Číslicové spracovanie a filtrácia statických obrazov ako WEB služba z oblasti aplikovaného ČSS

DIPLOMÁ PRÁCA

JOZEF POHORELEC

ŽILINSKÁ UNIVERZITA V ŽILINE
Elektrotechnická fakulta
Katedra telekomunikácií

Študijný odbor: TELEKOMUNIKÁCIE
Vedúci diplomovej práce: Ing. Róbert Hudec PhD.

Stupeň kvalifikácie: inžinier (Ing.)
Dátum odovzdania diplomovej práce: 18.5.2007

ŽILINA 2007

ZADANIE PRÁCE

Abstrakt

Anotácia v slovenskom jazyku:

Diplomová práca sa zaoberá problematikou číslicového spracovaniu statických obrazov. Hlavným cieľom tejto práce je vytvoriť GUI aplikáciu, ktorá dokáže spracovať a následne filtrovať statické obrazy cez web rozhranie a to z ktoréhokoľvek miesta na zemi bez zbytočných inštalácií či zakupovaní licenčných práv.

Anotácia v anglickom jazyku:

This diploma work deals with problems of digital processing of static images. Main goal of this work is to create a GUI application, which can process and then filtrate static images via web interface from any place in the world and without any extra installations or buying a program license.

ANOTAČNÝ ZÁZNAM

Priezvisko a meno: *Jozef Pohorelec* **rok:** *2007*

Názov práce: *Číslicové spracovanie a filtrácia statických obrazov ako WEB služba z oblasti aplikovaného ČSS*

ELEKTROTECHNICKÁ FAKULTA **KATEDRA TELEKOMUNIKÁCIÍ**

Počet strán: *50* **Počet obrázkov:** *36* **Počet tabuliek:** *1*

Počet grafov: *0* **Počet príloh:** *1* **Počet použ. lit.:** *16*

Anotácia v slovenskom jazyku:

Diplomová práca sa zaoberá problematikou číslicového spracovaniu statických obrazov. Hlavným cieľom tejto práce je vytvoriť GUI aplikáciu, ktorá dokáže spracovať a následne filtrovať statické obrazy cez web rozhranie a to z ktoréhokoľvek miesta na zemi bez zbytočných inštalácií či zakupovaní licenčných práv.

Anotácia v anglickom jazyku:

This diploma work deals with problems of digital processing of static images. Main goal of this work is to create a GUI application, which can process and then filtrate static images via web interface from any place in the world and without any extra installations or buying a program license.

Kľúčové slová:

číslicové spracovanie, filtrácia, statický obraz, pixel, šum

Vedúci diplomovej práce: *Ing. Róbert Hudec, PhD.*

Recenzent: *Doc. Ing. Ľudmila Maceková PhD.*

Dátum odovzdania práce: *18.5.2007*

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek a symbolov

FIR	filter s konečnou impulzovou odozvou (Finite Impulse Response Filter)
IIR	filter s nekonečnou impulzovou odozvou (Infinite Impulse Response Filter)
LMS	metóda najmenšej strednej kvadratickej hodnoty (Least Mean Square)
LMS IIR	LMS IIR filter
WEB	Počítačová sieť pozostávajúca z internetových stránok, ktorá používa hypertextový prenosový protokol
DFT	diskrétna Fourierova transformácia (Discrete Fourier Transformation)
FFT	rýchla Fourierova transformácia (Fast Fourier Transformation)
IFT	inverzná FFT
JDK	vývojový balík (Java Development Kit)
JVM	javový virtuálny stroj (Java Virtual Machine)
OOP	objektovo orientované programovanie
Bbp	počet bitov na kanál (bits per pixel)
G20	aditívny Gaussov šum s nulovou strednou hodnotou a smerodajnou odchýlkou 20
I10	10% impulzový šum
1-D	jednorozmerný
2-D	dvojrozmerný
3-D	trojrozmerný
m-D	m-rozmerný
3x3	9-prvkové okno v tvare štvorca 3x3
5x5	25-prvkové okno v tvare štvorca 5x5
σ^2_υ	disperzia šumu
P_f	výkonové spektrum signálu
P_υ	výkonové spektrum šumu
m_f	lokálny priemer
σ_f	disperzia vstupného obrazu
Y, y	výstupný obraz

X, x	vstupný obraz
S, s	šum
N	nezašumený obraz
h	konvolučná maska
VM	vektorový mediánový filter
w_i	vektor váhových koeficientov.
W	váhová funkcia
C_I	koeficient šumovej disperzie
V	disperzia filtrovaného okna
I	priemerná hodnota filtrovaného okna
r_{NN}	delta funkcia
P_{NN}	výkonové spektrum bieleho šumu

Úvod

Číslicové filtrácia je v súčasnej dobe veľmi rozšírený a dôležitý prostriedok na obnovu poškodených obrazov. Pri prechode signálu cez prenosový kanál môže vznikať veľké množstvo rôznych poškodení, vrátanie poškodení spôsobených šumom. Existuje široká škála rôznorodých typov šumov a ich kombinácie, a preto je potrebné dobre poznať ich vlastnosti.

V praxi sa však vyskytujú aj šumy, ktoré sa v čase menia alebo nepoznáme ich časové charakteristiky. V takýchto prípadoch je vhodné použiť na ich filtráciu adaptívne metódy. Keďže sa jedná o filtráciu farebných obrazov, dosiahnutie kvalitných výsledkov je oveľa ťažšie ako je to pri šedých obrazoch, pretože farebné obrazy sú zložené z troch zložiek a je medzi nimi vzájomná korelácia, ktorá by sa nemala porušovať. Toto pravidlo sa však pri filtrácii niekedy porušuje, pretože metódy ktoré takýmto spôsobom filtrujú statické obrazy sú veľmi sofistikované a teda časovo náročné.

Prvá polovica práce je venovaná vysvetleniu základných pojmov z oblasti číslicového spracovania obrazu. Uvádza sa v nej, čo to vlastne obraz je, z čoho sa skladá a aké rôzne typy digitálnych obrazov poznáme. Ďalej sú tu rozpísané najčastejšie vyskytujúce sa šumy, prečo vznikajú a aký je ich výsledný dôsledok na obraz. V týchto kapitolách sú uvedené a podrobnejšie vysvetlené aj filtre, ktoré dokážu špecifické šumy odstrániť.

Hlavnou úlohou tejto práce však bolo navrhnúť WEB službu využívajúcu metódy číslicového spracovania statického obrazu, ktorá implementuje vybrané filtre. Ako najvhodnejší prostriedok na vytvorenie takéhoto softwaru bolo zvolené Java prostredie, respektíve použitie jednej jeho súčasti – apletu. Pri výbere sa bralo do úvahy nielen to, že dokáže pracovať ako WEB aplikácia ale má tú výhodu, že Java má multiplaformovú kompatibilitu, a teda je možné ju spustiť z viacerých systémových platforiem

1 Statické obrazy

Statické obrazy sú zložené z pixelov. Každý farebný pixel v digitálnom obraze je vytvorený pomocou kombinácie troch primárnych farieb: červená, zelená a modrá. Každá primárna farba sa nazýva farebný kanál a môže nadobúdať ľubovolné hodnoty v rámci špecifikovanej bitovej hĺbky. Bitová hĺbka udáva koľko jedinečných farieb je k dispozícii pre vytvorenie obrazu vo farebnej palete a vyjadruje počet bitov na kanál. Pre šedé obrazy bitová hĺbka určuje koľko odtieňov je k dispozícii. Obrazy s väčšou bitovou hĺbkou môžu zakódovať viac odtieňov alebo farieb, pretože môžu použiť viac kombinácií jednotiek a núl. Počet bitov na kanál (bpp – ang. bits per pixel) udáva sumu bitov všetkých troch farebných kanálov a určuje koľko farieb je k dispozícii pre každý pixel.

Väčšina farebných obrazov z digitálnych obrazov má 8 bitov na kanál, a teda používa 8 jednotiek a núl na každý kanál. Toto umožňuje 2^8 alebo 256 rozdielnych kombinácií. Každá kombinácia prezentuje hodnotu intenzity pre každú primárnu farbu. Keď sú skombinované všetky primárne farby do jedného pixelu, možno vytvoriť 2^{8*3} alebo 16,777,216 rozličných farieb, čo predstavuje 24 bpp, pretože každý pixel je zložený z troch 8 bitových kanálov. Počet farieb, ktorý je k dispozícii pre X-bitový obraz môže byť vypočítaný ako 2^{3*X}.

Nasledujúca tabuľka ilustruje rozdielne typy obrazov v závislosti na použití bitovej hĺbky, celkového počtu farieb a ich bežné mená [6].

Počet bitov na pixel	Celkový počet farieb	Bežné mená
1	2	Monochrome
2	4	CGA
4	16	EGA
8	256	VGA
16	65536	XGA, High Color
24	16777216	SVGA, True Color
32	16777216 + Priehľadnosť	
48	$281*10^{12}$	

Tab.1 Vplyv bitovej hĺbky na kvalitu obrazu

2 Poškodenia obrazu - Šumy

Šum sa definuje ako nechcený signál, ktorý ruší komunikáciu, meranie, vnímanie alebo spracovanie signálu, ktorý prenáša informáciu. Prezentuje sa v rozličných stupňoch a takmer vo všetkých prostrediach. V digitálnych mobilných telefónnych systémoch sa môžu vyskytovať rôzne variácie šumu, ktoré môžu degradovať kvalitu komunikácie, ako sú napríklad akustický šum pozadia, teplotný šum, výstrelkový šum, elektromagnetický rádio - frekvenčný šum, šum spôsobený spracovaním signálu atď.

Šum môže spôsobovať chyby na vysielaní a dokonca môže porušiť komunikačný proces. Preto je spracovanie šumu veľmi potrebná a nedeliteľná časť moderných telekomunikačných systémov na spracovanie signálov. Úspech metód na spracovanie šumu závisí na schopnosti charakterizovať a modelovať šumový proces a na vhodnom používaní šumových charakteristík, ktoré oddeľujú signál od šumu [14].

V závislosti na spôsobe pridávania šumu do obrazu sa delia šumy na aditívne (2.1a), impulzové a multiplikatívne (2.1b) [1].

$$Y(x,y) = S(x,y) + N(x,y) \tag{2.1a}$$

$$Y(x,y) = S(x,y).N(x,y) \tag{2.1b}$$

kde $Y(x,y)$ je zašumený signál, $S(x,y)$ je originálny nezašumený signál a $N(x,y)$ je šum.

Aditívny šum sa delí na aditívny Gaussov šum (AWGN – ang. Aditive White Gaussian Noise) a „farebné šumy“. Biely šum je čisto náhodný šum, ktorý má ploché výkonové spektrum. Biely šum teoreticky obsahuje všetky frekvencie na rovnakej intenzite. Farebný šum je šum, ktorý nie je biely alebo iný širokopásmový šum, ktorého spektrum nemá plochý tvar. Je to napríklad ružový šum, hnedý šum a autoregresívny šum.

Impulzný šum pozostáva z krátko trvajúcich impulzov s náhodnou amplitúdou a trvaním. Znehodnotí len niektoré časti obrazu takým spôsobom, že pôvodnú hodnotu pixelu nahradí náhodnou hodnotou s rovnomerným rozložením pravdepodobnosti.

Multiplikatívny šum vzniká pri radarový snímkach alebo pri obrazoch z ultrazvuku. Medzi typické príklady multiplikatívneho šumu patrí bodkový šum. [14].

V praxi sa vyskytujú aj iné typy šumov, ale ich výskyt je nízky [10]. Medzi takéto šumy patria:

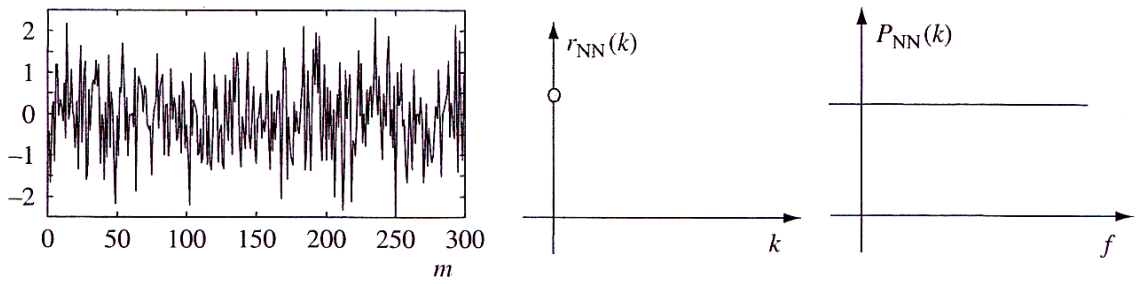
- šum s Laplace-ovým rozdelením
- šum s Γ rozdelením
- šum s logaritmickým rozdelením pravdepodobnosti
- long/short tail – šumy s tzv. dlhým a krátkym chvostom

2.1 Biely šum

Biely šum je definovaný ako nekorelovaný náhodný šumový proces s rovnakým výkonom na všetkých frekvenciách (Obrázok 2.2). Náhodný šum, ktorý má rovnaký výkon na všetkých frekvenciách v rozsahu $\pm\infty$ by potreboval k dispozícii nekonečne veľký výkon a preto je iba teoretickým konceptom. Avšak pásmovo limitovaný šum s plochým spektrom pokrývajúcim frekvenčný rozsah komunikačného systému je z určitého hľadiska bielym šumom. Napríklad v audio systémoch so šírkou pásma 10 kHz, akékoľvek ploché spektrum so šírkou rovnou alebo väčšou ako 10 kHz vyzerá ako biely šum.

Autokorelačná funkcia šumového procesu so spojitým priebehom, nulovou strednou hodnotou a disperziou σ^2 je delta funkcia a je daná vzťahom

$$r_{NN}(\tau) = \mathrm{E}[N(t)N(t+\tau)] = \sigma^2\delta(\tau) \tag{2.1}$$



Obr. 2.2 Zobrazenie bieleho šumu (vľavo), delta funkcie (v strede), výkonové spektrum je konštantné (vpravo)

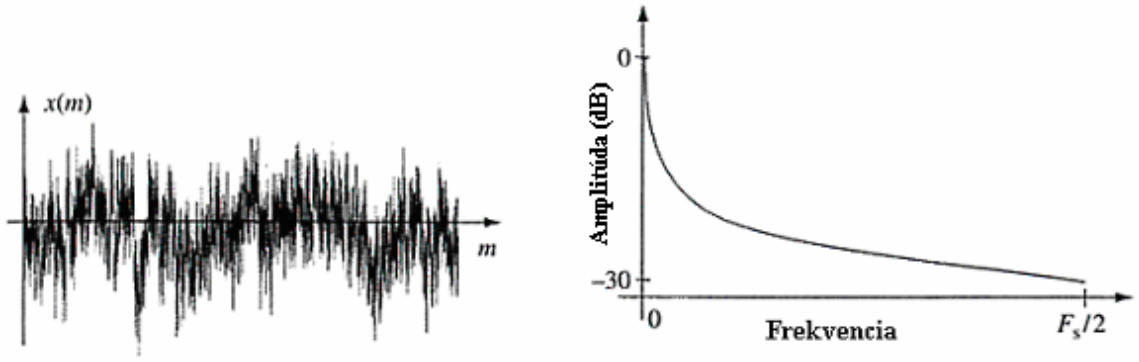
Výkonové spektrum bieleho šumu získané použitím Fourierovej transformácie na rovnicu (2.1) je dané

$$P_{NN}(f) = \int_{-\infty}^{\infty} r_{NN}(t) e^{-j2\pi ft} dt = \sigma^2 \tag{2.2}$$

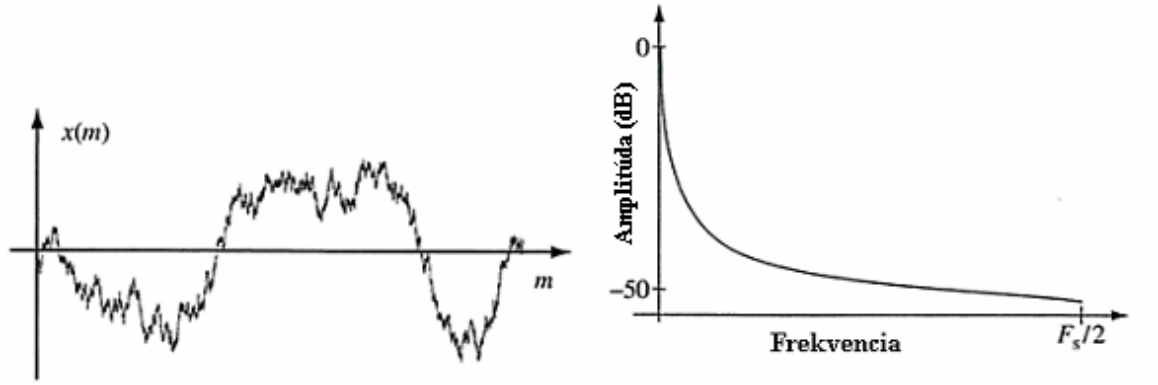
Rovnica (2.2) a obrázok 2.1(c) ukazujú, že biely šum má konštantné výkonové spektrum [14].

2.2 Farebný šum

Aj keď koncept bieleho šumu poskytuje primerane realistickú a matematicky vhodnú a užitočnú aproximáciu mnoho ďalších šumových procesov s ktorými sa stretávame v telekomunikačných systémoch nemajú charakter bieleho šumu. Termín „farebný šum“ poukazuje na akýkoľvek širokopásmový šum so spektrom, ktoré nemá charakter bieleho šumu. Napríklad väčšina audio-frekvenčných šumov, ako sú pohybujúce sa autá, šum z počítačových ventilátorov, elektrických vŕtačiek majú nízkofrekvenčné spektrum a nemajú tvar bieleho šumu. Rovnako biely šum prechádzajúci cez kanál sa stane „farebným“ podľa tvaru frekvenčnej odozvy kanálu. Používajú sa dve klasické variácie farebného šumu – „ružový šum“ a „hnedý šum“ (Obr. 2.3 a 2.4) [14].



Obr. 2.3 Ružový šumový signál (vľavo) a jeho amplitúdové spektrum (vpravo)



Obr. 2.4 Hnedý šumový signál (vľavo) a jeho amplitúdové spektrum (vpravo)

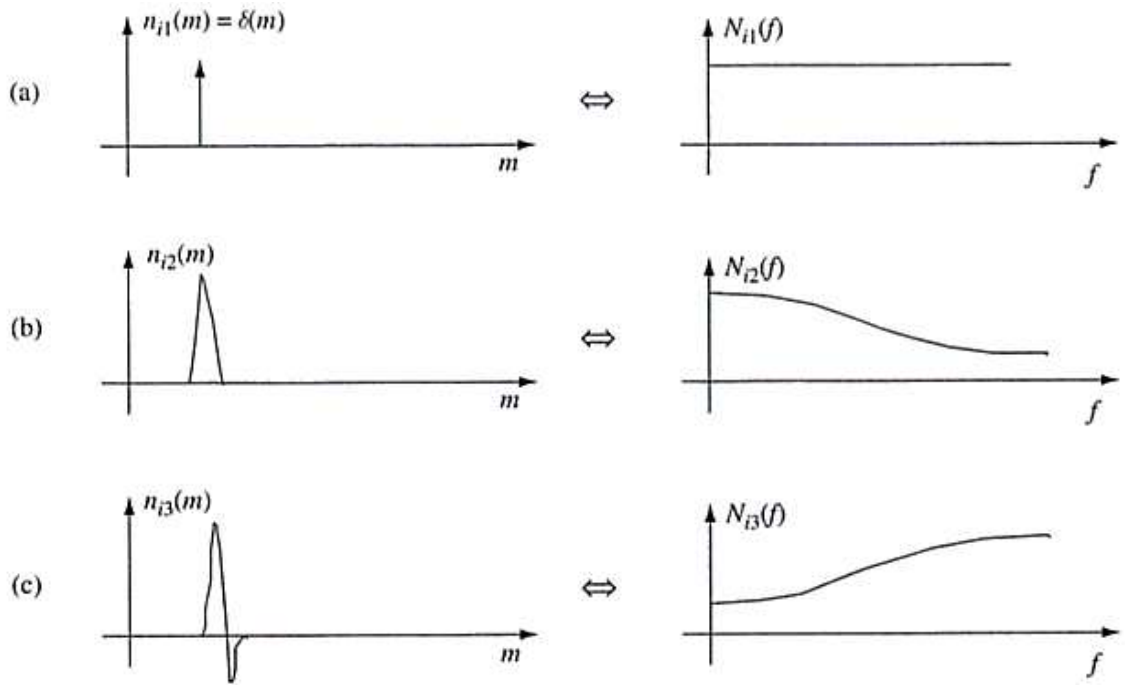
2.3 Impulzový šum

Impulzový šum pozostáva z náhodných krátko trvajúcich „on/off“ šumových impulzov, ktoré môžu byť spôsobené množstvom zdrojov ako je prepínací šum, elektromagnetické rušenie, nepriaznivé prostredie kanálu v komunikačných systémoch, výpadky alebo povrchová degradácia audio nahrávok atď.

Obrázok 2.5(a) predstavuje ideálny impulz a jeho frekvenčné spektrum. V komunikačných systémoch, reálny impulzový šum trvá dlhšie ako je dĺžka jednej vzorky. Napríklad v audio signáloch krátko trvajúce, ostré impulzy, ktoré trvajú dlhšie ako 3 ms (60 vzoriek pri 20 kHz vzorkovacej frekvencii) môžu byť považované za impulzový šum.

Obrázky 2.5(b) a 2.5(c) ilustrujú dva príklady krátko trvajúcich impulzov a ich jednotlivé spektrá.

V komunikačných systémoch vzniká impulzový šum v určitom bode v čase a priestore, a potom sa šíri cez kanál do prijímača. Prijímaný šum je časovo rozptýlený a tvarovaný kanálom a môže byť považovaný ako impulzová odozva kanálu. Vo všeobecnosti, charakteristiky komunikačného kanálu môžu byť lineárne alebo nelineárne, stacionárne alebo časovo premenlivé. Okrem toho veľké množstvo komunikačných systémov vykazuje nelineárne charakteristiky ako odozvu na veľké amplitúdové impulzy [14].



Obr. 2.5 Časový a frekvenčný priebeh: (a) ideálneho impulzu, (b) a (c) krátkodobého impulzu

2.4 Bodkový šum (Speckle noise)

Bodkový šum môžeme najčastejšie pozorovať v radarových snímacích systémoch, ale takisto sa môže objaviť v ktoromkoľvek diaľkovo snímaním obrazom založenom na koherentnom žiarení. Rovnako ako svetlo z lasera, vlny emitované aktívnymi senzorami putujú vo fáze a interferujú minimálne na ich ceste k cieľovej zóne. Po interakcii s cieľovou zónou, tieto dve vlny už nie sú vo fáze kvôli rozdielnej trase po ktorej putovali z cieľa alebo kvôli rozptylu. Ak sú vlny z radaru mimo fázy, môžu vytvárať biele a čierne pixely známe ako bodkovaný šum. Bodkovaný šum v radarových dátach by mal mať multiplikatívny chybový model a musí byť redukovaný predtým než sa dáta môžu použiť. V opačnom prípade sa šum pridruží do dát a degraduje tak kvalitu obrazu. Jednoduchý model pre bodkovaný šum má multiplikatívnu formu (2.1b) [12].

3 Rozdelenie filtrov

Existuje množstvo rozličných spôsobov ako sa môžu charakterizovať filtre:

- Linearita: z hľadiska linearity môžeme filtre deliť na lineárne a nelineárne. Lineárne filtre spĺňajú princíp superpozície, t.j. ak je vstupom lineárna kombinácia rozdielnych signálov, výstupom je rovnaká lineárna kombinácia zodpovedajúceho výstupného signálu. Lineárne filtre sa ďalej delia na nízkofrekvenčné a vysokofrekvenčné.

- Kauzalita: kauzálne filtre používajú predchádzajúce vzorky zo vstupných alebo výstupných signálov, zatiaľ čo nekauzálne filtre používajú budúce vstupné vzorky. Nekauzálne filtre sa môžu zmeniť na kauzálne pridaním oneskorenia.

- Variantnosť: časovo invariantné filtre sa v priebehu času nemenia. Naproti tomu adaptívne filtre sa naopak v čase menia.

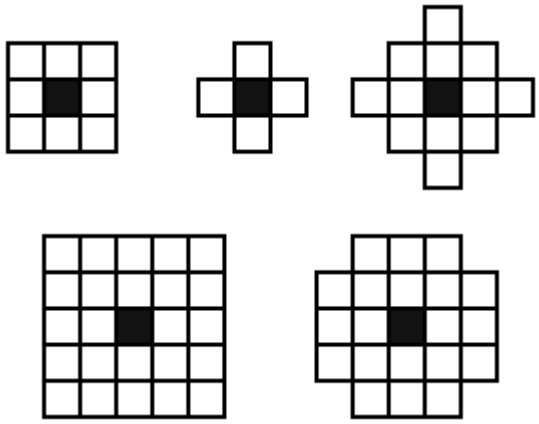
- Stabilita: filtre môžu byť stabilné alebo nestabilné. Stabilné filtre vytvárajú výstup, ktorý konverguje ku konštantnej hodnote v závislosti od času. Nestabilné filtre vytvárajú výstup ktorý diverguje.

- Impulzová odozva: FIR filtre (FIR – ang. Finite Impulse Response) – filtre s konečnou impulzovou odozvou používajú iba vstupný signál, zatiaľ čo IIR filtre (IIR – ang. Infinite Impulse Response) – filtre s nekonečnou impulzovou odozvou používajú vstupné aj predchádzajúce výstupne vzorky signálu. FIR filtre sú vždy stabilné, IIR môžu byť nestabilné.

Predchádzajúce rozdelenie patrí do časovej respektíve priestorovej oblasti. Existuje však aj filtrácia vo frekvenčnej oblasti. Signály sa konvertujú do frekvenčnej oblasti zvyčajne použitím Fourierovej transformácie. Fourierova transformácia konvertuje signálovú informáciu na amplitúdu a fázu pre každú frekvenciu. Často sa používa aj konverzia do výkonového spektra, t.j. amplitúdu každej frekvencie sa umocní na druhú [3].

4 Číslicová filtrácia obrazov

Číslicová filtrácia obrazov patrí medzi priestorové operácie. Nová hodnota určitého obrazového prvku je určená v závislosti na hodnotách určitého počtu okolitých prvkov. Filtrácia sa využíva v rade úloh ako je tzv. vyhladzovanie snímku, zvýrazňovanie a detekcia hrán, úprava výsledkov klasifikácie apod. Pri úpravách kvality obrazu sú dôležité rôzne typy filtrácie. Tieto filtrácie môžu odstrániť či zvýrazniť zrnitosť záznamu alebo zvýrazniť rôzne iné objekty v obraze. Jedná sa vlastne o úpravu intenzity obrazového bodu s väzbou na jeho okolie, pričom okolie bodu môže mať rôzny tvar, viď príklady uvedené na obr. 4.1. Sú tu uvedené symetrické okolia bodu, keď upravovaný bod (vyfarbený) je uprostred svojho okolia. Okrem symetrických okolí existujú i nesymetrické okolia bodu a adaptívne okolia bodu.



Obr. 4.1 Príklady rôznych typov symetrických okolí bodu

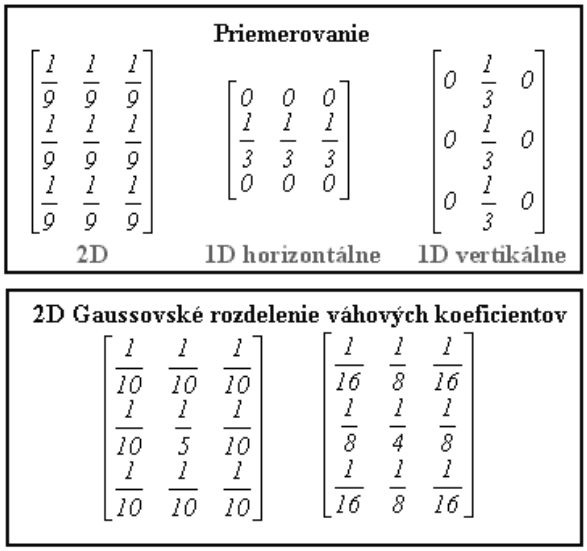
Filtre možno aplikovať buď na celý obraz, alebo len na dočasne vymedzenú oblasť obrazu. Pri vyhodnocovaní obrazu z vizualizačných experimentov je treba používať predovšetkým také filtre, ktoré neposúvajú obraz a nerozmazávajú hrany objektov. Hrany objektov možno zachovať vhodnou definíciou oblastí pre filtráciu alebo takisto zafarbením objektov (zafarbené objekty tak nemusia podliehať filtrácii).

V nasledujúcich kapitolách budú bližšie uvedené filtre typu FIR a IIR [5,8].

4.1 FIR filtre

4.1.1 Lineárne filtre

Pri lineárnych filtroch je intenzita upravovaného bodu rovná súčtu súčinov intenzít bodu v okolí a príslušných váhových koeficientov z matice váhových koeficientov. Lineárne filtre typu dolná priepust slúžia predovšetkým k odstráneniu vysokých priestorových frekvencií intenzít v obraze, čím dôjde k potlačeniu nežiaduceho šumu, ale takisto k potlačeniu detailov v obraze. Tieto filtre sa používajú predovšetkým k odstráneniu zrnitosti v obrazoch, akou je napr. zrnitosť v interferogramoch. Príklady matíc váhových koeficientov (s okolím 3x3 bodov) pre typické filtre typu dolná priepust sú uvedené na obr. 4.2. Vidíme tu matice váhových koeficientov pre tzv. priemerovanie v dvoch smeroch, priemerovanie v smere horizontálnom a priemerovanie v smere vertikálnom. Ďalej sú tu uvedené matice váhových koeficientov s gaussovým rozložením hodnôt. Pre filtre typu dolná priepust je charakteristické, že súčet váhových koeficientov v matici je rovný „1“. Lineárne filtre typu horná priepust umožňujú zvýrazniť detaily v obraze, ale zároveň dôjde k zvýrazneniu šumu. Pre filtre typu horná priepust je charakteristické, že súčet váhových koeficientov v matici je rovný „0“, viď matice váhových koeficientov uvedených na obr.4.3 [5].



Obr. 4.2 Príklady matíc váhových koeficientov pre filtre typu dolná priepust

Váhové koeficienty 2D filtra typu horná priepust

$$\begin{bmatrix} -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \\ -\frac{1}{9} & \frac{8}{9} & -\frac{1}{9} \\ -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \end{bmatrix}$$

Obr. 4.3 Príklad matice váhových koeficientov pre filtre typu horná priepust

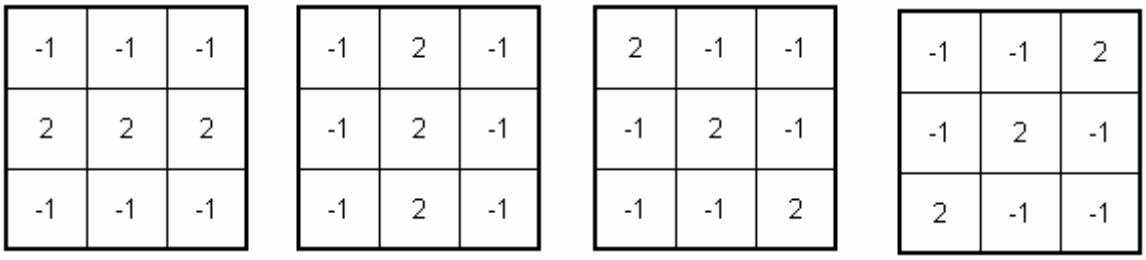
4.1.2 Nelineárne filtre

Nelineárne filtre nepočítajú intenzitu upravovaného bodu, ale vyberajú z okolia vhodnú hodnotu, ktorú potom dosadzujú do upravovaného bodu. Oproti lineárnym filtrom majú tu výhodu, že nepridávajú do obrazu žiadnu novú hodnotu intenzity. Napríklad filter minimum, označovaný ako erózia, vyberá z blízkeho okolia bod s minimálnou hodnotou intenzity a tú dosadí do upravovaného bodu. Filter minimum umožňuje eróziu tmavých prúžkov na svetlom pozadí či dilatáciu svetlých prúžkov na tmavom pozadí. Môže slúžiť k potlačeniu šumu v svetlej časti obrazu, alebo k zoslabeniu čiar v schémach. Filter medián vyberá z blízkeho okolia bod s strednou hodnotou intenzity, ktorú potom dosadí do upravovaného bodu. Tento filter je veľmi účinný pre potlačenie impulzového šumu. Jeho nevýhodou je, že vyhladzuje hrany objektov, čím mení ich tvary. Filter maximum, označovaný aj ako dilatácia, vyberá z blízkeho okolia bod s maximálnou hodnotou intenzity, ktorú potom dosadí do upravovaného bodu. Filter maximum umožňuje dilatáciu tmavých prúžkov na svetlom pozadí alebo eróziu svetlých prúžkov na tmavom pozadí. Môže slúžiť k potlačeniu šumu v tmavej časti obrazu, alebo takisto k zosilneniu čiar v schémach [5].

4.1.3 Nízkofrekvenčné filtre

Nízkofrekvenčné filtre potláčajú vysokofrekvenčnú informáciu v obraze. Redukujú odchýlky centrálneho pixelu od svojho okolia a produkujú obraz, ktorý je oproti originálnemu vyhladený - teda vždy určitým spôsobom zmenšujú rozptyl hodnôt pixelov v rámci filtrovacieho okna. Stupeň vyhladenia je priamo úmerný veľkosti použitého filtrovacieho okna - čím väčšie okno, tím väčšie vyhladenie. Nízkofrekvenčné filtre majú tendenciu redukovať rozsah výstupných hodnôt odtieňov šedej a často je nutné po filtrácii snímku pristúpiť k zvýrazneniu jeho kontrastu. Bežnými nízkofrekvenčnými filtrami sú napr. spriemerovací filter, Gaussov filter, Wienerov filter apod. [5].

Pre nízkofrekvenčnú filtráciu sa používajú aj filtre s rotujúcim oknom (Obr. 4.4.). V tomto prípade sa okolie filtrovaného obrazového prvku o veľkosti 3x3 pixely porovnáva s dopredu definovanými vzorovými filtrovacími oknami. Odpovedajúce si prvky okolia filtrovaného pixelu a okolia vzorového sa vzájomne vynásobia. Vzorové okolie s minimálnou výslednou hodnotou sa použije ako pri spriemerovacom filtre.

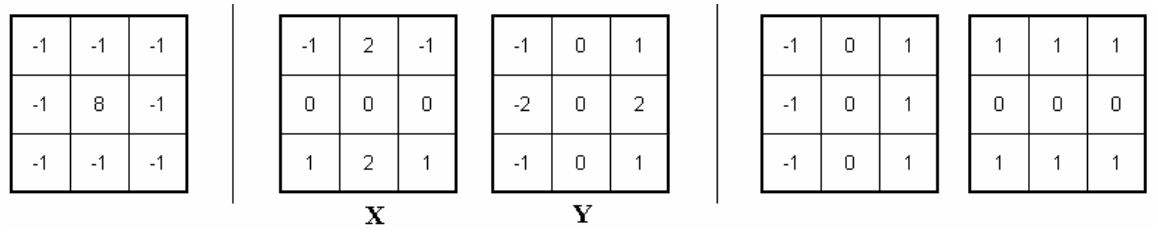


Obr. 4.4 Príklady filtra s rotujúcim oknom

4.1.4 Vysokofrekvenčné filtre

Obecnou funkciou vysokofrekvenčných filtrov je určitým spôsobom zvýšiť rozdiel hodnôt medzi filtrovaným centrálnym pixelom a jeho okolím. Tieto rozdiely reprezentujú predovšetkým hrany a línie. Hranou v obraze rozumieme hranicu medzi dvoma rôznymi povrchmi - napríklad okraj lesa. Na rozdiel od línie má hrana „nulovú“ šírku. Líniu potom v obraze reprezentujú predovšetkým komunikácie, vodné toky a podobne. Vysokofrekvenčné filtre obecne zdôrazňujú objekty, ktoré sú menšie než polovica filtrovacieho okna, širšie objekty potlačujú. Preto sa i tu používajú rôzne veľkosti filtrov.

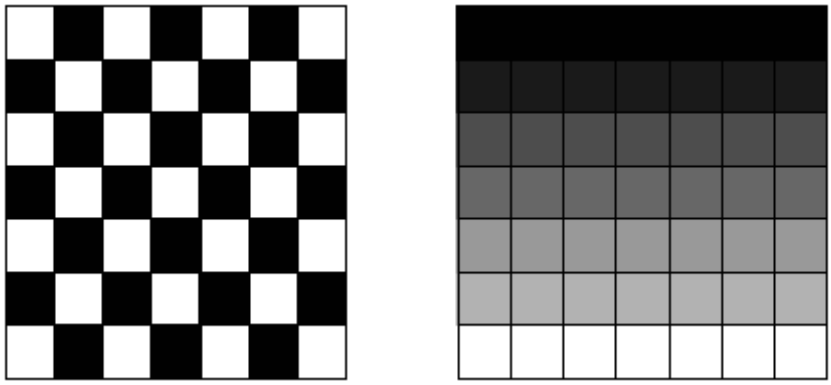
Ostriace filtre - Každý obraz možno považovať za prienik množín predstavujúcich vysokofrekvenčnú a nízkofrekvenčnú informáciu. Vysokofrekvenčnú informáciu možno získať odčítaním nízkofrekvenčnej informácie od pôvodného obrazu. Tieto filtre sa nazývajú diferenčné (High pass differential). Na tomto princípe je založený napríklad tzv. ostriaci filter (Edge sharpening filter). Najprv je pôvodný obraz filtrovaný spriemerovacím filtrom, ktorý potlačí línie a hrany. Takto vyhladený obraz je odčítaný od obrazu originálneho, čím získame obraz, v ktorom je vysokofrekvenčná informácia o hranách a líniách zachovaná. Nakoniec je tento obraz pričítaný k obrazu pôvodnému, čím sa získa výsledok, v ktorom sú hrany a línie „ostrejšie“ ohraničené voči okoliu. „Ostrenie“ (ang. sharpening) obrazu alebo detekcia a zvýraznenie línií a hrán majú široké uplatnenie v ČSS. Najpoužívanejšie filtre sú Laplaceovské filtre (Obr. 4.5a), Sobelov filter (Obr. 4.5a), Prewittov filter (Obr. 4.5a) atď. [8].



Obr. 4.5 Príklady masiek pre vysokofrekvenčú filtráciu a)Laplaceov filter, b)Sobelov filter a c)Prewitov filter

4.1.5 Priestorová frekvencia (spatial frequency)

Priestorová frekvencia charakterizuje relatívnu zmenu hodnoty daného pixelu voči hodnotám pixelov okolitých. Obrazový záznam obsahuje vysokofrekvenčnú a nízkofrekvenčnú priestorovú informáciu. Ich súhrn tvorí originálny obraz. Vysoké frekvencie popisujú veľké rozdiely v hodnotách pixelov pri prechode z jedného pixelu na druhý - predstavujú teda predovšetkým líniové prvky v obraze (riečna sieť, komunikácia, prírodné hranice). Nízke frekvencie popisujú postupné zmeny v hodnotách pixelov - vodné plochy, veľké polia, lesné komplexy atď. [8].



Obr. 4.6 Príklad vysokofrekvenčnej a nízkofrekvenčnej informácie v obraze

4.1.6 Princíp filtrácie obrazu

Je definované tzv. filtrovacie okno. Predstavuje ho štvorcová matica s nepárnym počtom riadkov a stĺpcov (napr. 3x3, 5x5 atď.). Každý pixel tohto okna obsahuje koeficient - váhu. Filtrovaný obraz je potom generovaný násobením každého koeficientu v okne hodnotou pixelu z originálneho snímku podľa súčasnej polohy okna. Výsledok je priradený centrálnemu pixelu vo filtrovanom snímku. Okno sa posúva po snímku po jednom pixele pohybom, ktorý býva označovaný ako „konvolúcia“ (obr. 4.7). Okrajové pixely po obvode snímku, sú v priebehu filtrovacej operácie replikované alebo je

výsledný filtrovaný snímok zmenšený o polovicu šírky filtrovacieho okna mínus 1 na každej strane [8]. Diskrétna konvolúcia má tvar:

$$g(x,y) = \sum_{(m,n\in 0)} \sum h(x-m)(y-n)\, f(m,n) \tag{4.1}$$

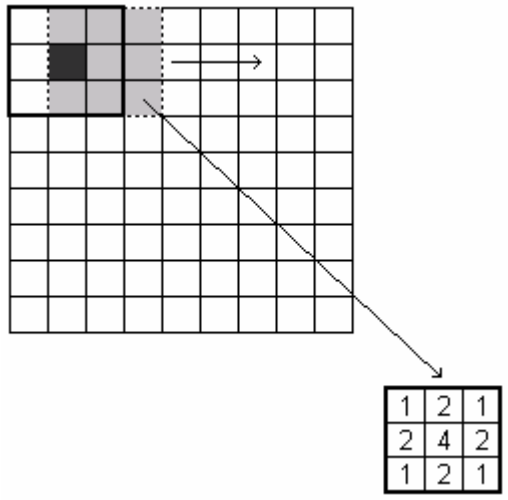
kde

„f“ je obrazová funkcia pôvodného obrazu,

„g“ je obrazová funkcia nového obrazu,

„h“ sa nazýva konvolučná maska alebo konvolučné jadro a udáva koeficienty jednotlivých bodov v okolí pixelu [2].

Najčastejšie sa používajú obdĺžnikové masky s nepárnym počtom riadkov a stĺpcov, pretože v tom prípade môže reprezentatívny bod ležať v strede masky [8].



Obr. 4.7 Princíp filtrácie obrazu

4.2 IIR filtre

V poslednom období s narastaním požiadaviek na spracovanie a kompresiu 2-D statických obrazov a 3-D video signálov v telekomunikáciách a iných multimediálnych aplikáciách sa stále viac uplatňujú viacrozmerné (m-D) adaptívne IIR filtre. Pri spracovaní 2-D obrazov je však nutné použiť 2-D verziu adaptívneho filtra, pretože filtrácia použitím 1-D adaptívneho filtra v horizontálnom, alebo vertikálnom smere nie je postačujúca.

Napríklad pri kompresii obrazov môže byť adaptívny algoritmus požitý na obnovu predikčných koeficientov adaptívneho diferenčne pulzne kódovaného modulačného

systému. M-D adaptívne filtre sa tiež často používajú pri potláčaní šumu v obrazoch, pretože využívajú štatistickú koreláciu medzi všetkými susednými obrazovými prvkami.

V adaptívnom spracovaní signálov je hlavne kvôli jednoduchej implementácii veľmi často používaný LMS algoritmus. Tento algoritmus je vhodný ako pre FIR, tak aj pre IIR filtre. V reálnych aplikáciách sú adaptívne IIR filtre preferované pred adaptívnymi FIR filtrami, pretože podstatne redukujú stupeň navrhovaného filtra. Avšak vo všeobecnosti ne je povrch MSE funkcie unimodálny a môže mať lokálne minimá, čo predstavuje istú nevýhodu takýchto filtrov. Práve nevhodná veľkosť konvergenčného parametra spôsobí nekonvergenciu algoritmu do jeho globálneho minima a dosiahnutý výsledok nemusí byť postačujúci. Adaptívne IIR filtre majú tiež ďalšie nevýhody medzi ktoré patrí vysoký stupeň výpočtovej náročnosti a malá rýchlosť konvergencie. Tieto nedostatky sa dajú redukovať pomocou algoritmov, ktoré sú výpočtovo menej náročné a zároveň rýchlejšie konvergujú [10].

Najznámejšie IIR filtre založené na LMS algoritme sú LMS filter, normalizovaný LMS filter, zjednodušený LMS filter a Zjednodušený normalizovaný LMS filter.

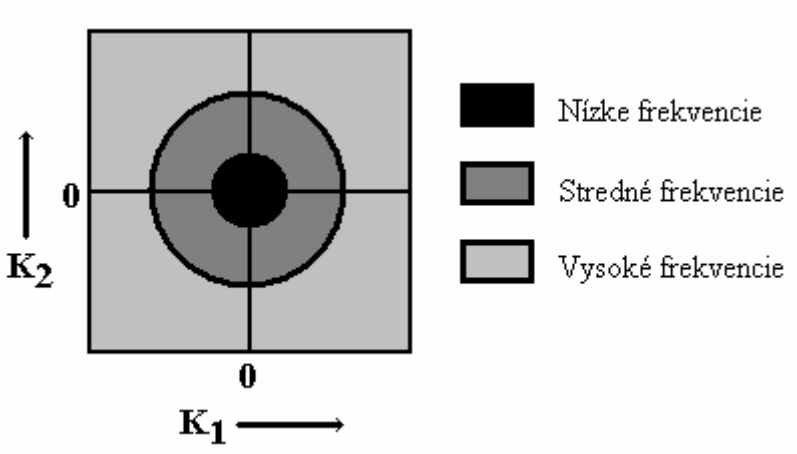
4.3 Fourierova transformácia

Prostredníctvom tzv. Fourierových transformácií možno priebeh akéhokoľvek jednorozmernej spojitej funkcie *f(x)* popísať pomocou série trigonometrických funkcií *sin* a *cos* s rôznymi amplitúdami a frekvenciami. Fourierova transformácia je veľmi dôležitým nástrojom v spracovaní obrazu, ktorý sa používa na rozloženie obrazu na sínusové a kosínusové zložky. Výstup z transformácie reprezentuje obraz vo frekvenčnej oblasti, zatiaľ čo vstupom je obraz v priestorovej oblasti. Vo Fourierovej transformácii každý bod predstavuje konkrétnu frekvenciu, ktorá sa nachádza v obraze v priestorovej oblasti. Využíva sa v širokom okruhu aplikácií ako je analýza obrazu, filtrácia obrazu, rekonštrukcia obrazu a kompresia obrazu. Pre štvorcový obraz o veľkosti NxN dvojdimenzionálna DFT sa dá vypočítať nasledujúcim vzťahom

$$F(k,l) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-t2\pi(\frac{ki}{N} + \frac{lj}{N})} \tag{4.2}$$

Snímok transformovaný do jednotlivých frekvencií je možné zobraziť v dvojrozmernom poli ako Fourierovo spektrum. Nízke frekvencie v originálnom obraze

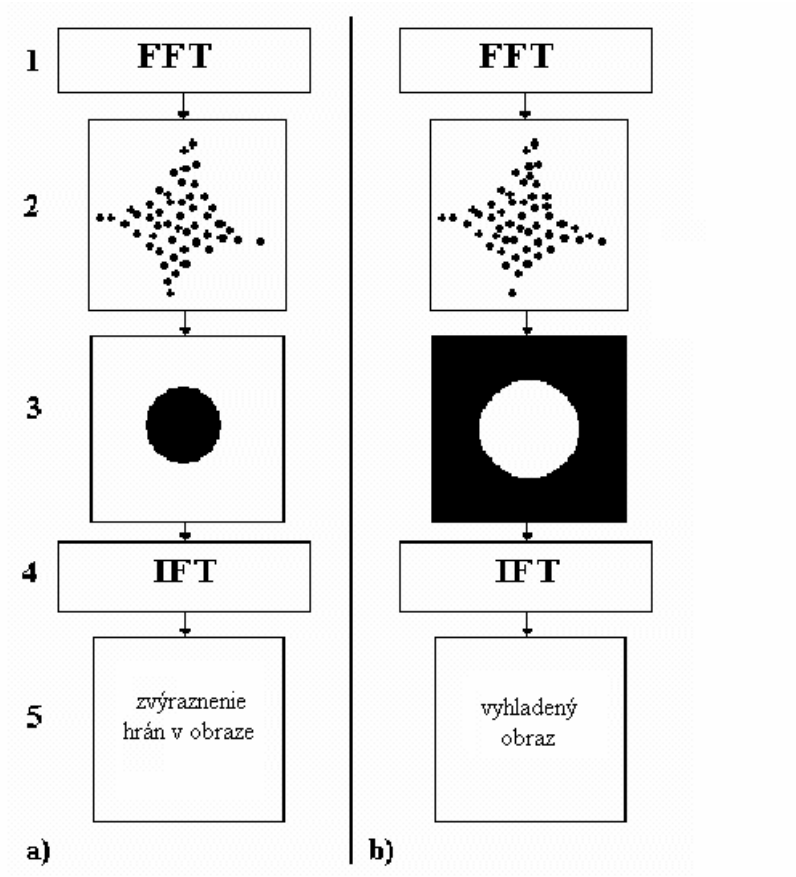
zodpovedajú strednej časti spektra, zatiaľ čo vysoké frekvencie sa posúvajú k jeho okrajom (obr. 1.8).



Obr. 4.8 Zobrazenie snímku vo Fourierovej transformácii

Zo spektra možno vyčítať orientáciu hrán a línií reprezentovaných danými frekvenciami. Hrany a línie orientované horizontálne v originálnom obraze sú vo Fourierovom spektre prezentované jeho vertikálnou zložkou a naopak. Stupeň šedi vo Fourierovom spektre vyjadruje početnosť výskytu danej frekvencie v obraze. Snímok transformovaný do jednotlivých frekvencií je možné zobraziť v dvojrozmernom poli ako Fourierovo spektrum. Uvedené frekvenčné spektrum možno tzv. inverznou Fourierovou transformáciou previesť späť do priestorového súradnicového systému a rekonštruovať tak pôvodný obraz. Pokiaľ pred touto inverznou transformáciou odstránime z Fourierovho spektra určité frekvencie, výsledný zrekonštruovaný obraz môže byť upravený podobne ako v prípade použitia nízko- či vysokofrekvenčného filtra. Fourierove transformácie možno využiť k potlačeniu šumu, k odstráneniu pruhov, ku detekcii línií, hrán a podobne. Na obr. 4.9 môžeme vidieť príklad využitia Fourierovej transformácie v praxi: 1.FFT – Fourierova transformácia, 2.Fourierovo spektrum 3.odfiltrovanie vhodných frekvencií pomocou masky, 4.IFT – inverzná Fourierova transformácia 5.výsledný upravený obraz.

Čierne plochy v použitej maske indikujú tie frekvencie, ktoré sú potlačené (odfiltrované) [8].



Obr. 4.9 Využitie Fourierovej transformácie pre vysokofrekvenčnú (a) a nízkofrekvenčnú (b) filtráciu obrazu

4.4 Filtre použité v programe

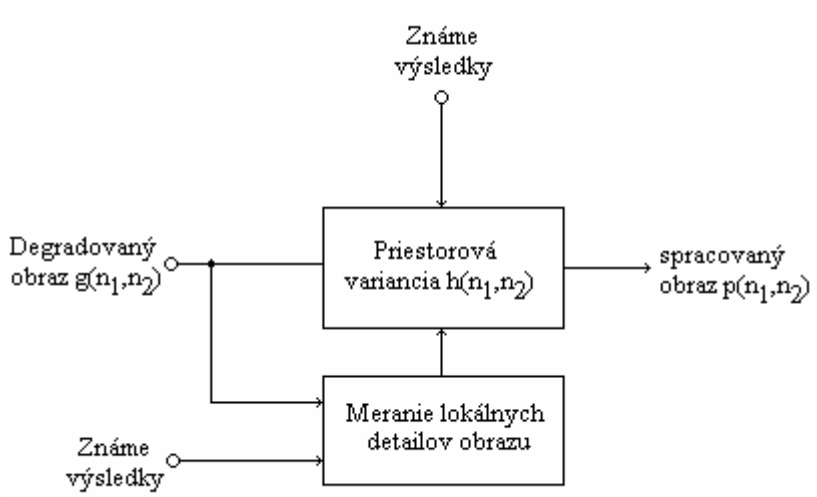
4.4.1 Číslicové filtre potláčajúce aditívny Gaussov šum

4.4.1.1 Adaptívny Wienerov filter

Väčšina adaptívnych obnovovacích algoritmov pre redukciu aditívneho šumu v obraze môže byť reprezentovaná systémom, ktorý je na obrázku 4.10. Z degradovaného obrazu a predošlých znalostí lokálnych štatistík obrazu sa určujú hodnoty lokálnych detailov z nezašumeného obrazu. Jedným z týchto hodnôt je lokálna disperzia. Priestorovo premenlivý filter $h(n_1,n_2)$ je funkciou lokálnych detailov obrazu a predošlej a znalosti lokálnych štatistík obrazu.

Priestorovo premenlivý filter je aplikovaný na degradovaný obraz v lokálnych regiónoch z ktorých bol filter navrhnutý. Keď je šum širokopásmový, priestorová disperzia má dolno-priepustný charakter. V regiónoch z malými detailmi, kde je šum viac

viditeľný ako v regiónoch z veľkými detailmi sa na potlačenie šumu používa dolno-priepustný filter s veľkou strmosťou. Vzhľadom na to, že v regiónoch z malými detailmi je prezentovaná malá signálová zmena, aj veľké množstvo dolno-priepustnej filtrácie významne neovplyvní zložky signálu. V regiónoch z veľkými detailmi ako sú napríklad hrany, kde sa nachádza veľké množstvo zložiek sa používa len malá dolno-priepustná filtrácia aby sa neskreslili zložky signálu. Toto neodstráni veľké množstvo šumu, ale rovnaký šum je viditeľný menej v regiónoch z veľkými detailmi ako z malými.



Obr. 4.10 Typický adaptívny obnovovací systém na redukciu aditívneho šumu

Je Možné vytvoriť množstvo rozličných algoritmov v závislosti na tom aké špecifické meranie použijeme na reprezentáciu lokálnych detailov obrazu, ako je určená priestorová disperzia ako funkcia lokálnych detailov a aké sú k dispozícii predchádzajúce znalosti. Wienerov filter požaduje poznať strednú hodnotou signálu m_f, strednú hodnotu šumu m_υ, Výkonové spektrum signálu $P_f(\omega_1, \omega_1)$ a Výkonové spektrum šumu $P_\upsilon(\omega_1, \omega_1)$. Namiesto toho aby sme získavali pevné m_f, m_υ, $P_f(\omega_1, \omega_1)$ a $P_\upsilon(\omega_1, \omega_1)$ pre celý obraz, môžeme ich odhadnúť lokálne. To sa priblíži k výsledku Wienerovho filtra z priestorovou varianciou. Dokonca i počas tohto priblíženia sú možné viaceré variácie v závislosti na tom ako sú lokálne odhadnuté m_f, m_υ, $P_f(\omega_1, \omega_1)$ a $P_\upsilon(\omega_1, \omega_1)$ a ako je implementovaný výsledok priestorovo-premenlivého Wienerovho filtra.

Predpoklad: aditívny šum $\upsilon(n_1,n_2)$ má nulový priemer a biely šum disperziu σ^2_υ. Jeho výkonové spektrum je potom dané vzťahom

$$P_v(\varpi_1, \omega_2) = \sigma^2{}_v \tag{4.3}$$

Uvažujme malý región v ktorom budeme predpokladať, že signál $f(n_1,n_2)$ sa nemení. V lokálnych regiónoch je signál $f(n_1,n_2)$ modelovaný

$$f(n_1, n_2) = m_f + \sigma_f w(n_1, n_2) \tag{4.4}$$

kde m_f a ňσ$_f$ sú lokálny priemer a štandardná odchýlka $f(n_1,n_2)$ a $w(n_1,n_2)$ je biely šum s nulovou strednou hodnotou a jednotkovou disperziou.

Vo vzťahu (4.4) signál $f(n_1,n_2)$ je modelovaný sumou priestorovej disperzie lokálnej strednej hodnoty m_f a bieleho šumu s priestorovou disperziou lokálnej odchýlky σ^2_f. V lokálnych regiónoch je potom Wienerov filter $H(\omega1,\omega1)$ daný vzťahom

$$H(\varpi_1,\varpi_2) = \frac{P_f(\varpi_1, \omega_2)}{P_f(\varpi_1, \omega_2) + P_v(\varpi_1, \omega_2)} = \frac{\sigma^2{}_f}{\sigma^2{}_f + \sigma^2{}_v} \tag{4.5}$$

Z tohto vzťahu je možné získať $h(n_1,n_2)$, čo je váhová impulzová odozva

$$h(n_1, n_2) = \frac{\sigma^2{}_f}{\sigma^2{}_f + \sigma^2{}_v}\delta(n_1, n_2) \tag{4.6}$$

Spracovávaný obraz $p(n_1,n_2)$ vo vnútri lokálneho regiónu je možné z predošlého vzťahu vyjadriť ako

$$p(n_1, n_2) = m_f + (g(n_1, n_2) - m_f) * \frac{\sigma^2{}_f}{\sigma^2{}_f + \sigma^2{}_v}\delta(n_1, n_2) =$$

$$= m_f + \frac{\sigma^2{}_f}{\sigma^2{}_f + \sigma^2{}_v}(g(n_1, n_2) - m_f) \tag{4.7}$$

Ak predpokladáme, že m_f a σ^2_f sa menia pre každý pixel potom je možné prepísať rovnicu (4.7) na tvar

$$p(n_1, n_2) = m_f(n_1, n_2) + \frac{\sigma^2{}_f(n_1, n_2)}{\sigma^2{}_f(n_1, n_2) + \sigma^2{}_v}(g(n_1, n_2) - m_f(n_1, n_2)) \tag{4.8}$$

Táto rovnica je jadrom pre algoritmus vyvinutý Leeom [11].

Algoritmus založený na (4.8) je braný ako špeciálny prípad dvojkanálového procesu. V dvojkanálových procesoch musí byť obraz, ktorý chceme spracovať do dvoch zložiek → lokálny priemer $m_f(n_1,n_2)$ a lokálny rozdiel $g(n_1,n_2)$. Lokálny priemer a lokálny rozdiel sa upravujú oddelene a výsledky sa potom skombinujú. V prípade (4.8) lokálny priemer zostáva nemenný zatiaľ čo lokálny rozdiel sa mení v závislosti na relatívnej amplitúde σ^2_f a σ^2_υ. Ak σ^2_f je oveľa väčšia ako σ^2_υ lokálny rozdiel $g(n_1,n_2)$ nadobúda predovšetkým hodnoty $f(n_1,n_2)$ a lokálny rozdiel $g(n_1,n_2)$ nie je tlmený. V tomto prípade $p(n_1,n_2)$ je približne rovnaký ako $g(n_1,n_2)$. Ak σ^2_f je oveľa menšia ako σ^2_υ lokálny rozdiel $g(n_1,n_2)$ nadobúda predovšetkým hodnoty $\upsilon(n_1,n_2)$ a lokálny rozdiel $g(n_1,n_2)$ je podstatne tlmený. V tomto prípade $p(n_1,n_2)$ je približne rovný m_f.

Ak je brané do úvahy, že m_f je identické k m_g keď m_υ je nulový, $m_f(n_1,n_2)$ v (4.8) sa odhadne z $g(n_1,n_2)$ vzťahom

$$\hat{m}_f(n_1,n_2) = \frac{1}{(2M+1)^2}\sum_{k_1=n_1-M}^{n_1+M}\sum_{k_2=n_2-M}^{n_2+M} g(k_1,k_2) \tag{4.9}$$

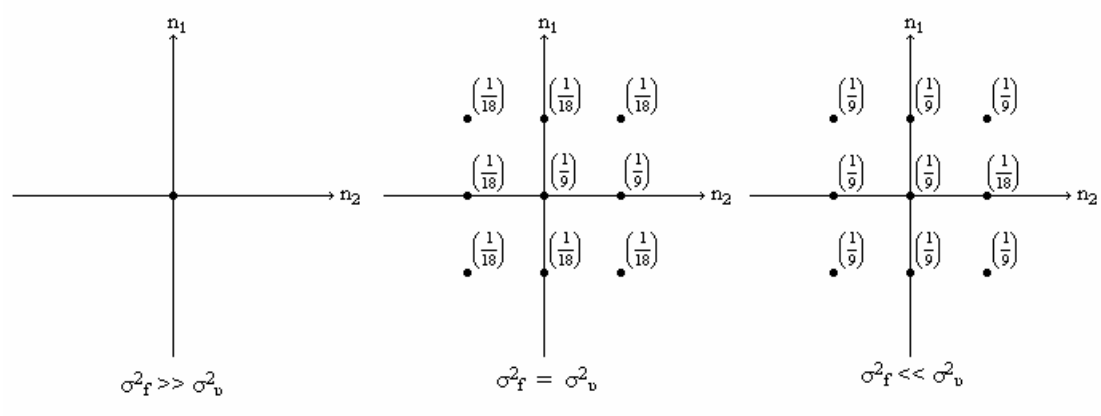
kde $(2M+1)^2$ je počet pixelov v lokálnom regióne použitých na odhad. Dosadenie (4.10) do (4.8)vedie ku tvaru

$$p(n_1,n_2) = g(n_1,n_2) * h(n_1,n_2) \tag{4.11a}$$

kde

$$h(n_1,n_2)\begin{cases} \dfrac{\sigma^2{}_f + \dfrac{\sigma^2{}_\upsilon}{(2M+1)^2}}{\sigma^2{}_f + \sigma^2{}_\upsilon}, & n_1 = n_2 = 0 \\ \dfrac{\dfrac{\sigma^2{}_\upsilon}{(2M+1)^2}}{\sigma^2{}_f + \sigma^2{}_\upsilon}, & -M \le n_1 \le M, -M \le n2 \le M, \\ & \text{s výnimkou } n_1 = n_2 = 0 \\ 0 & \text{inde} \end{cases} \tag{4.12b}$$

Filtre $h(n_1,n_2)$ keď $\sigma^2_f >> \sigma^2_v$, $\sigma^2_f = \sigma^2_v$ a $\sigma^2_f << \sigma^2_v$, sú zobrazené na obrázku 4.11 pre M = 1. Na tomto obrázku vidno, že σ^2_f relatívne klesá ku σ^2_v a prebieha väčšie vyhľadenie šumu [11].



Obr. 4.11 Impulzová odozva priestorovo variantného obnovovacieho

4.4.1.2 Spriemerovací filter

Spriemerovací filter je jednoduchá, intuitívna a ľahko implementovateľná metóda na vyhladenie obrazov ako aj na redukciu šumu v obrazoch. Hlavná myšlienka spriemerovacieho filtrovania je jednoducho nahradiť každý pixel v obraze s priemernou („strednou“) hodnotou jeho susedov, vrátene seba samého. Toto spôsobí elimináciu pixelových hodnôt, ktoré nie sú typické svojmu okoliu. Spriemerovací filter zvyčajne funguje na princípe konvolučného filtrovania. Ako každá iná konvolúcia, spriemerovací filter je založený na okolí masky, ktorá reprezentuje tvar a veľkosť okolia, ktoré sa použije na kalkuláciu strednej hodnoty.

Najčastejšie sa využívajú okná s veľkosťou 3x3 ako je to zobrazené vo vzťahu 3.9, ale takisto sa môžu použiť aj väčšie okná (napríklad štvorcové masky o veľkosti 5x5, 7x7, 9x9 atď.), ktoré však spôsobia väčšie rozmazanie obrazu [7]. Vyhladzovací efekt teda závisí výlučne na voľbe veľkosti okna. S veľkými oknami vyhladzovací efekt závisí aj na hodnotách, ktoré ležia ďalej od centrálneho pixelu, takže voľba veľkosti masky je kompromisom medzi požadovanou redukciou šumu a zachovaním zaostreného obrázku [9]. Poznamenajme, že maska s malým oknom sa môže použiť viackrát na to aby sa dosiahol podobný, ale nie identický efekt ako by sa dosiahlo použitím veľkej masky jeden krát.

Jednoduché štvorcové okno o veľkosti 3x3 je definované ako

$$C = \frac{1}{k}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.13}$$

k = 9

1/k – váhový faktor, ktorý môžeme aplikovať aby sme sa vyhli všeobecnému tlmeniu alebo zosilneniu dát

C – konvolučná matica prvkov

Vyfiltrovanú hodnotu potom môžeme vypočítať použitím konvolúcie nasledovným spôsobom

$$y(m,n) = \frac{\sum_{i=1}^{N} c(i) * x(i)}{N} \tag{4.14}$$

y(m,n) – vyfiltrovaná hodnota pixelu

x(i) – i-ta hodnota zašumeného obrazu v rámci masky

c(i) – i-ta hodnota z konvolučnej masky

N – počet prvkov masky

4.4.3 Číslicové filtre potláčajúce impulzový šum

4.4.3.1 Zložkový MF

Vďaka schopnosti zachovať vyhladené okraje, je mediánový filter často používaný pre šedé obrazy. Použiť mediánový operátor na farebné obrazy nie je jednoduché, pretože farbený vektor musí byť zoradený podľa určitého zoradenia.

Mediánový filter je nelineárny operátor, ktorý zoraďuje obrazové prvky v lokálnom okne podľa veľkosti ich vektora a nahradzuje hodnotu obrazového prvku vo výslednom obraze strednou hodnotou v tomto zoradení.

Ak je mediánový filter aplikovaný ku každej farebnej zložke oddelene, jedná sa o zložkový MF. Pri tomto type filtra dochádza k farebnému skresleniu vyhladených okrajov hrán a vytvorí sa hodnota, ktorá sa nenachádza v pôvodnom obraze. Pre výstup zložkového MF platí vzťah

$$y_c = med(x_c) \tag{4.15}$$

kde c predstavuje farebný kanál - červený, zelený alebo modrý a pre x_c platí $x_{c(1)} \le x_{c(2)} \le ... \le x_{c(N)}$

4.4.3.2 Vektorový mediánový filter s marginálnym zoradením

Pre súbor N vektorov $\boldsymbol{x_1,\ldots,x_N}$ vo vnútri okna a vektorovú normu $\|.\|_L$, je vektorový mediánový filter VM definovaný rovnicou

$$VM\{\boldsymbol{x_1}, \boldsymbol{x_2},..., \boldsymbol{x_N}\} = \boldsymbol{x_{VM}}$$

kde

$$\boldsymbol{x_{VM}} \in \{\boldsymbol{x_1}, \boldsymbol{x_2},..., \boldsymbol{x_N}\}$$

a

$$\sum_{i=1}^{N} \|\boldsymbol{x_{VM}} - \boldsymbol{x_i}\|_L \le \sum_{i=1}^{N} \|\boldsymbol{x_j} - \boldsymbol{x_i}\|_L , \quad j = 1,2,..., N . \tag{4.16}$$

Výsledkom tejto filtrácie je ten vektor v rámci okna, ktorý minimalizuje sumu vzdialeností k iným vektorom, podľa použitej ***L***-normy [13].

Dáta môžu byť zoraďované podľa hodnôt buď vzostupne alebo zostupne. Potom pre usporiadané vzorky $\mathbf{x}_i = (x_{i1}, x_{i2},..., x_{ip})^T$ platí

$$x_{j(1)} \le x_{j(2)} \le ... \le x_{j(N)} \qquad j = 1,\ldots, p, \tag{4.17}$$

kde p predstavuje rozmer priestoru, v našom prípade $p = 3$.

$x_{1(1)}, x_{2(1)},..., x_{p(1)}$ predstavujú minimálne prvky každého kanála a $x_{1(N)}, x_{2(N)},..., x_{p(N)}$ maximálne prvky každého kanála. Po zoradení (i_1,i_2,…,i_p) získavame nasledujúci vektor s rozmerom (p x 1)

$$\mathbf{x}_{(i1,i2,...,ip)} = \left(x_{1(i1)}, x_{2(i2)},..., x_{p(ip)}\right)^T , \; 1 \le i_j \le \mathrm{N}; \; j = 1,\ldots, p \tag{4.18}$$

Mediánom M-zoradenia je vektor

$$\mathbf{x}_{med} = \left[med\{x_{11},..., x_{N1}\},..., med\{x_{1p},..., x_{Np}\}\right]^T . \tag{4.19}$$

Pre výstup L-fitra s marginálnym zoradením dát platí

$$y = \sum_{i1=1}^{N} ... \sum_{ip}^{N} W_{i1,...,ip} x_{i1,...,ip} . \tag{4.20}$$

4.4.3.3 Vektorový mediánový filter s podmieneným zoraďovaním v HSV priestore

Vektorový mediánový filter (VMF – ang Vector Median Filter) patrí k základným filtrom pre potlačenie nelineárneho šumu. Má jednu typickú charakteristiku a to, že jeho výstup produkuje hodnoty len zo zašumeného farebného obrazu. Pre zoraďovanie a meranie vektorovej vzdialenosti farebných dát za zvyčajne používa norma L2 ($_{L2}$VMF). Hľadanie vektorového mediánu vo filtrovanom okne celého obrazu je veľmi časovo náročné. Z tohto uhla pohľadu, vektorové zoraďovanie môže byť modifikované napríklad redukovaným zoraďovaním, podraďovaním, podmieneným zoraďovaným a podobne.

Vo všeobecnosti podmienené zoraďovanie zoraďuje vektory podľa hodnoty z jednej zo zložiek. Popritom vektory, ktoré majú rovnakú hodnotu pre prvú zložku sú zoraďované podľa ďalšej zložky atď. VMF podmienené zoraďovanie v HSV farebnom priestore (obr. 4.12) používa transformáciu RGB farebných elementov do HSV. Chromatičnosť c každého pixelu je navrhnutá ako c(h,s,v) a tieto vektory sú zoradené nasledovne:

(1) Vektory sú zoradené vzostupne podľa parametra „v“ (Value – hodnota)

(2) Vektory, ktoré majú rovnakú hodnotu „v“ sú zoradené zostupne podľa parametra s (Saturation – sýtosť farby)

(3) Nakoniec vektory, ktoré majú rovnakú hodnotu „s“ sú zoradené vzostupne podľa parametra h (Hue – odtieň).

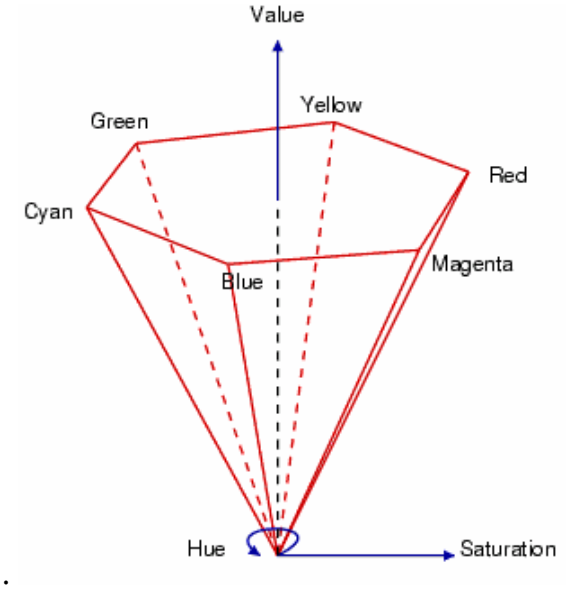
Pre matematickú interpretáciu podmieneného zoraďovania v HSV farebnom priestore sa stanovili dva operátory „$<_c$“ a „$=_c$“. Oba operátory slúžia pre porovnanie dvoch farieb a sú dané ako

$$c(h_i, s_i, v_i) <_c c(h_k, s_k, v_k) \Leftrightarrow (v_i < v_k) \vee (v_i = v_k \wedge s_i > s_k) \vee$$

$$\vee (v_i = v_k \wedge s_i = s_k \wedge h_i < k_k) \qquad (4.21a)$$

$$c(h_i, s_i, v_i) =_c c(h_k, s_k, v_k) \Leftrightarrow (v_i = v_k \wedge s_i = s_k \wedge h_i = k_k) \qquad (4.21b)$$

Rovnice (4.36a-b) definujú VMF s podmieneným zoraďovaním v HSV farebnom priestore (C $_{HSV}$VMF), kde stredný farebný vektor je mediánový vektor a predstavuje výstup z C $_{HSV}$VMF [13].



Obr. 4.12 Model farebného priestoru HSV

4.4.3.4 L-filtre

L-filtre sú veľmi účinné pri obnovovaní signálov, ktoré sú poškodené šumom. Avšak ich dizajn v real-timeových aplikáciách je prakticky nemožný, nakoľko výpočet optimálnych koeficientov L-filtra je veľmi časovo náročný. Ako je to pri väčšine takýchto filtrov, majú flexibilnejšiu a komplexnejšiu štruktúru [13].

Princíp L-filtrov: vstupné dáta, na ktoré je aplikovaná poriadková štatistika sa vynásobia z váhovými koeficientami. Výstupom takéhoto filtra je nasledovný vzťah

$$y_c = x_{ci} * w_i \tag{4.22}$$

kde $x_{ci} = (x_{c1}, x_{c2}, ..., x_{cn})$ je vektor vstupných dát (c predstavuje kanál) a w_i je vektor váhových koeficientov.

4.4.4 Číslicové filtre potláčajúce bodkový šum

4.4.4.1 Kuanov filter

Kuanov filter je založený na predpoklade, že stredná kvadratická chyba (MSE - mean square error) je minimálna (MMSE) [15]. Výstup lineárneho MMSE filtra je daný vzťahom

$$\hat{R}(t) = I(t).W(t) + \bar{I}(t).(1 - W(t)) \tag{4.23}$$

kde váhová funkcia W je daná ako

$$W(t) = \left[1 - C^2{}_u / C^2{}_I\right] / \left[1 + C^2{}_u\right]$$

a $C_I = \sigma_I / \bar{I}$ je koeficient šumovej disperzie

4.4.4.2 Frostov filter

Frostov filter nahrádza pixely z originálneho obrazu hodnotami váhovej sumy NxN pohyblivej masky. Váhové prvky sa zmenšujú so vzdialenosťou od centrálneho pixelu a naopak, zvyšujú sa k centrálnemu pixelu podľa toho ako sa zvyšuje disperzia v rámci masky [15]. Odhad sa robí na základe MMSE a multiplikatívnom šumovom modeli.

Frostov filter dáva pre každý pixel z filtrovacieho okna váhovú hodnotu, ktorá je daná

$$M = e^{-A*T} \tag{4.24}$$

kde $A = D * (V / I^2)$

T – vzdialenosť medzi daným pixelom a centrálnym pixelom

V – disperzia filtrovaného okna

I – priemerná hodnota filtrovaného okna

Frostov filter potom môžeme popísať nasledovne

$$R = \frac{\sum_{i=1}^{n} (Pi * Mi)}{\sum_{i=1}^{n} Mi} \tag{4.25}$$

kde Pi je šedá hodnota pixelu v rámci okna.

4.4.4.3 Leeov filter

Leeov filter používa štatistickú distribúciu DN (aktuálna hodnota pixelu z originálneho obrazu) hodnôt v rámci masky na odhad hodnôt vyfiltrovaných pixelov. Tento filter využíva Gaussovo rozdelenie pre šum v obraze. Leeov filter je založený na predpoklade, že stredná hodnota a disperzia pixelov celého obrazu je rovná lokálnej strednej hodnote a disperzii všetkých pixelov v rámci masky [15]. Na Leeov filter môžeme použiť rovnaký vzorec ako na výpočet Kuanovho filtra (4.23), len zmeníme váhovú funkciu W

$$W(t) = \left[1 - C^2{}_u / C^2{}_I\right] \tag{4.26}$$

4.4.4.4 Lopesov filter

Známy aj pod menom MAP filter bo prvýkrát predložený Kuanom. Kuan predpokladal, že rozloženie hustoty pravdepodobnosti (PDF – Probability Density Function) nezašumeného obrazu má Gaussovo rozloženie. Jeho predpoklad však nezodpovedá skutočnosti, a tak Lopes tento filter neskôr upravil. Lopes predpokladal, že PDF nezašumeného obrazu a šumu samotného majú Gama rozloženie a určil dva prahy pre tento filter [15]. Filter môžeme popísať rovnicou

$$R = \begin{cases} I & Ci \le Cu \\ B * I + \sqrt{D}/(2 * \alpha) & Cu \le Ci < C_{\max} \\ CP & Ci \ge C_{\max} \end{cases} \tag{4.27}$$

$$Cu = 1/\sqrt{NLook}$$

$$Ci = \sqrt{VAR}/I$$

$$C\max = \sqrt{2} * Cu$$

$$\alpha = (1 + Cu^2)/(Ci^2 - Cu^2)$$

$$B = \alpha - NLook - 1$$

$$D = I^2 * B^2 + 4 * \alpha * NLook * I * CP$$

kde „NLook“ je počet „lookov“, „VAR“ je disperzia, „I“ je stredná hodnota filtrovaného okna, „CP“ je šedá hodnota centrálneho pixelu a „R“ je vyfiltrovaná šedá hodnota.

Výrazom „šedá“ hodnota sa označujú štatistické informácie o pixele ako je priemerná hodnota alebo disperzia v danom okne.

5 Opis programového prostredia Java

Java je pomerne nový, ale silný programovací jazyk, ktorý sa vo svete začína razantne presadzovať. Tento jazyk má ohromný dopad z viacerých dôvodov. Jedným z najvýznamnejších je jeho široká nezávislosť na platforme, čo znamená, že aplikácie napísané pre jeden počítač veľmi pravdepodobne pobežia nezmenené i na inom počítači. S dosiahnutím skutočnej nezávislosti na platforme sú stále spojené začiatočné problémy a konečný cieľ, ktorým je úplná nezávislosť na platforme, nebol ešte dosiahnutý. Bol však už vykonaný významný pokrok a pokiaľ niektoré návrhy firmy Microsoft nebudú prijaté, mohla by sa nezávislosť na platforme skoro stáť skutočnosťou.

Programátor teda bude môcť napísať jedinú aplikáciu, ktorú je možno spustiť na rôznych typoch počítačov v rámci jednej organizácie, či už sa jedná o osobné počítače, počítače Macintosh alebo unixové pracovné stanice. Vďaka filozofii „raz napísaný program beží všade“ nie je organizácia viazaná na jeden typ počítačového hardwaru.

Java je objektovo orientovaný jazyk, čo je jeho ďalšou výhodou. Ako si neskôr ukážeme, objektovo orientované programovacie jazyky uľahčujú vytvorenie a udržovanie veľkých programov tým, že zapuzdria dáta a metódy pre ich zmenu do oddelených jednotiek, ktoré sa nazývajú *objekty*. Pretože objekty na seba vzájomne pôsobia iba prostredníctvom dobre nadefinovaného rozhrania, nežiadúce výsledky môžu byť znížené na minimum. Takže objekty možno oveľa jednoduchšie znovu použiť v iných programoch.

Ďalšou výhodou Javy je jej relatívna jednoduchosť. Samotný jazyk Javy má syntax jednoduchšiu ako jazyky C a C++ (z ktorých vychádza), čo uľahčuje jeho zvládnutie. Mnohé súčasti jazyka C (ako napríklad práca s ukazovateľmi), ktoré bývajú zdrojom častých problémov, v Jave vôbec neexistujú. Iné sú potom značne zjednodušené alebo spracované automaticky. Ako príklad môže slúžiť prideľovanie a uvoľňovanie pamäti v jazyku C, kde sa často robia chyby. V Jave sa prideľovanie a uvoľňovanie pamäti deje automaticky. Nie je to samotný jazyk Javy, čo spôsobuje problémy, ale požívanie obrovského počtu rôznych tried, ktoré prichádzajú spolu s Javou v jej aplikačnom programovaco rozhraní (API – z angl. Application Progamming Inteface).

Štandardný jazyk Javy naviac obsahuje grafiku nezávislú na zariadení. Zatiaľ čo grafický výstup možno vytvoriť i v iných jazykoch ako je napríklad C alebo Fortran, výsledný grafický kód nie je štandardný a líši sa na rôznych počítačoch a niekedy dokonca na rovnakom počítači podľa rôznych zariadení. Napríklad kód pre tlač grafiky na

obrazovku sa bude líšiť od kódu pre tlač grafiky na tlačiarni. Na druhej strane má Java grafiku nezávislú na zariadení zabudovanú priamo do jazyka. Program, ktorý vytvorí graf na jednom počítači, vytvorí ten istý graf i na inom počítači, a to nezávisle na type a operačnom systéme tohto počítača. A v neposlednom rade je obrovskou výhodou Javy jej voľná dostupnosť. Vývojový balík Java Development Kit (JDK) je možné si zadarmo stiahnuť zo serveru „*http://java.sun.com*“. Tento balík obsahuje prekladač Javy (*javac*), interpret pracujúci v dobe behu programu (*java*), ladiaci program (*jdb* - debugger) a všetky štandardné knižnice Javy. Užívateľsky príjemnejšie vývojové prostredia možno zakúpiť napríklad u firiem IBM, WebGain, Borland a iných predajcov, ale základná verzia je zadarmo.

Java sa ako nový jazyk vyvíja veľmi rýchlo, čo je jej podstatnou nevýhodou. I keď základný jazyk je už dobre ustálený, aplikačné programovacie rozhranie Javy (Java API) sa medzi verziami vývojových balíkov JDK 1.0, 1.1, 1.2 (teraz pod názvom Java 2) sa značne zmenilo. Už niekoľko málo mesiacov po svojom vytvorení sú vlastnosti programov písané staršími verziami JDK považované za zastaralé [16].

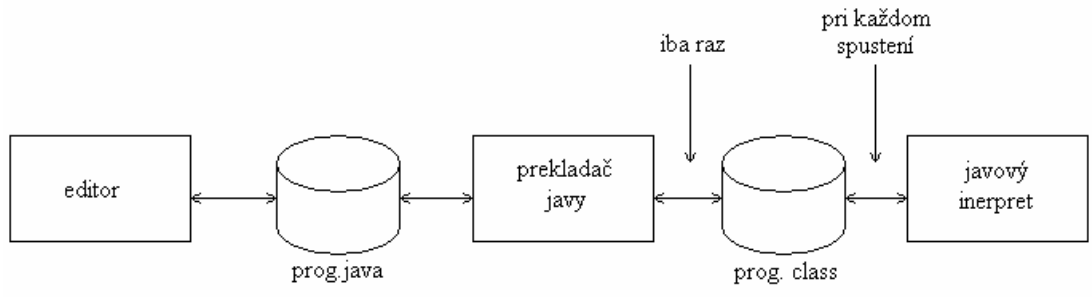
5.1 Základné prvky Javy

Java sa skladá z troch oddelených komponentov:

1. programovací jazyk Java
2. Java – virtuálny stroj (JVM – z angl. Java Virtual Machine)
3. Aplikačné programovacie rozhranie Java (API)

Java sa líši od ostatných programovacích jazykov tím, že všetky programy v Jave sa preloží do jazyka, v ktorom sa potom prevedú na zvláštnom počítači nazývanom javový virtuálny stroj (JVM). Tento JVM sa nazýva bajtový kód. Výstupom všetkých prekladačov Javy je bajtový kód, ktorý môže byť ihneď spustený na javovom virtuálnom stroji. Pretože procesy v skutočných počítačoch, ako napríklad PC alebo unixových pracovných staniciach, nie sú JVM, nemôžu spustiť bajtový kód priamo. Miesto toho má každý typ počítača interpreta (alebo prekladač, ktorý sa aktivuje pri spustený bajtového kódu), ktorý prevádza bajtový kód JVM do strojového jazyka konkrétneho počítača. Deje sa tak automaticky vo chvíli, keď je program Javy spustený. Na obrázku 5.1 môžeme vidieť proces prekladu a spustenie programu v Jave. Program v Jave možno vytvoriť s pomocou ľubovolného textového editora a je možné ho uložiť do súboru so špeciálnou príponou *.java*. Tento program prevádza prekladač Javy do bajtového kódu k spusteniu na

JVM a ukladá tento bajtový kód do súboru so špeciálnou príponou súboru *.class*. Tento preklad sa uskutoční iba raz. Keď sa program spustí, interpret Javy automaticky preloží bajtový kód JVM do inštrukcií toho konkrétneho počítača, na ktorom je program práve spúšťaný. Interpretácia sa uskutoční pri každom spustení programu v Jave. Všimnime si, že bajtový kód Javy je nezávislý na akomkoľvek konkrétnom počítačovom hardware, takže ľubovoľný počítač opatrený interpretom Javy môže spustiť preložený program v Jave nezávisle na type počítača, na ktorom bol program preložený



Obr. 5.1 Proces prekladu a spustenia programu v Jave

Java API je súborom dopredu pripravených softwarových komponent, ktoré sa vyznačujú mnohými užitočnými funkciami. Tieto komponenty poskytujú štandardné spôsoby pre čítanie a písanie súborov, prácou s reťazcami, stavbou grafického užívateľského rozhrania a uskutočňovania mnoho ďalších základných funkcií. Komponenty Java API sa združujú do knižníc (nazývaných balíky) príbuzných komponent. Objekty v týchto štandardizovaných balíkoch ušetrí programátorovi veľa času, lebo v prípade potreby ich môže použiť bez toho aby ich musel znovu písať. Komponenty týchto balíkov sú štandardné vo všetkých implementáciách Javy. To znamená, že program, ktorý ich používa k prevedeniu nejakej funkcie, bude fungovať na ľubovolnom počítačovom systéme, ktorý je vybavený Javou.
Naviac sú pri týchto komponentoch už vychytané chyby, čo uľahčuje prácu nutnú k napísaniu programu i jeho odladeniu všade tam, kde ho budeme používať [16].

5.2 Objektovo orientované programovanie v Jave

V tejto časti sú popísané základné pojmy z objektovo orientovaného programovania. Objektovo orientované programovanie (OOP) spočíva vo vytváraní objektov v software podľa reálnych vzorov. V nasledujúcich kapitolách budú popísané základné vlastnosti objektovo orientovaného programovania.

5.2.1 Objekty

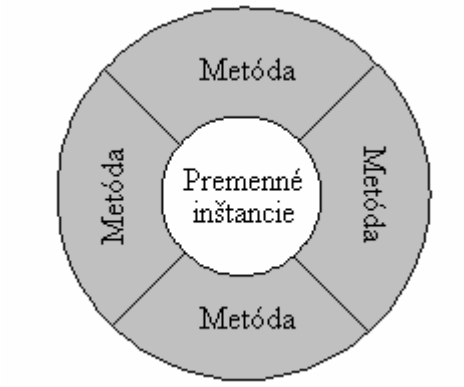
Objektovo orientované programovanie je procesom, počas ktorého sa podľa vlastností a chovania reálnych objektov vytvorí software odrážajúcich tieto vlastnosti a chovanie. Ľubovolný reálny objekt môže byť charakterizovaný svojimi vlastnosťami a svojím chovaním. Ako vzor pre vytvorenie objektu nám môže poslúžiť automobil. Automobil má určité vlastnosti (farbu, rýchlosť, cieľ cesty, spotrebu) a určité správanie (štartovanie, zastavenie, otáčanie kolies atď.).

Vo svete softwaru rozumieme *objektom* takú komponentu softwaru, ktorej štruktúra sa podobá štruktúre objektu v reálnom svete. Každý objekt sa skladá z kombinácie dát (označujeme ich ako *vlastnosti*) a správanie (ktoré nazývame *metódy*). Vlastnosti sú premenné, ktoré určujú základné charakteristiky objektov. Metódy potom popisujú ako sa objekt správa a ako môže byť modifikovaný. Objekt je teda prvok softwaru, ktorý sa skladá z premenných a s nimi spojených metód. Softwarový objekt je často znázornený podobne ako na obrázku 5.2. Objekt si môžeme predstaviť ako bunku s jadrom uprostred, ktorá je tvorená premennými a vonkajším obalom, ktorý reprezentuje metódy. Tieto metódy potom definujú rozhranie medzi premennými objektu a vonkajším svetom. Jadro s dátami je pred vonkajším svetom ukryté za obalom tvoreným metódami. Hovoríme, že premenné objektu sú zapuzdrené v objekte. To znamená, že žiadny kód mimo objektu ho nemôže vidieť ani s nimi priamo manipulovať. Prístup k dátam je možný iba prostredníctvom metód objektu.

Premenné a metódy pri objektov v Jave sa formálne nazývajú *premenné inštancie* a *metódy inštancie*, aby sme ich odlíšili od premenných triedy a metód triedy. Zapuzdrenie sa obvykle používa k zatajeniu podrobnosti o implementácii objektu iným objektom v programe. Ak ostatné objekty programu nemôžu vidieť vnútorný stav objektu, nemôžu ani pri náhodných modifikáciách stavu objektu do nej zaniesť chyby. Naviac zmeny týkajúce sa vnútorných operácii objektu nebudú mať vplyv na operáciu určenú pre ostatné objekty v programe. Pokiaľ zostane rozhranie s vonkajším svetom nezmenené, môžu sa podrobnosti implementácie objektu kedykoľvek zmeniť, bez toho aby to akokoľvek ovplyvnilo iné časti programu. Zapuzdrenie poskytuje tvorcom softwaru dve základné výhody, ktoré označujem ako:

- **Modularita** – Objekt je možné napísať a udržovať nezávisle na zdrojovom kódu ostatných objektov. Takto môže byť objekt veľmi ľahko znovu používaný a predávaný v rámci systému

- **Utajenie informácií** – Objekt má verejné rozhranie, prostredníctvom ktorého s ním môžu ostatné objekty komunikovať. Premenné inštancie objektu však nie sú ostatným objektom prístupné priamo. Preto ak zostane zachované verejné rozhranie, môžu sa premenné a metódy objektu kedykoľvek zmeniť bez toho, aby pritom tieto zmeny akokoľvek ovplyvnili ostatné objekty, ktoré sú na ňom závislé.



Obr. 5.2 Príklad

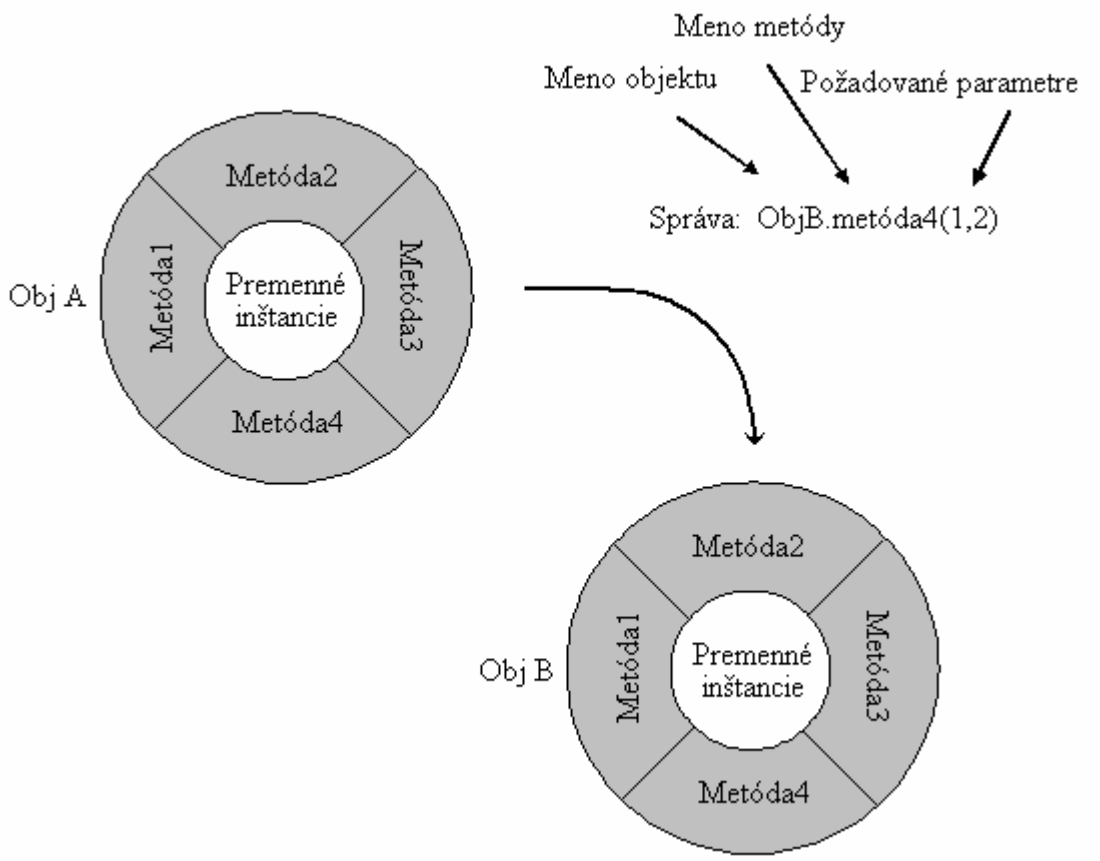
Niekedy objekt svoje premenné inštancie zverejní, aby mohli byť pre ostatné objekty prístupné. Robí sa to občas z dôvodu lepšej výkonnosti. Opakované používanie metódy umožňujúci prístup k často používanej premennej môže totiž v niektorých prípadoch neprimerane spomaliť celý program. Presnejšie povedané ide o porušenie zásad objektovo orientovaného programovania, ale zároveň to je kompromis s ktorým sa stretávame i v reálnom svete. Za normálnych okolností by sa premenná inštancia nikdy nemala zverejniť [16].

5.2.2 Správy

Objekty komunikujú tak, že sa medzi sebou odovzdávajú správy. Ak chce objekt „A“ aby objekt „B“ za neho uskutočnil nejakú akciu, pošle objektu „B“ žiadosť o uskutočnenie jednej z jeho metód (viď obrázok 5.3). Správa prinúti objekt „B“, aby špecifikovanú metódu zrealizoval. Každá správa sa skladá z troch komponent, ktoré poskytujú prijímaciemu objektu všetky potrebné informácie k uskutočneniu žiadanej metódy:

1. Objekt, ktorému je správa adresovaná
2. Meno metódy, ktorá sa na tomto objekte má uskutočniť
3. Parametre požadované metódou.

Správanie objektu je vyjadrené prostredníctvom jeho metód, takže odovzdávané správy podporujú všetky možné interakcie medzi objektmi.



Obr. 5.3 Princíp posielania

Objekt nemusí byť súčasťou rovnakého procesu, dokonca nemusí byť na rovnakom počítači, aby si vzájomne mohli posielať a dostávať správy. Pokiaľ existuje spôsob pre odovzdávanie správ, objekty môžu na seba vzájomne pôsobiť. Z tohto dôvodu je objektovo orientované programovanie veľmi vhodné pre aplikácie typu klient/server, pri ktorých objekt posielajúci správu je na inom počítači ako objekt, ktorý akciu vykonáva [16].

5.2.3 Triedy

Triedy sú softwarové predpisy, podľa ktorých sa vytvárajú objekty. Trieda je softwarový pojem, ktorý určuje počet a typ premenných inštancie, ktoré sú súčasťou objektu a metódy inštancie, ktoré sa na tento objekt budú aplikovať. Každá komponenta triedy sa nazýva *člen*. Medzi dva typy členov patrí *pole*, ktoré obsahuje dátové typy

definované triedou a *metódy*, ktoré predpisujú operácie na poliach. Trieda nám hovorí, ako bude objekt vyzerať a ako sa bude chovať, akonáhle bude vytvorený. Podľa tohto predpisu daného triedou sa vytvorí v pamäti objekt. Z rovnakej triedy je možne naviac inicializovať mnoho rôznych objektov [16].

5.2.4 Premenné a metódy triedy

Ako sa uviedlo vyššie, každý objekt vytvorený z triedy obdrží svoje vlastné kópie všetkých premenných inštancie, ktoré sú pre tieto triedu definované. Premenné inštancie každého objektu sú nezávislé na premenných inštanciách všetkých ostatných objektov. Spolu s premennými, inštancie možno definovať takisto *premenné triedy*. Premenné triedy sa od premenných inštancií odlišujú v tom, že sa vytvorí iba jedna premenná pre všetky objekty z tejto triedy a každý objekt k nej má prístup. Premenné triedy sú v skutočnosti spoločné všetkým objektom vytvoreným z rovnakej triedy. Vzniknú v okamihu, keď sa objekt prvýkrát inicializuje ako inštancia triedy a existujú pokiaľ nie je vykonávaný program dokončený. Môžeme takisto definovať *metódy triedy*. Metódami triedy rozumieme také metódy, ktoré existujú nezávisle na akýchkoľvek objektoch z tejto triedy. Tieto metódy majú prístup k premenným triedy a môžu ich modifikovať, ale nemajú prístup k premenným inštancie a nemôžu ani zavolať žiadne metódy inštancie [16].

5.2.5 Balíky aplikačného programovacieho rozhrania Javy

Skupiny príbuzných tried v Jave sa obvykle zoskupujú do zvláštnych knižníc, nazývaných *balíky*. Aplikačné programovacie rozhranie Javy, skrátene Java API, zahrňuje množstvo balíkov, ktoré implementujú dôležité vlastnosti tohto jazyka [16].

6 Popis softvéru

6.1 Navigačná tabuľka

Menu programu sa skladá z piatich kariet: *File*, *Edit*, *Tools*, *Adjust* a *View*.

File

- Load
- Load from URL...
- Save
- Save as...
- Page Setup
- Print
- Close this picture

Edit

- Undo
- Redo
- Info about image

Tools

- Median (C-WMF)
- Median (MMF)
- Median (CMF)
- Mean filter
- Wiener filter
- Variable coefficient filter
- Filters for speckle noise
- Resize image
- Flip/Mirror/Rotate
- Arithmetic
- Sharpen
- Blur
- Emboss
- Edge detection
- Options

Adjust

- Adjust Red/Green/Blue
- Adjust Hue/Saturation
- Contrast/Brightness
- Bright image by 10%
- Dark image by 10%
- Balance color
- Rescale
- Logarithmic/Exponential operators
- Switch channels
- Negative
- Gray scale
- Black & white

View

- Original size
- Increase zoom
- Decrease zoom
- Full screen
- Toolbars
  - Operational bar
  - Zoom bar
  - Image preview
- Histogram
- About

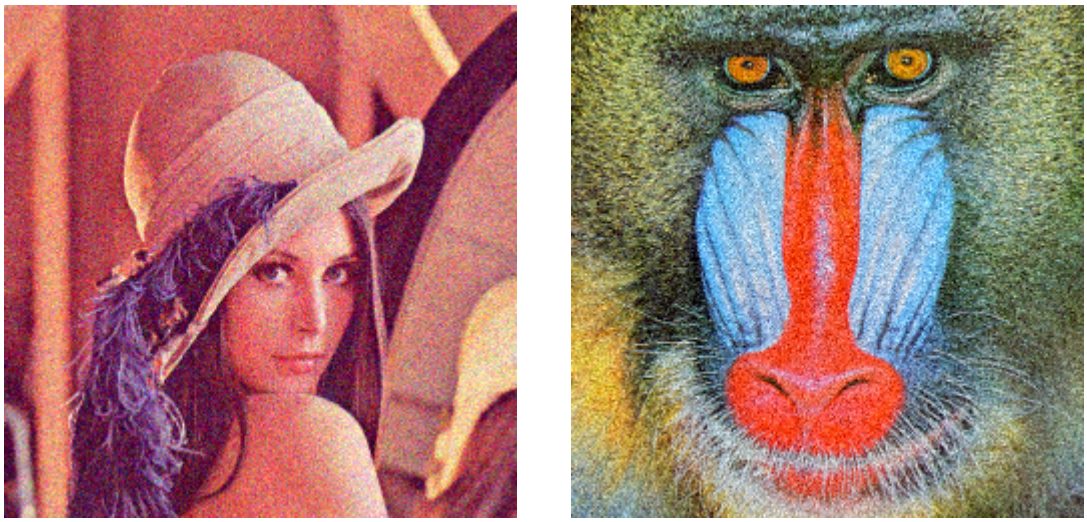
6.2 Výstupy z filtrov

Na aditívny Gaussov šum, impulzový šum a na filtre s voliteľnými koeficientmi masky sa použili dva obrazy. Prvým z nich je obraz *Lena* a druhý *Mandrill*. Obidva majú rozlíšenie 256x256 bodov s 24-bitvou hĺbkou a sú zobrazené na obr. 6.1.

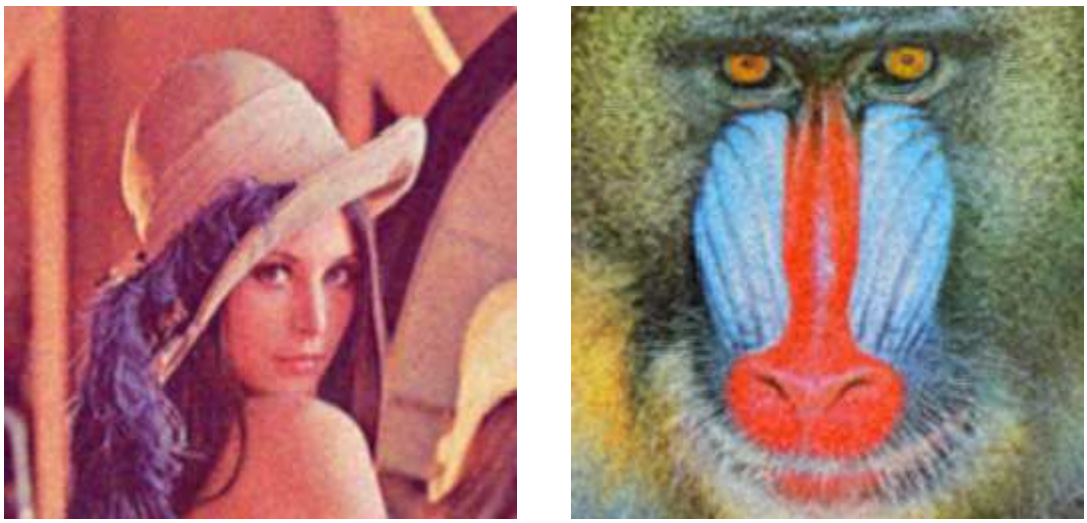
Obr. 6.1 Obrazy Lena a Mandrill (originály)

6.2.1 Filtre potláčajúce aditívny Gaussov šum

Na filtráciu aditívneho Gaussovho šumu boli použité dva filtre. Adaptívny Wienerov filter a spriemerovací filter so štvorcovou maskou o veľkosti 3x3. Obrazy boli znehodnotené Gauussovým šumom so strednou hodnotou $\mu = 0$ a smerodajnou odchýlkou $\sigma = 20$. Zašumené obrazy sú znázornené na obr. 6.2 a vyfiltrované na obr. 6.3a-b. Lepšie výsledky vykazoval adaptívny Wienerov filter. Na obraze Mandrill vidno, že je filtrácia obrazu lepšia, ale to je spôsobené tým, že obraz obsahuje viac detailov ako Lena a ľudské oko nedokáže rozpoznať takúto zmenu.



Obr. 6.2 Obrazy Lena a Mandrill znehodnotené aditívnym Gaussovým šumom s $\mu = 0$ a $\sigma = 20$



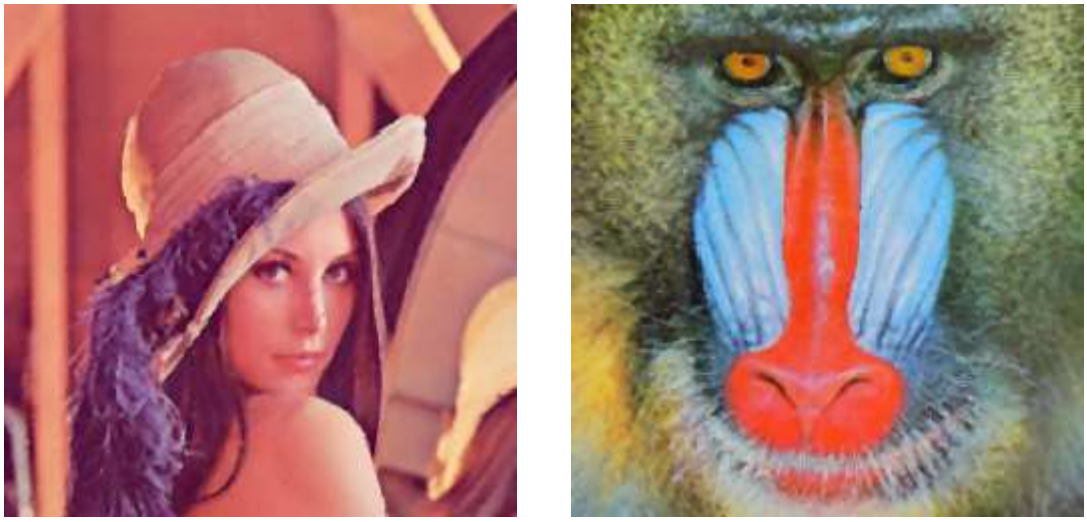
Obr. 6.3a Filtrácia zašumených obrazov pomocou spriemerovacieho filtra s veľkosťou okna 3x3

Obr. 6.3b Filtrácia zašumených obrazov pomocou Wienerovho adpatívneho filtra s veľkosťou okna 3x3

6.2.2 Filtre potláčajúce impulzový šum

Na filtráciu impulzového šumu sa použili tri druhy filtrov. Mediánový zložkový filter, vektorový mediánový filter s marginálnym zoradením a vektorový mediánový filter s podmieneným zoraďovaním v HSV priestore. Obrazy boli znehodnotené korelovaným impulzovým šumom s premenlivou hodnotou a pravdepodobnosťou p = 10%. Zašumené obrazy sú znázornené na obr. 6.4 a vyfiltrované na obr. 6.5a-c. Najlepšie výsledky vykazoval vektorový mediánový filter s marginálnym rozdeleným. Zo všetkých mediánových filtrov bol však najpomalší.

Obr. 6.4 Obrazy Lena a Mandrill znehodnotené korelovaným impulzovým šumom s premenlivou hodnotou a pravdepodobnosťou p = 10%



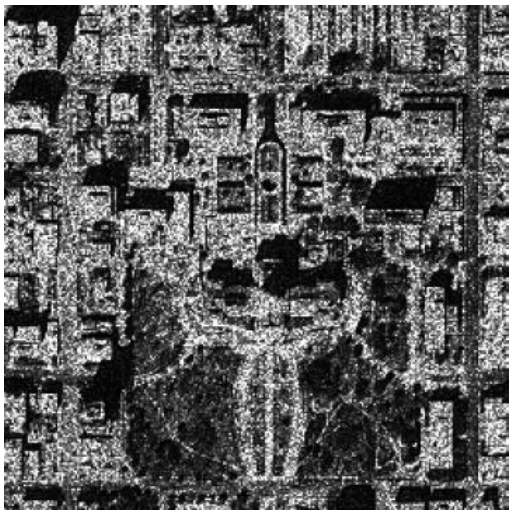
Obr. 6.5a Filtrácia zašumených obrazov pomocou zložkového mediánového filtra s veľkosťou okna 3x3

Obr. 6.5b Filtrácia zašumených obrazov pomocou vektorového mediánového filtra s marginálnym zoradením s veľkosťou okna 3x3

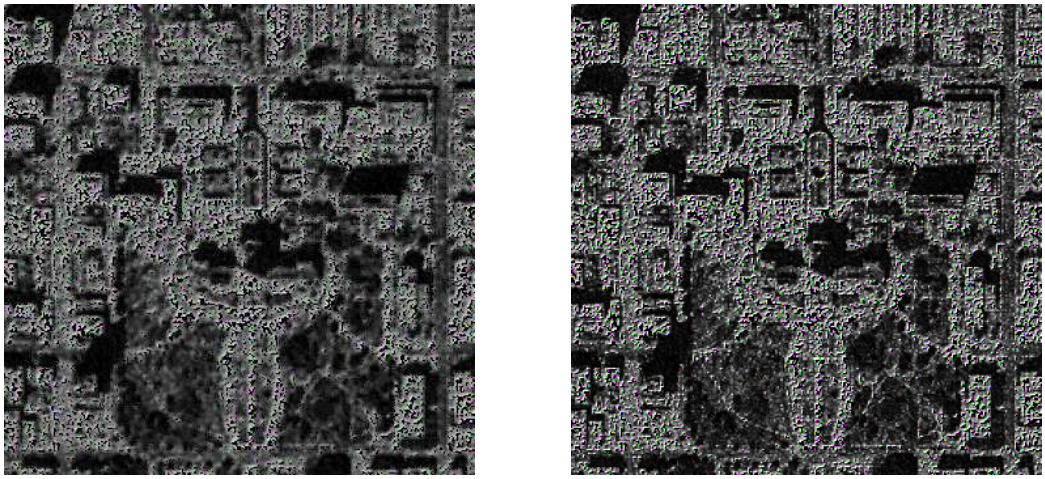
Obr. 6.5c Filtrácia zašumených obrazov pomocou vektorového mediánového filtra s podmieneným zoraďovaním v HSV priestore s veľkosťou okna 3x3

6.2.3 Filtre potláčajúce multiplikatívny šum

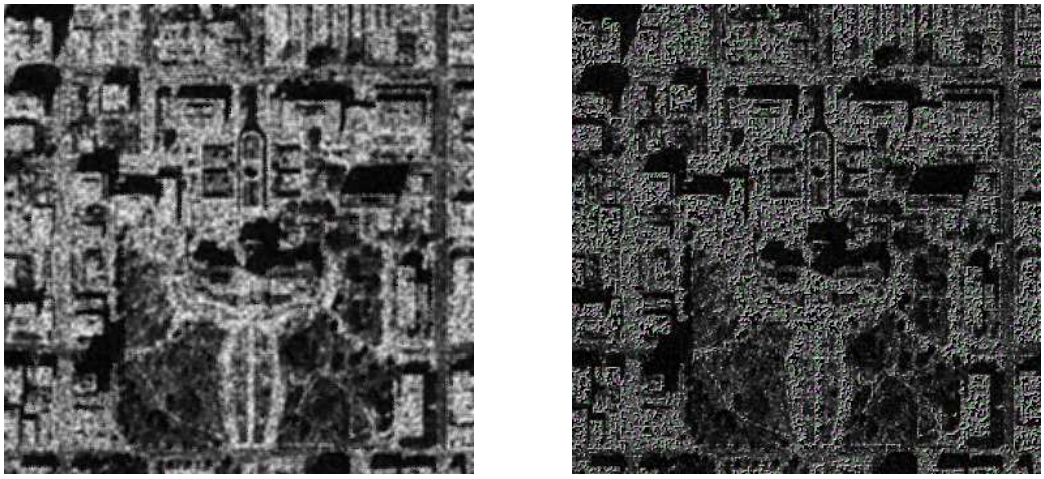
Na multiplikatívny šum bol použitý obraz, ktorý bol vytvorený pri snímaní z veľkej výšky a prejavil sa tu v značnej miere bodkový šum. Obraz má rozmery 360x360 bodov. Na filtráciu bolo použitých šesť filtrov. Frostov filter, Rozšírený Frostov filter, adaptívny Frostov filter, Kuanov filter, Leeov filter a Lopesov filter, všetky s veľkosťou okna 3x3. Zašumený obraz je na obr. 6.6 a vyfiltrované na obr. 6.7a-c. Najlepšie výsledky vykazoval adpatívny Wienerov filter.



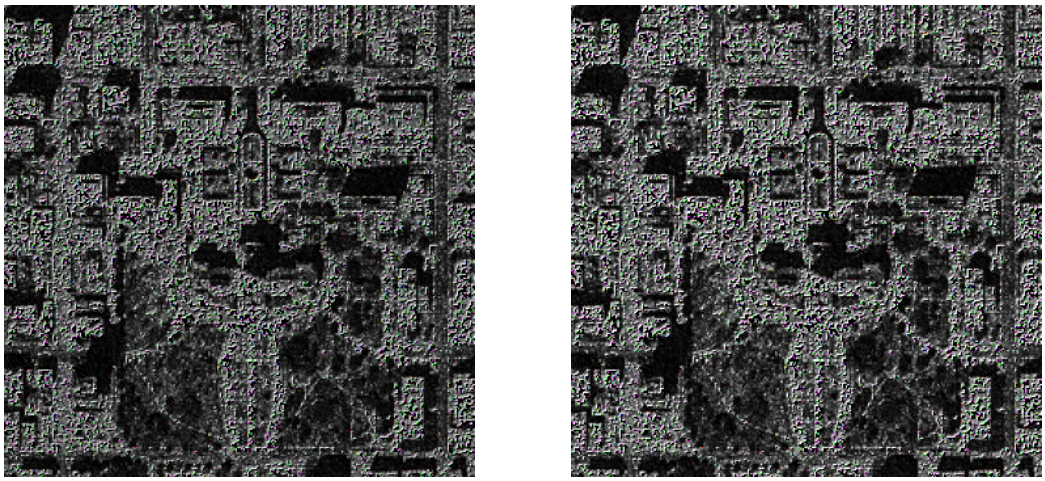
Obr. 6.6 Obraz znehodnotený bodkovým šumom



Obr. 6.7a Obraz znehodnotený bodkovým šumom filtrovaný Frostovym filtrom (vľavo) a rozšíreným Frostovym filtrom (vpravo)



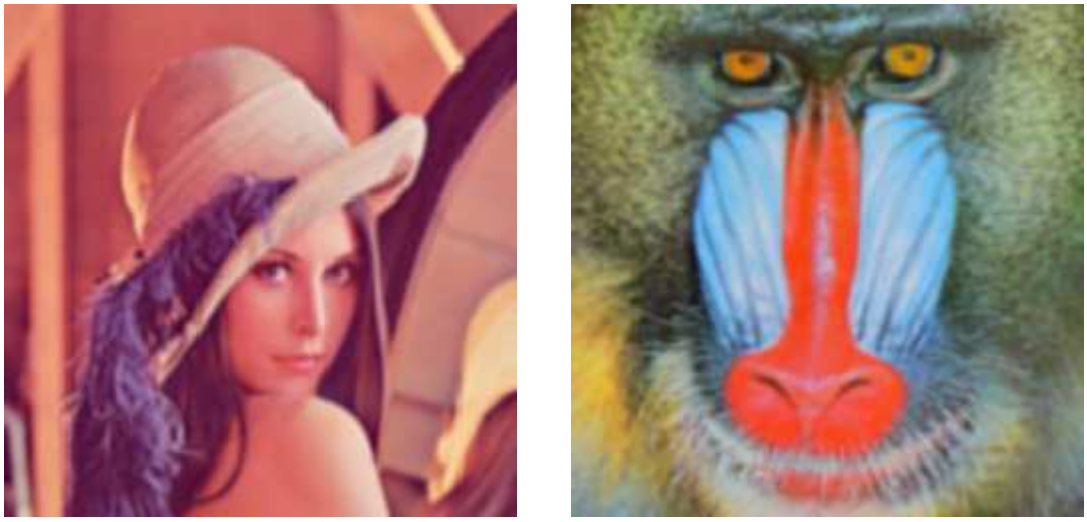
Obr. 6.7b Obraz znehodnotený bodkovým šumom filtrovaný adaptívnym Frostovym filtrom (vľavo) a Kuanovym filtrom (vpravo)



Obr. 6.7c Obraz znehodnotený bodkovým šumom filtrovaný Leeovym filtrom (vľavo) a Lopesovym filtrom (vpravo)

6.2.4 Filtre s voliteľnými koeficientmi masky

Na obrázkoch 6.8a-d sú zobrazené obrazy Lena a Mandrill po aplikovaný filtrov s voliteľnými koeficientmi. Ostriaci filter je znázornený na obr. 6.8a, rozmazávací na obr. 6.8b, filter na detekciu hrán vo všetkých smeroch na obr. 6.8c a filter, ktorý vytvára 3D efekt na obr. 6.8d.



Obr. 6.8a Aplikácia rozmazávacieho filtra na obrazy Lena a Mandrill

Obr. 6.8b Aplikácia filtra na detekciu hrán na obrazy Lena a Mandrill

Obr. 6.8c Aplikácia ostriaceho filtra na obrazy Lena a Mandrill

Obr. 6.8d Aplikácia filtra vytvárajúceho 3D efekt na obrazy Lena a Mandrill

7 Záver

Úlohou tejto diplomovej práce bolo vyvinúť GUI aplikáciu, ktorá dokáže spracovať statické obrazy a aplikovať na ne vybrané číslicové filtre. Spracovanie obrazu je v súčasnej dobe veľmi rozšírené, ale chýba určitý nástroj, ktorý by dokázal spracovať obrazy on-line, bez zbytočných inštalácií. A práve takýto systém som mal navrhnúť v mojej diplomovej práci.

Na túto prácu som sa rozhodol použiť Java prostredie použitím jedného jeho programu – apletu. Java sa totiž ukázala byť veľmi dobrým prostriedkom na dosiahnutie tejto úlohy. Je veľmi rozšírená a zároveň poskytuje multiplatformovú podporu, čo je v dnešnej dobe veľmi dôležitým faktorom. Existujú samozrejme aj iné programovacie jazyky, ktoré by sa mohli použiť na tento projekt, ale majú isté nevýhody oproti Jave.

Pri spracovaní práce som sa stretol s množstvom problémov, ktoré sa mi však podarilo vyriešiť a myslím si, že sa mi podarilo vypracovať originálny program, ktorý rieši komplexnejšie otázku číslicového spracovania obrazu použitím web rozhrania. Najväčšie starosti mi spôsobila zabezpečovacia časť programu, keďže má Java veľmi komplikovaný systém zabezpečenia. Ale aj tento problém sa mi aj za pomoci môjho spolužiaka podarilo vyriešiť, a tým som uspel vo všetkých bodoch diplomovej práce.

Do programu by sa dali vložiť ešte mnohé ďalšie nápady ako je napríklad pridanie kresliacich nástrojov, vylepšenie dizajnu, alebo podpora pracovať s viacerými obrázkami naraz, ale pre nedostatok času sa mi ich nepodarilo zrealizovať.

Na záver by som chcel ešte dodať, že diplomová práca priniesla mnohé pozitíva. Jedným z nich bolo to, že som sa naučil programovať vo veľmi rozšírenom programovacom jazyku, čo mi môže priniesť prospech v budúcnosti. S programovaním súvisí aj to, že som sa zdokonalil v logickom myslení, čo je veľmi dôležitým faktorom a v neposlednom rade ma to donútilo samostatne rozmýšľať, pracovať s literatúrou a vyskúšať si niečo nové.

8 Použitá literatúra

[1] Abrishami Moghaddam, H., Valadan Zouj, M. J., Dehghani, M.*: Bayesian-based despeckling in wavelet domain using "a trous"algorithm*, XXth ISPRS Congress, pp. 27-31, Commission 7, 12-23 July 2004, Istanbul, Turkey.

[2] Blázsovits, G.: Interaktívna učebnica spracovania obrazu, 2006, Bratislava, ISBN 80-89186-08-4. Dostupné na internete: http://dip.sccg.sk/predspra/predspra.htm

[3] Digital signal processing. Dostupné na internete: http://en.wikipedia.org/wiki/Digital_signal_processing

[4] Fisher, R., Perkins, S., Walker, A., Wolfart, E.: The Hypermedia Image Processing Reference, 2004 University of Edinburgh. Dostupné na internete: http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm

[5] Pavelek, M., Janotková, E., Štětina, J.: Vizualizační a optické měřicí metody, december 2001 Brno. Dostupné na internete: http://ottp.fme.vutbr.cz/skripta/optika/1506.htm

[6] McHugh, S.: Digital photography tutorials, 2005 Cambridge. Dostupné na interntete: http://www.cambridgeincolour.com/tutorials/bit-depth.htm

[7] Fisher, R., Perkins, S., Walker, A., Wolfart, E.: The Hypermedia Image Processing Reference, 1994 University of Edinburgh. Dostupné na internete: http://www.cee.hw.ac.uk/hipr/html/mean.html

[8] Dobrovolný, P.: Digitální zpracování materiálů DPZ, Masarykova Univerzita Brno. Dostupné na internete: http://www.geogr.muni.cz/archiv/vyuka/DPZ_CVICENI/Texty/DZO_05_ zvyrazneni_2.pdf2.pdf

[9] F. Jørgensen, J.F.: Scanning Probe Image Processor, CEO Image Metrology A/S. Dostupné na internte: http://www.imagemet.com/WebHelp/spip.htm#hid_filters_smoothing_mean.htm

[10] Hudec, R.: Viacrozmerná číslicová filtrácia šedých obrazov znehodnotených zmiešaným šumom pomocou adaptívnych LMS Wienerových filtrov, L a IIR filtrov, Písomná práca k dizertačnej skúške, KEMT TU v Košiciach, 2000.

[11] Lim, S. J.: Two-Dimensional Signal and Image Processing. Prentice Hall, New Jersey, 1990, pp.536-540,694 s., ISBN 01-3935-322-4.

[12] Mansourpour, M., Rajabi, M. A. , Blais, J.A.R.: Effects and performance of speckle noise reduction filters on active radar and SAR images, ISPRS Workshop, 5th Session, 14-16 February 2006, Ankara, Turkey, ISPRS Volume Number: XXXVI-1/W41.

[13] Mních, M.: Analýza vplyvu zoradenia viacrozmerných dát pri filtrácii zašumených farebných obrazov, Diplomová práca, Katedra telekomunikácií ŽU v Žiline, 2004.

[14] Vaseghi, V. S.: Advanced digital signal processing and noise reduction, Third edition, 480 s., pp. 23-28, 2004, ISBN 04-7009-494-X.

[15] Wang, Z., Zhang, J., Wang, T.: The contrast research of the methods of restraining the speckle noise of SAR images, XXth ISPRS Congress, pp. 129-133, Commission 2, 12-23 July 2004, Istanbul, Turkey.

[16] Chapman, Stephen J. Začínáme programovat v jazyce Java, 2. vyd. Computer Press Brno, 2003, 307 s., ISBN 80-7226-472-9.

9. Zoznam príloh

1. CD médium
   - Diplomová práca v elektronickej podobe
   - Zdrojový kód Java aplikácie v elektronickej podobe
2. Používateľská príručka

Čestné vyhlásenie

Vyhlasujem, že som zadanú diplomovú prácu vypracoval samostatne, pod odborným vedením vedúceho diplomovej práce Ing. Róberta Hudeca PhD. a používal som len literatúru uvedenú v práci.

Súhlasím so zapožičiavaním diplomovej práce.

V Žiline dňa ....

podpis diplomanta

Poďakovanie

Na tomto mieste by som sa chcel poďakovať svojmu vedúcemu diplomovej práce Ing. Róbertovi Hudecovi PhD. za jeho cenné rady, ktoré som potreboval pri spracovaní tejto práce. Ďalej by som chcel poďakovať svojim dvom spolužiakom Branislavovi Staroňovi, ktorý mi pomáhal so zabezpečovacou časťou programu a Lukášovi Fudálymu, ktorý mi pomohol s návrhom web stránky. V neposlednom rade by som chcel poďakovať Ing. Penke Martincovej PhD. za jej postrehy, ktoré mi pomohli pri návrhu programu.

ŽILINSKÁ UNIVERZITA V ŽILINE
Elektrotechnická fakulta
Katedra telekomunikácií

„Číslicové spracovanie a filtrácia statických obrazov“ ako WEB služba z oblasti aplikovaného ČSS

Prílohová časť

Jozef Pohorelec

2007

Obsah

Zoznam obrázkov

1 Informácie o programe

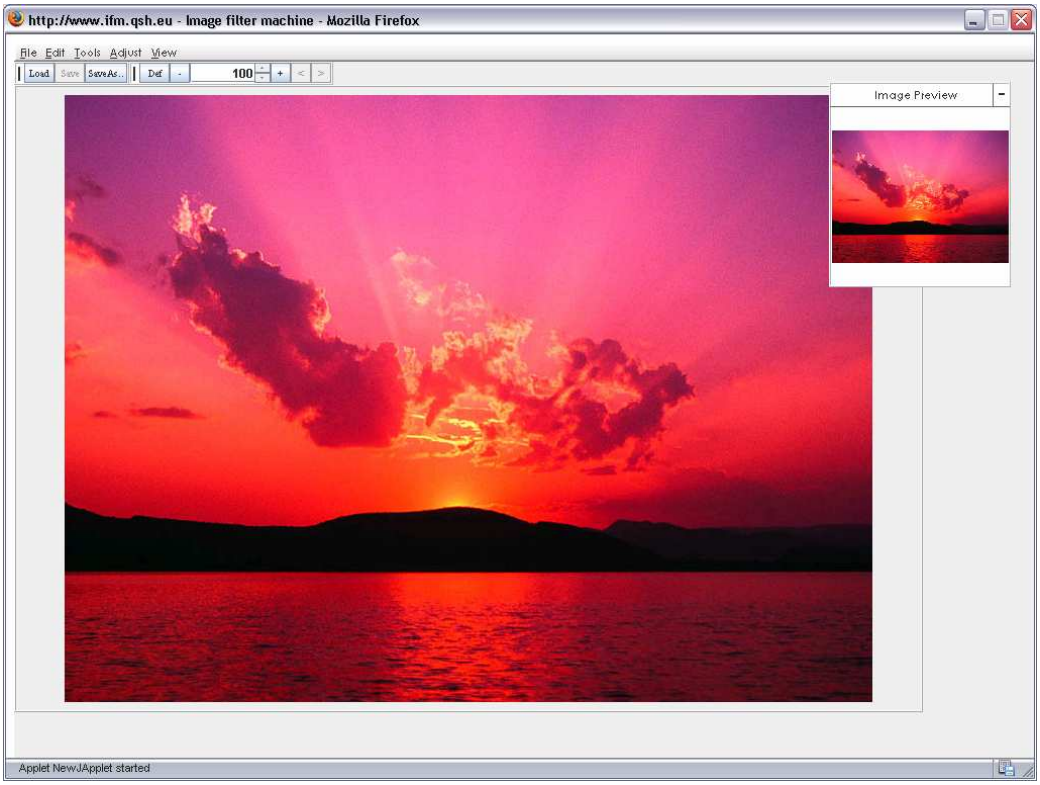
Názov programu: Image filter machine v.1.0
Autor: (c) 2007, Jozef Pohorelec, pohorelecjozef@gmail.com
Typ licencie: Open-source

Doporučené systémové prostriedky:

- Operačný systém – Multiplatformová podpora
- Operačná pamäť aspoň 128MB, odporúčaná 512MB
- Pripojenie na internet
- Nainštalované JAVA prostredie (asi 270MB)

2 Popis prostredia

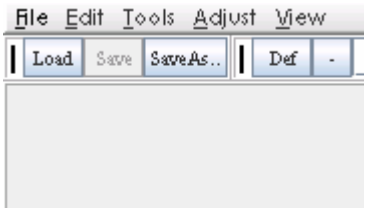
Základné okno programu



Obr. 1 Základné okno programu po načítaní obrazu.

2.1 Hlavné menu

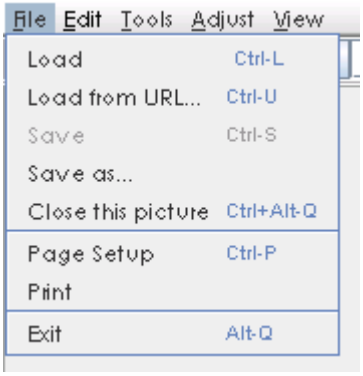
Hlavné menu sa skladá z piatich častí:



File – Súbor
Edit – Editácia
Tools – Nástroje
Adjust - Prispôsobenie
View – Zobrazenie

Obr. 2 Ukážka hlavného menu.

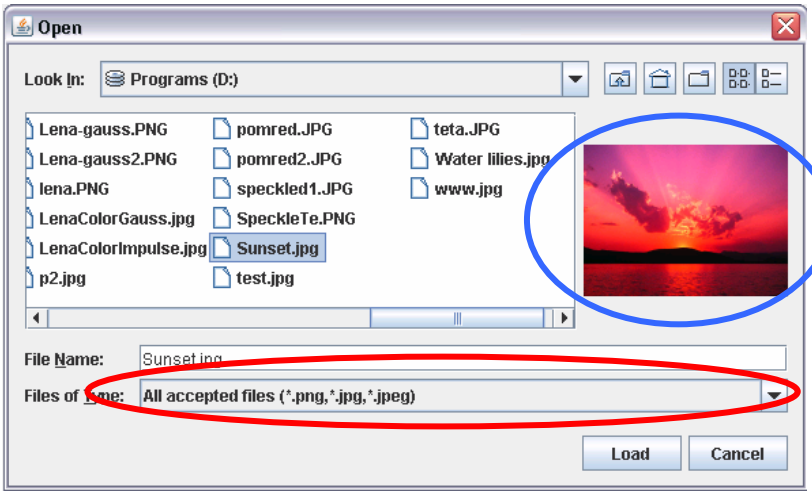
2.1.1 File – Súbor



Load – načítanie obrázku z lokálneho disku užívateľa
Load from URL – načítanie obrázku zo špecifikovaného URL
Save – uloženie obrázku na disk (prepísanie pôvodného)
Save as – uloženie obrázku na disk (vytvorenie nového)
Close this picture – zatvorenie obrázku
Page setup – nastavenie strany pred tlačou
Print – vytlačenie obrázku
Exit – opustenie systému

Obr. 3 Karta File.

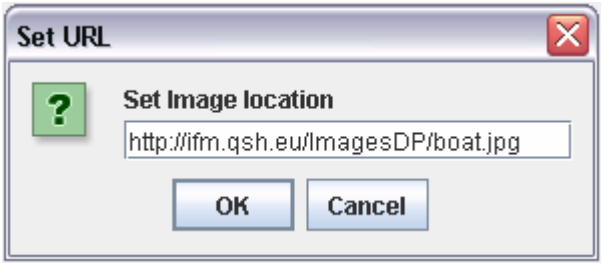
2.1.1.1 Load



Obr. 4 Komponenta na otvorenie obrázku z lokálneho disku.

Vpravo je malá ukážka obrázku (modrý kruh). Na spodku je možné si typ súboru (Files of Type) – „*.jpg“ a „*.png“. (červený ovál)

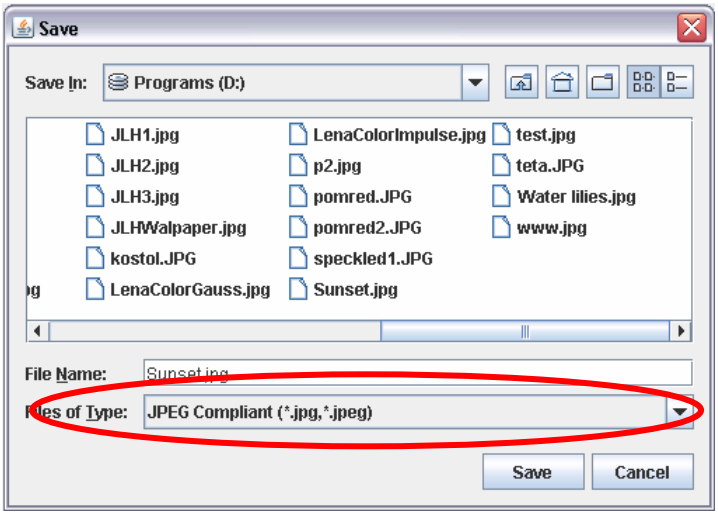
2.1.1.2 Load from URL



Obr. 5 Komponenta na otvorenie obrázku z URL.

Do Textového pola je môžné zadať cestu k obrázku, ktorý sa nachádza na webe.

2.1.1.3 Save, Save as

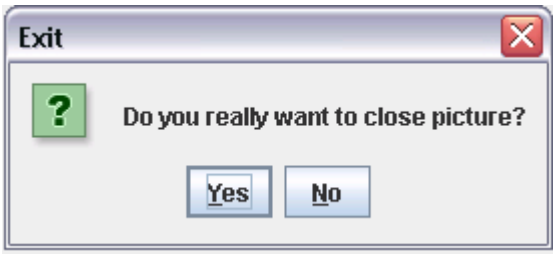


Obr. 6 Komponenta na uloženie obrázku.

Dole je možné si vybrať v akom formáte chceme obrázok uložiť. K dispozícii sú „*.jpg“, „*. bmp“ a „*. png“.

POZOR!!!
Existujúci súbor sa prepíše bez vyzvania automaticky.

2.1.1.4 Close picture



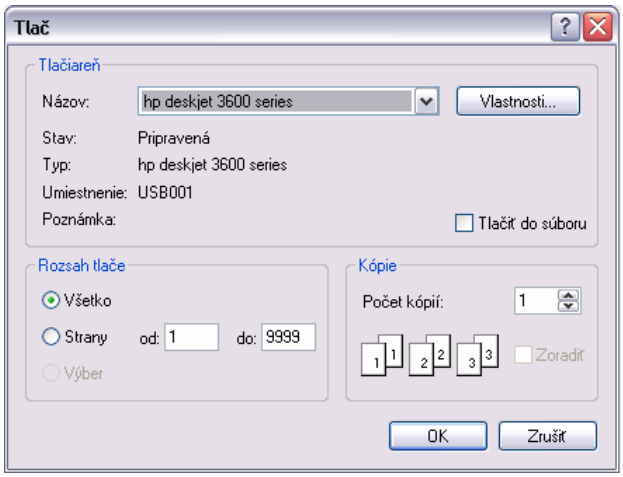
Obr. 7 Okno pre zatvorenie obrázku.

Po kliknutí na „Close this picture“ sa otvorí toto okno s výzvou, či skutočne chceme obrázok zatvoriť.

2.1.1.5 Page setup, Print

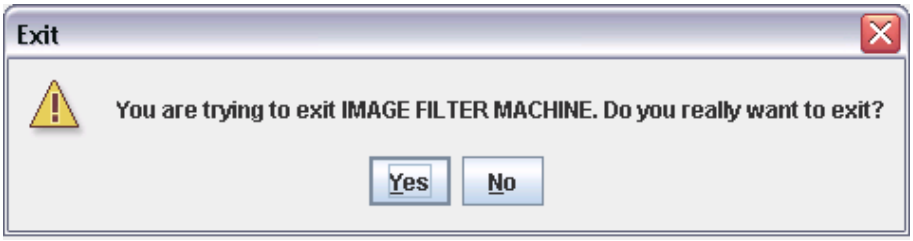
Po kliknutí na „Page setup“ sa otvorí okno s nastavím strany. Vzhľad tohto okna sa mení v závsilsosti na používanej platforme, a preto je uvedený iba príklad pre Windows XP.

Tlačidlo „Print“ vytlačí obrázok automaticky v predvolenej kvalite, ktorá závisí od danej tlačiarni.



Obr. 8 Príklad okna pre nastavenie strany pred tlačou.

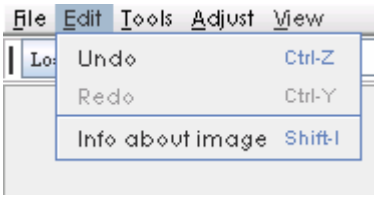
2.1.1.6 Exit



Obr. 9 Potvrdzovacie okno pre ukončenie programu.

Po kliknutí na „Exit“ sa zobrazí toto potvrdzovacie okno pre ukončenie programu. Pre správnu činnosť programu je odporúčané ukončovať program takýmto spôsobom. Nie cez“ krížik“. Applet sa totiž vypne až po zatvorený všetkých okien prehliadača. POZOR!!! Pri ukončení programu sa vypne prehliadač aj z jeho zvyšnými oknami.

2.1.2 Edit – Editácia



Undo – späť k predchádzajúcej zmene
Redo – opakovanie zmeny
Info about image – informácia o obraze

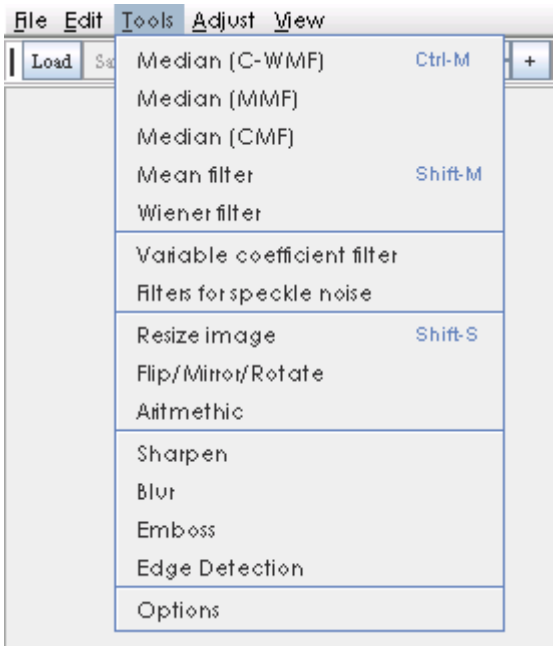
Obr. 10 Karta Edit.

2.1.2.1 Info about image

Obr. 11 Komponenta pre zistenie informácii o obraze.

Sú tu informácie o obraze šírka (Width), výška (Height), rozlíšenie (DPI) a meno súboru (Name).

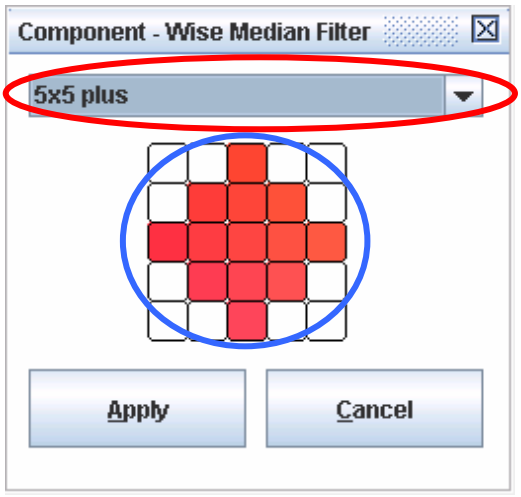
2.1.3 Tools – Nástroje



Obr. 12 Karta Tools.

Median (C-WMF) – zložkový medián
Median (MMF) – medián založený na marginálovom zoradení
Median (CMF) – medián založený na podmienenom zoradení v HSV priestore
Mean filter – spriemerovací filter
Wiener filter – Wienerov adaptívny filter
Variable coefficent filter – filter s voliteľnými koeficientmi
Filters for speckle noise – Filtre na odstránenie bodkového šumu
Resize image – zmena veľkosti obrázku
Flip/Mirror/Rotate – preklopenie, zrkadlenie a otočenie obrázku
Aritmethic – aritmetika medzi dvomi obrázkami
Sharpen, Blur, Emboss, Edge Detection – ostrenie, rozmazanie, 3D efekt a detekcia hrán v obrázku.
Options – užívateľské nastavenia súboru a vzhľadu programu

2.1.3.1 Median, Wiener, Mean filter

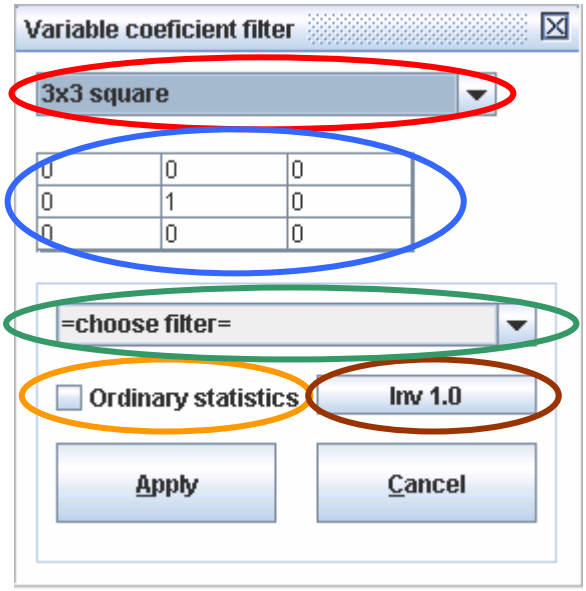


Obr. 13 Komponenta pre DP filtráciu obrazu.

Toto okno je zhodné pre všetky mediánové filtre, spriemerovací filter a Wienerov filter. Menia sa len vo farebnom označený masky. Medián (C-WMF) má červenú masku, medián (MMF) zelenú masku, medián (CMF) oranžovú masku, spriemerovací filter má modro-zelenú masku a Wienerov filter má tmavo-modrú masku. Hore sa nastavuje veľkosť a tvar okna – 3x3(s,x,+), 5x5(s,x,+), 7x7(s) a 9x9(s) (červený ovál) a v strede sa zobrazuje prislúchajúca ukážka masky (modrý kruh).

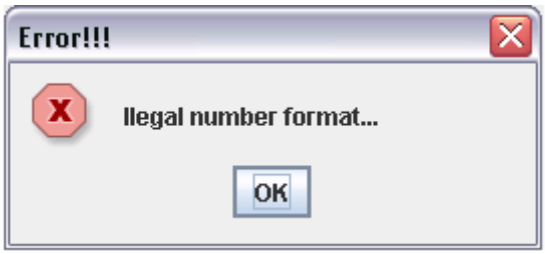
s – štvorcový tvar okna
x – krížový tvar okna
+ - plusový tvar okna

2.1.3.2 Variable coefficent filter



Obr. 14 Komponenta pre filter s variabilnými koeficientmi masky.

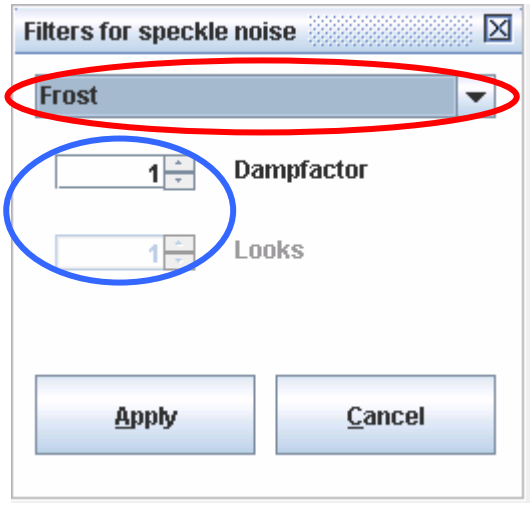
Hore sa nachádza výberové pole kde sa vyberá veľkosť masky (3x3,5x5,7x7 a 9x9 – červený ovál). Pod ním sa nachádza tabuľka (modrý ovál), kde môžeme vpisovať reálne čísla. V spodnom panely sa nachádza výberové pole (zelený ovál), kde sú nadefinované filtre – blur, blur more, motion blur (rozmazanie obrázku), find edges horizontal, find edges vertical, find edges 45°, find edges all directions, (detekcia hrán) sharpen, sharpen subtle, sharpen exessively (ostrenie obrázku), emboss, emboss exaggerated (vytvorenie 3D efektu). Ďalej je tu zaškrtávacie tlačidlo „Ordinary statistics“ (oranžový ovál), ktoré slúži na zoradenie vstupných dát od najmenšieho po najväčší. Tlačidlo „Inv“ (hnedý ovál) slúži k zisteniu či zadané koeficienty spĺňajú podmienku invariability, teda či súčet koeficientov masky je rovný „1“.



Obr. 15 Chybové hlásenie.

Toto okno sa zobrazí ak sa zadá do tabuľky nesprávny číselný formát (napr. „d1“, „1,5“ atď.) a stlačí sa tlačidlo „Inv“ alebo „Apply“

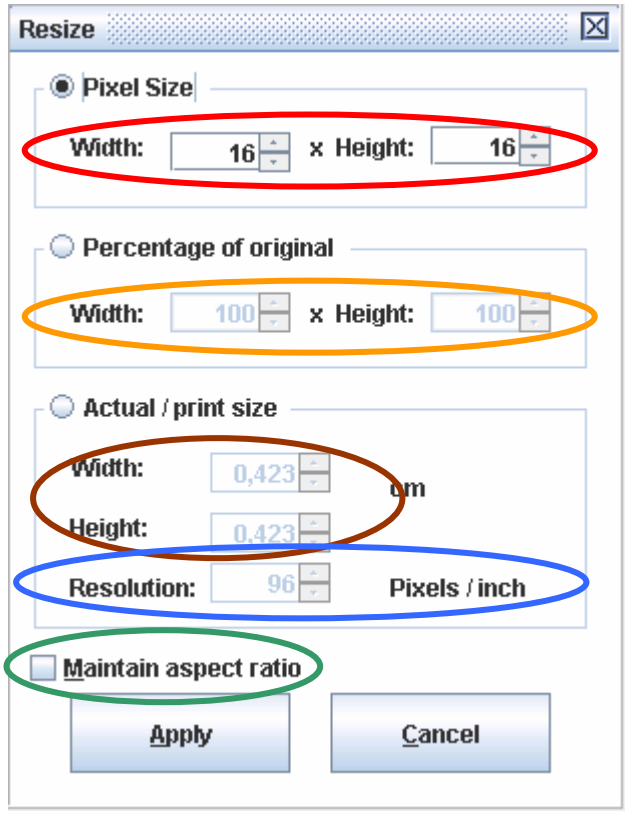
2.1.3.3 Filters for speckle noise



Obr. 16 Komponenta pre filtre určené na bodkový šum.

Hore je možné vybrať šesť druhov filtrov určených pre tento typ filtrácie – Frostov, rozšírený Frostov, adpatívny Frostov, Kuanov, Leeho, Lopesov (červený ovál). Ďalej je možné vybrať faktor vlhkosti (Dampfactor) a počet lookov (Looks). Je možné zadať celé čisla od -100 do 100 (modrý ovál).

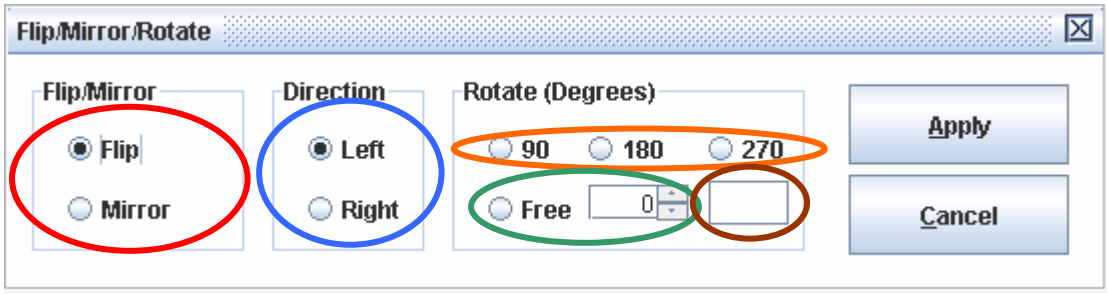
2.1.3.4 Resize



Obr. 17 Komponenta na zmenu veľkosti a rozlíšenia obrazu

K dispozícii sú tri možnosti ako zmeniť veľkosť obrázku. Prvá je zmena pixelov (Pixel size – červený ovál), percentuálna zmena veľkosti (Percentage of original – oranžový ovál) a zmena aktuálnej veľkosti (Actual/print size – hnedý ovál). Na zmenu veľkosti rozlíšenia slúži políčko „Resolution“ (modrý ovál).
Na spodku komponenty sa nachádza zaškrtávacie políčko, ktoré slúži na zachovanie pomeru strán (Maintain aspect ratio – zelený ovál).

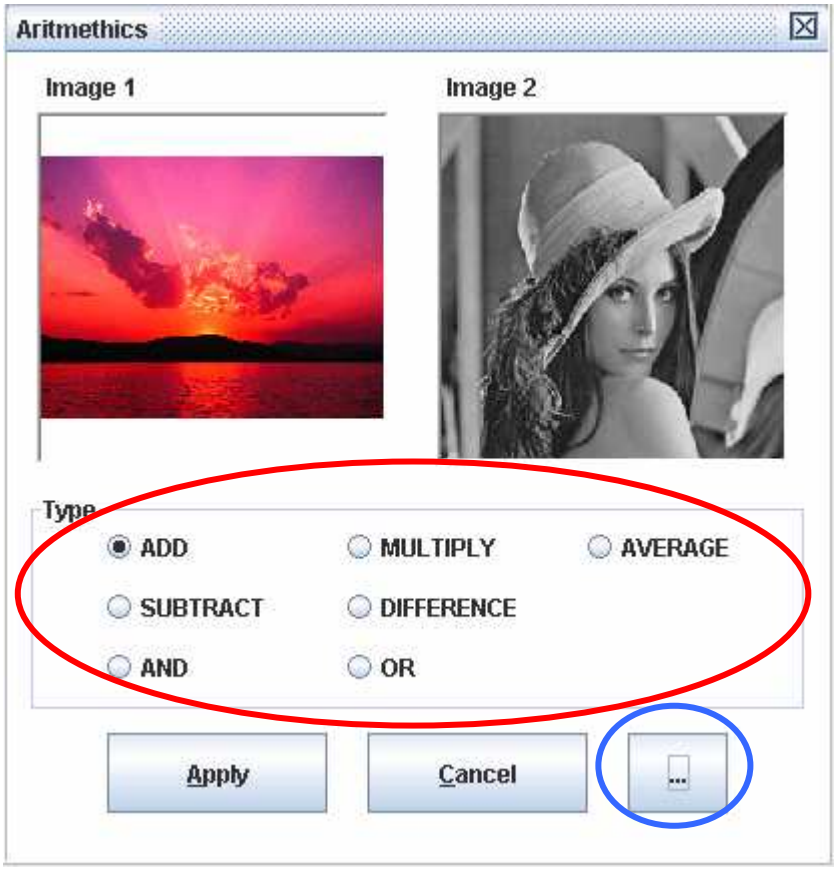
2.1.3.5 Flip/Mirror/Rotate



Obr. 18 Komponenta na prevrátenie obrázku, zrkadlový obraz a otočenie obrázku.

V prvom panely sa nastavuje prevrátenie (Flip) a zrkadlenie (Mirror) – červený ovál. V druhom panely sa určuje smer otočenia vľavo(Left) alebo vpravo (Right) – modrý ovál. V treťom sa nastavuje uhol otočenia 90°, 180°, 270° – oranžový ovál, alebo voľný uhol (Free) – zelený ovál. Nachádza sa tu ešte tlačidlo na zmenu pozadia pri otočení obrázku – hnedý ovál. Veľkosť otočenia obrázku cez „Free“ môže byť v rozsahu 0 - 360°.

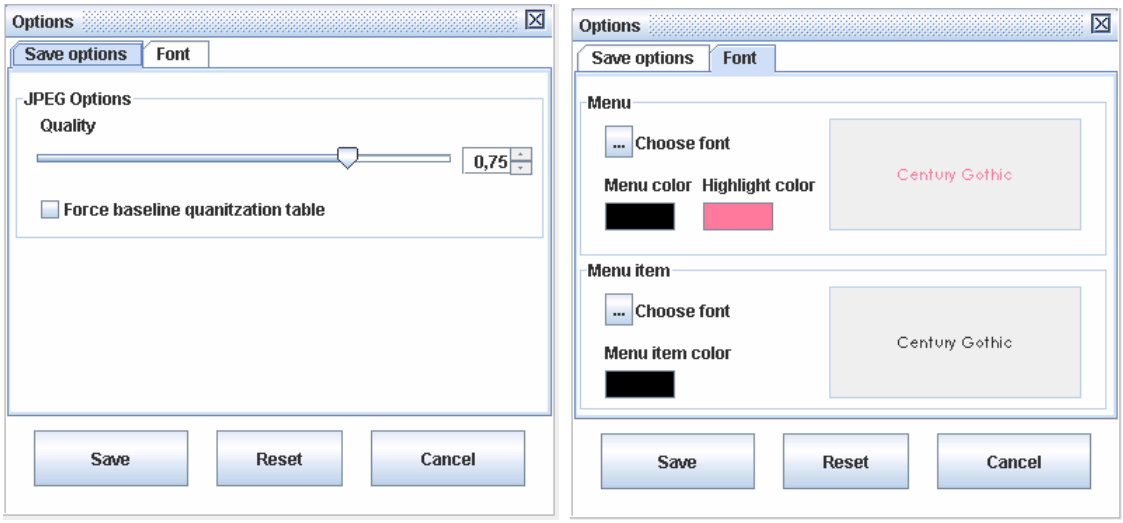
2.1.3.6 Aritmethics



Obr. 19 Komponenta určená na aritmetiku medzi dvomi obrázkami.

Prvý obrázok je znázornený vľavo(Image 1). Druhý obrázok je nutné načítať z disku cez tlačidlo ktoré sa nachádza vpravo dolu s označením „...“ (modrý ovál) a zobrazí sa vpravo hore (Image 2). V paneli type je možné vybrať akú funkciu chceme medzi dvoma obrázkami vykonať (červený ovál) – pričítať (Add), násobiť (Multiply), priemer (Average), odčítať (Subtract), rozdiel (Difference), alebo použitie logických operátorov AND a OR. Ak sa zabudne vložiť druhý obrázok a stlačí sa „Apply“ zobrazí sa chybové hlásenie o tom, že druhý obrázok chýba.

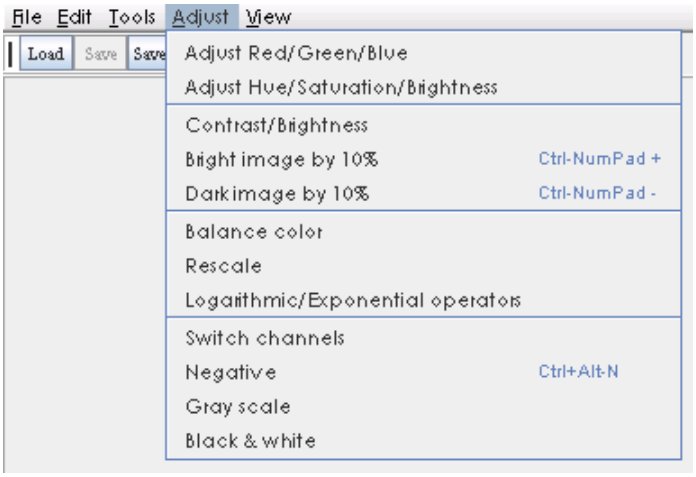
2.1.3.7 Options



Obr. 20 Komponenta Options.

Na ľavom obrázku je zobrazená prvá karta z komponenty Možnosti. Nastavuje sa tu kvalita JPEG obrázkov. Na pravom obrázku je zobrazená druhá karta. Užívateľ si tu môže nastaviť farbu, veľkosť a typ písma v menu. Tlačidlo „Reset“ slúži k pôvodnému nastaveniu.

2.1.4 Adjust – Prispôsobenie

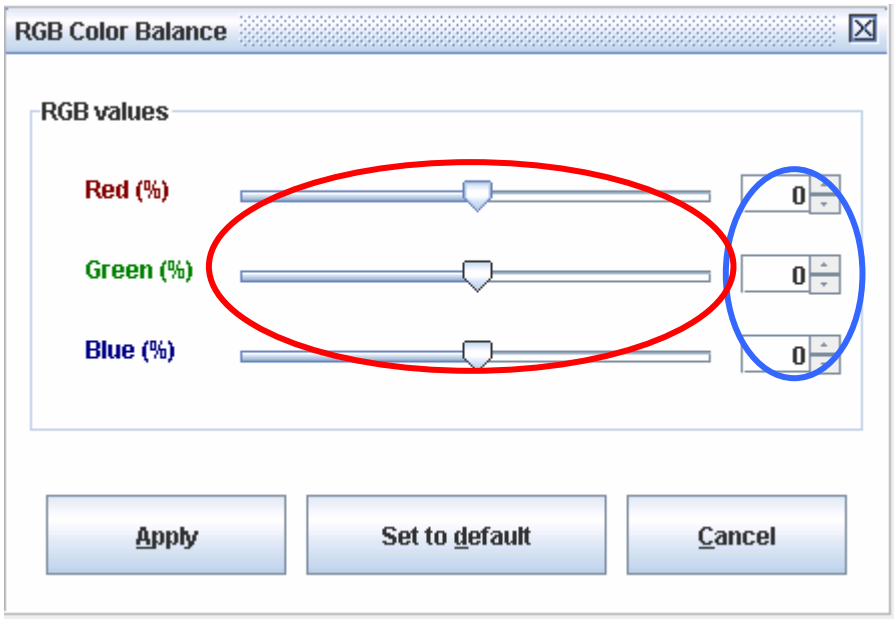


Adjust Red/Green/Blue – prispôsobenie RGB hodnôt
Adjust Hue/Saturation/Brightness- prispôsobenie HSB hodnôt
Contrast/Brighness – zmena kontrastu a jasu
Bright image by 10% - pridanie jasu o 10%
Dark image by 10% - odobratie jasu o 10%
Balance color – vyváženie farieb
Rescale – zvýšenie jasu obrázku

Obr. 21 Karta Adjust

použitím škálovacieho operátora.
Logarithmic/Exponencial operators – operátory, ktoré slúžia na vyváženie jasu pri podexponovaných alebo preexponovaných obrázkoch
Switch channels – zamenenie farebných kanálov obrázku
Negative – vytvorenie negatívu z obrázku
Gray scale – vytvorenie šedého obrázku
Black&white – vytvorenie čiernobieleho obrázku

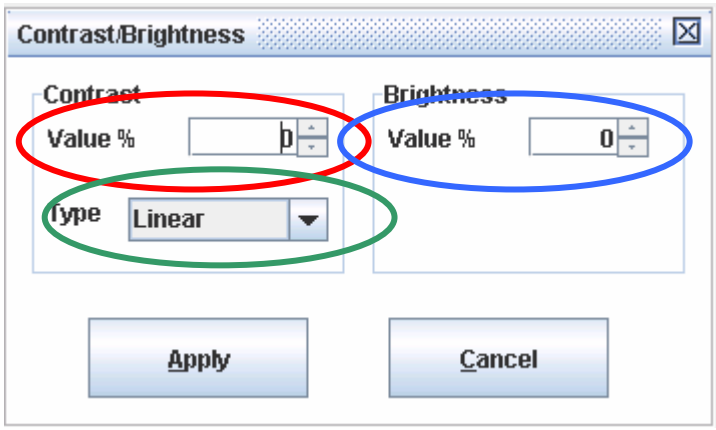
2.1.4.1 Adjust Red/Green/Blue, Adjust Hue/Saturation/Brighness



Obr. 22 Komponenta určená na zmenu RGB resp. HSB

Komponenta je podobná ako pre RGB tak aj pre HSB. Prispôsobenie je možné vykonať buď pomocou posuvníkov (červený ovál) alebo pomocou spinerov (modrý ovál). Hodnoty sa menia v rozsahu -100 až 100 a znamenajú percentá. Tlačidlo „Set to default“ slúži k pôvodnému nastaveniu - 0,0,0.

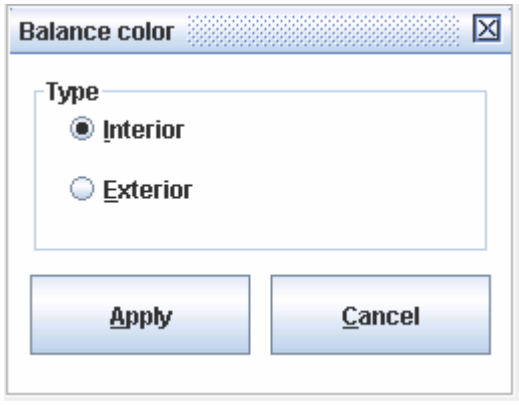
2.1.4.2 Contrast/Brightness



Obr. 23 Komponenta na prispôsobenie kontrastu a jasu.

Hodnoty kontrastu sa menia v rozsahu -200 až 100 (červený ovál), jas v rozsahu -200 až 200 (modrý ovál). Na zmenu kontrastu je možné použiť tri aproximácie – lineárnu, exponenciálnu a logaritmickú (zelený ovál).

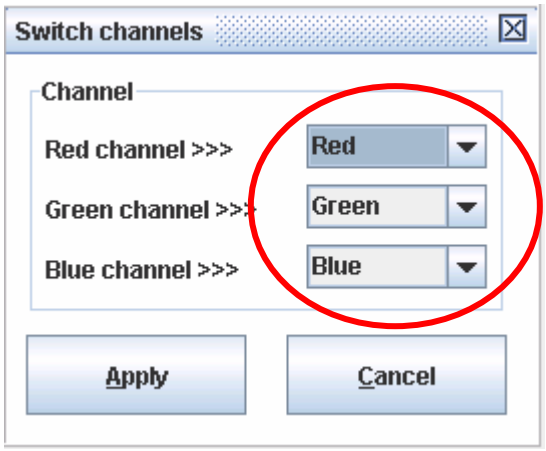
2.1.4.3 Balance color



Obr. 24 Komponenta na vyváženie farieb.

Vyváženie môže byť použité buď na obrázky, ktoré boli vytvorené v interiéry (Interior) alebo v exteriéry(Exterior).

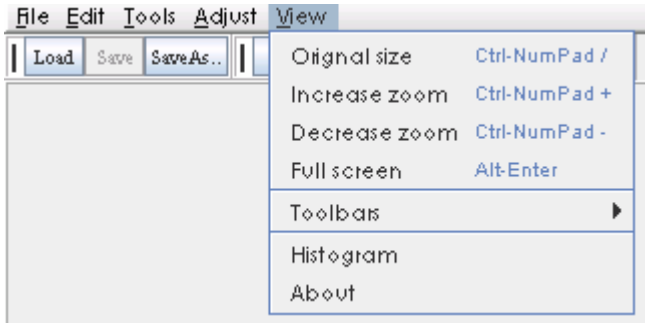
2.1.4.4 Switch channels



Obr. 25 Komponenta na zamenenie kanálov v obrázku.

Vo výberovom poli si vyberieme kanál ktorého hodnoty sa budú priraďovať danému kanálu (červený ovál).
Napr.: Red channel>>>Green, hodnoty zo zeleného kanálu budú pridelené červenému.

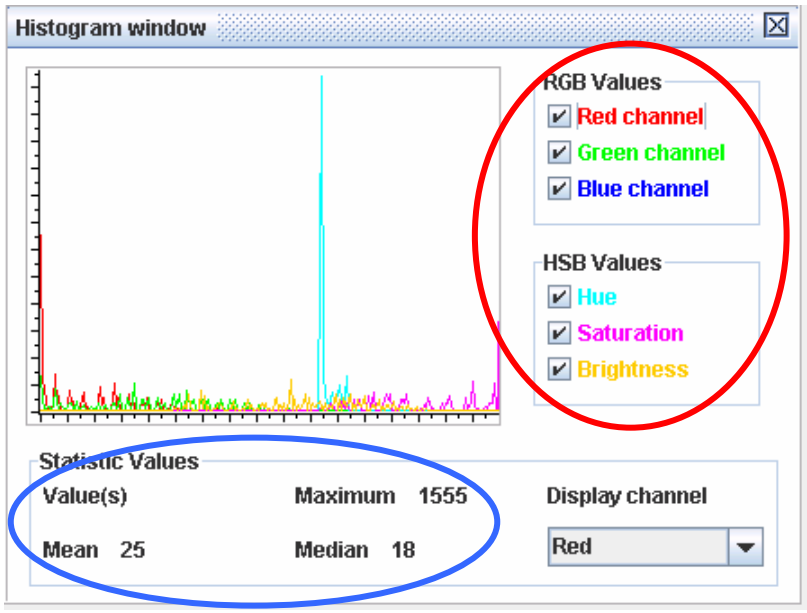
2.1.5 View - Zobrazenie



Orignal size – pôvodné zobrazenie obrázku
Increase zoom – zväčšenie obrázku
Decrease zoom – zmenšenie obrázku
Full screen – zobrazenie obrázku na celú plochu
Toolbars – viditeľnosť toolbarov
Histogram – histogram obrázku
About – informácie o programe

Obr. 26 Karta View.

2.1.5.1 Histogram

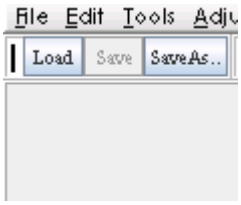


Obr. 27 Komponenta na určenie histogramu

Zaškrtávacie políčka vpravo slúžia k viditeľnosti jednotlivých kanálov (červený ovál). Dolný panel „Statistic Values“ slúži na zobrazenie štatistických údajov z obrázku (modrý ovál) – maximum, stredná hodnota, medián pre kanál, ktorý je zvolený vo výberovom poli (Display channel).

2.2 Toolbary

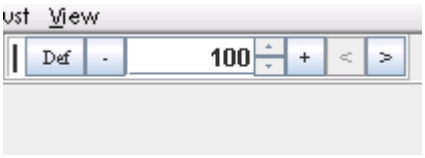
2.2.1 Operačný bar



Load – načítanie obrázku
Save – uloženie obrázku (prepísanie pôvodného)
Save as – uloženie obrázku

Obr. 28 Operačný bar

2.2.2 Zoom bar



Def – pôvodné zobrazenie obrázku
- – zmenšenie obrázku
+ – zväčšenie obrázku
< – undo
> – redo

Obr. 29 Zoom bar

2.2.3 Image preview

V tomto okne sa zobrazuje náhľad obrázku. Výhodné je to pre veľké obrázky, kde vo viditeľnej časti nie je možné vidieť celý obrázok Po dvojitom kliknutí na túto komponentu sa aplikuje pôvodné zobrazenie obrázku.

Obr. 30 Náhľad obrázku

3. Klávesové skratky

V programe boli použité nasledujúce klávesové skratky:

Klávesová skratka	Funkcia
Ctrl+L	načítanie obrázku z disku
Ctrl+U	načítanie obrázku z URL
Ctrl+S	uloženie obrázku
Ctrl+Alt+Q	zatvorenie obrázku
Ctrl+P	nastavenie strany pred tlačou
Alt+Q	ukončenie programu
Ctrl+Z	undo
Ctrk+Z	redo
Shift+I	informácie o obrázku
Ctrl+M	spustenie mediánového filtra
Shift+M	spustenie spriemerovacieho filtra
Ctrl+W	spustenie Wienerovho filtra
Shift+S	zmena veľkosti obrázku
Ctrl++	zväčšenie jasu o 10%
Ctrl+-	zmenšenie jasu o 10%
Ctrl+Alt+N	vytvorenie negatívneho obrázku
Alt+/	pôvodné zobrazenie obrázku
Alt++	zväčšenie obrázku
Alt+-	zmenšenie obrázku
Alt+Enter	režim celej obrazovky