VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION DEPARTMENT OF TELECOMMUNICATIONS

VYTVOŘENÍ JAVA APPLETŮ PRO ZPRACOVÁNÍ OBRAZU

BAKALÁŘSKÁ PRÁCE

BACHELOR´S THESIS

AUTOR PRÁCE — ROMAN ČIŠECKÝ

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ ŠMIRG

SUPERVISOR

BRNO 2010

SEM VLOŽTE ORIGINÁL ZADANIA

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Roman Čišecký ***ID:*** 106396
Ročník: 3 ***Akademický rok:*** 2009/2010

NÁZEV TÉMATU:

Java applety pro zpracování obrazu

POKYNY PRO VYPRACOVÁNÍ:

Bakalářská práce se bude zabývat tvorbou JAVA apletů pro algoritmy na zpracování obrazu. Konkrétně se bude jednat o metody pro vypočet a ekvalizaci histogramu, filtraci v časové a frekvenční oblasti, vytváření panoramatických obrázků a vytváření disparitní mapy.

DOPORUČENÁ LITERATURA:

[1] ŘÍHA, K.: Pokročilé techniky zpracování obrazu. Elektronické texty VUT, Ústav telekomunikací FEKT VUT v Brně, 2007.
[2] ŠMIRG, O., Zpracování obrazu za účelem řízení Visikonu, Bakalářská práce,Brno: VUT, Fakulta elektrotechniky a komunikačních technologií, 2006,50,pp.3
[3] KOHOUTEK,M. Metody analýzy obrazové scény pro řízení videokonferenčních zařízení, Pojednání o disertační práci.Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2006,21s.

Termín zadání: 29.1.2010 ***Termín odevzdání:*** 2.6.2010

Vedoucí práce: Ing. Ondřej Šmirg

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Bakalářská práce se zabývá technikami zpracování obrazu, jako jsou výpočet a ekvalizace histogramu, filtrace obrazu ve frekvenční a v prostorové oblasti, tvorba panoramatických obrázků a vytváření disparitní mapy. V této práci jsou popsané jednotlivé metody zpracování obrazu a jejich využití v praxi. Poslední část se zabývá vlastní tvorbou aplikací na zpracování obrazu a jejich přetvořením na aplety spustitelné z webu.

Klíčová slova

Histogram, ekvalizace histogramu, filtrace obrazu, konvoluce, Fourierova transformace, panorama, perspektivní transformace, homografie, disparita, Java

Abstract

The bachelor´s thesis is concerned with image processing techniques such as calculating histogram and its equalization, image filtering in frequency and time domain, creating panoramic pictures and creating a disparity map This thesis describes the various methods of image processing and their use in practice. The last part deals with creating particular applications for image processing and its transformation to applets executable on the web.

Keywords

Histogram, histogram equalization, image filtering, convolution, Fourier transform, panorama, perspective transform, homography, disparity, Java

Bibliografická citácia mojej práce:

ČIŠECKÝ, R. *Java applety pro zpracování obrazu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 52 s. Vedoucí bakalářské práce Ing. Ondřej Šmirg.

PREHLÁSENIE

Prehlasujem, že som svoju bakalársku prácu na tému „*Java applety pro zpracování obrazu*“ vypracoval samostatne pod vedením vedúceho bakalárskej práce a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích osobnostných autorských práv a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Zb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia § 152 trestného zákona č. 140/1961 Zb.

V Brne dňa 14. mája 2010 .....

(podpis autora)

POĎAKOVANIE

Ďakujem vedúcemu bakalárskej práce Ing. Ondřejovi Šmirgovi a Ing. Zdeňkovi Průšovi za účinnú metodickú pedagogickú a odbornú pomoc a za ďalšie cenné rady pri vypracovávaní mojej bakalárskej práce.

V Brne dňa 19. mája 2010 .....

(podpis autora)

Obsah

Zoznam obrázkov

Úvod

Zadanie bakalárskej práce je zamerané na vytváranie Java apletov pre algoritmy na spracovanie obrazu. V rámci tejto práce boli rozobrané metódy pre výpočet a ekvalizáciu histogramu, filtráciu obrazu vo frekvenčnej a v priestorovej oblasti, vytváranie panoramatických obrázkov a vytváranie disparitnej mapy.

V prvej kapitole budú vysvetlené niektoré základné pojmy úzko späté s problematikou spracovania obrazových dát.

Cieľom druhej kapitoly je vysvetliť problematiku histogramu, jeho ekvalizácie, spôsoby jeho implementácie v programe a jeho využitie pri spracovaní obrazových dát.

Tretia a štvrtá kapitola sa zaoberá filtráciou obrazu. Tretia kapitola je zameraná na filtráciu obrazu v priestorovej oblasti. V úvode kapitoly je vysvetlený spôsob implementácie filtrov v priestorovej oblasti, ktorý je založený na princípe konvolúcie. V tejto časti sú podrobne rozobrané jednotlivé typy filtrov, princípy ich činnosti, výhody a nevýhody ich použitia. Štvrtá kapitola pojednáva o filtrácii obrazu vo frekvenčnej oblasti, je tu vysvetlený jej postup založený na Fourierovej transformácii a popis základných filtrov vo frekvenčnej oblasti.

Náplňou piatej a šiestej kapitoly je tvorba panoramatických obrázkov, resp. tvorba disparitnej mapy. V úvode piatej kapitoly sú bližšie popísané niektoré geometrické transformácie, ktoré sú v tejto oblasti spracovania obrazu často využívané. Taktiež je tu vysvetlený pojem homografie ako aj princíp jej výpočtu. Šiesta kapitola rozoberá problematiku snímania priestorovej scény dvoma kamerami, sú v nej vysvetlené zásady epipolárnej geometrie, ako aj pojem disparita a jej význam.

Siedma a ôsma kapitola sa zaoberá praktickým riešením techník spracovania obrazu, ktoré sú náplňou tejto práce. V siedmej kapitole sú podrobne popísané aplikácie na spracovanie obrazu vytvorené v programovacom jazyku Java, resp. Matlab. V ôsmej kapitole je popísaná problematika prevodu aplikácií písaných v jazyku Java na aplety, ako aj postup pri zavádzaní natívnych knižníc do apletov. V prílohe práce sú uvedené najdôležitejšie zdrojové kódy k týmto programom.

1 Spracovanie obrazu a jeho interpretácia

1.1 Interpretácia obrazu a jeho parametrov

Pre lepšie porozumenie problematiky spracovania obrazu začnem s vysvetlením základných pojmov súvisiacich s touto tematikou. Informácie čerpané z [2] a [3].

Definícia obrazu a jeho vznik: Obraz možno vo všeobecnosti získať použitím vhodného média (napr. digitálny fotoaparát) v rôznych oblastiach viditeľného spektra, akým je napríklad denné svetlo, prípadne aj z iných oblastí žiarenia, ktoré ľudské oko nevníma. Príkladom sú röntgenové snímky, infračervené snímky a podobne. Obraz teda predstavuje dvojrozmerné zobrazenie trojrozmernej scény.

Jas obrazu: Jas udáva množstvo svetelnej energie prijatej zo zachytávanej scény. Veľkosť jasu závisí od zdroja svetla, jeho polohy voči scéne, od povrchu a reflektivity predmetov. Vo farebnom priestore RGB predstavuje jas obrazu aritmetický priemer červenej, zelenej a modrej farebnej zložky.

Kontrast: Je to vlastnosť obrazu, ktorá určuje vzdialenosť medzi dvomi jasovými hladinami dvoch susedných oblasti obrazových bodov s rovnakým jasom.

Sýtosť farieb: Sýtosť farby je daná kombináciou intenzity svetla, a jej rozložením do spektra rôznych vlnových dĺžok. Najväčšiu sýtosť dosiahneme pri farbe práve jednej vlnovej dĺžky s vysokou intenzitou. Platí teda, že pri poklese intenzity sa zníži aj sýtosť farby.

Farebná hĺbka: Farebná hĺbka obrazu je určená počtom farieb, ktoré je možné v obraze uplatniť. Každý bod obrazu je daný hodnotou, v ktorej je zakódovaná farba. Údaje sú vyjadrené v bitoch, napr. pre farebnú hĺbku s názvom TrueColor je každý bod reprezentovaný 24 bitovou hodnotou, teda každá zložka RGB obsahuje 256 odtieňov. U monochromatického obrazu je farebná hĺbka určená hladinami jasu, ktorý je reprezentovaný úrovňami šedej v intervale od čiernej po bielu.

1.2 Spracovanie obrazu

1.2.1 Vzorkovanie a kvantovanie

S digitalizáciou obrazu sú úzko späté dva dôležité termíny: vzorkovanie a kvantovanie.

Vzorkovanie sa uskutočňuje tým spôsobom, že vodorovnú os (x) rozdelíme na rovnomerné úseky a z každého úseku odoberieme jednu vzorku. Je zrejmé, že z pôvodného signálu stratíme mnoho detailov, pretože zo spojitej čiary dostaneme len množinu diskrétnych bodov s intervalom použitej vzorkovacej frekvencie. Ak rozmedzie dvoch vzoriek je Δx, potom bude vzorkovacia frekvencia $f_{vz} = 1/\Delta x$. Platí tu Shannonov teorém, ktorý hovorí, že signál (v tomto prípade obraz) možno rekonštruovať bez straty informácie, pokiaľ je vzorkovaný s frekvenciou aspoň dvojnásobnou ako je maximálna frekvencia signálu použitá v obraze. Pri prekročení maximálnej frekvencie dochádza k aliasingu.

Kvantovanie je proces, pri ktorom je nameranej veličine priradená diskrétna hodnota. rozlišujeme dva typy kvantovania: uniformné (rovnomerné), pri ktorom je vzdialenosť medzi jednotlivými úrovňami jasu f rovnaká, a neuniformné, pri ktorom je naopak interval medzi úrovňami jasu rôzny. Pri kvantovaní dochádza ku strate informácie, vzniká kvantizačná chyba. Pre zobrazenie N jasových úrovní je potrebný určitý počet bitov, pričom platí: $N = 2^K$, ide K je počet bitov. Informácie čerpané z [3]

1.2.2 Digitalizácia obrazu

Obraz uvažujeme ako 2D funkciu $f(x,y)$, kde x a y sú priestorové rovinné súradnice a funkčnou hodnotou funkcie f je jas obrazu v danom mieste. Funkciu $f(x,y)$ je možno definovať pomocou parciálnych funkcií charakterizujúcich zdroj osvetlenia – svietivosť $i(x,y)$, a zobrazovaný objekt – reflektivita $r(x, y)$:

$$(x, y) = i(x, y)\, r(x, y), \tag{1.1}$$

kde $0 < i(x, y) < \infty$

a $0 < r(x, y) < 1,$

pričom $r = 0$ zodpovedá úplnej absorpcii svetla a naopak pri $r = 1$ nastáva úplný odraz. Pre digitálne spracovanie obrazu treba vykonať operácie na diskretizáciu súradníc a jasu, k čomu slúži vzorkovanie a kvantovanie popísané v kapitole 1.2.1. Pri vzorkovaní sa obraz rozdelí na konečný počet pixelov, ktorým je pri kvantovaní priradená hodnota jasu z daného súboru diskrétnych hodnôt. Výsledok vzorkovania a kvantovania predstavuje maticu reálnych čísel, kde súradnice x predstavujú riadkový index a súradnice y stĺpcový index. Digitálny obraz možno vyjadriť ako dvojrozmernú funkciu:

$$f(x, y) = \begin{bmatrix} f\,(0,0) & f\,(0,1) & \dots & f\,(0, N-1) \\ f\,(1,0) & f\,(1,1) & \dots & f\,(1, N-1) \\ \vdots & \vdots & & \vdots \\ f\,(\mathrm{M}-1,0) & f\,(\mathrm{M}-1,1) & \dots & f\,(M-1, N-1) \end{bmatrix} \tag{1.2}$$

alebo ako maticu:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \dots & a_{M-1,N-1} \end{bmatrix}. \tag{1.3}$$

Pri digitalizácii môžu hodnoty M, N nadobúdať ľubovoľné prirodzené čísla a každý pixel môže mať L diskrétnych úrovni šedej, pričom

$$L = 2^k, \tag{1.4}$$

kde k = 0, 1, 2, ... Súbor všetkých jasových hladín tvorí vektor $\langle 0, L-1 \rangle$. Informácie boli čerpané z literatúry [1] .

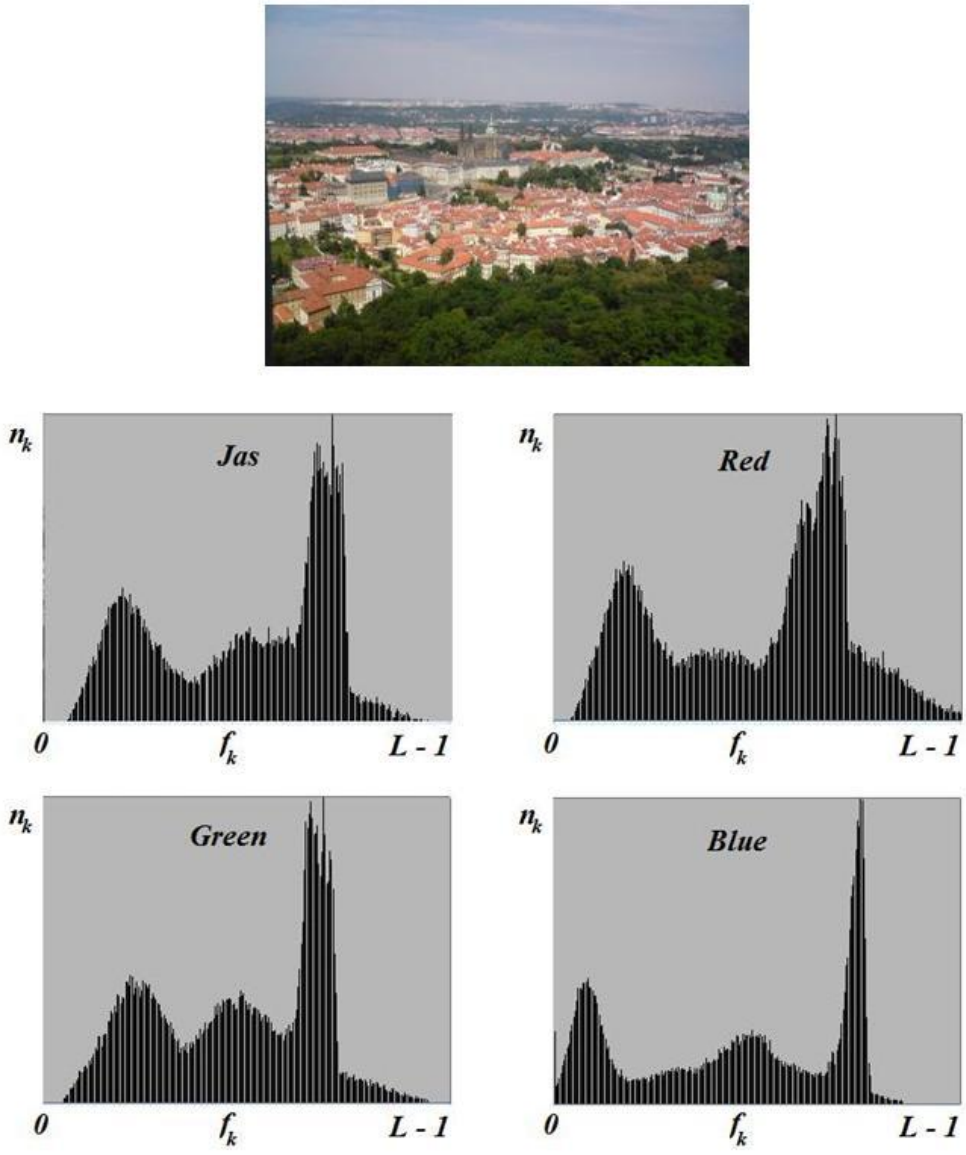
2 Metódy pre výpočet a ekvalizáciu histogramu

2.1 Histogram

Histogram predstavuje rozloženie jasových úrovní v obraze, teda početnosť jednotlivých hladín jasu. Histogram digitálneho obrazu, s úrovňami jasu (0,1,..., L-1) je diskrétna funkcia

$$h(f_k) = n_k, \tag{2.1}$$

kde f_k predstavuje k-tú úroveň jasu (k = 0,1,..., L – 1) a n_k udáva počet pixelov v obraze s úrovňou jasu f_k. Grafickým znázornením histogramu je stĺpcový diagram, v ktorom sú na horizontálnu os vynesené hodnoty f_k a os je rozdelená na L hodnôt. Vertikálnu os predstavuje početnosť n_k. Vo farebnom obraze bude každej farebnej zložke (RGB) zodpovedať jeden histogram, pričom histogramy jednotlivých zložiek sa môžu výrazne líšiť, ako to znázorňuje ***Obr. 2.1***.



Obr. 2.1 Histogram obrazu a jednotlivých zložiek RGB

Z *Obr. **2.1*** je zrejmé, že histogramu jasu (šedotónoveho obrazu) je najviac podobný histogram zelenej zložky. Je to spôsobené tým, že každá zložka farebného priestoru RGB má iný vplyv na celkový jas. Podiel jednotlivých farebných zložiek na celkovom jase je nasledujúci: červená 29,9%, zelená 58,7%, modrá 11,4%.

Pre praktické použite je výhodné histogram normalizovať. Normalizácia spočíva v podelení jednotlivých početností n_k celkovým počtom pixelov v obraze *N*.

$$p(f_k) = n_k/N. \tag{2.2}$$

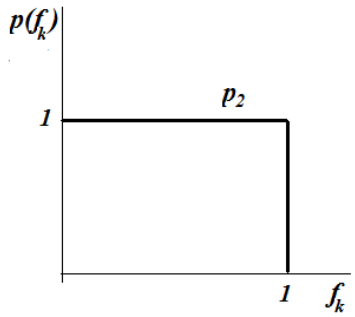
Normalizovaný histogram udáva odhad pravdepodobnosti výskytu jednotlivých úrovní jasu v obraze. Histogramy tvoria základ pre mnohé techniky spracovania obrazových dát, možno ich využiť napríklad pri segmentácii a kompresii digitálneho obrazu. Čerpané z lit. [3].

2.2 Ekvalizácia histogramu

V prípade, že máme nevhodne exponovaný obraz (nekontrastný, jasové zložky sú rozložené nerovnomerne), ako je napríklad na ***Obr. 2.3***, chceme docieliť toho, aby boli detaily v obraze lepšie rozlíšiteľné a kontrastnejšie. Využíva sa k tomu metóda ekvalizácie (vyrovnania) histogramu. Pomocou transformácie dosiahneme toho, aby sa všetky úrovne jasu v obraze vyskytovali rovnako často. V ideálnom prípade by histogram takéhoto obrazu vyzeral ako na ***Obr. 2.2***. Je však zrejmé, že v praxi pri diskrétnom obraze k dokonalej ekvalizácii väčšinou nedochádza a niektoré hladiny jasu sa ani nemusia vyskytnúť. Cieľom tejto transformácie je získať obraz s rovnomerným rozdelením hustoty pravdepodobnosti jasových úrovní. Táto transformácia predstavuje výpočet tzv. kumulatívnej distribučnej funkcie CDF, kde *k*-tú úroveň jasu g_k vypočítame:

$$g_k = \sum_{j=0}^{k} \frac{n_j}{N}, \tag{2.3}$$

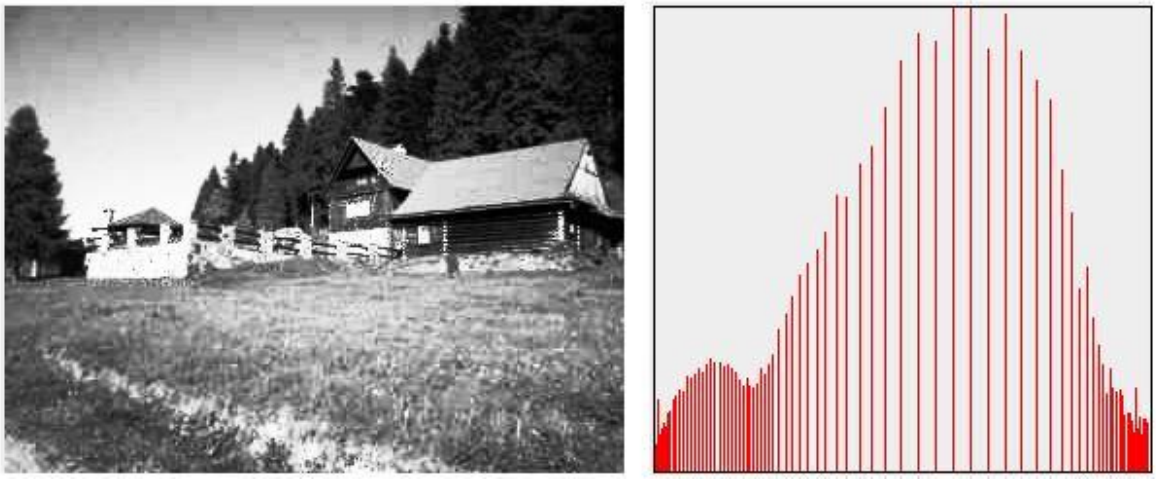
kde n_j je početnosť *j*-tej úrovne jasu f_j a *N* je celkový počet pixelov v obraze. Vzťah bol prevzatý z literatúry [3], kde je aj podrobne odvodený. Výsledné jasové úrovne f_j treba opäť previesť do intervalu diskrétnych hodnôt $\langle 0, L-1 \rangle$. Príklad ekvalizácie histogramu nekontrastného histogramu je na ***Obr. 2.4***



Obr. 2.2 Ideálne ekvalizovaný histogram

Obr. 2.3 Príklad nekontrastného obrazu (vľavo) a jeho histogram (vpravo)

Obraz s nízkym kontrastom je charakteristický tým, že má úzky histogram, ktorý je sústredený približne do stredu jasového rozsahu.



Obr. 2.4 Ekvalizovaný histogram (vpravo) a obraz po ekvalizácii (vľavo)

Na ekvalizovanom histograme je vidieť, že jednotlivé jasové zložky nemajú rovnakú úroveň, aká by bola v prípade spojitej funkcie hustoty pravdepodobnosti. Výhoda tejto metódy spočíva predovšetkým v automatickom výpočte bez zadávania ďalších parametrov, ktorý je založený iba na dátach získaných zo zdrojového obrazu.

3 Filtrácia obrazu v časovej oblasti

Pod pojmom filtrácia vo všeobecnosti rozumieme proces, pri ktorom sa určitá časť systému prepúšťa a iná časť sa zadržiava, prípadne zoslabuje. Nástrojom tohto procesu je filter. Pod filtráciou dát rozumieme nelokálny matematický proces, ktorý transformuje dáta takým spôsobom, že štruktúry určitého charakteru zosilňuje a iné naopak zoslabuje alebo potlačuje. Kvôli spomenutej nelokálnosti neexistuje spätná transformácia, ktorá by bola schopná z dát po filtrácii plne rekonštruovať pôvodné dáta. Dochádza teda k strate informácie.

Filtrácia signálov v časovej oblasti spočíva vo vyhladzovaní a priemerovaní dát v časovom smere. Keďže obraz uvažujeme ako 2D funkciu $f(x,y)$, môžeme hovoriť o filtrácii v priestorovej oblasti. U tohto druhu filtrácie je často využívaný princíp tzv. konvolúcie, preto bude najskôr vysvetlený tento pojem. Čerpané z [4].

3.1 Konvolúcia

Konvolúcia je matematická operácia dvoch funkcií $f(x,y)$ a $h(x,y)$, ktorá je často využívaná pri filtrácii obrazu. Funkciu $h(x,y)$ nazývame maska, prípadne konvolučné jadro (anglicky *kernel*). Konvolúciu možno rozviesť do troch krokov:

- Obrátenie masky h o 180º,
- Posun masky h vzhľadom k funkcii f pomocou zmeny x, y,
- Výpočet sumy súčinov všetkých koeficientov masky h pre každé posunutie (x, y).

Pre jednorozmernú funkciu a masku obsahujúcu 3 prvky možno konvolúciu vyjadriť ako:

$$g(x) = \sum_{s=-1}^{1} h(s).f(x-s) = h(x-1).f(x+1) + h(x).f(x) + h(x+1).f(x-1). \quad (3.1)$$

Pre dvojrozmernú obrazovú funkciu $f(x,y)$ s maskou 3×3 konvolúciu vyjadríme ako:

$$g(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} h(s,t).f(x-s, y-t). \quad (3.2)$$

Konvolúciu zapisujeme ako

$$g(x,y) = h * f = \sum_{s=-s_{max}}^{s_{max}} \sum_{t=-t_{max}}^{t_{max}} h(s,t).f(x-s, y-t). \quad (3.3)$$

Informácie boli čerpané z [1] a [3].

3.2 Vyhladzovanie obrazu

Obraz je ovplyvňovaný nežiaducimi javmi, ako napr. šum. Na potlačenie šumu, prípadne iných nežiaducich artefaktov v obraze slúži vyhladzovanie obrazu (angl. *image smoothing*). Metódy vyhladzovania obrazu sa rozdeľujú do dvoch kategórii:

a) lineárne
b) nelineárne

Myšlienkovo sú tieto metódy príbuzné filtrom typu dolná priepusť. Čerpané z [5]

3.2.1 Lineárne metódy vyhladzovania

Tieto metódy sú charakteristické tým, že novú hodnotu daného pixelu vypočítajú ako lineárnu kombináciu hodnôt zvoleného okolia. V prípade digitálneho obrazu možno lineárnu kombináciu určiť ako diskrétnu konvolúciu, ktorá je popísaná v kapitole **3.1** Jednotlivé lineárne filtre sa v podstate líšia len použitou konvolučnou maskou *h* (*x*,*y*).

Základnou metódou vyhladzovania obrazu je *obyčajné priemerovanie*, ktoré zaručuje, že výsledkom konvolúcie bude priemer hodnôt jasu z okolia bodu v obraze:

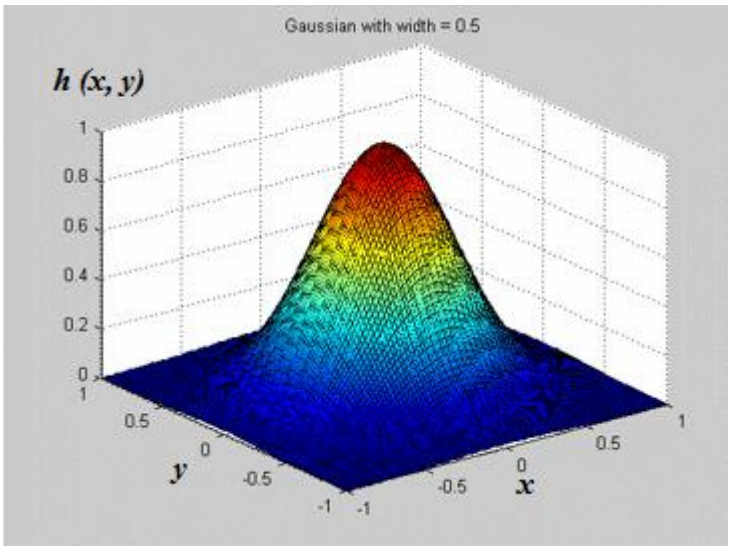
$$g(x,y) = \frac{1}{\mathrm{M}} \sum_{s=-\mathrm{s_{max}}}^{\mathrm{s_{max}}} \sum_{t=-\mathrm{t_{max}}}^{\mathrm{t_{max}}} f(x-s, \mathrm{y}-\mathrm{t}), \tag{3.4}$$

kde M udáva počet bodov v maske, M = (2.s_{max} + 1). (2.t_{max} + 1). Konvolučná maska tejto metódy pre okolie 3 × 3 má tvar:

$$h = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{3.5}$$

Nevýhodou praktického použitia metódy obyčajného priemerovania je rozmazávanie hrán v obraze, preto sa táto metóda väčšinou používa ako pomocná metóda pre výpočet strednej hodnoty jasu. Tento medzivýsledok sa ďalej používa v nelineárnych metódach vyhladzovania. Informácie čerpané z [5].

Ďalšou metódou je tzv. *Gaussovo vyhladzovanie*, typické tým, že koeficienty bližšie k stredu masky majú vyššiu váhu a zodpovedajú hodnotám na Gaussovej krivke.



Obr. 3.1 Ukážka Gaussovho rozdelenia (zdroj: Matlab)

Dvojrozmerné Gaussovo rozdelenie so strednou hodnotou je definované vzťahom:

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{3.6}$$

kde σ^2 je rozptyl. Od parametru σ závisí strmosť Gaussovej funkcie a zároveň aj veľkosť masky. Informácie čerpané z [3].

3.2.2 Nelineárne metódy vyhladzovania

Problémy spojené s rozmazávaním hrán dokážu čiastočne eliminovať nelineárne vyhladzovacie metódy. Znakom týchto metód je to, že sa v skúmanom okolí snažia nájsť tú časť, do ktorej patrí reprezentatívny bod. Iba pixely tejto oblasti sa využívajú pre hľadanie jasovej hodnoty, ktorá bude reprezentovať celé okolie vo výstupnom obraze.

Príkladom metódy, ktorá filtruje len vo vybranom okolí je *metóda rotujúcej masky*. Podľa homogenity (napr. rozptylu) jasu hľadá k filtrovanému bodu časť jeho okolia , ku ktorej pravdepodobne patrí, a tu potom použije pre výpočet. Metóda nerozmazáva hrany obrazu, dokonca má mierne ostriaci charakter. Čerpané z [5].

Ďalšou z kategórie nelineárnych metód je tzv. *mediánová filtrácia*. Princíp filtrácie je založený na posúvaní myslenej masky po obraze a výberu mediánu z hodnôt ležiacich pod touto maskou. Jedná sa o štatistický filter, teda nejde o konvolúciu. Pri hľadaní mediánu je treba najskôr vstupné hodnoty pixelov usporiadať podľa veľkosti od najmenšieho po najväčšie (vzostupne). Medián predstavuje hodnotu prvku s poradovým číslom čo najbližším polovici počtu vstupných prvkov. Výhodou je použitie masky s nepárnym počtom prvkov, ak je počet prvkov nepárny, medián sa určí ako aritmetický priemer oboch prostredných hodnôt. Mediánová filtrácia vykazuje v špecifických prípadoch veľmi dobré vlastnosti, napr. pri potlačovaní šumu typu „soľ a korenie“ ako je vidieť na **Obr. 3.2**. Nevýhodou tejto metódy je, že pri obdĺžnikovom okolí porušuje ostré hrany a tenké čiary v obraze. Čerpané z [1], [3] a [5].

Obr. 3.2 Odstránenie šumu "soľ a korenie" pomocou mediánovej filtrácie

3.3 Detekcia hrán

Hranou v obraze sa rozumie prudká (ideálne skoková) zmena intenzity jasu. Detekcia hrán teda spočíva v hľadaní miest v obraze, kde sa jas významne mení. Kritériom detekcie týchto zmien je veľkosť prvej, resp. druhej derivácie intenzity jasu alebo detekcia zmeny znamienka derivácie. Metódy detekcie hrán sú príbuzné tzv. hornopriepustným filtrom. Informácie čerpané z [3].

3.3.1 Detekcia obrazových hrán pomocou prvej derivácie

Výsledkom prvej derivácie obrazu v smere x a y je gradient. Gradient funkcie dvoch premenných $f(x,y)$ možno definovať ako vektor

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}. \tag{3.7}$$

Veľkosť tohto vektoru je

$$|\nabla f| = \sqrt{{G_x}^2 + {G_y}^2} = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}. \tag{3.8}$$

Z dôvodu, že pri stanovení gradientného obrazu je potrebné vypočítať veľkosť vektoru pre každý pixel, býva táto operácia nahradená vzťahom:

$$|\nabla f| \approx |G_x| + |G_y|. \tag{3.9}$$

Táto náhrada sa navonok chová ako derivácia: v oblastiach s konštantnou hodnotu jasu je nulová, pričom jej hodnota závisí na zmene intenzity jasu. Smer gradientu určuje uhol $\varphi(x,y)$, ktorý je určený vzťahom

$$\varphi(x,y) = \arctan\frac{G_y}{G_x}. \tag{3.10}$$

Informácie boli čerpané z [1] a [3].

Existuje viacero operátorov pre detekciu hrán založených na princípe gradientu. Gradientné operátory možno rozdeliť do dvoch skupín:

- izotropné operátory – reagujú na všetky hrany v obraze, bez ohľadu na smer
- anizotropné operátory – reagujú na len na hrany v určitom smere.

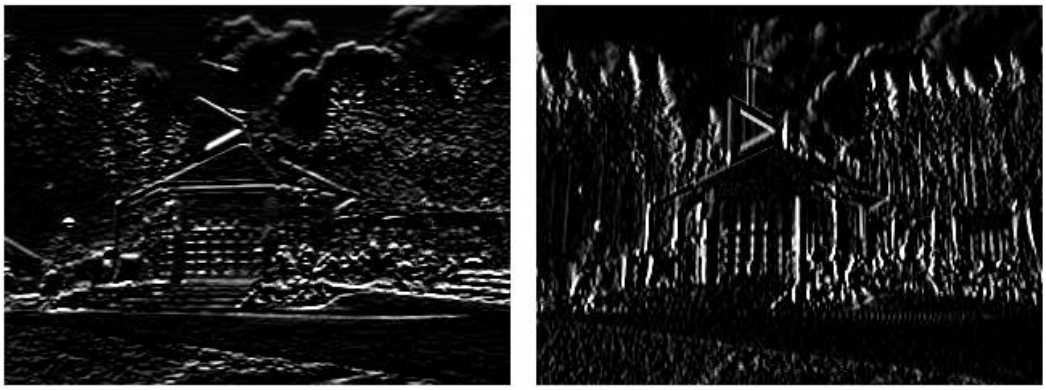
V nasledujúcej časti budú popísané operátory, ktoré boli použité v rámci tejto práce.

Gradientné operátory podľa Prewittovej

Patria do skupiny anizotropných operátorov. Gradient sa počíta z okolia 3×3, pričom je vybraná tá maska, ktorej zodpovedá najväčší modul gradientu. Výsledky filtrácie operátormi Prewittovej sú na ***Obr. 3.4***

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.11}$$

Obr. 3.3 Originálny obraz pred filtráciou

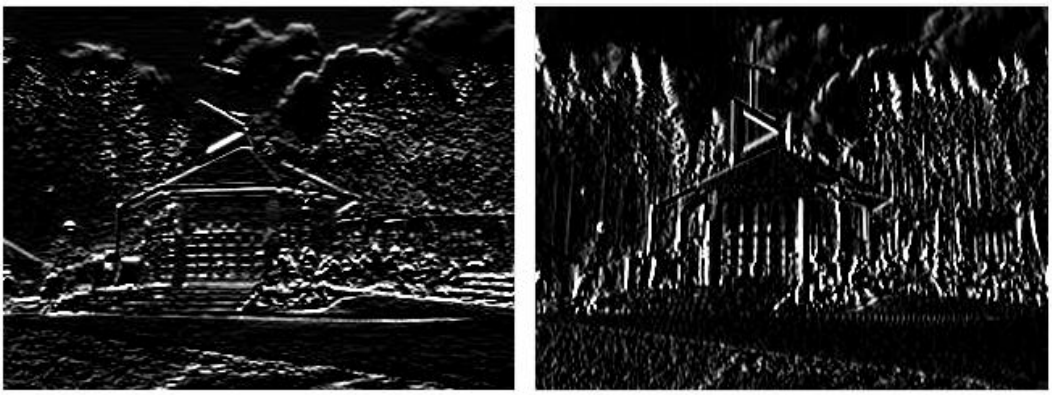


Obr. 3.4 Obraz po filtrácii operátorom Prewittovej v smere x, y

Sobelov gradientný operátor

Tento operátor možno použiť na detekciu vodorovných a zvislých hrán, je dokonca možné masky rotovať o 45º a počítať tak smerové derivácie nielen v smere x, y. Použitie konštánt 2 a -2 v pozícii stredných koeficientov vedie k zdôrazneniu stredných pixelov, čo má za následok určité vyhladenie priebehu výslednej funkcie. Na ***Obr. 3.5*** je zobrazený obraz po filtrácii Sobelovým operátorom.

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.12}$$



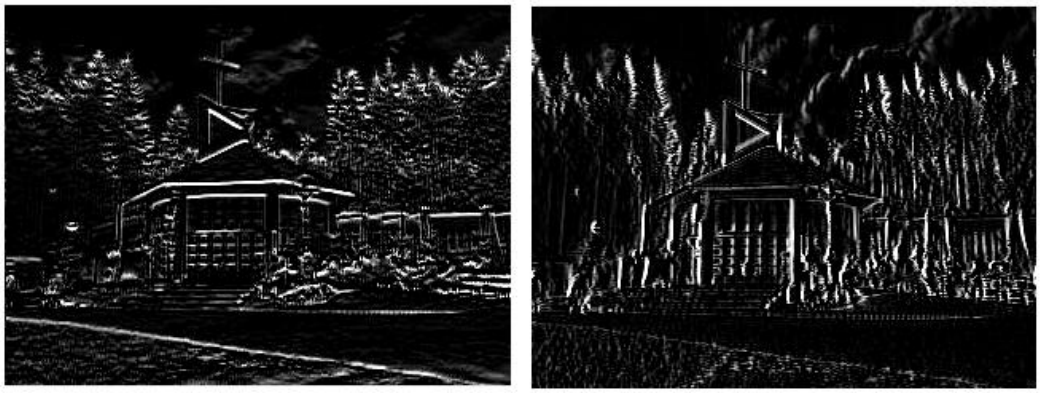
Obr. 3.5 Obraz po filtrácii Sobelovým operátorom v smere x, y

Robinsonov gradientný operátor

Tento operátor je v princípe podobný operátoru podľa Prewittovej, jeho konvolučné masky pre smer x, y majú tvar:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \tag{3.13}$$

Ostatné masky sa možno získať ich rotáciou, podobne ako u Sobelovho operátora. Ukážka filtrácie Robinsonovým operátom je na ***Obr. 3.6***



Obr. 3.6 Obraz po filtrácii Robinsonovým operátorom v smere x, y

Informácie boli čerpané z [1], [2] a [3].

3.3.2 Detekcia obrazových hrán pomocou druhej derivácie

Druhá derivácia predstavuje rýchlosť zmeny jasových hodnôt a prejavuje sa predovšetkým na strmých a izolovaných hrán, prípadne izolovaných bodov. Z tohto poznatku vyplýva, že druhá derivácia bude zvýrazňovať šum. Najjednoduchším izotropným operátorom aproximujúcim druhou deriváciou obrazovej funkcie f (x,y) je Laplaceov operátor. Čerpané z [3].

Laplaceov operátor

Z hľadiska výpočtu Laplacián zaujíma len veľkosť gradientu, bez ohľadu na jeho smer, ako tomu bolo u gradientných operátorov aproximujúcich prvou deriváciou funkcie $f(x,y)$. Laplaceov operátor vychádza z druhých parciálnych derivácii

$$\nabla^2 f = \left[\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right]. \tag{3.14}$$

Pre implementáciu Laplaciánu sa používajú konvolučné masky pre 4-okolie a 8-okolie.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.15}$$

Laplacián je charakteristický tým, že v obraze zvýrazňuje nespojitosti a naopak v oblastiach s pozvoľnou zmenou hodnôt jasu potlačuje jas. To má za následok, že hrany a iné nespojitosti sú svetlé a oblasti s konštantným jasom sú naopak tmavé, ako je to vidieť na **Obr. 3.7.**

Obr. 3.7 Obraz po Laplaciánovej filtrácii pre 4-okolie

Superpozíciou originálneho obrazu a výsledku filtrácie obrazu Laplaciánom je možné docieliť efektu „ostrenia obrazu“, ktorý má za následok zvýraznenie hrán, viď ***Obr. 3.8***. Tvar takejto superponovanej masky je

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.16}$$

Obr. 3.8 Výsledok operácie ostrenie obrazu

Informácie čerpané z [1] a [3].

4 Filtrácia obrazu vo frekvenčnej oblasti

Použitie frekvenčnej oblasti pre spracovanie obrazu je založené na myšlienke francúzskeho matematika Jean Baptiste Joseph Fouriera, podľa ktorej možno každú funkciu $f(x)$ rozložiť na súčet sínusových a kosínusových harmonických funkcií rôznych frekvencií, vynásobených váhovým koeficientom (v prípade periodickej funkcie, hovoríme o Fourierovej rade), resp. váhovaciou funkciou (v prípade neperiodickej funkcie, jedná sa o Fourierovú transformáciu). Matematické odvodenie Fourierovej transformácie pre funkciu dvoch premenných je uvedené v lit. [1], v kap. 6.2. Informácie čerpané z [1].

Filtrácia vo frekvenčnej oblasti pozostáva z 3 krokov (čerpané z [4]):

- Prevod obrazu do frekvenčnej oblasti pomocou Fourierovej transformácie – vzniká frekvenčné spektrum $F(u, v)$,
- Vynásobenie frekvenčného spektra filtračnou funkciou (filtrom) $H(u, v)$, čo spôsobí lokálne zvýšenie, zníženie, prípadne anulovanie amplitúd určitých harmonických funkcií – vznikne filtrované spektrum,
- Prevod filtrovaného spektra pomocou inverznej Fourierovej transformácie späť do priestorovej oblasti.

Operáciu filtrácie teda možno zapísať vzťahom

$$G(u,v) = H(u,v).F(u,v), \tag{4.1}$$

kde $G(u,v)$ je Fourierová transformácia filtrovaného obrazu, ktorý je potom vypočítaný ako

$$g(x,y) = \mathcal{F}^{-1}\left(G(u,v)\right). \tag{4.2}$$

Matematicky je možno dokázať, že oba spôsoby filtrácie – v priestorovej a frekvenčnej oblasti sú ekvivalentné a dávajú identický výsledok, ak je násobiaci filter vo frekvenčnej oblasti Fourierovým obrazom konvolučného filtra v priestorovej oblasti. Súvislosť medzi oboma spôsobmi filtrácie je popísaná v lit. [1], kap. 7.2. Informácie čerpané z [1].

4.1 Základné typy filtrov vo frekvenčnej oblasti.

4.1.1 Odstránenie jednosmernej zložky (Notch filter)

V každom obraze existuje taký prvok Fourierovej transformácie, ktorý nesie informáciu o priemernej hodnote jasu všetkých pixelov v obraze. Ak by sme tento prvok nastavili na nulu, bude priemerná hodnota všetkých pixelov v obraze nulová, pretože od každej hodnoty bude odpočítaná hodnota priemerného jasu. Zobrazenie výsledkov obsahujúcich záporné hodnoty jasu je možné realizovať buď nulovaním všetkých záporných hodnôt, alebo ich normovaním do podporovaného rozsahu (využíva sa častejšie). Po spätnej Fourierovej transformácii obraz nebude obsahovať jednosmernú zložku (namiesto pozadia a veľkých objektov v obraze bude čierna plocha). Tento filter, nazývaný tiež *Notch filter*, predstavuje úzkopásmovú zádrž a možno ho popísať prenosovou funkciou

$$H(u,v) = \begin{cases} 0 & ak\ (u,v) = \left(\frac{M}{2}, \frac{N}{2}\right) \\ 1 & inak \end{cases}, \tag{4.3}$$

kde M × N predstavuje veľkosť obrazovej funkcie $f(x, y)$. Čerpané z [1].

4.1.2 Dolno-priepustný filter (Low pass filter)

Filtre typu dolná priepusť slúžia k vyhladzovaniu obrazu, prípadne k potláčaniu štatistických fluktuácii. Ideálny dolno-priepustný filter orezáva všetky harmonické zložky spektra, ktorých vzdialenosť od počiatku spektra je väčšia než definovaná hranica D_0. Prenosovú funkciu tohto ideálneho dolno-priepustného filtra možno zapísať

$$H(u,v) = \begin{cases} 0 & ak\ D(u,v) > D_0 \\ 1 & ak\ D(u,v) \leq D_0 \end{cases}, \tag{4.4}$$

kde $D(u,v)$ predstavuje vzdialenosť bodu (u,v) od počiatku filtračnej funkcie $H(u,v)$. Vplyv takejto filtrácie je vidieť na ***Obr. 4.1***. Čerpané z [1].

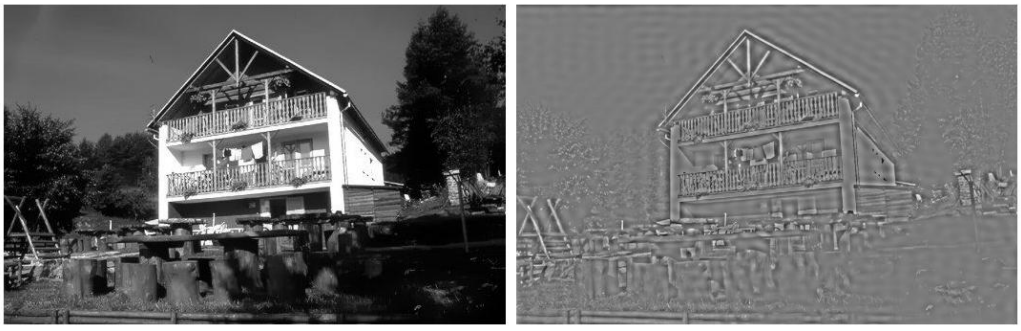
Obr. 4.1 Obraz pred (vľavo) a po filtrácii dolno-priepustným filtrom (vpravo)

4.1.3 Horno-priepustný filter (High pass filter)

Na základe analógie s predošlým filtrom je zrejmé, že ideálny horno-priepustný filter bude orezávať všetky nízkofrekvenčné zložky spektra, ktorých vzdialenosť od počiatku centralizovaného spektra bude menšia ako definovaná hranica D_0. Prenosová funkcia filtru má tvar

$$H(u,v) = \begin{cases} 0 & ak\ D(u,v) \leq D_0 \\ 1 & ak\ D(u,v) > D_0 \end{cases}, \tag{4.5}$$

Príklad obrazu po uplatnení tohto filtru je na ***Obr. 4.2***. Čerpané z [1].



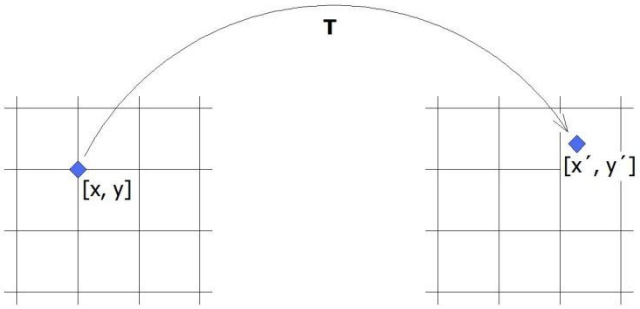
Obr. 4.2 Obraz pred (vľavo) a po filtrácii horno-priepustným filtrom (vpravo)

5 Tvorba panoramatických obrázkov

Pri tvorbe panoramatických obrázkov nám ide o napodobenie vnímania priestoru z jedného miesta. V rámci tejto kapitoly bude rozobratá problematika softvérového skladania snímok do panorámy. Jednotlivé snímky sú vyjadrené v rôznych súradnicových sústavách, preto treba nájsť transformácie všetkých obrazov do spoločného súradnicového systému. V úvode kapitoly bude preto rozobratá problematika geometrických transformácii.

5.1 Geometrické transformácie

Každý bod v rovine je určený dvojicou súradníc $[x, y]$ z R^2. Ak uvažujeme R^2 ako vektorový priestor, možno povedať, že bod je určený vektorom $[x, y]$. Geometrické transformácie umožňujú prevádzať bod $[x, y]$ z pôvodného obrazu do bodu $[x', y']$ v obraze výslednom (viď ***Obr. 5.1***), pričom výsledkom transformácie býva reálne číslo. Aby výsledné súradnice korešpondovali celočíselnej mriežke obrazu, treba použiť vhodnú interpoláciu. Čerpané z [1].



Obr. 5.1 Ukážka mapovania súradníc z pôvodného do výsledného obrazu

5.1.1 Euklidovské transformácie

Jedná sa o najčastejšie používané lineárne transformácie zahŕňajúce transláciu a rotáciu. Je pre ne typické, že zachovávajú dĺžky strán, veľkosti uhlov, ako aj rovnobežnosť strán. **Transláciu**, resp. posunutie bodu v rovine možno pomocou súradníc bodu v R^2 zapísať ako

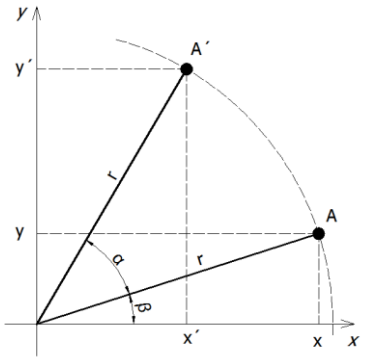
$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}, \tag{5.1}$$

kde t_x a t_y určujú veľkosť posuvu v danom smere. Zápis vo vektorovom tvare bude

$$x' = x + t, \text{ kde } t = \left[t_x, t_y\right]^T. \tag{5.2}$$

Rotáciou sa rozumie otočenie bodu o uhol β vzhľadom na počiatok súradnicovej sústavy proti smeru hodinových ručičiek. Bod $X[x, y]$ možno vyjadriť ako

$$\begin{aligned} \cos\beta &= \frac{x}{r}, \\ \sin\beta &= \frac{y}{r}. \end{aligned} \tag{5.3}$$



Obr. 5.2 Rotácia bodu A do bodu A´ podľa počiatku súradnicovej sústavy

Pre súradnice bodu X′[x', y'] otočeného okolo počiatku o uhol α platí

$$\begin{aligned} x' &= r \cdot \cos(\alpha + \beta) = r(\cos\alpha \cdot \cos\beta - \sin\alpha \cdot \sin\beta) \\ y' &= r \cdot \sin(\alpha + \beta) = r(\sin\alpha \cdot \cos\beta - \cos\alpha \cdot \sin\beta) \end{aligned} \tag{5.4}$$

Po úprave dostaneme opäť vektorový, resp. maticový zápis

$$\mathbf{X}' = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \mathbf{X}. \tag{5.5}$$

Obe vyššie popísané transformácie možno pomocou homogénnych súradníc zlúčiť do jednej, tzv. transformačnej matice **T**, ktorú možno zapísať ako

$$\mathbf{T} = \begin{bmatrix} \cos\alpha & -\sin\alpha & t_x \\ \sin\alpha & \cos\alpha & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_2^T & 1 \end{bmatrix}, \tag{5.6}$$

kde **R** je matica rotácie, **t** je vektor translácie a $\mathbf{0}_2^T = [0 \quad 0]$ je nulový riadkový vektor. Táto transformácia má tri stupne voľnosti, čo vyplýva z počtu nezávislých premenných v transformačnej rovnici. Informácie čerpané z [1].

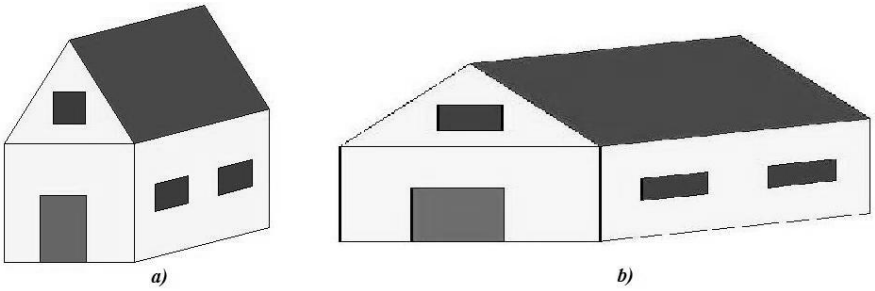
5.1.2 Afinné transformácie

Spoločným znakom týchto transformácií je, že rovnobežné priamky v pôvodnom obraze sú rovnobežné aj vo výslednom obraze, avšak rovnobežnosť medzi priamkami pôvodného a cieľového obrazu nemusí zostať zachovaná. Do tejto skupiny patria napr. škálovanie, skosenie a transformácie, ktoré vzniknú ich skladaním.

Škálovanie (zmena mierky) je veľmi často používaná transformácia, ktorá umožňuje zmenu veľkosti objektu. Všeobecný maticový tvar tohto zobrazenia je

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{5.7}$$

kde s_x a s_y sa nazývajú škálovacie faktory pre zmenu mierky v smere osi x, resp. y. Na ***Obr. 5.3*** je ukážka zmeny mierky, kde boli použité hodnoty $s_x = 2$ a $s_y = 0{,}8$.



Obr. 5.3 Ukážka zmeny mierky: a) pôvodný obraz, b) škálovaný obraz

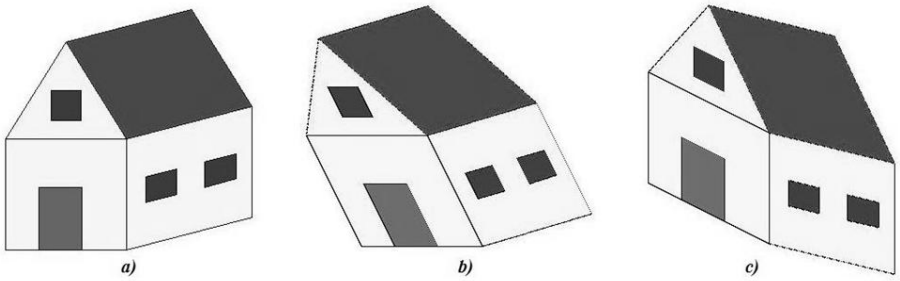
Skosenie je transformácia vyvolávajúca dojem, že sa objekty posúvajú po vrstvách. Rozoznávame 2 základné typy skosenia:

- v smere x (mení sa súradnica x, súradnica y ostáva nezmenená, viď ***Obr. 5.4*b**),
- v smere y (mení sa súradnica y, súradnica x ostáva nezmenená, viď ***Obr. 5.4*c**).

Všeobecný tvar transformácie skosenia má tvar

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & k_x & 0 \\ k_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{5.8}$$

kde k_x a k_y sú koeficienty miery skosenia pre jednotlivé osi.



Obr. 5.4 Skosenie obrazu: a) pôvodný obraz, b) skosenie v smere x, c) skosenie v smere y

Podobne ako u Euklidovských transformácii, aj v tomto prípade možno viaceré transformácie zlúčiť do jednej, čím sa vytvorí všeobecná afinná transformácia, ktorá má tvar matice

$$\mathbf{T} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}_2^T & 1 \end{bmatrix}. \tag{5.9}$$

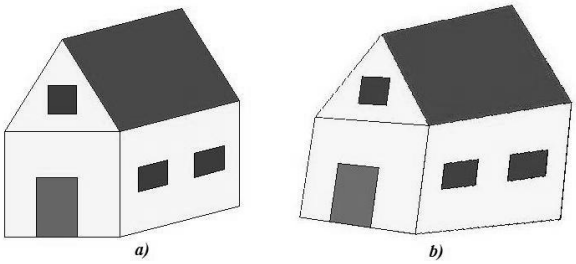
V porovnaní s maticou Euklidovských transformácií sa počet nezávisle premenných zvýšil, transformácia má teda 6 stupňov voľnosti. Čerpané z [1], [7].

5.1.3 Perspektívne (projektívne) transformácie

Táto skupina geometrických transformácií zovšeobecňuje afinnú transformáciu. Ide o nesingulárnu lineárnu transformáciu homogénnych súradníc, transformačná matica má tvar

$$\mathbf{T} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix}, \tag{5.10}$$

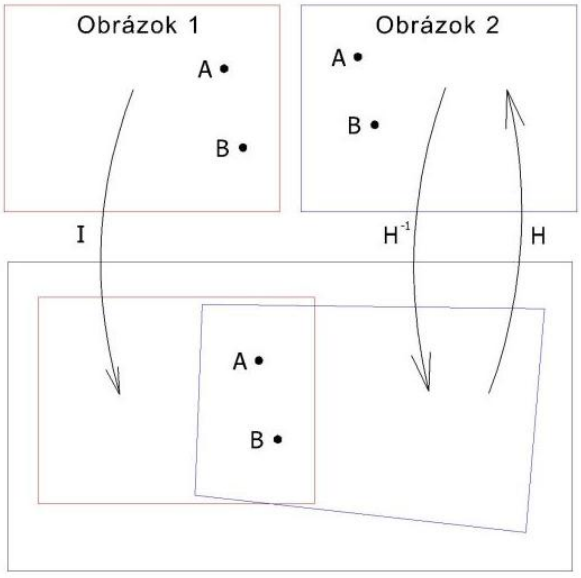
kde $\mathbf{v}^T = [v_1, v_2]$ a v je koeficient podobný ako pri homogénnej reprezentácii bodu. Matica má 9 prvkov, pričom podstatný je hlavne ich vzájomný pomer. Transformácia má teda 8 stupňov voľnosti. Príklad perspektívnej transformácie je na ***Obr. 5.5***, kde vidieť, že pri tejto transformácii rovnobežnosť priamok pôvodného obrazu vo výslednom obraze nemusí byť zachovaná. Informácie čerpané z [1].



Obr. 5.5 Perspektívna transformácia: a) pred transformáciou. b) po transformácii

5.2 Homografia

Ako bolo spomínané v úvode tejto kapitoly, panoramatický pohľad scény vznikne tak, že jednotlivé snímky premietneme do spoločného súradnicového systému. Túto situáciu znázorňuje ***Obr. 5.6***. Súradnice výsledného obrazu sú zhodné so súradnicami prvého obrazu, čo popisuje vzťah *I* (identita). Druhý obraz je transformovaný pomocou matice *H*, tzv. homografie, ktorá určuje korešpondencie medzi všetkými bodmi v oboch snímkach. Definícia homografie hovorí, že ide o spätné mapovanie bodov a priamok v projektívnej rovine. Ide teda o perspektívnu geometrickú transformáciu, popísanú v kapitole 5.1.3. Čerpané z [9].



Obr. 5.6 Vytvorenie panoramatického obrazu

5.2.1 Výpočet homografie

Homografiu možno určiť buď zo znalosti niekoľkých dvojíc korešpondujúcich bodov, alebo priamok v oboch obrazoch. V tejto práci bude riešený výpočet homografie z korešpondenčných bodov.

Pre každú dvojicu korešpondenčných bodov možno napísať

$$\alpha_i \boldsymbol{x'}_i = \mathbf{H} \cdot \boldsymbol{x}_i \,, \tag{5.11}$$

kde $i = 1, \dots, n$ označuje index dvojice bodov, $\boldsymbol{x}_i = [x_i, y_i, 1]^{\mathrm{T}}$ a $\boldsymbol{x'}_i = [x'_i, y'_i, 1]^{\mathrm{T}}$ sú homogénne súradnice *i*-tej korešpondencie, **H** je homografia a α_i je konštanta z $\mathbf{R}\backslash\{0\}$.

Ak $\boldsymbol{h}_1{}^{\mathrm{T}}$, $\boldsymbol{h}_2{}^{\mathrm{T}}$, $\boldsymbol{h}_3{}^{\mathrm{T}}$ sú riadky matice **H**, potom možno pre každú korešpondenciu s indexom *i* zostaviť sústavu 3 rovníc

$$\alpha_i x'_i = \boldsymbol{h}_1{}^{\mathrm{T}} \cdot \boldsymbol{x}_i \,, \tag{5.12}$$

$$\alpha_i y'_i = \boldsymbol{h}_2{}^{\mathrm{T}} \cdot \boldsymbol{x}_i \,, \tag{5.13}$$

$$\alpha_i = \boldsymbol{h}_3{}^{\mathrm{T}} \cdot \boldsymbol{x}_i \,. \tag{5.14}$$

Po dosadení vzťahu (5.14) do (5.12) a (5.13) vzniknú dve lineárne algebraické rovnice pre neznáme $\boldsymbol{h}_1, \boldsymbol{h}_1, \boldsymbol{h}_1$

$$\boldsymbol{h}_3{}^{\mathrm{T}} \cdot \boldsymbol{x}_i \cdot x'_i = \boldsymbol{h}_1{}^{\mathrm{T}} \cdot \boldsymbol{x}_i \,, \tag{5.15}$$

$$\boldsymbol{h}_3{}^{\mathrm{T}} \cdot \boldsymbol{x}_i \cdot y'_i = \boldsymbol{h}_2{}^{\mathrm{T}} \cdot \boldsymbol{x}_i \,. \tag{5.16}$$

Pre všetky korešpondencie bodov v obrazoch tak dostaneme systém 2*n* homogénnych lineárnych algebraických rovníc pre 9 neznámych, ktorý možno prepísať do maticového tvaru

$$\begin{bmatrix} -\boldsymbol{x}_i^{\mathrm{T}} & 0\,0\,0 & x'_i\,\boldsymbol{x}_i^{\mathrm{T}} \\ 0\,0\,0 & -\boldsymbol{x}_i^{\mathrm{T}} & y'_i\,\boldsymbol{x}_i^{\mathrm{T}} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix} = 0,\ i = 1, \dots, n \,, \tag{5.17}$$

ktorý možno symbolicky označiť

$$\mathbf{A}\,\boldsymbol{h} = 0 \,. \tag{5.18}$$

Pri použití homogénnych súradníc je homografia **H** jednoznačne určená, vďaka čomu má matica **A** hodnosť iba 8. Z toho vyplýva, že na určenie homografie musíme v matici **A** získať 8 lineárne nezávislých riadkov. Keďže každá dvojica korešpondenčných bodov je určená dvomi rovnicami, na výpočet stačia 4 korešpondenčné body ($n = 4$). Podmienkou je, aby tieto body neležali na priamkach. Aby však bol výpočet presnejší a odolnejší voči chybám, odporúča sa zvoliť týchto bodov viacero ($n \gg 4$). Informácie boli čerpané z [9].

5.3 Skladanie snímok do panorámy

Po nájdení matice homografie **H** možno realizovať na zdrojové snímky požadovanú transformáciu a následne snímky spojiť do panoramatického obrazu. Na ***Obr. 5.7*** sú zdrojové snímky. Pravá snímka bola podľa matice homografie transformovaná do súradnicového systému ľavej snímky. Následne sa snímky preložia (ľavá cez pravú), čím vznikne výsledný panoramatický obraz, ktorý je na ***Obr. 5.8***.

Obr. 5.7 Zdrojové snímky pre panorámu

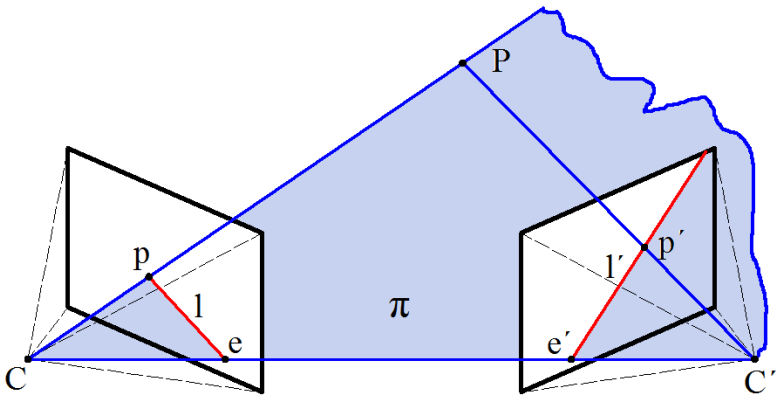
Obr. 5.8 Výsledná panoráma po transformácii a preložení snímok

6 Tvorba disparitnej mapy

Pojem disparita označuje rozdielnosť dvojice premietaných bodov (v zmysle ich pozície) pri snímaní pohľadu dvomi kamerami. Pomocou disparity možno zo scény snímanej dvomi kamerami extrahovať priestorové usporiadanie jednotlivých objektov. Pre určenie disparity je však nutné definovať bodové korešpondencie pixelov v oboch obrazoch. Preto bude v úvode kapitoly popísaný všeobecný princíp hľadania korešpondencií v obrazoch zachytených pomocou dvoch kamier.

6.1 Epipolárna geometria a jej geometrická podstata

Ak je bod P zobrazovaný v 3D priestore pomocou dvojice perspektívnych kamier, potom bod p odpovedá jeho obrazu v prvom pohľade, a bod p´ obrazu v druhom pohľade. Situáciu znázorňuje ***Obr. 6.1***. Stredy oboch kamier C a C´ ležia v jednej rovine s bodom P (sú teda koplanárne) a vytvárajú rovinu π. Obrazový bod p spoločne so stredom kamery C definuje v trojrozmernom priestore priamku l, ktorá je v obrazovej rovine druhej kamery zobrazená ako priamka l´. Ak teda obraz priestorového bodu p leží na priamke l, obraz toho istého bodu v druhom pohľade p´ musí ležať na priamke l´. Zo oboch známych premietaných bodov možno určiť na základe definovaného usporiadania kamier polohu priestorového bodu P. Výhodou epipolárnej geometrie teda je, že vyhľadávanie geometrických korešpondencií môže byť obmedzené len na priamku. Informácie čerpané z [1].



Obr. 6.1 Projekcia priestorového bodu P pomocou dvoch kamier

S epipolárnou geometriou sa spájajú tieto pojmy:

Kamerová báza – priamka prechádzajúca stredmi kamier.

Epipól – priesečník kamerovej báze s obrazovou rovinou.

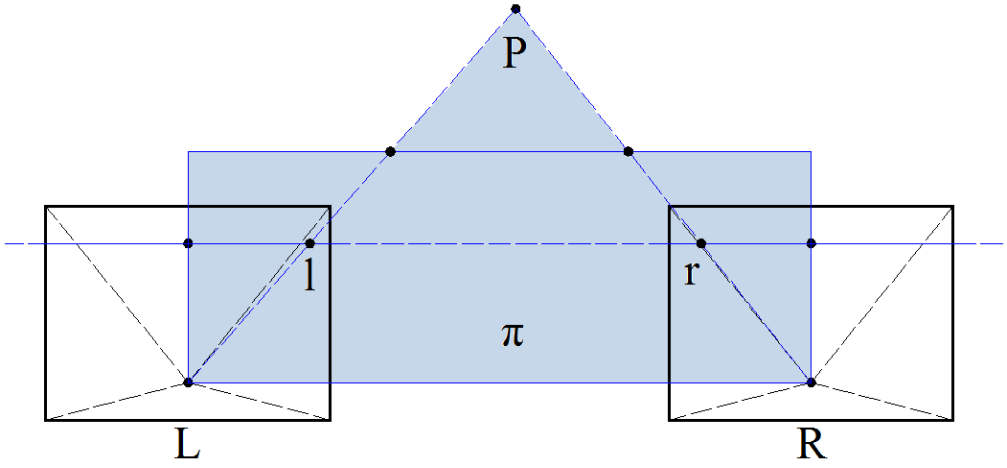
Epipolárna rovina – rovina obsahujúca bázu.

Epipolárna priamka – priesečník epipolárnej roviny s obrazovou rovinou.

Všetky popísané geometrické súvislosti v dvojici obrazov reprezentuje tzv. *fundamentálna matica*, ktorá je podrobne odvodená v literatúre [1].

6.2 Jednoduchá stereo geometria

Výrazné zjednodušenie čo sa týka hľadania korešpondencií v stereo páre obrazov možno dosiahnuť pri tzv. jednoduchej stereo geometrii, kedy sú osi oboch kamier stereo páru paralelné (rovnobežné), ako to znázorňuje ***Obr. 6.2***. V takomto prípade majú korešpondenčné body rovnaké súradnice v osi y, a teda epipolárnymi priamkami sú jednotlivé riadky. Mapovanie bodov z priestorovej scény do roviny obrazu možno potom uvažovať iba v jednom smere, teda pri hľadaní korešpondencií sa uvažuje len pohyb pixelov doľava alebo doprava. Čerpané z [1].



Obr. 6.2 Jednoduchá stereo geometria

Klasická disparitná rovnica, ktorá je podrobne odvodená v literatúre [1], určuje vzťah medzi projektívnymi korešpondenčnými bodmi pri ľubovoľnom usporiadaní kamier a má tvar

$$Z'\tilde{\mathbf{m}}' = Z\mathbf{A}'\mathbf{R}\mathbf{A}^{-1}\tilde{\mathbf{m}} + \mathbf{A}'\mathbf{t}\,, \tag{6.1}$$

kde $\tilde{\mathbf{m}}$ sú body v homogénnych súradniciach, Z je hĺbka pixelov, $\mathbf{A}$ je matica intrinsických parametrov kamery s rozmermi 3×3, matica $\mathbf{R}$ s rozmermi 3×3 určuje rotáciu kamery a vektor $\mathbf{t}$ určuje posunutie stredu perspektívneho premietania vzhľadom k počiatku súradnicového systému. Informácie čerpané z [1].

Keďže pri jednoduchej stereo geometrii má vektor $\mathbf{t}$ tvar $\mathbf{t} = (-t_x, 0{,}0)^{\mathrm{T}}$, matica rotácie $\mathbf{R}$ je jednotková (nemá vplyv na obraz) a matice intrinsických parametrov sú rovnaké $\mathbf{A} = \mathbf{A}'$, platí

$$\mathbf{A}'\mathbf{R}\mathbf{A}^{-1} = \mathbf{A}'\mathbf{A}^{-1} = \mathbf{I}\,, \tag{6.2}$$

kde $\mathbf{I}$ je jednotková matica. Hĺbková súradnica Z je pri tejto konfigurácii kamier pre oba body rovnaká, preto možno vzťah zjednodušiť na tvar

$$\tilde{\mathbf{m}}' = \tilde{\mathbf{m}} + \frac{1}{Z}\mathbf{A}'(-\mathrm{t_x}, 0{,}0)^{\mathrm{T}}, \text{resp.}\,(u_{\mathrm{R}}, v_{\mathrm{R}}, 1)^{\mathrm{T}} = (u_{\mathrm{L}}, v_{\mathrm{L}}, 1)^{\mathrm{T}} + \frac{1}{Z}(-t_x\alpha_u, 0{,}0)^{\mathrm{T}}\,, \tag{6.3}$$

kde t_x je kamerová báza a α_u je vnútorný priemer kamery. Premietané body sa budú líšiť iba v súradniciach osi x. Ak ako referenčný zvolíme ľavý obraz, potom platí vzťah

$$u_{\mathrm{R}} = u_{\mathrm{L}} - \frac{t_x\alpha_u}{Z}\,. \tag{6.4}$$

Ak označíme obrazové diskrétne funkcie ľavého a pravého obrazu ako $L(x_L, y_L)$ a $R(x_R, y_R)$ a ako referenčný bude opäť zvolený ľavý obraz, možno napísať

$$x_R = x_L - d(x_L, y), \tag{6.5}$$

pričom $x_L = x_R = y$ a $d(x_L, y)$ označuje dvojrozmerné pole diskrétnych hodnôt disparít, ktoré sa nazýva disparitná mapa. Vzťah platí pre väčšinu obrazových bodov, s výnimkou okludovaných, teda prekrytých bodov. Medzi disparitou a hĺbkovou súradnicou Z existuje vzťah

$$d \cdot \text{šírka } pixelu = \frac{t_x \alpha_u}{Z}. \tag{6.6}$$

Pri vzrastajúcej bodovej hĺbke teda hodnota disparity medzi premietanými bodmi klesá.

7 Riešenie algoritmov na spracovanie obrazu

Teoretický princíp jednotlivých metód spracovania obrazu bol vysvetlený v predchádzajúcich kapitolách. V tejto kapitole bude rozobraná praktická realizácia týchto metód v programovacom jazyku Java, resp. Matlab.

7.1 Algoritmy pre výpočet a ekvalizáciu histogramu

Táto problematika bola riešená programovacím jazykom Java vo vývojovom prostredí NetBeans 6.7.1. Program bol vytvorený ako aplikácia s využitím grafického užívateľského rozhrania Swing. Vývojový diagram programu je v prílohe ***A.2.*** Dokáže spracovať jednak šedotónový obraz, ako aj farebný obraz. Program pozostáva z nasledujúcich častí:

- predspracovanie obrazu
- výpočet a normalizácia histogramu
- ekvalizácia histogramu
- vykreslenie histogramu

7.1.1 Predspracovanie obrazu

Programovací jazyk Java rozoznáva dva obrazové typy: *Image* a *BufferedImage*. Pre spracovanie obrazu je výhodnejšie použiť *BufferedImage*, ktorý poskytuje viaceré operácie s obrazovými súbormi. Načítaný obraz teda najskôr konvertujeme na *BufferedImage*, následne metódou *BufferedImage.getRaster()* obraz prevedieme do numerických hodnôt pixelov. Keďže hodnoty pixelov majú dátový typ byte bez znamienka (*unsigned byte*) a Java bezznamienkovú aritmetiku nepodporuje je pre uľahčenie ďalšej práce je vhodné previesť hodnoty jednorozmerného poľa pixelových dát typu byte na trojrozmerné pole celočíselných hodnôt typu integer. Tento prevod je realizovaný prostredníctvom metódy *convertTo3D()*. Pole má tvar: *int [riadok][stĺpec][hĺbka]*, kde prvé dva rozmery zodpovedajú riadkom a stĺpcom pixelov v obraze. Tretí rozmer má vždy veľkosť rovnú 4 a obsahuje nasledujúce hodnoty podľa veľkosti indexu:

- 0 – alpha,
- 1 – red,
- 2 – green,
- 3 – blue.

Načítaný obraz sa zobrazuje do komponentu *JPanel*. K zobrazeniu sa využíva trieda *Paint*, ktorá je definovaná ako potomok triedy *JPanel*.

7.1.2 Výpočet a normalizácia histogramu

Zdrojový kód metódy na výpočet a normalizácia histogramu, ktorý je asi najdôležitejšou časťou tohto programu je zobrazený n**a *Obr. 7.1***. Bolo vytvorené jednorozmerné pole celočíselných hodnôt *hist*, do ktorého sa postupným prechodom všetkých pixelov ukladali vzorky jednotlivých RGB zložiek z celého rozsahu obrazu.

V ďalšom kroku sa z vytvoreného poľa položiek histogramu vybrala najväčšia hodnota, na základe ktorej bol potom histogram normalizovaný do požadovaných hodnôt tak, aby maximálna hodnota bola zobrazená na výšku grafu. Na konci metóda vracia jednorozmerné pole prvkov normalizovaného histogramu.

```
    //Metoda pre vypočet histogramu
public int[] calcHistogram(int[][][] imgPixels,int height,int width){
    int[] hist = new int[n];
    for (int i=0; i<width; i++) {
        for (int j=0; j<height; j++) {
        hist[imgPixels[i][j][1]]++;
        hist[imgPixels[i][j][2]]++;
        hist[imgPixels[i][j][3]]++;
  } //end for j
} //end for i
// hladanie najvacsej hodnoty histogramu
int max = 0;
    for(int cnt = 1;cnt < hist.length - 1;cnt++){
      if(hist[cnt] > max){
        max = hist[cnt];
      }//end if
    }//end for cyklus
// Normalizácia histogramu do hodnot 0 az height
 for(int cnt = 0;cnt < hist.length;cnt++){
    hist[cnt] = 240 * hist[cnt]/max;
    }//end for loop
 return hist;
}
```

Obr. 7.1 Výpis zdrojového kódu pre výpočet histogramu

7.1.3 Ekvalizácia histogramu

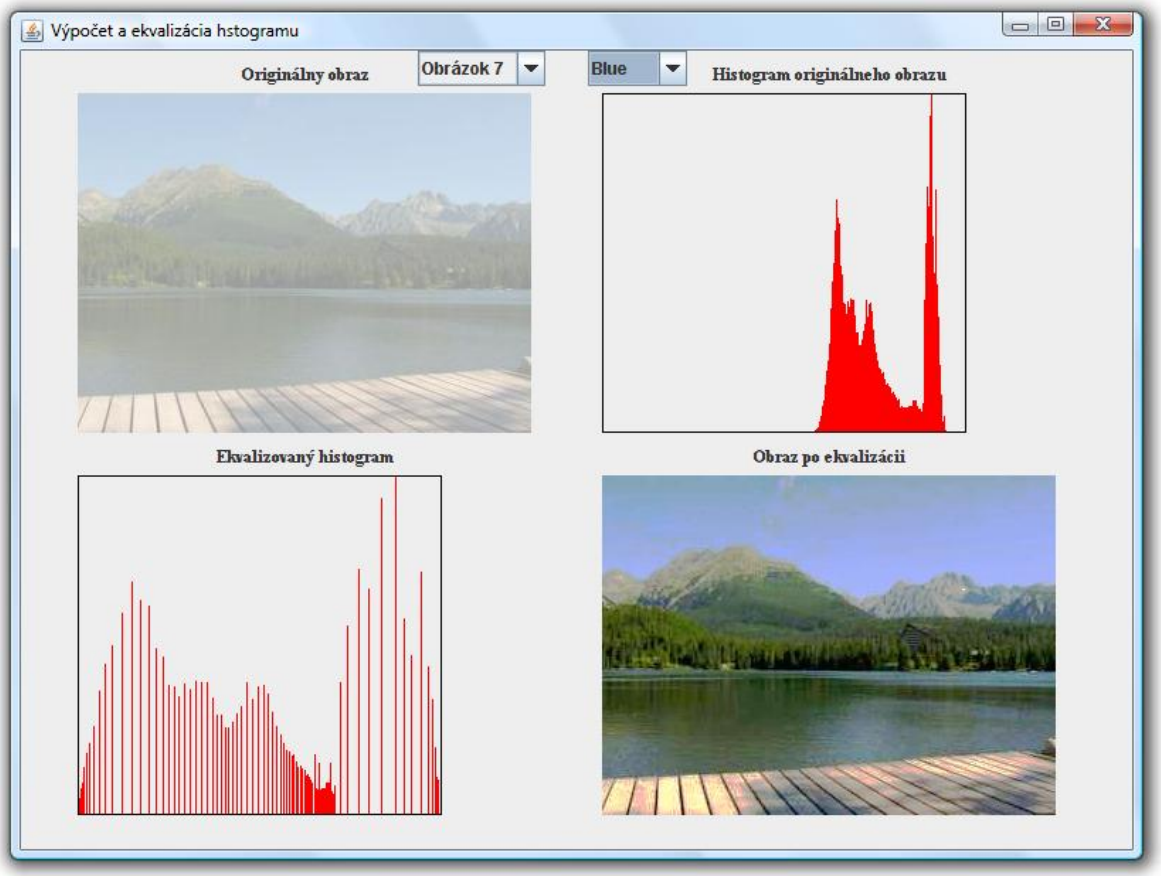
Program umožňuje zobraziť ekvalizovaný histogram pre každú farebnú zložku RGB spektra zvlášť, na základe výberu. Ekvalizácia histogramu vychádza z výpočtu histogramu pre jasové hodnoty (u šedotónového obrazu), resp. histogramu pre farebné zložky RGB (u farebného obrazu).

Aby nebolo treba nové jasové úrovne počítať opakovane pre každý pixel v obraze, je výhodné tieto jasové úrovne počítať len raz a následne ich uložiť do tzv. vyhľadávacej tabuľky LUT (angl. Look Up Table). Táto LUT tabuľka predstavuje jednorozmerné pole, ktoré obsahuje vypočítané transformované hodnoty jasu g_k pre hodnoty jasu $f_k = 0$ až 255. Po naplnení LUT tabuľky na základe histogramu sa vypočíta koeficient zmeny jednotlivých farebných zložiek obrazu, ktorým sa postupne násobia prvky poľa pixelových dát pôvodného obrazu. Výsledky sa ukladajú do trojrozmerného poľa *resultArray*, ktoré metóda vracia ako návratovú hodnotu. Takto získané pole obrazových dát sa ďalej použije ako vstupný parameter metódy *calcHistogram* (viď ***Obr. 7.1***), ktorá z tohto poľa vyráta ekvalizovaný histogram.

Pre získanie upraveného obrazu je potrebné konvertovať obrazové dáta späť do obrazového typu *Image*, resp. *BufferedImage*. Najskôr treba previesť dáta z trojrozmerného poľa do jednorozmerného, čo je realizované pomocou metódy *convertTo1D()*. Táto metóda je inverznou metódou voči metóde *convertTo3D()*, ktorá bola popísaná v kapitole 7.1.1. Napokon pomocou metódy *createImage()* vytvoríme z jednorozmerného poľa pixelových dát výsledný obraz *Image*, ktorý potom zobrazíme obdobne ako obraz pôvodný využitím triedy *Paint*.

7.1.4 Vykreslenie histogramu

Histogramy sú podobne ako obrázky zobrazované do komponentu *jPanel*. Na kreslenie histogramu bola vytvorená nová trieda s názvom *HistogramPanel*, ako potomok triedy JPanel. Metóda *paint()* tejto triedy kreslí pre každý prvok poľa histogramu zvislé čiary od 0 po veľkosť prvku. Postupným vykreslením čiar pre všetky indexy poľa histogramu dostaneme vo výsledku stĺpcový diagram – histogram. Na *Obr. 7.2* je ukážka výslednej aplikácie.



Obr. 7.2 Ukážka okna aplikácie na výpočet histogramu

Na obrázku je vidieť, že použitý obraz má zvýšený jas (jeho histogram je posunutý k pravému okraju). Pomocou ľavého roletového menu možno vybrať ďalšie farebné alebo šedotónové obrázky s inými deformáciami, ako napr. nízky jas, nízky kontrast, prípadne ich kombinácie. Pravým roletovým menu možno vybrať farebnú zložku RGB spektra pre výpočet.

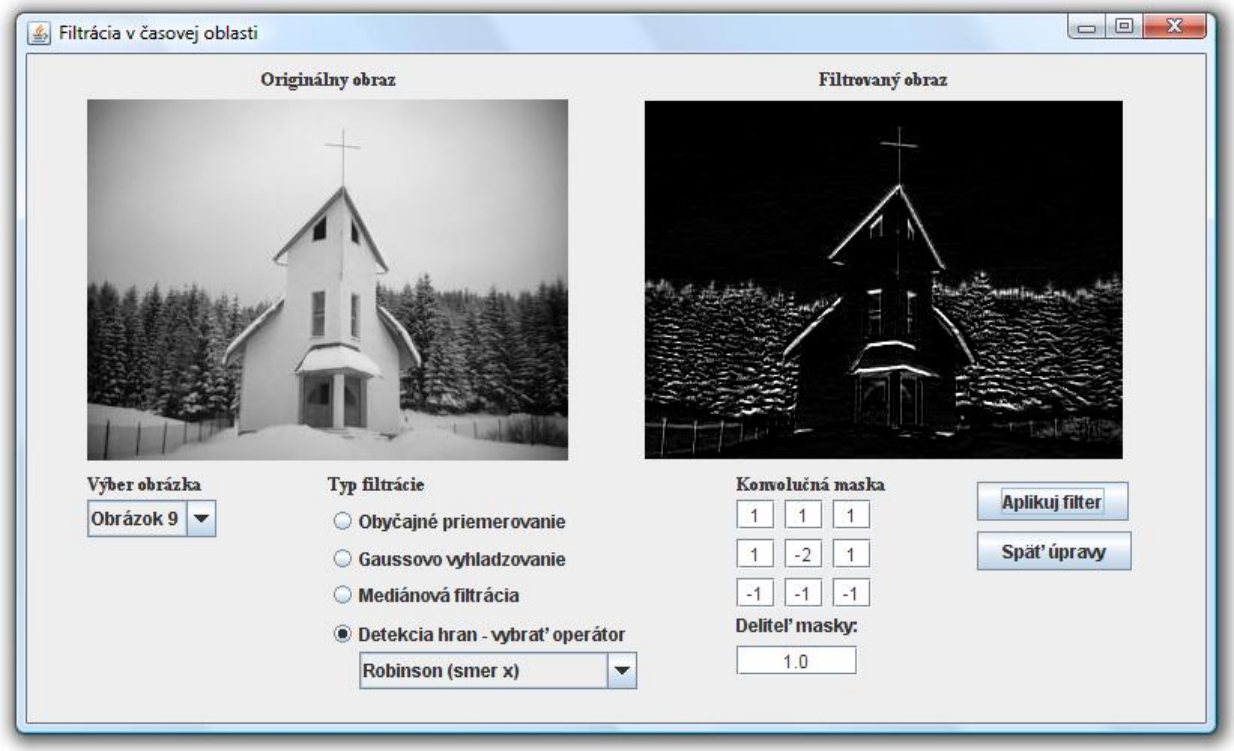
7.2 Algoritmy pre filtráciu v časovej a frekvenčnej oblasti

7.2.1 Filtrácia v časovej oblasti

Podobne ako program pre výpočet histogramu, aj program na realizáciu filtrácie v časovej oblasti je riešený vo vývojovom prostredí NetBeans 6.7.1 programovacieho jazyka Java. Program bol vytvorený opäť ako aplikácia s využitím grafického užívateľského rozhrania Swing a pozostáva z nasledujúcich častí:

- predspracovanie obrazu
- výber konkrétneho filtra
- aplikovanie filtru na zdrojový obraz

V rámci tohto programu boli otestované filtre popísané v kapitole 3, teda vyhladzovacie filtre obyčajné priemerovanie, Gaussovo vyhladzovanie, mediánový filter, a hranové detektory aplikujúce operátory Prewittovej, Sobelove, Robinsonove a Laplaceove. U gradientných operátorov prvého rádu je možnosť výberu dvoch smerov hranovej detekcie. Väčšina použitých filtrov (s výnimkou mediánovej filtrácie) je realizovaná pomocou podprogramu pre výpočet konvolúcie, ktorý na zdrojový obraz aplikuje konvolučnú masku konkrétneho operátora. Ukážka okna aplikácie je na ***Obr. 7.3***. V programe je možné konvolučné masky ľubovoľne meniť, je tu teda možnosť aplikovať aj iné operátory okrem predvolených. Jedinou nevýhodou tohto programu je, že dokáže pracovať len s maskami o rozmeroch 3 × 3. Vývojový diagram programu je v prílohe ***A.1***.



Obr. 7.3 Okno aplikácie pre časovú filtráciu obrazu

Podprogram pre výpočet konvolúcie

Pre aplikáciu jednotlivých operátorov na zdrojový obraz slúži trieda *ConvFilter*, ktorá tvorí základ programu. Na výpočet vlastnej konvolúcie slúži metóda *filteredImage()*, ktorej vstupnými parametrami sú zdrojový obraz a deliteľ masky, návratová hodnota je typu *ImageProducer*. V rámci predspracovania sa najskôr zistia rozmery zdrojového obrazu, následne sa extrahujú pixely zo zdrojového obrazu, tentoraz prostredníctvom metódy *PixelGrabber()*.

Spracovanie obrazu prebieha tak, že sa najskôr zálohuje pracovná kópia pixelov okolia 3 x 3, následne sa na toto okolie uplatnia váhové koeficienty konvolučnej masky pre všetky farebné zložky a tieto nové hodnoty sa uložia do jednorozmerného poľa pixelových dát. Takýmto spôsobom sa postupne prejde cez všetky body zdrojového obrazu a na konci získame pole pixelových dát filtrovaného obrazu. Prostredníctvom metódy *MemoryImageSource*() je toto pole prevedené na *ImageProducer*, z ktorého sa potom pomocou metódy *createImage()* vytvorí výsledný filtrovaný obraz.

7.2.2 Filtrácia vo frekvenčnej oblasti

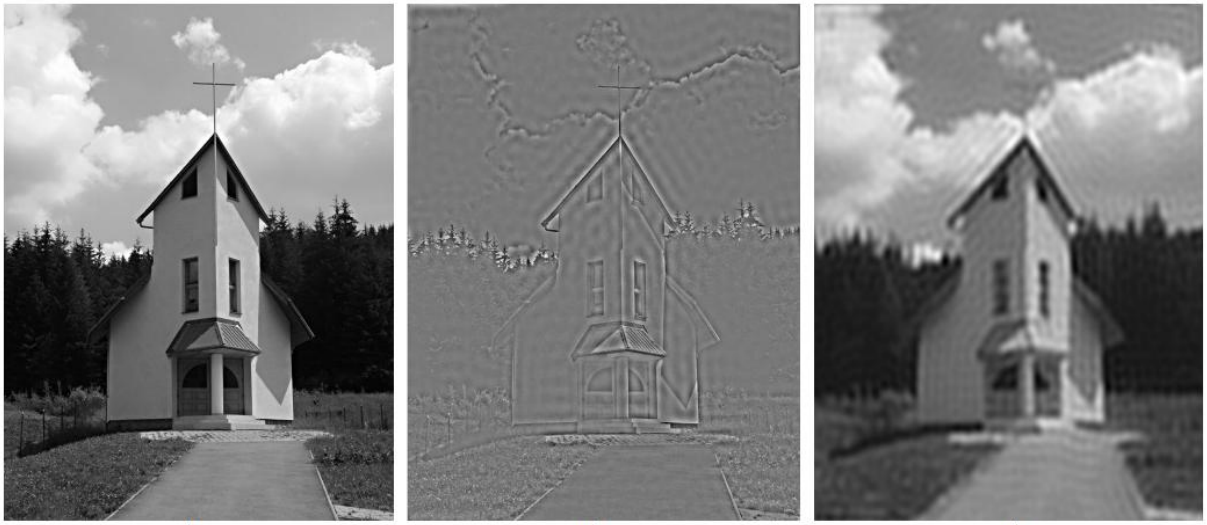
Táto problematika bola riešená pomocou programovacieho jazyka Matlab. Tento programovací jazyk je primárne určený na simuláciu matematických operácií, ale jeho zásadnou výhodou je podpora spracovania obrazových dát. Sú v ňom implementované napríklad funkcie pre výpočet Fourierovej transformácie a inverznej Fourierovej transformácie, ktoré sú nevyhnutné pre filtráciu vo frekvenčnej oblasti.

Boli vytvorené skripty pre filtre popisované v kapitole 4.1. Ukážka skriptu pre ideálny dolnopriepustný filter je na *Obr. 7.4*. Obdobne vyzerá aj skript pre horno-priepustný filter, líši sa len znamienkom v 14. riadku (H = double (D > P).

```
1   %IDEALNY DOLNOPRIEPUSTNY FILTER
2   function idealLPF(I,P)
3 - f=imread(I);                %nacitanie obrazka
4 - [M,N,Q]=size(f);            %rozmery obrazka
5 - F=fft2(double(f),M,N);      %transformacia do frekvencnej oblasti
6 - u=0:(M-1);                  %riadkový vektor
7 - v=0:(N-1);                  %stlpcovy vektor
8 - idx=find(u>M/2);
9 - u(idx)=u(idx)-M;
10 - idy=find(v>N/2);
11 - v(idy)=v(idy)-N;
12 - [V,U]=meshgrid(v,u);    % transform. vektorov v, u do poli V, U
13 - D=sqrt(U.^2+V.^2);      %vypocet vzdialenosti D od stredu filt. funkcie
14 - H=double(D<=P); %nastavenie filra
15 - Q=1;            %v pripade farebneho obrazu vyberie len 1 zlozku
16 - for i = 1:Q     %aplikovanie filtra na zdrojovy obraz
17 -    G(:,:,i) = H.*F(:,:,i);
18 - end
19 - g=(real(ifft2(double(G),M,N))); %transform. do casovej oblasti
20 - imshow(f), figure,imshow(g,[ ]);%zobrazenie oboch obrazkov
21 - end
```

Obr. 7.4 Skript pre ideálny DP filter

Problémom týchto filtrov bolo spracovanie farebných obrázkov, ktoré Matlab spracoval ako 3D matice s rozmerom [M]×[N]×[3], kde M, N sú rozmery obrazu a posledný rozmer predstavuje 3 zložky farebného modulu RGB. Z tohto dôvodu bolo treba z matice farebného obrázku vybrať len prvky odpovedajúce šedotónovému obrazu s rozmerom [M]×[N]. To bolo dosiahnuté v 15. riadku skriptu, kde bola všetkým farebným zložkám priradená rovnaká hodnota. Farebné zložky potom majú v každom pixelu obrázka rovnaké zastúpenie a z farebného obrazu sa stáva obraz šedotónový.



Obr. 7.5 Filtrácia vo frekvenčnej oblasti: a) originálny obraz, b) obraz po filtrácii HP filtrom, c) obraz po filtrácii DP filtrom

7.3 Algoritmy na vytváranie panoramatických obrázkov

Program bol vytvorený opäť v NetBeans ako aplikácia s využitím grafického užívateľského rozhrania Swing. Na spracovanie obrazu bola použitá knižnica OpenCV (**O**pen **S**ource **C**omputer **V**ision Library), voľne dostupná open source knižnica do C/C++ od firmy Intel. Je optimalizovaná a určená na real - time aplikácie. Na implementáciu knižnice OpenCV do jazyka Java bola použitá knižnica JavaCV (dostupná z http://code.google.com/p/javacv/), ktorá umožňuje použitie väčšiny funkcií OpenCV v Jave.

Vlastný program pozostáva z nasledujúcich častí:

- predspracovanie obrazu
- určenie korešpondenčných bodov
- výpočet matice homografie
- transformácia obrazu
- vykreslenie výsledného obrazu

7.3.1 Predspracovanie obrazu

V rámci predspracovania obrazu oproti postupu popísanom v kapitole 7.1.1 pribudol prevod obrazov typu *Buffered Image* do obrazového typu *IplImage* knižnice OpenCV. Najskôr je vytvorený prázdny obraz *IplImage* funkciou *cvCreateImage*(), do ktorého sú následne uložené dáta obrazu *Buffered Image* funkciou *IplImage.copyFrom*(). Pri vykresľovaní transformovaného obrazu je na spätný prevod *IplImage* do *Buffered Image* použitá metóda *IplImage.getBufferedImage*(*gamma*), kde sa hodnotou parametru *gamma* upravuje gamma korekcia obrazu, pretože pri tomto prevode vzniká určitá deformácia jasu pôvodného obrazu. Na prevod obrazu do numerických hodnôt pixelov je tento krát použitá metóda *BufferedImage.getRGB()*.

7.3.2 Určenie korešpondenčných bodov a výpočet homografie

Výber korešpondenčných bodov, teda dvojíc odpovedajúcich si bodov v oboch obrazoch je v aplikácii umožnený buď ručným zadávaním, alebo použitím predvolených bodov, ktoré sú definované napevno. Pri ručnom výbere sa po kliknutí pravého tlačidla myši na ľubovoľné miesto v obraze uloží poloha kurzoru myši do príslušného poľa. Body treba zadávať pomerne presne, aby pri následnej transformácii obrazu nedošlo k pádu programu. Po zadaní všetkých korešpondenčných bodov možno tlačidlom *Hotovo* potvrdiť výber a pristúpiť k výpočtu matice homografie. V prípade potvrdenia výberu v momente, keď nie sú zadané všetky korešpondenčné body, objaví sa dialóg s chybovým hlásením.

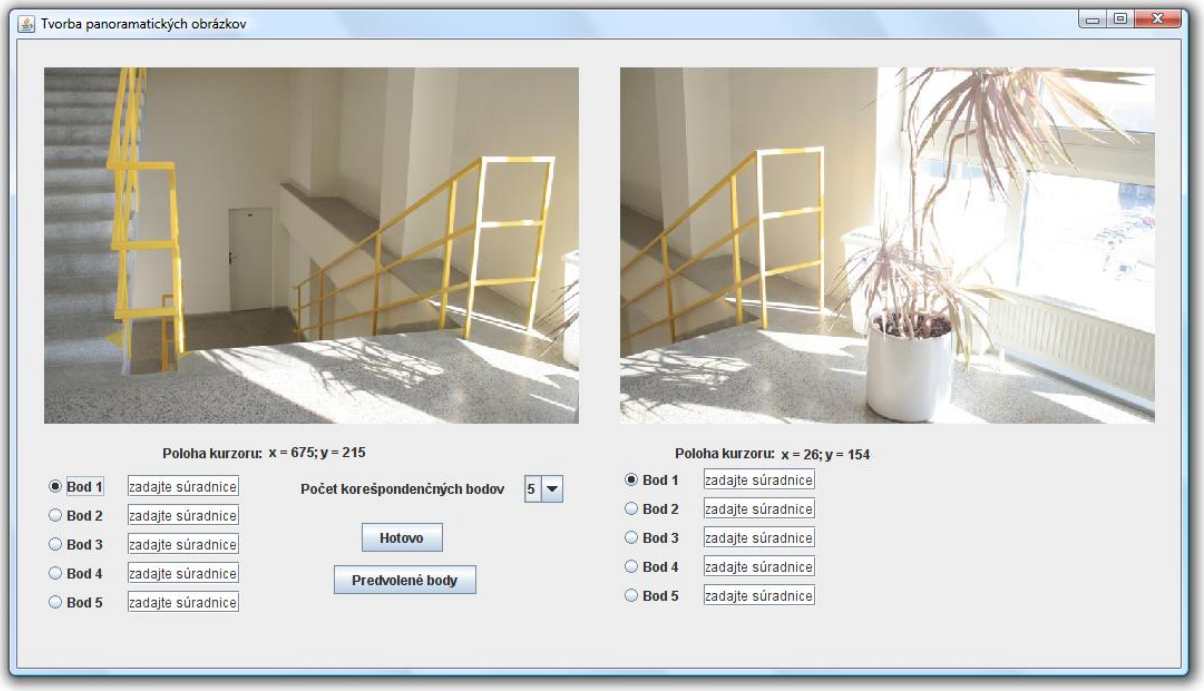
Pri výpočte homografie sú najskôr vytvorené prázdne matice (*matrix1*, *matrix2*) typu *CvMat* funkciou *CvMat.create*(*x*,*y*), kde *x* udáva počet korešpondenčných bodov, a *y* je nastavené na 2, keďže súradnice bodov v obraze majú dva rozmery. Funkciou *CvMat.put*() sú následne do týchto matíc vložené hodnoty korešpondenčných bodov. Ďalej je vytvorená prázdna matica *H* s rozmermi 3×3. Pri volaní funkcie *cvFindHomography*(*matrix2, matrix1, H*) sa z matíc korešpondenčných bodov vypočíta matica homografie a jej prvky sa uložia sa do vytvorenej matice *H*.

7.3.3 Transformácia obrazu

Pred vlastnou transformáciou je nutné vypočítať rozmery výsledného obrazu. Výpočet je realizovaný tak, že sa pomocou koeficientov získanej matice homografie zistia súradnice *x, y* štyroch rohových bodov obrazu. Najvyššia hodnota súradnice x je potom šírka výsledného obrazu a najvyššia hodnota súradnice y bude výška obrazu. Táto časť kódu bola s menšími úpravami prevzatá z riešenia v jazyku C++ poskytnutého vedúcim práce.

Po získaní výsledných rozmerov obrazu je vytvorený prázdny obraz *imageDST*. Funkciou *cvWarpPerspective*() je na pravý obraz uplatnená perspektívna transformácia. Pixely pôvodného obrazu sú teda mapované do výsledného obrazu *imageDST* podľa koeficientov matice homografie. Pozadie v tomto obraze je vyplnené hodnotou šedej v rozsahu 0 až 255, ktorú

určuje posledný parameter funkcie *cvWarpPerspective*(). Následne sú na tento transformovaný obraz „prilepené“ pixely ľavého obrazu, čím vznikne výsledný obraz, ktorý sa zobrazí v novom okne. Ukážka okna aplikácie je na ***Obr. 7.6*** a výsledný obraz je na ***Obr. 7.7***. Vývojový diagram celého programu je v prílohe ***A.4***.

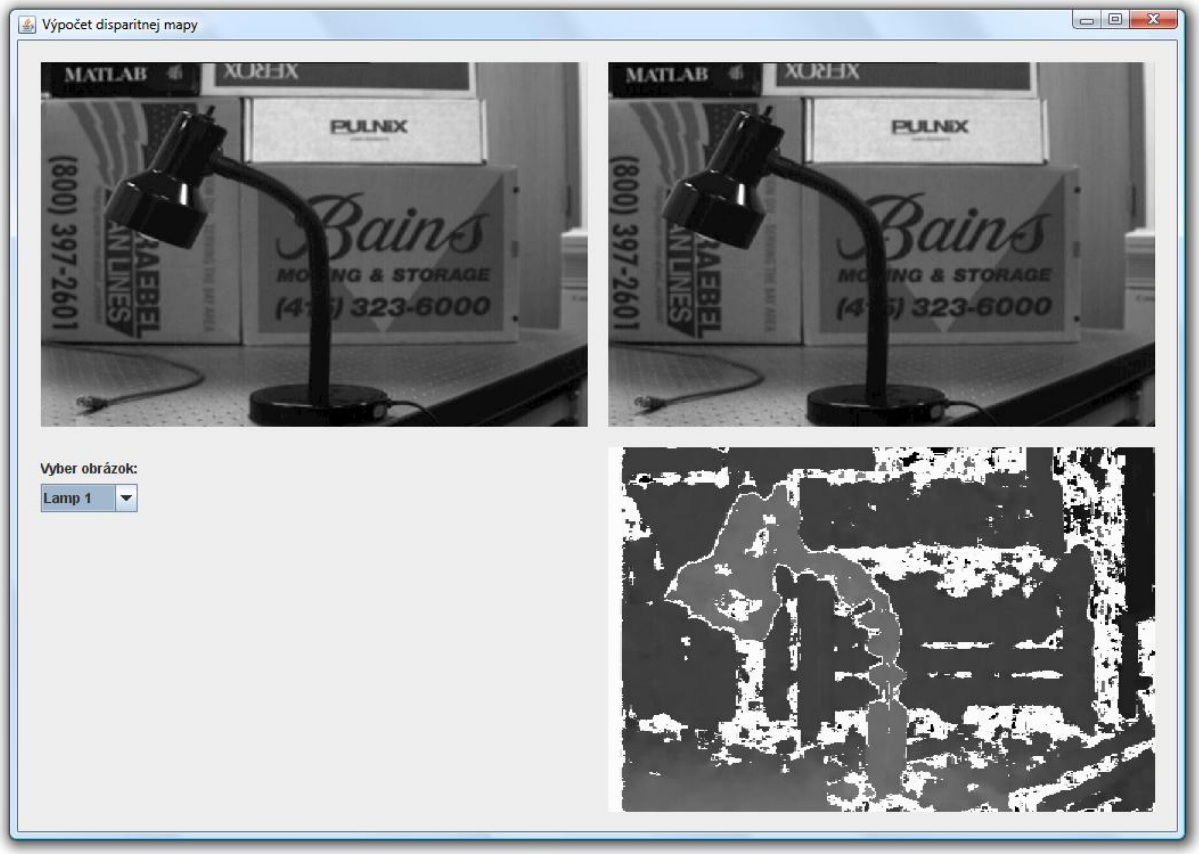


Obr. 7.6 Ukážka okna aplikácie na tvorbu panoramatických obrázkov

Obr. 7.7 Ukážka výsledného zloženého obrazu

7.4 Algoritmy na tvorbu disparitnej mapy

Aplikácia bola opäť vytvorená v NetBeans s využitím funkcií knižnice OpenCV a jej vývojový diagram je v prílohe ***A.3***. Predspracovanie obrazu prebieha rovnako ako v predošlej aplikácii. Jediným rozdielom je to, že pred výpočtom disparitnej mapy treba načítané obrázky najskôr skonvertovať na šedotónové. Toho sa dosiahne už pri načítavaní obrázkov, volaním funkcie *cvLoadImage*(*imgfile*, 1), kde *imgfile* určuje cestu k obrázku a druhý vstupný parameter 1 značí, že bude načítaný obraz s jedným kanálom. Na výpočet samotnej disparitnej mapy slúži funkcia *cvFindStereoCorrespondenceBM*(). Funkcia využíva tzv. Block Matching algoritmus, ktorý bude podrobnejšie popísaný v ďalšej kapitole. Okrem toho knižnica OpenCV poskytuje na výber ešte viacero algoritmov, avšak pri ich implementácii do jazyka Java dochádzalo k chybám a následne k pádu programu. Výsledný obraz (viď ukážka aplikácie na ***Obr. 7.8***) však čo sa kvality týka ďaleko zaostáva za očakávaniami a sú na ňom viditeľné rôzne chyby (biele miesta). Princíp je však z obrázku zrejmý. Lampa v popredí má svetlejší odtieň šedej ako škatule v pozadí a oblasť za otvorenými dverami je takmer čierna. Použité stereo obrázky boli prevzaté z [10]. Pri použití vlastných snímok bol výsledný efekt ešte horší a na výslednom obraze boli pozorovateľné sotva obrysy predmetov, hoci boli nasnímané pomocou zrkadlovky CANON EOS 40D so statívom a panoramatickou hlavicou, a teda posun medzi riadkami by mal byť eliminovaný.



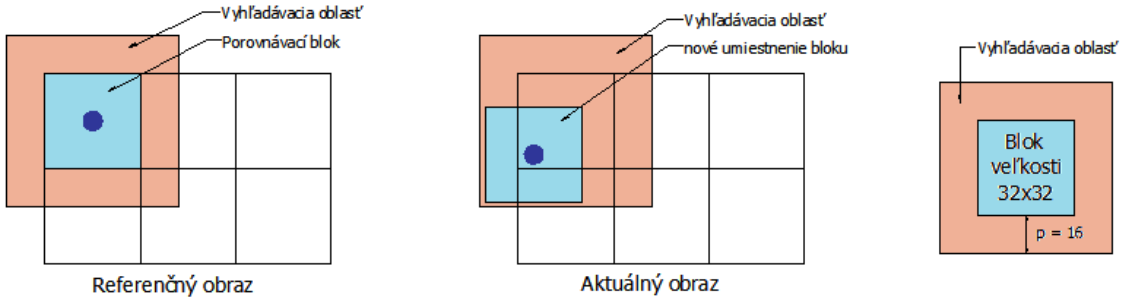
Obr. 7.8 Ukážka okna aplikácie na výpočet disparitnej mapy

7.4.1 Algoritmus priradzovania blokov (Block Matching Algorithm)

Tento algoritmus možno rozdeliť do troch hlavných krokov:

- určenie veľkosti blokov,
- výber vhodného kritéria,
- výber vyhľadávacej metódy.

Algoritmus spočíva v tom, že sa aktuálny obraz rozdelí na bloky s rovnakou veľkosťou a hľadá sa blok, ktorý sa najviac zhoduje s blokom v referenčnom obraze. Pohyb bloku po obraze je však pri vyhľadávaní obmedzený len na určitú oblasť, viď ***Obr. 7.9***. Cieľom je teda u oboch obrazov nájsť najviac podobné bloky a vypočítať medzi nimi pohybové vektory. Algoritmus počíta iba s transláciou hľadaných blokov. Parameter maximálneho posunutia *p* vymedzuje oblasť vyhľadávania, a všeobecne platí, že čím je jeho hodnota vyššia, tým je algoritmus výpočtovo náročnejší.



Obr. 7.9 Porovnávanie blokov v oboch obrazoch

Vyhľadávacím kritériom zhodnosti blokov je tzv. nákladová funkcia. Najznámejšími nákladovými funkciami algoritmu sú *priemerná absolútna chyba* MAD (Mean Absolute Difference) a stredná kvadratická odchýlka MSE (Mean square error), ktoré sú určené vzťahmi

$$\mathrm{MAD}(d_1, d_2) = \frac{1}{N^2} \sum_{i=1}^{N-1} \sum_{j=1}^{N-1} |A(i,j,N) - R(i+d_1, j+d_2, N)|,$$

$$\mathrm{MSE}(d_1, d_2) = \frac{1}{N^2} \sum_{i=1}^{N-1} \sum_{j=1}^{N-1} [A(i,j,N) - R(i+d_1, j+d_2, N)]^2,$$

kde N je veľkosť strany štvorcového bloku, d_1, d_2 určujú hodnoty posunu pixelu bloku, $A(i,j,N)$ a $R(i+d_1, j+d_2, N)$ sú porovnávané pixely v aktuálnom, resp. referenčnom bloku so začiatočnými súradnicami (i,j) a veľkosťou bloku *N*. Informácie boli čerpané z [8].

Medzi metódy vyhľadávania patrí napr. úplné prehľadávanie, 2D logaritmické prehľadávanie, ortogonálne prehľadávanie, špirálové prehľadávanie a iné. Implementačne najjednoduchšia ale zároveň časovo najviac náročná je metóda úplného prehľadávania (angl. *Full Search)*, ktorá počíta nákladovú funkciu na všetkých možných pozíciách bloku. Po porovnaní všetkých hodnôt je k referenčnému bloku priradený blok s najmenšou chybou. Čerpané z [8].

8 Prevod aplikácii na aplety

Aplikácie sú samostatné programy, ktoré zahŕňajú aspoň jednu triedu s metódou *main*(), ktorá sa spúšťa automaticky po štarte programu. Aplety sa od aplikácii odlišujú predovšetkým v tom, že neobsahujú metódu *main*(), namiesto toho sú v rôznych fázach činnosti apletu volané viaceré rôzne metódy. Pri prevode aplikácie na aplet treba dodržať tieto kroky:

- Musí byť vytvorená podtrieda triedy java.applet.*Applet*, v ktorej bude metóda *init*() prepísaná tak, aby inicializovala všetky komponenty apletu, podobne ako metóda *main*() u aplikácie.
- V metóde *init*() treba do apletu pridať komponent *Panel* na najvyššej úrovni, podobne ako sa zvyčajne pridáva aj do rámca (*Frame*) v metóde *main*().

8.1 Implementácia apletov

8.1.1 Nahrávanie obrázkov do apletov

Keďže všetky aplikácie vytvorené v tejto práci využívali grafické užívateľské rozhranie *Swing*, hlavné triedy apletov boli vytvorené ako podtriedy triedy *javax.swing.JApplet*. Od klasických apletov triedy *Applet* sa swing aplety líšia v spôsobe práce s obrázkami. Zatiaľ čo trieda Applet na načítanie obrázkov používa metódu *getImage*(), ktorá vráti objekt typu Image, swing aplety vytvárajú inštancie triedy *ImageIcon* – ikony načítané z obrazových súborov. Výhodou použitia *ImageIcon* je úspora kódu, pretože dokáže automaticky nahrávať obrázky.

8.1.2 Načítanie rozširujúcich natívnych knižníc v apletoch

Niektoré aplety v tejto práci pri svojej činnosti využívajú rozširujúce knižnice, obsahujúce natívny kód. Štandardne aplety z dôvodu bezpečnosti nepovoľujú načítavanie žiadnych knižníc ani definovanie natívnych metód. Nasadenia takýchto natívnych knižníc v apletoch umožňuje *JNLPAppletLauncher* (**J**ava **N**ework **L**aunching **P**rotocol). *JNLPAppletLauncher* používa súbor s príponou *.jlnp*, v ktorom musí byť definovaná cesta k *.jar* súborom obsahujúcim požadované triedy. Súbor *applet-launcher.jar* obsahujúci triedu *JNLPAppletLauncher* musí byť podpísaný rovnakým certifikátom ako rozširujúce natívne zdroje. Užívateľ bude pri spúšťaní apletu vyzvaný k prijatiu certifikátu pre *JNLPAppletLauncher*. Jar súbor samotného apletu treba taktiež digitálne podpísať. Aby aplet správne fungoval, musia byť splnené tieto požiadavky:

- Parameter *archive* v značke `<applet>` html súboru musí obsahovať položky *applet-laucher.jar*, jar súbory knižníc a všetky jar súbory súvisiace s vlastným apletom.
- Názov hlavnej triedy apletu a textový popis musia byť špecifikované v parametroch značky `<applet>` *subapplet.classname* a *subapplet.displayname*.

- URL odkazy na súbory *.jlnp* k rozširujúcim knižniciam musia byť definované ako parametre. Parameter *jnlpNumExtensions* určuje počet *.jnlp* súborov a ich URL sú špecifikované parametrom *jnlpExtension1* až *jnlpExtension[n]*.

Informácie boli čerpané z [11].

8.2 Zavádzanie apletov

Postup zavádzania apletu závisí predovšetkým na type používaného prehliadača. Pri použití apletu:

- výhradne v aplikácii Internet Explorer sa používa značka `<object>`.
- výhradne pre prehliadače zo skupiny Mozilla sa používa značka `<embed>`
- v prostredí s rôznymi prehliadačmi použitie značky `<applet>`

Príklad zdrojového html kódu značky `<applet>`:

```
<applet code="HistogramApplet.class"
        archive ="Applets/HistogramApplet2.jar"
        width=800
        height=600>
      <param name="BackgroundColor" value="CCFFCC">
</applet>
```

Pomocou atribútu `archive` je definovaná cesta k jednému alebo viacerým *jar* súborom apletu. Cesta začína domovským adresárom www stránky. Voliteľným atribútom `code` môžeme prehliadaču určiť hlavnú triedu apletu. Atribútmi `width` a `height` nastavujeme rozmery okna, do ktorého sa aplet má zobraziť. Okrem toho možno použitím rôznych doplňujúcich parametrov nastaviť rozličné vlastnosti, ako napr. zmenu farby pozadia apletu parametrom `BackgroundColor`.

V domovskom adresári webovej stránky by mali byť uložené všetky súbory, s ktorými aplet pracuje, ideálne je tieto súbory zabaliť do spolu s vlastným apletom do jedného *jar* súboru. Pre testovanie správnej funkčnosti vytvorených apletov bola vytvorená webová stránka http://www.image-processing.tym.cz.

9 Záver

Zadaním tejto bakalárskej práce bolo vytvorenie Java apletov na problematiku spracovania obrazu. Konkrétne išlo o aplety na tvorbu a ekvalizáciu histogramu obrazu, filtráciu obrazu, vytváranie panoramatických obrázkov a vytváranie disparitnej mapy.

V teoretickej časti práce boli rozobraté konkrétne metódy spracovania obrazových dát týkajúce sa zadania práce. Na začiatku boli vysvetlené niektoré základné pojmy týkajúce sa spracovania obrazu. V ďalších kapitolách bola postupne prebratá celá problematika zadania. V obsahovo najrozsiahlejšej kapitole týkajúcej sa obrazovej filtrácie bolo uvedené rozdelenie filtrov podľa princípu ich činností do dvoch kategórií (horno-priepustné a dolno-priepustné) a ku každej kategórii bolo uvedených zopár príkladov. Jadrom piatej kapitoly bolo matematické odvodenie homografie a ozrejmenie jej významu pri vytváraní panoramatického obrazu. Šiesta kapitola ponúka ľahký úvod do základov epipolárnej geometrie, zameriava sa na snímanie obrazu dvomi kamerami pri jednoduchej stereo geometrii a odvodenie klasickej disparitnej rovnice pre tento konkrétny prípad.

V praktickej časti boli na odsimulovanie týchto metód najskôr vytvorené aplikácie v programovacom jazyku Java alebo skripty v Matlabe. Predovšetkým kvôli ladeniu programu bolo výhodnejšie najskôr vytvoriť aplikácie, ktoré boli napokon pretvorené do podoby apletov. Na vývoj aplikácii som využíval vývojové prostredie Netbeans 6.7.1. V rámci siedmej kapitoly boli postupne popísané všetky vytvorené aplikácie, boli spomenuté ich výhody a nevýhody, taktiež boli znázornené ukážky výstupu aplikácii. Na vytvorenie zložitejších algoritmov bola využitá knižnica OpenCV, ktorá obsahuje metódy na spracovanie obrazu. Keďže ide o knižnicu v jazyku C, bolo potrebné túto knižnicu implementovať do jazyka Java. Knižnica bola použitá predovšetkým z dôvodu uľahčenia zložitých výpočtov a šetrenia kódu. Umožňuje napr. použitím jedinej funkcie vypočítať maticu homografie (viď kap. 7.3.2) alebo vytvoriť zo zdrojových stereo obrázkov obraz hĺbkový (viď kap. 7.4). Celkovo boli v práci vytvorené štyri samostatné aplety, ktorých funkčnosť bola otestovaná na webovej stránke http://www.image-processing.tym.cz.

Aplet na výpočet histogramu umožňuje zobraziť histogram jedného z deviatich predvolených obrázkov, ktoré sú zámerne rôznym spôsobom deformované (nízky kontrast, jas atď.). Aplet automaticky ekvalizuje histogram vybraného obrazu a zobrazí ho spolu s obrazom po ekvalizácii. Ponúka tiež možnosť výberu jednotlivých RGB kanálov buď samostatne, alebo všetky dohromady.

Aplet na filtráciu obrazu ponúka takisto výber deviatich predvolených obrázkov a širokú škálu filtrov: horno-priepustné filtre na detekciu hrán a dolno-priepustné vyhladzovacie filtre. Veľkou výhodou je možnosť ľubovoľne editovať konvolučné masky, vďaka čomu môže užívateľ zadávať okrem predvolených filtrov použiť aj iné (avšak len s maskou 3×3).

Ďalšie dva aplety majú nevýhodu v tom, že využívajú vyššie spomínanú knižnicu OpenCV a pre správnu činnosť vyžadujú, aby na hostiteľskom počítači bola táto knižnica nainštalovaná. Aplet na tvorbu panorámy funguje tým spôsobom, že po zadaní bodových korešpondencií a kliknutí na tlačidlo *Hotovo* by mal do nového okna vykresliť výsledný zložený obraz. Ak sa tak nestane, došlo s najväčšou pravdepodobnosťou k nepresnému zadaniu korešpondencií a následnej chybe pri výpočte. Preto sa treba po obnovení stránky pokúsiť zadať body nanovo. Občas sa stáva, že sa pri transformácii zníži gamma korekcia v obraze, chyba je zrejme na strane tvorcu knižnice *JavaCV*. Aplet na tvorbu disparitnej mapy vypočíta s použitím funkcií knižnice OpenCV z dvoch zdrojových stereo obrazov hĺbkový obraz. Ako som už v kapitole 7.4 spomínal, obraz nedosahuje očakávanú kvalitu, ale podstatu problematiky z neho možno vyčítať.

Význam tejto práce spočíva v tom, že vytvorené aplety môžu byť použité ako interaktívna pomôcka pri laboratórnej výučbe predmetu „Pokročilé techniky zpracování obrazu“. Vytvorenie tejto práce bolo zároveň aj pre mňa obrovským prínosom. Získal som komplexnejší prehľad v niektorých oblastiach spracovania obrazu a nadobudol som nové skúsenosti čo sa týka programovania v jazyku Java a tvorby apletov. Verím, že nadobudnuté poznatky využijem aj v budúcnosti, možno aj pri diplomovej práci.

Zoznam použitej literatúry

[1] ŘÍHA, K.: *Pokročilé techniky zpracování obrazu*. Elektronické texty VUT, Ústav telekomunikací FEKT VUT v Brně, 2007.

[2] ŠMIRG, O.: *Zpracování obrazu za účelem řízení Visikonu*, Bakalářská práce, Brno: VUT, Fakulta elektrotechniky a komunikačních technologií, 2006,50, pp. 3

[3] DOBEŠ, M.: *Zpracování obrazu a algoritmy v C#,* BEN – technická literatura, Praha 2008 ISBN 978-80-7300-233-6

[4] ULLMAN, V.: *Filtry a filtrace v nukleární medicíně*, [online] Dostupné z WWW: <http://astronuklfyzika.cz/Filtry.htm>

[5] LINKA, A., VOLF P., KOŠEK M.: *Zpracování obrazu a jeho statistická analýza*, [online] Dostupné z WWW: <http://e-learning.tul.cz/cgibin/elearning/elearning.fcgi?IDtema=67&stranka=publ_tema>

[6] BLÁZSOVITS, G.: *Interaktívna učebnica spracovania obrazu*, Knižničné a edičné centrum FMFI UK, Bratislava, [online] Dostupné z WWW: <http://dip.sccg.sk>

[7] RUŽICKÝ, E., FERKO, A.: *Počítačová grafika a spracovanie obrazu,* Bratislava : Sapientia, 1995. 324 s.

[8] PAUCO, M. *Kryogelové fantómy pre ultrasonografickú elastografiu*. Praha, 2008. 96 s. Diplomová práce. ČVUT v Praze.

[9] KŘÍŽEK, Pavel. *Počítačové vidění pro informatiku* [online]. 2003-06-08 [cit. 2010-05-20]. PVI - Panorama. Dostupné z WWW: <http://cmp.felk.cvut.cz/cmp/courses/pvi2003/Projects/Uloha1/krizekp1/>.

[10] *Depth Discontinuities by Pixel-to-Pixel Stereo* [online]. 1999 [cit. 2010-05-20]. Dostupné z WWW: <http://www.ces.clemson.edu/~stb/research/stereo_p2p/>.

[11] *Applet-launcher: Project Home Page* [online]. 2007 [cit. 2010-05-20]. Dostupné z WWW: <https://applet-launcher.dev.java.net/>.

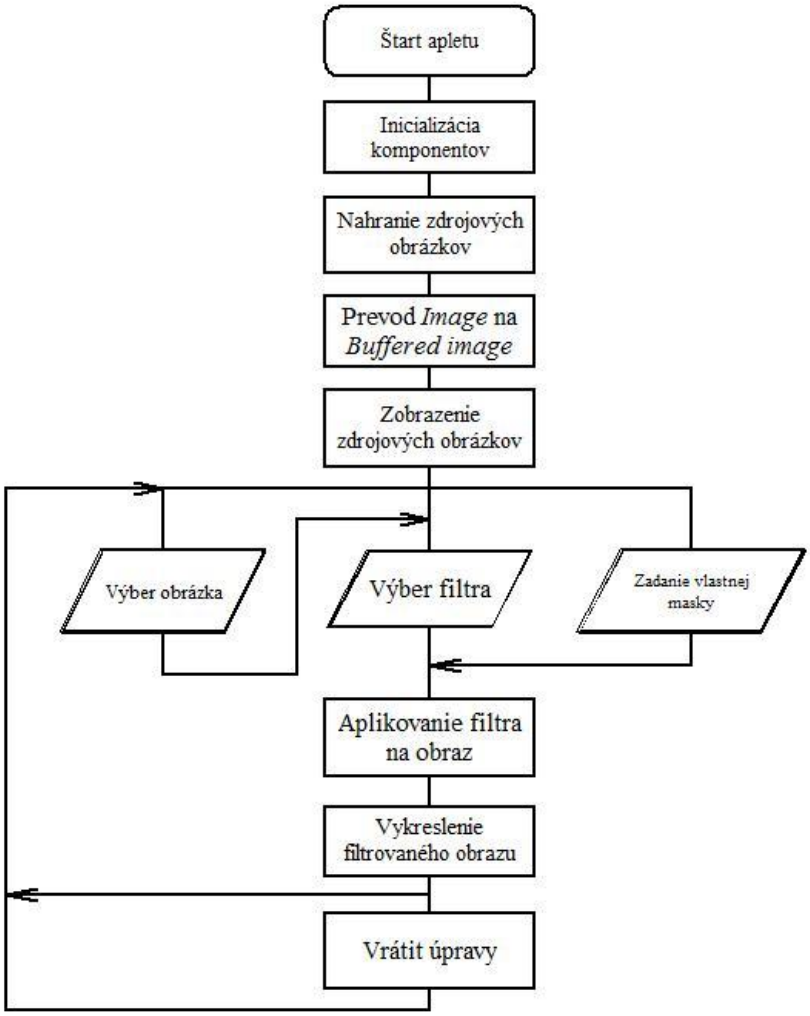
[12] SEMECKÝ, J.: *Naučte se Javu*, [online] Dostupné z: < http://interval.cz/clanky/author/jiri-semecky>

[13] *OpenCV 2.0 C Reference Manual*, [online] Dostupné z: <http://opencv.willowgarage.com/documentation/image_processing.html>

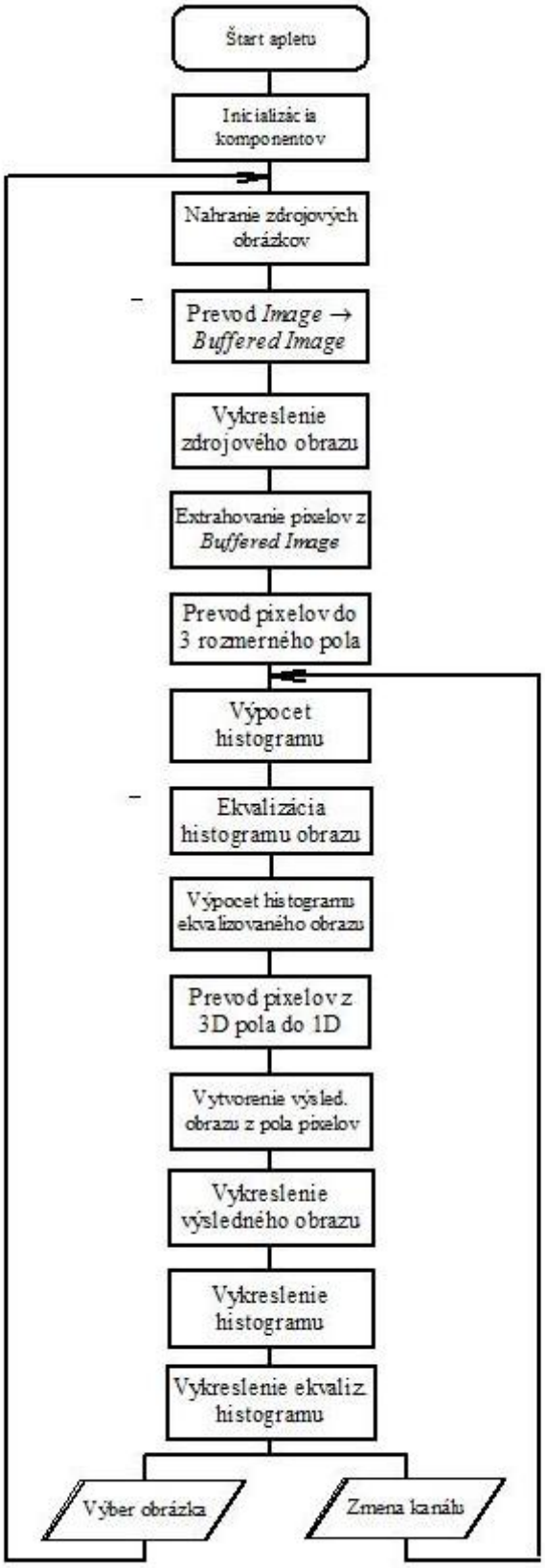
[14] BRUYLANDT, B.: *Java Applet Tutorial*, [online] Dostupné z: <http://www.realapplets.com/tutorial>

A. Prvá príloha

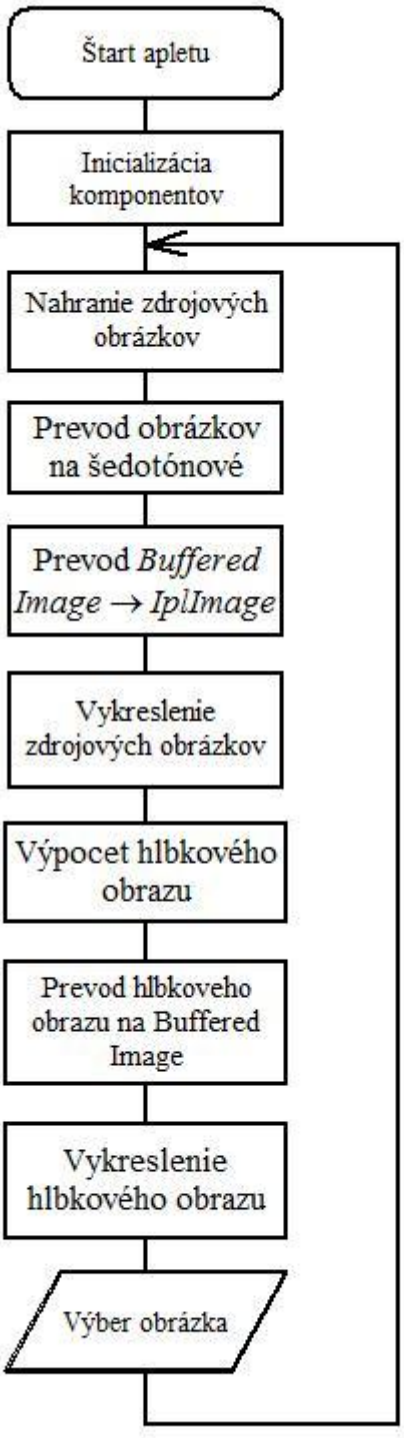
A.1. Vývojový diagram pre program na filtráciu obrazu



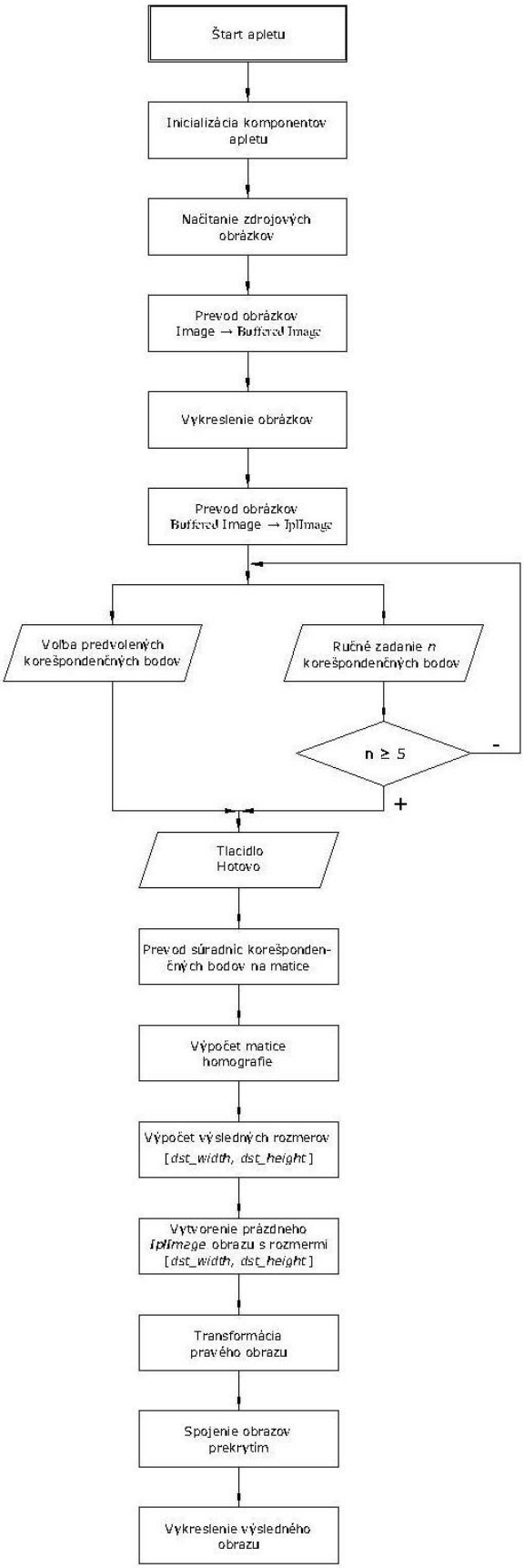
A.2. Vývojový diagram pre program na výpočet histogramu obrazu



A.3. Vývojový diagram pre program na vytváranie disparitnej mapy



A.4. Vývojový diagram pre program na tvorbu panoramatických obrazov



B. Druhá príloha

Obsah priloženého CD:

- Elektronická verzia bakalárskej práce
- Screenshoty s ukážkami výsledkov
- Kompletná webová stránka pre testovanie apletov v prostredí *Offline*
- zdrojové súbory k projektom vytvoreným v Netbeans 6.7.1