VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

NUMERICKÁ METODA FILTRACE ADITIVNÍHO ŠUMU V OBRAZE POMOCÍ ADAPTIVNÍHO FILTRU

NUMERICAL METHOD OF ADDITIVE NOISE FILTRATION BY MEANS OF ADAPTIVE FILTER IN IMAGE PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB VENCLOVSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL ŠTARHA, Ph.D.

BRNO 2014

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav matematiky
Akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Jakub Venclovský

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Matematické inženýrství (3901T021)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Numerická metoda filtrace aditivního šumu v obraze pomocí adaptivního filtru

v anglickém jazyce:

Numerical Method of Additive Noise Filtration by Means of Adaptive Filter in Image Processing

Stručná charakteristika problematiky úkolu:

Při pořizování snímků pomocí CCD či CMOS kamer dochází k jejich znehodnocování adaptivním šumem. Tento adaptivní šum znemožňuje jednoduchou segmentaci obrazu a proto je nutné jej potlačit. V některých případech není vhodné použít filtr s konstantními vlastnostmi, ale je třeba navrhnout a použít tzv. adaptivní filtr, který reaguje na lokální vlastnosti obrazu.

Cíle diplomové práce:

Popsat metodu adaptivního filtru k potlačení aditivního šumu v obraze a vytvořit aplikaci, která tuto metodu využije.

Seznam odborné literatury:

M. Klíma,M. Bernas,J. Hozman,P. Dvořák: Zpracování obrazové informace.CVUT Praha.
William, K. Pratt.: Digital image processing (2nd ed.). John Wiley \& Sons, Inc. New York, NY, USA 1991.

Vedoucí diplomové práce: Ing. Pavel Štarha, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2013/2014.

V Brně, dne 21.11.2013

L.S.

prof. RNDr. Josef Šlapal, CSc. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c. Děkan fakulty

Abstrakt
Tato diplomová práce se zabývá problematikou, kterou je vhodné řešit metodou adaptivního filtru aditivního šumu v obraze. Odvozuje algoritmus metody adaptivního filtru. Následně studuje vlastnosti tohoto filtru. Na závěr aplikuje odvozený algoritmus na konkrétní obraz a výsledky porovnává s lineárním filtrováním.

Summary
This master's thesis deals with issues, which are suitable to solve by the method of adaptive filtering of additive noise in image processing. It deduces an algorithm of the adaptive filtering method. Then it studies properties of this filter. In the end it applies deduced algorithm on a particular picture and compares the results with linear filtering.

Klíčová slova
aditivní šum, segmentace, lineární filtr

Keywords
additive noise, segmentation, linear filter

VENCLOVSKÝ, J. *Numerická metoda filtrace aditivního šumu v obraze pomocí adaptivního filtru.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014. 46 s. Vedoucí Ing. Pavel Štarha, Ph.D.

Prohlašuji, že jsem diplomovou práci *Numerická metoda filtrace aditivního šumu v obraze pomocí adaptivního filtru* vypracoval samostatně pod vedením Ing. Pavla Štarhy, Ph.D. za použití materiálů uvedených v seznamu literatury.

Bc. Jakub Venclovský

Rád bych poděkoval svému školiteli Ing. Pavlu Štarhovi, Ph.D. za vedení mé diplomové práce.

Bc. Jakub Venclovský

Obsah

1. Úvod

Zpracování obrazu je matematická disciplína, které se v poslední době dostává stále větší pozornosti. Nachází uplatnění jak ve výzkumu tak i v praxi. S rozvojem různých snímacích technik tvoří stále významnější součást lékařství. Je nedílnou součástí studia vesmíru. Používá se v mnoha technicky zaměřených oborech. A takto bychom mohli pokračovat.

Problematiku zpracování obrazu lze obecně rozdělit do více odvětví, a to dle cíle, jehož se snažíme dosáhnout.

V prvé řadě se jedná o tzv. **image restoration** (obnovení obrazu). Toto odvětví se zabývá snahou minimalizovat negativní dopady, jež byly způsobeny technickou realizací sejmutí obrazu. Do těchto negativních dopadů spadá vznik šumu v obraze, ztráta informací při přenosu apod. Nejčastější aplikací spadající do tohoto odvětví, je pravděpodobně potlačení šumu.

Dalším odvětvím je tzv. **image compression** (komprese obrazu). Cílem tohoto odvětví je zkomprimovat obraz tak, aby na jednu stranu nadále zaujímal co nejméně bytů paměti, ale aby na druhou stranu nedošlo ke ztrátě požadovaných vlastností obrazu. Vlastnosti, které nechceme ovlivnit, se pak obecně liší. Např. pro obraz určený pro potěchu lidského oka se komprese provádí tak, aby se zachovaly vlastnosti, na něž je citlivé právě lidské oko. V některých technických aplikacích naopak nemusí hrát roli barevná hloubka atd.

Třetí odvětví je tzv. **image enhancement** (posílení, zvýraznění obrazu). Zde je snaha posílit (zvýraznit) některé konkrétní vlastnosti obrazu. Nejčastěji sem tak spadá zvýraznění obrazových hran, úprava kontrastu a jasu, ale také například zvětšení.

Posledním, z hlediska výzkumu a vědy pravděpodobně nejvýznamnějším odvětvím, je tzv. **image analysis** (analýza obrazu). Do tohoto odvětví patří analýza objektů v obrazech. Spadá sem například nalezení a zvýraznění požadovaných oblastí obrazu. Nalezení, popřípadě spočítání jistých prvků v obraze apod. Patří sem ale například i nalezení obličeje při fotografování. Obecně lze říci, že snahou tohoto odvětví je dosáhnout stavu, kdy stroj bude schopen vyhodnotit obraz stejně jako člověk. Nutno podotknout, že k tomuto stavu je třeba ujít ještě dlouhou cestu. Dílčí aplikace jsou úspěšné. To je ale dáno jejich specifickým zaměřením. Člověk je tak stále nenahraditelným „analytikem“ obrazu.

2. Základní pojmy

V této kapitole budou představeny některé pojmy, které budou používány v diplomové práci. Tato kapitola čerpá z [3] a [1].

2.1. Obrazová funkce

Obrazovou funkcí $f(x, y)$ rozumíme spojitou dvourozměrnou funkci f argumentů $x, y \in \mathbb{R}$. Funkční hodnoty této funkce pak v každém (x, y) nabývají hodnot dané fotometrické veličiny (např. jasu) určené na dané ploše. Stejně jako x, y, také $f(x, y) \in \mathbb{R}$.

2.2. Obrazová matice

Kvůli jednoduchosti si obrazovou matici definujme poněkud nestandardně. Obrazovou maticí F rozumějme zobrazení:

$$F : X \times Y \to \mathbb{R}, \text{ kde } X, Y \text{ jsou konečné podmnožiny } \mathbb{Z}.$$

Čísla $\overline{x}, \overline{y} \in \mathbb{Z}$ představují řádkové a sloupcové indexy této matice. Na rozdíl od klasické matice tedy indexy mohou nabývat i hodnot 0 a záporných. Řádkové i sloupcové indexy dané matice pak nabývají svých minimálních a maximálních hodnot, jejichž rozdíly určují velikost matice. Zobrazení této matice do klasické matice by bylo prostým posunem.

V praxi potom platí $F(\overline{x}, \overline{y}) \in \langle 0, 255 \rangle \subset \mathbb{N}$. Zatím se však touto skutečností neomezujme a nadále předpokládejme $F(\overline{x}, \overline{y}) \in \mathbb{R}$.

Dílčí prvky obrazové matice nazýváme obrazové body nebo také pixely.

2.3. Aditivní šum

Mějme dvě obrazové matice:

F – obrazová matice dokonalého obrazu,
A – obrazová matice, jejíž pixely jsou stochasticky nezávislé realizace dané náhodné veličiny.

Položme:

$$F_A = F + A.$$

Pak řekneme, že obraz daný obrazovou maticí F_A obsahuje aditivní šum. Matici A potom označíme jako šumovou matici.

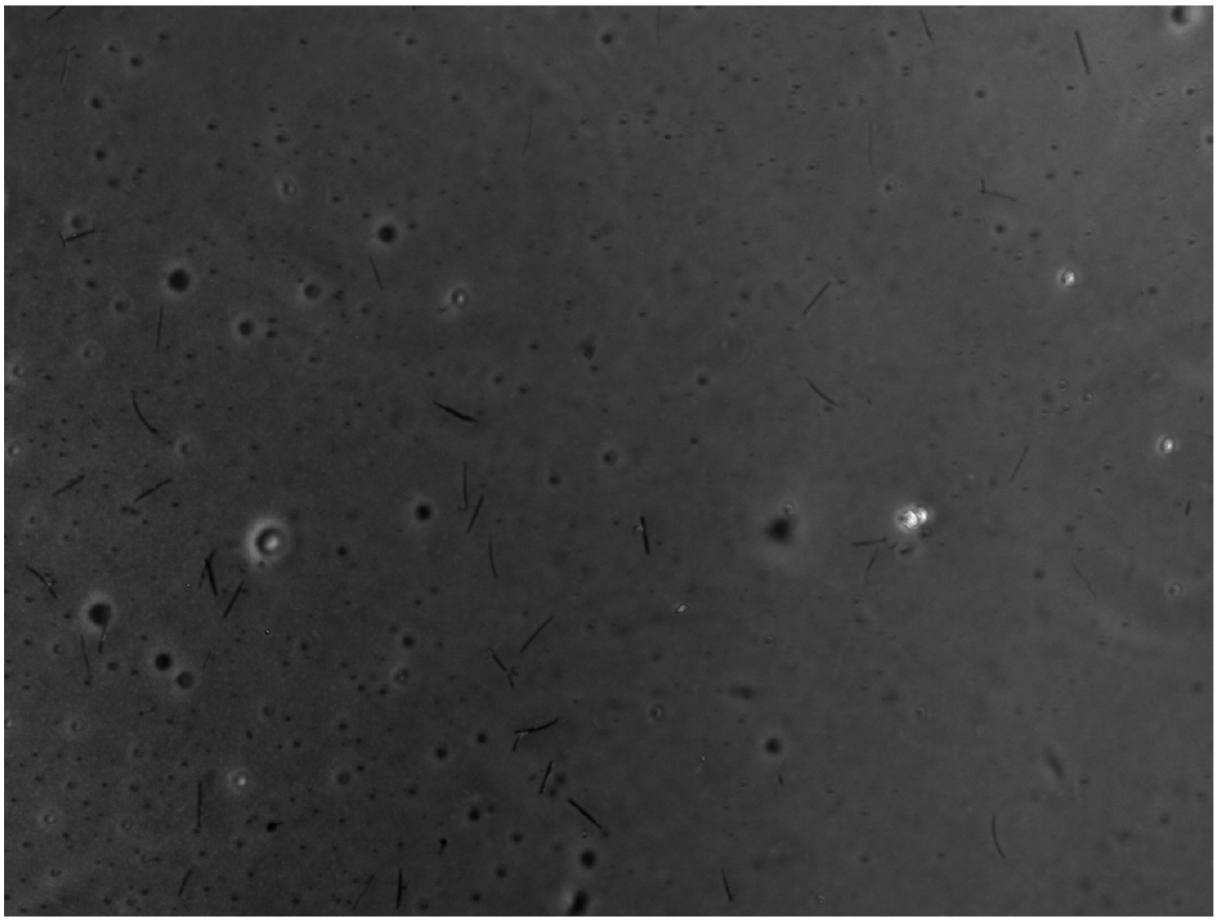
2.4. Histogram

Histogram je definován jako tzv. funkce četnosti

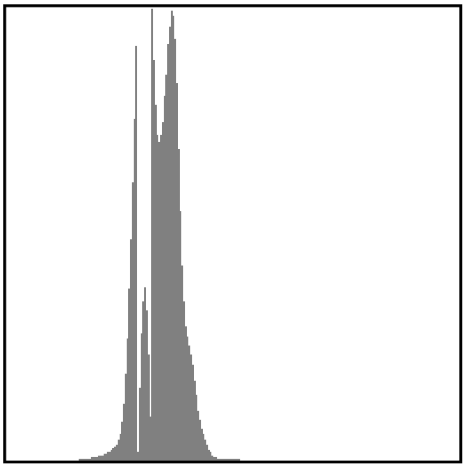
$H = H(u)$, kde
u – příslušná úroveň, hladina,
H – četnost výskytu hladiny u v obraze.

3. Problematika

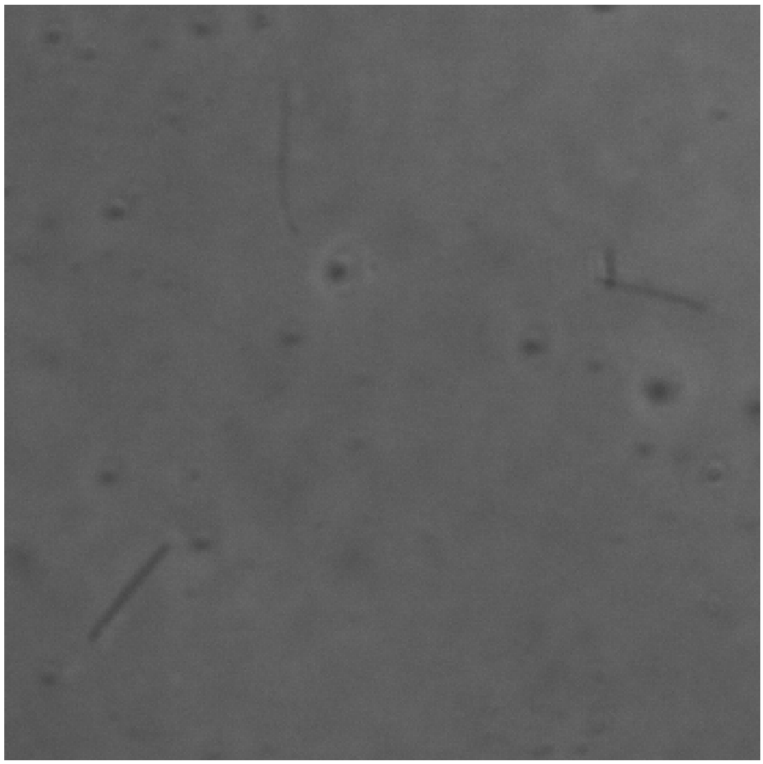
V této kapitole si ukážeme problematiku, kterou bude vhodné řešit použitím adaptivního filtru. Kapitola čerpá z [4]. Mějme následující snímek (obrázek 1), jeho histogram (obrázek 2) a kvůli detailům i jeho zvětšený výřez (obrázek 3).



Obrázek 1: Původní snímek.



Obrázek 2: Histogram původního snímku.



Obrázek 3: Výřez původního snímku.

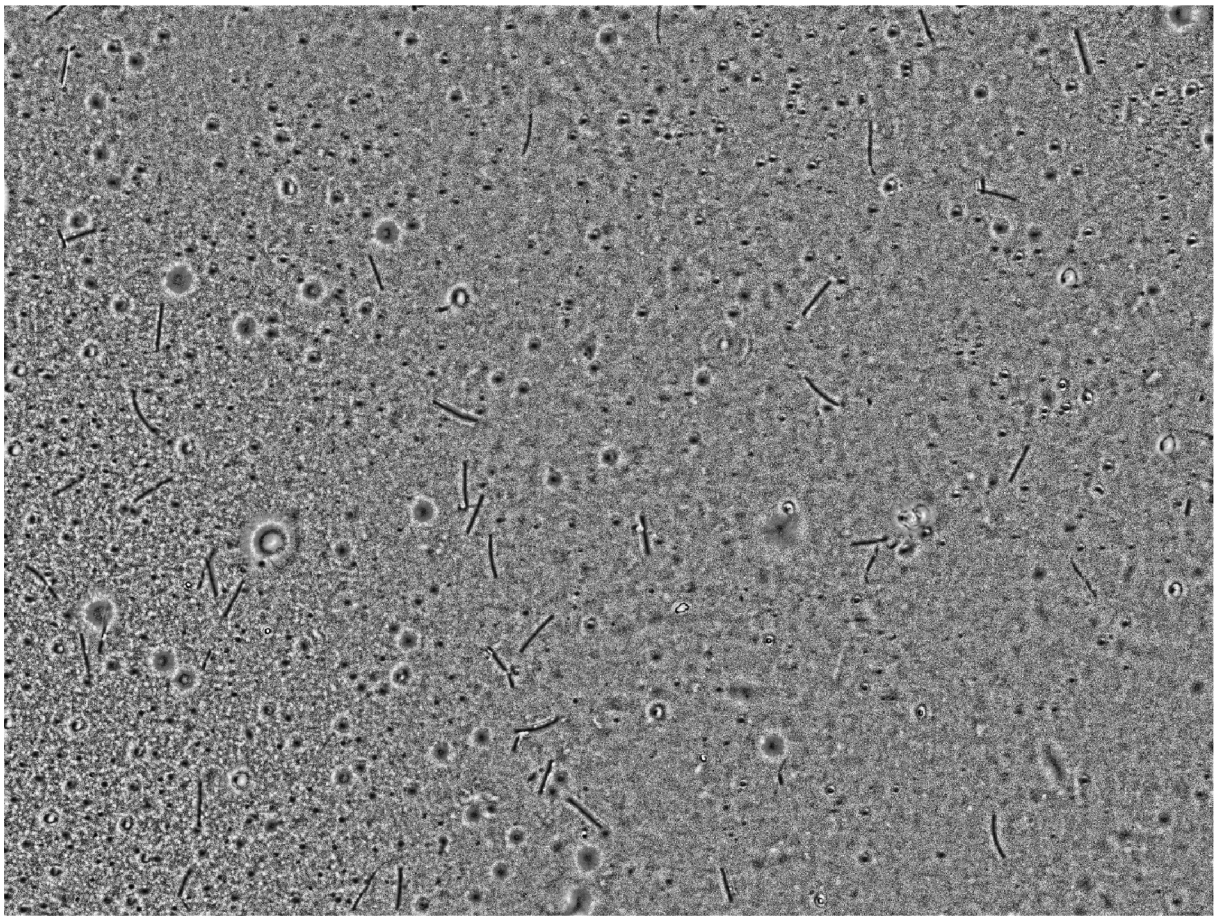
Všimněme si, že snímek je v odstínech šedi. Má tedy 8-bitovou barevnou hloubku. Jednotlivé hodnoty z intervalu $\langle 0, 255 \rangle$ pak v příslušné obrazové matici reprezentují jas v daném bodě. V diplomové práci budeme pracovat pouze s obrázky tohoto typu.

Tento konkrétní snímek je mikroskopickým snímkem skleněných vláken. Naším úkolem je tato vlákna separovat z obrazu pro další práci. Obraz tedy bude třeba segmentovat. Jako ideální způsob segmentace se jeví prahování. Daný úkol bychom rádi automatizovali. Současný snímek evidentně není pro prahování vhodný. Podívejme se na jeho histogram na obrázku 2. Odtud je patrné, že jasové hodnoty obrazu zaujímají pouze malou část celkového možného rozpětí $\langle 0, 255 \rangle$. Na snímek tedy bude vhodné použít metodu adaptivní ekvalizace histogramu (např. v [2]). Provedením této metody získáme následující obraz (obrázek 4) a opět jeho histogram (obrázek 5a) a výřez (obrázek 6).

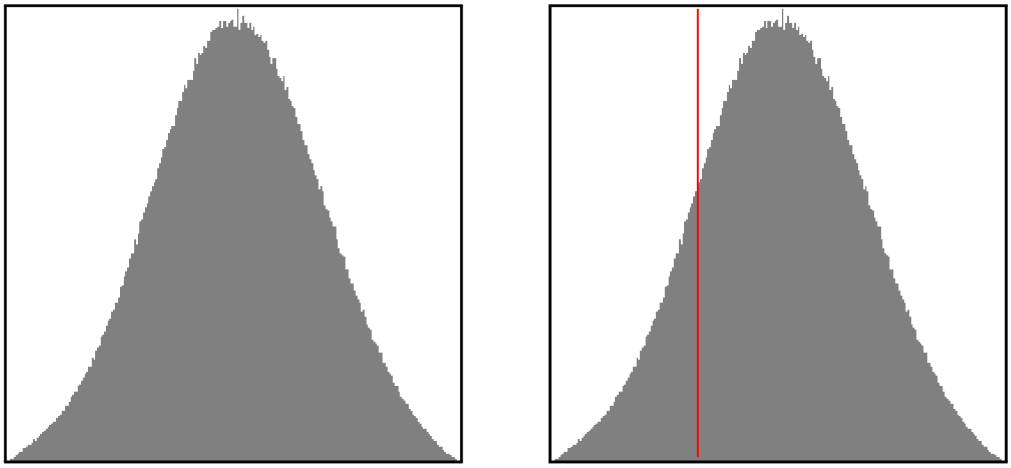
Z histogramu je patrné, že upravený snímek již má na intervalu $\langle 0, 255 \rangle$ rovnoměrné rozložení jasových hodnot. Tím byly zdůrazněny detaily obrazu, současně s nimi však také přítomný aditivní šum. Kdybychom nyní chtěli provádět segmentaci obrazu na základě prahování, jednak bychom měli problém se stanovením hodnoty prahu, a jednak by takto prahovaný obraz byl znehodnocen zmíněným šumem. Histogram s naznačeným prahem je na obrázku 5b. Výsledek prahování potom na obrázku 7.

Práh byl zvolen poměrně vysoký, aby byly zachovány tvary vláken. Postupnou volbou nižších prahů by se z prahovaného obrazu ztrácel šum, ale i okraje vláken. Schůdnější cestou k řešení našeho problému, bude filtrovat šum v obraze a současně pokud možno zachovat tvary vláken. Filtrovat šum můžeme i lineárním filtrem typu dolní propust. Tento filtr ale poruší také vlákna. Výhodnější tedy bude filtr, jež bude reagovat na lokální vlastnosti obrazu – filtr adaptivní. Ten bude ve svém spojitém tvaru představen v další kapitole. Dle dělení uvedeného v úvodu, bychom tento filtr mohli zařadit jak do odvětví

image analysis (vzhledem k tomu, že našim cílem je segmentace), tak i do odvětví image enhancement.

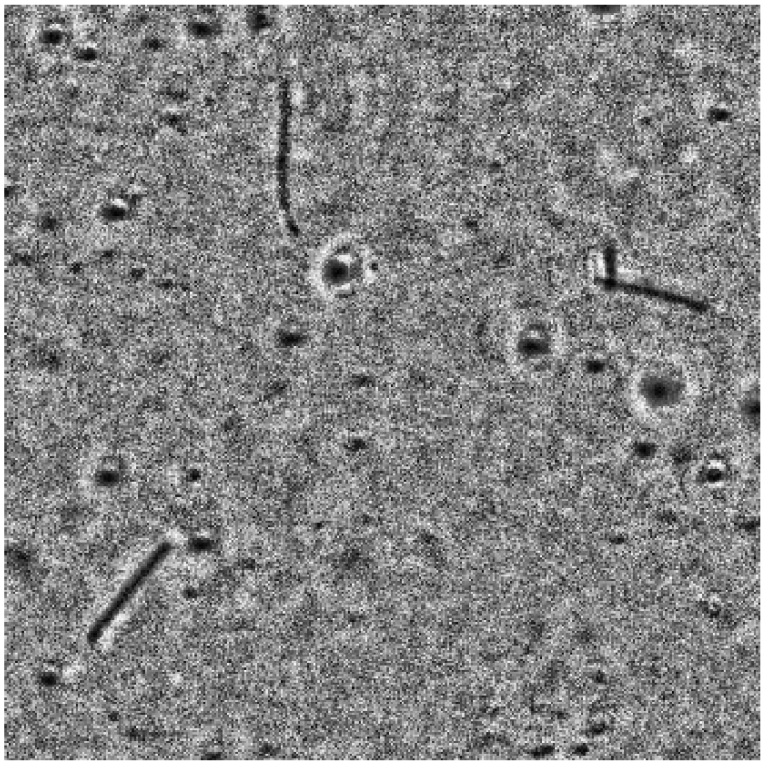


Obrázek 4: Snímek po adaptivní ekvalizaci histogramu.

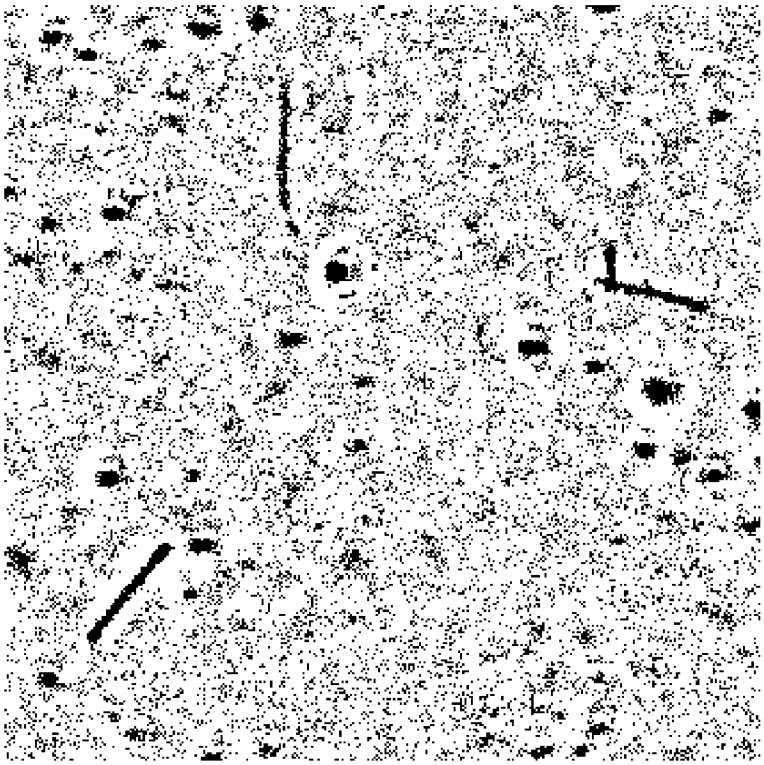


(a) Histogram upraveného snímku. (b) Naznačený práh 83.

Obrázek 5: Histogram upraveného snímku.



Obrázek 6: Výřez upraveného snímku.



Obrázek 7: Prahovaný výřez.

4. Adaptivní filtr aditivního šumu

V této kapitole si představíme adaptivní filtr aditivního šumu, a to na spojité obrazové funkci. Jeho praktické řešení na obrazové matici a následná algoritmizace budou předměty kapitoly příští. Abychom si mohli ukázat adaptivní filtr, je potřeba si nejprve ukázat filtr lineární.

Tato kapitola čerpá z [4].

4.1. Lineární filtr

Lineární filtr využívá konvoluce funkcí dvou proměnných.

4.1.1. Konvoluce funkcí dvou proměnných

Konvoluce funkcí dvou proměnných je definována jako:

$$\tilde{f}(x,y) = (f * h)(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-\alpha, y-\beta)\ h(\alpha,\beta)\ d\alpha\ d\beta,$$

kde $f(x,y)$ je naše obrazová funkce a $h(\alpha,\beta)$ je tzv. konvoluční jádro.

Proveďme transformaci souřadnic α, β do polárních souřadnic ρ, ϕ:

$$\alpha = \rho\cos\phi$$
$$\beta = \rho\sin\phi.$$

Dosazením dostaneme:

$$\tilde{f}(x,y) = \int_{0}^{2\pi}\int_{0}^{\infty} f(x-\rho\cos\phi, y-\rho\sin\phi)\ h(\rho\cos\phi,\rho\sin\phi)\ \rho\ d\rho\ d\phi. \tag{4.1}$$

4.1.2. Konvoluční jádro

Nejprve předpokládejme, že konvoluční jádro bude radiálně symetrické, tedy pro něj bude platit:

$$\forall\phi_1,\phi_2 \qquad h(\rho\cos\phi_1,\rho\sin\phi_1) = h(\rho\cos\phi_2,\rho\sin\phi_2).$$

Všimněme si, že díky radiální symetrii se z konvolučního jádra jako funkce ρ a ϕ stává funkce pouze ρ. Označme:

$$g(\rho) = \ h(\rho\cos\phi,\rho\sin\phi).$$

Aby konvoluce zachovala průměrnou hodnotu obrazové funkce, musí platit:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\alpha,\beta)\ d\alpha\ d\beta = 1, \tag{4.2}$$

tedy v polárních souřadnicích:

$$\int_{0}^{2\pi}\int_{0}^{\infty} g(\rho)\ \rho\ d\rho\ d\phi = 1,$$

tedy:

$$\int_0^\infty g(\rho)\ \rho\ d\rho = \frac{1}{2\pi}. \tag{4.3}$$

Jako konvoluční jádro zvolme dvoudimenzionální Gaussovu funkci, která je pro filtr typu dolní propust zcela běžně používaná:

$$h(\alpha,\beta) = \frac{1}{2\pi\sigma_\alpha\sigma_\beta}\exp\left(-\frac{(\alpha-\alpha_0)^2}{2\sigma_\alpha^2} - \frac{(\beta-\beta_0)^2}{2\sigma_\beta^2}\right).$$

Tato fukce splňuje 4.2. Aby navíc byla radiálně symetrická, musí platit:

$$\alpha_0 = \beta_0 = 0$$
$$\sigma_\alpha = \sigma_\beta = \sigma.$$

Dostáváme tedy:

$$h(\alpha,\beta) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{\alpha^2+\beta^2}{2\sigma^2}\right).$$

V polárních souřadnicích pak:

$$g(\rho) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{\rho^2}{2\sigma^2}\right). \tag{4.4}$$

Po dosazení do 4.1 tak dostáváme klasický Gaussovský lineární filtr:

$$\tilde{f}(x,y) = \int_0^{2\pi}\int_0^\infty f(x-\rho\cos\phi, y-\rho\sin\phi)\ \frac{1}{2\pi\sigma^2}\exp\left(-\frac{\rho^2}{2\sigma^2}\right)\ \rho\ d\rho\ d\phi.$$

4.2. Adaptivní filtr

Náš adaptivní filtr se od klasického lineárního bude lišit pouze přidáním směrových vah.

4.2.1. Směrové váhy

Označme tyto váhy jako $\omega(x,y,\phi)$. Adaptivní filtr potom bude mít tvar:

$$\tilde{f}(x,y) = \int_0^{2\pi}\int_0^\infty f(x-\rho\cos\phi, y-\rho\sin\phi)\ g(\rho)\ \omega(x,y,\phi)\ \rho\ d\rho\ d\phi. \tag{4.5}$$

Pro zachování průměrné hodnoty obrazové funkce žřejmě musí platit:

$$\int_0^{2\pi}\int_0^\infty g(\rho)\ \omega(x,y,\phi)\ \rho\ d\rho\ d\phi = 1. \tag{4.6}$$

Využitím 4.3 dostaneme:

$$\int_0^{2\pi}\frac{1}{2\pi}\ \omega(x,y,\phi)\ d\phi = 1.$$

Musí tedy platit:

$$\int_0^{2\pi}\omega(x,y,\phi)\ d\phi = 2\pi. \tag{4.7}$$

4.2.2. Určení směrových vah

Hodnotu směrových vah můžeme určit například pomocí rozptylu. Čím menší bude v daném směru rozptyl, tím větší budeme chtít váhu tohoto směru. Nejprve tedy spočtěme střední hodnotu obrazové funkce f od bodu (x, y) do bodu ve směru ϕ a ve vzdálenosti R. Označme ji $\mu(x, y, \phi)$:

$$\mu(x, y, \phi) = \frac{1}{R} \int_0^R f(x - \rho \cos \phi, y - \rho \sin \phi) \; d\rho.$$

Nyní pro stejnou část obrazové funkce spočtěme rozptyl. Označme jej $s^2(x, y, \phi)$:

$$s^2(x, y, \phi) = \frac{1}{R} \int_0^R \big[f(x - \rho \cos \phi, y - \rho \sin \phi) - \mu(x, y, \phi) \big]^2 d\rho.$$

Jak již bylo řečeno, chceme aby váha v daném směru klesala s rostoucím rozptylem. Uvažujme proto:

$$v(x, y, \phi) = \frac{1}{s^2(x, y, \phi)}.$$

Nezapomeňme však, že musí platit 4.7. Zaveďme proto:

$$V(x, y) = \int_0^{2\pi} v(x, y, \phi) \; d\phi.$$

Váhy $\omega(x, y, \phi)$ pak budou tvaru:

$$\omega(x, y, \phi) = 2\pi \; \frac{v(x, y, \phi)}{V(x, y)}.$$

5. Numerický přístup

V této kapitole si ukážeme numerické řešení problémů nastolených v předchozí kapitole.

5.1. Obrazová matice

Nejprve se zaměřme na fakt, že naše obrazová funkce $f(x, y)$ se skládá z jednotlivých pixelů. Spojitou dvourozměrnou obrazovou funkci $f(x, y)$ tak nahradí obrazová matice $F(\overline{x}, \overline{y})$, kde $\overline{x}, \overline{y} \in \mathbb{Z}$. Situaci si však lze představit také tak, že i nadále máme spojitou dvourozměrnou obrazovou funkci $f(x, y)$, která je ale nyní po částech konstantní. Tato funkce je rozprostřena na síti čtverců o stranách délky 1, přičemž je konstantní v každém jednotlivém čtverci. Zaveďme tuto čtvercovou síť tak, že body $(\overline{x}, \overline{y})$ budou představovat středy jednotlivých čtverců. Pak jistě platí $F(\overline{x}, \overline{y}) = f(\overline{x}, \overline{y})$. Hranice těchto čtverců jsou potom tvořeny přímkami, které lze rozdělit do dvou množin:

$$\text{a)} \left\{(x, y),\ x = k + \frac{1}{2},\ k \in \mathbb{Z},\ y \in \mathbb{R}\right\} \text{ – vertikální hranice pixelů,}$$

$$\text{b)} \left\{(x, y),\ y = l + \frac{1}{2},\ l \in \mathbb{Z},\ x \in \mathbb{R}\right\} \text{ – horizontální hranice pixelů.}$$

5.2. Diskretizace adaptivního filtru

Nejprve si stanovme, že hodnoty pixelů nového filtrovaného obrazu, reprezentovaného obrazovou maticí $\tilde{F}(\overline{x}, \overline{y})$, budou odpovídat hodnotám určených adaptivním filtrem ve středech daných pixelů původního obrazu (myšlenka je patrná z 5.1).

Vycházejme z 4.5. Diskretizaci provedeme tak, že konvoluci nadále nebudeme provádět vzhledem k ϕ na celém intervalu $\langle 0, 2\pi\rangle$, ale pouze na předem daném počtu N směrů $i = 1, 2, \ldots, N$. Tyto směry jsou dány úhly ϕ_i. V rovnici tak bude nadále chybět i Jakobián ρ (naším záměrem je aplikovat na každý směr funkci $g(\rho)$ jakožto jednorozměrnou). Volme:

$$\phi_i = (i - 1)\frac{2\pi}{N},\ i = 1, 2, \ldots, N.$$

Současně vzhledem k ρ omezíme konvoluci pouze do vzdálenosti D. Dostáváme:

$$\tilde{F}(\overline{x}, \overline{y}) = \sum_{i=1}^{N} \int_0^D f(\overline{x} - \rho\cos\phi_i, \overline{y} - \rho\sin\phi_i)\ g(\rho)\ \omega(\overline{x}, \overline{y}, \phi_i)\ d\rho. \tag{5.1}$$

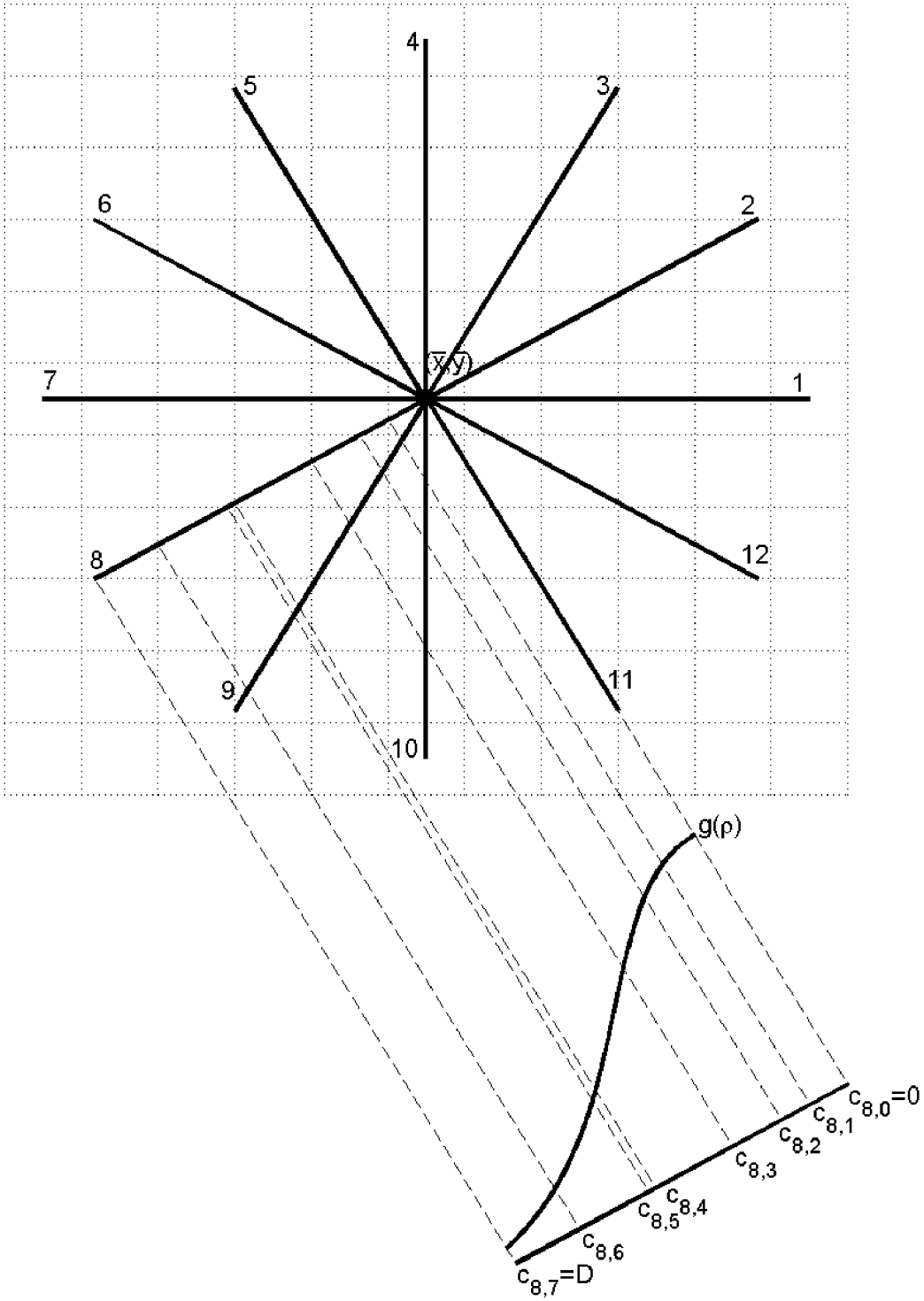
Čísla N a D se stávají prvními dvěma parametry našeho adaptivního filtru. Třetím a posledním parametrem bude číslo σ z $g(\rho)$ (viz 4.4).

Vzhledem k těmto změnám, musíme obměnit i podmínku pro zachování průměrné hodnoty obrazové funkce 4.6. Nově musí platit:

$$\sum_{i=1}^{N} \int_0^D g(\rho)\ \omega(\overline{x}, \overline{y}, \phi_i)\ d\rho = 1. \tag{5.2}$$

Dále využijme faktu, že obrazová funkce $f(x, y)$ je po částech konstantní na čtvercové síti pixelů. Každý směr tak bude s touto čtvercovou sítí tvořit průsečíky. Podívejme se na

obrázek 8. Obrázek ukazuje bod $(\overline{x}, \overline{y})$ jakožto střed jednoho z pixelů a nadále rozkreslených $N = 12$ směrů $i = 1, \ldots, N$. U jednoho ze směrů (zde směr 8) jsou pak rozkresleny jednotlivé průsečíky tohoto směru s čtvercovou sítí pixelů.



Obrázek 8: Adaptivní filtr o $N = 12$ směrech a délce $D = 5$.

5.2.1. Průsečíky daných směrů s hranicemi pixelů

Především nás zajímají vzdálenosti těchto průsečíků od bodu $(\overline{x}, \overline{y})$, a to pouze ty, jež jsou menší nebo rovny D. Současně je chceme mít seřazeny podle velikosti. Označme vzdálenosti průsečíků od bodu $(\overline{x}, \overline{y})$ jako $c_{i,j}$, kde i je indexem směru a j udává pořadí průsečíků dle vzdálenosti od $(\overline{x}, \overline{y})$ (viz obrázek 8). Jelikož počet průsečíků bude v různých směrech obecně různý, zaveďme pro každý směr množinu J_i indexů j.

Prozatím se omezme pouze na první kvadrant (tj. $\phi_i \in \left\langle 0, \frac{\pi}{2} \right\rangle$) a zároveň položme $(\overline{x}, \overline{y}) = (0, 0)$. Průsečíky směru daného ϕ_i s vertikálními hranicemi pixelů potom budou odpovídat bodům:

$$(k + 1/2,\ c_v(i,k)\sin\phi_i), \text{ kde} \quad k \in \mathbb{N},$$
$$c_v(i,k) \text{ – vzdálenost daného průsečíku od bodu } (\overline{x}, \overline{y}).$$

Samozřejmě platí:

$$\left(k + \frac{1}{2},\ c_v(i,k)\sin\phi_i\right) = \left(c_v(i,k)\cos\phi_i,\ c_v(i,k)\sin\phi_i\right)$$
$$k + \frac{1}{2} = c_v(i,k)\cos\phi_i.$$

Pro $\cos\phi_i \neq 0$ tedy vzdálenost $c_v(i,k)$ určíme:

$$c_v(i,k) = \frac{k + \frac{1}{2}}{\cos\phi_i}.$$

Analogicky pro průsečíky s horizontálními hranicemi pixelů určíme vzdálenost $c_h(i,l)$ pro $\sin\phi_i \neq 0$ jako:

$$c_h(i,l) = \frac{l + \frac{1}{2}}{\sin\phi_i}, \quad l \in \mathbb{N}.$$

Snadno nahlédneme, že pro $\cos\phi_i = 0$ průsečíky s vertikálními hranicemi pixelů neexistují. Analogicky pro $\sin\phi_i = 0$ neexistují průsečíky s horizontálními hranicemi. Prakticky budou tyto situace vyřešeny tak, že pro $\cos\phi_i = 0$ nastavíme hodnoty $c_v(i,k)$ na libovolná čísla větší než D. Analogicky pro $\sin\phi_i = 0$ a hodnoty $c_h(i,l)$. Jelikož nás zajímají pouze vzdálenosti menší nebo rovny D, jsou tak neexistující průsečíky vyloučeny.

Vzdálenosti $c_v(i,k)$, $c_h(i,l)$ se zřejmě nezmění posunem z bodu $(0,0)$ do libovolného bodu $(\overline{x}, \overline{y})$. Snadno také nahlédneme, že přestaneme-li se omezovat na první kvadrant, potřebujeme dát výrazy, jimiž jsou vyjádřeny vzdálenosti $c_v(i,k), c_h(i,l)$, do absolutních hodnot. Tedy platí:

$$c_v(i,k) = \frac{k + \frac{1}{2}}{|\cos\phi_i|}, \qquad c_h(i,l) = \frac{l + \frac{1}{2}}{|\sin\phi_i|}.$$

Pro $k_1 < k_2$, $l_1 < l_2$ zřejmě platí:

$$c_v(i,k_1) < c_v(i,k_2), \qquad c_h(i,l_1) < c_h(i,l_2).$$

Nejbližším průsečíkem bodu $(\overline{x}, \overline{y})$ ve směru ϕ_i zřejmě bude průsečík ve vzdálenosti:

$$c_{i,1} = \min(c_v(i,0),\ c_h(i,0)).$$

V závislosti na tom, zda to bude $c_v(i,0)$ či $c_h(i,0)$, pak pro $c_{i,2}$ bude platit:

a) $c_v(i,0) < c_h(i,0)$:
$$c_{i,2} = \min(c_v(i,1),\ c_h(i,0)),$$

b) $c_v(i,0) > c_h(i,0)$:
$$c_{i,2} = \min(c_v(i,0),\ c_h(i,1)),$$

c) $c_v(i,0) = c_h(i,0)$:
$$c_{i,2} = \min(c_v(i,1),\ c_h(i,1)).$$

Zjednodušeně řečeno, bude-li použit průsečík s vertikálními hranicemi, zvětšíme koeficient k u $c_v(i,k)$ o 1. Bude-li použit průsečík s horizontálními hranicemi, zvětšíme l u $c_h(i,l)$. A budou-li použity oba dva ($c_v(i,k) = c_h(i,l)$ odpovídá situaci, kdy směr daný ϕ_i prochází přesně vrcholem pixelu), zvětšíme k i l. Dále se bude pokračovat analogicky. Získání vzdáleností $c_{i,j}$ a množiny J_i můžeme vidět zapsané v pseudokódu v algoritmu 1:

Algoritmus 1 Určení $c_{i,j}$ a J_i

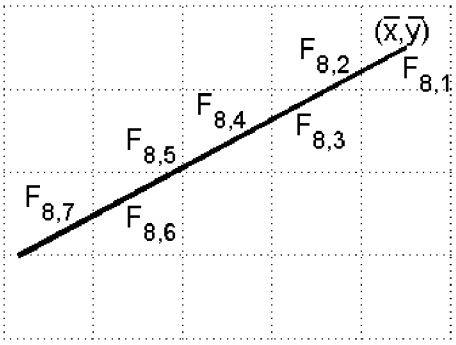
$c_{i,0} := 0$
$j := 1;\ k := 0;\ l := 0$
$c_v := \frac{1/2}{|\cos\phi_i|};\ c_h := \frac{1/2}{|\sin\phi_i|}$
while $c_v < D$ **or** $c_h < D$ **do**
  $c_{i,j} := \min(c_v,\ c_h)$
  if $c_v \leq c_h$ **then**
    $k := k + 1$
  if $c_h \leq c_v$ **then**
    $l := l + 1$
  $c_v := \frac{k+1/2}{|\cos\phi_i|};\ c_h := \frac{l+1/2}{|\sin\phi_i|}$
  $j := j + 1$
end while
$c_{i,j} := D$
$J_i := \{1, \ldots, j\}$

Pozn. Kvůli zkrácení zápisu nebyl použit přesný zápis pro výpočet hodnot c_v a c_h – při dělení nulou ($\cos\phi_i = 0$ nebo $\sin\phi_i = 0$) se příslušná hodnota nastaví na libovolné číslo větší než D (standardně by bylo zapsáno pomocí **if** ... **then**).

5.2.2. Pixely v daných směrech

Označme $F_{i,j}(\overline{x}, \overline{y})$ hodnotu v pořadí j-tého pixelu v i-tém směru od bodu $(\overline{x}, \overline{y})$. Indexy j jsou zřejmě opět z množiny J_i. Situace je znázorněna na obrázku 9. Na obrázku se opět jedná o směr 8 z obrázku 8.

Zaveďme:
$$F_{i,j}(\overline{x}, \overline{y}) = F(\overline{x} + a_{i,j,1},\ \overline{y} + a_{i,j,2}).$$



Obrázek 9: Znázornění hodnot $F_{i,j}(\overline{x}, \overline{y})$.

Protože máme čtvercovou síť pixelů, je zřejmé, že $a_{i,j,1}, a_{i,j,2} \in \mathbb{Z}$. Ačkoliv se hodnoty $F_{i,j}(\overline{x}, \overline{y})$ mění s měnícím se $(\overline{x}, \overline{y})$, hodnoty $a_{i,j,1}, a_{i,j,2}$ jsou na $(\overline{x}, \overline{y})$ zřejmě nezávislé. Je potom výhodné mít tyto hodnoty předpočítány. Zřejmě:

$$(a_{i,1,1},\ a_{i,1,2}) = (0, 0) \quad \forall i = 1, \ldots, N.$$

Dvojice $(a_{i,2,1},\ a_{i,2,2})$ se v závislosti na úhlu ϕ_i určí změnou dvojice $(a_{i,1,1},\ a_{i,1,2})$, a to pomocí porovnání vzdáleností $c_v(i, 0),\ c_h(i, 0)$ následovně:

a) $c_v(i, 0) < c_h(i, 0)$:

$$a_{i,2,1} = a_{i,1,1} + \operatorname{sgn}(\cos\phi_i) \qquad a_{i,2,2} = a_{i,1,2},$$

b) $c_v(i, 0) > c_h(i, 0)$:

$$a_{i,2,1} = a_{i,1,1} \qquad a_{i,2,2} = a_{i,1,2} + \operatorname{sgn}(\sin\phi_i),$$

c) $c_v(i, 0) = c_h(i, 0)$:

$$a_{i,2,1} = a_{i,1,1} + \operatorname{sgn}(\cos\phi_i) \qquad a_{i,2,2} = a_{i,1,2} + \operatorname{sgn}(\sin\phi_i).$$

Současně se zvětší koeficienty k nebo l nebo oba o 1 (stejně jako při určování vzdálenosti průsečíků). Dále se postupuje analogicky. Algoritmus můžeme vidět zapsaný v pseudokódu jako algoritmus 2. Je zřejmé, že algoritmy 1 a 2 bude výhodné skloubit do jednoho.

Algoritmus 2 Určení $(a_{i,j,1},\ a_{i,j,2})$

$(a_{i,1,1},\ a_{i,1,2}) := (0,\ 0)$
$j := 2;\ k := 0;\ l := 0$
$c_v := \frac{1/2}{|\cos\phi_i|};\ c_h := \frac{1/2}{|\sin\phi_i|}$
while $c_v < D$ **or** $c_h < D$ **do**
  if $c_v \leq c_h$ **then**
    $k := k + \operatorname{sgn}(\cos\phi_i)$
  if $c_h \leq c_v$ **then**
    $l := l + \operatorname{sgn}(\sin\phi_i)$
  $(a_{i,j,1},\ a_{i,j,2}) := (k,\ l)$
  $c_v := \frac{|k|+1/2}{|\cos\phi_i|};\ c_h := \frac{|l|+1/2}{|\sin\phi_i|}$
  $j := j + 1$
end while

Pozn. Stejně jako u předchozího algoritmu 1, nebyl ani v algoritmu 2 použit přesný zápis pro výpočet hodnot c_v a c_h – při dělení nulou ($\cos\phi_i = 0$ nebo $\sin\phi_i = 0$) se příslušná hodnota nastaví na libovolné číslo větší než D (standardně by bylo zapsáno pomocí **if** ... **then**).

5.2.3. Konvoluční maska

Díky tomu, že obrazová funkce $f(x, y)$ je po částech konstantní a díky určení vzdáleností $c_{i,j}$ a množin J_i, nyní můžeme konvoluční jádro nahradit konvoluční maskou.

Řekli jsme, že funkce $g(\rho)$ se řídí parametrem σ, a navíc jsme omezili horní hranici integrálu na D. Hodnota

$$\int_0^D g(\rho)\ d\rho$$

se tak zřejmě bude měnit s měnícími se parametry D, σ. V závislosti na těchto parametrech pak bude třeba provést normování, abychom dodrželi podmínku zachování průměrné hodnoty obrazové funkce. V dané chvíli si tak ale můžeme usnadnit výpočet zanedbáním normování, které bude provedeno až v budoucnu. Zavedeme (porovnej s 4.4):

$$g'(\rho) = \exp\left(-\frac{\rho^2}{2\sigma^2}\right).$$

Označme potom prvky konvoluční masky pro j-tý pixel ve směru i jako:

$$g_{i,j} = \int_{c_{i,j-1}}^{c_{i,j}} g'(\rho)\ d\rho.$$

Kvůli normování zaveďme:

$$G = \int_0^D g'(\rho)\ d\rho.$$

Prakticky bude $g_{i,j}$ spočítáno pomocí numerického integrálu. Jelikož integrály $g_{i,j}$ budou počítány $(\sum_{i=1}^{N} |J_i|)$-krát počítejme $g_{i,j}$ následovně. Zaveďme dostatečně malé h jako krok na ρ. Spočtěme n jako nahoru na celá čísla zaokrouhlený podíl $\frac{D}{h}$. Dostaneme:

$$\rho_k = kh,\ k = 0, 1, \ldots, n-1, \qquad \rho_n = D.$$

Dále spočtěme funkční hodnoty $g'(\rho)$ ve středech intervalů (ρ_{k-1}, ρ_k) a označme je z_k:

$$z_k = g'\left(\frac{\rho_{k-1} + \rho_k}{2}\right),\ k = 1, 2, \ldots, n,$$

což odpovídá:

$$z_k = g'\left(\rho_k - \frac{h}{2}\right),\ k = 1, 2, \ldots, n-1, \qquad z_n = g'\left(\frac{\rho_{n-1} + D}{2}\right).$$

Nyní aproximujme:

$$\int_{\rho_{k-1}}^{\rho_k} g'(\rho)\ d\rho \approx P_k = z_k(\rho_k - \rho_{k-1}),\ k = 1, 2, \ldots, n,$$

což odpovídá:

$$P_k = z_k h,\ k = 1, 2, \ldots, n-1, \qquad P_n = z_n(D - \rho_{n-1}).$$

Nakonec zaveďme kumulovaný součet hodnot P_k a označme jej S_k:

$$S_0 = 0, \qquad S_k = \sum_{l=1}^{k} P_l,\ k = 1, 2, \ldots, n.$$

Zajímají nás zejména hodnoty S_k. Zřejmě platí $G \approx S_n$. Příslušný algoritmus je zde:

Algoritmus 3 Určení S_k

$n :=$ Zaokrouhli nahoru $(\frac{D}{h})$
$\rho_0 := 0;\ S_0 := 0$
for $k := 1$ **to** $n - 1$ **do**
  $\rho_k := \rho_{k-1} + h$
  $z_k := g'\left(\rho_k - \frac{h}{2}\right)$
  $S_k := S_{k-1} + z_k h$
end for
$\rho_n := D$
$z_n := g'\left(\frac{\rho_{n-1}+D}{2}\right)$
$S_n := S_{n-1} + z_n(D - \rho_{n-1})$

Nyní určíme přibližné hodnoty $g_{i,j}$. Začněme s:

$$g_{i,1} = \int_0^{c_{i,1}} g'(\rho)\ d\rho.$$

Najděme nejmenší k, pro něž platí $\rho_k \geq c_{i,1}$. Pomocí lineární interpolace pak určeme přibližnou hodnotu $g_{i,1}$:

$$L = \frac{\rho_k - c_{i,1}}{\rho_k - \rho_{k-1}},$$

$$g_{i,1} \approx S_{k-1}L + S_k(1 - L).$$

Pokračujme s:

$$g_{i,2} = \int_{c_{i,1}}^{c_{i,2}} g'(\rho)\ d\rho.$$

Jako předtím najděme nejmenší k, pro něž platí $\rho_k \geq c_{i,2}$ a pomocí lineární interpolace určeme přibližnou hodnotu $g_{i,2}$:

$$L = \frac{\rho_k - c_{i,2}}{\rho_k - \rho_{k-1}},$$

$$g_{i,2} \approx S_{k-1}L + S_k(1 - L) - g_{i,1}.$$

Dále budeme pokračovat analogicky. Následuje příslušný algoritmus v pseudokódu.

Algoritmus 4 Určení $g_{i,j},\ G$

$k := 0;\ G := 0$
for all $j \in J_i$ **do**
  while $\rho_k < c_{i,j}$ **do**
    $k := k + 1$
  $L := \frac{\rho_k - c_{i,j}}{\rho_k - \rho_{k-1}}$
  $g_{i,j} := S_{k-1}L + S_k(1 - L) - G$
  $G := G + g_{i,j}$
end for

Prvky konvoluční masky $g_{i,j}$ tak máme spočítány. Nyní zaveďme zmíněné normování. Pro libovolné i bude platit:

$$\frac{1}{G}\sum_{j\in J_i} g_{i,j} = 1. \tag{5.3}$$

Toto bude důležité pro zachování průměrné hodnoty obrazové funkce.

Důvod, proč byl algoritmus 4 oddělen od algoritmu 3 je ten, že algoritmus 3 lze provést nezávisle na směru i, zatímco algoritmus 4 již musí být proveden pro každý směr zvlášť.

5.2.4. Diskrétní konvoluce

Abychom se mohli zabývat diskretizací směrových vah, zaveďme nejdřív finální tvar naší diskrétní konvoluce. Díky odvození prvků $c_{i,j}$, $g_{i,j}$, $a_{i,j,1}$ a $a_{i,j,2}$ přejde konvoluce z tvaru 5.1 do tvaru:

$$\tilde{F}(\overline{x},\overline{y}) = \frac{1}{G}\ \sum_{i=1}^{N}\sum_{j\in J_i} F(\overline{x} + a_{i,j,1},\ \overline{y} + a_{i,j,2})\ g_{i,j}\ \omega(\overline{x},\overline{y},\phi_i). \tag{5.4}$$

Pozn. Ke změně znamének v argumentech obrazové funkce f dojde kvůli struktuře odvozených prvků.

Podmínka pro zachování průměrné hodnoty obrazové funkce 5.2 přejde v podmínku:

$$\frac{1}{G}\ \sum_{i=1}^{N}\sum_{j\in J_i} g_{i,j}\ \omega(\overline{x},\overline{y},\phi_i) = 1.$$

Z této podmínky a platnosti vztahu 5.3 pak vyplývá podmínka pro směrové váhy:

$$\sum_{i=1}^{N}\omega(\overline{x},\overline{y},\phi_i) = 1. \tag{5.5}$$

Nyní se můžeme pustit do diskretizace směrových vah.

5.2.5. Směrové váhy

Vycházejme z podkapitoly 4.2.2. Využijeme toho, že funkce $f(x, y)$ je po částech konstantní a známe vzdálenosti $c_{i,j}$. Pro střední hodnotu tak platí:

$$\begin{aligned}\mu(\overline{x}, \overline{y}, \phi_i) &= \frac{1}{D} \int_0^D f(\overline{x} - \rho \cos \phi_i,\ \overline{y} - \rho \sin \phi_i)\ d\rho = \\ &= \frac{1}{D} \sum_{j \in J_i} \left[(c_{i,j} - c_{i,j-1})\ F(\overline{x} + a_{i,j,1},\ \overline{y} + a_{i,j,2}) \right].\end{aligned}$$

Její výpočet je potom také zapsán v následujícím algoritmu:

Algoritmus 5 Výpočet $\mu(\overline{x}, \overline{y}, \phi_i)$

$\mu(\overline{x}, \overline{y}, \phi_i) := 0$
for all $j \in J_i$ **do**
  $\mu(\overline{x}, \overline{y}, \phi_i) := \mu(\overline{x}, \overline{y}, \phi_i) + (c_{i,j} - c_{i,j-1})\ F(\overline{x} + a_{i,j,1},\ \overline{y} + a_{i,j,2})$
$\mu(\overline{x}, \overline{y}, \phi_i) := \frac{1}{D}\ \mu(\overline{x}, \overline{y}, \phi_i)$

Pro rozptyl bude platit:

$$\begin{aligned}s^2(\overline{x}, \overline{y}, \phi_i) &= \frac{1}{D} \int_0^D \left[f(\overline{x} - \rho \cos \phi_i,\ \overline{y} - \rho \sin \phi_i) - \mu(\overline{x}, \overline{y}, \phi_i) \right]^2 d\rho = \\ &= \frac{1}{D} \sum_{j \in J_i} \left\{ (c_{i,j} - c_{i,j-1})\ \left[F(\overline{x} + a_{i,j,1},\ \overline{y} + a_{i,j,2}) - \mu(\overline{x}, \overline{y}, \phi_i) \right]^2 \right\}.\end{aligned}$$

Opět následuje příslušný algoritmus v pseudokódu:

Algoritmus 6 Výpočet $s^2(\overline{x}, \overline{y}, \phi_i)$

$s^2(\overline{x}, \overline{y}, \phi_i) := 0$
for all $j \in J_i$ **do**
  $s^2(\overline{x}, \overline{y}, \phi_i) := s^2(\overline{x}, \overline{y}, \phi_i) + (c_{i,j} - c_{i,j-1})\ \left[F(\overline{x} + a_{i,j,1},\ \overline{y} + a_{i,j,2}) - \mu(\overline{x}, \overline{y}, \phi_i) \right]^2$
$s^2(\overline{x}, \overline{y}, \phi_i) := \frac{1}{D}\ s^2(\overline{x}, \overline{y}, \phi_i)$

Váhy chceme tím větší, čím menší bude rozptyl. Tedy:

$$v(\overline{x}, \overline{y}, \phi_i) = \frac{1}{s^2(\overline{x}, \overline{y}, \phi_i)}.$$

Chceme, aby se nezměnila průměrná hodnota obrazové funkce. Kvůli tomuto jsme odvodili podmínku 5.5. Zaveďme tedy:

$$V(\overline{x}, \overline{y}) = \sum_{i=1}^{N} v(\overline{x}, \overline{y}, \phi_i).$$

Váhy jednotlivých směrů pak budou mít tvar:

$$\omega(\overline{x}, \overline{y}, \phi_i) = \frac{v(\overline{x}, \overline{y}, \phi_i)}{V(\overline{x}, \overline{y})}.$$

Dosaďme zpět do vzorce 5.4. Změňme uspořádání jednotlivých operací tak, aby bylo z numerického hlediska co nejúspornější. Dostaneme:

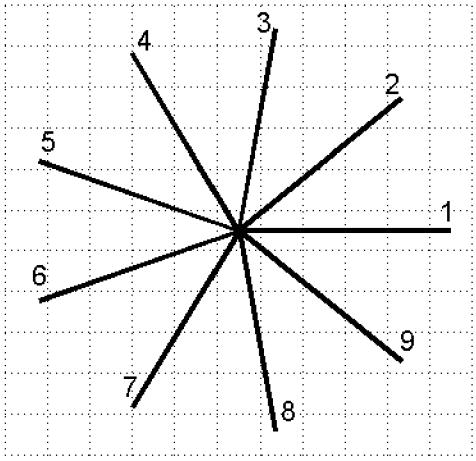
$$\tilde{F}(\overline{x},\overline{y}) = \frac{1}{V(\overline{x},\overline{y})}\ \frac{1}{G}\ \sum_{i=1}^{N}\left[v(\overline{x},\overline{y},\phi_i)\sum_{j\in J_i} F(\overline{x}+a_{i,j,1},\ \overline{y}+a_{i,j,2})\ g_{i,j}\right]. \qquad (5.6)$$

Než se pustíme do vytvoření celkového algoritmu pro náš adaptivní filtr, zmiňme ještě možnost zkrátit délku výpočtu prvků $c_{i,j}$, $g_{i,j}$, $a_{i,j,1}$ a $a_{i,j,2}$ pomocí zrcadlení (osové souměrnosti).

5.2.6. Zrcadlení

Pro zkrácení některých výše zmíněných výpočtů je možné využít zrcadlení námi zvoleného systému směrů. Zdůrazněme, že souměrné musí být jak směry samotné, tak i čtvercová síť. Jedině tak budou souměrné i jejich vzájemné průsečíky. Pro jednoduchost položme $(\overline{x},\overline{y}) = (0,0)$. Posunem do jiného $(\overline{x},\overline{y})$ zrcadlení zřejmě zůstanou stejná. Dle počtu směrů N rozlišíme 3 možná zrcadlení.

- a) Pro libovolný (tedy i lichý) počet směrů se jedná o souměrnost dle osy $y = 0$ (jelikož vždy volíme $\phi_1 = 0$). Viz obrázek 10:



Obrázek 10: Souměrnost $N = 9$ směrů.

Označme $N_2 = N \div 2 + 1$.

Pozn. $\div$ je binární operace (jež má přednost před sčítáním), která dvěma celým číslům přiřadí podíl jejich celočíselného dělení, zbytek je zanedbán – např. **div** z jazyka Pascal.

Obecně bude platit, že směr $N-k+1$ je zrcadlením směru $k+1$ pro $k = 1,\ldots,N-N_2$.

Tvrzení bychom dokázali přímo, a to zvlášť pro lichá a zvlášť pro sudá N.

Konkrétně pro náš obrázek 10, kde $N = 9$, tedy $N_2 = 5$, tedy $k = 1,\ldots,4$, platí souměrnost směrů $9,8,7,6$ se směry $2,3,4,5$ respektive, což přesně splňuje očekávání.

Například pro $N = 10$ (viz obrázek 11), platí $N_2 = 6$, tedy opět $k = 1,\ldots,4$, což dává souměrnost směrů $10,9,8,7$ se směry $2,3,4,5$ respektive, opět dle očekávání.

Zjednodušeně řečeno zůstane v takto zrcadlených směrech zachována množina J_i, vzdálenosti $c_{i,j}$, a tedy i hodnoty $g_{i,j}$. Zachová se i souřadnice $a_{i,j,1}$, souřadnice $a_{i,j,2}$ pak změní znaménko. Vyjádřeno matematicky tedy pro $k = 1, 2, \ldots, N - N_2$ platí:

$$\left.\begin{aligned} c_{N-k+1,j} &= c_{k+1,j} \\ g_{N-k+1,j} &= g_{k+1,j} \\ a_{N-k+1,j,1} &= a_{k+1,j,1} \\ a_{N-k+1,j,2} &= -a_{k+1,j,2} \end{aligned}\right\} j \in J_{N-k+1} = J_{k+1}$$

Tohoto lze zřejmě využít tak, že pomocí předchozích algoritmů určíme zmíněné hodnoty J_i, $c_{i,j}$, $g_{i,j}$, $a_{i,j,1}$, $a_{i,j,2}$ pouze pro směry $i = 1, \ldots, N_2$ a pro zbývající směry $i = N_2 + 1, \ldots, N$ je určíme pomocí algoritmu 7.

Algoritmus 7 Zrcadlení podle osy $y = 0$

for $k := 1$ **to** $N - N_2$ **do**

$J_{N-k+1} := J_{k+1}$

$c_{N-k+1,0} := 0$

$c_{N-k+1,J_{N-k+1}} := c_{k+1,J_{k+1}}$

$g_{N-k+1,J_{N-k+1}} := g_{k+1,J_{k+1}}$

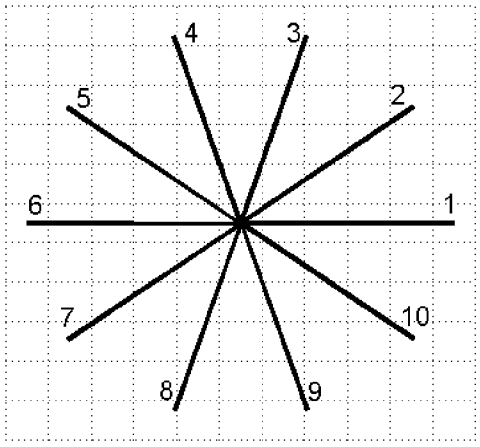
$a_{N-k+1,J_{N-k+1},1} := a_{k+1,J_{k+1},1}$

$a_{N-k+1,J_{N-k+1},2} := -a_{k+1,J_{k+1},2}$

end for

Pozn. Kvůli zkrácení zápisu jsme místo indexování čísly j použili indexování množinami J. Zřejmě máme na mysli postupně všechna $j \in J$.

- b) Pro sudý počet směrů lze kromě zrcadlení podle osy $y = 0$ využít i zrcadlení podle osy $x = 0$. Viz obrázek 11:



Obrázek 11: Souměrnost $N = 10$ směrů.

Lze snadno ukázat, že pro N sudé bude pro směr N_2 vždy platit:

$$N \text{ – sudé} \implies N_2 = \frac{N}{2} + 1,$$

$$\phi_{N_2} = \left(\frac{N}{2} + 1 - 1\right)\frac{2\pi}{N} = \pi.$$

Pro N sudé je tedy směr N_2 vždy dán úhlem π, a je tedy zrcadlením směru 1 podle osy $x = 0$. (Současně je to i důvod, proč bychom v zrcadlení dle $y = 0$ museli zmíněný důkaz provádět zvlášť pro lichá a zvlášť pro sudá N. Pro N sudé totiž směr N_2 leží přímo na ose $y = 0$, a nemůže se tak dle ní zrcadlit.)

Dále se soustřeďme pouze na směry $i = 1, \ldots, N_2$. Zbývající směry, jak již víme, určíme pomocí zrcadlení dle osy $y = 0$.

Označme $N_4 = N \div 4 + 1$.

Obecně bude platit, že směr $N_2 - k + 1$ je zrcadlením směru k pro $k = 1, \ldots, N_2 - N_4$.

Tvrzení bychom potom opět mohli dokázat přímo, a to zvlášť pro N sudé ale nedělitelné čtyřmi a zvlášť pro N dělitelné čtyřmi (obdobná situace jako v minulém případě, viz rovnici 5.7).

Konkrétně pro náš případ $N = 10$ (obrázek 11) máme $N_2 = 6$, $N_4 = 3$. Dostáváme tedy $k = 1, 2, 3$ a platí, že směry $6, 5, 4$ jsou zrcadlení směrů $1, 2, 3$ respektive, což splňuje naše očekávání.

Například pro $N = 12$ (viz obrázek 12a) bychom dostali $N_2 = 7$, $N_4 = 4$. Měli bychom $k = 1, 2, 3$, tedy směry $7, 6, 5$ jako zrcadlení směrů $1, 2, 3$ respektive. Přesně dle očekávání.

Zjednodušeně řečeno se v zrcadlených směrech opět zachová množina J_i, vzdálenosti $c_{i,j}$, tedy i hodnoty $g_{i,j}$. Tentokrát ale souřadnice $a_{i,j,1}$ změní znaménko, a naopak souřadnice $a_{i,j,2}$ zůstane zachována. Pro $k = 1, \ldots, N_2 - N_4$ tedy platí:

$$\left.\begin{aligned} c_{N_2-k+1,j} &= c_{k,j} \\ g_{N_2-k+1,j} &= g_{k,j} \\ a_{N_2-k+1,j,1} &= -a_{k,j,1} \\ a_{N_2-k+1,j,2} &= a_{k,j,2} \end{aligned}\right\} j \in J_{N_2-k+1} = J_k$$

Pomocí algoritmů 1 až 5 tak stačí určit prvky J_i, $c_{i,j}$, $g_{i,j}$, $a_{i,j,1}$, $a_{i,j,2}$ pouze pro směry $i = 1, \ldots, N_4$, směry $i = N_4 + 1, \ldots, N_2$ budou určeny pomocí následujícího algoritmu 8 a zbývající směry $i = N_2 + 1, \ldots, N$ určeny předchozím algoritmem 7.

Algoritmus 8 Zrcadlení podle osy $x = 0$

for $k := 1$ **to** $N_2 - N_4$ **do**

$J_{N_2-k+1} := J_k$

$c_{N_2-k+1,0} := 0$

$c_{N_2-k+1,J_{N_2-k+1}} := c_{k,J_k}$

$g_{N_2-k+1,J_{N_2-k+1}} := g_{k,J_k}$

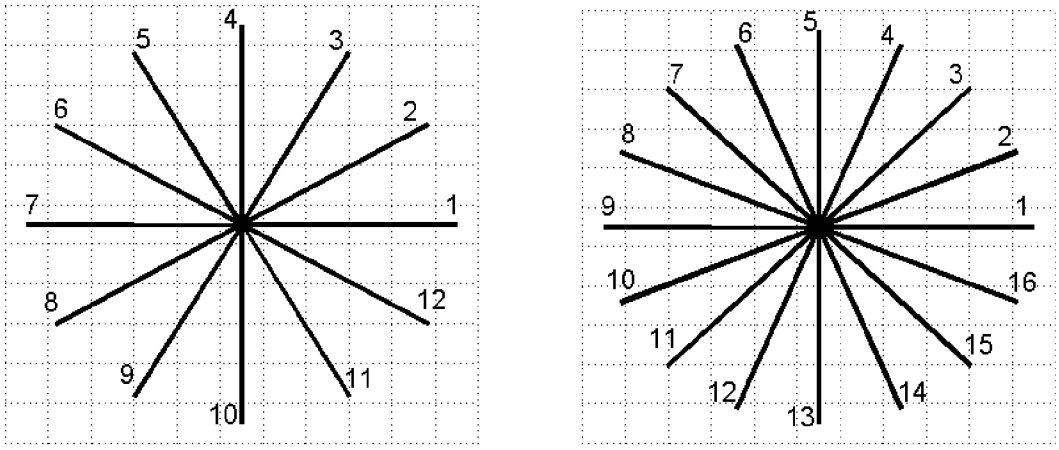
$a_{N_2-k+1,J_{N_2-k+1},1} := -a_{k,J_k,1}$

$a_{N_2-k+1,J_{N_2-k+1},2} := a_{k,J_k,2}$

end for

Pozn. Bylo použito stejné zkrácení zápisu jako v algoritmu 7.

- c) Pro počet směrů, jež je dělitelný čtyřmi, pak kromě předchozích zrcadlení podle os $y = 0$ a $x = 0$ bude platit i zrcadlení podle osy $x = y$. Viz obrázek 12.



(a) Souměrnost $N = 12$ směrů. (b) Souměrnost $N = 16$ směrů.

Obrázek 12: Souměrnost 12 a 16 směrů.

Analogicky jako v předchozích případech lze pro N dělitelné čtyřmi pro směr N_4 odvodit:

$$N \text{ – dělitelné čtyřmi } \implies N_4 = \frac{N}{4} + 1,$$

$$\phi_{N_4} = \left(\frac{N}{4} + 1 - 1\right)\frac{2\pi}{N} = \frac{\pi}{2}. \tag{5.7}$$

Pro N dělitelné čtyřmi je tedy směr N_4 vždy dán úhlem $\frac{\pi}{2}$ a je tedy zrcadlením směru 1 podle osy $x = y$.

Dále se soustřeďme pouze na směry $i = 1, \ldots, N_4$. Zbývající směry potom lze určit už zmíněnými zrcadleními. Postupujme obdobně jako v předchozích případech.

Označme $N_8 = N \div 8 + 1$.

Obecně bude platit, že směr $N_4 - k + 1$ je zrcadlením směru k pro $k = 1, \ldots, N_4 - N_8$.

Tvrzení lze dokázat přímo, a to zvlášť pro N dělitelné čtyřmi ale nedělitelné osmi a zvlášť pro N dělitelné osmi. (Důvod je opět analogický jako v předchozích zrcadleních. Pro N dělitelné osmi platí $\phi_{N_8} = \frac{\pi}{4}$, což je směr ležící přímo na ose $x = y$.)

Opět uveďme oba příklady. Pro $N = 12$ (obrázek 12a) máme $N_4 = 4$, $N_8 = 2$. Dostáváme tedy $k = 1, 2$ a platí, že směry $4, 3$ jsou zrcadlení směrů $1, 2$ respektive. Dle očekávání.

Pro $N = 16$ (obrázek 12b) bychom dostali $N_4 = 5$, $N_8 = 3$. Měli bychom $k = 1, 2$, tedy směry $5, 4$ jako zrcadlení směrů $1, 2$ respektive. Opět dle očekávání.

Zjednodušeně řečeno se v tomto zrcadlení vymění hodnoty $a_{i,j,1}$ a $a_{i,j,2}$. Zbývající prvky zůstanou zachovány. Pro $k = 1, \ldots, N_4 - N_8$ tedy platí:

$$\left.\begin{aligned} c_{N_4-k+1,j} &= c_{k,j} \\ g_{N_4-k+1,j} &= g_{k,j} \\ a_{N_4-k+1,j,1} &= a_{k,j,2} \\ a_{N_4-k+1,j,2} &= a_{k,j,1} \end{aligned}\right\} j \in J_{N_4-k+1} = J_k$$

Pomocí algoritmů 1 až 5 tak postačí určit prvky J_i, $c_{i,j}$, $g_{i,j}$, $a_{i,j,1}$, $a_{i,j,2}$ pouze ve směrech $i = 1, \ldots, N_8$, směry $i = N_8 + 1, \ldots, N_4$ budou určeny novým algoritmem 9. Zbývající směry následně algoritmy 8 a 7.

Algoritmus 9 Zrcadlení podle osy $x = y$

for $k := 1$ **to** $N_4 - N_8$ **do**
  $J_{N_4-k+1} := J_k$
  $c_{N_4-k+1,0} := 0$
  $c_{N_4-k+1,J_{N_4-k+1}} := c_{k,J_k}$
  $g_{N_4-k+1,J_{N_4-k+1}} := g_{k,J_k}$
  $a_{N_4-k+1,J_{N_4-k+1},1} := a_{k,J_k,2}$
  $a_{N_4-k+1,J_{N_4-k+1},2} := a_{k,J_k,1}$
end for

Pozn. Opět bylo použito stejné zkrácení zápisu jako v algoritmech 7 a 8.

Dodejme, že další zrcadlení již není možné. Čtvercová síť pixelů to zřejmě neumožňuje.

5.2.7. Shrnutí

Máme tedy odvozen adaptivní filtr, a to ve tvaru 5.6:

$$\tilde{F}(\overline{x}, \overline{y}) = \frac{1}{V(\overline{x}, \overline{y})} \frac{1}{G} \sum_{i=1}^{N} \left[v(\overline{x}, \overline{y}, \phi_i) \sum_{j \in J_i} F(\overline{x} + a_{i,j,1},\ \overline{y} + a_{i,j,2})\ g_{i,j} \right].$$

V tomto tvaru pak filtr zachovává průměrnou hodnotu obrazové funkce.

Nyní k samotnému algoritmu. Nejprve máme samozřejmě vstup v podobě konkrétních N, D a σ.

Snadno si můžeme všimnout, že námi odvozené hodnoty $c_{i,j}$, $g_{i,j}$, G, $a_{i,j,1}$ a $a_{i,j,2}$ nezávisí na konkrétním $(\overline{x}, \overline{y})$. Je proto výhodné si tyto hodnoty předpočítat. K tomu jsou potřeba i hodnoty S_k (podkapitola 5.2.3). U těch si můžeme všimnout, že nezávisí ani na indexech i, j (tedy daném směru a pořadí v tomto směru). Je tedy výhodné určit je jako první.

1. Zadej N, D a σ
   Algoritmus 3 – určení S_k

Nyní již musíme začít pracovat s jednotlivými směry (indexem i). Využívejme zrcadlení a určeme hodnoty $c_{i,j}$, $a_{i,j,1}$, $a_{i,j,2}$, $g_{i,j}$ a G (G stačí určit pouze jednou).

---

2. $N_2 := N \div 2 + 1$
   $N_4 := N \div 4 + 1$
   $N_8 := N \div 8 + 1$

   for $i := 1$ **to** N_8 **do**
   Algoritmy 1, 2 a 4 (vhodně skloubené) – určení $c_{i,j}$, J_i, $(a_{i,j,1},\ a_{i,j,2})$, $g_{i,j}$ a G

   if (N dělitelné 4) **then**
   Algoritmus 9 – zrcadlení podle osy $x = y$
   else
   for $i := N_8 + 1$ **to** N_4 **do**
   Algoritmy 1, 2 a 4 – určení $c_{i,j}$, J_i, $(a_{i,j,1},\ a_{i,j,2})$, $g_{i,j}$ a G

   if (N dělitelné 2) **then**
   Algoritmus 8 – zrcadlení podle osy $x = 0$
   else
   for $i := N_4 + 1$ **to** N_2 **do**
   Algoritmy 1, 2 a 4 – určení $c_{i,j}$, J_i, $(a_{i,j,1},\ a_{i,j,2})$, $g_{i,j}$ a G

   Algoritmus 7 – zrcadlení podle osy $y = 0$

Tímto jsme si předpočítali veškeré náležitosti našeho adaptivního filtru nezávislé na konkrétním $(\overline{x}, \overline{y})$. Nyní tedy přejděme k postupnému procházení všech bodů obrazové matice našeho obrazu. Označme:

I – původní obraz, obdélníkového tvaru, složen z dílčích pixelů,
F – obrazová matice obrazu I,
X – interval všech obrazových souřadnic $\overline{x}$ obrazu I,
Y – interval všech obrazových souřadnic $\overline{y}$ obrazu I,

$\tilde{I}$ – nový obraz, vzniklý daným adaptivním filtrováním obrazu I,
$\tilde{F}$ – obrazová matice obrazu $\tilde{I}$,
$\tilde{X}$ – interval všech obrazových souřadnic $\overline{x}$ obrazu $\tilde{I}$,
$\tilde{Y}$ – interval všech obrazových souřadnic $\overline{y}$ obrazu $\tilde{I}$.

Nejprve je třeba si říct, že výsledný obraz $\tilde{I}$ bude mít menší rozměr než původní obraz I. Definujme D_{int} jako nahoru na celá čísla zaokrouhlené číslo $D - \frac{1}{2}$. Číslo D_{int} zřejmě určuje počet pixelů (vyjma samotného pixelu $(\overline{x}, \overline{y})$), do kterých zasáhnou směry dané úhly $0, \frac{\pi}{2}, \pi$ a $\frac{3\pi}{2}$. Aby tedy bylo možné pro libovolný počet směrů N určit uvedeným

způsobem hodnotu $\tilde{F}(\overline{x}, \overline{y})$, je nutné, aby množiny $\tilde{X}$ a $\tilde{Y}$ byly „z obou stran o D_{int} menší“. Přesněji řečeno, označíme-li:

$$x_{min} = \min X, \qquad y_{min} = \min Y,$$
$$x_{max} = \max X, \qquad y_{max} = \max Y,$$

pak pro množiny $\tilde{X}, \tilde{Y}$ bude platit:

$$\min \tilde{X} = x_{min} + D_{int}, \qquad \min \tilde{Y} = y_{min} + D_{int},$$
$$\max \tilde{X} = x_{max} - D_{int}, \qquad \max \tilde{Y} = y_{max} - D_{int}.$$

Teď tedy začněme postupně procházet všechny body $(\overline{x}, \overline{y})$, $\overline{x} \in \tilde{X}$, $\overline{y} \in \tilde{Y}$. Pro každý tento bod postupně projdeme všechny směry $i = 1, \ldots, N$. Pro daný směr i odečteme hodnoty $F(\overline{x} + a_{i,j,1},\ \overline{y} + a_{i,j,2})$ pro všechna $j \in J_i$ a spočítáme jejich rozptyl, jak je uvedeno v podkapitole 5.2.5. Potom, co toto pro daný bod provedeme ve všech směrech, již můžeme určit hodnotu $\tilde{F}(\overline{x}, \overline{y})$. Takto projdeme všechny body $(\overline{x}, \overline{y})$.

---

3. **for all** $\overline{x} \in \tilde{X}$ **do**
  for all $\overline{y} \in \tilde{Y}$ **do**
    $\tilde{F}(\overline{x}, \overline{y}) := 0$
    $V(\overline{x}, \overline{y}) := 0$
    for $i := 1$ **to** N **do**
      $\tilde{F}_P := 0$
      for all $j \in J_i$ **do**
        $\tilde{F}_P := \tilde{F}_P + f(\overline{x} + a_{i,j,1},\ \overline{y} + a_{i,j,2})\ g_{i,j}$
      Algoritmus 5 – výpočet $\mu(\overline{x}, \overline{y}, \phi_i)$
      Algoritmus 6 – výpočet $s^2(\overline{x}, \overline{y}, \phi_i)$
      $v(\overline{x}, \overline{y}, \phi_i) := \frac{1}{s^2(\overline{x}, \overline{y}, \phi_i)}$
      $V(\overline{x}, \overline{y}) := V(\overline{x}, \overline{y}) + v(\overline{x}, \overline{y}, \phi_i)$
      $\tilde{F}(\overline{x}, \overline{y}) := \tilde{F}(\overline{x}, \overline{y}) + v(\overline{x}, \overline{y}, \phi_i)\ \tilde{F}_P$
    end for
    $\tilde{F}(\overline{x}, \overline{y}) := \frac{1}{G}\ \frac{1}{V(\overline{x}, \overline{y})}\ \tilde{F}(\overline{x}, \overline{y})$
  end for
end for

Představili jsme tedy algoritmus pro provedení adaptivního filtrování aditivního šumu v digitálním obraze. Závěrem ještě zmiňme několik věcí spíše technického rázu:

- Jak již bylo zmíněno v kapitole 2, v praxi pixely obrazové matice nabývají hodnot $F(\overline{x}, \overline{y}) \in \langle 0, 255 \rangle \subset \mathbb{N}$. My jsme dosud pracovali s $F(\overline{x}, \overline{y}) \in \mathbb{R}$. Toto zřejmě vyřešíme zaokrouhlováním.

  Pozn. Vzhledem k tomu, že vstupem do adaptivního filtru je také obrazová matice, a vzhledem k tomu, jak jsme definovali náš adaptivní filtr, je nemožné tímto filtrem docílit hodnoty nižší než 0. Teoreticky by pak bylo možné docílit hodnoty vyšší než 255 (navíc o tolik vyšší, aby se zaokrouhlila nahoru), je to ale velmi nepravděpodobné. Přesto je v programu dáno, že libovolná hodnota vyšší než 255 se nastaví na 255.

- Při výpočtu směrových vah se používá formule:

$$v(\overline{x},\overline{y},\phi_i) := \frac{1}{s^2(\overline{x},\overline{y},\phi_i)},$$

v níž může zřejmě dojít k dělení nulou. Prakticky je toto vyřešeno takto:

$$v(\overline{x},\overline{y},\phi_i) := \frac{1}{s^2(\overline{x},\overline{y},\phi_i) + 10^{-6}}. \tag{5.8}$$

Tento algoritmus potom byl (s jistými drobnými odchylkami) použit v aplikaci, jež je součástí diplomové práce. Aplikace byla vytvořena v programovacím prostředí Delphi. Tento algoritmus využívá třídu „TDirections“, jež obsahuje všechny zmíněné prvky nezávislé na konkrétních hodnotách $F(\overline{x},\overline{y})$, které se vypočítají hned v jejím konstruktoru. Pomocí nich se následně provádí filtrace na konkrétním obraze. Algoritmus je dostupný v záložce „Filtrování“ pod příkazem „Adaptivní filtrování“. Prvním spuštěním si program vyžádá zadání parametrů N, D a σ. Objekt „Directions“ jako instance třídy „TDirections“ potom zůstává v paměti pro každé další adaptivní filtrování. Změna parametrů je možná pouze příkazem „Zvol parametry“.

Zmiňme se ještě o rychlosti počítání adaptivního filtrování ve zmíněném programu. Předpočítání hodnot nezávislých na konkrétních $(\overline{x},\overline{y})$ trvá velmi krátce, řádově kolem sekundy (na použitém stroji). Problémem ale je procházení všech $(\overline{x},\overline{y})$ daného obrazu. Rychlost počítání pak je velmi silně závislá na velikosti filtrovaného obrazu.

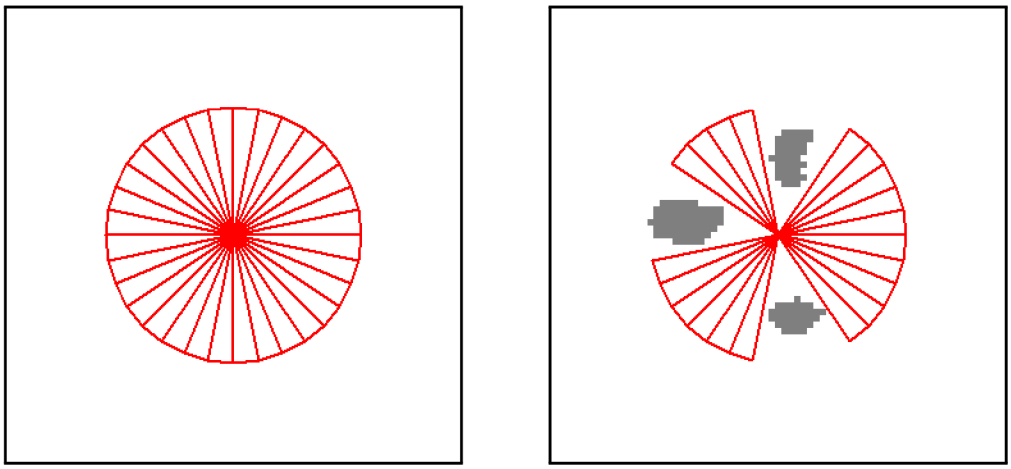
6. Chování směrových vah

Podívejme se nyní, jak se chovají směrové váhy. Toto chování demonstrujme na příkladech.

Zvolme počet směrů $N = 32$. V aplikaci je pak pod záložkou „Filtrování“ příkaz „Ukaž váhy směrů ad. filtru“. Pohybováním myší po nějakém obrázku, potom uvidíme 25-ti násobný zoom (jeden pixel je zobrazen na plochu 5×5 pixelů) výřezu tohoto obrázku. Jasové hodnoty jsou v tomto výřezu transformovány do intervalu $\langle 128, 255 \rangle$. Ve výřezu v daném bodě obrazové matice jsou vyobrazeny váhy adaptivního filtru, který je momentálně v objektu „Directions“. Vyobrazení je nastaveno tak, že největší váha má vždy stejnou délku, a to 100. Při nastavení vzdálenosti adaptivního filtru $D = 20$, potom nejdelší reprezentace vykreslených vah zasáhne do obrazových bodů, které skutečně náleží do daného směru. Mezi váhami v jednotlivých směrech je pak vykreslena lomená čára.

6.1. Jednoduché obrázky

Podívejme se nejprve na chování vah na jednoduchých obrázcích, které mají velmi omezené množství hladin jasu. Je-li okolí pozorovaného bodu zcela homogenní (obrázek 13a), vytvoří lomená čára mezi váhami pravidelný N-úhelník. Budou-li se na homogenním pozadí vyskytovat nějaké další objekty (obrázek 13b), budou se jim vykreslené reprezentace vah „vyhýbat“. To je způsobeno tím, že nenormovaná váha směru, který zasahuje pouze do pixelů se stejnou hodnotou jasu, je dána podílem $\frac{1}{10^{-6}} = 10^6$ – jelikož má nulový rozptyl a bylo zavedeno 5.8. Taková váha je pak mnohonásobně větší, než váhy směrů, které zasahují do pixelů, jejichž jasové hodnoty se různí.



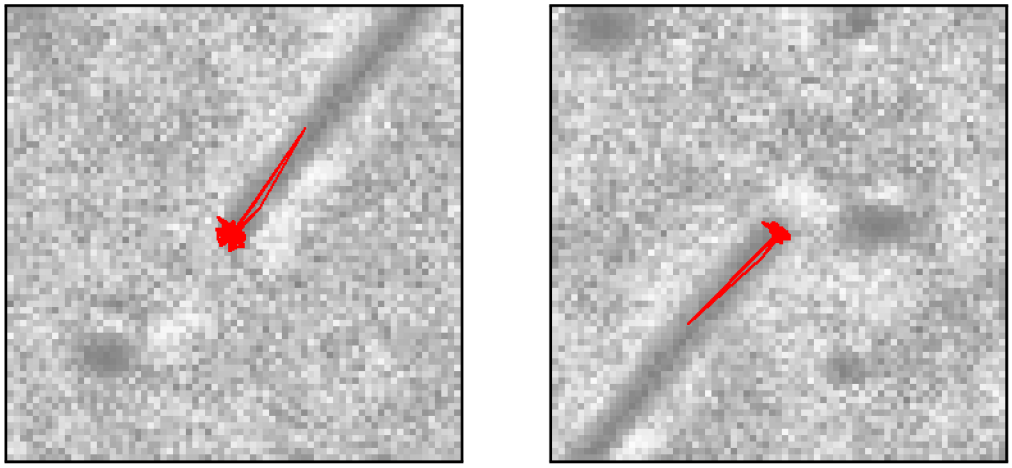
(a) (b)

Obrázek 13: Chování směrových vah na jednoduchých obrázcích.

6.2. Snímky skleněných vláken

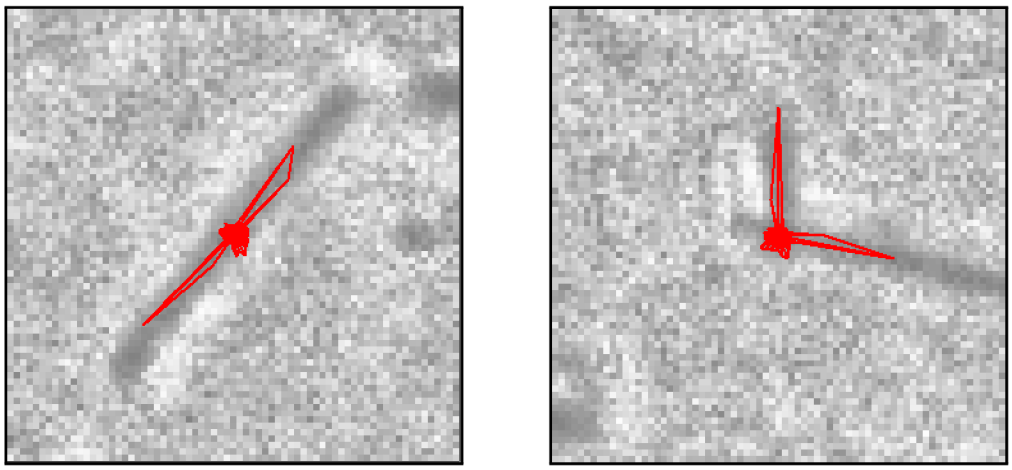
Důležitější pro nás je, jak se směrové váhy budou chovat ve studovaných snímcích skleněných vláken. Podívejme se na obrázek 6. V tomto obrázku se nachází 3 studovaná vlákna. Ukažme si nyní, jak se budou směrové váhy chovat na těchto vláknech.

Začněme s vláknem na obrázku 6 vlevo dole. Na obrázku 14 můžeme vidět, jak se váhy chovají v okrajích tohoto vlákna. Na obrázku 15a pak můžeme vidět chování směrových vah uvnitř vlákna. Zřejmě největší váhu mají směry, ve kterých se nachází vlákno. Za povšimnutí stojí také vlákno (nebo spíše dvě vlákna) v obrázku vpravo nahoře. Situace je na obrázku 15b.



(a) (b)

Obrázek 14: Chování směrových vah ve vláknech.

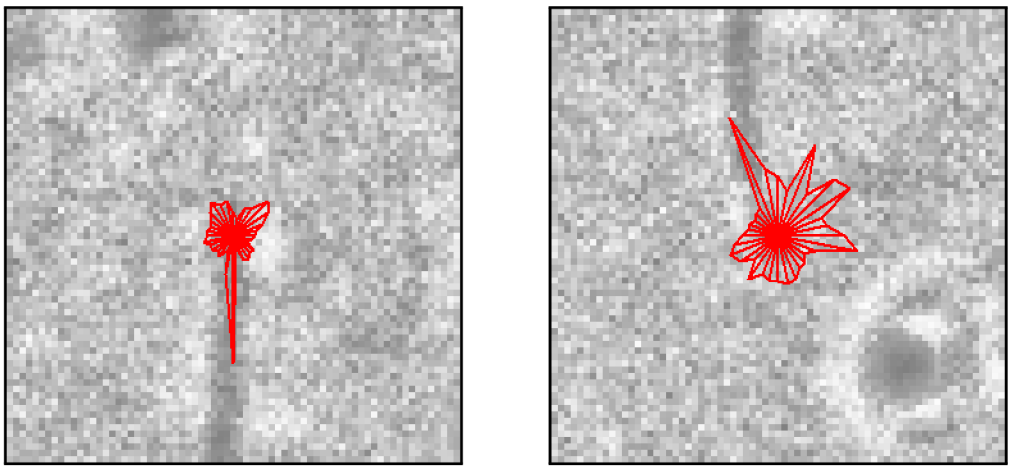


(a) (b)

Obrázek 15: Chování směrových vah ve vláknech.

Za zmínku stojí i poslední třetí vlákno v obrázku 6 vlevo nahoře. Podívejme se na váhy v jeho okrajích. Váhy na horním okraji vlákna (obrázek 16a) jsou v pořádku – váha ve směru vlákna je jasně dominantní. Na dolním okraji (obrázek 16b) je už ale situace složitější. Ačkoliv je váha ve směru vlákna stále největší, je poměr mezi ní a ostatními

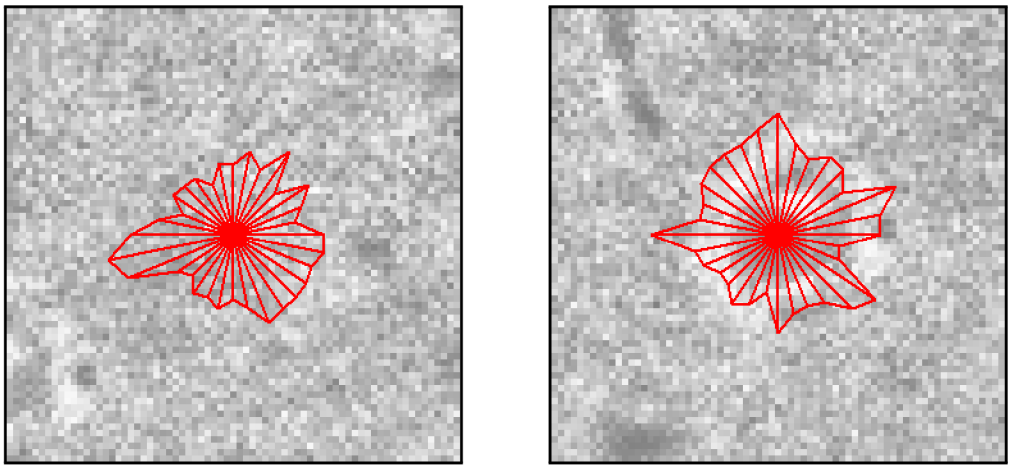
váhami srovnatelný. To je zřejmě způsobeno tím, že dolní okraj tohoto vlákna je od zbytku vlákna částečně „oddělen“. Se zpracováním tohoto vlákna se dají očekávat problémy.



(a) (b)

Obrázek 16: Chování směrových vah ve vláknech.

Nakonec se nezapomeňme podívat na chování vah v „prázdné“ části obrazu (obrázek 17a), popřípadě v jednom z kruhových útvarů (obrázek 17b). V obou případech jsou váhy v různých směrech srovnatelné, a proto se v těchto bodech bude adaptivní filtr chovat do jisté míry jako filtr lineární.



(a) (b)

Obrázek 17: Chování směrových mimo vlákna.

Chování směrových vah adaptivního filtru naznačuje, že filtr bude ideální pro řešení problematiky dané v kapitole 3.

7. Filtrování obrazu

Podívejme se teď na samotné filtrování obrazu. Nejprve zdůrazněme, že při vhodné volbě parametrů očekáváme, že histogram adaptivně filtrovaného obrazu dostane bimodální podobu, a bude tedy vhodný pro segmentaci pomocí prahování.

7.1. Výběr nejlepších parametrů adaptivního filtru

Nejprve se soustřeďme na výběr nejlepších parametrů adaptivního filtru. Za tímto účelem jsem do programu přidal příkaz „Sekvence filtrování“. Tam je možno nastavit počáteční hodnoty všech tří parametrů N, D a σ, dále velikost kroků, o které se budou parametry zvětšovat, a nakonec samozřejmě i stop hodnoty. Jsou pak vytvořeny filtrované obrazy podle všech kombinací těchto parametrů. Současně jsem zavedl hodnotu, kterou jsem nazval bimodalita.

Označme u_{max} hladinu jasu, v níž histogram H nabývá svého maxima, tj. platí:

$$H(u_{max}) = \max_{\forall u} H(u).$$

Dále označme:

$$U_{max} = \langle 0, u_{max} \rangle .$$

Pro konkrétní hodnotu $u \in U_{max}$ potom označme:

$$\begin{aligned} U_L(u) &= \langle 0, u \rangle , \\ U_R(u) &= \langle u, u_{max} \rangle , \\ L_{max}(u) &= \max_{t \in U_L(u)} H(t), \\ R_{min}(u) &= \min_{t \in U_R(u)} H(t). \end{aligned}$$

Bimodalitou b pak označme hodnotu:

$$b = \max_{u \in U_{max}} (L_{max}(u) - R_{min}(u)).$$

Řečeno zjednodušeně postupně procházíme histogram všemi hladinami až do hladiny u_{max}, v níž histogram nabývá svého maxima. Pro každou tuto hladinu určíme největší hodnotu histogramu nalevo a nejmenší hodnotu histogramu napravo (pouze do hladiny u_{max}) a spočítáme jejich rozdíl. Největší tento rozdíl je bimodalita b. V programu jsem navíc místo hodnoty u_{max} použil hodnotu u takovou, kde hodnota histogramu poprvé přerostla $\frac{H(u_{max})}{4}$. Bimodalita, která nás zajímá, se totiž obecně nachází mezi nižšími četnostmi. Označíme-li tuto hladinu $u_{max/4}$ redefinujeme potom intervaly U_{max} a $U_R(u)$ jako:

$$\begin{aligned} U_{max} &= \left\langle 0, u_{max/4} \right\rangle , \\ U_R(u) &= \left\langle u, u_{max/4} \right\rangle . \end{aligned}$$

Vzhledem k povaze histogramů (skokové rozdíly), funguje tato bimodalita pouze orientačně. Pro malé obrazy je zcela nefunkční, jelikož skokové rozdíly v histogramu zcela

převáží nad jeho celkovým průběhem. Svůj smysl má až pro velké množství (řádově desítky a více) dostatečně velkých obrazů. Tam je možné tuto bimodalitu využít k orientaci mezi jednotlivými obrazy.

Dále jsem tedy prováděl adaptivní filtrování obrazu 4 (i dalších obrazů) mnoha různými adaptivními filtry. Používal jsem příkaz „Sekvence filtrování“ nejprve s velkým krokem a velkým rozdílem počátečních a koncových hodnot. Výběrem nejlepší skupiny výsledků (mezi nimiž jsem se orientoval pomocí zmíněné bimodality) jsem pak rozdíl počátečních a koncových hodnot postupně zmenšoval. Současně s tím jsem zmenšoval i délku kroku. Sledoval jsem vliv jednotlivých parametrů na výsledek filtrace a snažil se určit jejich optimální hodnoty.

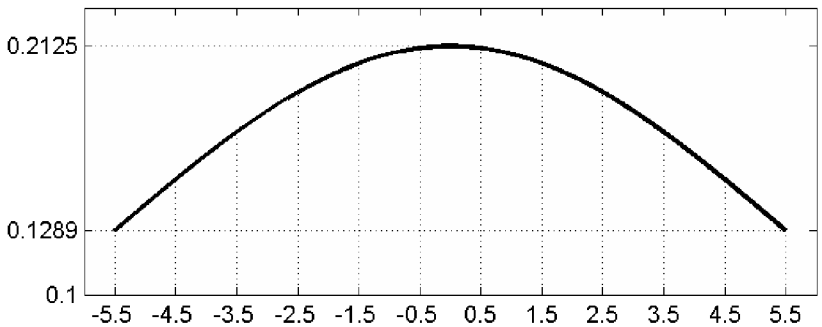
Ohledně vlivu parametrů na výsledek filtrace jsem dospěl k následujícím závěrům (jedná se pouze o mé domněnky):

- Vzdálenost D:
  Nejvýznamnější parametr. Zásadně ovlivňuje výsledek adaptivní filtrace. Konkrétně k obrazu 4 jsem její ideální hodnotu určil na 5,5. Ideální hodnota však závisí na konkrétním obraze. K diplomové práci je přiloženo několik obrazů a také seznam určených ideálních hodnot parametrů jejich adaptivní filtrace.

- σ z Gaussovy funkce:
  Významně ovlivňuje výsledek filtrace. Jeho ideální hodnota se odvíjí od vzdálenosti. Ideální hodnoty σ jsou často přibližně rovny hodnotě D. Konkrétně pro obraz 4 jsem ideální hodnotu určil na 5,5 – tedy stejnou hodnotu jako D.

- Počet směrů N:
  Pokud je směrů dostatečné množství mají na výsledek filtrace zanedbatelný vliv. Poměrně významně však ovlivňují délku výpočtu. Dá se soudit, že se zvětšováním ideální hodnoty D bude potřeba zvětšovat i N. Mé vzdálenosti D však byly ve všech případech poměrně nízké. Proto jsem jednotně zavedl $N = 16$ jako kompromis mezi kvalitou filtrace a délkou jejího výpočtu.

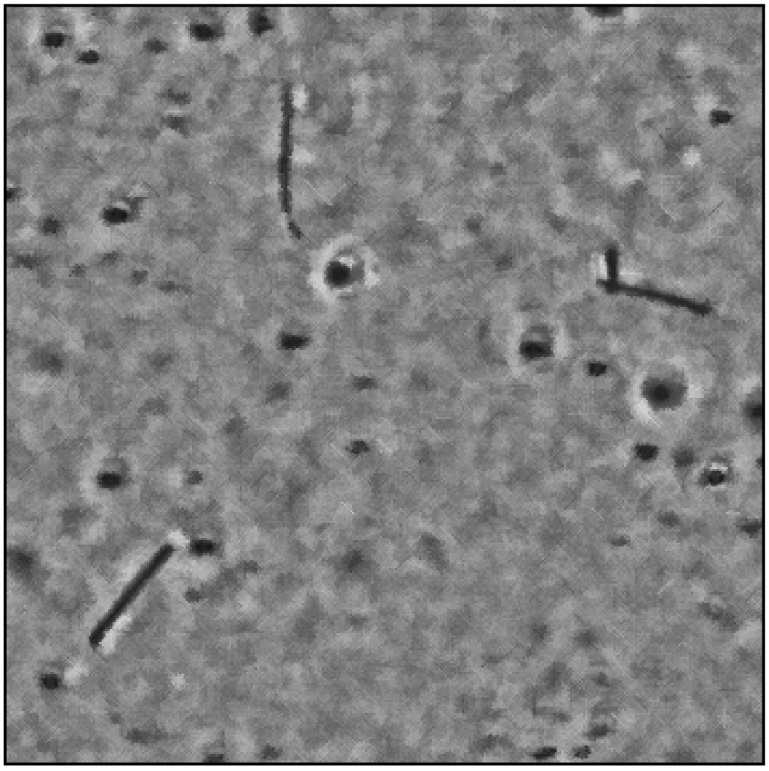
Řekněme si něco k určeným hodnotám $D = 5{,}5$ a $\sigma = 5{,}5$. Na obrázku 18 se podívejme na tvar normované Gaussovy funkce dané těmito parametry (funkce je normovaná na intervalu $\langle 0, D\rangle \subset \mathbb{R}$, na intervalu $\langle -D, D\rangle$ tedy její integrál bude roven 2). Jak je z obrázku patrné, tato funkce přikládá i vzdáleným bodům poměrně vysoké váhy.



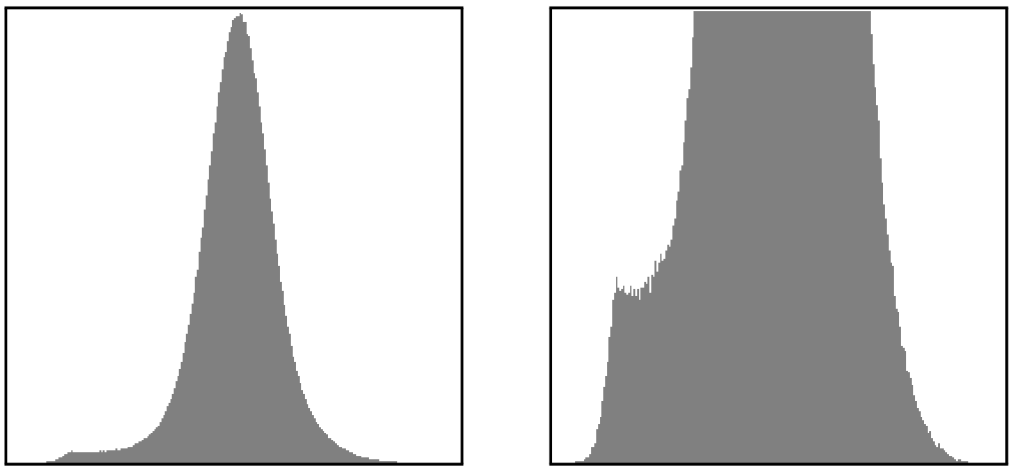
Obrázek 18: Tvar Gaussovy funkce (normované na $\langle 0, D\rangle$) pro $D = 5{,}5$, $\sigma = 5{,}5$.

7.2. Filtrování obrazu adaptivním filtrem

Obraz 4 jsem tedy filtroval pomocí parametrů $N = 16$, $D = 5{,}5$ a $\sigma = 5{,}5$. Výřez z výsledného obrazu potom můžeme vidět na obrázku 19, histogram na obrázku 20. Porovnejme s obrázky 6 a 5a. Jak je z obrázků patrné, histogram už dostává bimodální podobu, zatím ale situace není zcela ideální.



Obrázek 19: Výřez snímku po adaptivním filtrování.

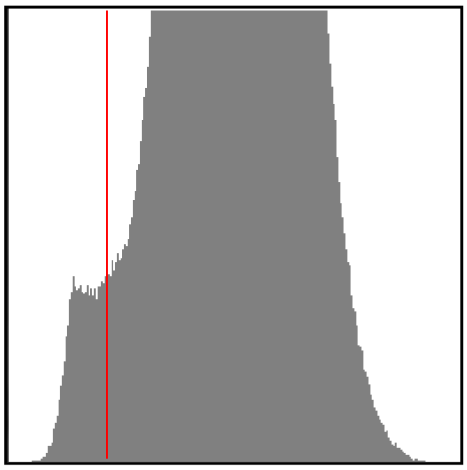


(a) Histogram snímku po adaptivním filtrování.

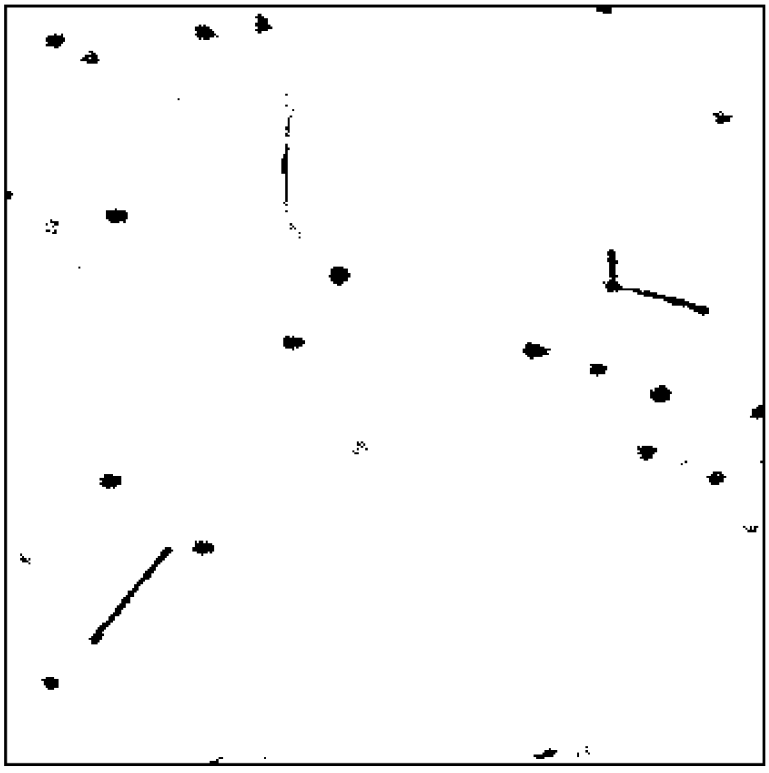
(b) Přiblížený histogram.

Obrázek 20: Histogram snímku po adaptivním filtrování.

Podívejme se, jak by pro tento obraz dopadlo prahování. Naznačený práh (byl položen do jednoho z lokálních minim) je na obrázku 21. Výsledek prahování na obrázku 22. Ačkoliv jsme významně potlačili šum, prahování nedopadlo ideálně, zejména pro vlákno vlevo nahoře (podotkněme, že by dopadlo mnohem lépe, kdybychom práh položili výše).



Obrázek 21: Histogram s naznačeným prahem 56.

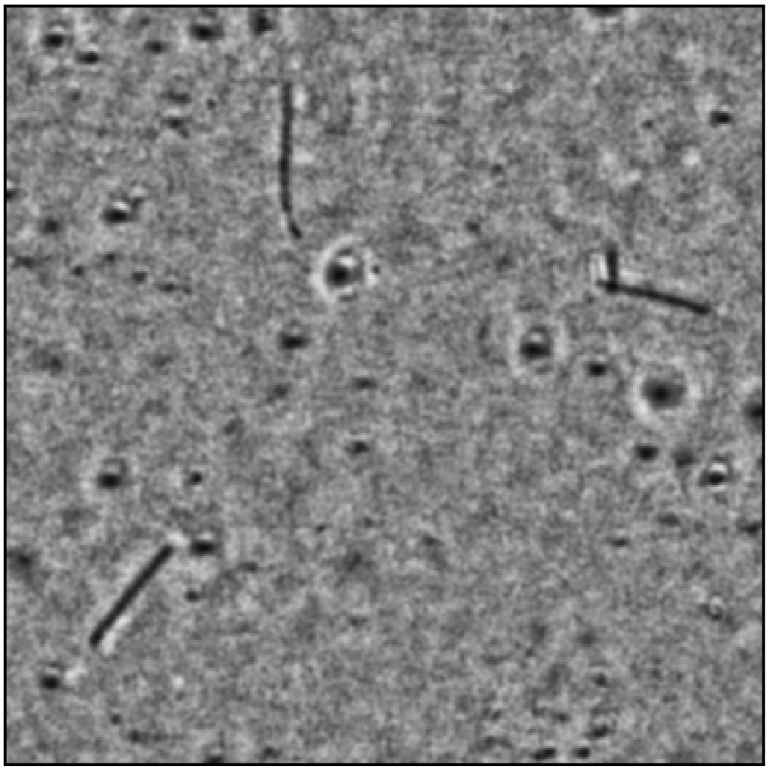


Obrázek 22: Prahování výřezu snímku po adaptivním filtrování.

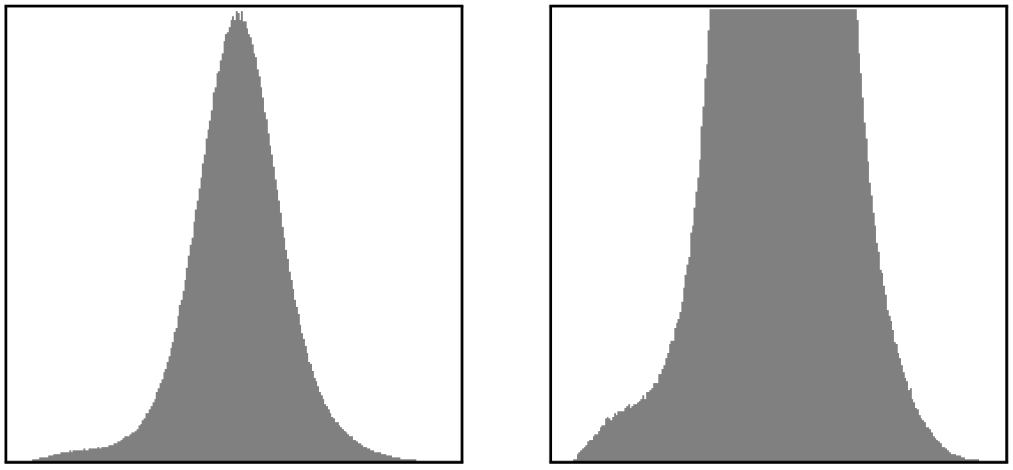
7.3. Porovnání s lineárním filtrem

Podobně jako pro adaptivní filtr, jsem se snažil vybrat nejlepší parametry i pro filtr lineární. Opět pomocí „Sekvence filtrování“. Výběr byl problematický. Lineární filtr byl velmi citlivý na parametry D i σ. Nakonec jsem ale vybral $D = 2$ a $\sigma = 1$. Vzdálenost

filtru byla zvolena takto nízká, protože se zvětšováním vzdálenosti dochází stále k většímu rozmazávání vláken. Obraz 4 jsem pak filtroval lineárním filtrem se zmíněnými parametry. Výřez výsledného obrazu je na obrázku 23. Histogram výsledného obrazu na obrázku 24.



Obrázek 23: Výřez snímku po lineárním filtrování.



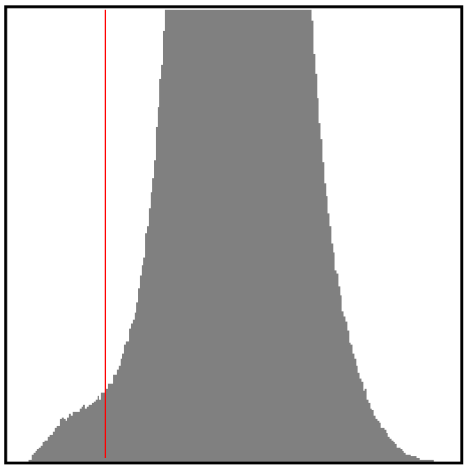
(a) Histogram snímku po lineárním filtrování.

(b) Přiblížený histogram.

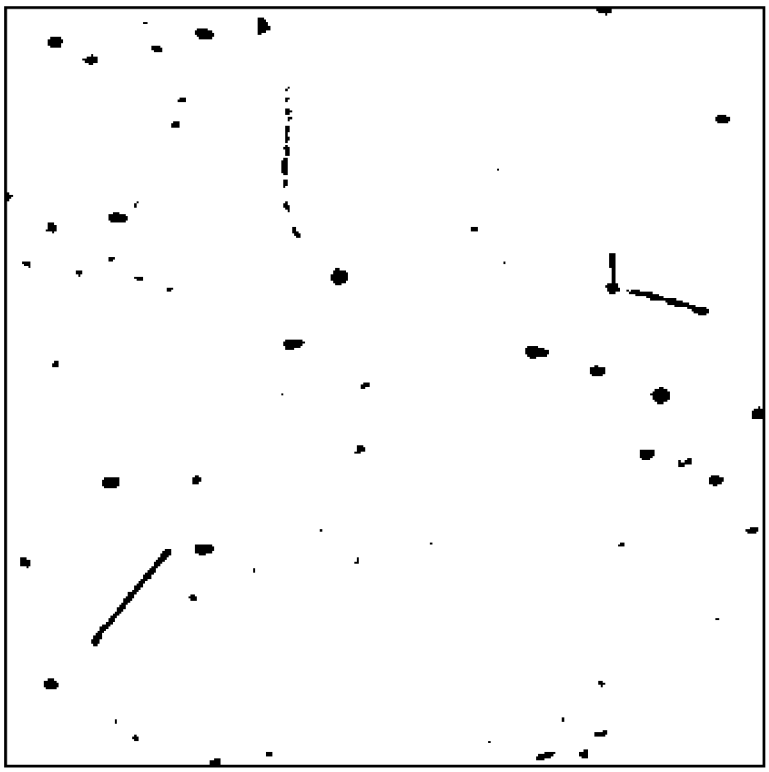
Obrázek 24: Histogram snímku po lineárním filtrování.

Obrázky porovnejme s výsledky adaptivního filtru. Všimněme si rozdílů mezi obrázky 19 a 23. Zaměřme se teď na histogramy. Pro lineární filtr histogram nedostal bimodální podobu. Při prahování by tak bylo problematické najít vhodnou hodnotu prahu.

Ukažme si, jak by dopadlo prahování se stejným prahem, jako při předchozím prahování po adaptivním filtrování. Naznačený práh je na obrázku 25. Výsledek prahování na obrázku 26. Porovnejme s 22. Výsledky jsou si podobné. Po lineárním filtru však v obraze zůstalo více objektů, které nás do budoucna nezajímají. Navíc nám adaptivní filtrování dá výhodu při hledání prahu. Podstatně větší výhodu nám však dá skutečnost, že adaptivní filtrování je možné provádět opakovaně se zlepšujícími se výsledky. Naproti tomu filtrování lineární, bude s každým dalším filtrováním zřejmě pouze více rozmazávat obraz.



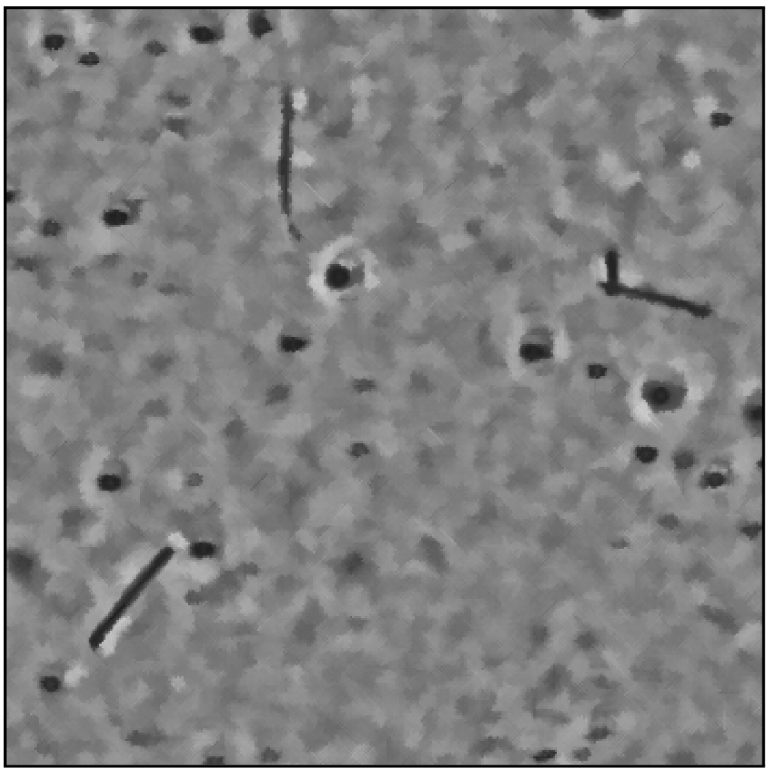
Obrázek 25: Histogram s naznačeným prahem 56.



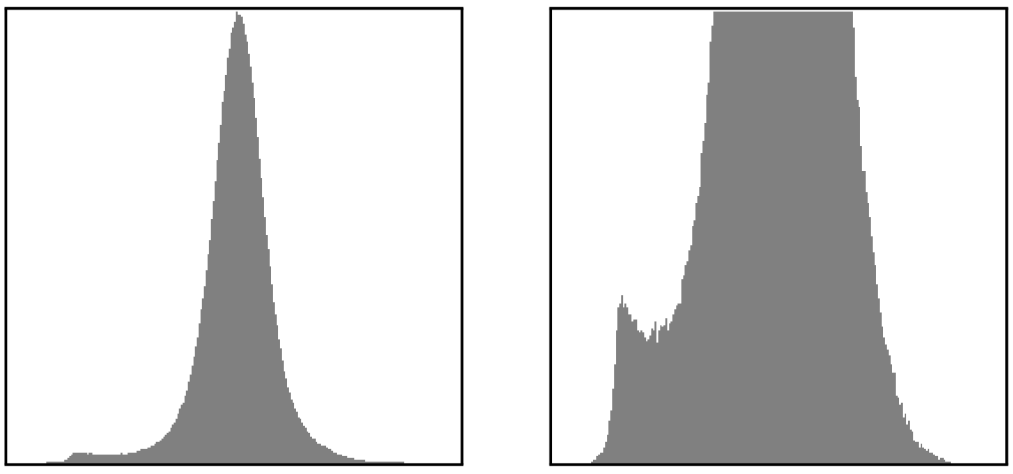
Obrázek 26: Prahování výřezu snímku po lineárním filtrování.

7.4. Opakované adaptivní filtrování

Vycházejme z adaptivně filtrovaného obrazu. Pro něj jsem se opět pokusil najít ideální parametry adaptivního filtru. Vzhledem k výsledkům jsem dospěl k závěru, že tato obměna není nutná. Parametry tedy zůstanou nastaveny na $N = 16$, $D = 5{,}5$ a $\sigma = 5{,}5$. Výřez výsledného obrazu je na obrázku 27, histogram na obrázku 28.



Obrázek 27: Výřez snímku po druhém adaptivním filtrování.

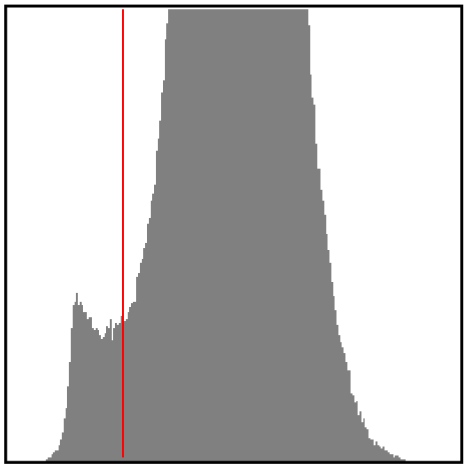


(a) Histogram snímku po druhém adaptivním filtrování.

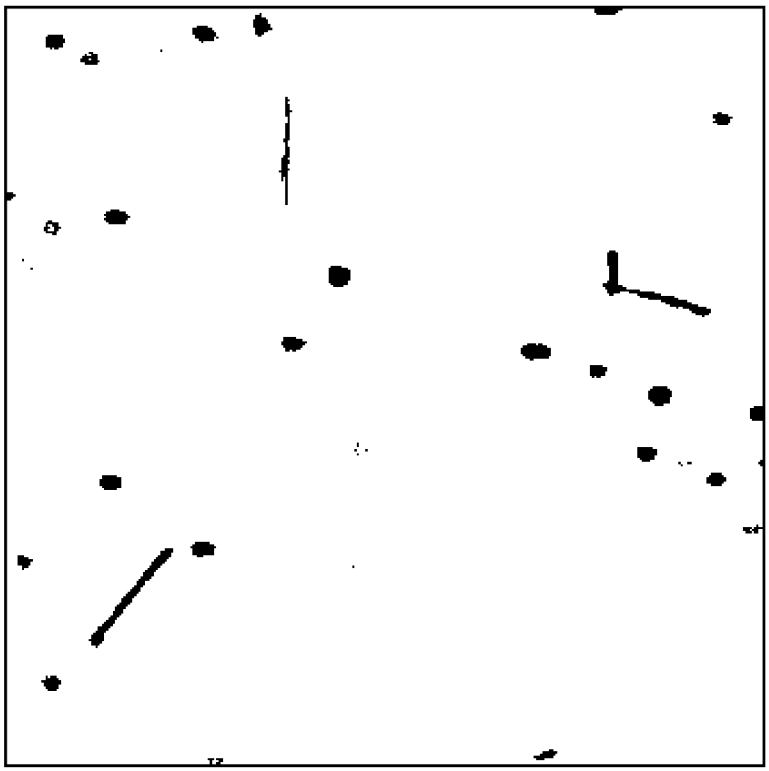
(b) Přiblížený histogram.

Obrázek 28: Histogram snímku po druhém adaptivním filtrování.

Jak z obrázků vidíme, histogram už získal bimodální tvar. Opět si ukažme, jak by dopadlo prahování. Práh opět položme do jednoho z lokálních minim. Naznačený práh je na obrázku 29. Výsledek prahování na obrázku 30.



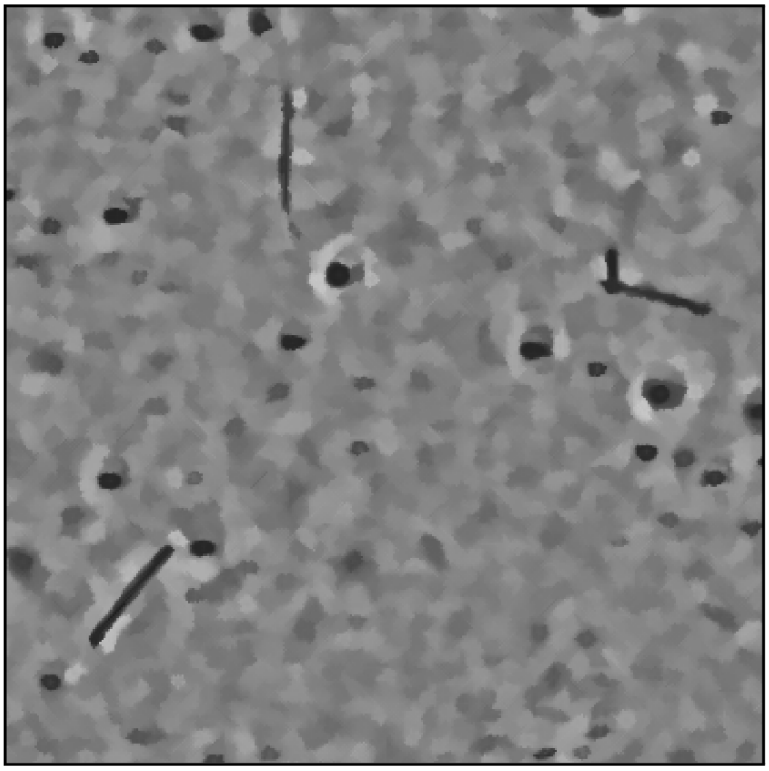
Obrázek 29: Histogram s naznačeným prahem 66.



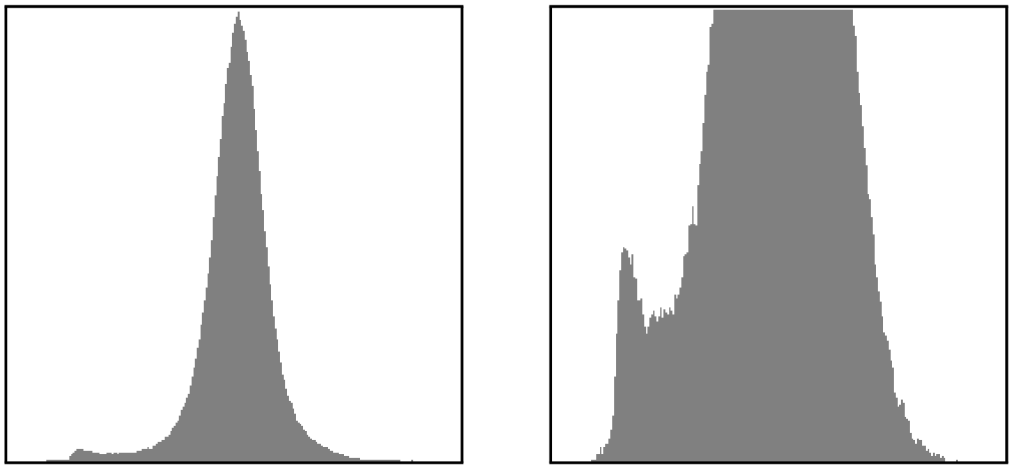
Obrázek 30: Prahování výřezu snímku po druhém adaptivním filtrování.

Vlákna vpravo nahoře a vlevo dole již vypadají velmi dobře. Vlákno vlevo nahoře vypadá ale problematicky.

Proveďme tedy adaptivní filtrování ještě potřetí. Testováním jsem opět dospěl k závěru, že parametry není třeba měnit. Opět se soustřeďme pouze na výřez výsledného obrazu, obrázek 31. Histogram výsledného obrazu je na obrázku 32.



Obrázek 31: Výřez snímku po třetím adaptivním filtrování.

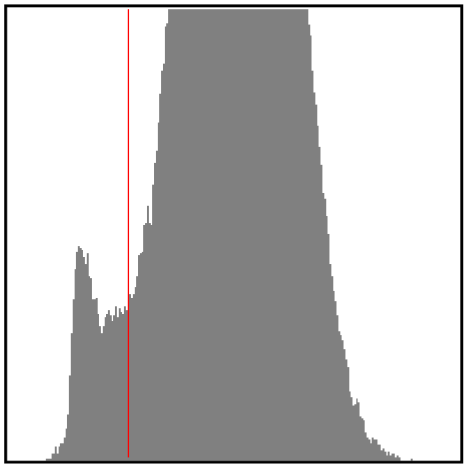


(a) Histogram snímku po třetím adaptivním filtrování.

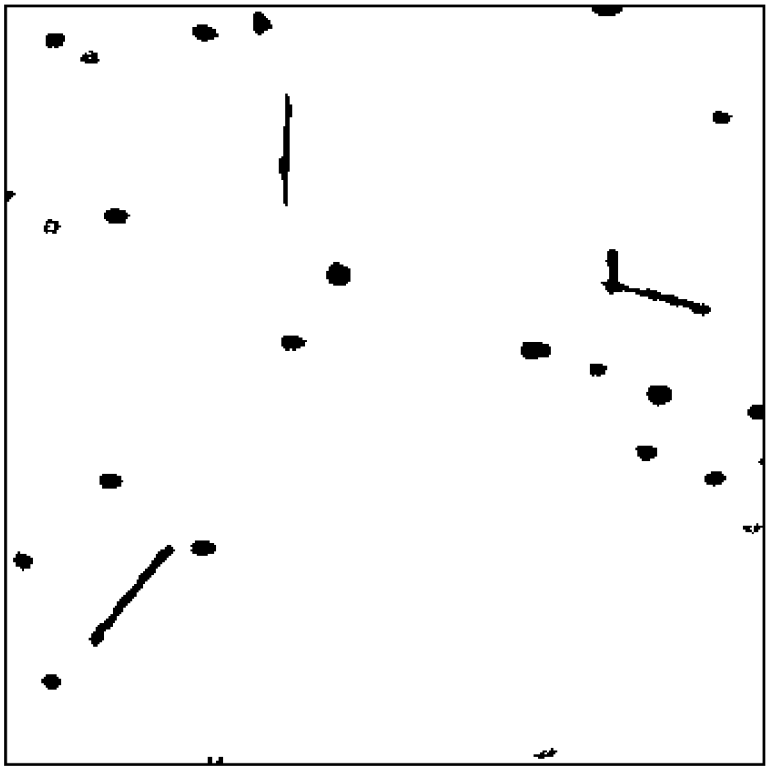
(b) Přiblížený histogram.

Obrázek 32: Histogram snímku po třetím adaptivním filtrování.

Z obrázků vidíme, že po třetím adaptivním filtrování, nabude histogram značně bimodálního tvaru. To je pro prahování zcela ideální. Ukažme si výsledek prahování. Práh opět položme do jednoho z lokálních minim. Naznačený práh je na obrázku 33. Výsledek prahování na obrázku 34.



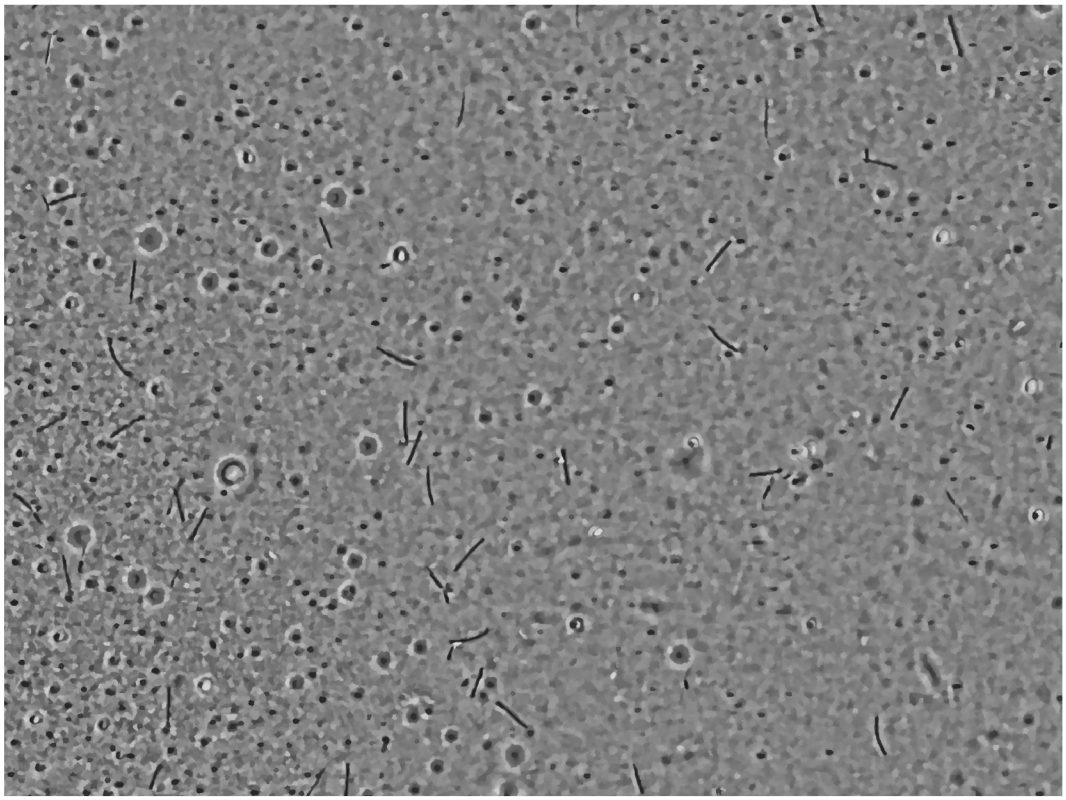
Obrázek 33: Histogram s naznačeným prahem 69.



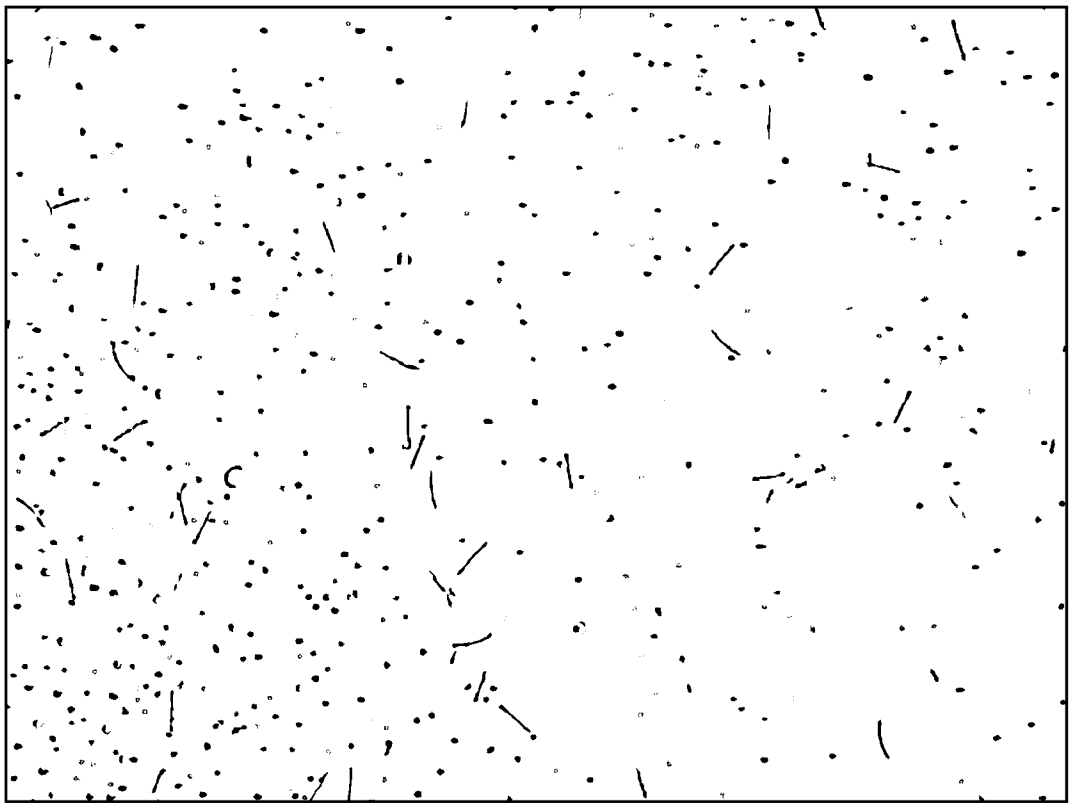
Obrázek 34: Prahování výřezu snímku po třetím adaptivním filtrování.

Nyní již všechna vlákna vypadají dobře. Pouze ve vláknu vlevo nahoře došlo ke „ztrátě“ její „oddělené“ části. To souvisí s problémem, který jsme zmínili v kapitole 6, konkrétně na obrázku 16b.

Za výsledek adaptivního filtrování obrazu 4 tak považujme obrázek 35. Celý tento obrázek prahovaný, s prahem vyznačeným na obrázku 33, pak je na obrázku 36. Toto prahování již vypadá dobře. Obraz je segmentován a připraven k dalšímu zpracování. Dalším krokem k separaci vláken, by bylo použití vhodného geometrického filtru. To už ale přesahuje cíle diplomové práce.



Obrázek 35: Snímek po třetím adaptivním filtrování, vždy při parametrech $N = 16$, $D = 5{,}5$ a $\sigma = 5{,}5$.



Obrázek 36: Prahování snímku po třetím adaptivním filtrování s prahem 69.

8. Závěr

Jedním z cílů diplomové práce bylo popsat metodu adaptivního filtru k potlačení aditivního šumu v obraze.

Metoda adaptivního filtru je pak v diplomové práci popsána více způsoby. Nejprve je to její popis na spojité obrazové funkci. Následuje její diskretizace, a tedy popis na obrazové matici. Metoda je algoritmizována a jsou představeny parametry adaptivního filtru N, D a σ. Následně je na příkladech popsáno chování směrových vah. Nakonec jsou uvedeny výsledky adaptivního filtrování konkrétního obrazu, stejně jako histogramy a prahování těchto výsledků. Adaptivní filtrace je porovnána s filtrací lineární. Daný obraz je zpracován do podoby, kdy bude možné pomocí geometrického filtru určit v tomto obraze požadované objekty.

Druhým cílem diplomové práce bylo vytvořit aplikaci, která využije zmíněnou metodu.

Aplikace byla vytvořena, a to v programovacím prostředí Delphi. Aplikace využívá objektového programování a kromě adaptivního filtru obsahuje i filtr lineární. Dále poskytuje možnost vytvářet histogramy a prahovat obrazy. Právě této aplikace bylo využito při zpracování obrazů adaptivním a lineárním filtrováním, při vytváření histogramů těchto obrazů a při jejich segmentaci pomocí prahování.

Popis software

Pro účely diplomové práce byla vytvořena aplikace v prostředí Delphi. Ve zkratce teď bude uveden popis jednotlivých součástí aplikace. Cílem je usnadnit případnou práci s touto aplikací.

- Soubor
  - Otevřít – Otevře obrázek. Podporovány jsou pouze soubory typu „bmp“ a „tif“.
  - Uložit – Uloží označený obrázek. Podporován je pouze typ „bmp“.
  - Konec – Ukončí běh programu.
- Filtrování
  - Adaptivní filtrování – Provede adaptivní filtraci označeného obrázku podle zadaných parametrů, které jsou viditelné v dolní liště. Pokud parametry nejsou zadány, bude požadovat jejich zadání.
  - Ukaž váhy směrů ad. filtru – Otevře okno, ve kterém se při pohybu myší po nějakém obrázku, budou vykreslovat směrové váhy adaptivního filtru. Klikem do nějakého obrázku se vykreslování vah zastaví v daném bodě. Opětovným kliknutím do libovolného obrázku se znovu spustí. Parametry filtru jsou opět v dolní liště.
  - Lineární filtrování – Analogické jako adaptivní filtrování, pouze s lineárním filtrem.
  - Sekvence filtrování – Na označeném obrázku provede více adaptivních nebo lineárních filtrování. Jsou zadány počáteční a koncové hodnoty jednotlivých parametrů a také délky jejich kroků. Provedou se všechny možné kombinace parametrů. Běh lze zastavit stiskem tlačítka stop (k jeho zastavení ale dojde až po dokončení současně běžící filtrace). Výsledky filtrování se uloží do seznamu. Klikem na daný řádek v seznamu se otevře příslušný obrázek a jeho histogram. (Případným klikem na „zavírací křížek“ tohoto obrázku se obrázek pouze skryje). Pozor, zavřením okna se seznamem, jsou všechny obrázky vytvořené pomocí sekvence filtrování ztraceny (a to i ty momentálně otevřené). Pozn. Nebudou však ztraceny případné filtrace nebo prahování těchto obrázků. Ztratí se skutečně pouze obrázky vytvořené přímo pomocí sekvence filtrování.
  - Zvol parametry – jediný způsob, jak změnit již zadané parametry adaptivního nebo lineárního filtru. Tyto parametry jsou vypsány v dolní liště.
- Histogram – Vytvoří histogram daného obrázku. V tom je následně možné provést prahování obrázku, jemuž histogram přísluší.
- Uspořádání – „Kaskáda“ a „Dlažba“ patřičně upořádají obrázky. „Zavřít všechny obrazy“ zavře všechny obrázky.

Pro práci se soubory „tif“ aplikace využívá knihoven z internetových stránek http://www.awaresystems.be/imaging/tiff/delphi.html. Tyto knihovny jsou součástí zdrojových souborů programu.

Literatura

[1] KLÍMA, Miloš, Martin BERNAS, Jiří HOZMAN a Pavel DVOŘÁK. *Zpracování obrazové informace.* Praha: ČVUT, 1996. ISBN 80-01-01436-3.

[2] KVAPIL, Jiří. *Adaptivní ekvalizace histogramu.* Brno: 2009. Diplomová práce. FSI VUT v Brně. Vedoucí diplomové práce Miloslav Druckmüller.

[3] PRATT, William K. *Digital Image Processing: PIKS Inside.* 3rd. ed. New York: Wiley, 2001. ISBN 0-471-22132-5.

[4] ŠTARHA, Pavel, Hana DRUCKMÜLLEROVÁ, Miloslav BĚLKA, František LÍZAL a Jan JEDELSKÝ. *Application of Adaptive Radial Convolutional Filter.* In: *Mendel 2013.* Brno: 2013, s.357–362. ISBN 978-80-214-4755-4.

Seznam příloh

1. CD s diplomovou prací (pdf), programem „Filtr.exe“, jeho zdrojovými soubory z Delphi a několika obrázky
2. CD s programem „Filtr.exe“, jeho zdrojovými soubory z Delphi a několika obrázky