Seznam příloh

CD-ROM

- Spustitelný soubor MIPS.exe
- Zdrojový kód
- HTML dokumentace

1. Úvod

Program MIPS *(Microscopy Image Processing Software)* je koncipován jako výukový software, který umožňuje různé metody předzpracování obrazu a je možné nad obrazem provádět různé operace. Jedná se o rozšíření programu [1]. Je zaměřen především na následující specifické moduly pro mikroskopii:

- Operace s histogramem
- Aritmetické a logické operace s obrazem
- Korekce nehomogenity osvětlení
- Dvojrozměrná konvoluce
- Binární a šedotónová Morfologie
- Statistický popis obrazu
- Filtrace pomocí Fourierovy transformace
- Operace pro zpracování barevných obrazů

V této diplomové práci nejdříve popisuji současnou podobu digitálního mikroskopu. Po této kapitole uvádím několik podobných výukových programů. Dále následuje stručný výčet používaných mikroskopických metod, které je potřeba vzít v úvahu při interpretaci získaných obrazových dat. Další rozsáhlý blok je věnován popisu principů jednotlivých operací zahrnutých v MIPS. U každé operace nechybí informace o tom, k čemu je vhodné ji použít. V další kapitole se zaměřuji na popis prostředí a způsob implementace jednotlivých funkcí. U pokročilejších funkcí nechybí zobrazení vzhledu pracovního okna. V kapitole ‘Přínos pro výuku‘ uvádím několik praktických ukázek, jak lze program použít při výuce. Nakonec nechybí závěr a zhodnocení konečné podoby programu.

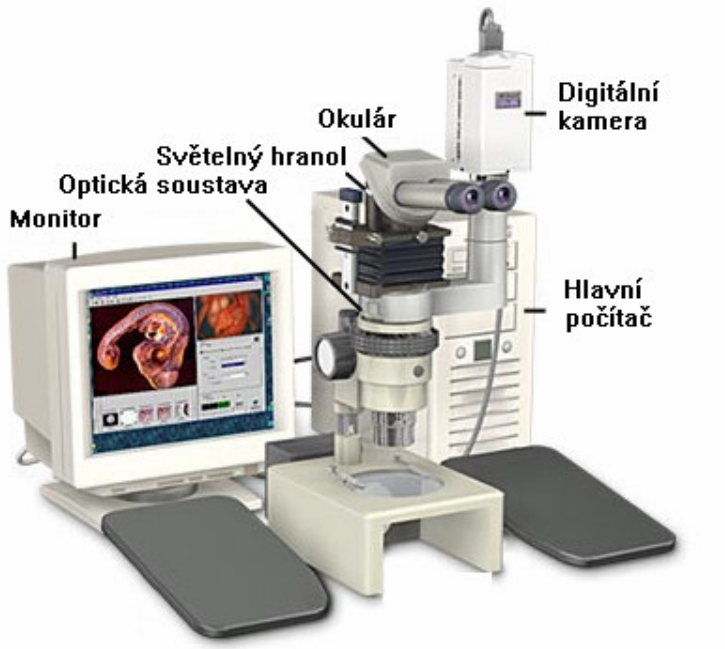
Hlavní snahou nebylo navrhnout dokonalý program, obsahující pokročilé metody zpracování obrazu. Snažil jsem se vytvořit prostředek pro předvedení mnoha operací a principů v této oblasti zpracování obrazu. K realizaci programu jsem použil prostředí C++ Builder 5.0. Veškeré zdrojové soubory jsou psány v jazyce C++.

2. Současný stav problematiky

2.1. Digitální mikroskop

Jednotlivé části digitálního mikroskopu jsou podrobně popsány v [19]. Na obr. 1 je příklad takovéhoto mikroskopu.

Důležité je si uvědomit, že dnes již pod pojmem mikroskop nerozumíme pouze optickou soustavu, ale jeho důležitou součástí je digitální kamera a systém předzpracování obrazu. Ten lze realizovat softwarově, pomocí vhodné zásuvné karty v počítači a příslušného programu. Často jsou ovšem podobné systémy již součástí mikroskopu.



Obr. 1. Mikroskop Nikon SMZ10

2.2. Výukové programy

Existuje řada programů, sloužících k předzpracování obrazu. Často však chybí objasnění, jak a proč jsme k danému výsledku dospěli. Některé z nich jsou také zbytečně složité.

V programu MIPS je proto kladen důraz na jednoduchost a přehlednost všech operací. Snažil jsem se implementovat všechny důležité operace pro práci s obrazem charakteristické pro snímky z digitálního mikroskopu.

Mezi programy, zabývající se podobnou tématikou patří například:

- ProImPro pro Windows (Stanislav Saic, Pavel Dvořáček)
- DIPS 4.0, obrazový analyzátor (SOFO, Druckmüller, Heriban)
- Lucia, obrazový analyzátor (Laboratory Imaging)

Tento program není omezený pouze na práci se snímky z mikroskopu. Řeší spoustu obecných metod zpracování obrazu, je tedy vhodný i na obrázky například z digitálního fotoaparátu.

3. Mikroskopické metody

V současné době pracují jednotlivé mikroskopy na odlišných principech. Díky tomu existuje mnoho různých výstupů v podobě obrazových dat, která jsou charakteristická například vlnovým rozsahem použitého záření apod. Proto je potřeba, abychom uvedli níže uvedený stručný výčet, který na tyto specifika upozorňují a při zpracování obrazu je nutné je vzít do úvahy.

Optická mikroskopie

Optická mikroskopie [11] umožňuje pozorovat a zaznamenávat mikroskopické objekty a struktury až do 1000-násobného zvětšení bez speciálních úprav mikroskopu. Umožňuje pozorovat vzorky v přirozeném stavu včetně vlhkosti a s malými úpravami také při nižších nebo vyšších teplotách.

Konfokální mikroskopie

Světelným zdrojem je laserové záření. Konfokální mikroskop [12] poskytuje mimořádně ostrý, kontrastní, vysoce informativní obraz s vysokým rozlišením. Struktury nacházející se nad a pod rovinou fokusace nemají téměř žádný vliv na kvalitu obrazu. Hloubka ostrosti je vždy minimální.

Buněčné struktury, které chceme pozorovat, obarvíme fluorochromem. Laserový paprsek je zaostřen do zvolené roviny, ve které dojde k osvícení preparátu laserovým paprskem a fluorochrom emituje viditelné záření. Získaný obraz je zaznamenán počítačem.

Fluorescenční mikroskopie

Fluorescenční mikroskopie [12] umožňuje zobrazit určité látky obsažené v buňkách často v minimálním množství. Metoda je založena na skutečnosti, že některé chemické látky (fluorochromy) po dopadu světla o kratší vlnové délce září světlem o delší vlnové délce (tedy světlem jiné barvy). Tento jev se nazývá fluorescence.

Princip nejčastěji užívaného postupu spočívá ve vazbě fluorochromu na určitou buněčnou složku (polysacharid, protein), která pak např. v modrém budícím světle září světlem žlutým.

Rastrovací elektronová mikroskopie

Rastrovací elektronová mikroskopie [11] umožňuje pozorování kovových nebo pokovených suchých vzorků až do 100 tisícinásobného zvětšení.

K dosažení obrazu musí být vzorek zbaven organických nečistot a umístěn ve vakuové komoře, aby dopadající elektronový svazek i odražené nebo vyražené elektrony nebyly rozptylovány srážkami s molekulami vzduchu.

Tunelovací mikroskopie

Tunelovací mikroskopie [11] umožňuje studium atomové struktury látek s rozlišením $1x10^{-7}$. Podstatou metody je kontakt jednoatomového hrotu s atomy povrchu vzorku, kde při vzdálenosti asi $10\dot{A}$ dochází v důsledku rozdílu elektrického potenciálu ke vzniku proudu, který je scanován a synchronně zobrazován.

4. Principy použitých operací

4.1. Bodové operace

4.1.1. Převodní charakteristika

Pomocí převodních charakteristik je možné provádět jasové korekce. Jasové korekce jsou takové transformace jasu, při kterých hodnota jasu ve výstupním obraze závisí pouze na hodnotě jasu ve vstupním obraze se stejnými souřadnicemi. Tyto transformace se snadno realizují pomocí tzv. vyhledávací tabulky (angl. Look-up table). Vyhledávací tabulka je pole o tolika paměťových místech jako je počet všech možných hodnot šedi v obraze. Index jednotlivých míst představuje hodnotu jasu vstupního obrazu, hodnota na místě paměťové buňky představuje korigovanou hodnotu jasu ve výstupním obraze.

4.1.2. Histogram

Histogram [5] je funkce, definována jako četnost výskytu jednotlivých úrovní šedi v obraze. Histogram lze chápat jako odhad hustoty pravděpodobnosti rozdělení jasu v obraze. Podstatné je pouze zastoupení jednotlivých úrovní šedi. Sečteme-li počty jednotlivých bodů pro všechny úrovně šedi, dostaneme počet všech bodů v obraze.

Histogram bývá velmi často jedinou globální informací o obraze. Můžeme ho použít při nastavování podmínek při snímání a digitalizaci, při změnách jasové stupnice, nebo při prahování za účelem oddělit objekty od okolí. Přímo z histogramu můžeme určit dynamický rozsah jasu v obraze, nebo zjistit jaké hodnoty jasu jsou v obraze zastoupeny.

Roztažení histogramu (angl. histogram stretching)

V ideálním případě by měl digitalizovaný obraz obsahovat všechny úrovně šedé. Toto není často splněno. V tomto případě lze použít bodovou funkci Roztažení histogramu [16].

Předpokládejme, že obsahuje původní obraz úrovně šedé od a_L do a_H. My však požadujeme, aby se tyto meze změnily na b_L a b_H. Výsledkem je obraz, u kterého je maximální rozsah jasových úrovní. Takto dochází ke zvýšení kontrastu ve snímku.

Postup pro získání výsledného obrazu lze popsat následující rovnicí:

$$I = (I_0 - a_L) \cdot \left[\frac{b_H - b_L}{a_H - a_L} \right] + b_L \tag{1}$$

kde I_0 je úroveň šedé původního obrazu, I je úroveň šedé výsledného obrazu.

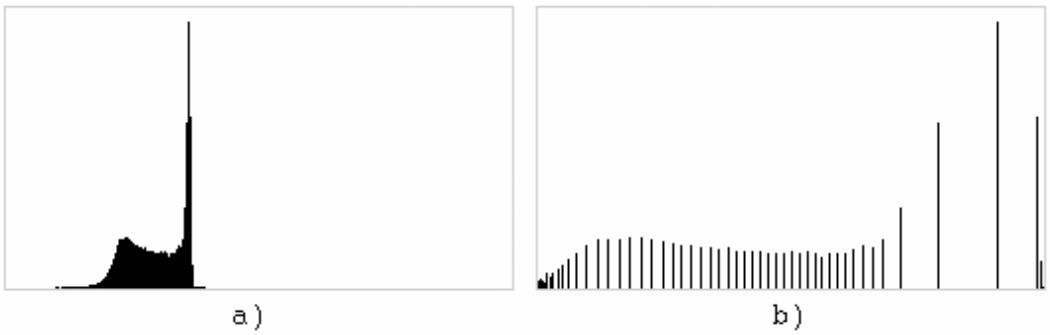
Obr. 2. Roztažení histogramu: a) původní histogram, b) histogram výsledného obrazu

Ekvalizace histogramu

Ekvalizace histogramu [16] je algoritmus, který změní rozložení intenzit v obraze tak, aby se v něm vyskytovaly intenzity přibližně se stejnou četností. U obrazů s konečným počtem obrazových bodů se lze tomuto cíli jen přiblížit. Ekvalizace umožňuje v obraze s celkově vysokým kontrastem zvýraznit špatně rozpoznatelné detaily s nízkým kontrastem. Vypočítá se jako:

$$q_j = \frac{q_k - q_0}{M \cdot N} \cdot \sum_{i=0}^{j} H(p_i) + q_0 \tag{2}$$

kde q_0 je požadovaná minimální hodnota ve výstupním histogramu, q_k hodnota maximální, $M \cdot N$ je rozměr obrázku, $H(p)$ je původní histogram.



Obr. 3. Ekvalizace histogramu: a) původní histogram, b) histogram výsledného obrazu

Modifikace histogramu

Modifikace histogramu umožňuje změnit v obraze rozložení intenzit jasu pomocí předem zvoleného pravidla. Při výpočtu vycházíme z kumulativního histogramu. Z něj vypočteme přenosovou funkci a pomocí ní transformujeme jasovou stupnici původního obrazu.

Různé přenosové funkce [10]:

- Uniform $$g(i) = g_{\min} + (g_{\max} - g_{\min}) \cdot P_N(i) \tag{3}$$
- Exponential $$g(i) = g_{\min} - \frac{1}{\alpha} \cdot \ln[1 - P_N(i)]^{\frac{1}{\alpha}} \tag{4}$$
- Rayleigh $$g(i) = g_{\min} + \left[2 \cdot \alpha^2 \cdot \ln\left(\frac{1}{1 - P_N(i)} \right) \right]^{\frac{1}{2}} \tag{5}$$
- Hyperbolic (Cube Root) $$g(i) = \left[\left(g_{\max}{}^{\frac{1}{3}} - g_{\min}{}^{\frac{1}{3}} \right) \cdot P_N(i) + g_{\min}{}^{\frac{1}{3}} \right]^3 \tag{6}$$
- Hyperbolic (Logarithmic) $$g(i) = g_{\min} \cdot \left(\frac{g_{\max}}{g_{\min}} \right)^{P_N(i)} \tag{7}$$

kde $(g_{\min}, g_{\max})$ je výsledný rozsah jasů, $P_N(i)$ je kumulativní histogram, viz rovnice (8).

Specifikace histogramu

Hlavní snahou je při specifikaci histogramu [9] najít takovou funkci, která by transformovala obraz tak, aby měl histogram získaného obrazu předem zvolený tvar. Tento proces může být žádoucí např. při odečítání dvou obrazů, které byly odlišně digitalizovány.

Postup pro stanovení požadované transformace rozdělíme do několika kroků Máme k dispozici jak vstupní, tak požadovaný histogram. Rovnice (8) a (9) popisují, jak z obou histogramů spočítat histogramy kumulativní.

$$P_N(i) = \frac{1}{M \cdot N} \cdot \sum_{j=0}^{i} H_{VST}(j) \tag{8}$$

$$P_{NM}(i) = \frac{1}{M \cdot N} \cdot \sum_{j=0}^{i} H_{POŽ}(j) \tag{9}$$

kde P_N je kumulativní histogram vstupního obrazu, P_{NM} je kumulativní histogram požadovaného obrazu, $M \cdot N$ je plocha obrazu.

Sloučíme-li oba kroky, tj. obě transformace, obdržíme námi hledanou výslednou transformační funkci:

$$f(i) = p_{NM}^{-1}[p_N(i)] \tag{10}$$

Pomocí této transformační funkce transformujeme jas všech pixelů v obraze. Má-li pixel v původním obrázku hodnotu jasu i, bude mít po transformaci hodnotu $f(i)$.

4.1.3. Aritmetické a logické operace

Aritmetické operace pro jeden obraz

Pro celý obraz provádíme bod po bodu danou aritmetickou operaci. Tyto operace je vhodné použít, chceme-li měnit jas o přesně daný počet jasových úrovní, případně provést úpravu kontrastu.

Je možno provést:

- přičtení konstanty $I = I_1 + k$ (11)
- odečtení konstanty $I = I_1 - k$ (12)
- násobení konstantou $I = k \cdot I_1$ (13)
- dělení konstantou $I = I_1 \div k$ (14)

Operace přičtení konstanty zvýší jas obrazu, naopak odečtením konstanty se výsledný jas sníží. Podobně operace násobení konstantou sníží kontrast obrazu a dělením konstantou kontrast snížíme.

Aritmetické operace pro dva obrazy

Na každé dva pixely (polohou stejné v obou obrazech) provedeme danou aritmetickou operaci. Mezi základní aritmetické operace patří:

- součet $I = I_1 + I_2$ (15)
- rozdíl $I = I_1 - I_2$ (16)
- násobení $I = I_1 \cdot I_2$ (17)
- dělení $I = I_1 \div I_2$ (18)
- průměr $I = (I_1 + I_2) \div 2$ (19)
- vážený průměr $I = k \cdot I_1 + (1-k) \cdot I_2, k \in \langle 0,1 \rangle$ (20)

Logické operace

Tyto operace jsou vhodné především u binárních obrazů (pouze jasové úrovně 0 a 255). Lze je ovšem použít i na kombinaci obrazu binárního a šedotónového, kdy můžeme šedotónový obraz různě modifikovat. Logické operace:

- a současně $I = I_1 \cap I_2$ (21)
- nebo $I = I_1 \cup I_2$ (22)
- výhradně nebo $I = I_1 \oplus I_2$ (23)

Je nutné si nadefinovat, jaká barva představuje Log 1 - zda bílá, či černá. Opačná barva odpovídá Log 0.

Ošetření dynamického rozsahu

Může se stát, že po provedení aritmetické operace "nepadne" hodnota jasu pixelu do intervalu $[0,255]$. Nabízí se několik způsobů, jak je to možné ošetřit:

- Limitace - jestliže je výsledek menší než 0 zobrazí se daný bod s jasem 0, obdobně pokud je větší než 255 zobrazí se jako bod s jasem 255.
- Přetečení a podtečení – pokud se nějaká hodnota pixelu dostane mimo interval 0 až 255 „vrátí se do něj" druhou stranou rozsahu. Takto se zobrazí např. číslo 256 jako 0 a číslo (-3) jako 253.
- Normalizace – najdeme minimální a maximální hodnotu ve výsledné matici (po provedení jedné z operací) a všechny hodnoty přepočítáme dle vztahu (1). Minimální hodnota bude poté odpovídat jasu 0 a maximální hodnota jasu 255.

4.1.4. Průměrování více snímků

Průměrování [3] je vhodné použít, máme-li více odpovídajících si snímků statické scény, které jsou zašuměné šumem s normálním rozložením. Výsledná hodnota každého pixelu se spočítá jako aritmetický průměr hodnot pixelů ze všech vstupních obrázků. Máme-li vstupní obrázky $I_1, I_2, ..., I_N$, výsledný obrázek I_{AVG} spočítáme z následující rovnice:

$$I_{AVG}(x, y) = \frac{1}{N} \cdot \sum_{j=1}^{N} I_j(x, y) \tag{24}$$

Tímto typem průměrování zvětšíme odstup užitečné informace v obraze od šumu.

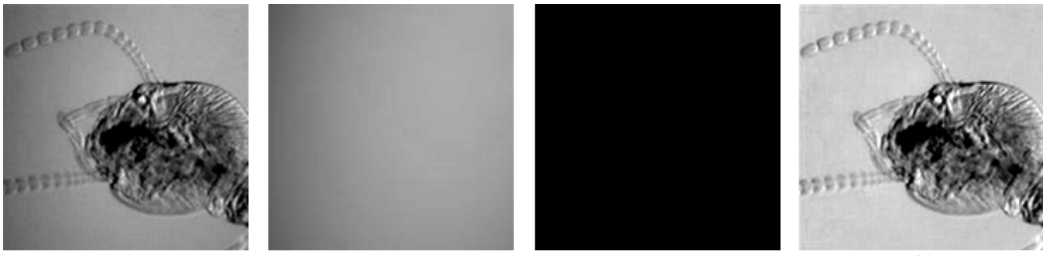
4.1.5. Flat-Field korekce

Při zpracovávání obrazových dat z digitálního mikroskopu je často problém s nerovnoměrným osvětlením podložky, která je umístěná pod snímaným objektem. Pomocí Flat-Field korekce [7] [3] lze tento nežádoucí jev omezit.

Nejdříve je potřeba získat snímek požadovaného objektu I_0, včetně nehomogenního pozadí. Potom posuneme sklíčko pod mikroskopem a nasnímáme pouze pozadí I_f. Použitím tohoto snímku odstraníme chyby způsobené nehomogenitou obrazového snímače. Nemáme-li k dispozici tento snímek, postačí použít nulové pole (černá plocha). Nakonec zakryjeme okulár a nasnímáme snímek za tmy I_b. Následující rovnice popisuje princip Flat-Field korekce.

$$I_c(x, y) = M \cdot \frac{I_0(x, y) - I_b(x, y)}{I_f(x, y) - I_b(x, y)} \tag{25}$$

Konstantou M ovlivňujeme výsledný dynamický rozsah, I_c je výsledný obraz.



Obr. 4. Ukázka Flat-Field korekce: a) před korekcí, b) pozadí, c) zatemněný snímač, d) výsledný snímek

4.1.6. Gama korekce

Rozložení intenzity jasu v obraze, který byl pořízen optickým mikroskopem, závisí nejen na osvětlení vzorku, ale také na citlivosti a linearitě snímacího detektoru (TV snímací elektronka, TV CCD, či CMOS snímací senzor). Také charakteristika zobrazovacího zařízení (obrazovka CRT, LCD) ovlivňuje toto rozložení. Zobrazený obraz proto nemusí odpovídat skutečnému. Gama korekce [2] je proces, který kompenzuje tento efekt.

Následující rovnice popisuje gama korekci pro osmibitový obraz (2^8 úrovní šedi):

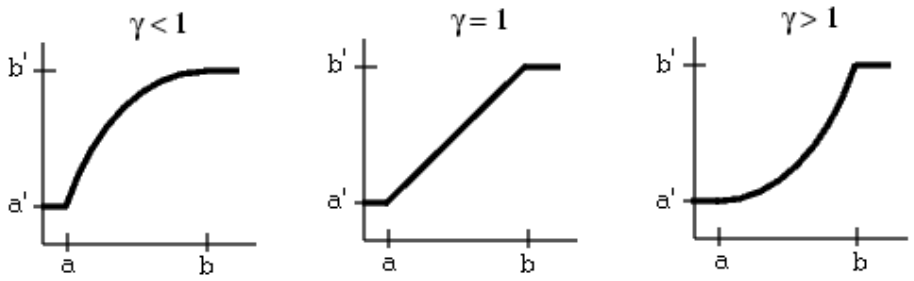
$$I_G(i,j) = 255 \cdot \left[\frac{I(i,j)}{255} \right]^{\frac{1}{\gamma}} \tag{26}$$

kde $I(i,j)$ je hodnota jasu pixelu v původním obraze o souřadnicích (i,j), γ je konstanta nabývající obvykle hodnoty 0,1 až 10.

Pokud je rozsah jasů v původním obraze a až b a požadujeme rozsah jasů ve výstupním obraze a' až b', lze rovnici (26) přepsat jako:

$$I_G(i,j) = \begin{cases} a' & , pro\ \ I(i,j) < a \\ b' & , pro\ \ I(i,j) > b \\ a' + (b' - a') \cdot \left[\frac{I(i,j) - a}{b - a} \right]^{\frac{1}{\gamma}} & , jindy \end{cases} \tag{27}$$

Na následujícím obrázku je zobrazena převodní charakteristika gama korekce pro různé hodnoty konstanty γ.



Obr. 5. Převodní charakteristika gama korekce pro různé γ

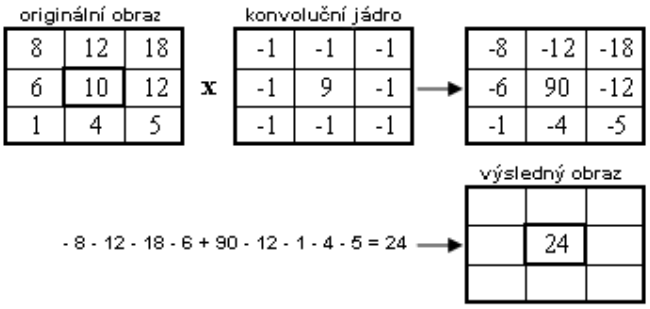
4.2. Lokální operace

4.2.1. Dvojrozměrná konvoluce

Podle účelu lze konvoluční operace [5] rozdělit na dvě skupiny – *vyhlazování obrazu* (potlačení šumu) a *gradientní operace* (zvýrazňování hran). Rovnice (28) popisuje dvojrozměrnou diskrétní konvoluci s konvoluční jádrem a. Toto konvoluční jádro bývá někdy označováno jako konvoluční maska, v teorii filtru jako impulsní odezva filtru. Vhodný výběr konvolučního jádra bývá založen na experimentech, popř. znalosti kmitočtových vlastností obrazového signálu.

$$c[m,n] = a[m,n] \otimes b[m,n] = \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} a[j,k] \cdot b[m-j, n-k] \tag{28}$$

kde a je konvoluční jádro o rozměru $J \times K$, b je obraz, který chceme filtrovat.



Obr. 6. Princip filtrace pomocí konvoluce

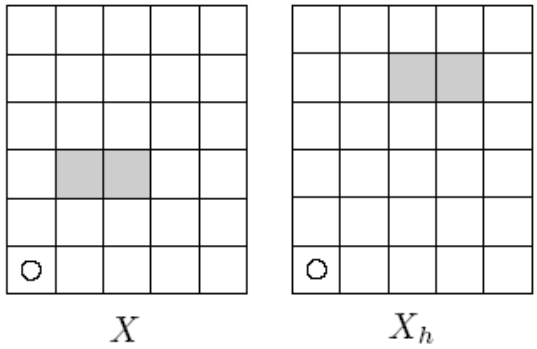
4.2.2. Binární a šedotónová Morfologie

Morfologie [14] je matematický nástroj pro rozbor struktury obrazu. Jedná se o rychlé algoritmy implementované pomocí operací se strukturními elementy.

Posun

Na obr. 7 je zobrazena operace "posun" při $h=(1,2)$. Tuto operaci později využijeme při definování morfologické dilatace a eroze. Operace se dá popsat vztahem:

$$X_h(p)=X(p-h),\quad p\in X\subset Z^2 \tag{29}$$

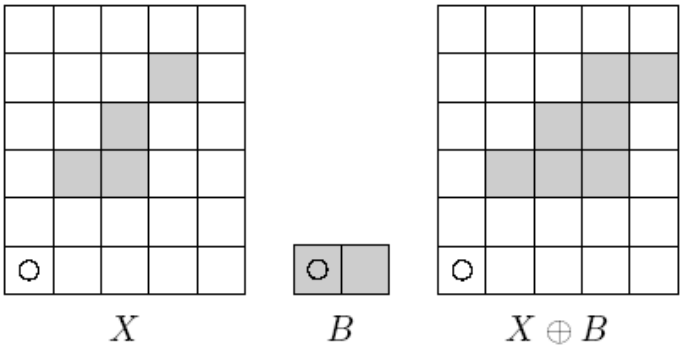


Obr. 7. Posun

Binární Dilatace

Na obr. 8 je zobrazen příklad operace "dilatace". Dilatace slouží ke zvětšení (rozmazání) objektů v obrazu dle použité masky.Vypočítá se podle vztahu:

$$X\oplus B=\bigcup_{\{y,B(y)=1\}}X_y \tag{30}$$



Obr. 8. Dilatace

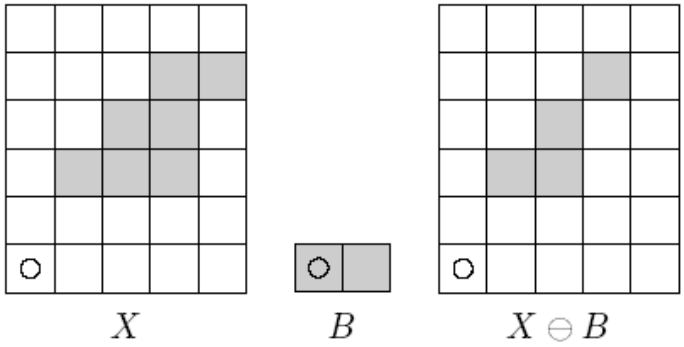
Pomocí posunu je možné popsat předchozí obrázek vztahem:

$$X \oplus B = X_{[0,0]} \bigcup X_{[1,0]} \tag{31}$$

Binární Eroze

Eroze je opak dilatace, dochází při ní ke zmenšování objektů v obraze. Přesto ovšem nejsou tyto dvě operace vzájemně inverzní. Příklad je zobrazen je na obr. 9 a lze popsat rovnicí:

$$X \ \Theta \ B = \bigcap_{\{y, B(y)=1\}} X_{-y} \tag{32}$$



Obr. 9. Eroze

Eroze dle obr. 9 zapsaná pomocí posunu:

$$X \ \Theta \ B = X_{(0,0)} \bigcap X_{(-1,0)} \tag{33}$$

Otevření, Uzavření, Hranice objektu

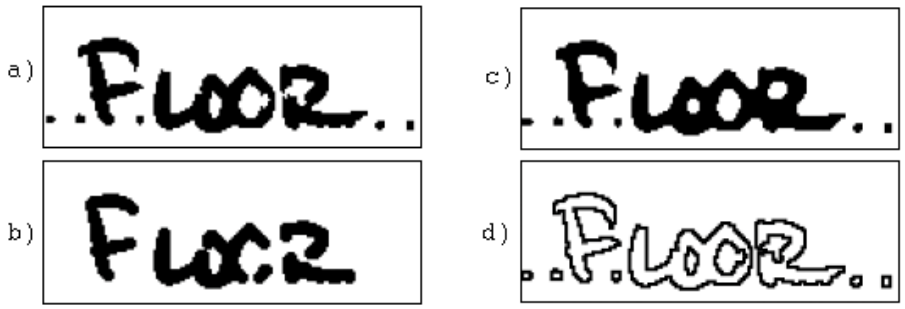
Kombinací dilatace a eroze lze vytvořit další důležité operace. Rovnice (34) popisuje operaci "otevření", rovnice (35) popisuje operaci "uzavření", rovnice (36) popisuje operaci "hranice objektu". Otevřením odstraníme v obraze osamocené body a drobné škrábance. Uzavření nám umožňuje uzavřít oblasti, které byly z různých příčin přerušené. Hranice objektu je operace, která lze použít například před segmentací, kdy identifikujeme v obraze jednotlivé oblasti.

$$X \circ B = \left(X \ \Theta \ B\right) \oplus B \tag{34}$$

$$X \bullet B = \left(X \oplus B\right) \Theta \ B \tag{35}$$

$$\beta(X) = (X \oplus B) - (X \ominus B) \tag{36}$$

Následující obrázek demonstruje, jak bude vypadat výsledný obraz po provedení každé z těchto tří morfologických operací.



Obr. 10. Morfologické operace: a) původní obrázek, b) otevření, c) uzavření, d) hranice objektu

Podmíněná dilatace

Tato operace [4] se skládá ze dvou kroků. Nejdříve do obrázku přidáme obrazec, který budeme později dilatovat. Poté, dokud je splněna předem definovaná podmínka, provádíme opakovaně dilataci.

Podmíněnou dilataci můžeme použít například k odfiltrování okrajových elementů v obraze. Jako obrazec použijeme okraj snímku, na který aplikujeme opakovaně dilataci. Pokud oblast, nově vzniklá dilatací nepřekryje žádnou část libovolného elementu v obraze, skončíme.

Odstranění šumu typu sůl a pepř

Kombinací otevření a uzavření lze vytvořit filtr, který eliminuje šum typu sůl a pepř. Jedná se o šum, kdy se v obraze vyskytují náhodně světlé a tmavé pixely. Volbou velikosti masky lze měnit prostorovou rozlišitelnost filtru. Tento filtr má podobné vlastnosti jako mediánový filtr.

$$\phi(X) = (X \circ B) \bullet B \tag{37}$$

4.2.3. Korekce nehomogenity osvětlení

Tato korekce [15] je vhodná na snímky, které obsahují drobné objekty. Podobně jako Flat-Field korekce omezí nehomogenitu pozadí.

Korekce se skládá z několika kroků:

- rozdělíme obraz na čtverce s definovanou velikostí
- spočítáme průměrnou hodnotu jasu u každého čtverce
- aproximací z průměrných hodnot získáme snímek pozadí
- odečteme původní obraz a získaný obraz pozadí

4.3. Globální operace

4.3.1. Filtrace pomocí Fourierovy transformace

Diskrétní Fourierova transformace

Dvojrozměrná přímá diskrétní Fourierova transformace [5] je definovaná rovnicí (38). Rovnice (39) popisuje zpětnou Fourierovu transformaci.

$$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\exp\left[-j2\pi\frac{(ux+vy)}{N}\right];\ u,v=0,1,2,...N-1 \tag{38}$$

$$f(x,y)=\frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}F(u,v)\exp\left[j2\pi\frac{(ux+vy)}{N}\right];\ x,y=0,1,2,...N-1 \tag{39}$$

Vzhledem k tomu, že je jádro separabilní, je možné počítat přímou Fourierovu transformaci jako:

$$F(u,v)=\frac{1}{N}\sum_{y=0}^{N-1}F(u,y)\exp\left[-j2\pi\frac{vy}{N}\right] \tag{40}$$

Díky této důležité vlastnosti můžeme rozložit výpočet 2D DFT do dvou kroků. Nejdříve spočítáme jednorozměrnou DFT všech řádků $f(x,y)$, takto získáme $F(u,y)$. Na všechny sloupce této funkce aplikujeme znovu jednorozměrnou DFT a tím získáme $F(u,v)$.

Filtrace

Pomocí 2D DFT lze realizovat filtraci v prostorové frekvenční oblasti. Transformací obrazu $f(x,y)$ získáme $F(u,v)$. Představuje-li $H(u,v)$ použitý filtr, lze

potom filtraci zapsat jako:

$$G(u,v) = H(u,v) \cdot F(u,v) \tag{41}$$

Zpětnou transformací získáme z $G(u,v)$ vyfiltrovaný obraz $g(x,y)$.

Filtrace ve frekvenční oblasti umožňuje jednoduchý návrh filtru. Jako příklad uvádím ideální filtr typu dolní propust, který může být definován jako:

$$H(u,v) = \begin{cases} 1 \ , pro \quad D(u,v) \le D_0 \\ 0 \ , pro \quad D(u,v) > D_0 \end{cases} \tag{42}$$

$$D(u,v) = \sqrt{u^2 + v^2} \tag{43}$$

kde $D(u,v)$ představuje vzdálenost od počátku a D_0 hraniční frekvenci. Podobně lze vytvořit horní propust, pásmovou propust, či pásmovou zádrž.

4.3.2. Centroid

Centroid [8] představuje míru geografického středu objektu v obraze. Pro binární obraz se jeho souřadnice vypočítají z rovnic (44) a (45).

$$X_C = \frac{1}{A} \cdot \sum_{i=1}^{N} X_i \tag{44}$$

$$Y_C = \frac{1}{A} \cdot \sum_{i=1}^{N} Y_i \tag{45}$$

kde X_i a Y_i jsou souřadnice i-tého pixelu objektu, A je plocha objektu.

4.3.3. Statistické veličiny

Mezi základní statistické veličiny [20], které je možno v obraze zjišťovat patří:

- Aritmetický průměr

$$\overline{x_a} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{46}$$

- Rozptyl

$$\operatorname{var} x = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(x_i - \bar{x}_a \right)^2 \tag{47}$$

- Směrodatná odchylka

$$\sigma = \sqrt{\operatorname{var} x} \tag{48}$$

- Entropie

$$E = \frac{1}{n} \cdot \sum_{i=1}^{255} h[i] \cdot \log_{10} h[i] \tag{49}$$

kde n je počet všech obrazových bodů, x_i je úroveň šedi i-tého pixelu v obraze, $h[i]$ je počet bodů s hodnotou jasu i (histogram).

4.3.4. SNR/PSNR

SNR (Signal-to-Noise Ratio) a PSNR (Peak Signal-to-Noise Ratio) [13] odhadují kvalitu obrazu po rekonstrukci v porovnání s původním obrazem. Jsou popsány rovnicemi (51) a (52).

Konkrétní hodnota této veličiny o ničem nevypovídá, míru kvality rekonstrukce získáme porovnáním výsledků dvou rozdílně rekonstruovaných obrazů.

$$R = \| f - F \| = \sum_{k=1}^{N} \left(f_k - F_k \right)^2 \tag{50}$$

$$SNR_{dB} = 10 \cdot \log_{10} \left(\frac{\sum_{k=1}^{N} f_k^2}{R} \right) \tag{51}$$

$$PSNR_{dB} = 10 \cdot \log_{10} \left(\frac{255^2 \cdot N}{R} \right) \tag{52}$$

kde f je původní obraz, F je obraz po rekonstrukci, N je celkový počet pixelů v obraze.

Pomocí následující rovnice můžeme vypočítat chybový obrázek.

$$E(i,j) = c_1 \cdot \left[f(i,j) - F(i,j) \right] + c_2 \tag{53}$$

Volbou konstant c_1 a c_2 můžeme měnit kontrast a stejnosměrnou složku výsledného obrazu.

4.4. Operace s barevnými obrazy

4.4.1. Reprezentace barev

Jedním z nejdůležitějších atributů, používaných při zpracování obrazu je barevná informace. Každá barva odpovídá určité frekvenci elektromagnetického vlnění. Rozsah všech barev odpovídá intervalu vlnových délek zhruba od 350 do 780nm (od fialové až po červenou). V rámci viditelného spektra je člověk schopen rozlišit více než $4x10^5$ barevných odstínů.

4.4.2. Barevné modely

Mnoho fyziologických studií prokázalo, že barevná informace je třísložková. To znamená, že k popsání všech barevných odstínů je nutné použít tři proměnné. Libovolnou barvu tedy můžeme popsat vektorem v $\Re^3$.

Existuje mnoho barevných modelů, kterými se popisuje barevná informace a na jejich základě je možné každý barevný odstín dekomponovat na základní složky.

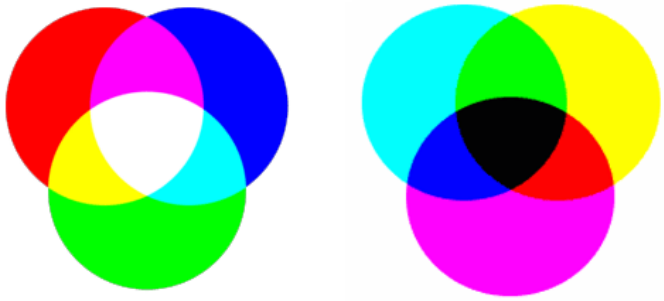
Barevné modely [17], kterými se zde budu zabývat, lze rozdělit do dvou skupin:

- Modely založené na fyziologii oka – RGB model, CMY(K) model
- Modely psychologické a psychofyzikální – HSV, HSL

RGB, CMY, CMYK

Základem RGB modelu jsou tři barevné složky (red, green, blue). Pro tyto barvy je charakteristické právě to, že lidské oko má největší spektrální citlivost právě pro jejich vlnové délky (630nm, 530nm a 450nm). RGB model pracuje na principu aditivního míchání barev. Základem tohoto principu je fakt, že tři světelné svazky, tmavě modrý, sytě červený a sytě zelený, dohromady vytvářejí jasné, zářivě bílé světlo. Na základě tohoto modelu pracují všechny monitory, televizní obrazovky apod.

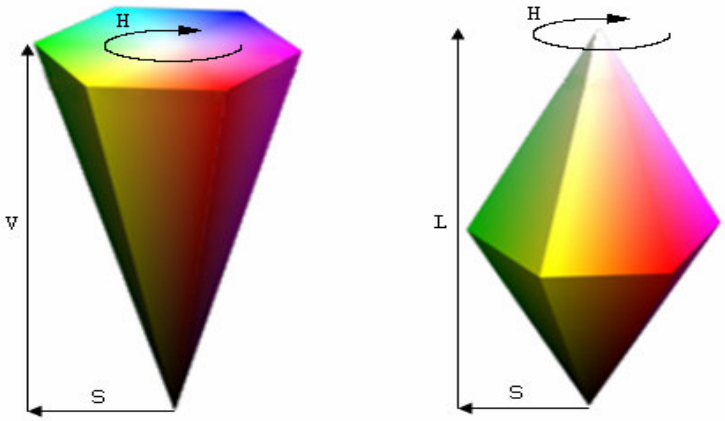
CMY (cyan, magenta, yellow) model pracuje na principu subtraktivního míchání barev. Na rozdíl od RGB modelu, který je v podstatě založen na vyzařování světla, je založen na odrazu světla. Zatímco při aditivním míchání skutečně získáme čistě černou barvu (nepřítomnost světla), subtraktivním mícháním čistě černé barvy nedosáhneme. CMY model je proto obvykle obohacován o čtvrtou černou složku (black). Získáme tak výsledný model CMYK. V módu CMYK pracuje většina tiskáren.



Obr. 11. Způsoby míchání barev: aditivní (vlevo), subtraktivní (vpravo)

HSV, HSL

HSV model je definován parametry hue (barevný odstín), saturation (sytost) a value (světlost). U modelu HSL se namísto value používá termín lightness. Předností těchto modelů je intuitivní způsob míchání barev. Při manipulaci s barvou je v první fázi zvolen příslušný barevný odstín, ve druhé fázi pak sytost a světlost. Barevný odstín se při manipulaci se sytostí a světlostí nemění, čehož se dá u modelu RGB, či CMYK dosáhnout jen stěží.



Obr. 12. Barevné modely: HSV (vlevo), HSL (vpravo)

4.4.3. Metody transformace do šedotónové stupnice

Může se stát, že potřebujeme z obrazu vypustit barevnou informaci. Chceme například použít matematickou operaci, která lze použít jen pro šedotónový obraz. Transformaci do šedotónové stupnice můžeme provést několika způsoby [17]. Výsledná hodnota bude vždy záviset na všech třech barevných složkách (červená, modrá, zelená). Musíme pouze zvolit vhodné zastoupení jednotlivých složek:

- CIE standard (Rec.709)

$$Y(i,j) = 0{,}2126 \cdot R(i,j) + 0{,}7152 \cdot G(i,j) + 0{,}0722 \cdot B(i,j) \tag{54}$$

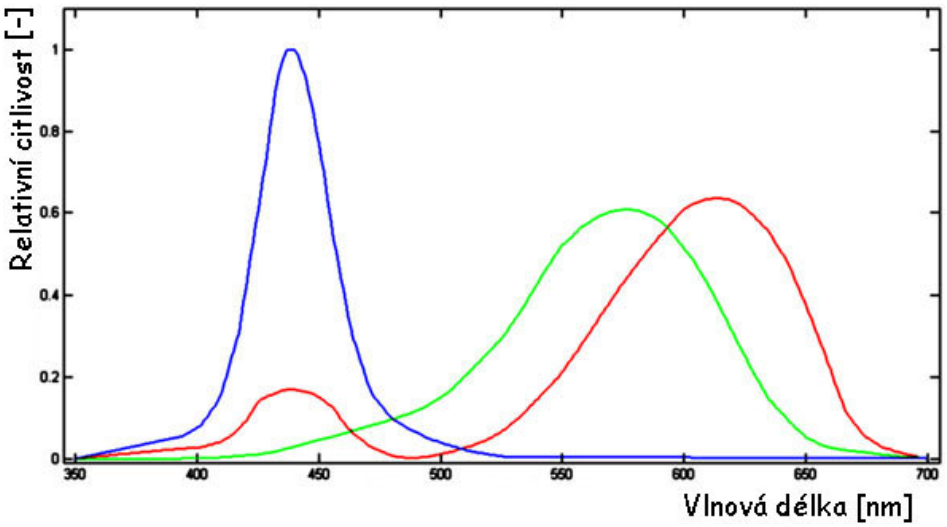
- Nonlinear luma

$$Y(i,j) = 0{,}2990 \cdot R(i,j) + 0{,}5870 \cdot G(i,j) + 0{,}1140 \cdot B(i,j) \tag{55}$$

- Average

$$Y(i,j) = 0{,}3333 \cdot R(i,j) + 0{,}3333 \cdot G(i,j) + 0{,}3333 \cdot B(i,j) \tag{56}$$

CIE standard (Rec.709) a Nonlinear luma jsou přizpůsobené vjemu barev lidským okem, které není pro všechny barvy stejně citlivé - zobrazeno na obr. 13. Třetí metoda, kdy získáme výslednou hodnotu jasu pouhým průměrováním přes všechny barevné složky, je spíše než pro zobrazení vhodná pro další zpracování.



Obr. 13. Citlivost lidského oka na jednotlivé barvy

4.4.4. Pseudobarevné obohacení obrazu

Účelem pseudobarevného obarvení [18] je zvýšit resp. usnadnit vnímavost důležitých částí nebo detailů obrazu. Malé jasové rozdíly, v šedotónovém obraze téměř nepostřehnutelné, jsou pseudobarevným zobrazením barevně odlišeny a tím je zvětšena jejich viditelnost.

Při pseudobarevném zobrazení se stupnici šedé přiřazuje barevné zobrazení. Důvodem je skutečnost, že lidské oko rozezná podstatně více barev než jednotlivých úrovní šedi. To je způsobeno tím, že v oku jsou tři čidla na zpracování barevné informace. Pseudobarevné obohacení obrazu lze použít ve spektrální analýze, pro ultrazvukové snímky, nebo ve fluorescenční mikroskopii.

Nepravé barvy

Pokud požadujeme změnit barevné odstíny u již reálného barevného obrazu na nereálné (například purpury) použijeme tzv. nepravé barvy.

5. Implementace uživatelského rozhraní

Načítání obrazu

Program umožňuje načíst obraz až do velikosti 1024x1024 obrazových bodů. Výjimkou je funkce (View / Zoom In), která umožňuje zobrazovat neomezeně zvětšený obrázek. Podle toho, zda je obraz šedotónový či barevný, se v nabídce menu zpřístupní odpovídající položky. Přesto, že mají šedotónové i barevné obrazy stejnou příponu, musíme před načtením upřesnit, jaký typ otevíráme.

Windows Bitmap

V programu je možné pracovat se soubory typu Windows Bitmap [6], načíst lze 8-bitové a 24-bitové obrázky. 8-bitové mohou být buď šedotónové, nebo mohou obsahovat 256-ti barevnou paletu. 24-bitové jsou ty, které mají pro každý pixel definované tři 8-bitové složky (červená, modrá, zelená).

Hlavička	informace o obrázku (typ, počet barev, velikost souboru, výška, šířka)												
LUT tabulka	255	0	0	10	50	80	0	0	0	...	20	25	16
Data	postupně pro každý pixel pořadí jeho barvy v LUT tabulce												

B	G	R	
255	0	0	→ První barva LUT tabulky. V tomto případě modrá.

Tab 1: Obrázek 8bit (256 barev)

Hlavička	informace o obrázku (typ, počet barev, velikost souboru, výška, šířka)												
LUT tabulka	0	0	0	1	1	1	2	2	2	...	255	255	255
Data	postupně pro každý pixel pořadí jeho barvy v LUT tabulce												

Tab 2: Obrázek 8bit (šedotónová stupnice)

Hlavička	informace o obrázku (typ, počet barev, velikost souboru, výška, šířka)												
Data	0	255	0	10	10	10	87	87	87	...	64	64	64

B	G	R	
0	255	0	→ Barva prvního pixelu v obrázku. V tomto případě zelená.

Tab 3: Obrázek 24bit (16,7 milionů barev)

Měření vzdáleností

Nejdříve zvolíme první bod v obraze. Ten vybereme tak, že na něj přesuneme kurzor myši a klikneme levým tlačítkem myši. Poté přesuneme kurzor na druhý bod. V dolní části hlavního okna se zobrazuje vzdálenost mezi těmito dvěma body.

Funkce „Undo“

Funkce Undo umožňuje návrat k obrázku před danou operací. Lze opakovaně použít až 10x za sebou. Přejdeme-li od barevného obrázku k šedotónovému, není možnost se již vrátit zpět k původnímu barevnému. Je to způsobené rozdílnosti typů se kterými neumí funkce Undo současně pracovat.

5.1. VIEW Menu

5.1.1. Lupa

Lupa (View / Magnifying Glass) umožňuje zobrazit v samostatném okně detail části obrazu. Měnit můžeme polohu tohoto místa v původním obraze a velikost zvětšení. Je možné snížením jasu odlišit oblast, která není zvětšena. Z důvodu, že dochází ke zvětšování obrazu, musíme zvolit metodu dopočítání všech hodnot v nově vytvořeném obraze. V programu je implementována lineární interpolace, nebo výpočet na základě nejbližšího souseda.

```
class TMagnifyingGlassForm : public TForm
{
public:
      // na základě zvolené metody se vyřízne a zvětší část obrazu
      void __fastcall ViewDetail(int x, int y);
      // snížení jasu v celém původním obraze
      void __fastcall MakeDark(GreyImage* im_in, GreyImage* im_out);
      // zobrazení původního obrazu, oblast která není zvětšena je odlišena nižším jasem
     void __fastcall CombineDetailAndDark(…);
};
```

5.1.2. Zobrazení více obrázků

Funkce je umístěna v nabídce menu (View / Show more Images). Lze načíst až 6 obrázku a současně je zobrazit. Je možno načítat šedotónové i barevné obrázky.

```
// Zobrazení dvou obrázků
Class TShowTwoImagesForm : public TForm{
public:
     // zobrazení, je li obraz šedotónový
     void __fastcall Load_grey_image(TImage* _image);
     // zobrazení, je li obraz barevný
     void __fastcall Load_color_image(TImage* _image);
};

// Zobrazení čtyř a šesti obrázků
Class TShowFourImagesForm{}; Class TshowSixImagesForm{};
```

5.2. IMAGE Menu

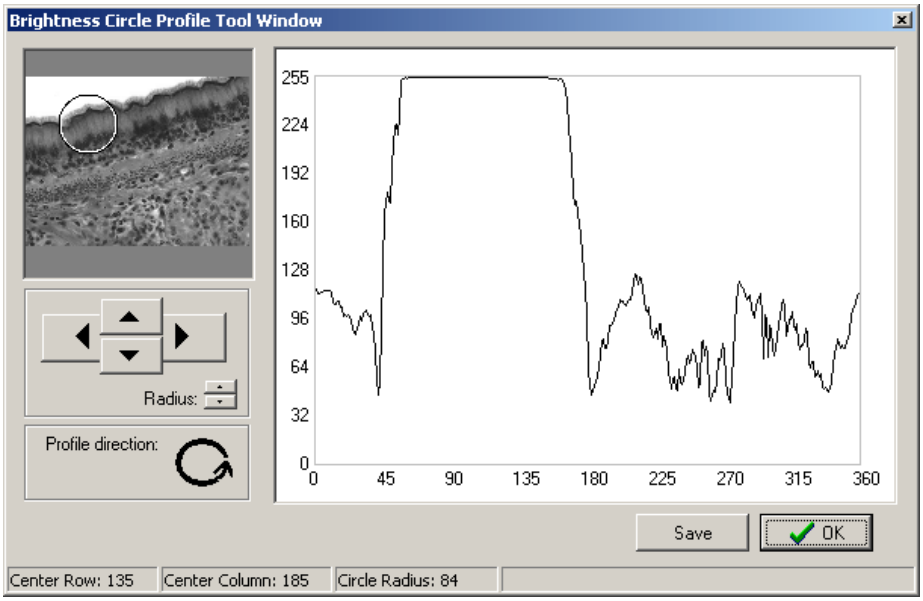
5.2.1. Jasový profil

Tato funkce je přístupná v položce menu (Image / Brightness Profile). Máme možnost si zvolit, zda chceme zobrazit profil

- řádku / sloupce
- po libovolné úsečce
- po kružnici

Zvolíme-li profil řádku (sloupce) stačí přesunout ukazatel myši na obrázek v hlavním okně programu a posunem myši volíme požadovaný řádek (sloupec), jehož profil chceme zobrazit. Pro zobrazení profilu libovolné úsečky, určíme počáteční a koncový bod. V případě profilu po kružnici můžeme měnit střed a poloměr této kružnice.

Výsledný profil je zobrazen v pracovním okně. Je také možnost si jej uložit do souboru a dále s ním pracovat, například v programu Matlab.



Obr. 14. Jasový profil po kružnici

```
// profil řádku (sloupce)
class TBrightnessProfileForm : public TForm{
public:
      // vykreslení os (dle velikosti obrázku)
      void __fastcall ShowAxes();
      // vykresleni profilu
      void __fastcall ShowGraph(int x, int y);
};

// profil po úsečce
class TLineProfileForm{};

// profil po kružnici
class TCircleProfileForm{};
```

5.2.2. Práce s histogramem

Zobrazení histogramu

Histogram zobrazíme volbou (Image / Histogram / View). Na vodorovné ose jsou jednotlivé úrovně šedi a na svislé počet bodů k nim náležících. Pokud přesuneme ukazatel myši na tento graf, zobrazí se v dolním řádku tento počet.

```
class Histogram
{
public:
      // výpočet histogramu z obrazu
      void InitFromGreyImage(const GreyImage& init,…);
      // vykreslení histogramu
      void Display(TImage* Image);
      // roztažení histogramu
      void HistogramStretching(const GreyImage& or_image, st_image, float omission);
      // ekvalizace histogramu
      void HistogramEqualization(const GreyImage& or_image, eq_image);
};
```

Roztažení histogramu

Funkce je přístupná v nabídce (Image / Histogram / Stretching). V zobrazeném okně se zobrazí obraz původní, obraz po roztažení histogramu a jejich histogramy. Můžeme si zvolit tzv. omission level. Při hledání minimálního a maximálního jasu budou ignorovány body, jejichž četnost je v obraze menší než tato úroveň. Je udávána v procentech a je vztažena k číslu, které udává počet pixelů, jejichž zastoupení v obraze je největší.

```
// roztažení histogramu, zobrazení histogramu před a po roztažení
class THistogramStrechingForm : public TForm{};
```

Ekvalizace histogramu

V menu zvolíme (Image / Histogram / Equalization). Pracovní okno je obdobné jako u roztažení histogramu. Omission level není k dispozici, u ekvalizace histogramu nemá význam.

```
// ekvalizace histogramu,  zobrazení histogramu před a po ekvalizaci
class THistogramEqualizationForm : public TForm{};
```

Modifikace histogramu

Po zvolení (Image / Histogram / Modification) vybereme jednu z transformačních funkcí. Ta je poté použita na vstupní obraz a zobrazí se výsledný obraz včetně jeho histogramu. U některých funkcí můžeme měnit konstantu α a tak měnit parametry příslušné funkce.

```
class THistogramModificationForm : public TForm
{
public:
      // výpočet transformační funkce
      void __fastcall UniformTransform();
      void __fastcall ExponencialTransform();
      void __fastcall RayleighTransform();
      void __fastcall CubeRootTransform();
      void __fastcall LogarithmizeTransform();
      // zobrazení náhledu
      void __fastcall ComputePreviewImage();
      // výpočet výsledného obrazu
      void __fastcall ComputeFinalImage();
      // zobrazení původního histogramu
      void __fastcall ShowOriginalHistogram();
      // zobrazení histogramu výsledného obrazu
      void __fastcall ShowModifHistogram();
};
```

Specifikace histogramu

Zvolíme (Image / Histogram / Specification). Použil jsem již vytvořenou třídu *LUT*, která obsahuje předefinované převodní charakteristiky (LUT tabulky). Tu zde používám k nadefinování tvaru požadovaného histogramu. Lze také ručně nakreslit libovolný průběh převodní charakteristiky a tak vytvořit libovolný tvar požadovaného histogramu.

```
class THistogramSpecificationForm : public TForm
{
public:
      // transformace LUT tabulky na pole čísel
      void __fastcall SetData(int data[]);
      void __fastcall SetData(LUT* lut);
      // normalizace požadovaného histogramu
      void __fastcall NormalizeSurface();
      // aplikace transformační funkce na vstupní obraz
      void __fastcall ComputeTransform();
      void __fastcall ComputePreview();
      void __fastcall ComputeFinalImage();
      // zobrazení
      void __fastcall ShowHistogram();
      void __fastcall ShowModifHistogram();

};
```

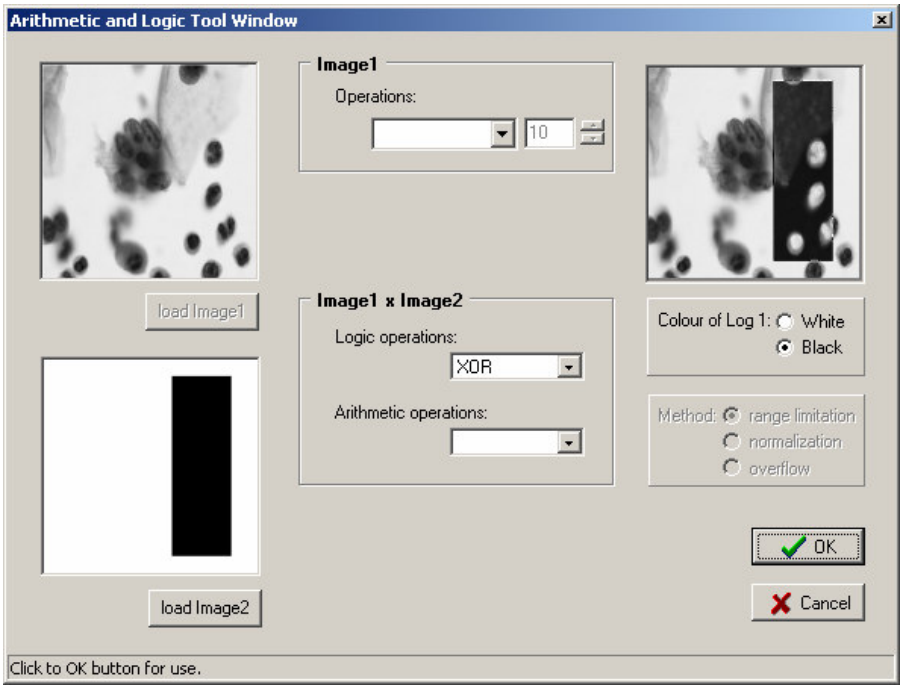
5.2.3. Logické a aritmetické operace

Funkce je přístupná v položce (Image / Logic and Arithmetic Operations). Při otevření tohoto okna máme k dispozici 2 obrazy: jeden je načten z hlavního okna programu a druhý si můžeme zvolit.

S prvním obrazem je možno provádět základní aritmetické operace: přičtení a odečtení konstanty, násobení a dělení konstantou. Pokud vybereme přičtení nebo odečtení konstanty, zpřístupní se nám další funkce. Daná operace bude aplikována pouze na ty body v obraze, jejichž jas spadá do zvoleného intervalu.

Pokud vybereme i druhý obraz, zpřístupní se aritmetické a logické operace mezi dvěma obrazy . Operace je aplikována na obrazy ihned po výběru jedné z operací. U logických operací volíme jaká úroveň odpovídá logické 1 (0 či 255). U aritmetických operací volíme ošetření hodnoty po provedení příslušného výpočtu (limitace, normalizace, přetečení).

```
enum BitFunction
{
     AND, OR, XOR, ADD, SUB, MULT, DIV, MIN, MAX, AVE, OVERLAY,
     InvAND, InvOR, InvXOR, PLUS, MINUS, MULTIP, fDIV
};
```



Obr. 15. Ukázka logické operace XOR

```
class FrameImage{
public
      // konstruktor:
       __fastcall FrameImage();
       // destruktor
      __fastcall ~FrameImage() {};
       // aplikuje na obraz  zvolenou matematickou operaci
      void __fastcall ApplicationFunction(BitFunction combine_type,…);
      //  sečte 2 obrazy, parametr percent určuje zastoupení prvního obrazu
      void __fastcall  WeightImages(int percent,…);
      // výpočet mezi dvěma obrazy dle zvolené matematické operace
      void __fastcall  CombineImages(BitFunction combine_type, int method_type,…);
};
```

5.2.4. Dvojrozměrná konvoluce

Konvoluce se nachází v položce menu (Image / Convolution). Nejdříve musíme nadefinovat konvoluční masku. Zvolíme předdefinovanou (Kernel / MIPS's), nebo si můžeme vytvořit novou (Kernel / New Kernel). Lze vytvářet masky až do velikosti 25x25. Nově vytvořenou masku si můžeme uložit a později opět načíst a použít. Tlačítkem 'Perform' aplikujeme konvoluci.

```
class Convolution{
public:
    // provede se  konvoluce
    void PerformConvolution(…);
    // přiřazení požadovaného konvolučního jádra
    void SetKernel(Kernel& _kernel);
};
```

Implementovaná konvoluční jádra

MIPS obsahuje 110 předdefinovaných konvolučních jader. Nemá význam zde uvádět podobu jednotlivých matic. Uvedu jen přehled jejich typů:

- Gradientní hranové detektory (ve všech osmi směrech)
- Matematické funkce (compass max, kirch max, mean-square)
- Filtry (horní propust, dolní propust)
- Rozmazávací a vyhlazovací funkce

Každou z matic (představující konvoluční jádro) je možno editovat, uložit do souboru, nebo vytvořit zcela novou a to až do velikosti 25x25.

```
// definice předdefinovaných masek
class Kernel {
…
};
```

5.2.5. Morfologie

K dispozici jsou binární (Image / Morphological Filtres / Binary) a šedotónové (Image / Morphological Filtres / Greylevel) morfologické funkce. Zvolíme li binární, provede se před výběrem požadované operace prahování obrazu dle zvolené hodnoty prahu. Poté bude obraz obsahovat pouze 2 jasové úrovně (0 a 255).

Masku, která bude použita pro vybranou morfologickou operaci, můžeme měnit. I zde je k dispozici několik předdefinovaných masek.

```
class   Morpholog{
public:
    // inicializace masky
    void NewMask(VMask* init);
    // dilatace
    virtual void Dilation(){};
    // eroze
```

```
    virtual void Erosion(){};;
    // otevření
    virtual void Open(){};
    // uzavření
    virtual void Close(){};
    // filtrace šumu typu "sůl a pepř"
    virtual void Filter(){};
    // podmíněná dilatace
    virtual void ConditionalDilatation(){};
    // hledání hranic objektu
    virtual void Outline(){};
};
```

5.2.6. Průměrování z více snímků

Tato položka (Image / Averaging) je přístupná i v případě, kdy nemáme doposud otevřený žádný soubor. Nejdříve postupně vybereme všechny obrázky, které chceme zprůměrovat. Vybírat je můžeme po jednom, nebo jich vybrat i více najednou. Všechny načítané obrázky musí mít stejnou velikost, program neumožní načtení obrázků různých velikostí.

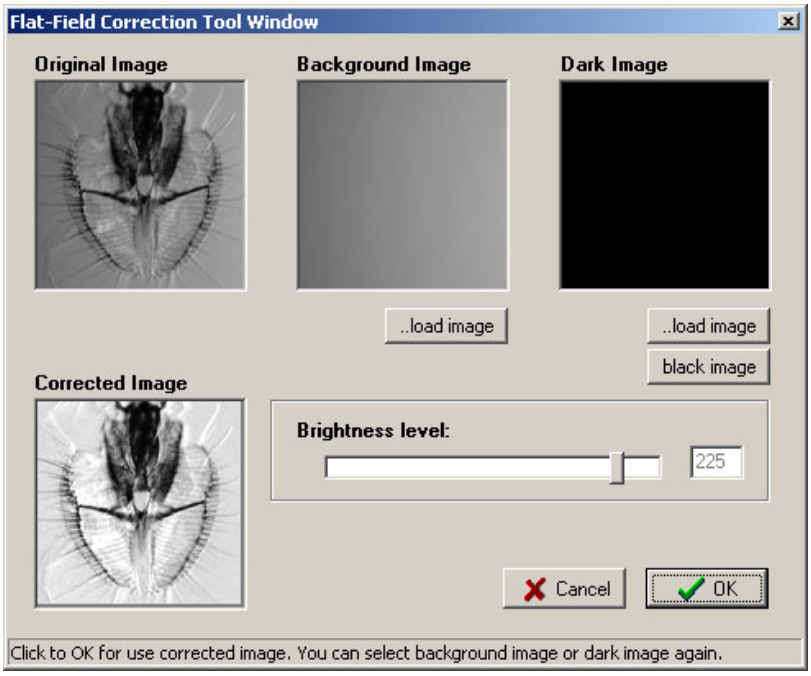
```
class TAverageForm : public TForm{
__published:
    // provádí průměrování
    void __fastcall ArithmeticAverage();
public:
    // kontroluje stejný rozměr všech obrazů určených k průměrování
    bool __fastcall Check_Validity(AnsiString file_path);
};
```

5.2.7. Jasové korekce

Flat-Field korekce

Funkce je umístěna v nabídce menu (Image / Correction / Flat-Field). Vstupní obraz je načten z hlavního okna programu. Nejdříve zvolíme snímek pozadí. Máme-li k dispozici snímek za tmy (se zatemněným snímačem), načteme ho také. Nemáme-li ho, použijeme snímek předdefinovaný (černý obraz).

Po zvolení všech tří snímků se vypočte Flat-field korekce a zobrazí se výsledný obraz. Poté můžeme měnit dynamický rozsah výsledného obrazu - konstanta M ve vztahu (25).

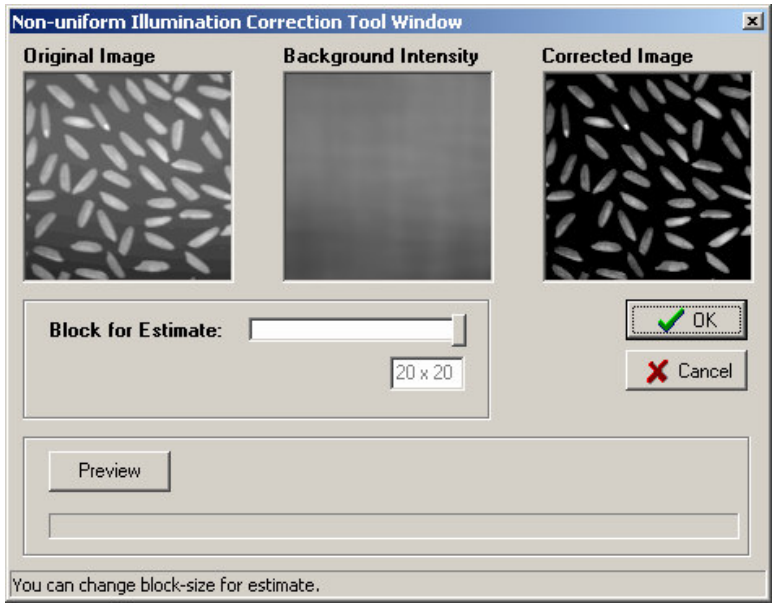


Obr. 16. Ukázka Flat-Field korekce

```
class TFlatFieldForm : public TForm{
public:
    // aritmetické operace mezi obrazy
     void __fastcall FlatFieldSubtract(…);
     // výpočet flat-field, volá se funkce FlatFieldSubtract
     void __fastcall FlatFieldCompute();
};
```

Korekce nehomogenního osvětlení

Položka menu (Image / Correction / Non-uniform Illumintion). Postup výpočtu je vysvětlen v kapitole 4.2.3. Velikost dílčích čtverců volíme v intervalu 5x5 až 20x20 obrazových bodů. Tlačítkem ‘Preview‘ se zobrazí zmenšený náhled po korekci. Tlačítkem ‘OK‘ se vypočte korekce pro obraz v původní velikosti. Tato operace je časově náročnější, dlouhou dobu trvá vytváření modelu pozadí.



Obr. 17. Ukázka korekce nehomogenního osvětlení

```
class TNonUniformIlluminationForm : public TForm{
published:
    // výpočet snímku pozadí a odečtení od původního obrazu
    void __fastcall ComputeBackground(…);
};
```

Gama korekce

V menu zvolíme (Image / Correction / Gamma Correction). V pracovním okně lze měnit vstupní a výstupní rozsah jasových úrovní pro které bude gamma korekce provedena. Konstantu γ lze měnit v rozsahu 0,1 až 9,99.

```
class TGammaCorrectionForm : public TForm
{
private:
    // výpočet náhledu
    void __fastcall ComputePreview();
    // výpočet výsledného obrazu
    void __fastcall ComputeFinalImage();
    // výpočet transformační fnkce
    void __fastcall ComputeTransform();
    // zobrazení původního histogramu
    void __fastcall ShowHistogram();
    // zobrazení histogramu po provedení gama korekce
    void __fastcall ShowModifHistogram();
};
```

5.2.8. Filtrace pomocí Fourierovy transformace

Metody doplnění obrázku

Chceme-li použít výpočet DFT pomocí FFT, je vyžadováno, aby měl obraz rozměr $2^k \times 2^k; k \in N$. Pokud by neměl obraz tento rozměr, bylo by možné použít DFT, výpočet by však trval déle. V programu je implementováno několik způsobů transformace na větší rozměr. Původní obraz zůstane nezměněn a je na nejbližší rozměr $2^k \times 2^k$ doplněn:

- nulami (černou plochou)
- průměrnou hodnotou jasu obrazu
- zrcadlením obrazu
- periodickým prodloužením obrazu
- opakováním okrajové hrany obrazu

Typy implementovaných filtrů

Po provedení přímé FFT zvolíme typ frekvenčního filtru a nastavíme jeho vlastnosti - hraniční frekvence, intenzitu, případně další konstanty. Pro snadnější návrh filtru lze ve spektru zjišťovat souřadnice. Stačí přesunout kurzor myši na místo, které nás zajímá a příslušná souřadnice se zobrazí v dolní části okna.

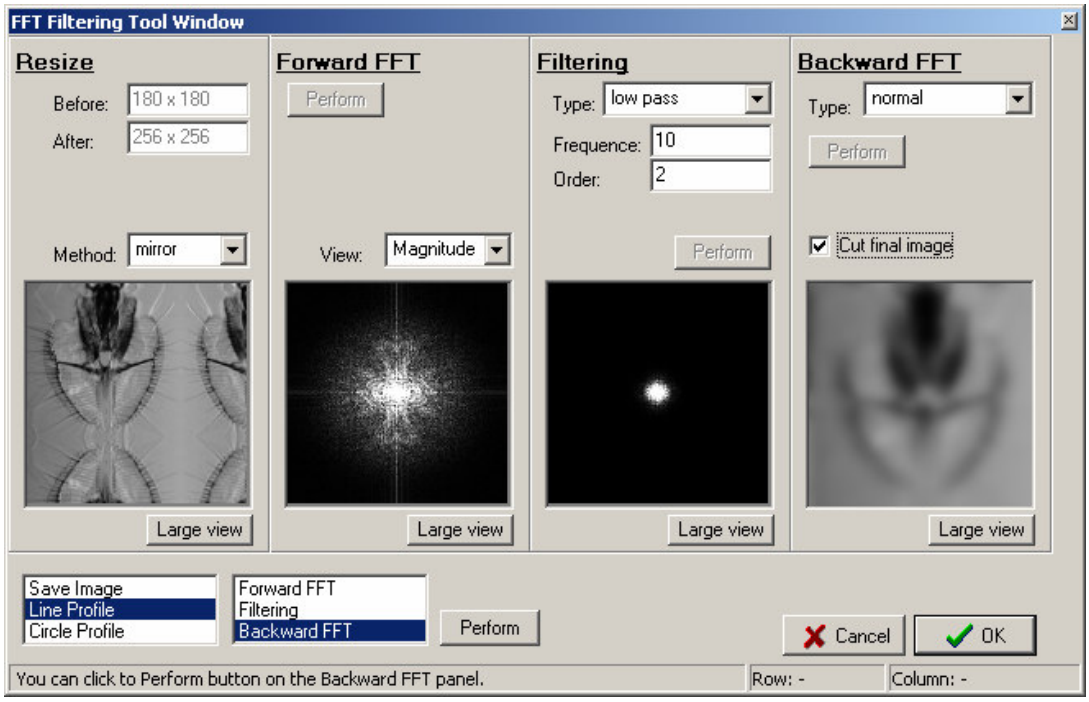
V programu jsou zahrnuté filtry:

- dolní propust
- horní propust
- násobení konstantou
- horizontální nůžky (motýlek)
- vertikální nůžky (motýlek)
- pásmová propust
- pásmová zádrž
- vymazání stejnosměrné složky

Zobrazení

Můžeme si zvolit, zda chceme po provedení přímé FFT vykreslit reálnou či imaginární složku, amplitudu nebo fázovou složku. V každém kroku lze zobrazit

výsledek po dané operaci ve zvětšeném okně. Po zpětné FFT je možné výsledný obraz ořezat, aby velikostí odpovídal obrazu původnímu.



Obr. 18. Ukázka filtrace pomocí 2D DFT

```
class TFftFiltrationForm : public TForm{
public:
    // metody doplnění obrazu
    void __fastcall FindOptimalSize();
    void __fastcall NoResizeMaybe();
    void __fastcall MakeMirror();
    void __fastcall MakePeriodic();
    void __fastcall MakeRepetition();
    // přímá a zpětná FFT
    void __fastcall FFTForward(DFTOptions &options);
    void __fastcall PerformBackwardFftButtonClick(TObject *Sender);
    // nastavení FFT a vlastností filtru
    void __fastcall SetDFTOptions(DFTOptions &opt);
    void __fastcall SetFilterOptions(DFTOptions &opt);
    // předdefinované filtry
    void __fastcall PerformLowPass();
    void __fastcall PerformHighPass();
    void __fastcall PerformMultiple();
    void __fastcall PerformScissorsHorizont();
    void __fastcall PerformScissorsVertical();
    void __fastcall PerformBandPass();
    void __fastcall PerformBandStop();
};
```

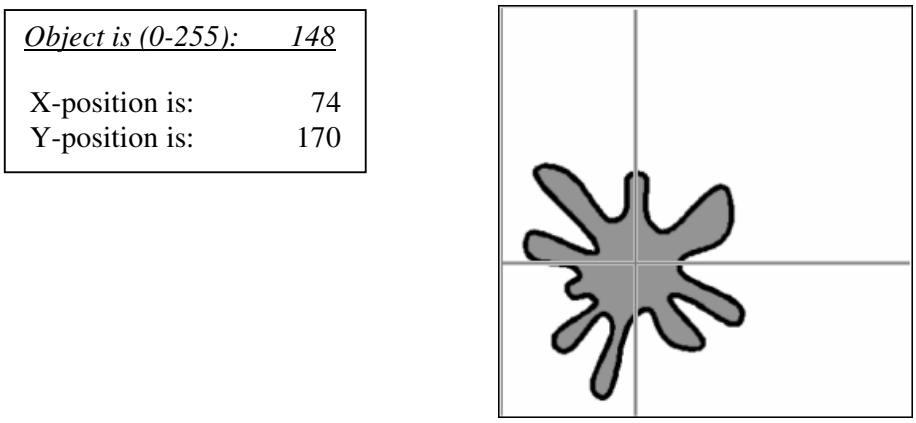
Uložení spektra, zobrazení jasového profilu

Po vygenerování jednotlivých náhledů se zpřístupní další funkce. Je možné uložit obraz představující spektrum, a to jak před použitím filtru, tak i po jeho aplikaci. Dále lze ve spektru zobrazit jasový profil po přímce, či po kružnici. Pro provedení vybereme jednu z těchto funkcí, poté obraz na který ji chceme aplikovat a stiskneme tlačítko 'Perform'.

5.2.9. Charakteristické výpočty

Centroid

Položka menu (Image / Characteristics / Centroid). V pracovním okně této funkce si nadefinujeme, jakou úroveň šedi má objekt u kterého chceme zjistit těžiště. Nalezené těžiště je vykresleno přímo do obrázku a současně je vypsána jeho X-ová a Y-ová souřadnice.



Obr. 19. Příklad nalezení těžiště objektu

```
class TCentroidForm : public TForm{
published:
    // nastavení jasové hodnoty objektu, jehož těžiště hledáme
    void __fastcall PercentUpDownClick(…);
public:
    // výpočet a zobrazení těžiště objektu v obraze
    void __fastcall ComputeCentre();
};
```

Statistické výpočty

Po spuštění (Image / Characteristics / Statistics) se vypočítá minimální a maximální hodnota jasu, průměrný jas, rozptyl, směrodatná odchylka a entropie. Všechny tyto vypočtené hodnoty se zobrazí s přesností na 3 desetinná místa.

```
class TStatisticsForm : public TForm
{
public:
      // výpočet průměrné hodnoty jasu
      void __fastcall ComputeMean(GreyImage* image);
      // výpočet rozptylu a směrodatné odchylky
      void __fastcall ComputeDispersion(GreyImage* image);
      // hledání maximálního a minimálního jasu v obraze, výpočet entropie
      void __fastcall ComputeEntropy(GreyImage* image);
};
```

SNR / PSNR

Zvolíme (Image / Characteristics / SNR/PSNR). Z hlavního okna programu se načte obraz. V pracovním okně zadáme umístění druhého obrazu. Poté se vypočítají hodnoty SNR a PSNR. Obě hodnoty se zobrazí, zobrazí se také chybový rozdílový obraz.

```
class TSnrPsnrForm : public TForm
{
public:
      // výpočet SNR
      void __fastcall ComputeSNR();
      // výpočet PSNR
      void __fastcall ComputePSNR();
      // Výpočet a zobrazení chybového obrazu
      void __fastcall ComputeAndShowErrorImage();
};
```

5.3. COLOR Menu

Pro práci s barevnými obrazy slouží třída *ColorImage*. Umožňuje načíst barevný obraz, pracovat s ním a zpětně ho uložit do souboru. Jsou zde také nadefinované transformace mezi různými barevnými prostory (RGB, HSL, HSV, CMY, CMYK), včetně převodu barevného obrazu do šedotónového.

```
class ColorImage : public Image<unsigned int>{
public:
    // constructor
    ColorImage() :Image<unsigned int>() {};
    // destructor
    virtual ~ColorImage() {};
    // jednotlivé transformace pro barevný obraz
    void __fastcall SplitToRGB(…);
    void __fastcall CombineFromRGB(…);
    void __fastcall GreyScale(int type_conversion, GreyImage* Y);
    void __fastcall RGBtoHSL(…);
    void __fastcall RGBtoHSV(…);
    void __fastcall RGBtoCMY(…);
    void __fastcall RGBtoCMYK(…);
    void __fastcall HSLtoRGB(…);
    void __fastcall HSVtoRGB(…);
    void __fastcall CMYtoRGB(…);
    void __fastcall CMYKtoRGB(…);
};
```

5.3.1. Operace nad jednotlivými barevnými složkami

Barevný obraz lze rozložit do jednotlivých šedotónových složek, zastupující červenou, modrou a zelenou barvu v obraze (Color / Split to RGB). Po vybrání jedné z těchto složek nad ní můžeme provádět všechny operace popisované pro šedotónový obraz. Výjimkou jsou samozřejmě operace, které mění velikost obrazu. Ty nejsou k dispozici. Po každé změně dojde k sloučení všech tří složek do výsledného barevného obrazu a k jeho zobrazení.

```
class TRGBComponentsForm : public TForm
{
public:
    // výběr barevného kanálu, který chceme editovat
    void __fastcall BitBtn1Click(TObject *Sender);
    void __fastcall BitBtn2Click(TObject *Sender);
    void __fastcall BitBtn3Click(TObject *Sender);
    // z upravených RGB kanálů se vypočte výsledný barevný obraz
    void __fastcall AssignChangedImage(GreyImage _image, int _type);
    // zobrazení položek v menu pro barevný obraz
    void __fastcall PrepareMenuEdit();
    // zobrazení položek v menu pro šedotóvý obraz
    void __fastcall PrepareMenuColor();
};
```

5.3.2. Barevné prostory

Po zvolení nabídky z menu (Color / Adjust / HSL, HSV, CMY či CMYK) se provede převod původního obrazu do zvoleného barevného prostoru. Poté můžeme libovolně měnit intenzitu jednotlivých složek (například odstín, nasycení a světlost u modelu HSL). Nově vzniklý obraz se po každé změně převede zpět do RGB prostoru a zobrazí se jeho náhled.

class TRGBToolForm : public Tform{};
class THSLToolForm : public TForm{};
class THSVToolForm : public Tform{};
class TCMYToolForm : public Tform{};
class TCMYKToolForm : public Tform{};

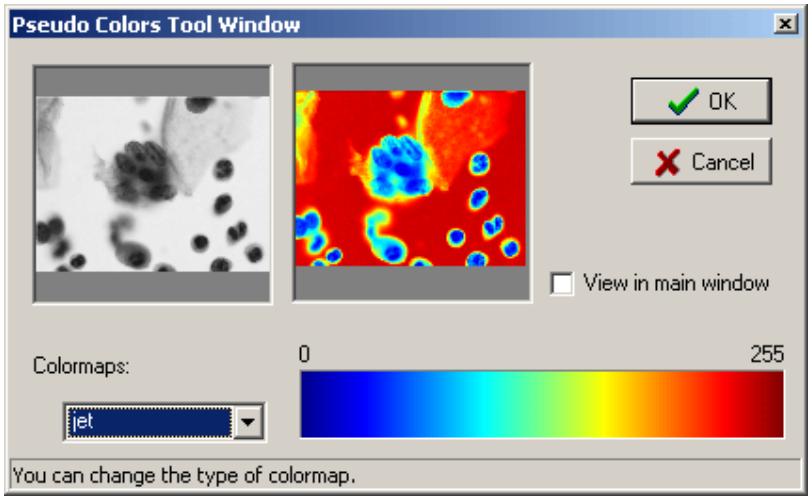
5.3.3. Pseudobarevné obohacení obrazu

Tato funkce je dostupná pouze u šedotónových, nebo 256-ti barevných obrázků. Při spuštění (Color / PseudoColors) je k dispozici několik předdefinovaných barevných palet:

- bone
- copper
- hot
- jet
- HSV
- flag

Při výběru jedné z nich se v obrázku změní informace v LUT tabulce, kde je informace o tom jaká barva odpovídá dané jasové úrovni. Změní se tedy pouze informace o tom, jak má být daný obrázek zobrazen. Např. šedotónový obrázek je vždy možné po aplikaci barevné palety zobrazit opět jako šedotónový.

K dispozici je i možnost vytvořit si vlastní barevnou paletu (poslední položka v nabídce předdefinovaných palet). Můžeme si zvolit jasový rozsah, který se zobrazí vybranou barvou, nebo použít gradientní přechod od jedné barvě k druhé.



Obr. 20. Pseudobarevné obohacení obrazu

```
class PseudoColor{
enum Palette_Type { GREY, BONE, COPPER, HOT, JET, HSV, FLAG};

public:
    // na obraz se použije vytvořená barevná paleta, volá se funkce Transform
    void __fastcall CreateImageFromPalette(int palette[3][256],…)
    void __fastcall Transform(GreyImage* image1,…, int palette[3][256]);
    // změna aktuální palety barev
    void __fastcall ChangePalette(int palette[3][256], Palette_Type pal);
    // převod čísla 0 až 16.7milionů na RGB složky (0-255)
    void __fastcall IntToRBG(int NotRGB, int RGB[3]);
    // vytvoří se gradientní přechod v barevné paletě
    void __fastcall Gradient(int first_level, int second_level, int first_color[3],
    int second_color[3]);
};
```

6. Přínos pro výuku

V této kapitole uvádím několik praktických ukázek, jak lze použít program MIPS k ilustraci principů při výuce. Ovládací prvky jsou voleny tak, aby bylo možné co nejsrozumitelněji popsat danou problematiku. Algotitmus výpočtu a způsob zobrazení často odpovídá postupu, užívaným ve škole při popisu funkce jednotlivých operací.

Zobrazení obrazu

Pokud přesouváme ukazatel myši v hlavním okně přes zobrazený obraz, v dolní části se zobrazuje souřadnice (řádek, sloupec) a hodnota jasu příslušného pixelu. Lze ukázat, že černá barva odpovídá hodnotě 0, bílá hodnotě 255. Je-li načtený obraz

barevný, zobrazují se čísla tři, každé představuje příspěvek jedné z barev (červená, zelená, modrá).

Aritmetické a logické operace

Načteme dva obrazy a zvolíme logickou operaci. Nejdříve vybereme možnost, kdy logické 1 odpovídá černá a logické 0 odpovídá bílá. Ve druhém případě barvy prohodíme. Lze ukázat, že výsledný obraz bude pokaždé jiný.

Můžeme se také pokusit o odečtení pozadí. Mějme obraz, kde jsou objekty s jasem 100 až 255 umístěné na tmavém nehomogenním pozadí. Použijeme aritmetickou funkci odečtení konstanty. Konstantu volíme 255, rozsah pro který se tato operace provede volíme 0 až 100. Výsledný rozsah hodnot jasů ošetříme limitací (hodnoty menší než 0 se zobrazí jako 0).

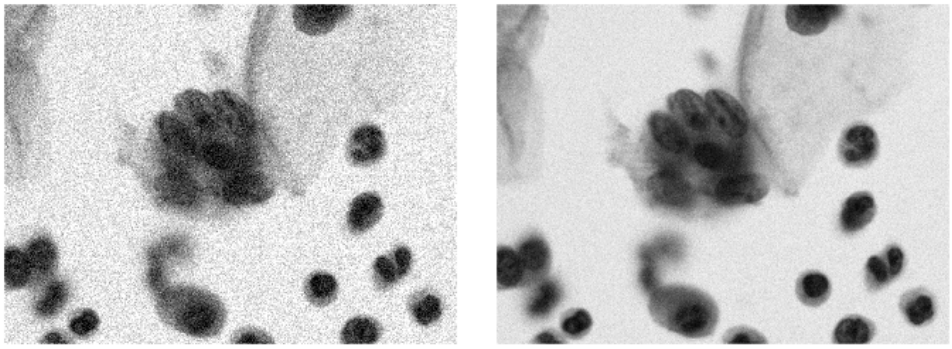
Velmi názorné z hlediska výuky je možnost ilustrovat přetečení a podtečení rozsahu 0-255.

Práce s histogramem

Po provedení jasové transformace (pomocí LUT tabulky) máme možnost okamžitě zobrazit histogram obrazu. Můžeme tak například měnit jas, či kontrast obrazu a pozorovat jaký to má vliv na výsledný histogram.

Průměrování více snímků

Mějme několik snímků, které obsahují šum s normálním rozložením (způsobený například CCD snímačem). Na následujícím obrázku je zobrazeno průměrování deseti takových snímků. I v takovémto malém souboru je již vidět snížení velikosti šumu ve výsledném obraze.



Obr. 21. Průměrování: jeden z 10-ti vstupních snímků (vlevo), výsledný obraz (vpravo)

Pseudobarevné obohacení obrazu

Máme možnost si nadefinovat vlastní paletu barev. Jasovým úrovním 0 až 128 můžeme přiřadit jdnu barvu, jasovým úrovním 129 až 255 barvu jinou. Výsledný obraz bude poté obsahovat pouze tyto 2 barvy (spojitost s prahováním, podobně lze ukázat i víceúrovňové prahování).

Pomocí této funkce si lze také nadefinovat konkrétní barvu k určitému jasu, k určité oblasti, nebo dokonce gradientní přechod mezi zvolenými úrovněmi jasu. Takto můžeme v obraze zvýraznit a barevně odlišit objekty, které mohly být v původním obraze pouhým okem nerozlišitelné.

Odstranění stejnosměrné složky v obraze pomocí FFT filtrace

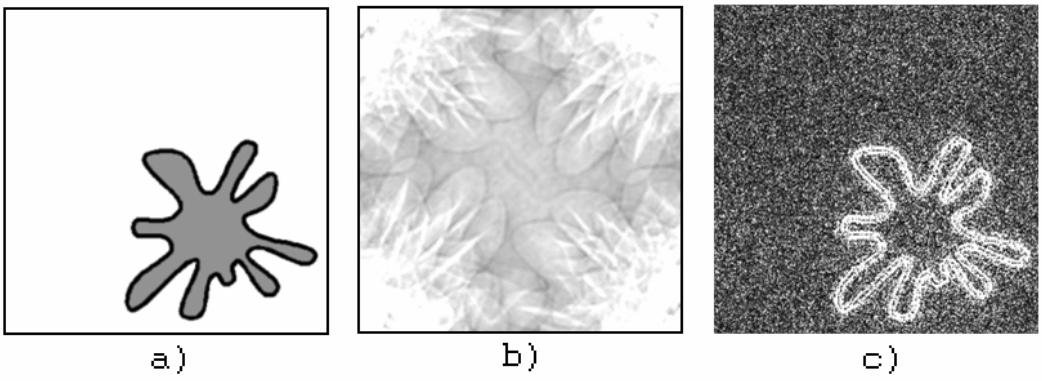
Pomocí 2D DFT převedeme obraz do frekvenčního oblasti. Zde změníme středový bod (představující nejnižší prostorové frekvence v obraze) na nulu. Po zpětné 2D DFT nebude obraz obsahovat stejnosměrnou složku (namísto pozadí a velkých objektů v obraze bude černá plocha).

Zachování amplitudy, či fáze

Před provedením zpětné FFT si můžeme zvolit, jak se má provést. Nabízí se:

- Normální režim
- Konstantní amplituda
- Konstantní fáze

Výsledek je přehledně zobrazen na následujícím obrázku:



Obr. 22. Obraz po zpětné FFT: a) původní obraz, b) konstantní fáze, c) konstantní amplituda

7. Závěr

Problematika předzpracování a úpravy snímků z mikroskopu je velice rozsáhlá. Tato diplomová práce pokrývá nejdůležitější operace, se kterými se můžeme setkat při zpracování snímků pořízených digitálním mikroskopem. Všechny operace mají jednu společnou vlastnost - snažíme se z obrazu získat informaci, která není na první pohled zřejmá a vhodným způsobem ji zobrazíme.

Při programování jsem se snažil vytvořit jednoduše a pochopitelně realizovaný nástroj, který je možno použít pro objasnění dané problematiky. MIPS lze použít jako pomocný prostředek k předvádění nejrůznějších principů v této oblasti. Od začátku jsem se snažil zachovat modularitu programu, je proto možné ho doplnit o další nové funkce.

Program jsem si prakticky ověřil, když jsem vedl praktika na kurzu *Získání a zpracování obrazu v mikroskopii*. Tento několikadenní kurz je pravidelně pořádán mikrobiologickým ústavem Akademie věd. Obsahoval vše, s čím se můžeme v mikroskopii setkat. Metody zpracování obrazu byly jeho nedílnou součástí.

Všechny části zadání diplomové práce byly splněny, několik operací bylo dodatečně přidáno. Některé vyplynuly z požadavků na zmiňovaný kurz mikroskopie, jiné posloužily jako vhodná součást programu. Přesto vše by bylo možné každou z dílčích částí MIPS dále rozšířit a navázat na ni dalšími pokročilejšími metodami zpracování obrazu.

8. Literatura

[1] Pop, M. *MIPS - Medical Image Processing Software*. Diplomová práce. ČVUT - Fakulta elektrotechnická. Praha, 2000.

[2] Gonzalez R.C., Woods R.E. *Digital Image Processing*. Reading – Ma. Addison-Wesley, 1992.

[3] Rosenheimer L., Roper Scientific GmbH. *Flat Field Correction*. Germany [online]. Poslední revize 2004-05-11 [cit.2004-05-20]. <http://www.roperscientific.de/tflatfield.html>.

[4] Klíma M., Bernas M., Hozman J., Dvořák P. *Zpracování obrazové informace*. Praha: ČVUT, 1996.

[5] Hlaváč V., Sedláček M. *Zpracování signálů a obrazů*. Praha: ČVUT, 2002

[6] Murray J.D., Van Ryper, W. *Encyklopedie grafických formátů*. Druhé vydání. Brno: Computer Press, 1997.

[7] Michael W. Davidson, The Florida State University. *Molecular Expressions* [online]. Poslední revize 2004-12-11 [cit.2004-05-20] <http://micro.magnet.fsu.edu>.

[8] Halley R.Myler, Artur R.Weeks. *The Pocket Handbook of Image Processing Algorithms in C*. Prentice Hall. New Jersey, 1993

[9] Sobotka Z., Šorm R. *Základy digitálního zpracování obrazu*. Druhé vydání. Praha: ODIS VTEI TESLA VÚST, 1986

[10] Varde A. *Ameya Varde's Homepage*. [online] Poslední revize 2004-02-15 [cit.2004-12-14] <http://members.tripod.com/~ameya_varde/403.htm>.

[11] Centrum experimentální geotechniky. *Mikroskopie*. Praha [online] Poslední revize 2003-10-03 [cit.2005-01-06] <http://ceg.fsv.cvut.cz/CZ/ceg-vyzkum/mikroskopie.htm>.

[12] Ústav biologie Lékařské fakulty Univerzity Palackého. *Metody mikroskopie*. Olomouc [online] Poslední revize 2005-01-06 [cit.2005-01-06] <http://biologie.upol.cz/mikroskopie>.

[13] The Regents of the University of Kalifornia. *Image Quality Computation*. Kalifornia [online] Poslední revize 1997-12 [cit.2005-01-09] <http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html>.

[14] Šára R. *Mathematical Morpholog*. Praha [online] Výukový materiál k předmětu Digitální zpracování obrazu na ČVUT FEL. [cit. 2005-01-06] <http://cmp.felk.cvut.cz/cmp/courses/dzo/resources/lecture_morphology_sara.pdf>.

[15] The MathWorks, Inc. *Matlab 6.5. Toolbox Image Processing - Correction of Non-uniform Illumination*. [CD-ROM] United States, 2002.

[16] Quantitative Imaging Group. *Histogram-based Operations*. Netherlands [online] Poslední revize 2004-08-09. [cit.2005-01-06] <http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-istogram.html>.

[17] Sachs J., Digital Light & Color. *Digital Image Basics*. [online] Cambridge, Poslední revize 1999. [cit.2005-01-06] <http://www.dl-c.com/basics.pdf>.

[18] Poynton Ch. *Frequently Asked Questions about Color*. Toronto [pdf] Poslední revize 2004-04-02. [cit.2005-01-06] <http://www.poynton.com/PDFs/ColorFAQ.pdf>.

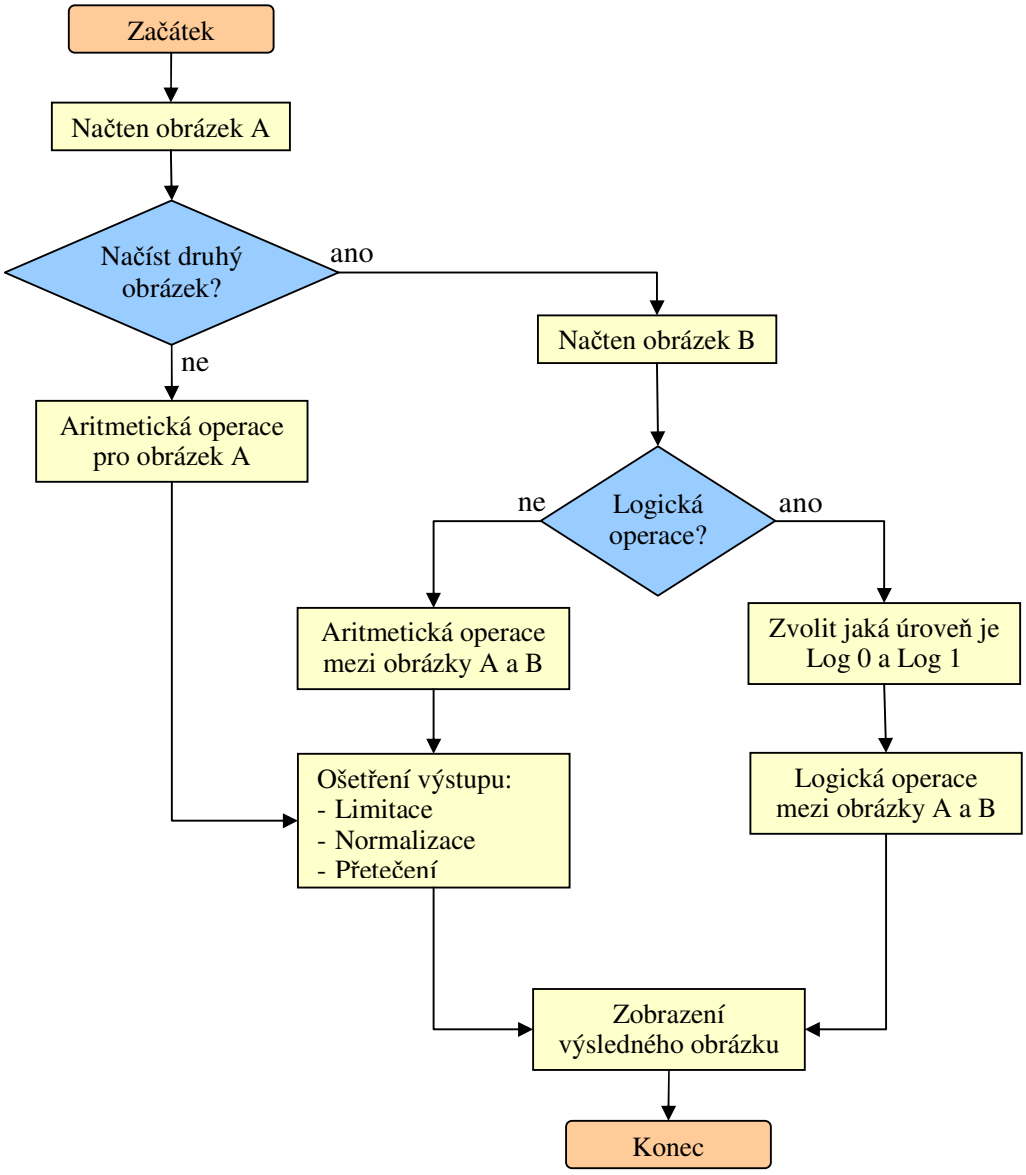
[19] Mortimer Abramowitz. *Microscope - Basics and Beyond*. New York [online] Poslední revize 2005-01-02 [cit.2005-01-06] http://micro.magnet.fsu.edu/primer/pdfs/basicsandbeyond.pdf.

[20] Rogalewicz, V. *Pravděpodobnost a statistika pro inženýry*. 2. vydání, Praha: ČVUT, 2000.

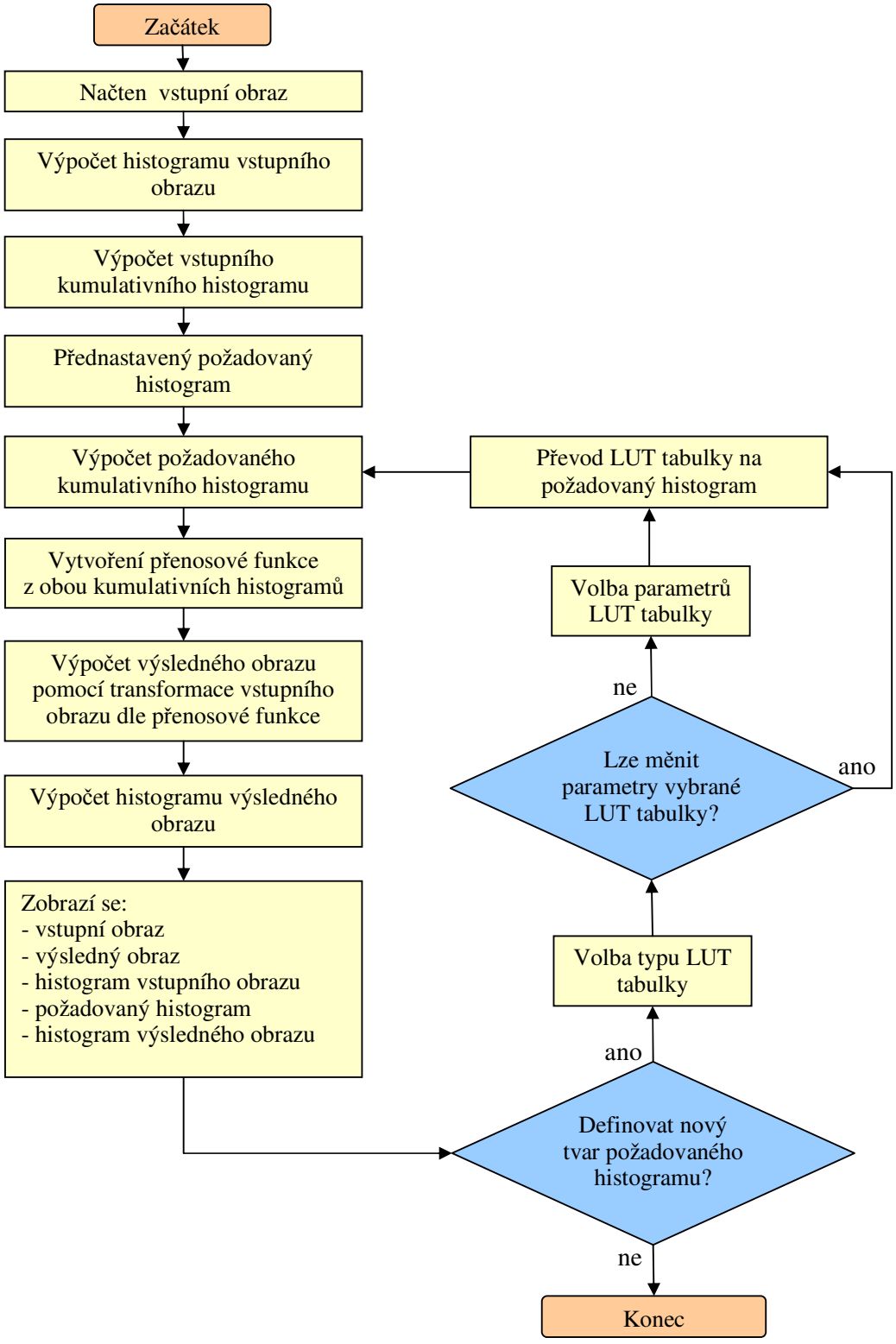
9. Přílohy

A. Vývojové diagramy

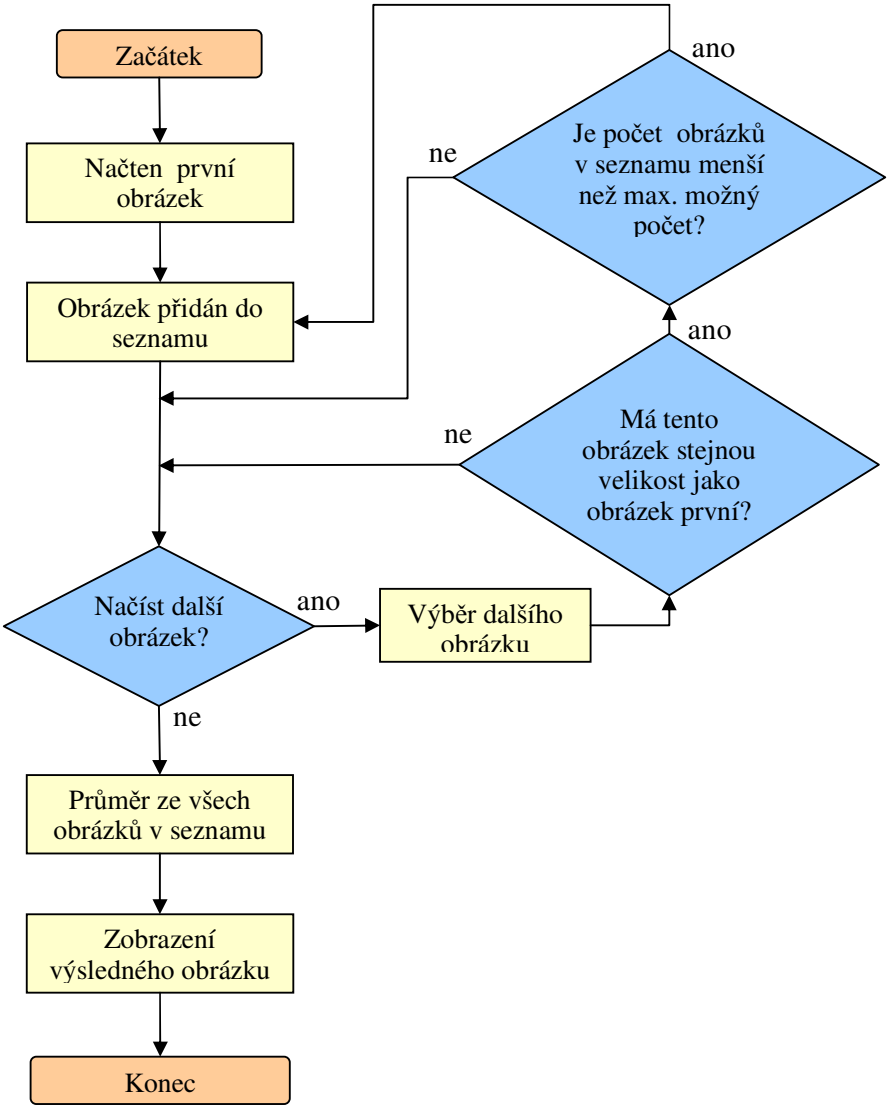
Aritmetické a logické operace



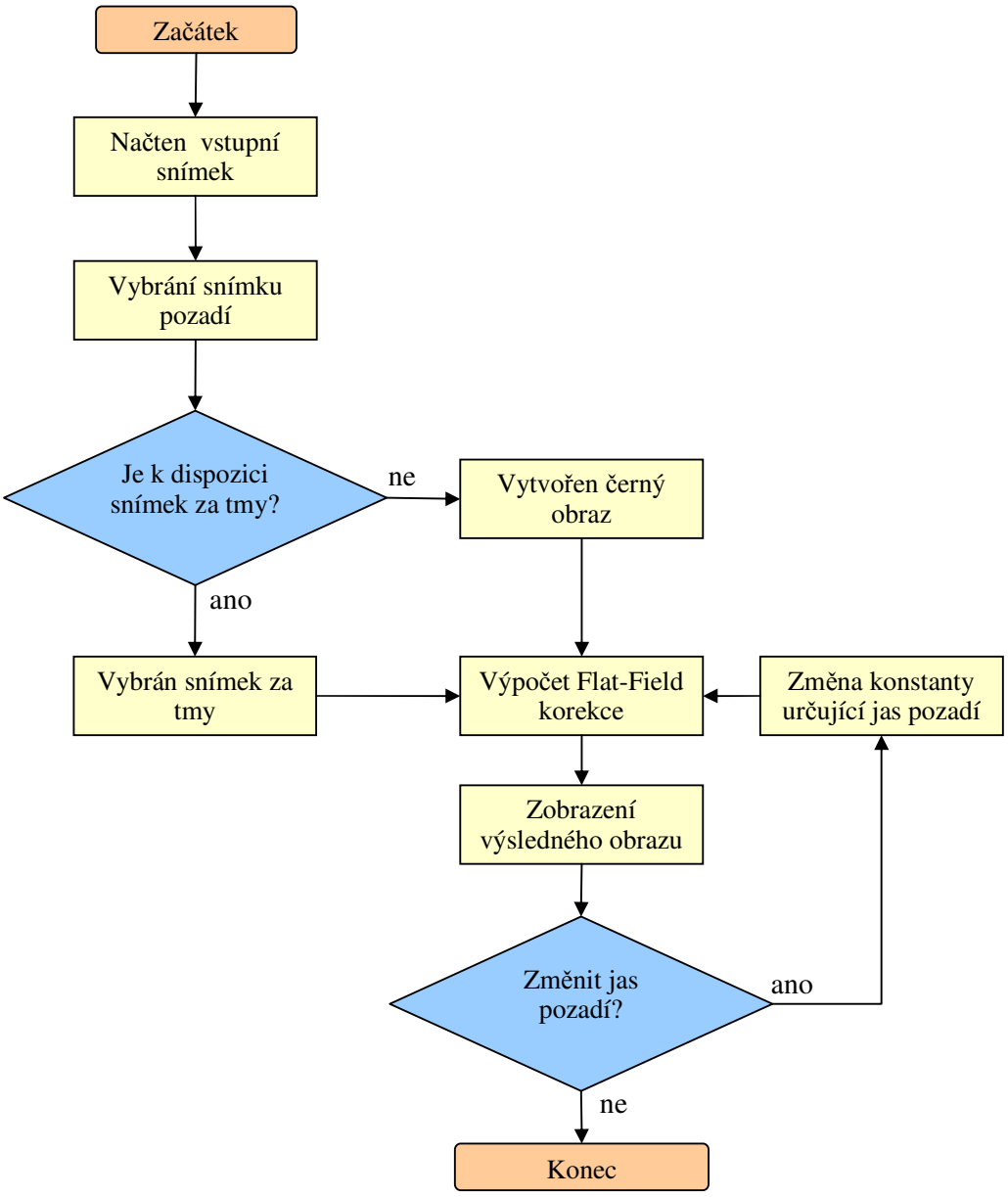
Specifikace histogramu



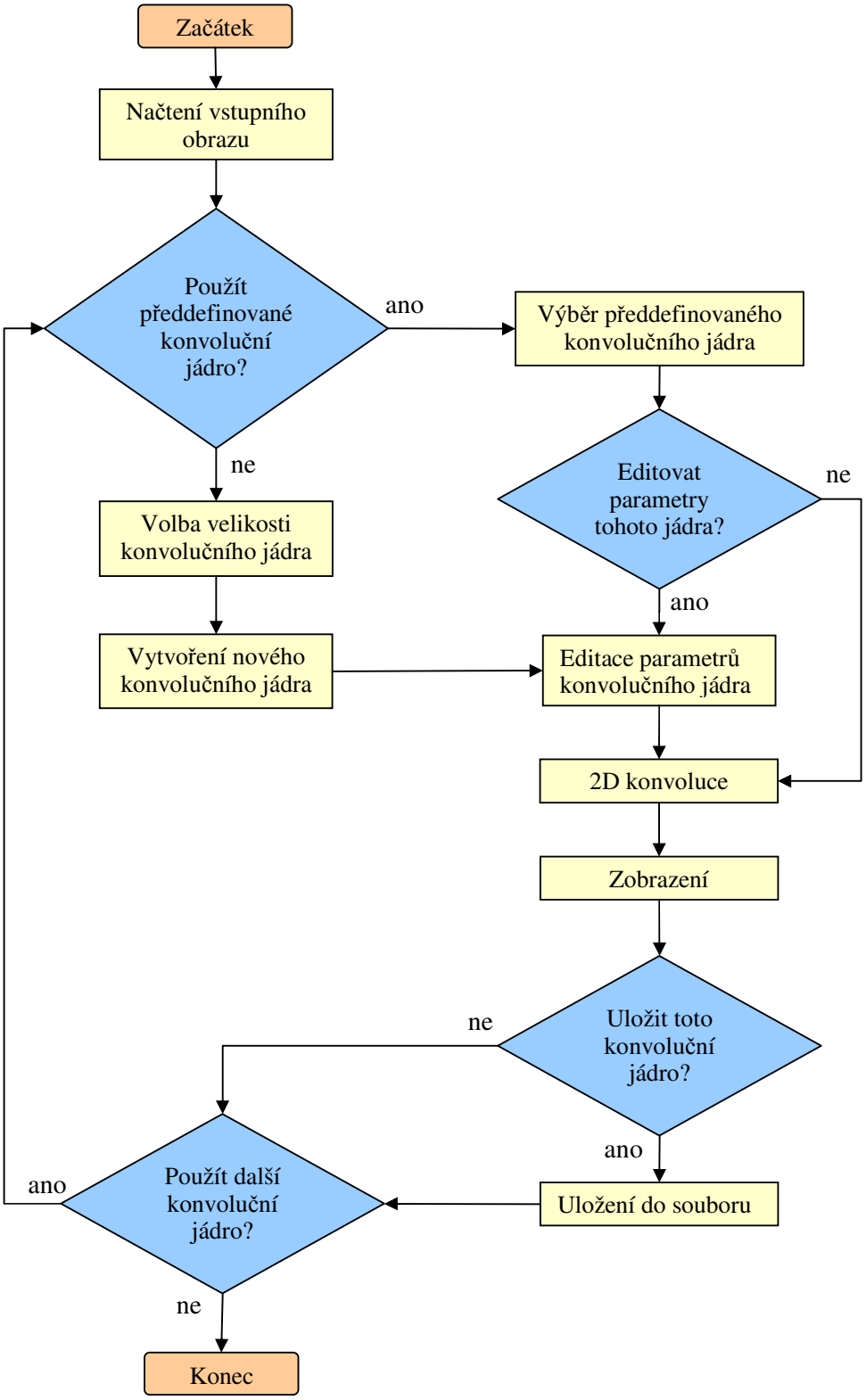
Průměrování



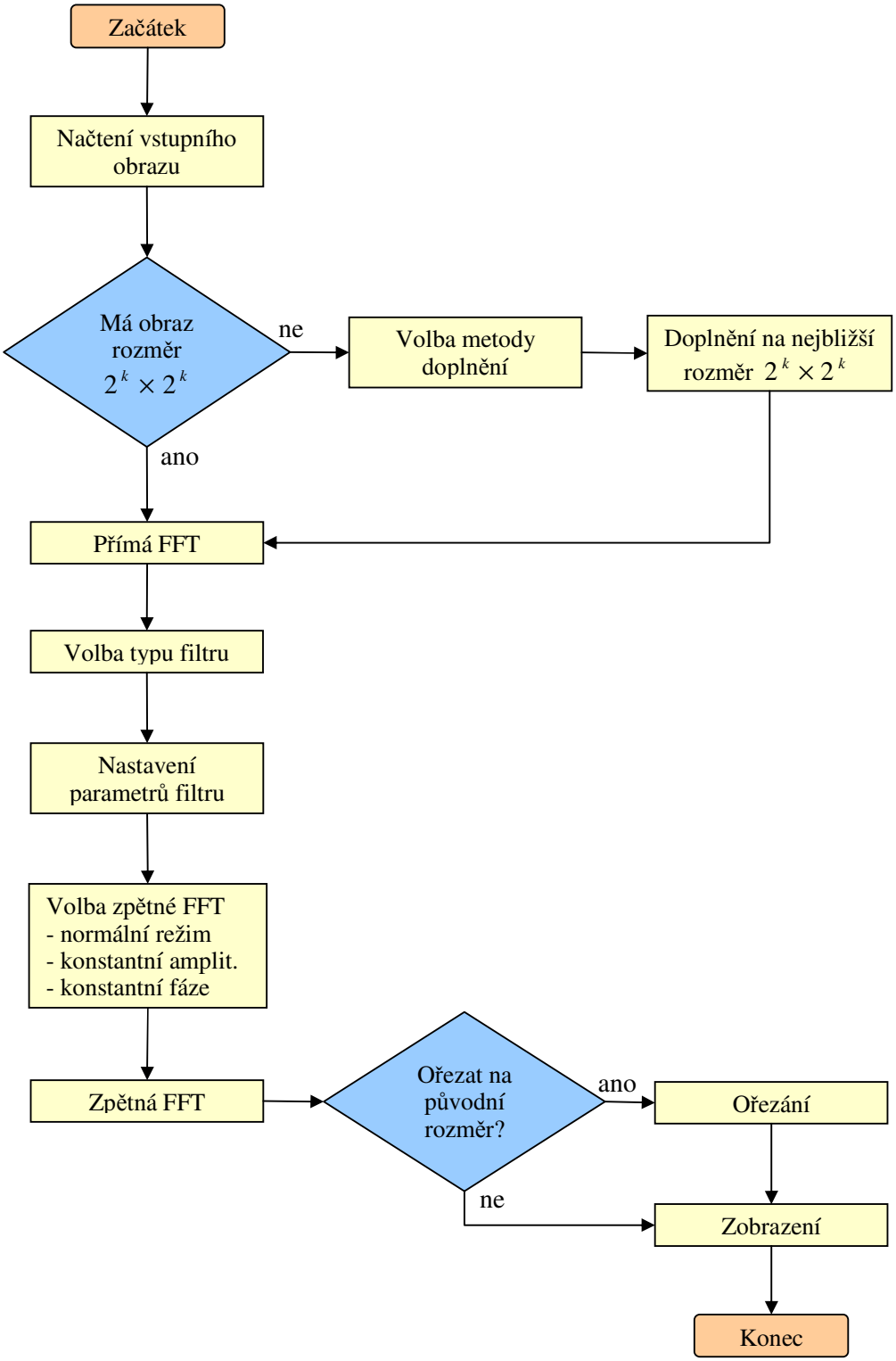
Flat-Field korekce



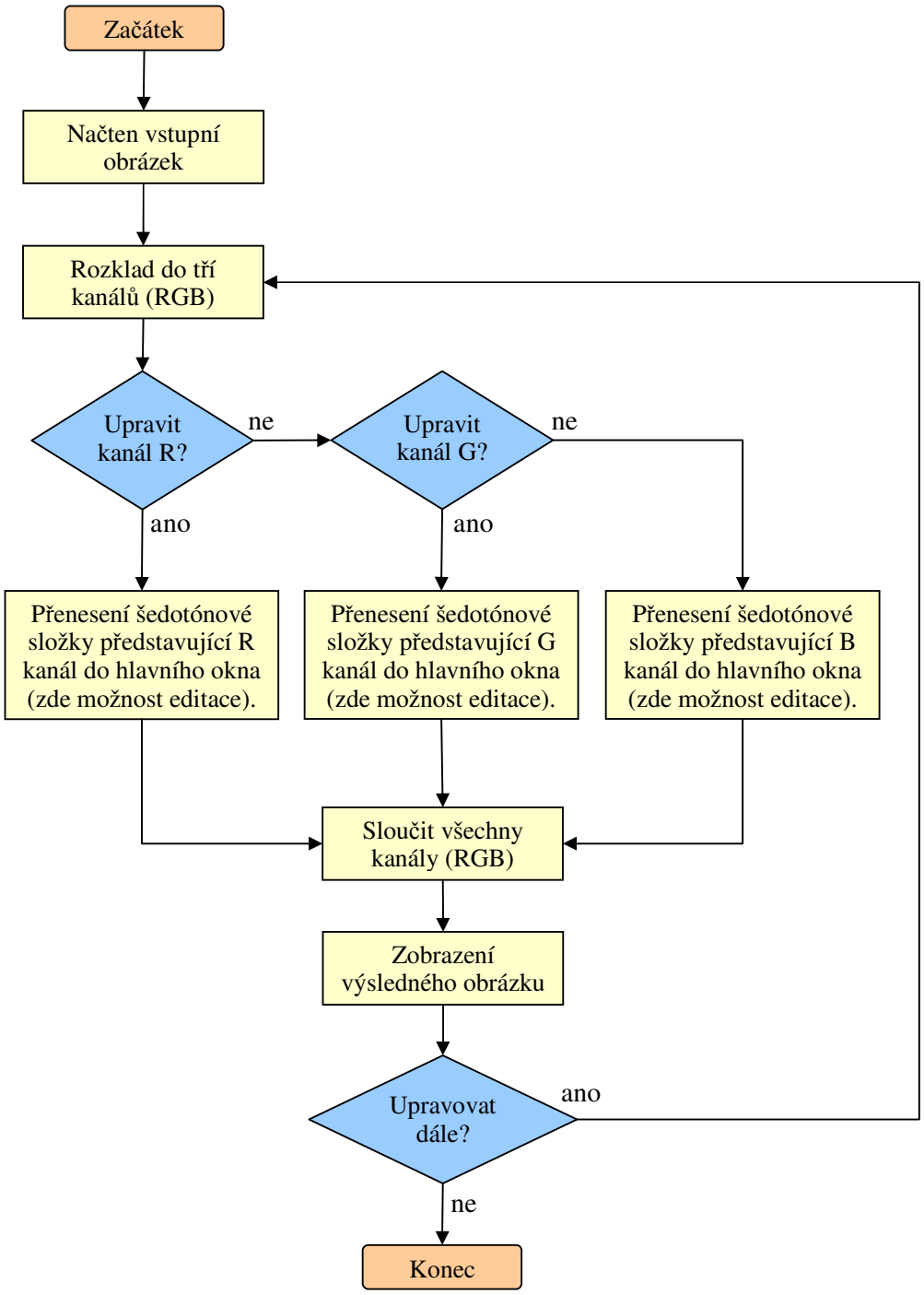
Dvojrozměrná konvoluce



Filtrace pomocí Fourierovy transformace



Operace nad jednotlivými barevnými složkami



Pseudobarevné obohacení obrazu