považované za významné.

4.1.2 Nelineárne metódy

Nelineárne metódy odstraňovania šumu pracujú na princípe lokálnej štatistiky v okolí bodu, kde sa snažia nájsť tú časť, do ktorej reprezentovaný bod patrí. Tú potom dosadia na miesto upravovaného bodu. Ich výhoda oproti lineárnym filtrom je, že výsledný obraz nie je tak rozmazaný.

4.1.2.1 Medián

Je najčastejšie používanou metódou. Medián je prostredný prvok v usporiadanej množine hodnôt. Hodnoty získame z okolia bodu. Tie zoradíme a hodnotou v strede nahradíme jas filtrovaného bodu. Táto metóda veľmi dobre odstraňuje šum impulzného (bodového) charakteru. Nevýhodou je narušovanie tenkých čiar. Okolie bodu, ktoré sa používa, nemusí byť len štvorcové. Niekedy je vhodné použiť aj okolie v tvare kríža, ktoré neporuší diagonálne čiary, prípadne v tvare písmena X, ktoré neporuší vodorovné čiary.

4.1.2.2 Rotujúca maska

Vychádza sa z toho, že sa bude spriemerovať iba tá časť okolia bodu, ku ktorej bod pravdepodobne prináleží. Z toho vyplýva, že body ležiace na hrane sa nebudú počítať do spriemerovania. Okolie musí byť homogénne. Top sa dá vyšetriť na základe rozptylu bodov.

4.2 Detekcia hrán

Keď chceme nájsť hranu v obraze, musíme hľadať zmenu jasu. Detekcia hrán v obraze je postavená na nájdení miesta, kde je táto zmena najvýraznejšia. Možnosťou, ako sa dajú jednoducho tieto zmeny nájsť, je prvá a druhá derivácia intenzity jasu v obraze.

Pri aplikovaní prvej derivácie obrazu v smere x a y dostávame gradient, ktorý je definovaný ako vektor

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (27)$$

Veľkosť tohto vektoru je

$$|\nabla f| = \sqrt{G_x^2 + G_y^2} = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad (28)$$

Pre zjednodušenie výpočtu sa táto hodnota niekedy nahradzovala vynechaním odmocniny hodnotou $G_x^2 + G_y^2$, alebo pomocou súčtu absolútnej hodnoty $|G_x| + |G_y|$. Tieto náhrady sa chovajú ako derivácie, čiže sú nulové v oblastiach s konštantnou intenzitou jasou a ich hodnoty závisia na zmene. Tento gradient môžeme využiť ako informáciu pri hľadaní hrany. [10]

Taktiež je možné ho využiť pri ostrení obrazu a to tak, že obraz upravíme, aby v ňom boli strmšie hrany.

V praxi existuje niekoľko operátorov na hľadanie hrán založených na princípe gradientu.

4.2.1 Pomocou prvej derivácie

4.2.1.1 Robertsov operátor

Najjednoduchší a najstarší operátor. Používa okolie len 2×2 . Nevýhodou Robertsovho operátora je veľká citlivosť na šum, nakoľko okolie použité pre aproximáciu je veľmi malé.

Veľkosť gradientu aproximuje podľa vzťahu:

$$|\nabla f(x, y)| \approx |f(x + 1, y + 1) - f(x, y)| + |f(x, y + 1) - f(x + 1, y)| \quad (29)$$

Robertsov operátor v smere x :

Tab. 1. Robertsov operátor

v smere x

-1	1
-1	1

$$h_1 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad h_2 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad (30)$$

4.2.1.2 Prewittovej operátor

Gradient sa odhaduje v okolí masky 3x3 pre osem smerov. Vybraná je tá, ktorá má najväčší modul gradientu. Robí sa rozdiel hodnôt

$$f(x + 1, y) - f(x - 1, y), \quad (31)$$

alebo pre y je to

$$f(x, y + 1) - f(x, y - 1). \quad (32)$$

Výsledná maska má tvar:

Tab. 2. Maska Prewittovej

-1	0	1	-1	-1	-1
-1	0	1	0	0	0
-1	0	1	1	1	1

4.2.1.3 Sobelov operátor

Jeho koeficienty sú volené tak, aby zdôrazňovali hodnoty bližšie k strede masky. Maky môžu rotovať po 45° a počítať smerové derivácie nie len v smere x a y. V praxi sa však často používa pri detekcii vodorovných a zvislých čiar. Pre toto použitie postačujú nasledovné masky:

Tab. 3. Maska Sobelovho operátora v troch smeroch

-1	0	1	-1	-2	-1	0	-1	-2
-2	0	2	0	0	0	1	0	-1
-1	0	1	1	2	1	2	1	0

4.2.1.4 Robinsonov operátor

Prvé tri masky sú dané vzťahom:

Tab. 4. Maska Robinsonovho operátora

-1	1	1	1	1	1	1	1	1
-1	2	1	1	-2	1	-1	-2	1
-1	1	1	-1	-1	-1	-1	-1	1

Samotný operátor je podobný ako Prewittovej operátor.

4.2.1.5 Kirschov operátor

Jeho konvolučné masky majú tvar

Tab. 5. Maska Kirschovho operátora

3	3	3	3	3	3	-5	3	3
-5	0	3	3	0	3	-5	0	3
-5	-5	3	-5	-5	-5	-5	3	3

4.2.2 Pomocou druhej derivácie

Pomocou druhej derivácie dostaneme rýchlosť zmeny hodnoty jasu. Prejavuje sa hlavne na strmých a izolovaných hranách. Môžeme ju taktiež použiť aj na detekciu izolovaných bodov. Nevýhodou je, že pri tejto operácii sa do veľkej miery zvyrazňuje aj šum. [15]

4.2.2.1 Laplaceov operátor

Je vhodný k detekcii izolovaných bodov. Jeho operátor sa označuje ∇^2 . Operátor sa nemení v závislosti od pootočenia o násobky 45° , ale udáva len veľkosť hrany, nie jej smer.

Výsledok aplikácie Laplaceovho operátora na funkciu $f(x,y)$ je:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \quad (33)$$

Nevýhodou tohto operátora je, že je veľmi citlivý na šum.

Tab. 6. Maska Laplaceovho operátora

0	-1	0	0	1	0
-1	4	-1	1	4	1
0	-1	0	0	1	0

Tab. 7. Maska Laplacian of Gaussian

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	24	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

5 VÝVOJOVÉ PROSTREDIE .NET A JAZYK C#

Microsoft Visual C# je výkonný, ale pritom jednoduchý jazyk zameraný predovšetkým na vývojárov aplikácií na platforme .NET Framework. Zdedil veľké množstvo toho najlepšieho z jazykov C++ a Visual Basic. Výsledkom je čistý a logický jazyk.

Platforma .NET bola predstavená roku 2000. Najskôr vyšla ako beta verzia pod názvom Next Generation Windows Services.

Jazyk C# nemôžeme chápať izolovane, ale je nutné o ňom uvažovať súbežne s platformou .NET Framework. Kompilátor jazyka C# je špeciálne zameraný na platformu .NET, čo znamená, že kompletný kód napísaný v jazyku C# bude vždy spúšťaný pomocou nej.

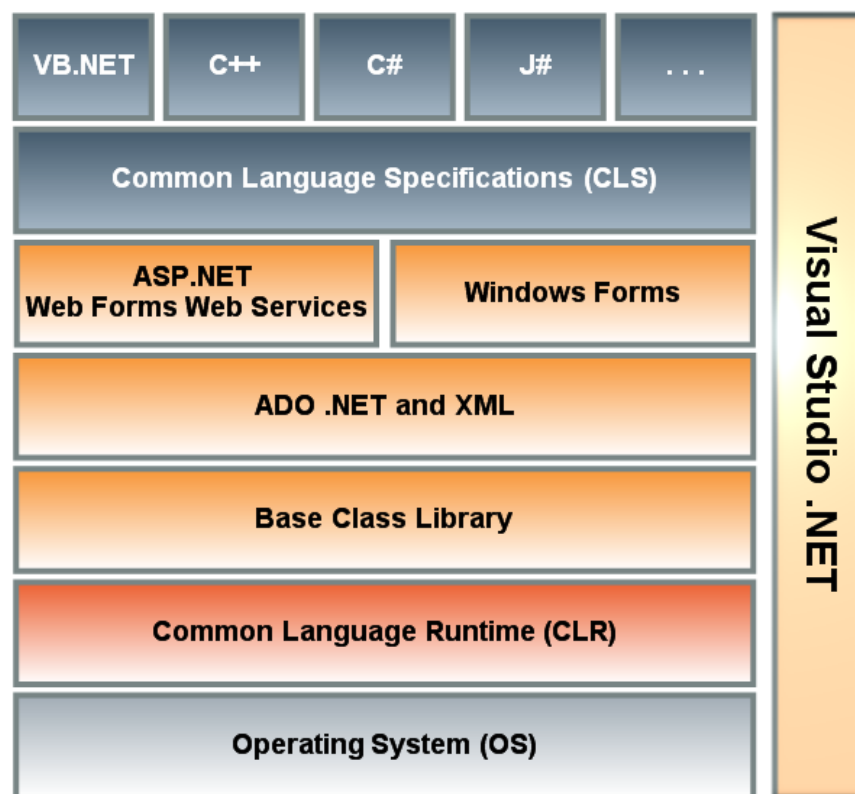
Jazyk C# je samostatným jazykom. Je síce navrhnutý tak, aby generoval kód určený pre prostredie .NET, ale nie je sám o sebe súčasťou platformy .NET. Platforma .NET poskytuje aj niektoré funkcie, ktoré jazyk C# nepodporuje a naopak existujú aj funkcie jazyka C# ktoré nie sú podporované platformou .NET. [16]

5.1 Architektúra .NET

V súčasnej dobe sa vyvíjajú aplikácie, ktoré svojim rozsahom a komplexnosťou prevyšujú schopnosti jedného človeka. Preto sa na vývoji takýchto aplikácií zúčastňuje viacero programátorov. Pri svojej práci využívajú časti (komponenty) svojich aplikácií, alebo iných firiem. Programátor sa nemôže spoliehať len sám na seba. Prvým nástrojom pre modulárny vývoj aplikácií na platforme Microsoft Windows boli dynamické knižnice. Začiatkom 90. rokov boli vytvárané samostatné aplikácie s veľmi malou schopnosťou vzájomnej komunikácie.

Neskôr firma Microsoft uviedla technológiu COM (Component Object Model), ktorá tento nedostatok odstránila. Výhodou komponentovej technológie je jej jazyková neutralita v binárnej forme. Pre každú komponentu bolo definované rozhranie, ktoré sprostredkováva komunikáciu medzi klientom a príslušnou komponentou. Časom sa však ukázalo, že i táto technológia má svoje obmedzenia. V súčasnej dobe je modulárna technológia veľmi populárna a používaná. Používané komponenty sú väčšinou malé, jednoduché a nepotrebujú odbornú znalosť riešenia problematiky. Ich hlavnou nevýhodou je, že zakrývajú vnútornú štruktúru a jedinú čo ich popisuje, je ich rozhranie. To znemožňuje dedičnosť na úrovni zdrojových kódov.

.NET Framework funguje ako substrát, na ktorom sa môže pestovať software. Jeho jadro je založené na princípoch OOP (Objektovo Orientovaného Programovania) a všetky jeho základné služby sprístupňuje širokej škále programovacích jazykov. Podporuje triedy, metódy, konštruktory, vlastnosti, polymorfizmus, udalosti, vlákna, atď. To všetko znamená, že nezáleží v akom programovacom jazyku komponenty vytvárame, prípadne aké komponenty používame. .NET Framework taktiež rieši niektoré problémy súvisiace s bezpečnosťou. Medzi ďalšie problémy, ktoré rieši, je napríklad nasadzovanie a inštalácia aplikácií označované ako DLL Hell. Technológia .NET nerobí rozdiely medzi 32 a 64 bit. systémami. Programový kód je spoločný pre všetky platformy. Prípadné rozdiely medzi koncovou architektúrou sa riešia až pri jeho preklade. [16], [21]



Obr. 14. Architektúra platformy .NET Framework [21]

Na obrázku (Obr. 14) môžeme vidieť štruktúru architektúry .NET Framework. Na najnižšej úrovni sa nachádza operačný systém na ktorom beží celé rozhranie. Ako prvá vrstva sa nachádza CLR (Common Language Runtime). Tá realizuje základnú infraštruktúru, nad ktorou je framework vybudovaný. Nad CLR sa nachádzajú hierarchicky umiestnené knižnice, ktoré sú rozdelené do menných priestorov. Základom je knižnica

Base Class Library. Nad ňou sa nachádza knižnica pre prístup k dátam a prácu s XML súbormi. Poslednou vrstvou knižníc je sada uľahčujúca prácu s užívateľským rozhraním. Je rozdelená do dvoch skupín. Prvá pre uľahčenie vytvárania webových aplikácií a vytváranie klasických oknových aplikácií. Poslednou skupinou je nelimitovaná množina programovacích jazykov. Ich základné vlastnosti definuje CLS (Common Language Specifications). V súčasnej dobe sú firmou Microsoft podporované jazyky Visual Basic, C++, C#, J#, F#, atď. Táto množina nie je uzavretá. Ktorýkoľvek výrobca ju môže rozšíriť. Celá architektúra je podporovaná vývojovým nástrojom Visual Studio.

5.2 Programovací jazyk C#

Jazyk C# bol vyvinutý firmou Microsoft. Bol predstavený spolu s vývojovým prostredím .NET. Celkovo vychádza z programovacieho jazyka C++. V mnohých ohľadoch je veľmi podobný jazyku Java.

Základné charakteristiky jazyka C#:

- Čisto objektovo orientovaný programovací jazyk
- Obsahuje natívnu podporu komponentového programovania
- Obsahuje jednoduchú dedičnosť s možnosťou viacejnásobnej implementácií rozhrania
- Popri členských dát a metód pridáva vlastnosti a udalosti
- Automatická správa pamäti – o uvoľňovanie sa stará garbage collector
- Podpora zapracovávania chýb pomocou výnimiek
- Zaisťuje typovú bezpečnosť a podporuje riadenie verzií
- Podpora atribútového programovania
- Zaisťuje spätnú kompatibilitu s aktuálnym kódom ako aj na binárnej tak aj na zdrojovej úrovni

Väčšina uvedených vlastností vychádza priamo z funkcionality vývojového rámca .NET.

Prekladač jazyka C# je case sensitive, čo znamená, že rozlišuje veľké a malé písmená, podobne ako jazyk C++. Mená balíkov, tried, rozhraní a väčšina ďalších položiek začínajú veľkým písmenom. Malým písmenom začínajú lokálne parametre, premenné, atribúty, atď.

[17]

II. PRAKTICKÁ ČASŤ

6 IMPLEMENTÁCIA ALGORITMOV VO VÝVOJOVOM PROSTREDÍ

6.1 Možnosti úpravy v jazyku C#

6.1.1 Trieda Bitmap a Image

Prostredie .NET poskytuje viacej možností pre prácu s grafikou. Obsahuje metódy pre základné úpravy obrázkov. Výhodou je jednoduchá aplikácia algoritmu. Medzi nevýhody patrí nemožnosť pozmeniť hodnoty výpočtu a funkciu a rýchlosť. Pre načítavanie a vkladanie bodov slúžia metódy *GetPixel()* a *SetPixel()*. Výhoda je jednoduché použitie, ale najväčšia nevýhoda je rýchlosť. V porovnaní s inými možnosťami sa jedná až o niekoľkonásobne pomalšiu metódu manipulácie s obrazovými dátami.

6.1.2 Trieda ColorMatrix

Trieda *ColorMatrix* patrí do menného priestoru *System.Drawing.Imaging*. Pomocou nej môžeme veľmi pružne meniť informácie o farbách. Inštancia triedy *ColorMatrix* umožňuje transformáciu hodnôt ARGB každého pixelu obrázku jednotlivivo. Nevýhodou tejto triedy je na prvý pohľad vysoká zložitosť a veľmi slabá dokumentácia platformy .NET. Pre transformáciu jednotlivých pixelov obrázku je nutné vytvoriť inštanciu triedy *ColorMatrix*. Takýto prvok obsahuje maticu o rozmere 5*5 v tvare poľa float `[,]`. Prvky 0 až 2 v oboch smeroch obsahujú transformačné hodnoty pre červenú, zelenú a modrú. Štvrtý prvok uchováva hodnotu transformácie alfa a pomocou piateho riadku môžeme pripočítať hodnoty k hodnotám farieb RGBA. Prvok v bunke 4,4 musí mať vždy hodnotu 1. Piaty stĺpec sa nepoužíva a musí obsahovať všetky hodnoty 0 (okrem spomenutej hodnoty na pozícii 4,4). Príklad matice ktorá nezmení informácie o farbe je v tabuľke č. 1.

Tab. 8. Transformačná matica

ktorá nespôsobí zmeny

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Pri transformácií farieb sa zmenia hodnoty červenej, zelenej a modrej farby a hodnota alfa kanálu každého pixelu podľa matice farieb. Transformácia je jednoduchá, avšak je zložitejšie ju vysvetliť. Stĺpce sú určené pre výsledné hodnoty farieb. Prvý stĺpec udáva po transformácií hodnotu červenej, druhý hodnotu zelenej, tretí hodnotu modrej a štvrtý hodnotu alfa. Riadky sú určené pre farebné zložky transformovaného pixelu. Momentálne platná hodnota červenej, zelenej a modrej farby a hodnota alfa kanálu sa násobia činiteľom, zadaným v príslušnom riadku. Výsledky jednotlivých násobení sa zrátajú, pričom k výsledku sa ešte pripočítajú hodnoty posledného riadku, vynásobené číslom 255. Vďaka tomu je možné ju použiť aj na nelineárne transformácie. Výsledné hodnoty RGBA z jednotlivých stĺpcov sa nakoniec zostavia do jednej výslednej farby, ktorá nám udáva výsledný pixel. Výsledný vzorec je nasledovný:

$$\begin{aligned} \text{červená} = & \text{červená} * (S_0, R_0) + \text{zelená} * (S_0, R_1) + \text{modrá} \\ & * (S_0, R_2) + \text{alfa} * (S_0, R_3) + (S_0, R_4) * 255 \end{aligned} \quad (34)$$

$$\begin{aligned} \text{modrá} = & \text{červená} * (S_1, R_0) + \text{zelená} * (S_1, R_1) + \text{modrá} \\ & * (S_1, R_2) + \text{alfa} * (S_1, R_3) + (S_1, R_4) * 255 \end{aligned}$$

$$\begin{aligned} \text{zelená} = & \text{červená} * (S_2, R_0) + \text{zelená} * (S_2, R_1) + \text{modrá} \\ & * (S_2, R_2) + \text{alfa} * (S_2, R_3) + (S_2, R_4) * 255 \end{aligned}$$

$$\begin{aligned} \text{alfa} = & \text{červená} * (S_3, R_0) + \text{zelená} * (S_3, R_1) + \text{modrá} \\ & * (S_3, R_2) + \text{alfa} * (S_3, R_3) + (S_3, R_4) * 255 \end{aligned}$$

R_x – x znamená číslo riadku, S_x – x znamená číslo stĺpca.

Výsledok pri výpočte je obmedzený na rozsah jedného bajtu, nikdy teda nie sú hodnoty menšie ako 0 a väčšie ako 255.

Použitie spočíva v kreslení metódou DrawImage objektu Graphics. Využíva sa argument ImageAttributes. Pred týmto však musíme najskôr predať tomuto objektu pomocou metódy SetColorMatrix inštanciu typu ColorMatrix. Na príklade môžeme vidieť zosvetlenie obrázku. [17]

6.1.3 Použitie ukazateľov

Medzi ďalšie možnosti ako pracovať rýchlo s bitmapou, je za pomocou použitia ukazateľov (pointrov). S nástupom programovacieho jazyka C# a platformy .NET došlo k zmenám v ich používaní. Boli vyradené, resp. prestali sa používať, a tým sa mal stať kód C# bezpečnejší. Mnoho užívateľov nabralo dojem, že boli odobrané úplne. To je však mylné. S ukazovateľmi je v C# možné pracovať v takzvanom „unsafe kóde“. Vo väčšine prípadov pri programovaní toto nie je potrebné a je doporučené sa tomu vyhýbať. Využíva sa hlavne v prípadoch, kedy potrebujeme každý kúsok výkonu navyše. Nevýhodou je, že pri ich použití nebude funkčný garbage collector a nebude sa používať typová bezpečnosť. To kladie oveľa väčšie nároky na programátora pri písaní kódu. Musí sa sám postarať o uvoľňovanie pamäte. Použitie unsafe kódu je nutné povoliť v nastavení projektu, nakoľko východiskové nastavenia ho zakazujú. Podľa oficiálnej dokumentácie modul CLR sa unsafe-nebezpečný kód nazýva neoverený kód. Preto sa nejedná nutne o nebezpečný kód, ale iba kód, ktorý nie je možnosť overiť podľa CLR. Preto je jeho použitie na vlastnú zodpovednosť a programátor musí zabezpečiť minimalizovať riziká a chyby. [21]

6.1.4 Použitie poľa

Medzi ďalšie, avšak menej používané metódy je metóda pomocou poľa. Na začiatku sa obrázok načíta bod po bode do poľa. V tom sa nijako nelíši od predchádzajúcich spôsobov. Po tomto načítaní sa však všetky operácie robia iba v rámci poľa. Výhodou v tomto spôsobe použitia je rýchlosť. Operácie s poľom sú niekoľkonásobne rýchlejšie ako operácie s bitmapou. Najväčšie využitie tohto spôsobu úpravy je pri kombinácií viacerých transformácií. Pri jednej transformácii (napríklad chceme odstrániť len šum) a následného ukončenia nie je efektívny a dokonca najpomalší.

6.1.5 Porovnanie rýchlosti

Pre porovnanie rýchlosti spracovania obrázku boli vybrané všetky tri spôsoby. Posledný spôsob s poľom nebol do testovania zaradený, nakoľko jeho výhoda je v spojení viacerých transformácií. Na tento účel testovania bol vytvorený vlastný program, ktorý meral celkový čas behu algoritmu na úpravu. Ako vzorový algoritmus bol vybraný prevod do odtieňov šedej vďaka tomu, že algoritmus prechádza a upravuje všetky farebné zložky a je jednoduchý na vysvetlenie. Výsledné vyhodnotenie času je uvedené v tabuľke (Tab. 9).

Výsledný program umožňuje zmerať rýchlosť spracovania ľubovoľného obrázku u všetkých transformácií.

Metóda č. 1: bola zvolená za použitia `GetPixel()` a `SetPixel()`. Ide principiálne o najjednoduchšiu metódu úpravy. Bohužiaľ podľa merania taktiež o najpomalšiu. Dôvodom je, že metóda načítava, prepočítava a ukladá každý pixel zvlášť. Ak ma napríklad obrázok rozlíšenie 4288x2848, znamená to že tento proces prebehne viac ako 12 miliónov krát. Nejedná sa teda o najúčinnšie metódy ako dostať dáta z obrázku.

Verdikt: Tieto metódy sú vhodné pre menšie obrázky, kde užívateľovi nezáleží veľmi na rýchlosti. Je vhodná na vzdelávacie účely, kde sa dá jednoducho demonštrovať princíp práce algoritmu v .NET. Nie je vhodná tam, kde užívateľ potrebuje rýchlosť.

Metóda č. 2: funguje na podobnom princípe ako metóda č. 1. Budeme prechádzať všetky body v obrázku, ale použijeme na spracovanie *unsafe* kód, aby sme dostali dáta oveľa rýchlejšie.

Verdikt: Táto metóda je vhodná tam, kde potrebujeme rýchlosť. Je oveľa rýchlejšia ako predchádzajúca metóda. Veľkou nevýhodou je značná zložitosť algoritmu a programátor musí mať na pamäti, že k tomuto kódu by mal pristupovať oveľa opatrnejšie a precíznejšie.

Metóda č. 3: je založená na použití triedy `ColorMatrix`. Je to v súčasnosti najrýchlejšia metóda na úpravu obrázku. Princíp fungovania je opísaný v kapitole 6.1.2. Na prvý pohľad je zápis a výpočet náročný, avšak po pochopení princípu fungovania sa jedná o veľmi ľahkú a jednoducho použiteľnú metódu.

Verdikt: Ide o najrýchlejšiu a zápisom jednoduchú metódu. Nevýhodou je na prvý pohľad zložitejší zápis.

Pre testovanie boli vybrané 3 rôzne obrázky:

- Obrázok 1: Rozlíšenie 4288x2848, Veľkosť 1,52 MB
- Obrázok 2: Rozlíšenie 1940x1315, Veľkosť 237 kB
- Obrázok 3: Rozlíšenie 800x531, Veľkosť 86,8 kb

Originál obrázky na ktorých bol robený, sa nachádzajú v prílohe P-II a na CD.

Tab. 9. Test rýchlosti algoritmu

Metóda - číslo	Obrázok 1	Obrázok 2	Obrázok 3
GetPixel(), SetPixel() -	35,08 s	7,57 s	1,23 s
Unsafe kód	6,15 s	1,32 s	0,23 s
ColorMatrix	1,52 s	0,38 s	0,07 s

V tabuľke (Tab. 9) môžeme vidieť porovnanie časov vybraných metód spracovania obrázku. Ako najvýhodnejšie vyšlo použitie triedy ColorMatrix.

6.2 Použité algoritmy

Všetky algoritmy boli naprogramované v jazyku C# bez použitia špeciálnych knižníc. Veľkou nevýhodou jazyka C# je už spomenuté pomalé načítavanie a zapisovanie hodnoty bodov pomocou operácií *GetPixel* a *SetPixels*. V programe sú samotné funkcie výpočtov bodov spracovávané vo vláknach. V programe sú u niektorých operácií dostupné aj iné možnosti. Prvou sú operácie pomocou ColorMatrix a druhou pomocou nechráneného kódu. Tieto funkcie sú skôr ukázkového charakteru a slúžia len pre porovnanie náročnosti a času výpočtu.

6.3 Algoritmy pre spracovanie farieb

6.3.1 Odtieň šedej

Konvertovanie obrázku do odtieňa šedej je pomerne ľahké. Z každého bodu v obraze zoberieme všetky farebné zložky a spojíme hodnoty. Bohužiaľ ľudské oko je však rôzne citlivé na rôzne frekvencie a nemôžeme spraviť z farieb len priemer. Musíme ich pridať do výslednej farby v určitých pomeroch. Riadime sa podľa vzorca (13), kde červenú je to

pomer 0.299, pre zelenú 0.587 a pre modrú 0.114. Výslednú hodnotu potom priradíme bodu.



Obr. 15. Príklad na prevod do odtieňu šedej

Príklad zdrojového kódu za použitia metód `GetPixel()` a `SetPixel()`:

```
1     int R, G, B, A;  
2     Color pixelColor;  
3     for (int y = 0; y < bitmapImage.Height; y++)
```

```
4     {
5         for (int x = 0; x < bitmapImage.Width; x++)
6         {
7             pixelColor = bitmapImage.GetPixel(x, y);
8             A = pixelColor.A;
9             R = (byte)((red * pixelColor.R) + (green *
10            pixelColor.G) +
11            (blue * pixelColor.B));
12            G = B = R;
13            bitmapImage.SetPixel(x, y, Color.FromArgb((int)A,
13            (int)R, (int)G, (int)B));
14        }
15    }
```

6.3.2 Negatív

Vytvorenie negatívu obrázku, čiže invertovanie farieb je pomerne jednoduchá úprava. Invertovanie farby nám vytvára negatív aký poznáme z analógových fotografických filmov. Úprava spočíva v tom, že musíme načítať každú farbu každého bodu v obrázku a odpočítať ju od hodnoty 255.

Po dosadení do základného vzorca (15) dostaneme vzťah:
 $255 - \text{hodnota červená}, 255 - \text{hodnota zelená}, 255 - \text{modá}$ pre každý bod v obrázku.



Obr. 16. Příklad na prevodu obrázku na negativ

Príklad zdrojového kódu za použitia metód GetPixel() a SetPixel():

```
1   int R, G, B, A;
2   Color pixelColor;
3   for (int y = 0; y < bitmapImage.Height; y++)
4   {
5       for (int x = 0; x < bitmapImage.Width; x++)
6       {
7           pixelColor = bitmapImage.GetPixel(x, y);
8           A = pixelColor.A;
9           R = (byte)(value - pixelColor.R);
```

```
10         G = (byte) (value - pixelColor.G);
11         B = (byte) (value - pixelColor.B);
12         bitmapImage.SetPixel(x, y, Color.FromArgb((int)A,
13             (int)R, (int)G, (int)B));
14     }
15 }
```

6.3.3 Úprava gamy

Ako parametre algoritmu pre zmenu gamy sú hodnoty gamy pre každú zložku RGB samostatne.

Princíp fungovania gama filtru je vo vytvorení poľa o veľkosti 256 hodnôt pre červenú, modrú a zelenú farbu. Hodnota gamy by mala byť v rozmedzí 0,2 až 5,0.

Základný vzorec (16) pre výpočet je po dosadení:

$$255 * \left(\frac{i}{255} \right)^{\frac{1}{\text{gamma}}} + 0,5 \quad (35)$$



Obr. 17. Příklad úpravy gamy

Příklad zdrojového kódu za použití metod `GetPixel()` a `SetPixel()`:

```
1     int R, G, B, A;
2     Color pixelColor;
3     byte[] redGamma = new byte[256];
4     byte[] greenGamma = new byte[256];
5     byte[] blueGamma = new byte[256];
6     for (int i = 0; i < 256; ++i)
7     {
8         redGamma[i] = (byte)Math.Min(255, (int)
9             ((255.0 * Math.Pow(i / 255.0, 1.0 / r)) + 0.5));
```

```
10         greenGamma[i] = (byte)Math.Min(255, (int)((255.0* Math.Pow(i
11         / 255.0, 1.0 / g)) + 0.5));
12         blueGamma[i] = (byte)Math.Min(255, (int)((255.0 *
13         Math.Pow(i / 255.0, 1.0 / b)) + 0.5));
14     }
15     for (int y = 0; y < bitmapImage.Height; y++)
16     {
17         for (int x = 0; x < bitmapImage.Width; x++)
18         {
19             pixelColor = bitmapImage.GetPixel(x, y);
20             A = pixelColor.A;
21             R = redGamma[pixelColor.R];
22             G = greenGamma[pixelColor.G];
23             B = blueGamma[pixelColor.B];
24             bitmapImage.SetPixel(x, y, Color.FromArgb((int)A,
25                 (int)R, (int)G, (int)B));
26         }
27     }
```

Program umožňuje nastaviť hodnoty gamy pre každú farbu zvlášť. Pokiaľ je hodnota väčšia ako 255, potom bude automaticky nastavená na 255. Program umožňuje aj zmenu mierky pre jemnejšie doladenie. Užívateľ má možnosť nastaviť aj hodnoty mimo doporučenú stupnicu a sledovať zmeny na zvolenom obrázku.

6.3.4 Zmena Jasu

Úprava jasu patrí medzi jednoduchšie transformácie. Ak chceme zmeniť hodnotu jasu, musíme pridať, alebo ubrať určitú hodnotu z každej farebnej zložky pre každý bod obrázku. Nevýhodou takej úpravy je, že je často nenávratna. To znamená, že ak napríklad hodnota farby je blízka k hodnote 255 a my ju zvýšime nad túto hodnotu, bude automaticky orezaná na túto maximálnu hodnotu a my nenávratne stratíme informáciu o predošlej. To isté platí aj pri znížení hodnoty, kedy je hodnota menšia ako 0 automaticky nastavená na nulovú hodnotu.



Obr. 18. Příklad na úpravu jasu obrázku

Příklad zdrojového kódu za použití metod `GetPixel()` a `SetPixel()`:

```
1   for (int y = 0; y < bitmapImage.Height; y++)
2   {
3       for (int x = 0; x < bitmapImage.Width; x++)
4       {
5           pixelColor = bitmapImage.GetPixel(x, y);
6           A = pixelColor.A;
7           R = pixelColor.R + brightness;
8           if (R > 255)
9           {
10              R = 255;
11          }
12          else if (R < 0)
13          {
14              R = 0;
15          }

```

.... pre hodnoty G a B je výpočet rovnaký ako pre R

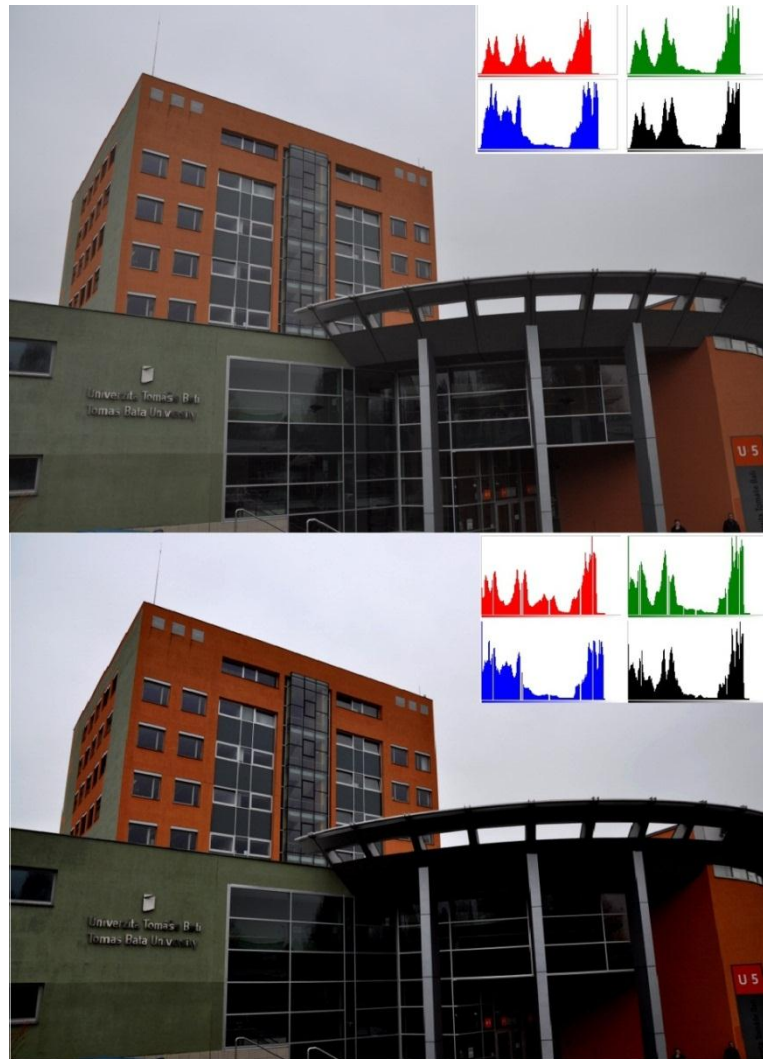
```
16              bitmapImage.SetPixel(x, y, Color.FromArgb(A, R, G, B));
17          }
18  }
```

6.3.5 Zmena Kontrastu

Úprava kontrastu patrí medzi ďalšie často používané úpravy. Transformácia upravuje hodnoty medzi bodmi. Môžeme rozdiel buď zväčšiť, alebo zmenšiť. Pri použitej transformácii sa prednastavená hodnota pre rozdiel pohybuje v rozmedzí -100 až 100. Výsledná hodnota sa určuje zo vzťahu:

$$\left(\frac{100 + \textit{kontrast}}{100}\right)^2 \quad (36)$$

Postup výpočtu je taký, že zoberieme každý pixel v obraze a každú farebnú zložku vydáme hodnotou 255, aby sme dostali hodnotu medzi 1 a 0. Potom odpočítame hodnotu 0,5. Následne vynásobíme hodnotu hodnotou kontrastu. Všetkým zložkám, ktoré budú mať hodnoty záporne sa kontrast zníži, zatiaľ čo kladným hodnotám sa kontrast zvýši. Potom opäť pripočítame hodnotu 0,5 a prevedieme ju na rozsah 0-255. Pokiaľ je výsledná hodnota väčšia ako 255 je hodnota nastavená na túto maximálnu hodnotu. Pokiaľ je menšia ako 0 je nastavená na 0.



Obr. 19. Příklad na úpravu jasu obrázku

Příklad zdrojového kódu za použití metod GetPixel() a SetPixel():

```
1   for (int y = 0; y < bitmapImage.Height; y++)
2   {
3       for (int x = 0; x < bitmapImage.Width; x++)
4       {
5           pixelColor = bitmapImage.GetPixel(x, y);
6           A = pixelColor.A;
7           R = pixelColor.R / 255.0;
8           R -= 0.5;
9           R *= kontrast;
10          R += 0.5;
11          R *= 255;
12          if (R > 255)
13          {
14              R = 255;
```

```
15         }
16         else if (R < 0)
17         {
18             R = 0;
19         }
        .... pre hodnoty G a B je výpočet rovnaký ako pre R

20         bitmapImage.SetPixel(x, y, Color.FromArgb((int)A,
21         (int)R, (int)G, (int)B));
21     }
221 }
```

6.4 Grafické filtre

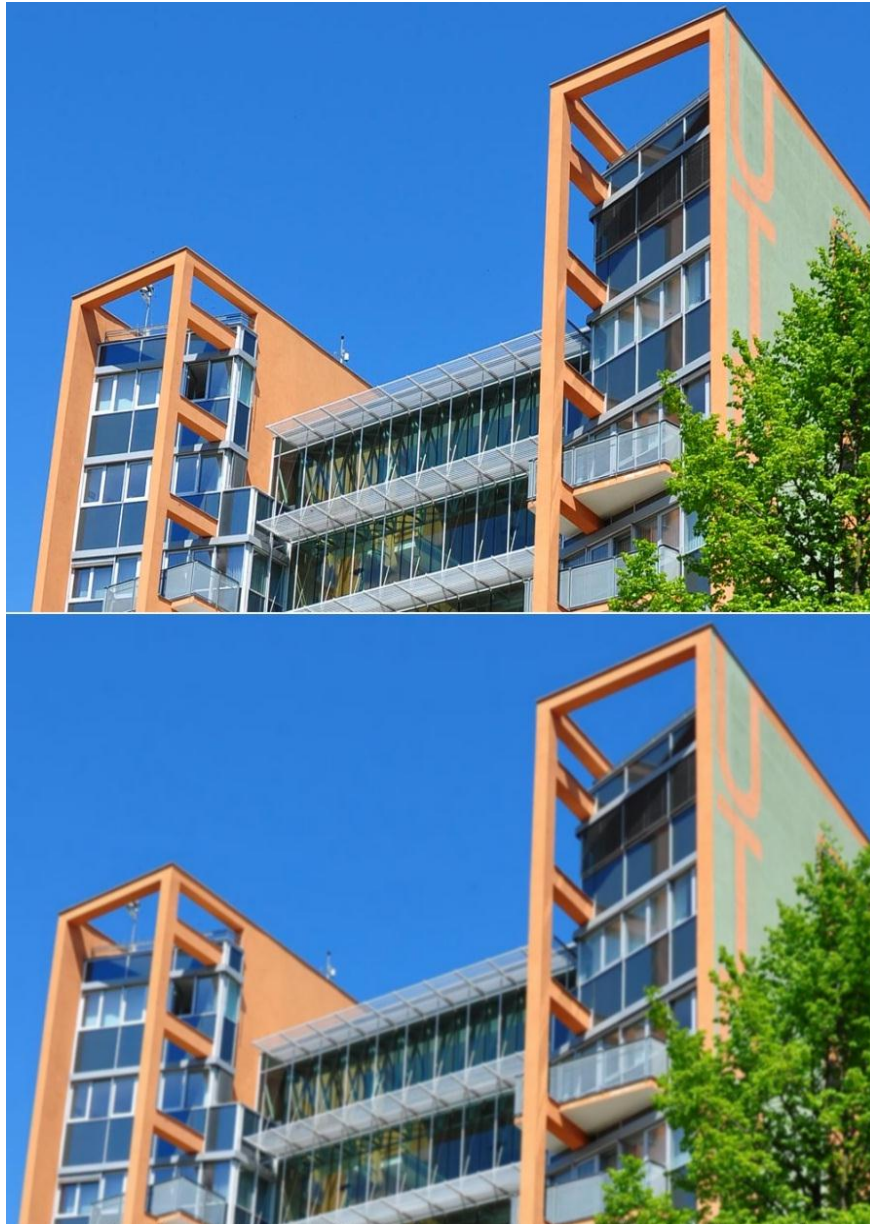
Grafické filtrácie sa používajú napríklad na vyhladzovanie obrazu, ostrenie obrazu, odstraňovanie šumu, hľadanie nespojitosti a hrán. Filtrácie zahrňujúce okolie fungujú väčšinou na princípe konvolúcie.

6.4.1 Konvolučná maska

Konvolučná maska je v podstate matica o určitých rozmeroch. V našom prípade používame konvolučnú maticu o veľkosti 3x3. Výhoda našej veľkosti je, že čím väčšia matica je, tým máme menej bodov, ktoré budú mať vplyv po okrajoch. Princíp funkcie je, že pixel v strede obklopuje osem ďalších s určitou váhou. Celková hodnota matice sa delí faktorom a prípadne je možné k nemu pripočítať ľubovoľnú hodnotu. Zvyčajne hodnota faktoru je hodnota všetkých hodnôt v matici. Pri konvolúcií sa ešte musí riešiť úprava okrajových bodov. V praxi sa používajú dve metódy. Prvá zväčší maticu o polovicu a doplní nulami, prípadne hodnoty doplní zrkadlovým skopírovaním. Druhá a častejšie používaná možnosť je, že sa výpočet robí len v oblasti dosahu masky a nespracovaný okraj sa oreže, čoho výsledkom je zmenšený obrázok. V našom prípade sa okrajové pixely nespracovávajú. Zobrazenie nastavenia konvolučnej masky v programe je v prílohe P-III.

6.4.2 Vyhladzovanie obrazu

Vyhladzovanie obrazu je vhodné pri veľmi veľkých detailoch, alebo ostrých prechodoch. Je vhodné aj na odstránenie šumu z obrazu.



Obr. 20. Příklad na vyhladzovanie obrázku

V strede masky je predvolená hodnota 8 a po okrajoch hodnoty 1. Faktor je v tomto prípade nastavený na 16 a prírastok je 0.

6.4.3 Gaussovo vyhladzovanie obrazu

Gaussovo vyhladzovanie je veľmi podobné normálnemu vyhladzovaniu obrazu. Rozdiel je, že Gaussovo vyhladzovanie dáva prirodzenejšie a živšie rozostrenie. Rozloženie hodnoty v maske vytvára takzvaný kruhový efekt, kde pixely ďalej od okraja majú menšiu váhu.



Obr. 21. Příklad na Gaussovo vyhladzovanie obrázku

6.4.4 Ostrenie obrazu

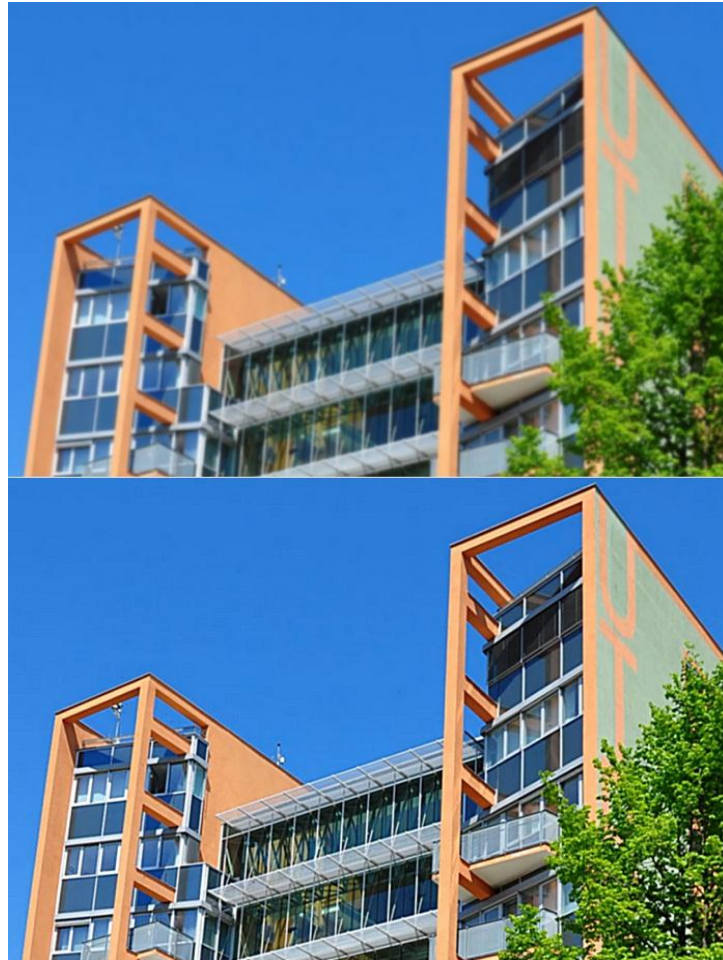
Ostrenie obrazu je opačná funkcia, ako sme pri vyhladzovaní. V prípade aplikácie ostrenia na obrázok, kde sme použili gaussove vyhadzovanie zistíme, že sa jedná o takmer pravý opak. Princíp je vo zvyšovaní rozdielu medzi hodnotami susedných pixelov. Maska odoberá len vo vertikálnom a horizontálnom smere. Ak potrebuje zväčšiť stupeň ostrosti, zväčšíme hodnotu stredného bodu masky.



Obr. 22. Příklad na ostrenie obrázku

6.4.5 Stredné ostrenie obrazu

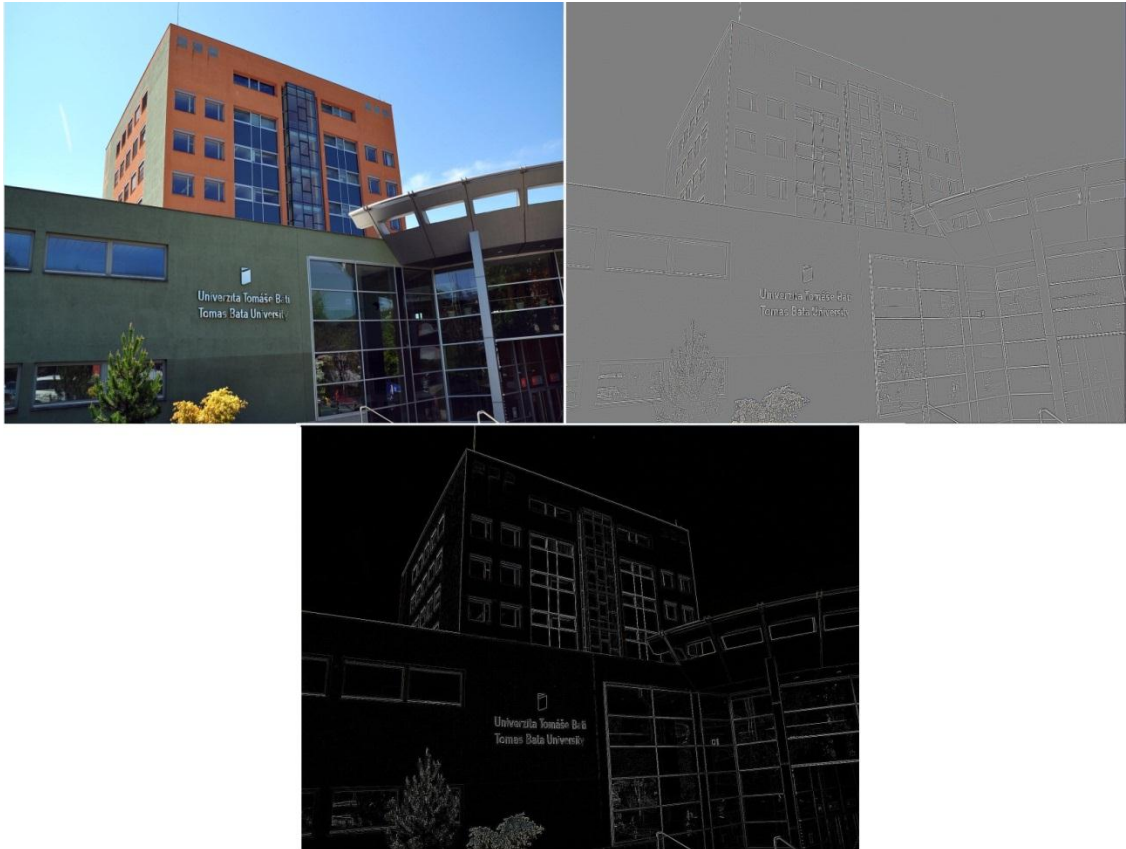
Stredové ostrenie obrazu funguje na princípe klasického ostrenia, ale odoberá hodnoty zo všetkých susedných bodov v maske.



Obr. 23. Příklad na středné ostrenie obrázku

6.4.6 Vytlačený vzor – Emboss

Účelom tejto transformácie je poskytnúť dojem, že obraz je plastický a je akoby vytlačený do povrchu materiálu – reliéf obrázku. Princíp je vo zvýraznení hrán. Využíva sa Laplaceov operátor. Konvolučná maska má hodnotu prírastku k deliteľu až 127. To nám zaručí zosvetlenie obrazu, inak by bol výsledok čierny obrázok. Aplikácia filtra môže byť vo viacerých smeroch. Na obrázku (Obr. 24) je príklad použitia konvolučnej masky podľa tabuľky (Tab. 6).



Obr. 24. Príklad na použitie Laplacovho operátora (pozitívna/negatívna maska)

6.4.7 Detekcia hrán

Spôsobov ako hľadať hrany v obrázku je veľa. Pomocou konvolučnej masky sa jedná o veľmi jednoduchý spôsob. Filtre na detekciu hrán nehýbu s hodnotou pixelu v strede, ale iba s pixelom, ktoré ho obklopujú. Funguje podobne ako filtre s Laplacovým operátorom, avšak tieto zvyšujú účinok. Podobne môže byť maska aplikovaná vo viacerých smerov. Príklad je na obrázku (Obr. 25). Použitá maska bola podľa tabuľky (Tab. 2). Pre každý ďalší obrázok bola maska pootočená o 90° . Program obsahuje aj ďalšie šablóny masiek uvedené v tabuľkách (Tab. 3), (Tab. 4) a (Tab. 5).



Obr. 25. Příklad na detekciu hrán v smeroch pootočenja po 90°

7 GRAFICKÝ EDITOR PRE TRANSFORMÁCIE OBRÁZKOV

7.1 Programátorský pohľad

7.1.1 Štruktúra programu

Program sa skladá z hlavnej triedy, kde je obsluha všetkých tlačidiel, otvorenie a uloženie súborov. Hlavnou triedou, v ktorej sa nachádzajú algoritmy pre výpočet je *ImageAlgorithms*. Trieda *ConvolutionMatrix* obsahuje štruktúru konvolučnej masky používanej pre filtre. Program obsahuje ešte dialógové okná *AboutBox*, ktorá podáva používateľovi informácie o programe a dialógové okno *Peogress*, ktorá sa spúšťa počas výpočtu a informuje používateľa, že program stále pracuje.

Diagram Tried sa v prehľadnej forme nachádza v prílohe P-IV.

7.1.2 Trieda *ImageAlgorithms*

Trieda obsahuje metódy zo samotnými algoritmami. Samotné algoritmy sú umiestnené v privátnych triedach a pre operácie s nimi sú dostupné operácie verejné. Každá verejná operácia sa stará o spustenie metódy výpočtu vo vlákne a zobrazenie dialógového okna o prebiehaní operácie.

Hlavné metódy triedy

UseBrightness(double value) – je metóda na úpravu jasů v obrázku. Ako vstupný parameter sa zadáva hodnota jasů.

UseGamma(double red, double green, double blue, double merge, bool separate) – je metóda na úpravu gamy v obrázku. Vstupné parametre sú hodnota červenej, zelenej, modrej, celkovej hodnoty a logická hodnota, či chceme použiť hodnoty zadávané samostatne alebo z celkovej hodnoty všetky rovnako. Napríklad, ak chceme meniť hodnoty farieb zvlášť, hodnota merge nie je použitá a neberie sa do úvahy.

UseContrast(int value) – je metóda na úpravu kontrastu v obrázku. Ako vstupný parameter je hodnota, o akú sa má jas v obrázku upraviť.

UseNegative(decimal value) – je metóda na prevod obrázku do negatívu. Vstupnú hodnotu, od ktorej sa odraťavaje možné meniť.

UseGreyscale(decimal red, decimal green, decimal blue) – je metóda na prevod obrázku do odtieňov šedej. Vstupné hodnoty sa dajú nastavovať pre každú farbu zvlášť.

7.1.3 Trieda *ConvolutionMatrix*

Trieda je určená pre zostavenie konvolučnej matice. Bola vytvorená pre uľahčenie práce. Má len jednu metódu a viacej atribútov, pričom všetky sú verejné.

SetAll(double value) – metóda je určená na nastavenie alebo zresetovanie matice. Vstupná hodnota nahradí všetky pozície v maske.

7.2 Užívateľský pohľad



Obr. 26. Hlavná obrazovka programu

Grafický editor je naprogramovaný v programe Microsoft Visual Studio 2010 Express, v programovacom jazyku C#. Ovládanie programu je jednoduché a všetko je dostupné z hlavného menu. Program disponuje množstvom nastavení pre úpravu obrázkov. Je možné podrobne nastavovať vstupné parametre algoritmov pre spracovanie.

7.2.1 Systémové požiadavky

Operačný systém: Windows XP, Vista, 7, 2003, 2008

Procesor: Intel/AMD 1000 MHz

Pamäť: 1024 MB RAM

Disk: 500 MB

CD-ROM/DVD-ROM mechanika

Softvér: Microsoft .NET Framework 4.0

7.2.2 Inštalácia

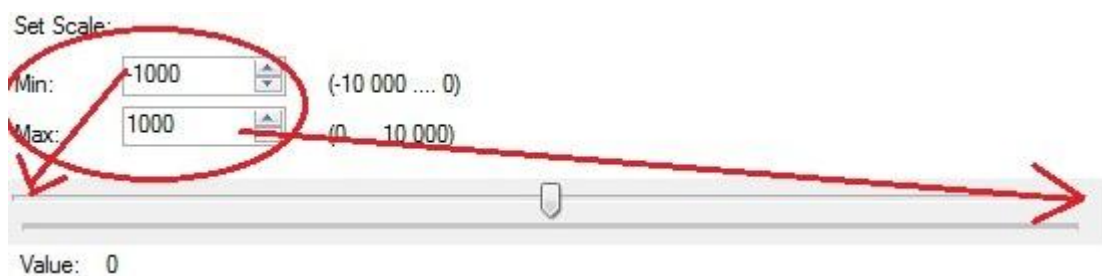
Spustením súboru setup.exe sa spustí sprievodca inštalátorom, ktorý naviguje počas inštalácie v niekoľkých krokoch. Možnosť je aj spustiť program bez inštalácie. Umiestnenie súborov je v obsahu podľa prílohy P-I.

7.2.3 Popis programu

Program je rozdelený na tri hlavné časti, ako je vidno z obrázku (Obr. 26). V hornom panely (1) sa nachádzajú všetky ovládacie prvky programu. Hlavné okno je rozdelené na dve časti. V ľavej časti sa nachádzajú ovládacie prvky a nastavenia úprav (2). Kliknutím na záložku, sa zobrazia nastavenia zvolenej transformácie. Každá záložka obsahuje tlačidlo pre návrat na hlavnú záložku. V pravej časti (3) sa nachádza obrázok. Ten sa rozmerovo automaticky prispôsobí veľkosti otvoreného okna tak, že užívateľ ho vidí celý.

Program obsahuje nápovede. Pri podržaní kurzoru na tlačidle, alebo nastaveniach, sa zobrazí popis.

Každá transformácia ma po spustení programu prednastavené základné hodnoty. Užívateľ má možnosť meniť ich hodnotu. V prípade posuvníkov má možnosť ešte meniť ich rozsah, a tým aj ich citlivosť (Obr. 27).



Obr. 27. Príklad nastavenia rozsahu posuvníka

ZÁVER

Cieľom tejto práce bolo preskúmať možnosti a spôsoby transformácií obrázkov pod platformou .NET.

V úvode práce som podrobne objasnil najbežnejšie používané farebné modely a formáty rastrových obrázkov. Následne som sa venoval najbežnejšie používaným geometrickým transformáciám a ich spájaním. Ako posledné boli vysvetlené základné farebné filtre a filtrácie obrazu.

V praktickej časti som sa venoval tvorbe programu. Zdrojové kódy boli písané tak, aby im porozumel aj začiatočník, alebo užívateľ neznalý programovania v .NET. Používateľ a prípadný záujemca tak nemusí podrobne študovať zdrojové kódy, ale môže použiť už hotovú vytvorenú triedu bez ďalších úprav. Tá je naprogramovaná tak, že používateľ má maximálnu kontrolu nad transformačným algoritmom. Pre väčšinu transformácií neboli použité žiadne existujúce triedy a metódy. Tie boli použité len pre porovnanie náročnosti výpočtu. Pre túto diplomovú prácu bol použitý programovací jazyk C#.

Program umožňuje vykonávať všetky transformačné algoritmy uvedené v diplomovej práci. Oproti bežne používaným programom, tento program umožňuje:

- Načítanie a ukladanie bežne používaných grafických súborov.
- Možnosť meniť parametre výpočtov, ktoré bežne neumožňujú ani triedy na to určené.
- Vytvorenie vlastnej konvolučnej masky pre filtráciu obrazu.
- Porovnanie dĺžky času rovnakej transformácie v rôznych spôsobov prevedenia algoritmu.

Po preštudovaní tejto práce a programu, čitateľ získa poznatky o možnostiach a spôsoboch úpravy obrázkov pod platformou .NET. Vďaka možnosti zmerať čas výpočtu na vlastných obrázkoch, si môže jednoducho zvoliť pre každý typ transformácie vhodnú metódu.

Program je vhodný aj pre výučbu, nakoľko umožňuje takmer ľubovoľne meniť parametre algoritmov a používateľ má tak možnosť sledovať zmeny a lepšie pochopiť princíp transformácie.

Súčasťou práce je aj užívateľský a programátorský manuál.

K práci bola vytvorená podrobná elektronická dokumentácia k transformačným algoritmom, spôsobu ich aplikácie a možnosti využitia. Bol braný zreteľ na jednoduchosť a stručnosť výkladu. Dokumentáciu je možné použiť vo výučbe.

V budúcnosti chcem pokračovať vo vývoji a zlepšovaní aplikácie. Chcem pridať univerzálnu konvolučnú masku, kde bude mať používateľ možnosť takmer ľubovoľne meniť jej veľkosť. Taktiež pridám do programu systém na detekciu identifikáciu tváre a steganografiu.

ZÁVER V ANGLIČTINE

The aim of this thesis was to explore the possibilities and ways of transformations of images under .NET platform.

In the introduction of the thesis I have properly clarified the most commonly used colour models and formats of raster images. Consequently, I have attended to the most commonly used geometric transformations and joining them together. Eventually, the basic colour filters and image filtrations have been explained.

In the practical part I have dealt with the program creation. The source codes have been written in such a way that can also be understood by beginner or by user who is not able to program in .NET. A user or a possible person interested does not have to study the source codes in detail, but he/she can use already created data class without any other modifications. The data class has been programmed in such a way that a user has maximal control of transformation algorithm.

No existing data classes and methods have been used in most transformations. They have been used only for the comparison of difficulty of calculation.

The algo C# has been used for this thesis.

The program enables to do all transformation algorithms mentioned in the thesis. Compared to commonly used programs, this one enables to:

- Download and save commonly used graphic files.
- Change the parametres of calculations which are not usually enabled to such data classes.
- Create the own convolute mask for image filtration.
- Compare time length of the same transformation in different ways of algorithm realization.

After studying of this thesis and program, a reader will gain the knowledge of possibilities and ways of adjusting of images under .NET platform. He/she can simply choose the appropriate method for each type of transformation just thanks to the possibility to measure the time of his/her own images.

The program is suitable also for training because it enables to change parameters of algorithms almost arbitrarily. Therefore a user can follow the changes and better understand the principle of transformation.

The manual for users and programmers is a part of the thesis, as well.

The detailed electronic documentation for transformation algorithms and the way of its application and usage has also been created for this thesis. The attention has been paid to the simplicity and briefness of the explanation.

The documentation is possible to use for training.

In the future I would like to continue in development and improvement of the application. My intention is to add the universal convolute mask where a user will have the possibility to change its size almost arbitrarily. I will also add the system for detection of face identification and steganographics to the program.

ZOZNAM POUŽITEJ LITERATURY

- [1] ŽÁRA, Jiří, et al. *Moderní počítačová grafika : kompletní průvodce metodami 2D a 3D grafiky*. druhé, přepracované a rozšířené vydání. Brno : Computer Press, a.s., 2004. 609 s. ISBN 80-251-0454-0.
- [2] MARTIŠEK, Dalibor. *Matematické principy grafických systémů* . Brno : Littera, 2002. 296 s. ISBN 80-85763-19-2.
- [3] *Color Management* [online]. 2008 [cit. 2011-05-11]. Minikurz správy farieb. Dostupné z WWW: <<http://www.colormangement.sk/CMkurz/Lekcia02.php>>.
- [4] MURRAY, James D.; VANRYPER, William. *Encyklopedie grafických formátů*. Druhé vydání. Praha : Computer press, a.s., 1997. 922 s. ISBN 80-7226-033-2.
- [5] SOVIČ, Dušan. *Dušanové stránky o kompresii* [online]. 4. Októbra 2005 [cit. 2011-05-11]. Grafický formát JPG. Dostupné z WWW: <<http://www.pakuj.host.sk/jpeg/jpeg.html>>.
- [6] SOVIČ, Dušan. *Dušanové stránky o kompresii* [online]. 4. Októbra 2005 [cit. 2011-05-11]. Grafický formát PNG. Dostupné z WWW: <<http://www.pakuj.host.sk/png/png.html>>.
- [7] JOHNSON, James T. *The Code Project* [online]. 2002 [cit. 2011-05-11]. Image Rotation in .NET. Dostupné z WWW: [<http://www.codeproject.com/KB/graphics/rotateimage.aspx>].
- [8] VINCE, John. *Mathematics for Computer Graphics*. Second Edition. London : Springer, 2006. 248 s. ISBN 1-84628-034-6.
- [9] ŠTUGEL, Juraj. *Netgraphics : výuka počítačovej grafiky* [online]. 2011 [cit. 2011-01-26]. Dostupné z WWW: [<http://www.netgraphics.sk/sk>].
- [10] DOBEŠ, Michal. *Zpracování obrazu a algoritmy v c#*. 1. vydání. Praha : BEN - Technická literatura, 2008. 143 s. ISBN 978-80-7300-233-6, EAN 9788073002336.
- [11] ŽÁRA, Jiří, et al. *Moderní počítačová grafika : kompletní průvodce metodami 2D a 3D grafiky*. druhé, přepracované a rozšířené vydání. Brno : Computer Press, a.s., 2004. 609 s. ISBN 80-251-0454-0.

- [12] FOLEY, James D., et al. *Computer Graphics : PRINCIPLES AND PRACTICE*. SECOND EDITION in C. Boston (MA) : ADDISON-WESLEY, 1997. 1174 s. ISBN 0-321-21056-4.
- [13] DVOŘÁK, Hynek . *Fotopatracka* [online]. c2010 [cit. 2011-05-11]. Co je histogram. Dostupné z WWW: <<http://www.fotopatracka.cz/articles/4/hm.php?member=1&tm=0&id=4>>.
- [14] AUER, Brian. *Epic Edits* [online]. April 14th, 2007 [cit. 2011-05-11]. Working With Image Histograms. Dostupné z WWW: <<http://blog.epicedits.com/2007/04/14/working-with-image-histograms/>>.
- [15] Microsoft. *Craig's Utility Library* [online]. 2011 [cit. 2011-05-11]. Dostupné z WWW: [<http://cul.codeplex.com/SourceControl/changeset/view/61004#707831>].
- [16] NAGEL, Christian, et al. *C# 2008 : Programujeme Profesionálně*. Vydání první. Brno : Computer Press, a.s., 2009. 1126 s. ISBN 978-80-251-2401-7.
- [17] BĚHÁLEK, Marek. *VŠB-TU Ostrava* [online]. 2007 [cit. 2011-05-11]. Programovací jazyk C#. Dostupné z WWW: <<http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/index.html>>.
- [18] BAYER, Jürgen. *C# 2005 : Velká kniha řešení*. Vydání první. Brno : Computer Press, a.s., 2007. 813 s. ISBN 978-80-251-1620-3.
- [19] *Počítačová Grafika* [online]. 2010 [cit. 2011-01-25]. TRANSFORMÁCIE V PG . Dostupné z WWW: [<http://pg.kpi.fei.tuke.sk/?q=node/6>].
- [20] *MathWorks* [online]. c2011 [cit. 2011-05-13]. R2011a Documentation → Image Processing Toolbox. Dostupné z WWW: <<http://www.mathworks.com/help/toolbox/images/f8-20792.html>>.
- [21] *Visual C# Developer Center* [online]. c2011 [cit. 2011-05-13]. Dostupné z WWW: <<http://msdn.microsoft.com/en-us/vcsharp/aa336706>>.
- [22] MAREŠ, Amadeo. *1001 tipů a triků pro C#*. Vydání první. Brno : Computer Press, a.s., 2008. 360 s. ISBN 978-80-251-2125-2.

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

RGB	Red-Green-Blue (Farebný model – červená, zelená, modrá)
CMYK	Cyan-Magenta-Yellow (Farebný model – modrozelená, purpurová, žltá)
HVS	Hue-Saturation-Value (Farebný model – farebný tón, sýtosť, jasová hodnota)
HLS	Hue-Lightness-Saturation (Farebný model - farebný tón, svetlosť, sýtosť)
BMP	Bit Mapped Picture (Formát súboru)
JPEG	Joint Photographic Experts Group (Formát súboru)
DWT	Diskrétna vlnková transformácia (Komprimačný algoritmus)
TIFF	Tagged-Image File Format (Formát súboru)
GIF	Graphics Interchange Format (Formát súboru)
LZW	Lempel-Ziv-Welch (Komprimačný algoritmus)
PNG	Portable Network Graphics (Formát súboru)
GUI	Graphical User Interface (Grafické rozhranie)
RAW	Takzvané „surové“ nespracované a nekomprimované dáta (Formát súboru)
DCT	Discrete cosine transform (Komprimačný algoritmus)
W3C	World Wide Web Consortium
COM	Component Object Model (Vrstva v .NET)
OOP	Objektovo Orientované Programovanie
OS	Operačný Systém
CLR	Common Language Runtime (Vrstva v .NET)
CLS	Common Language Specifications (Vrstva v .NET)

ZOZNAM OBRÁZKOV

Obr. 1. Aditívny spôsob miešania farieb	13
Obr. 2. Aditívny spôsob miešania farieb	13
Obr. 3. Geometrické zobrazenie priestoru RGB.....	14
Obr. 4. Geometrické zobrazenie priestoru CMY	15
Obr. 5. Geometrické zobrazenie priestoru HSV [20]	16
Obr. 6. Geometrické zobrazenie priestoru HLS [20].....	16
Obr. 7. Princíp fungovania warpingu.....	23
Obr. 8. Transformačná krivka hodnôt jasu pre negatív	25
Obr. 9. Závislosť hodnoty bodu a jeho jasu:.....	26
Obr. 10. Histogram obrázku	28
Obr. 11. Príklad na nevyrovnaný histogram	29
Obr. 12. Príklad na vyrovnaný histogram.....	30
Obr. 13. Príklad na gausovo rozdelenie.....	32
Obr. 14. Architektúra platformy .NET Framework [21]	39
Obr. 15. Príklad na prevod do odtieňu šedej.....	47
Obr. 16. Príklad na prevodu obrázku na negatív	49
Obr. 17. Príklad úpravy gamy.....	51
Obr. 18. Príklad na úpravu jasu obrázku	53
Obr. 19. Príklad na úpravu jasu obrázku	55
Obr. 20. Príklad na vyhladzovanie obrázku.....	57
Obr. 21. Príklad na Gaussovo vyhladzovanie obrázku.....	58
Obr. 22. Príklad na ostrenie obrázku	59
Obr. 23. Príklad na stredné ostrenie obrázku.....	60
Obr. 24. Príklad na použitie Laplacovho operátora (pozitívna/negatívna maska).....	61
Obr. 25. Príklad na detekciu hrán v smeroch pootočená po 90°	62
Obr. 26. Hlavná obrazovka programu	64
Obr. 27. Príklad nastavenia rozsahu posuvníka.....	65

ZOZNAM TABULIEK

Tab. 1. Robertsov operátor	35
Tab. 2. Maska Prewittovej	35
Tab. 3. Maska Sobelovho operátora v troch smeroch.....	36
Tab. 4. Maska Robinsonovho operátora	36
Tab. 5. Maska Kirschovho operátora.....	36
Tab. 6. Maska Laplaceovho	37
Tab. 7. Maska Laplacian.....	37
Tab. 8. Transformačná matica	43
Tab. 9. Test rýchlosti algoritmu.....	46

ZOZNAM PRÍLOH

- P I Obsah priloženého CD
- P II Obrázky na ktorých bol robený test rýchlosti algoritmu
- P III Konvolučná maska
- P IV Diagram Tried

PRÍLOHA P I: OBSAH CD

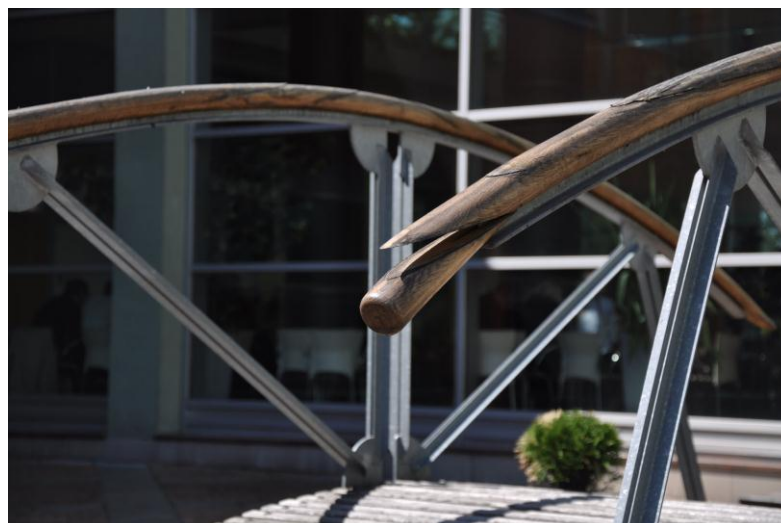
Obsah priloženého CD/DVD nosiča:

- /obsah.txt - obsah CD/DVD
- /praca/ - obsahuje prácu vo formáte .docx a pdf
- /program/source/ - obsahuje zdrojové súbory programu (projekt Visual Studio 2010)
- /program/setup/ - obsahuje inštalátor hotového programu
- /program/run/ - obsahuje samotne spustiteľný .exe súbor
- /prilohy/ - obsahuje prílohy (testovacie obrázky)
- /addons/net/ - obsahuje inštalátor .net framework 3.5 a 4.0
- /dokumentacia/ - obsahuje elektronickú dokumentáciu k diplomovej práci vo formáte .pptx a .pdf

PRÍLOHA P II: TESTOVACIE OBRÁZKY



Obrázok 1



Obrázok 2



Obrázok 3

PRÍLOHA P III: KONVOLUČNÁ MASKA

Convolution Matrix

0	0	0
0	0	0
0	0	0

/ 0 + 0

Apply Convolution Matrix

Algorithms

- Smooth
- Gaussian Blur
- Sharpen
- Mean Filter
- Emboss ----- Laplacian
- Horizontal
- Vertical
- Loosy
- All Directions
- Edge Detect - Edge Detect
- Prewitt
- Sobell
- Robinson
- Kirsha
- Default

Obrázok 4 – Konvolučná maska

PRÍLOHA P IV: DIAGRAM TRIED

