TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

DIPLOMOVÁ PRÁCE

Rozpoznávání vláken v řezech přízí metodou analýzy obrazu

Fiber recognition in yarn slices using image analysis

Liberec 2005 **Jiří Horčička**

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Katedra měření Akademický rok: 2004/2005

ZADÁNÍ DIPLOMOVÉ PRÁCE

pro: Jiřího Horčičku

studijní program: M 2612 – Elektrotechnika a informatika

obor: 3902T005 – Automatické řízení a inženýrská informatika

Vedoucí katedry Vám ve smyslu zákona o vysokých školách č.111/1998 Sb. určuje tuto diplomovou práci:

Název tématu:

Rozpoznávání vláken v řezech přízí metodou analýzy obrazu

Zásady pro vypracování:

1. Seznamte se s problematikou získávání a snímání řezů přízí. Seznamte se stávajícími způsoby získávání informací z řezů dosavadními metodami a proveďte rozbor.
2. Seznamte se s algoritmy počítačového zpracování obrazu a metodami rozpoznávání
3. Vypracujte v prostředí Matlab sadu funkcí, které budou schopny usnadnit a zautomatizovat analýzu řezu přízí prostřednictvím algoritmů počítačového zpracování obrazů a metod rozpoznávání.
4. Zpracujte dokumentaci navržených funkcí a algoritmů v potřebném rozsahu

Rozsah grafických prací: dle potřeby dokumentace

Rozsah průvodní zprávy: cca 40 až 50 stran

Seznam odborné literatury:

[1] Šonka, M., Hlaváč, V., Boyle, R.: Image Processing, Analysis, and Machine Vision, PWS, Boston, USA, 1998

[2] Hlaváč, V., Sedláček, M.: Zpracování signálů a obrazů, ČVUT FEL Praha 2000

[3] http://www.mathworks.com/, zvláště část Image Processing Toolbox

[4] Fischer, J.: Optoelektronické senzory a videometrie, ČVUT FEL Praha 2002

Vedoucí diplomové práce:	Doc. Ing. Ivan Jaksch, CSc.
Konzultant:	Ing. Lukáš Matela
	Ing. Gabriela Krupincová

Zadání diplomové práce:	--. --. ----
Termín odevzdání diplomové práce:	--. --. ----

L.S.

....

Vedoucí katedry Děkan

V Liberci dne

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Anotace

Cílem diplomové práce bylo navrhnout funkce založené na počítačovém zpracování obrazu a metodách rozpoznávání schopné v co nejvyšší míře automatizovat analýzu řezů přízí. Všechny navržené funkce byly realizovány jako samostatné algoritmy a implementovány do programu s grafickým uživatelským rozhraním. K tomuto účelu bylo použito softwarového prostředku MATLAB. V programu byl též vytvořen grafický editor určený k základní editaci řezů. Tato práce si neklade za cíl plně zautomatizovat a nahradit manuální zpracování, ale umožnit co největší zjednodušení při dosažení minimálně srovnatelných výsledků s možností korekce získaných výsledků nebo případné upřesnění ve formě doplnění informací uživatelem.

Abstract

The aim of thesis was to design functions based on computer processing of image and methods of recognition. These functions are able to automate analysis of yarn profiles. All of the functions were pre-set as separated algorithms and implemented into program with graphical user interface. For this purpose was used software tool called MATLAB. A graphical editor was added into program for option of a simple editing of the textile profiles. This work is not an instrument to fully automate of fiber recognition but it allows obtaining of the same result as manual processing with fewer requirements. User can specify and complete results obtained from program which allows making the recognition with usage of this program only.

Obsah

Seznam obrázků a tabulek

Tabulka symbolů

∇	gradient (∇ = grad)
Ψ	směr gradientu
∇^2	operátor Laplacián
*	konvoluce
$\oplus$	dilatace
Θ	eroze
$\circ$	otevření
$\bullet$	uzavření
σ^2	rozptyl
Σ	kovariační matice nebo suma (podle kontextu)
E	střední hodnota
$P(A\|B)$	podmíněná pravděpodobnost

1. Úvod

Tato diplomová práce si klade za cíl co nejvíce zautomatizovat a zefektivnit zpracování řezů příze. Do současné doby nebyly navrženy žádné algoritmy a zpracovávání z velké části bylo realizováno manuálně. To je velmi zdlouhavé a v některých případech lze tuto činnost plně automatizovat programem. Úspěšnost navržených algoritmů a kvalita získaných výsledků je velmi závislá na vstupujících datech, tedy digitálním obrázku. Tyto algoritmy jsou implementovány do programu s grafickým uživatelským rozhraním. Program je možno rozdělit do dvou částí. První část obsahuje algoritmy analýzy obrazu, algoritmus hledající kružnice pomocí Houghovy transformace a algoritmus založený na metodě rozpoznávání pomocí řetězového kódu. Druhou částí je vlastní grafický editor, který byl vytvořen na základě požadavků na zpracování obrázků řezů. Pro práci s obrázky bylo nutno navrhnout několik algoritmů, které realizují vykreslování obrázků do sebe, sčítání, rozkládání a několik další operací s obrazovými maticemi. Jako praktický výstup z programu slouží data, která se používají pro výpočet textilních norem. Program je realizován pomocí softwarového produktu MATLAB verze 7.01. Tato práce se dá tedy chápat jako jakýsi spojovací článek mezi získáním a snímáním řezů a jejich vyhodnocováním na základě norem.

2. Řezy textilií

Řezem textilie se označuje protnutí textilie rovinou svírající příslušný úhel s danými osami. Jedna osa textilie je většinou totožná se směrem průchodu strojem.

Rozlišujeme dva základní typy řezů, a to řez příčný a řez podélný. Tato práce se zabývá pouze řezy příčnými. Příčný řez neboli průřez je vedený kolmo ke směru průchodu textilie strojem a řez podélný je k tomuto směru rovnoběžný. Pro vlákna, hedvábí a příze se nejčastěji používají řezy příčné.

Před vlastní tvorbou řezu je textilie zalita v příslušném médiu, které má za úkol její fixaci. Podle druhu média se řezy dělí na řezy měkké a tvrdé. U řezů tvrdých se jako médium volí materiál na bázi epoxidové pryskyřice. Provedením metalografického výbrusu lze u tvrdých řezů ještě zvýšit kvalitu obrazu. Včelího vosku nebo parafinu je jako média užito u řezů měkkých. Při tvorbě těchto řezů se navíc médium s textilií před řezáním mrazí. K vlastnímu řezání se používá zařízení mikrotom. Jako materiál pro nože je použita ocel u řezů měkkých, respektive skla nebo diamantu pro řezy tvrdé.

Minimální tloušťka řezů se pohybuje cca 3 - 5μm u tvrdých řezů, respektive cca 10μm u řezů měkkých, které jsou obyčejně méně pracné než řezy tvrdé.

3. Obraz

3.1. Digitalizace obrazu

Čidla pro vstup obrazové funkce jsou většinou zdrojem spojitého signálu a pro počítačové zpracování tedy musíme získat její digitální podobu. Digitalizace obrazu spočívá ve *vzorkování* v matici MxN bodů a v *kvantování* spojité jasové úrovně každého vzorku do *k* intervalů. Díky kvantování nabývá jasová funkce celočíselných hodnot. Čím jemnější je vzorkování (čím větší je M a N) ve vzorkovací matici a čím jemnější je kvantování, tím lépe se digitální reprezentace přiblíží původnímu spojitému obrazovému signálu.

3.1.1. Vzorkování

Nejdříve je třeba určit interval vzorkování (vzdálenost mezi nejbližšími vzorkovacími body v obraze). Podle Shannonovy věty musí být vzorkovací frekvence alespoň 2x větší než nejvyšší frekvence ve vzorkovaném signálu. Tedy interval vzorkování se musí volit tak, aby byl menší nebo roven polovině rozměru nejmenších detailů v obraze.

Dále je třeba určit plošné uspořádání bodů pro vzorkování (výběr vzorkovací mřížky).

Hexagonální mřížka je výhodná pro ekvidistantnost, protože je ale její implementaci do digitální techniky složitější, používá se nejčastěji mřížka čtvercová. Ta se technicky snadno realizuje a odpovídá reprezentaci obrazu pomocí pravoúhlé matice.

Jednomu vzorkovacímu bodu odpovídá v digitalizovaném obraze obrazový element (*pixel*, z angl. picture element). Po uspořádání do vzorkovací mřížky pokrývají pixely celý digitalizovaný obraz.

3.1.2. Kvantování

Amplituda ve vzorkovaném obraze musí být pro počítačové zpracování vyjádřena jako digitální údaj.

Většina systémů pro digitální zpracování obrazu používá kvantování do *k* stejných intervalů. Jestliže je pro reprezentaci informace o obrazovém elementu použito *b* bitů, je počet jasových úrovní $k = 2^b$. Obvykle se používá osm bitů na pixel. Binární obrazy reprezentují informaci o obrazovém bodě jedním bitem. Jedničky označují objekty, nuly pozadí.

Největším problémem je vznik falešných obrysů. To se stává při kvantování do příliš malého počtu jasových úrovní. Pro člověka tento jev začíná být patrný, je-li počet úrovní jasu menší než 50 (asi tolik úrovní je člověk schopen v monochromatickém obraze rozlišit). Částečným řešením je tzv. nelineární kvantování, při kterém se zvětšuje rozsah nejčastěji zastoupených intervalů jasu v obraze.

3.1.3. Kvalita obrazu

Při snímání a zpracovávání obrazu dochází k jeho degradaci a obraz tedy může obsahovat různé nežádoucí poruchy, tzv. *šum.*

Šum se většinou popisuje jeho pravděpodobnostními charakteristikami. Idealizovaný šum, ve kterém jsou rovnoměrně zastoupeny všechny frekvence, se nazývá ***bílý šum*** a představuje nejhorší možnou degradaci obrazu.

Při přenosu obrazu vzniká v přenosovém kanále šum, který je obvykle na obrazovém signálu nezávislý. Takové poruchy se označují jako ***aditivní šum*** a lze je popsat vztahem

$$f = g(x,y)+n(x,y), \tag{1}$$

kde šum *v* a vstupní obraz *g* jsou nezávislé veličiny. Ovšem v mnoha případech závisí velikost šumu na velikosti obrazového signálu. Je-li úroveň šumu dostatečně velká vzhledem k užitečnému signálu, platí

$$f = g+n \cdot g = g \cdot (1+n) == g \cdot n \ . \tag{2}$$

Model daný předchozím vztahem popisuje ***multiplikativní šum.***

Není-li počet jasových úrovní dostatečně vysoký, objeví se ***kvantizační šum.***

V binárních obrazech se vlivem šumu v oblastech objektů (reprezentovaných bílými body) objevují body černé a naopak. Takové poruchy se označují jako ***šum typu pepř a sůl*** nebo také ***impulsní šum.***

3.2. Předzpracování obrazu

Jako předzpracování se označují společné operace s obrazem na nízké úrovni abstrakce. Cílem předzpracování je potlačit šum vzniklý při přenosu a digitalizaci obrazu, odstranit zkreslení dané vlastnostmi snímacího zařízení nebo potlačit či zvýraznit další rysy obrazu důležité pro další zpracování.

3.2.1. Bodové jasové transformace

Transformace hodnot jasu se dělí do dvou skupin, kterými jsou jasové korekce a modifikace jasové stupnice. U jasových korekcí závisí jas v bodě výstupního obrazu na jasu odpovídajícího bodu ve vstupním obraze (a případně na jasu jeho malého okolí). U modifikace jasové stupnice je jen určitá hodnota jasu ve vstupním obraze transformována na jinou výstupní hodnotu, a to bez ohledu na polohu v obraze.

3.2.1.1. Jasové korekce

Snímací a digitalizační zařízení má mít v ideálním případě stejnou citlivost bez ohledu na umístění bodu v obraze, což ovšem nebývá v praxi někdy splněno. V optických soustavách je obvykle světlo procházející dále od optické osy více zeslabováno. Zdrojem poruch je i nerovnoměrné osvětlení.

Jsou-li tyto poruchy systematické, lze je zmírnit jasovými korekcemi na základě znalosti odchylky citlivosti každého bodu obrazu od ideální převodní charakteristiky. Nejčastěji

dochází ke zkreslení obrazu multiplikativním koeficientem *e(i,j)*. Pro každý bod původního obrazu *g(i,j)* získáme na výstupu zkreslený bod *f(i,j)* podle

$$f(i,j) = e(i,j)g(i,j). \tag{3}$$

Pak můžeme systematické chyby korigovat podle vztahu

$$g(i,j) = \frac{f(i,j)}{e(i,j)} = \frac{c \cdot f(i,j)}{f_c(i,j)}, \tag{4}$$

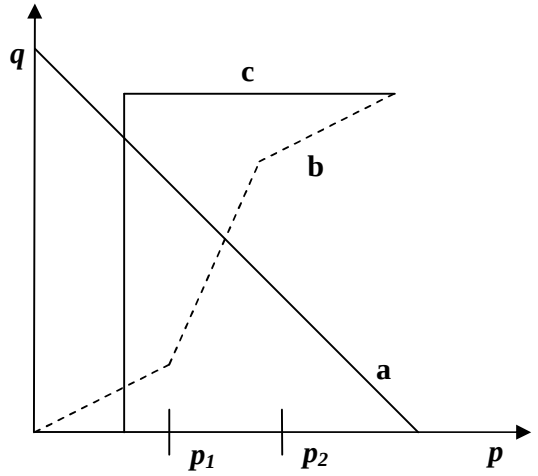
kde *c* je konstantní jas obrazu, po nasnímání a digitalizaci označený $f_c(i,j)$. Tato korekce chyb platí pro stálé snímací podmínky.

3.2.1.2. Modifikace jasové stupnice

U jasových korekcí závisí jas v bodě výstupního obrazu pouze na jasu bodu na vstupu se stejnými plošnými souřadnicemi. U modifikací jasové stupnice je jen určitá hodnota jasu ve vstupním obraze transformována na jinou výstupní hodnotu, a to bez ohledu na pozici v obraze.

Obvyklými transformacemi jasové stupnice jsou *negativ*, *změna kontrastu* a *prahování*. Tyto transformace se poměrně snadno technicky realizují.

Transformační vztah pro zvyšování kontrastu se nejčastěji hledá metodou *ekvalizace histogramu*. Ve výsledném vyrovnaném histogramu jsou jednotlivé jasové úrovně zastoupeny zhruba stejně četně. Ekvalizace zvýší kontrast pro úrovně blízko maxim histogramu a sníží v okolí minim.



Obrázek 1: obvyklé transformace jasové stupnice; (a) negativ; (b) zvětšení kontrastu mezi jasy p_1 a p_2; (c) prahování, jehož výsledkem je obrázek obsahující jen černou a bílou (každý pixel lze tedy kódovat jedním bitem)

3.2.2. Lokální předzpracování

Metody lokálního předzpracování, využívající pro výpočet jasu bodu ve výstupním obraze jen lokálního okolí odpovídajícího vstupního bodu, se dají rozdělit na dvě skupiny: vyhlazování a gradientní operace.

Vyhlazování obrazu vede k potlačení vyšších frekvencí obrazové funkce. Výsledkem je potlačení náhodného šumu, ale dochází také k vyhlazování ostatních náhlých změn jasové funkce, jako jsou ostré čáry a hrany nesoucí významnou informaci.

Gradientní operace a s nimi související ostření obrazu vedou naopak ke zdůraznění vyšších frekvencí. Současně jsou zvýrazněny ty obrazové elementy, ve kterých se obrazová funkce náhle mění. Výsledkem je zvýraznění hran v obraze, ale dochází také ke zvýraznění šumových bodů.

3.2.2.1. Filtrace

Jako filtrace se označuje skupina transformací, které převádějí hodnoty jasu vstupního obrazu na jiné jasové hodnoty výstupní s cílem zvýraznit nebo potlačit některé jeho vlastnosti. Častým cílem filtrace je vyhlazování šumu v obraze.

Obyčejné průměrování filtruje obraz tak, že bodu přiřadí aritmetický průměr jasu bodů obdelníkového okolí. Potlačí se tak skvrny šumu menší, než je nejmenší významný detail v obraze. Nevýhodou obyčejného průměrování je rozmazávání hran v obraze, proto se tato technika většinou používá jako pomocná pro výpočet střední hodnoty jasu nebo jeho rozptylu v daném bodě. Tohoto výsledku pak využívají propracovanější filtrační metody, jako je *OS filtrace* (order statistic) nebo metoda *rotující masky*, která podle homogenity jasu nachází k filtrovanému bodu tu část okolí, ke které bod pravděpodobně patří.

3.2.2.2. Detektory hran

Místa v obraze kde se náhle mění hodnota jasu se označují jako hrany. Hrana v obraze je dána vlastnostmi obrazového elementu a jeho okolí. Je to vektorová veličina a je určena tím, jak náhle se mění hodnota obrazové funkce *f(x,y)*. K matematickému popisu se používá parciálních derivací, které jsou určeny pro vyjádření změn funkce dvou proměnných. Změnu funkce udává její gradient, vektorová veličina ∇ , určující směr největšího růstu funkce (směr gradientu) a strmost tohoto růstu (velikost, modul gradientu). Hranami jsou potom pixely s velkým modulem gradientu.

Pro spojitou obrazovou funkci *f(x,y)* jsou velikost gradientu $|\nabla f(x,y)|$ a směr gradientu ψ dány vztahy

$$|\nabla f(x, y)| = \sqrt{\left(\frac{df}{dx}\right)^2 + \left(\frac{df}{dy}\right)^2}\ , \tag{5}$$

$$y = \arg\left(\frac{df}{dx}, \frac{df}{dy}\right), \tag{6}$$

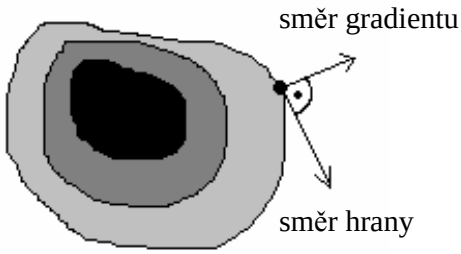
kde *arg(x, y)* je úhel (v radiánech) mezi souřadnou osou *x* a radiusvektorem k bodu *(x, y)*.

Pokud nás zajímají hrany bez ohledu na jejich směr lze použít všesměrový lineární Laplaceův operátor Laplacián ∇^2. Tento operátor vychází z druhých parciálních derivací jak

je patrno ze zápisu. Pro monotónně rostoucí jasovou funkci *f* v příslušném okolí je Laplacián nulový tam, kde je velikost gradientu | ∇ | maximální.

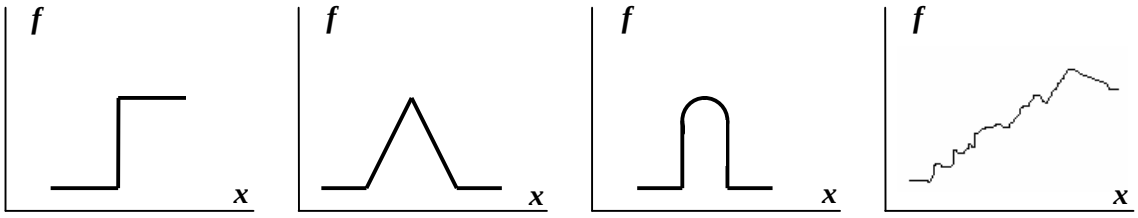
$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial f(x,y)}{\partial y^2} \tag{7}$$

Hrany nalezené v obraze lokálními operátory se někdy používají pro hledání hranic objektů. Pokud je objekt tvořen oblastí homogenního jasu, jsou body hranice právě pixely s vysokou hodnotou gradientu. Pixely které tvoří hrany (hranové pixely) jsou spojovány do hranic a proto se směr hrany definuje jako kolmý na směr gradientu jak je patrné z obrázku 2.



Obrázek 2: hrana v obraze

Hrany se dají třídit podle jasového profilu, který je jednorozměrný a je určován ve směru gradientu. Jasové profily nejběžnějších hran jsou zobrazeny na obrázku 3. První tři profily jsou idealizované, v reálném obrázku se vyskytují pouze zašuměné hrany.



(a) **(b)** **(c)** **(d)**

Obrázek 3: jasové profily hran: (a) skoková hrana, (b) střechová hrana, (c) liniová hrana, (d) zašuměná hrana

Gradientních operátorů lze použít i pro ostření obrazu. Jeho cílem je získat strmější hrany v obraze.

Gradientní operátory udávající strmost obrazové funkce můžeme rozdělit do tří kategorií. První skupina aproximuje derivace obrazové funkce pomocí diferencí realizovaných diskrétní konvolucí. Operátory invariantní vůči rotaci se realizují jedinou konvoluční maskou (např. Laplaceův oparátor). Operátory, které aproximují první derivaci, používají několik masek. Směr gradientu se odhaduje hledáním té masky, která odpovídá největší velikosti gradientu. Například Robertsův operátor, Sobelův operátor, Robinsonův, Kirschův a Prewittové. Druhá skupina operátorů je založena na hledání hran v místech, kde je druhá derivace obrazové funkce nulová. Mezi tyto operátory se řadí například operátor Marra a Hildrethové nebo Cannyho hranový detektor. Třetí skupina operátorů se snaží lokálně aproximovat obrazovou funkci poměrně jednoduchým parametrickým modelem.

Některé operátory se dají vyjádřit pomocí konvoluční masky, která je potom použita pro konvoluci podle vztahu

$$g(x,y)=\sum_{(i,j)}\sum_{\in O} h(x-i,y-j)f(i,j), \tag{8}$$

kde *f(x,y)* je vstupní obraz, *g(x,y)* je výstupní obraz a *h(x,y)* je konvoluční maska.

Velmi jednoduchý Robertsův operátor používá okolí reprezantativního bodu o velikosti 2x2 a jeho konvoluční masky jsou

$$h_1=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix},\ h_2=\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix} \tag{9}$$

Hlavní nevýhodou tohoto operátoru je značná citlivost na šum vyplývající z použití malého okolí.

Konvoluční masky Laplaceova operátoru pracují již v okolí o velikosti 3x3 a jejich základní tvar pro 4-okolí a 8-okolí je tento

$$h_4=\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix},\ h_8=\begin{bmatrix}1 & 1 & 1\\ 1 & -8 & 1\\ 1 & 1 & 1\end{bmatrix}. \tag{10}$$

Existuje i verze Laplaciánu s větší váhou pixelů blíže reprezentativnímu bodu masky. V tomto případě už ale neplatí invariantnost vůči otočení.

Mezi operátory aproximující první derivaci patří operátor Prewittové. Gradient je odhadován v okolí 3x3 pro osm směrů. Vybrána je jedna maska z osmi, a to ta, které odpovídá největší modul gradientu.

$$h_1=\begin{bmatrix}1 & 1 & 1\\ 0 & 0 & 0\\ -1 & -1 & -1\end{bmatrix},\ h_2=\begin{bmatrix}0 & 1 & 1\\ -1 & 0 & 1\\ -1 & -1 & 0\end{bmatrix},\ h_3=\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix},\ h_4=\begin{bmatrix}-1 & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix},\ \ldots h_8$$

Je možné vytvářet i masky s podrobnějším směrovým rozlišením, které jsou tím pádem větší.

Operátorem, který se často používá pro detekci vodorovných a svislých hran je Sobelův operátor, jehož masky jsou

$$h_1=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix},\ h_2=\begin{bmatrix}0 & 1 & 2\\ -1 & 0 & 1\\ -2 & -1 & 0\end{bmatrix},\ h_3=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix},\ h_4=\begin{bmatrix}-2 & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & 2\end{bmatrix},\ \ldots h_8$$

Nevýhodou operátorů aproximujících derivace diferencemi v malém okolí je značná závislost jejich chování na zpracovávaném obrázku. Závislost na zvoleném měřítku a citlivost na šum jsou také značné.

Na základě tohoto byla formulována Marrova teorie, která měla za cíl vytvořit matematický model detekce skokových hran na základě neurofyziologického měření na sítnici

oka. V místě hrany nabývá první derivace obrazové funkce svého maxima. Tato teorie vychází z derivace druhé. V místě hrany prochází druhá derivace obrazové funkce nulou. Použití druhé derivace místo první je vhodnější, protože maximum u derivace první je velmi často ploché.

K robustnímu odhadu druhé derivace je použito konvoluce s lineárním vyhlazujícím filtrem, jehož koeficienty v konvoluční masce odpovídají 2D gaussovskému rozložení a je nazýván jako Gaussián

$$G(x, y, s) = e^{-\frac{x^2+y^2}{2s^2}}. \tag{11}$$

Parametry x a y jsou souřadnice v obraze a σ je středněkvadratická odchylka, která říká na jak velkém okolí filtr pracuje. Pro odhad druhé derivace je použit Laplacián ∇^2. Potom tedy dostaneme

$$\nabla^2 \left(G(x, y, s) * f(x, y)\right) \tag{12}$$

Tento postup je označován jako *LoG operátor* (angl. Laplacian of Gaussian). Protože použité operace jsou lineární, můžeme zaměnit pořadí derivace a konvoluce. Tím docílíme nezávislosti obrazu, respektive obrazové funkce, na derivaci Gaussiánu ∇^2 G a můžeme napsat

$$\nabla^2 \left(G(x, y, s) * f(x, y)\right) = \left(\nabla^2 G(x, y, s)\right) * f(x, y)$$

Hodnoty derivace Gaussiánu poté můžeme spočítat analyticky. Výsledkem je konvoluční maska, které se říká mexický klobouk. Pro velikost masky 5x5 vypadá aproximace *LoG* operátoru následovně

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}. \tag{13}$$

Nevýhoda operátorů používajících konvoluční masky je závislost jejich chování na daném obrázku. Velikost zvolené masky musí odpovídat velikosti detailů v obrázku. Tímto neduhem netrpí Cannyho hranový detektor, který byl navržen na principu různých rozlišení a hledání nejlepšího z nich. Jeho základní myšlenkou je hledání hrany filtrem. Návrh tohoto filtru byl řešen jako úloha variačního počtu. Byla také formulována tři kritéria, jejichž splnění zaručuje optimalitu detektoru. Kritéria požadují, aby všechny významné hrany byly detekovány, aby na žádnou hranu nebyla vícenásobná odezva a aby rozdíl mezi skutečnou a nalezenou hranou byl minimální. Odvození Cannyho detektoru je zdlouhavé a je mimo rozsah této práce.

3.3. Segmentace

Úkolem segmentace je rozdělit obraz do částí, které mají souvislost s předměty nebo oblastmi reálného světa zachyceného na obraze.

Segmentace může být buď *kompletní*, kdy výsledkem je soubor vzájemně se nepřekrývajících oblastí, které jednoznačně odpovídají objektům vstupního obrazu, nebo vytvořené segmenty nemusí přímo souhlasit s objekty obrazu, případně se překrývají, pak se jedná o *částečnou segmentaci*.

3.3.1. Segmentace prahováním

Prahování je nejjednodušší segmentační postup. Vychází ze skutečnosti, že mnoho objektů nebo oblastí obrazu má konstantní odrazivost či pohltivost povrchu. Pak se může využít určená jasová konstanta - *práh* - k oddělení objektů od pozadí. Vzhledem k nenáročnosti výpočtu je prahování nejrychlejší segmentační metoda, lze ji provádět v reálném čase.

Prahování je tedy transformace vstupního obrazu *f* na výstupní binární obraz *g* daná vztahem

$$g(i,j)=1 \text{ pro } f(i,j)\geq T,\ g(i,j)=0 \text{ pro } f(i,j)<T, \tag{14}$$

kde *T* je předem určená konstanta (práh), *g(i,j)=1* pro obrazové elementy náležející objektům, *g(i,j)=0* pro elementy pozadí (nebo naopak).

3.3.1.1. Metody určování prahu

Klíčovým úkolem při segmentaci prahováním je určení vhodného prahu. Hodnoty prahu lze určovat interaktivně nebo metodami automatického určování prahu.

Procentní prahování využívá apriorní znalosti poměru ploch objektů a pozadí. Víme-li, že objekty zaujímají *1/p* plochy obrazu, na základě histogramu snadno určíme takovou hodnotu prahu *T*, aby právě *1/p* plochy měla úroveň jasu menší než *T*.

Složitější metody se opírají o analýzu tvaru histogramu.

3.3.2. Segmentace na základě detekce hran

Segmentace na základě detekce hran vychází ze skutečnosti, že hranice oblastí v obraze se skládají z hran, které jsou nalezeny aplikací některého z hranových operátorů. Takto nalezené hrany označují místa v obraze, kde dochází k jisté nespojitosti - obvykle v hodnotě jasu, nebo v barvě či textuře.

Ovšem obraz, který vznikne aplikací hranového operátoru, je jako výstup segmentace ve své prvotní podobě téměř nepoužitelný. Proto po detekci hran následuje další zpracování, které tyto hrany spojuje do řetězců lépe odpovídajících původní hranici.

Nejčastějším problémem hranových segmentačních metod je výskyt hran v místech, kde není skutečná hranice, a naopak absence hran tam, kde hranice ve skutečnosti probíhá.

Nejčastěji používanými metodami jsou *prahování obrazu hran*, kdy se nevýznamné hrany vzniklé vlivem šumu odstraňují prahováním vhodným prahem, *sledování hranice*, jejímž cílem je určit vnitřní hranice všech oblastí obrazu, nebo *heuristické sledování hranice*, kdy se využívá znalosti určitých vlastností hranice. V případě, že obraz obsahuje předměty, jejichž tvar a velikost jsou známy, chápeme segmentaci jako úlohu nalezení daného předmětu

v obraze a lze použít *Houghovu transformaci* (viz. kapitola 4), která vychází ze vzorového tvaru hledané hranice.

3.3.3. Segmentace narůstáním oblastí

Metoda narůstání oblastí je výhodná v obrazech se šumem, v nichž se hranice určují zvlášť obtížně.

Základní myšlenkou této metody je rozdělit obraz do maximálních souvislých oblastí tak, aby byly z hlediska zvoleného způsobu popisu homogenní. Kritériem homogenity mohou být jasové vlastnosti nebo komplexnější způsoby popisu, jako je například textura, nebo dokonce sémantická reprezentace obrazu.

Nejpřirozenějším způsobem je zahájit narůstání v původních obrazových datech, kde každý pixel představuje samostatnou oblast. Tyto existující oblasti jsou postupně spojovány tak dlouho, dokud by jejich dalším spojením nebyla porušena homogenita.

Štěpení oblastí je principiálně opačný přístup k segmentaci než jejich spojování. Tyto algoritmy vycházejí z počátečního rozdělení obrazu do jediné oblasti. Tento postup je teoreticky duální k postupům spojování oblastí, ale přesto nedává ani při použití stejných kritérií homogenity tytéž výsledky při aplikaci na reálná data.

3.3.4. Segmentace srovnáváním se vzorem

Dalším přístupem k segmentaci je vyhledávání známých objektů v obraze pomocí srovnávání se vzorem *(matching)*. Obecně jde o nalezení míst v obraze, kde se vyskytuje daný vzor, který má charakter obrazu.

Kromě hledání objektů a oblastí lze srovnávací metody použít i pro stereoskopické určování vlastností objektů scény, máme-li dva obrazy stejné scény snímané z různých míst. Srovnávat lze na úrovni velmi malých vzorů až po vzory pokrývající celé hledané objekty.

Touto metodou jsou v obraze nalezena všechna místa, kde se nacházejí velmi přesné kopie vzoru. Abychom mohli využít srovnávacích technik i pro hledání vzorů různě natočených nebo zvětšených, bylo by nutné vytvořit pro každou možnou velikost a orientaci samostatný vzor. Jiná možnost je použít sice jediný vzor, ale srovnávat obraz se všemi jeho dovolenými transformacemi.

3.4. Matematická morfologie

Matematická morfologie svým matematickým aparátem vycházejícím z algebry nelineárních operací do značné míry při zpracování signálů či obrazů předstihuje tradiční lineární přístup, který využívá lineární kombinace (konvoluci) bodových zdrojů představovaných Diracovými impulsy. Jde např. o předzpracování obrazu, o segmentaci s důrazem na tvar hledaných objektů, o kvantitativní popis nalezených objektů. Operátory matematické morfologie se obvykle používají tam, kde je požadavek na krátký čas zpracování. Aplikačními oblastmi jsou biologie, geologie, kriminalistika, obrazová inspekce v průmyslu, rozpoznávání znaků a dokumentů, aj. Morfologické metody lze použít jak pro 2D obrazy, tak je možné je využít i pro zpracování 1D signálů.

3.4.1. Základní morfologické pojmy

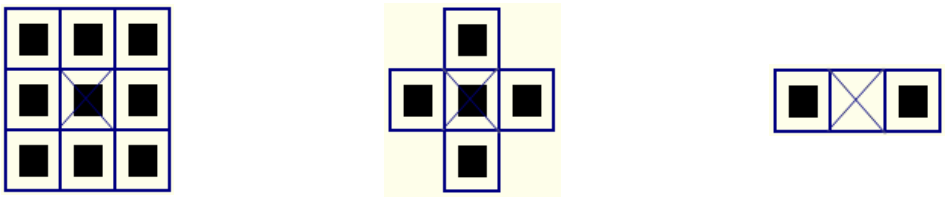
Matematická morfologie využívá vlastností bodových množin, výsledky z integrální geometrie a topologie. Základním předpokladem je představa, že reálné obrázky lze modelovat pomocí bodových množin libovolné dimenze (např. *N*-rozměrný euklidovský prostor).

Morfologická transformace je dána relací mezi obrazem a jinou, typicky menší množinou, které se říká strukturní element. Strukturní element je vztažen k „lokálnímu“ počátku, který se nazývá reprezentativní bod.

Aplikaci morfologické transformace si lze představit jako systematické posouvání strukturního elementu po obraze. Výsledek relace mezi obrazem a strukturním elementem se zapíše do výstupního obrazu v reprezentativním pixelu.

Ke každé morfologické operaci existuje duální transformace, což vyplývá z množinových doplňků.

Základními transformacemi matematické morfologie jsou *dilatace*, *eroze*, *otevření* a *uzavření*.



(a) (b) (c)

Obrázek 4: typické strukturní elementy

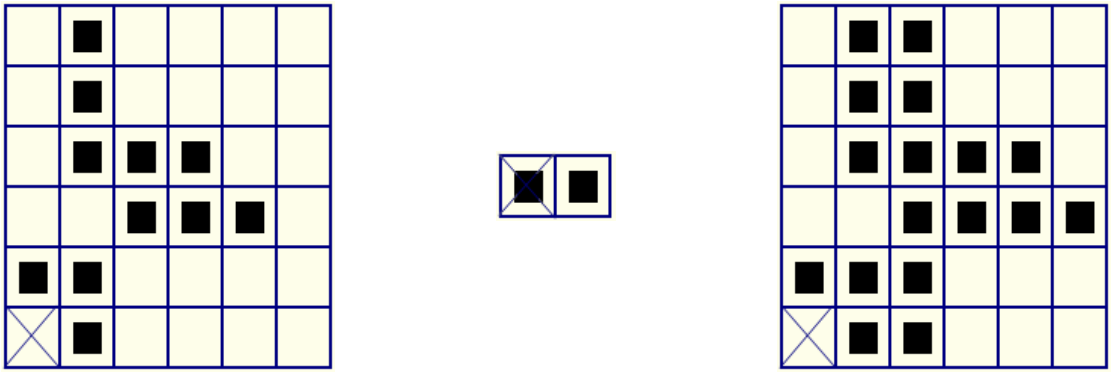
3.4.2. Binární matematická morfologie

3.4.2.1. Dilatace

Dilatace $\oplus$ skládá body dvou množin pomocí vektorového součtu. Dilatace $M \oplus E$ je bodovou množinou všech možných součtů pro dvojice pixelů, vždy pro jeden z množiny A a jeden z množiny B

$$M \oplus E = \left\{ p \in e^2 : p = m + e, m \in M, e \in E \right\}. \tag{15}$$

Dilatace je operací komutativní, asociativní a je invariantní vůči posunu. Používá se samostatně k zaplnění malých děr, úzkých zálivů a jako stavební kámen složitějších operací. dilatace zvětšuje objekty. Často je kombinována s morfologickou operací eroze pro zachování původního rozměru objektů.



(a) (b) (c)

Obrázek 5: dilatace: (a) vstupní obrázek, (b) strukturní element, (c) výsledný obrázek

3.4.2.2. Eroze

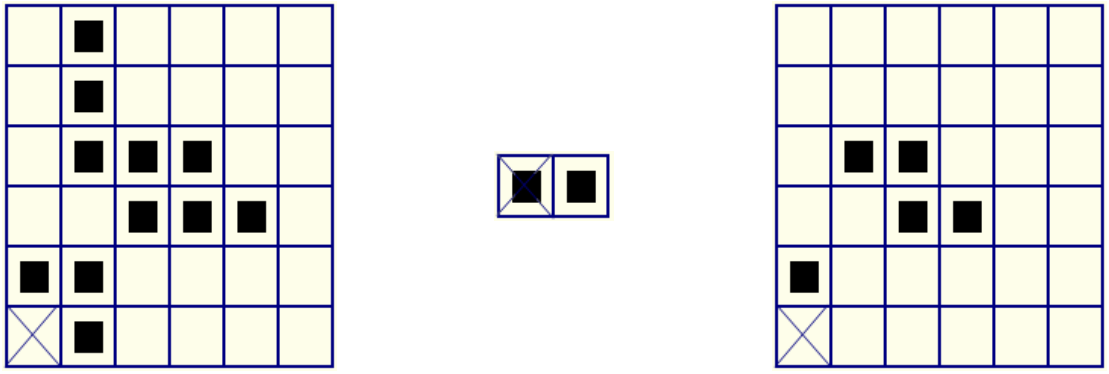
Eroze Θ skládá dvě množiny podle předpisu

$$M \ominus E = \{ p \in e^2 : p + m \in M \text{ pro každé } e \in E \}. \quad (16)$$

Tento vztah říká, že pro každý bod obrazu p se ověřuje, zda pro všechna možná $p + m$ leží výsledek v M. Pokud ano, zapíše se v reprezentativním bodě do výsledného obrázku 1 a v opačném případě 0.

Eroze není narozdíl od dilatace komutativní a používá se pro zjednodušení struktury objektů. Eroze ani dilatace nejsou invertovatelné. Mezi těmito operacemi platí vztah duality a tyto transformace označujeme jako duální, což charakterizuje vztah

$$(M \ominus E)^C = M^C \oplus E. \quad (17)$$



(a) (b) (c)

Obrázek 6: eroze: (a) vstupní obrázek, (b) strukturní element, (c) výsledný obrázek

3.4.2.3. Otevření a uzavření

Morfologické operace otevření a uzavření jsou tvořeny kombinacemi duálních operací, kterými jsou dilatace a eroze. Výsledkem otevření i uzavření je obraz obsahující méně detailů, dá se tedy považovat za zjednodušený.

Eroze následovaná dilatací vytváří transformaci otevření. Otevření množiny M strukturním elementem E se označuje $M \circ E$ a je definováno jako

$$M \circ E = (M \ominus E) \oplus E. \tag{18}$$

Transformace uzavření je tvořena sledem transformací dilatace a eroze. Oproti otevření je tedy pouze přehozeno pořadí dilatace a eroze. Uzavření množiny M strukturním elementem E se označuje $M \bullet E$ a jeho definice je

$$M \bullet E = (M \oplus E) \ominus E. \tag{19}$$

Pokud se obraz po otevření strukturním elementem nezmění, říkáme, že je vhledem k němu *otevřený*. Analogicky se definuje *uzavřenost* u transformace uzavření a můžeme říct, že obraz je uzavřený vhledem ke strukturnímu elementu pokud nedojde k jeho změně po aplikaci transformace uzavření daným strukturním elementem.

Otevření a uzavření se používá pro odstranění detailů v obraze, které jsou menší než strukturní element. K tomu se používá strukturního elementu, který nezávisí na směru a označuje se tedy izotropický. Celkový tvar objektu potom není porušen. Otevření oddělí objekty spojené úzkou šíjí a tak zjednoduší strukturu objektů. Transformace uzavření spojí objekty, které jsou blízko u sebe, zaplní malé díry a vyhladí obrys tím, že zaplní úzké zálivy. Blízkost objektů, velikost děr a úzkost zálivů se chápe relativně vzhledem k velikosti strukturního elementu.

Otevření a uzavření jsou duálními transformacemi stejně jako dilatace a eroze

$$(M \bullet E)^C = M^C \circ E. \tag{20}$$

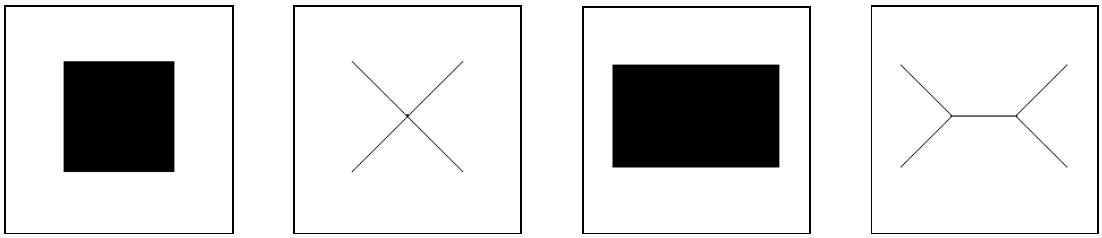
Další důležitou vlastností otevření a uzavření je idempotentnost. To znamená, že opakované použití obou operací již nemění výsledek. Má tedy smysl mluvit o otevřené či uzavřené množině vzhledem ke strukturnímu elementu. Platí tedy následující vztahy

$$M \circ E = (M \circ E) \circ E \tag{21}$$

$$M \bullet E = (M \bullet E) \bullet E \tag{22}$$

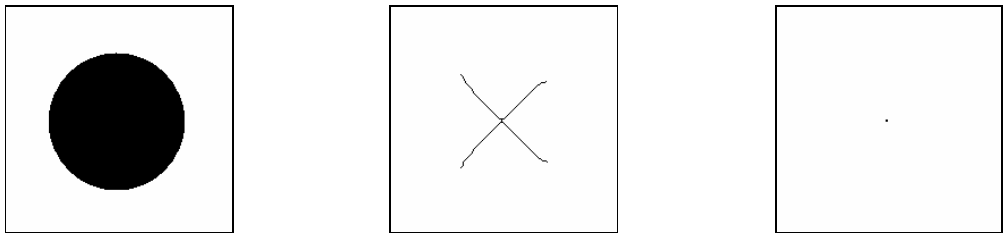
3.4.3. Skelet

Původní název pro skelet byla střední osa (angl. medial axis transform). Jedná se o morfologickou operaci, která odstraňuje pixely na hranicích objektů za předpokladu zachování jejich souvislosti. Skelet lze laicky vysvětlit na představě o požáru. Pokud na hranici nějakého rovinného plošného útvaru v jednom okamžiku zapálíme oheň, pak je skelet tvořen všemi body, ve kterých se setkají dva a více ohňů. Toto platí za předpokladu šíření ohně všemi směry stejnou a konstantní rychlostí. Například skeletem kruhu je jeho střed, skeletem čtverce jsou jeho úhlopříčky.



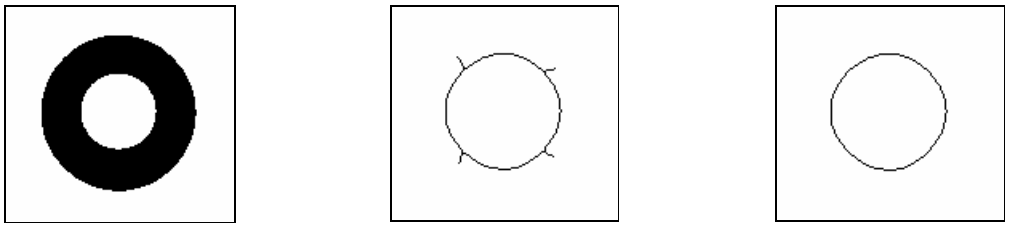
(a) **(b)** **(c)** **(d)**

Obrázek 7: (a) čtverec; (b) skelet čtverce; (c) obdélník; (d) skelet obdélníku (obrázky jsou invertované kvůli tisku)



(a) **(b)** **(c)**

Obrázek 8: (a) kruh; (b) skelet získaný programem (neodpovídá teorii); (c) skelet kruhu (jediný bod - střed) odpovídající teorii (obrázky jsou invertované kvůli tisku)



(a) **(b)** **(c)**

Obrázek 9: (a) mezikruží; (b) skelet mezikruží získaný programem (neodpovídá teorii); (c) skelet mezikruží odpovídající teorii (obrázky jsou invertované kvůli tisku)

3.5. Popis nalezených objektů

Cílem popisu nalezených objektů je buď číselný vektor příznaků nebo nečíselný syntaktický popis charakterizující tvarové nebo jiné vlastnosti popisované oblasti.

K popisu oblastí je důležité definovat jejich tvar. Jedním z hledisek, podle kterých lze rozdělit postupy tvarového popisu, může být charakter vstupní reprezentace - jestli popis vychází ze znalosti oblasti nebo jen z její hranice. Jiným hlediskem je množství zachované informace, jestli lze objekt z popisu znovu rekonstruovat. další možností je rozdělit metody na matematické a heuristické. Důležité je také to, jestli navržený způsob reprezentace vede k příznakovému nebo syntaktickému popisu.

Nutným předpokladem k popisu oblastí je jejich identifikace, která umožňuje jednoznačně se odvolávat na každou oblast obrazu. Jedním z možných způsobů je přidělit každé oblasti (resp. každé hranici oblasti) jedinečné přirozené číslo. Taková identifikace se nazývá *barvení*.

3.6. Porozumění obsahu obrazu

K celkovému porozumění je třeba pracovat s vnitřním modelem, reprezentujícím "představy" systému počítačového vidění o zpracovávané části reálného světa. Na základě apriorních znalostí zpracovávané scény dochází k vytváření vnitřních modelů, k jejich ověřování a průběžné aktualizaci. K tomu je vždy nutné vykonat vhodnou posloupnost kroků zpracování.

V reprezentaci prostředí jsou nové údaje porovnávány se stávajícím modelem a mohou být použity k modifikaci modelu. Interpretace údajů přitom není jednoznačně závislá jen na údajích samotných, v případě různých výchozích modelů nebo různé předchozí zkušenosti mohou být data interpretována pokaždé odlišným způsobem.

3.6.1. Sémantické metody segmentace a interpretace obrazu

Sémantická informace je znalostí "vyšší" úrovně. Např. při spojování oblastí při segmentaci je logické spojovat ty sousední oblasti, které mají stejnou interpretaci. Sémantická informace je v procesu spojování oblastí používána až v pozdějších krocích - když na základě obecných heuristik není další spojování možné, jsou určovány sémantické vlastnosti dosud vytvořených oblastí a je buď povoleno nebo zakázáno další spojování sousedních oblastí.

Pro praktické využití je třeba navrhnout odpovídající model vzájemných vztahů jednotlivých interpretací oblastí, metody určování věrohodnosti interpretace atd.

4. Houghova transformace

Houghova transformace je metoda sloužící k hledání definovaných objektů v obrázku. Protože vyžaduje, aby hledaný objekt byl parametricky popsán, klasická Houghova transformace slouží především k detekci pravidelných křivek jako jsou úsečky, kružnice, elipsy atd. Zobecněná Houghova transformace může být použita tam, kde není možný jednoduchý analytický popis objektu.

Houghova transformace má mnoho praktických aplikací jako například v lékařství. V těchto aplikacích je transformace realizována především k hledání hranic objektů v podobě již zmíněných křivek a úseček.

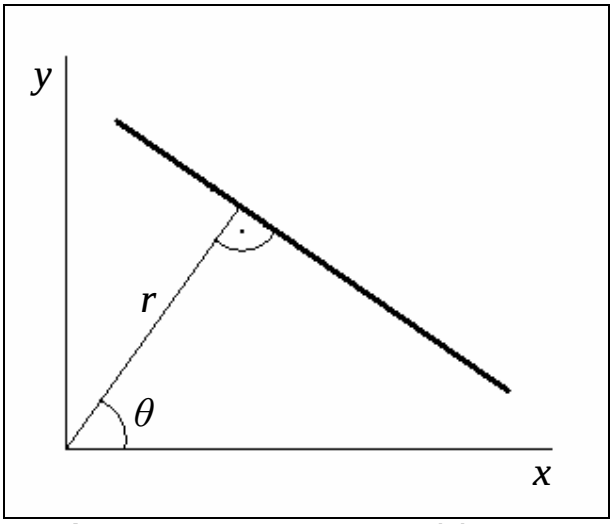
Hlavní výhoda Houghovy transformace je určitá tolerance rozdílů rozpoznávaných objektů oproti jejich parametrickému popisu a relativní odolnost proti zašumění vstupního obrázku.

Asi nejjednodušší případ je hledání úseček, respektive přímek. Klasická rovnice přímky

$$y = k \cdot x + q \tag{23}$$

nelze použít kvůli neomezenosti parametru *k* definujícího sklon, tedy směrnici přímky. Jako parametrický popis se proto použije polární tvar rovnici přímky

$$x \cdot \cos q + y \cdot \sin q = r\,. \tag{24}$$



Obrázek 10: parametricky popsaná úsečka

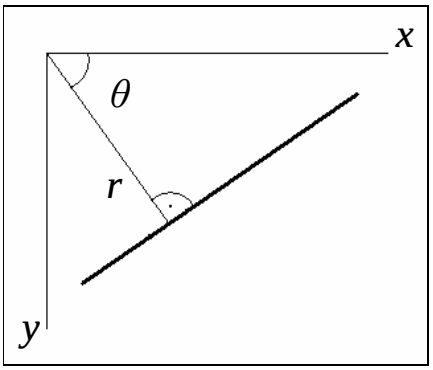
Jak je zřejmé z obrázku 10, parametr *r* představuje vzdálenost přímky od počátku (v digitálním obrázku jsou rozměrem pixely) a úhel θ svírají osa *x* a kolmice vztyčená k přímce procházející počátkem. Intervaly těchto dvou parametrů jsou omezené pro celou množinu všech přímek. Volba těchto dvou intervalů je možná dvěma způsoby, které jsou patrné z obrázku 10. První možností je interval <0;360) pro úhel θ, parametr *r* poté nabývá pouze kladných hodnot, teoreticky až do velikosti úhlopříčky obrázku, kterou lze jednoduše spočíst Pythagorovou větou. Druhou variantou je možnost uvažovat i zápornou hodnotu parametru *r*, konkrétně tedy dvojnásobný interval oproti první možnosti čímž je interval úhlu θ zredukován analogicky na polovinu, tedy <0;180). Pro náš případ je vhodnější první verze kvůli použití MATLABu, který umožňuje pouze kladnou indexaci od jedničky. Intervaly jsme v počítačové

realizaci zredukovaly na celá čísla a ještě o jedničku posunuly, aby bylo možné dosáhnout nulových hodnot parametrů.

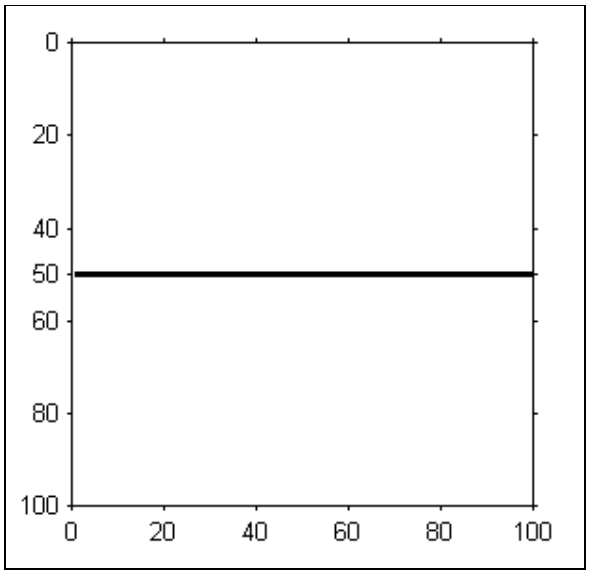
Před aplikací transformace se nejprve nadefinuje prostor obsahující všechny možné hodnoty parametrů hledaného objektu, který se nazývá akumulátor. Tento prostor je reprezentován *n*-dimenzionální maticí, kde *n* je určeno počtem parametrů objektu. V případě úsečky jsou to samozřejmě parametry r a θ, které budou představovat souřadnice akumulátoru. Na počátku musí být všechny hodnoty akumulátoru shodné, obvykle se volí nula.

Transformace funguje tak, že se prochází systematicky obrázek pixel po pixelu. Pokud se narazí na pixel objektu (v našem nejjednodušším případě je vstupem binární obrázek tudíž pozadí…0, objekt…1), dosazuje se do rovnice (24). Proměnné x a y jsou souřadnice nalezeného pixelu. Protože v rovnici jsou dvě neznámé r a θ, za jednu z nich se postupně dosazují všechny hodnoty jejího intervalu vycházející z akumulátoru a druhá hodnota se vypočítá. Pro dvojici takto získaných hodnot $[r , \theta]$ se přičte konstanta, nejčastěji jednička, na příslušné místo do akumulátoru. Hodnoty se tedy akumulují, proto je matice parametrů nazvaná akumulátor. Po projití celého obrázku se z akumulátoru vyberou lokální maxima, která definují jednotlivé objekty. Z toho vyplývá, že v případě obrázků s různými velikostmi objektů, tedy úsečky různých délek nebo kružnice s různými poloměry, jsou zjednodušeně řečeno detekovány vždy ty objekty, které jsou tvořeny nejvíce pixely, protože jim odpovídá větší maximum v akumulátoru.

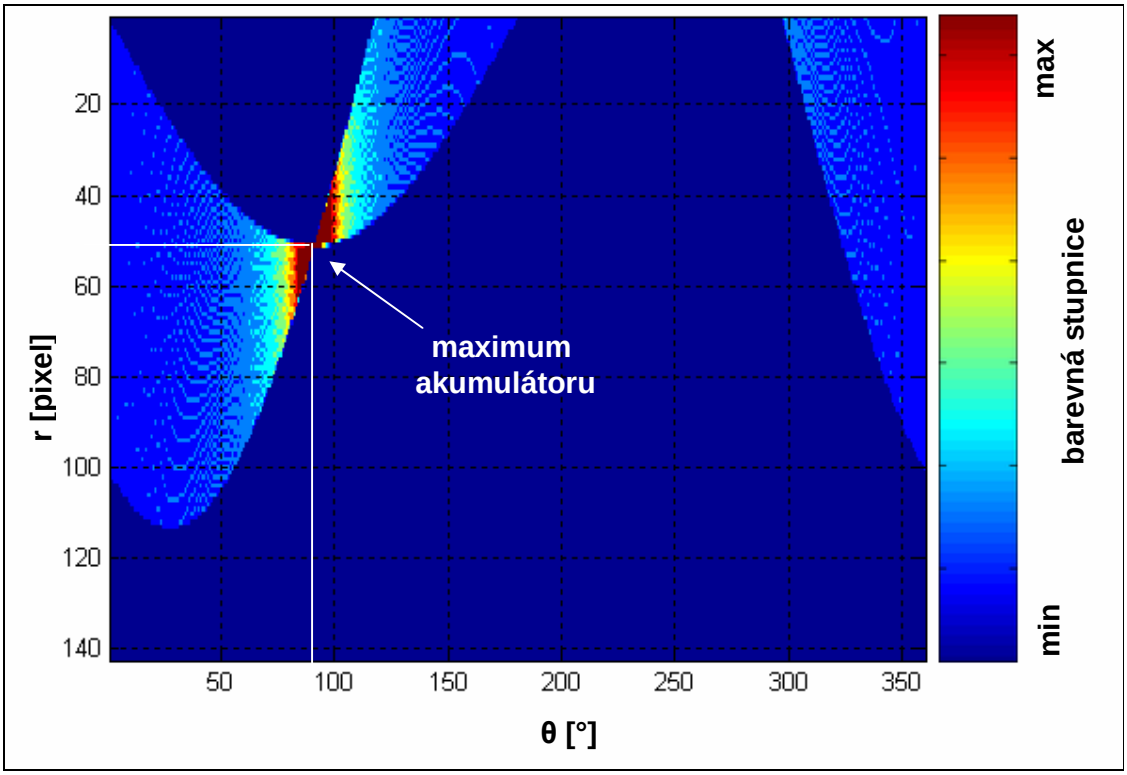
V důsledku způsobu indexace obrazové matice MATLABem je orientace souřadného systému jiná oproti konvenci jak je vidět na obrázku 11.



Obrázek 11: orientace souřadného systému v MATLABu



Obrázek 12: binární obrázek vstupující do houghovy transformace (obrázek je invertován kvůli tisku)



Obrázek 13: akumulátor odpovídající obrázku 12 jako vstupu Houghovy transformace

K hledání kružnic se použije opět analytický popis, tedy rovnice kružnice ve tvaru

$$(x-a)^2 + (y-b)^2 = r^2 \tag{25}$$

Počet neznámých ve srovnání s přímkou vzrost o jedna na hodnotu tři. Tudíž i výpočetní náročnost se zvýšila obecně o jeden řád. Akumulátor je tedy třírozměrný. Parametry jsou střed

kružnice o souřadnicích *[a, b]* a poloměr *r*. Intervaly hodnot parametrů se dají zvolit podle požadavků a nastavení není jednoznačné. Lze umožnit detekci i kružnic, které nebudou v obrázku zakresleny celé, což vyplývá z principu transformace.

Principielně je výpočet opět stejný, prochází se tedy systematicky obrázek, příslušné hodnoty jsou dosazovány do rovnice (25) a dochází k akumulaci hodnot v akumulátoru.

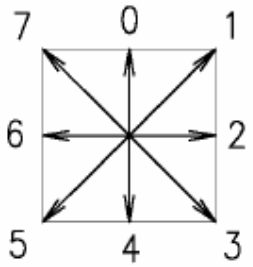
5. Rozpoznávání objektů na základě příznaků

5.1. Řetězový kód a jeho použití

Řetězový kód je stručnější způsob reprezentace množiny pixelů. Může být definován s ohledem na pixely nebo na hranice mezi nimi. Řetězový kód je používán k získání jednorozměrného signálu z obrazu, protože z jednorozměrného signálu, který odpovídá funkci jedné proměnné, je možno získat příznaky popisující objekty s daleko menšími výpočetními nároky.

Tento kód lze použít obecně pro libovolný rastr (mřížku). Nejčastěji se používají varianty pro osmiokolí nebo čtyřokolí čtvercového rastru. Čtvercový rastr ale i přes hojné použití představuje nepříliš ideální volbu. V případě osmiokolí je problémem nestejná vzdálenost (neekvidistantnost) pixelů vzájemně sousedících v horizontálním, respektive vertikálním směru oproti pixelům ve směru diagonálním. U čtyřokolí se s tímto faktem sice nesetkáváme, nastává zde ale jiný problém, a to omezenost pouze na čtyři směry, což je většinou nedostačující. Nevýhodu čtvercové mřížky odstraňuje mřížka hexagonální, která má podobu včelí plástve. Rozšíření tohoto rastru brání dva důvody. Prvním z nich je ten fakt, že většina digitalizačních zařízení preferuje rastr čtvercový. Druhým důvodem je nevhodnost pro některé operace jako například Fourierovu frekvenční filtraci. Problémy čtvercového rastru řeší použití diskrétní topologie opírající se o celulární komplexy.

V našem případě je řetězový kód tvořen z binárního obrázku, který byl získán prahováním nebo použitím hranového detektoru. Objektům odpovídají uzavřené hranice, jedná se tedy o hranový obrázek. Ten vstupuje do algoritmu, který pracuje na principu postupného procházení jednotlivých hranic pixel po pixelu v osmiokolí. Podle vzájemné polohy procházených pixelů získáváme řetězový kód v podobě řady číslic z intervalu <0; 7>. Na obrázku 14 je vidět přiřazení číslic k jednotlivým směrům, ve kterých je možné hranici procházet.



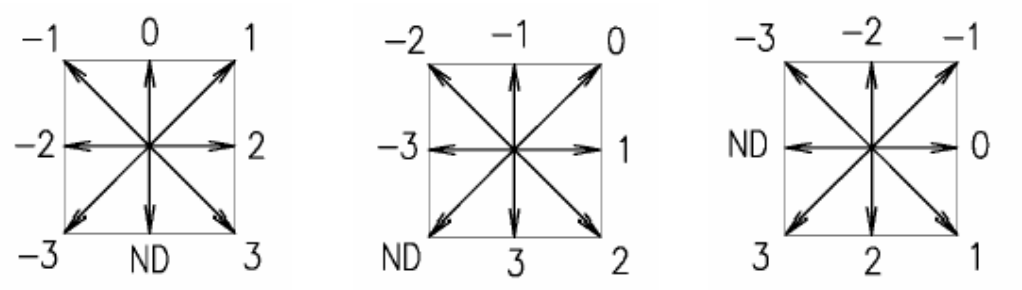
Obrázek 14: absolutní řetězový kód pro osmiokolí

Pokud máme hranový obrázek, ve kterém se mohou vyskytovat hranice, které nejsou dokonalé, obsahují tedy slepé větve nebo nejsou uzavřené, může nastat problém. Z toho důvodu bylo potřeba navrhnout robustní algoritmus, který by byl schopen se s touto skutečností vyrovnat. Navržený algoritmus předpokládá uzavřenou hranici a je schopen řetězový kód vytvořit i při výskytu slepých větví.

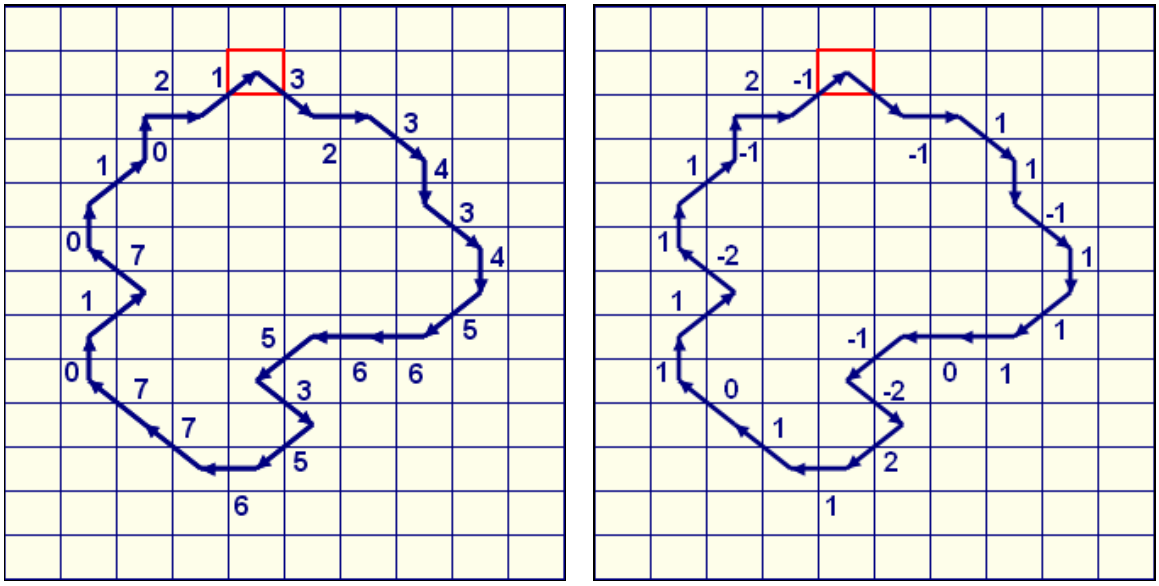
Takto získaná řada čísel se nazývá absolutní řetězový kód. Tento kód je závislý na natočení obrázku. Abychom tuto závislost odstranili, musíme zkonstruovat relativní řetězový kód. Ten se počítá z kódu absolutního jako jeho diskrétní derivace, tedy diference. Dostaneme tedy také posloupnost čísel, nyní ale v intervalu <-3; 3>. Jednotlivé hodnoty udávají změnu

směru průchodu hranových pixelů (edgelů) jako násobek 45°. Orientace je volena tak jak je patrné z obrázku 15. Kladných hodnot je tedy dosahováno ve směru hodinových ručiček. Označení směrů absolutního a relativního řetězového kódu je možno volit samozřejmě i jinak. Další modifikací řetězového kódu je kumulovaný řetězový kód. Ten se vypočítá z relativního řetězového kódu tak, že se postupně načítá (kumuluje). Vlastností tohoto kódu je, že pro uzavřený objekt (hranu objektu) začíná hodnotou o sedm menší než je koncová hodnota pokud hranu procházíme po směru hodinových ručiček, respektive hodnotou o sedm větší pokud procházíme proti směru hodinových ručiček. Tento kód se jeví jako vhodnější pro zpětnou interpretaci změny tvaru hranice.

Na obrázku 16 je uveden názorný příklad tvorby řetězového kódu. Jednotlivé šipky znázorňují přechody mezi sousedními edgely a ke každé z nich je přiřazena příslušná hodnota absolutního, respektive relativního řetězového kódu. V tabulce 1 jsou potom jednotlivé kódy vypsány.



Obrázek 15: relativní řetězový kód pro osmiokolí



(a) **(b)**

Obrázek 16: způsob tvorby řetězového kódu; (a) absolutní řetězový kód; (b) relativní řetězový kód

absolutní kód	3	2	3	4	3	4	5	6	6	5	3	5	6	7	7	0	1	7	0	1	0	2	1
relativní kód		-1	1	1	-1	1	1	1	0	-1	-2	2	1	1	0	1	1	-2	1	1	-1	2	-1
kumulovaný kód		-1	0	1	0	1	2	3	3	2	0	2	3	4	4	5	6	4	5	6	5	7	6

Tabulka 1: řetězové kódy odpovídající obrázku 16

5.2. Spektrální příznaky

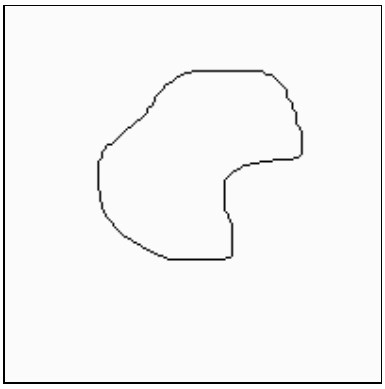
Byl tedy získán popis nezávislý na natočení obrázku, je však závislý na počátku určování řetězového kódu. Aby bylo možné ho použít, museli bychom zaručit, aby řetězový kód začínal vždy ze stejného bodu. Tento problém lze snadno vyřešit pomocí Fourierovy transformace. Konkrétně jsme použily rychlou diskrétní Fourierovu transformaci (FFT). Tato transformace totiž předpokládá, že vstupující signál je periodický. Nezáleží tedy na umístění počátku. Aplikací transformace na řetězový kód získáme spektrum, které obsahuje spektrální příznaky. Abychom u spektra předešli vysoké hodnotě na první spektrální čáře, tedy stejnosměrné složce, byla od kumulovaného kódu odečtena lineární funkce, která prochází počátkem souřadného systému [0, 0] a bodem se souřadnicemi [x_n, 7], kde x_n představuje poslední *x*-ovou souřadnici.

V důsledku použití osmiokolí je do řetězového kódu zanesena určitá nepřesnost, která vyplývá z neekvidistantnosti jednotlivých sousedních bodů, jak již bylo zmíněno. V případě absolutního řetězového kódu to nevadí, protože ten je závislý na natočení. V našem případě je ale modifikace řetězového kódu již na natočení nezávislá. To by mohlo vést k určité deformaci získaného spektra. Tuto nepřesnost by bylo možné odstranit tak, že kromě řetězového kódu by byla vytvářena i posloupnost, která by vyjadřovala vzdálenosti mezi jednotlivými sousedními hranovými pixely (edgely). V případě označení vzdálenosti horizontálně nebo vertikálně sousedících pixelů hodnotou jedna by pro diagonálně sousedící pixely vzdálenost odpovídala samozřejmě odmocnině ze dvou. My jsme neekvidistantnost osmiokolí zanedbali a uvažovali ekvidistantní dělení.

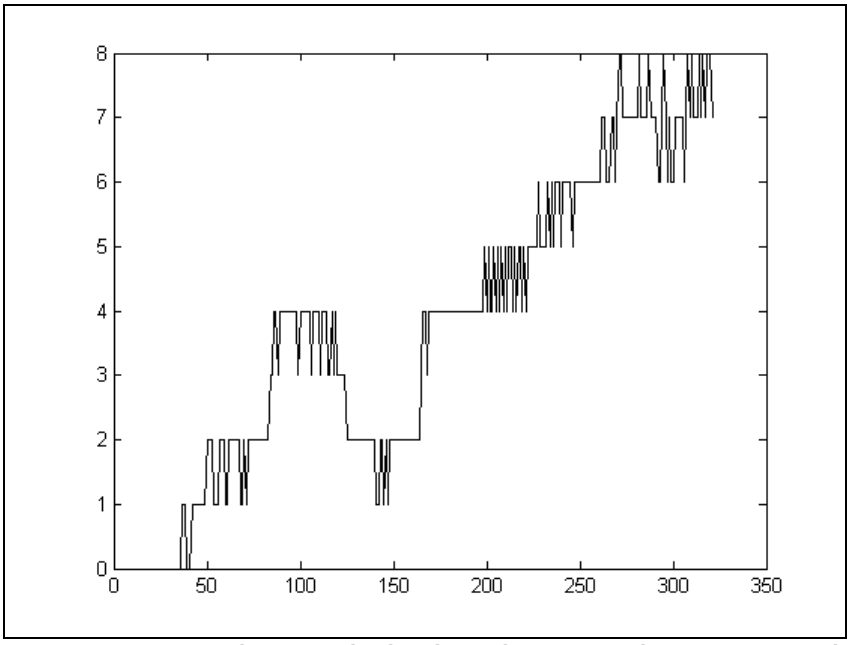
Ze zmíněného spektra získaného z upraveného kumulovaného kódu již určíme příznaky charakterizující jednotlivé objekty a to tak, že spektrum vhodně rozdělíme na několik pásem. Vzhledem k rozdílným délkám spekter jednotlivých objektů je potřeba každé spektrum rozdělit tak, abychom získali vždy stejný počet příznaků sečtením hodnot v jednotlivých pásmech. K tomuto účelu jsme spektra dělili logaritmicky a šířku jednotlivých pásem počítali vzhledem k délce konkrétního spektra. Logaritmické dělení zaručí lepší rozlišení pro nízké frekvence, což je pro náš případ výhodné.

Na závěr bychom tedy jenom shrnuli, že ze vstupních informací reprezentovaných binárními obrázky obsahujícími objekty zastoupené jejich hranicemi jsme výše zmíněným postupem získali příznaky, které jsou použity pro trénování nebo rozpoznávaní a následné rozřazování objektů do tříd, které je popsáno v kapitole 5.4.

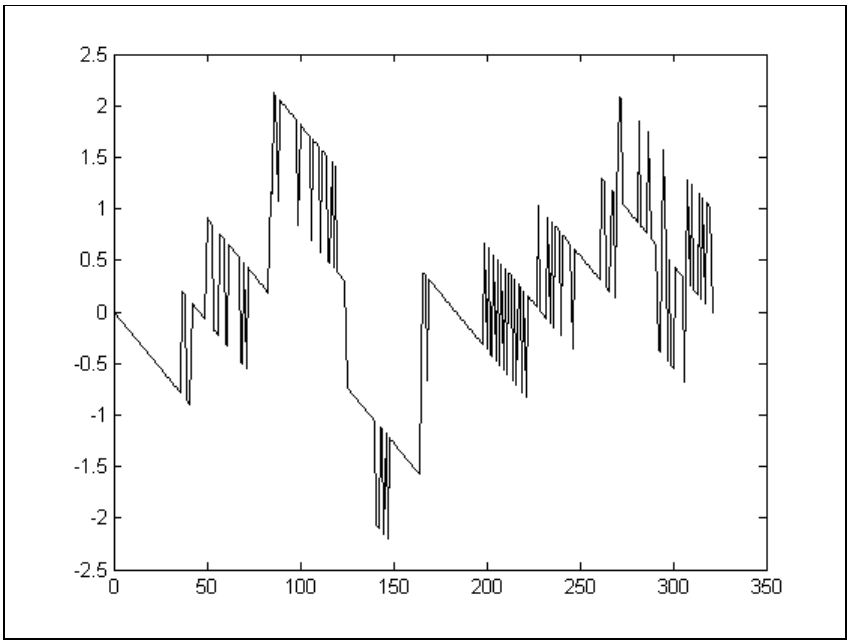
Na obrázcích 17-20 je ukázkový příklad.



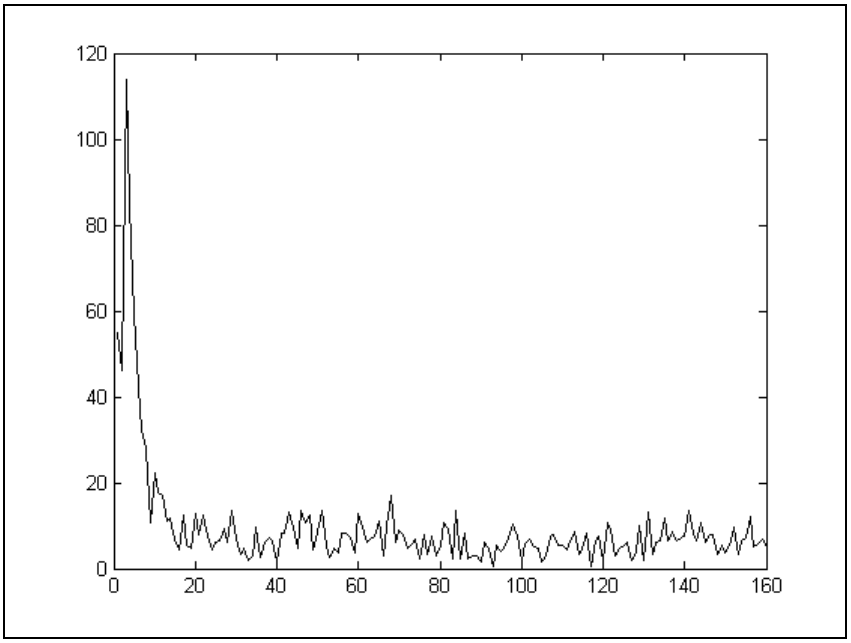
Obrázek 17: uzavřená hranice (obrázek je invertován kvůli tisku)



Obrázek 18: kumulovaný řetězový kód získaný z uzavřené hranice na obrázku 17



Obrázek 19: upravený kumulovaný kód vypočítaný z kumulovaného kódu na obrázku 18



Obrázek 20: spektrum vytvořené z upraveného kumulovaného kódu na obrázku 19

5.3. Další příznaky

Kromě příznaků spektrálních, které se získávají pomocí aplikace řetězového kódu a Fourierovy transformace, lze jako příznak použít kteroukoli charakterizující vlastnost jako například plocha, obvod, rozměry atd. Důležité je vždy umět zvolit co nejvhodnější příznaky. V případě plochy, obvodu nebo rozměrů je důležité neopomenout to, že tyto příznaky jsou závislé na rozlišení a jsou vhodné především tam, kde je velikost objektu klíčová. Zajímavým příznakem je nekompaktnost. Tento příznak se spočte z hodnot obvodu, respektive délky hranice objektu, a jeho plochy pomocí vzorce

$$nekompaktnost = \frac{(délka\_hranice)^2}{plocha}. \tag{26}$$

Tento příznak lze použít v případech kde velikost objektů jednoho typu není vždy stejná. Nutno je ale vědět, že se zvětšujícím se rozlišením obrázku bude tento příznak obecně narůstat nade všechny meze, což vyplývá z faktu, že délka hranice při zvětšování rozlišení roste narozdíl od plochy, která konverguje k určité hodnotě. Je nutno se tedy pohybovat v blízkých rozlišeních.

5.4. Trénování, testování a výpočet pravděpodobnosti

Pro vlastní rozpoznávání je použito příznaků, jejichž získávání je popsáno v předchozích kapitolách. Používá se dvou množin příznaků, a to množiny trénovací a testovací. Trénovací množina slouží jako reprezentativní vzorek na jehož základě se bude rozhodovat o zařazení objektu, respektive prvku do třídy. K tomu je určena množina testovací, ve které jsou příznaky rozpoznávaných objektů.

Samotné rozpoznávaní na základě zmíněných množin je prováděno metodou maximální pravděpodobnosti. Její výhodou je, že bere v úvahu kromě středních hodnot příznaků $\vec{x}$ = (x_1, x_2, x_3, ...) i jejich rozptyly σ^2. Podmíněná pravděpodobnost $P(A|B)$ říká jaká je pravděpodobnost jevu *A* za podmínky *B*. Z vektorů příznaků je definováno normální rozložení pro danou třídu T_k, kterou v našem případě představuje některý druh vlákna, a to zobecněným rozložením pro vstupní vektor příznaků:

$$P(\vec{x}|T_k) = \frac{1}{\sqrt{(2p)^p \det \sum}} e^{\left(-\frac{1}{2}(\vec{x}-\bar{x}_k)^T \sum^{-1} (\vec{x}-\bar{x}_k)\right)}, \tag{27}$$

Σ je kovariační matice definovaná takto:

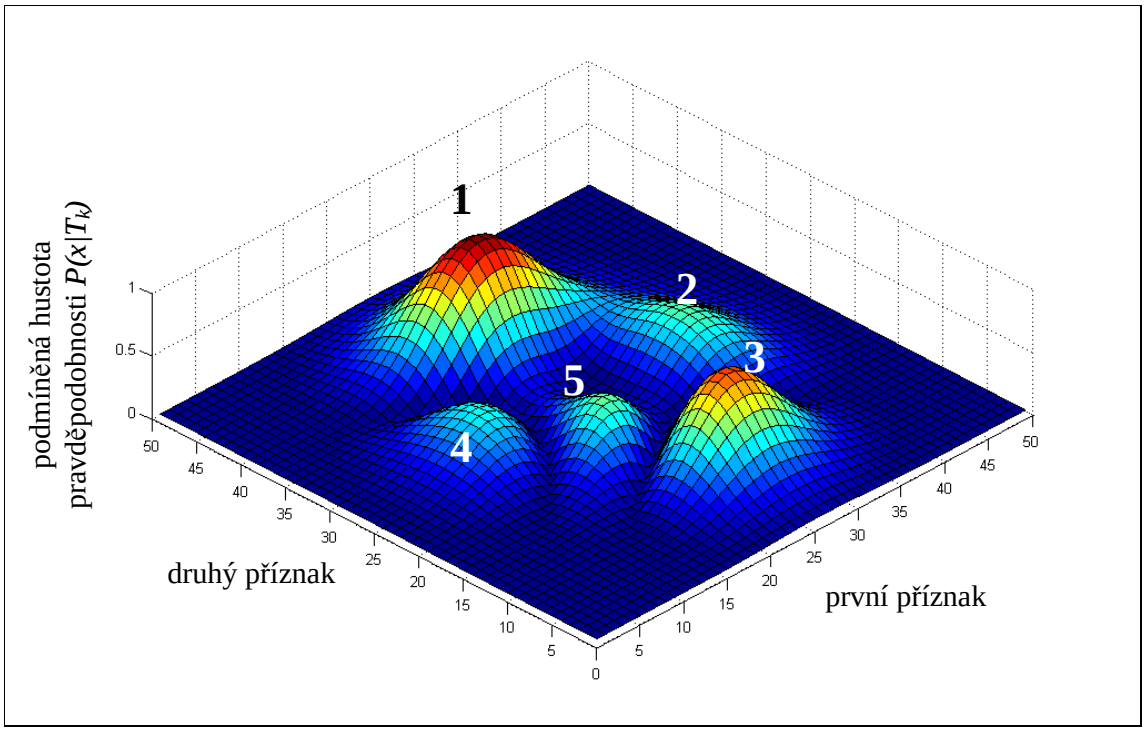
$$\sum = \begin{bmatrix} s_{11}^2 & s_{12}^2 & \cdots & s_{1n}^2 \\ s_{21}^2 & s_{22}^2 & \cdots & s_{2n}^2 \\ \cdots & \cdots & & \cdots \\ s_{n1}^2 & s_{n2}^2 & \cdots & s_{nn}^2 \end{bmatrix}, \tag{28}$$

a pro rozptyly σ^2 platí, že:

$$s_{ij}^2 = E\left[(x_i - \bar{x}_i)(x_j - \bar{x}_j)\right] \tag{29}$$

kde *E* znamená střední hodnota výrazu v závorce.

Jak může pravděpodobnostní rozložení vypadat pro vektor příznaků se dvěma složkami je vidět na obrázku 21.



Obrázek 21: podmíněná hustota pravděpodobnosti pro vektory příznaků se dvěma složkami

Za předpokladu znalosti apriorní pravděpodobnosti všech tříd můžeme použít Bayesovského rozpoznávání a tím rozhodnout o zařazení vektoru příznaků do příslušné třídy i ve sporných případech. Apriorní pravděpodobnost nás informuje o tom, jaké je celkové pravděpodobnostní zastoupení dané třídy. A pokud třídy T_1, ..., T_n tvoří úplný systém, potom pro libovolný vektor příznaků x platí

$$P(x) = \sum_{i=1}^{n} P(T_i) P(x \mid T_i), \tag{30}$$

$$P(T_k \mid x) = \frac{P(T_k) P(x \mid T_k)}{\sum_{i=1}^{n} P(T_i) P(x \mid T_i)} = \frac{P(T_k) P(x \mid T_k)}{P(x)} \quad \text{pro } k = 1 \ldots n \quad . \tag{31}$$

$P(x)$ se nazývá absolutní pravděpodobnostní hustota rozložení, $P(T_i)$ se označuje jako již zmíněná apriorní pravděpodobnost a $P(T_i|x)$ je výsledná aposteriorní pravděpodobnost.

Absolutní pravděpodobnostní hustota rozložení je nezávislá na zvolené třídě pro níž se provádí výpočet aposteriorní pravděpodobnosti. Můžeme počítat tedy bez této hodnoty. Výpočtem sice nezískáme přímo hodnoty aposteriorních pravděpodobností, ale při porovnání jednotlivých pravděpodobností získáme stejné výsledky.

6. MATLAB

MATLAB je vysoce výkonný programovací jazyk pro technické aplikace. Zahrnuje výpočetní, vizualizační a programovací části kde jsou problémy a jejich řešení vyjádřeny v matematickém zápisu. Typické využití MATLABu zahrnuje:

- matematické výpočty
- vývoj algoritmů
- získávání dat
- modelování a simulaci
- datovou analýzu a vizualizaci
- vědecké a inženýrské výpočty a grafiku
- vývoj aplikací zahrnujících grafické uživatelské rozhraní

MATLAB je interaktivní systém jehož základní datovou strukturou je matice, která nevyžaduje nastavení dimenzí. Tento fakt umožňuje řešení mnoha technických problémů zejména těch s maticovou a vektorovou formulací.

Jméno MATLAB vzniklo ze slov matrix laboratory (maticová laboratoř). Původně MATLAB vznikl jako interaktivní nadstavba pro usnadnění práce s knihovnami LINPACK a EISPACK pro práci s maticemi. V dnešní době je MATLAB spojen s knihovnami Lapack a Blas a přestavuje již mnohem více než jen nadstavbu maticové knihovny.

Vlastností, která patrně nejvíce přispěla k rozšíření MATLABu, je jeho otevřená architektura. Tato vlastnost vedla ke vzniku knihoven funkcí (ve skutečnosti adresáře s *.m a *.mex soubory), nazývaných toolboxy, které rozšiřují použití programu v příslušných vědních a technických oborech. MATLAB je úplný programovací jazyk, to znamená, že uživatelé v něm mohou vytvářet funkce nebo celé toolboxy "šité na míru" pro jejich aplikace. Vlastní funkce se způsobem volání nijak neliší od vestavěných funkcí a jsou uloženy v souborech v čitelné formě. Navíc jsou takto koncipované funkce snadno přenosné mezi různými platformami, na kterých je MATLAB implementován. Všechny moduly systému doprovází rozsáhlá tištěná i hypertextová on-line dokumentace. Otevřená architektura MATLABu inspirovala mnoho nezávislých firem k vývoji a distribuci vlastních produktů, které buď rozšiřují výpočetní prostředí MATLAB o další knihovny a nástroje nebo zajišťují propojení MATLABu s jinými specializovanými programy.

Další významnou předností programovacího jazyka MATLABu je jeho těsná integrace s jazykem Java. Objekty jazyka Java mohou být přímo použity programem v MATLABu, což umožňuje jednak vytvářet složitá grafická rozhraní s použitím grafických objektů Javy, jednak využít velkého množství volně dostupných knihoven, které byly v jazyce Java vytvořeny. Kromě toho je možné k MATLABu připojovat také moduly napsané v jazyce C a ve Fortranu.

6.1. Výpočetní jádro

Nejpodstatnější součástí numerického jádra MATLABu jsou algoritmy pro operace s maticemi reálných a komplexních čísel. MATLAB umožňuje provádět všechny běžné operace. Kromě datových typů jednodušších než tradiční matice podporuje MATLAB také typy složitější, jako jsou např. vícerozměrná pole reálných nebo komplexních čísel. Dalším datovým typem jsou tzv. pole buněk, tedy struktury podobné maticím, ve kterých ovšem každý prvek může být jiného typu. Podobně lze tvořit datové struktury, kde jsou prvky rozlišeny ne souřadnicemi, ale jménem, takže připomínají struktury známé z běžných programovacích jazyků. Skládáním těchto datových typů je pak možné vytvořit libovolně složité datové struktury. Vektor reálných čísel může v MATLABu představovat i polynom a operace s polynomy jsou v programu rovněž obsaženy. Vektory mohou také reprezentovat časové řady nebo signály a MATLAB obsahuje funkce pro jejich analýzu - výpočet střední hodnoty, hledání extrémů, výpočet směrodatné odchylky, korelačních koeficientů, rychlé Fourierovy transformace. MATLAB také podporuje speciální formát uložení tzv. řídkých matic, což jsou rozměrem velké matice, které obsahují většinu nulových prvků. Další významnou vlastností jazyka MATLABu je možnost práce s objekty. Ty uživateli umožňují rozšířit výpočetní prostředí o nové datové typy, na kterých je možno definovat libovolné funkce a operátory.

Samotné výpočetní jádro je implementováno s využitím nejmodernějších knihoven LAPACK a ARPACK. Výpočetní jádro je schopné adaptivně optimalizovat svou činnost podle konkrétní konfigurace uživatelského počítače (typ procesoru, cache, paměť, operační systém, ...), čímž je schopno skutečně využít výpočetní výkon. Pro uživatele z oblasti zpracování signálů je určena knihovna pro výpočet rychlé Fourierovy transformace (MIT FFTW Library), která používá nejrychlejší v současnosti známý algoritmus pro tuto úlohu.

6.2. Grafický subsystém

Grafika v MATLABu umožňuje snadné zobrazení a prezentaci výsledků získaných výpočtem. Je možné vykreslit různé druhy grafů: dvourozměrné, třírozměrné, histogramy, koláčové grafy a další. Všem grafickým objektům je možné téměř libovolně měnit vzhled, a to jak při jejich vytváření, tak po jejich nakreslení. Použit je algoritmus Z-buffer nebo technologie OpenGL. Každý nakreslený objekt má přiřazen identifikátor, jehož prostřednictvím je možné měnit vlastnosti objektu a tím i jeho vzhled. Vzhled grafických objektů je také možno měnit interaktivně, pomocí lišty nástrojů umístěné pod záhlavím obrázku. Grafický systém MATLABu, nazvaný Handle Graphics, dovoluje vkládat do obrazů ovládací prvky (tlačítka, apod.) a vytvořit tak aktivní graficky ovládané uživatelské rozhraní.

6.3. Guide

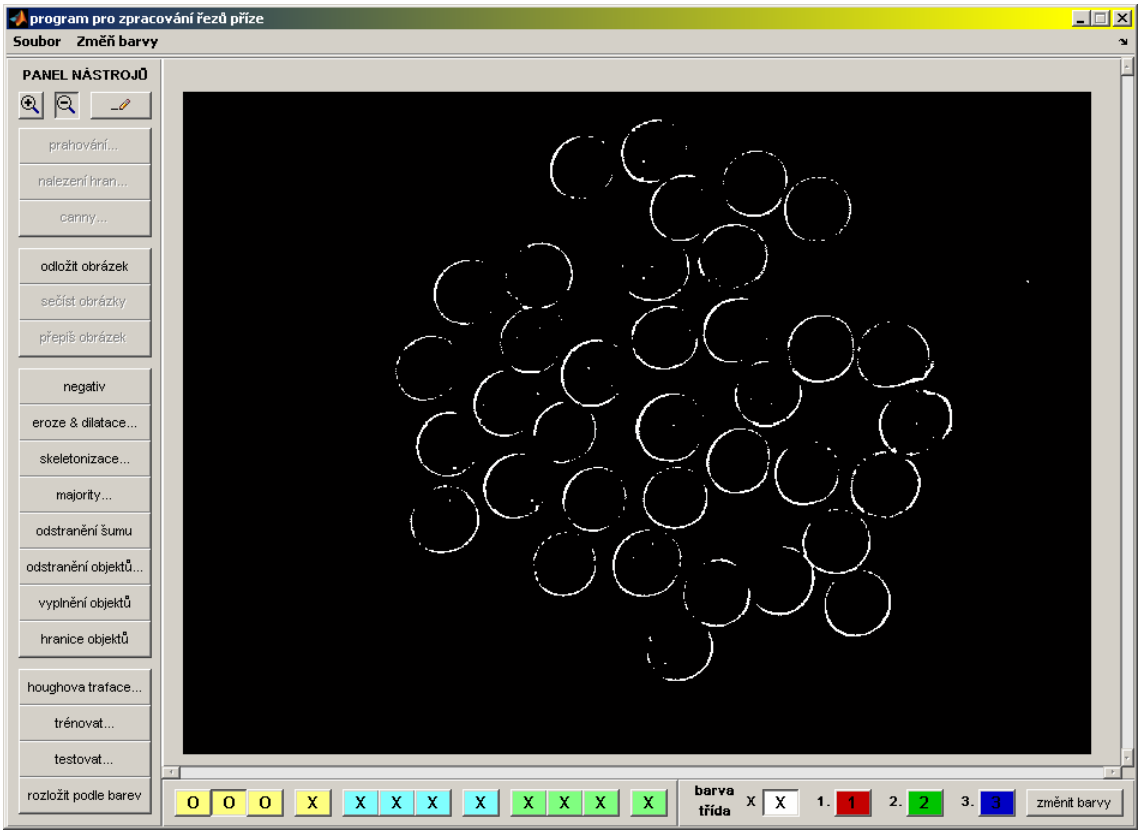
Guide (Graphical User Interface Development Environment) je součást MATLABu poskytující nástroje pro návrh grafického uživatelského rozhraní (GUI). Tyto nástroje velmi usnadňují návrh a tvorbu GUI. Guide obsahuje Guide Layout Editor pomocí něhož lze jednoduše rozmístit všechny komponenty na plochu formulář (figure). Navržené rozhraní je uloženo v souboru *.fig (figure). Guide automaticky vygeneruje *.m soubor který řídí operace v uživatelském rozhraní. Každá komponenta obsahuje události (callbacks) jako vytvoření, stisk tlačítka atd. na jejichž volání lze vykonat kód popř. zavolat funkci.

7. Návrh programu a algoritmů

7.1. Základní idea

K realizaci programu bylo použito programovacího jazyku MATLAB, který je popsán v kapitole 6. Za účelem snadné obsluhy pro všechny uživatele bylo navrženo grafické uživatelské rozhraní, k čemuž bylo použito součásti MATLABu Guide, o které je pojednáno v kapitole 6.3. Grafické rozhraní tedy slouží jako spojovací článek mezi operacemi programu a uživatelem. Program tedy může používat i ten, který vůbec neumí pracovat s MATLABem. Program je závislý na MATLABu a není tedy ve samospustitelné podobě. Ke zkompilování do samostatně fungujícího programu by muselo být použito compileru a všech potřebných knihoven. Compiler je standardní součástí MATLABu, knihovny už nikoliv. Vzhledem k tomu, že byl ale program vyvíjen pouze pro univerzitní účely nebyla jeho nezávislost na jádru MATLABu vyžadována.

Grafické rozhraní je tvořeno dvěma formuláři. Na obrázku 22 je hlavní formulář, který zprostředkovává všechny dostupné operace a slouží ke grafickému znázornění vstupních a výstupních obrázků. Druhým formulářem, který je na obrázku 23, je formulář nastavení barev. K čemu jsou nastavené barvy použity bude vysvětleno dále.



Obrázek 22: hlavní okno grafického uživatelského rozhraní

Obrázek 23: formulář nastavení barev

Jako grafické znázornění pro zpracovávaný digitální obrázek, respektive řez přízí, slouží tři obrázky, které jsou zpřístupňovány pomocí záložek na hlavním formuláři. Prvním obrázkem je vstupní RGB obrázek, který uživatel otevře pomocí nabídky hlavního formuláře. Druhým obrázkem je „vícebarevný binární“ obrázek, který je získán zpracováním vstupního obrázku. Třetím obrázkem je opět RGB obrázek, který vznikne vykreslením vícebarevného binárního obrázku do vstupního RGB obrázku. Význam tohoto řešení je například v editoru, kde je možno editovat vícebarevný binární obrázek podle obrázku vstupního.

7.2. „Vícebarevné binární“ obrázky

Protože je program určen i ke zpracování směsných přízí, tedy přízí s více druhy vláken, muselo být navrženo řešení, které zavádí tzv. „vícebarevné binární“ obrázky. To vyplývá z dvou podmínek. První podmínkou je možnost rozlišovat (značkovat) objekty, čímž jsou prakticky odlišovány typy vláken v řezech. Druhou podmínkou je možnost pracovat s obrázky jako binárními a například aplikace operací matematické morfologie.c

Praktická realizace tohoto řešení je ve více binárních obrázcích, kde jeden binární obrázek odpovídá jednomu typu objektu (vlákna). Tyto binární obrázky jsou pro zobrazení složeny do RGB obrázku. Každému obrázku je přiřazena barva, aby bylo vizuálně patrno jak je který objekt označen. Nastavení barev je uloženo v konfiguračním souboru jehož editace je možná pomocí formuláře nastavení barev. K dispozici jsou čtyři barvy. První barvou je bílá, která je také implicitní.

Řešení „vícebarevných binárních“ obrázků zahrnuje několik operací realizovaných pomocí čtyřech samostatných skriptů. První a druhý skript realizují převod binárního obrázku na RGB obrázek podle dané barvy a zpětnou operaci, tedy převod RGB obrázku na obrázek binární. Třetí skript vytváří „vícebarevný binární“ (ve skutečnosti RGB) obrázek ze vstupních binárních obrázků a jednotlivých barev, které jsou nastaveny. Čtvrtý skript realizuje opačnou operaci, tedy rozklad vícebarevného binárního obrázku na dílčí binární obrázky podle barevného nastavení.

7.3. Implementované operace

Do programu jsou implementovány následující operace:

- modifikace jasové stupnice – prahování a negativ (binární)
- hranové operátory – Sobel, Prewitt, LoG a Canny
- morfologické operace – dilatace, eroze, otevření a zavření
- odstranění šumu a odstranění malých objektů
- skeletonizace - ztenčování hran a skelet
- operace majority
- vyplnění objektů s uzavřenými hranicemi
- získání hranice objektů
- detekce kružnic pomocí Houghovy transformace
- rozpoznávání objektů pomocí spektrálních příznaků – trénování, testování
- rozložení obrázku na dílčí obrázky podle barev

Operace byly realizovány vlastními skripty nebo funkcemi Image Processing Toolboxu. Samostatnou částí programu je editor, který je popsán v kapitole 7.5.

7.4. Použité a navržené algoritmy

7.4.1. Hranové operátory

Pro LoG operátor a operátor Prewittové byly naprogromovány vlastní skripty. Vzhledem k jejich vyšší výpočetní náročnosti místo nich bylo ale použito funkce *edge*. Tato funkce zahrnuje kromě těchto dvou hranových operátorů i Sobelův operátor a Cannyho hranový detektor, které byly do programu též zahrnuty.

7.4.2. Modifikace jasové stupnice

K realizace prahování bylo použito funkce *im2bw*. Logická negace binární obrazové matice realizuje negativ, který je z praktických důvodů v programu dostupný pouze pro binární obrázky. Nutno dodat, že pro „vícebarevné binární“ obrázky funguje operace negativ tak, že všem objektům bez ohledu na jejich barvu přiřadí hodnotu pozadí (0). Pozadí je samozřejmě přiřazena hodnota objektu (1) a je ve „vícebarevném binárním“ obrázku zobrazena implicitní (bílou) barvou. Je tedy zřejmé, že dvojnásobná aplikace operace negativ způsobí to, že všechny objekty se zařadí do jedné skupiny, které odpovídá implicitní barva.

7.4.3. Morfologické operace

Operace matematické morfologie, tedy dilatace, eroze, otevření a uzavření, jsou realizovány pomocí funkce *bwmorph*. V případě dilatace byla řešena možnost kolize, která nastane v případě, že po aplikaci operace na „vícebarevný binární“ obrázek dojde k výskytu pixelů více typů (barev) se stejným umístěním v obrazové matici. To je zapříčiněno provedením operace na dílčí obrázky podle barevných složek a následnému složení těchto dílčích výsledků. Tento problém byl vyřešen nastavením priority barev sestupně od první z nich (tedy implicitní barvy).

7.4.4. Další obrazové operace

Odstranění malých objektů bylo realizováno funkcí *bwareaopen*. Jako vstupem funkci slouží číslo, který říká kolika pixely musí být objekt tvořen, aby nebyl odstraněn.

K odstranění šumu je použito funkce *bwmorph*. Za šum jsou považovány všechny izolované pixely hodnoty 1. Jsou to tedy pixely objektu, které nesousedí s žádným pixelem objektu.

Pomocí funkce *bwmorph* je realizována i operace majority. Tato operace nastavuje pixely pozadí s alespoň pěti sousedními pixely objektu také na hodnotu objektu (tedy logickou jedničku). Účelem této operace je tedy zaplnění malých děr nebo zálivů a získání hladší hrany. Operaci je možno provádět vícenásobně.

Ztenčování hran a získávání skeletu (tedy hrany tloušťky jedna), které jsme zařadili do oddílu skeletonizace je realizováno opět pomocí funkce *bwmorph*. Opakovaným ztenčováním hran až do stavu, kdy se již výsledný obrázek nemění, získáme skelet.

Vyplnění objektů provádí funkce *imfill*. Vyplnění je prováděno samozřejmě pouze u objektů s uzavřenou hranicí. Pokud uvnitř některého objektu s uzavřenou hranicí leží objekt jiného typu (barvy), provedením této operace je tento objekt zaniká.

Funkce *bwboundaries* slouží k získání hranic o tloušťce jedna ze všech objektů ve vstupním obrázku.

Rozložení obrázku podle barev vytvoří z „vícebarevného binárního“ obrázku čtyři binární obrázky, které obsahují jednotlivé objekty podle jejich původního zařazení odpovídající příslušné barvě.

7.4.5. Algoritmus hledající objekty pomocí Houghovy transformace

Houghovu transformace byla realizována pro přímky a kružnice v samostatné funkci a implementována do programu. Pro zpracování příčných řezů textilií je použito pouze vyhledávání kružnic, kterými se dají aproximovat průřezy některých typů vláken.

Jako vstup je vyžadován binární obrázek, který je možno z vstupního barevného nebo šedotónového obrázku získat například prahováním nebo použitím hranového detektoru a dalším předzpracováním.

Důsledkem robustnosti transformace v podobě tolerance rozdílů objektů a odolnost proti zarušení je vysoká výpočetní náročnost algoritmu. Výpočetní časy u dodaných obrázků řezů přízí se pro automaticky nastavenou velikost akumulátoru pohybovaly řádově v minutách, proto bylo nutné modifikacemi algoritmu časy redukovat. První modifikací bylo nahrazeno toto automatické nastavení velikosti akumulátoru, které nastavovalo zbytečně velký interval hodnot poloměru, nastavením uživatelským. Interval poloměru byl automaticky nastavován podle velikosti obrázku a to tak, aby bylo možno detekovat kružnice s maximálním průměrem odpovídajícím menšímu rozměru obrázku. Minimální poloměr omezen nebyl. To je samozřejmě pro tuto aplikaci, tedy rozpoznávání vláken, zbytečné. Touto úpravou se povedlo časy snížit více než desetkrát, jsou ale kladeny nároky na uživatele. Pokud uživatel velikost akumulátoru, respektive interval parametru *r*, nevhodně nastaví, může nastat, že nedojde k časové úspoře při výpočtu nebo jsou výsledky chybné. Druhou modifikací je nastavení koeficientu náhodnosti, který říká, zda se bude výpočet a akumulace provádět pro každý pixel objektu nebo jen pro některé. Pro implicitní hodnotu jedna je výpočet prováděn pro každý pixel, pro hodnotu n je výpočet prováděn v průměru pro každý n/2-tý pixel. Je použit generátor náhodných čísel s rovnoměrným rozdělením, který generuje hodnotu od nuly do $n-1$, která říká, kolik pixelů objektů se při výpočtu vynechá.

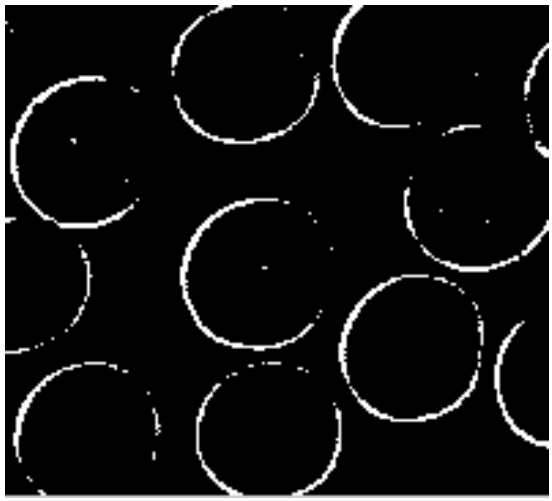
Další úpravou je možnost použít filtraci akumulátoru. Po dokončení akumulace před hledáním maxim akumulátoru je možno použít průměrovací filtr se symetrickou maskou velikosti tři nebo pět. Maska je trojrozměrná podle počtu dimenzí akumulátoru. V důsledku počtu dimenzí je výpočet algoritmu filtrace časově náročný i přes jeho optimalizaci. Doba filtrace je samozřejmě závislá na velikosti akumulátoru a je realizována vlastním skriptem.

Poslední fází je získání parametrů hledaných objektů reprezentovaných maximy akumulátoru. Pro tento účel bylo použito hledání globálních maxim akumulátoru. Nalezené maximum je v akumulátoru vynulováno a hledání se opět opakuje. Tento postup končí, pokud se v akumulátoru již nenachází hodnota vyhovující stanovené hranici.

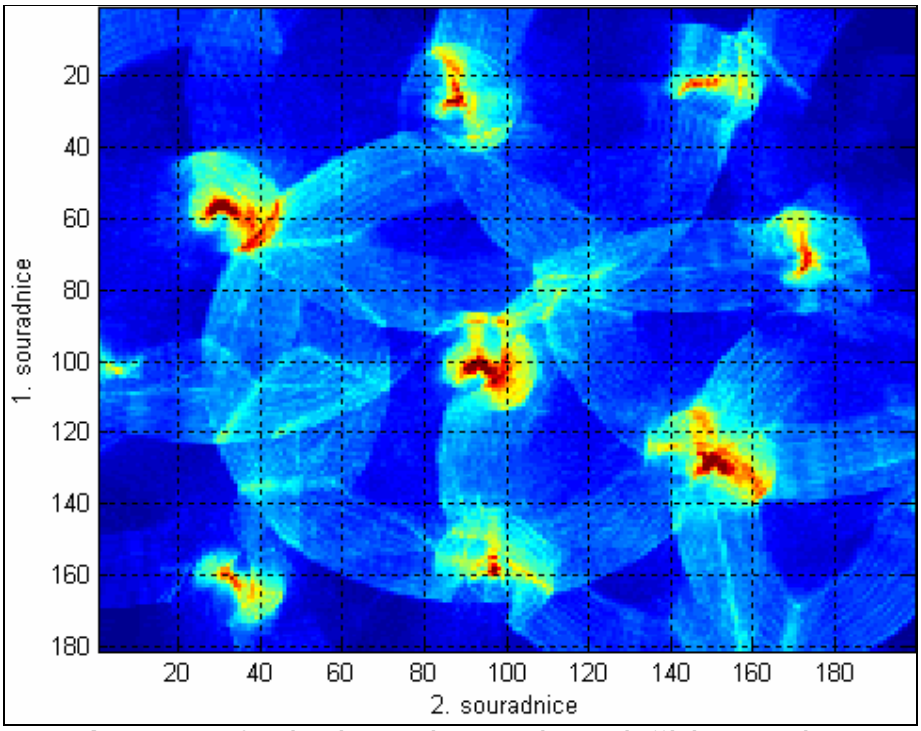
Před spuštěním algoritmu je tedy potřeba zadat interval poloměru, zmíněný koeficient náhodnosti a procentuální práh. Hodnota procentuálního prahu je použita k výpočtu hranice, která určuje, jaké nejmenší hodnoty z akumulátoru budou považovány za úspěšně detekovaný objekt. Tato hranice je počítána jako procentuální část globálního maxima akumulátoru. Dále je možno zvolit použití zmíněné filtrace akumulátoru. Při vykonávání algoritmu je uživatel informován v jaké fázi a s jakým aktuálním průběhem se algoritmus nachází. Jako výstup jsou získány parametry kružnic, podle kterých je zkonstruován obrázek pro kontrolu správnosti detekce, a odpovídající hodnoty akumulátoru. Míra korespondence hledaných a nalezených objektů je jednoznačně patrná z obrázku, který vznikne vykreslením zkonstruovaného obrázku do obrázku původního, který byl pro hledání objektů použit.

Obrázek 24: obrázek zobrazující vstupní a výstupní objekty umožňující vizuální kontrolu úspěšnosti detekce

Po výpočtu algoritmu je kromě výstupního obrázku zobrazen také akumulátor. Protože je v případě kružnic, jak již bylo zmíněno, akumulátor třírozměrný, je pro zobrazení zanedbán parametr poloměru a jsou zobrazeny maximální hodnoty pro všechny dvojice hodnot parametrů *[a, b]*, což se dá jednoduše přestavit jako promítnutí lokálních maxim do plochy o souřadnicích *[a, b]*. Získané hodnoty jsou převedeny do barevné teplotní škály a vykresleny do obrázku 26. Nejnižší hodnotě akumulátoru podle zvolené barevné stupnice odpovídá modrá, nejvyšší hodnotě, která ukazuje na detekovanou kružnici, odpovídá červená.



Obrázek 25: binární obrázek vstupující do Houghovy transformace



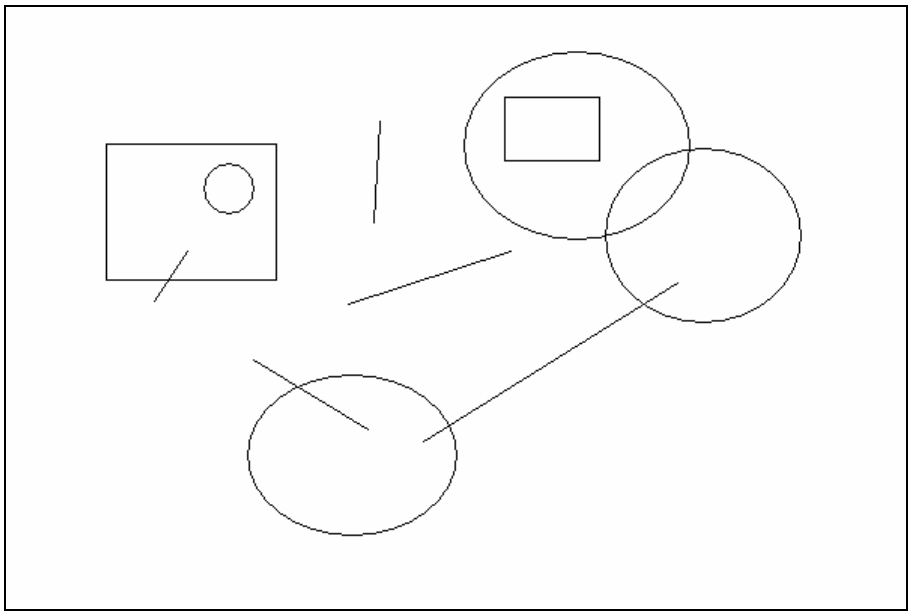
Obrázek 26: grafické znázornění akumulátoru náležícího k obrázku 25

Jako prakticky použitelný výstup ale neslouží již zmíněný obrázek, ale tabulka parametrů kružnic s příslušnými hodnotami akumulátoru. Tuto tabulku lze získat v souboru typu „csv“, který je možno vytvořit po vykonání algoritmu. Kromě tabulky parametrů je do souboru uložen i datum a čas, jméno zpracovávaného obrázku, jeho velikost, velikost akumulátoru a vstupní parametry funkce zadané uživatelem.

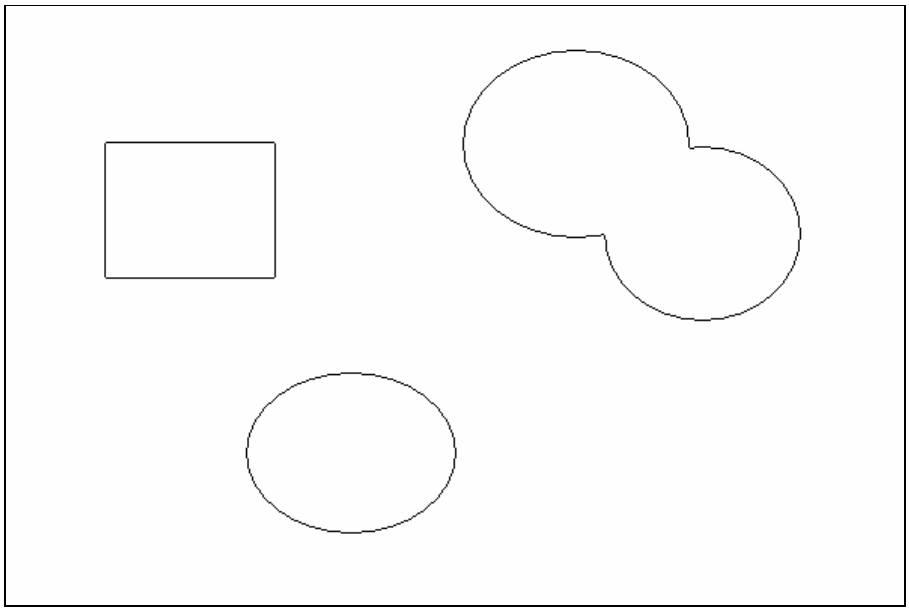
7.4.6. Algoritmus rozpoznávající objekty na základě příznaků

Rozpoznávání objektů na základě jejich příznaků je prováděno v několika krocích. Prvním krokem je úprava vstupního binárního obrázku, aby obsahoval pouze uzavřené hranové objekty. Za tímto účelem byl naprogramován skript, který toho docílí několika operacemi.

Skript umí odstranit objekty s neuzavřenými hranicemi, objekty které jsou tvořeny překrytím objektů s uzavřenými a neuzavřenými hranicemi. V případě překryvu dvou objektů s uzavřenými hranicemi dojde k jejich sloučení do jednoho objektu s uzavřenou hranicí. Tyto zmíněné případy jsou ilustrovány na obrázku 27. Na obrázku 28 potom vidíme výstupní obrázek. Ve skriptu je použito morfologických funkcí skelet, odstranění šumu, majority, funkce vyplnění objektů, získání hranice objektů a funkce, která hledá koncové body objektů od kterých trasuje a maže hranici dokud nenarazí na další koncový bod nebo průsečík.



Obrázek 27: vstup do algoritmu pro zisk uzavřených hranových objektů (obrázek je invertován kvůli tisku)



Obrázek 28: výstup z algoritmu upravujícího obrázek pro aplikaci řetězového kódu (obrázek je invertován kvůli tisku)

Na takto předpřipravený obrázek již lze úspěšně aplikovat algoritmus získání řetězového kódu a následně i spektrálních příznaků. To je realizováno celé v jednom skriptu jehož vstupem je kromě binárního obrázku i počet spektrálních příznaků. Skript umožňuje také výpočet příznaku poměru rozměrů objektu a příznaku nekompaktnosti. Z programu není umožněno měnit počet spektrálních příznaků ani volit použití dalších dvou příznaků. Implicitní nastavení je deset příznaků spektrálních.

Tento skript je použit samozřejmě jak pro trénování tak pro testování. Při trénování je uživatel vyzván k zadání názvu jednotlivých tříd, do kterých jsou získané vektory příznaků rozřazovány. Kromě zařazení objektu, respektive vektoru příznaků, do třídy je možno objekt také úplně vyřadit. Po dokončení trénování je trénovací množina uložena ve formě souboru *.mat.

Trénovací množiny je použito při testování. Nejdříve uživatel vybere soubor obsahující trénovací množinu, který je načten. Jsou zobrazeny základní informace jako názvy tříd a počty prvků v jednotlivých třídách. Dalším krokem je opět získání vektorů spektrálních příznaků, které představují testovací množinu. Pro vstup je opět nutný obrázek obsahující pouze hranové objekty, které lze získat aplikací zmíněného algoritmu pro předzpracování. Uživatel může zvolit rozpoznávání všech objektů nebo pouze objektů, které nejsou ještě zařazeny do žádné ze tří tříd a náleží tedy do třídy čtvrté, jejíž barva je bílá.

Posledním krokem je již výpočet pravděpodobnosti. To je realizováno skriptem do něhož vstupují trénovací a testovací vektory a jehož výstupem jsou hodnoty pravděpodobnosti příslušností prvků do dané množiny. Praktickým výstupem je ale zařazení objektu do některé ze tříd podle pravděpodobnosti a obarvení objektu barvou odpovídající dané třídě.

7.5. Grafický editor

Aby bylo možno provádět celé zpracování řezů v navrženém programu bylo nutno program doplnit o grafický editor. Ten obsahuje některé funkce běžně známé z jiných editorů. Mezi tyto funkce patří kreslení bodů, úseček a lomených čar. Dalšími funkcemi je mazání bodů, výběru, objektu a zvolené barvy. Kromě mazání je možno u objektu i změnit barvu.

Doplňkovou funkcí, která ale vybočuje oproti jiným editačním funkcím je možnost cyklicky procházet objekt po objektu. Vybraný objekt se automaticky přiblíží pomocí lupy a poté je možno provést jeho editaci. Je možno kromě dokreslení nebo umazání volbou barvy změnit jeho zařazení v rámci třídy nebo ho smazat. Objekty se procházejí postupně na základě funkce *regionprops*. Tato funkce je součástí Image Processing Toolboxu a je použita k získání informací o poloze daného objektu v obrázku. Objekty jsou postupně řazeny podle své polohy. Primárně je brána horizontální poloha ve směru zleva doprava, sekundárně je použita vertikální poloha ve směru seshora dolů.

Editor je doplněn o možnost vrácení jedné akce pomocí záloh editovaných obrázků.

7.6. Optimalizace a časová náročnost algoritmů

Všechny naprogramované algoritmy realizující jednotlivé funkce byly psány se snahou dosáhnout co největší optimalizace. Programovací jazyk MATLAB, který byl k tomuto účelu použit, je optimalizován na maticové výpočty, proto byla snaha nepoužívat cykly, které jsou poměrně časově náročné. V případě delších cyklů, pokud je to možné, je snížení výpočetní náročnosti dosaženo nahrazením maticovými výpočty při použití rozsahu indexů. V případě kratších cyklů byl v několika případech cyklus nahrazen příslušným počtem po sobě jdoucích příkazů vzájemně se lišících indexy. V mnoha případech ale bylo nutno cyklu použít.

Čas vykonávání jednotlivých algoritmů se také odvíjel od použité verze MATLABu. Například u algoritmu optimalizované Houghovy transformace byly časy pro stejné výchozí podmínky, tedy pro stejná vstupující data a stejný hardware a operační systém, velmi rozdílné. Pro MATLAB verze 6.5 se čas vykonávání rovnal 102.9s a u verze 7.01 bylo naměřeno času 28.1s. Ke zjištění výpočetních časů bylo použito funkcí *tic* a *toc*. Rozdíl časů v závislosti na verzi MATLABu je zřetelný a v tomto případě je u nové verze dosaženo více než trojnásobné snížení výpočetního času. To je způsobeno použitím nového výpočetního jádra. Cílem této práce ale není porovnat výkonnost jednotlivých verzí MATLABu. Byla pouze snaha dosáhnout co nejmenších časů i v případě náročných algoritmů, aby byl program z uživatelského hlediska přijatelný.

8. Závěr

Problematika rozpoznávání vláken v řezech a zpracování řezů je velmi rozsáhlá. Možných přístupů k řešení je celá řada. Tato diplomová práce pokrývá pouze některé z nich. Cílem nebylo pouze navrhnout algoritmy pro zpracování řezů, ale také ověřit vhodnost zvolených přístupů.

Použití hranových operátorů nebo jejich kombinace je sice vhodnější než použití prostého prahování histogramu, ale obsahuje problém, kterým je vnitřní i vnější hrana vláken. V případě Cannyho hranového detektoru jsme se přesvědčily, že hrany jsou sice detekovány nejlépe, ale také nastává největší problém s dvojnásobnými hranami. Použitím dalších operací v kombinaci s editorem lze získat binární obrázek, který je tvořen hranami jednotlivých vláken. Plně automatickým zpracováním dosáhneme dobrých výsledků pouze u velmi kvalitních obrázků. Kvalita dodaných obrázků byla ve většině případů nízká.

Algoritmus založený na Houghově transformaci hledající kružnice, kterými aproximuje průřezy vláken, poskytoval uspokojivé výsledky. Ikdyž je Houghova transformace z principu celkem robustním nástrojem, menší kvalita vstupních obrázku se přesto projevila na výsledcích o něco více než bylo očekáváno.

Rozpoznávání typů vláken na základě spektrálních příznaků a následné rozřazování do tříd fungovalo velmi dobře s úspěšností pohybující se kolem 70 - 75%. Nevýhoda je ale ta, že vstupní obrázek musí obsahovat pouze uzavřené hranové objekty čehož není normálně nikdy dosaženo. Algoritmus navržený za účelem předzpracování obrázků pro výpočet řetězového kódu byl ve většině případů úspěšný, ale i přesto bylo někdy potřeba obrázek předzpracovat i pomocí editoru.

Tato práce kromě ověření vhodnosti zvolených postupů zjišťuje i možnosti v této problematice. Do budoucna je počítáno s pokročováním automatizace zpracovávaní řezů a možným využitím některých algoritmů.

Použitá literatura

[1] Hlaváč V., Sedláček M. *Zpracování signálů a obrazů*. 2. vyd., skriptum ČVUT, Praha 2002, ISBN 80-01-02114-9

[2] Massey University – Institute of Information & Mathematical Science – Computer Science. [online] [cit. 22. listopadu 2004].
Dostupné na www: <http://cs-alb-pc3.massey.ac.nz/notes/59318/l11.html>

[3] Bubeník F., Pultar M., Pultarová I. *Matematické vzorce a metody*. vydavatelství ČVUT, Praha 1994, ISBN 80-01-01643-9

[4] Matela L. *Rozpoznávání druhů stromů podle tvarů jejich listů*. [online] K7 – vědecko populární časopis Fakulty mechatroniky TU v Liberci, 02/2004 [cit. 1. září 2005].
Dostupné na www: <http://k7.vslib.cz/files/k7_04_2.pdf>. ISSN 1214-7370

[5] Dušek F. *MATLAB a SIMULINK úvod do používání*. skriptum Univerzita Pardubice, 2002, ISBN 80-7194-475-0

[6] Sedláček M., Šmíd R *MATLAB v měření*. skriptum ČVUT, Praha 2004, ISBN 80-01-02851-8

[7] The Mathworks - Image Processing – Demos and Webinars. [online]. 2005 [cit. 10. září 2005].
Dostupné na www:
<http://www.mathworks.com/applications/imageprocessing/index.html>

[8] Morphology-based Operations. [online]. 2005 [cit. 2. září 2005].
Dostupné na www:
<http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Morpholo.html#Heading99>

[9] Contour Representations - Chain code. [online]. 2005 [cit. 2. září 2005].
Dostupné na www:
<http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Contour.html#Heading26>

[10] Bock Rudolf K. Hough Transform. [online]. 2005 [cit. 10.prosince 2004].
Dostupné na www:
<http://ikpe1101.ikp.kfa-juelich.de/briefbook_data_analysis/node122.html>

[11] Fisher R., Perkins S., Walker A. and Wolfart E. *Image processing learning resources.* [online]. 2005 [cit 5.září 2005]
Dostupné na www:
<http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm>

Přílohy

Příloha 1: Popis a ovládání programu

Všechny dostupné funkce analýzy obrazu a metody rozpoznávání jsou vloženy do programu s grafickým uživatelským rozhraním vytvořeným v Guide (již zmíněná součást MATLABu).

Program obsahuje jeden hlavní formulář. V horní liště formuláře se nachází nabídka. V levé částí se nachází panel nástrojů. Většinu místa zabírá náhled obrázků, ke kterému náleží pod ním se nacházející panel záložek, který je popsán dále. V pravé dolní části je umístěn panel barev. Tento panel obsahuje čtyři přepínače sdružené do skupiny (tudíž je možno, aby byl stisknut pouze jeden z nich). Jejich funkce je volba aktuální barvy, která se používá k editaci obrázků. Každý přepínač je vybarven odpovídající barvou, kterou volí. Kromě přepínačů je na tomto panelu umístěno i tlačítko změny nastavení barev. Toto tlačítko otvírá formulář barev, který umožní jejich nastavení. Formulář obsahuje ke každé barvě náhled a hodnoty barevných složek v barevném prostoru RGB (Red Green Blue). Nastavení barvy je možno přímým vložením hodnot jednotlivých barevných složek nebo použitím dialogu. Na formuláři se nacházejí čtyři tlačítka, která umožňují aktualizovat náhledy po editaci barevných složek, uložení aktuálního nastavení do konfiguračního souboru, načtení nastavení z konfiguračního souboru a zavření formuláře. Z konfiguračního souboru je nastavení načítáno při každém spuštění programu, provedené změny jsou tudíž zachovány i po zavření programu.

Výše zmíněný panel záložek obsahuje tři skupiny barevně odlišených záložek. Každá skupiny odpovídá jednomu nezávislému pracovnímu projektu. Je tedy možno pracovat zároveň na třech projektech. Všechny skupiny obsahují čtyři záložky. První záložka aktivuje náhled na vstupní barevný nebo šedotónový obrázek, který uživatel otevře pomocí nabídky. Pomocí druhé záložky je k dispozici náhled na výstupní obrázek z jednotlivých operací. Tento obrázek je v principu binární, ale objekty mohou být odlišeny k čemuž je použito barevné nastavení. Toto řešení bylo nazváno jako „vícebarevný binární“ obrázek a je popsáno v kapitole 7.2. Z tohoto důvodu není tento obrázek vždy binární. Pokud nejsou objekty zařazeny do skupin, obrázek je skutečně binární a barva objektů je bílá. Pokud jsou v obrázku objekty rozřazeny do více skupin, objekty každé skupiny jsou obarveny příslušnou barvou. Potom je obrázek již barevný se složkami R, G a B, ale při úkonech je obrázek rozložen na více binárních obrázků, se kterými je pracováno individuálně a po dokončení operace je opět vytvořen jeden obrázek. Poslední čtvrtá záložka slouží pro zálohu obrázku. Toho se dá později využít například pro součet mezivýsledků do výsledného obrázku. K tomu slouží tři níže popsané operace.

Panel nástrojů nacházející se v levé části formuláře osahuje především tlačítka jednotlivých operací sdružených do skupin, které je možno provádět s obrázky. Na panel jsou ještě doplněna dvě tlačítka lupy, a to pro zvětšení a pro vrácení náhledu do původní velikosti. Pomocí tlačítka pro zvětšení se volí obdélníkový výběr, který chceme zobrazit do celé zobrazovací plochy. Funkce tlačítka pro vrácení do původní velikost zobrazí opět obrázek celý v maximálním zobrazovacím rozlišení, ve kterém se vejde na zobrazovací plochu. Další tlačítko je tlačítko editace obrázku, které umožňuje kreslit a mazat jednotlivé objekty. Funkce editace je popsána v kapitole 7.5. Všechny ostatní operace jsou sdruženy do skupin. První skupina obsahuje operace s barevnými nebo šedotónovými obrázky a zahrnuje operaci prahování a hranové operátory. Další skupina tří operací umožňuje zálohovat, sčítat nebo přepisovat obrázky získané pomocí jednotlivých operací. To je výhodné v případě, že potřebujeme porovnat dva postupy zpracování, nebo sloučit dílčí výsledky nebo jenom odložit získaný obrázek, abychom se k němu mohli vrátit. Tyto tří operace jsou prováděny s obrázky, které jsou umístěny na druhých a čtvrtých záložkách ve skupině.

Další a současně největší skupinou operací jsou operace prováděné na binárních obrázcích. Těmito operacemi je binární negativ, eroze, dilatace, otevření, uzavření, skeletonizace, operace majority, odstranění šumu, odstranění objektů podle jejich velikosti, vyplnění objektů s uzavřenou hranicí a operace, kterou je možno získat hranice jednotlivých objektů obrázku. Jak jednotlivé operace fungují je popsáno v kapitolách 7.4.2 - 7.4.4.

Poslední skupina zahrnuje algoritmus vyhledávání objektů, v našem případě kružnic, pomocí Houghovy transformace. Houghova transformace a i samotný algoritmus jsou pojednány v kapitolách 4 a 7.4.5. Dalším algoritmem v této skupině je algoritmus, který slouží k rozpoznání objektů pomocí příznaků získaných především z hranic jednotlivých objektů. Tento algoritmus narozdíl od Houghovy transformace obsahuje dvě tlačítka. První z nich umožňuje trénování, druhé z nich testování. Tento postup i realizovaný algoritmus jsou popsány v kapitolách 5 a 7.4.6.

Většina operací má vlastní panel, kde je možno danou operaci nastavit. Operace záloha, sečtení a přepsání obrázku, negativ, odstranění šumu, vyplnění objektu, získání hranice objektů a rozložení obrázku podle barevného nastavení neobsahují žádné nastavení a proto jsou vykonány okamžitě.

Pro operaci prahování je možno zvolit hodnotu prahu nebo ponechat automatické nastavení. Další panel nabízí Sobelův operátor, operátor Prewittové a LoG operátor. Všechny tři oparátory umožňují nastavit hodnotu prahu nebo ponechat nastavení automatické. U Sobelova hranového operátorů a operátoru Prewittové lze nastavit, zda chceme hledat hrany všechny nebo omezit hledání na vertikální, respektive horizontální hrany. LoG operátor ještě zahrnuje možnost nastavení parametru sigma, který říká v jak velkém okolí operátor pracuje (pokud je okolí velikosti n x n tak platí, že n=ceil($sigma$*3)*2+1, kde „ceil“ označuje zaokrouhlování nahoru). V případě Cannyho hranového je vedle automatického nastavení možno použít uživatelské nastavení, které umožňuje zvolit hodnoty dvou prahů popřípadě i hodnotu sigma.

Pro operace eroze, dilatace, ztenčování hran (skeletonizace) a operaci majority je možno nastavit jediný parametr, a to počet opakování průběhu algoritmu.

U algoritmu hledajícím kružnice pomocí Houghovy transformace je možné nastavit procentuální práh, koeficient náhodnosti, minimální a maximální poloměr nebo zvolit případné použití filtrace akumulátoru. Význam těchto parametrů je vysvětlen v kapitole 7.4.5. Po vykonání algoritmu je umožněno uložení výsledků do souboru.

Po otevření panelu pro trénování, respektive testování, a následném spuštění operace je uživatel dotázán, zda chce provést předzpracování vstupních obrázků. Předzpracování je umožněno i pomocí tlačítka umístěného na obou panelech. Trénování i testování probíhá v několika krocích.

Prvním krokem trénování je zadání názvů tříd objektů pokud vytváříme novou trénovací množinu. Pokud chceme rozšířit stávající množinu, nezadáváme již názvy tříd, ale načteme databázi s trénovací množinou, kterou jenom rozšíříme. Dalším krokem je aplikace algoritmu, kterým získáme vektory spektrálních příznaků jednotlivých vektorů. Posledním krokem je již rozřazování objektů do tříd. Na obrázku se postupně zobrazují jednotlivé objekty, které pomocí tří tlačítek tříd a čtvrtého tlačítka, které objekt vyřadí, můžeme vložit do příslušné třídy.

Prvním krokem testování je načtení databáze obsahující trénovací množinu. Po jejím načtení jsou zobrazeny základní informace ve formě názvů tříd a počtu prvků ve třídách. Poté je umožněno provést algoritmus, kterým získáme vektory příznaků. Do testování (rozpoznávání objektů) můžeme zařadit všechny objekty nebo pouze ty objekty, které ještě nejsou zařazeny v žádné ze tří tříd, ale příslušejí do implicitní třídy, které odpovídá bílá barva. Na základě vypočtených pravděpodobností jsou rozpoznávané objekty obarveny barvou příslušející množině, do které byly zařazeny.

Editace obrázků grafickým editorem

V horní části panelu nástrojů se nachází tlačítko, které otevře panel pro editaci obrázků. Editaci je možno provádět při náhledu do binárního obrázku (druhá záložka skupiny) nebo do vstupního obrázku, do kterého je binární obrázek vykreslen (třetí záložka skupiny). V případě dokreslování hranic je nutné mít ve vstupním obrázku předlohu a poté uživatel může hranice přesně obkreslit.

Na panelu editace je v horní části umístěno tlačítko zpět, které panel zavře. pod ním jsou tlačítka lupy, která slouží pro zobrazení detailu, aby bylo možné editovat s větší přesností. Dále je na panelu umístěn náhled pro vybranou barvu, kterou je prováděno kreslení, respektive kterou jsou objekty přebarvovány. Barva se volí pomocí přepínačů v panelu barev, který je umístěn v pravé dolní části hlavního formuláře. Pod náhledem je umístěn přepínač, který informuje, zda je či není program v editačním. Tímto přepínač také slouží k vypnutí editačního módu.

Ještě níže se nacházejí přepínače jednotlivých editačních funkcí. Editačními funkcemi je kreslení bodů, úseček a lomených čar, mazání bodů, výběru objektů a barvy. Další funkcí je změna barvy objektů, respektive zařazení objektu do jiné třídy charakterizované příslušnou barvou. Editační funkce se aktivuje příslušným přepínačem a vypíná přepínačem, který je společný pro všechny funkce nebo levým tlačítkem myši. Zjednodušeně řečeno je tedy buď zvolen editační mód a zablokováno přepínání záložek a volba barvy nebo jsou tyto volby zpřístupněny a editace je vypnuta. První ze zmíněných funkcí, tedy kreslení bodů, je realizování pomocí volby bodu kurzorem a levého tlačítka myši. Kreslení úseček se provádí výběrem dvou krajních bodů, po jejichž volbě je úsečka bezprostředně vykreslena. Lomené čáry jsou konstruovány volbou zlomových bodů levým tlačítkem myši. Naproti tomu prostřední tlačítko zajistí přerušení čáry a je možno kreslit další. Funkce mazání bodů je inverzní ke kreslení bodů. Pomocí mazání výběru je vymazán obdélníkový výběr určený uživatelem. Mazání objektů slouží k vymazání kurzorem vybraného objektu. Mazání barvy vymaže všechny objekty, které jsou stejné barvy jako je barva zvolená. Třída objektů odpovídající dané barvě je tedy potom prázdná.

Funkce změna barvy objektu provede fakticky přeřazení objektu do třídy reprezentované vybranou barvou.

Zvláštní funkcí je funkce cyklického procházení objektů v obrázku. Tato funkce prakticky pomocí dvou tlačítek umožňuje procházení mezi jednotlivými objekty a jejich snadnější editaci. Funkce je blíže popsána v kapitole 7.5.

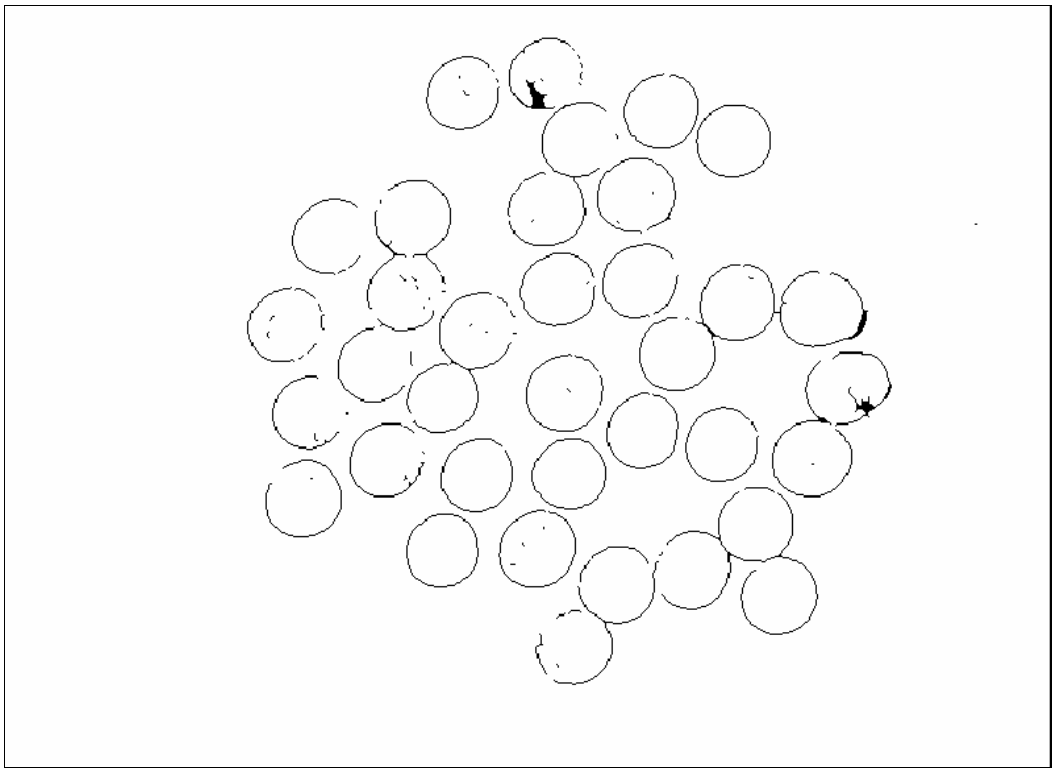
Pro možnost návratu poslední provedené akce je určeno tlačítko umístěné dole ve stejném panelu.

Obrázek 29: editor (ilustrační obrázek)

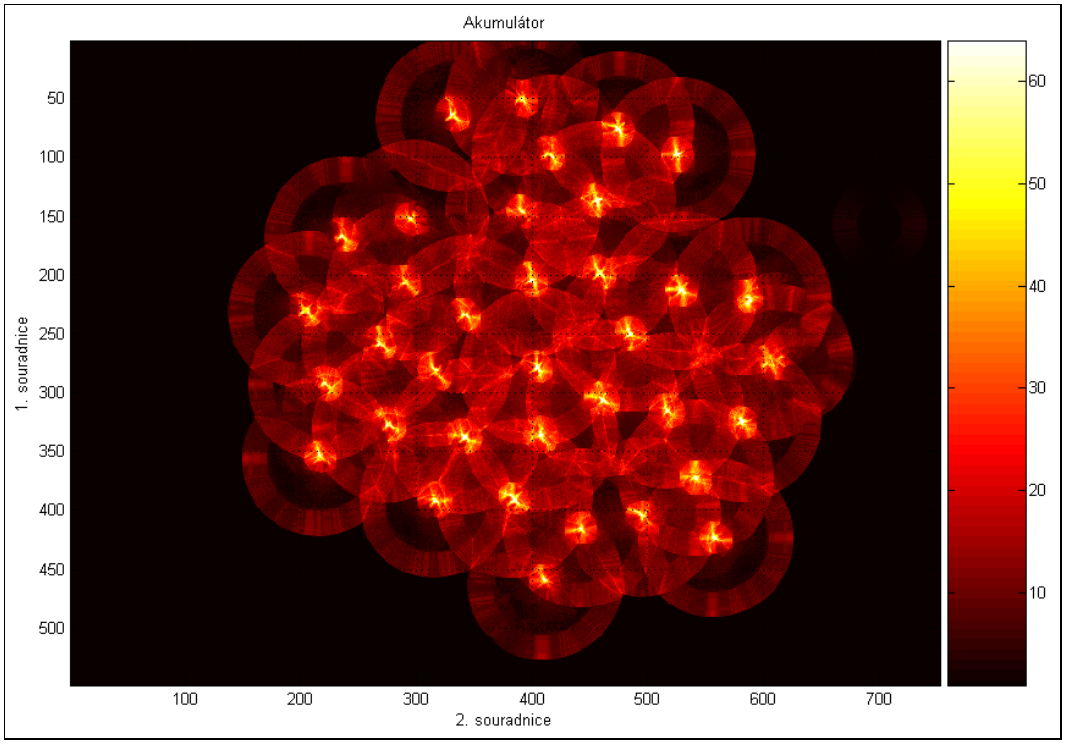
Příloha 2: Hledání kružnic

V této části je uveden příklad hledání kružnic pomocí Houghovy transformace. Kružnice v tomto příkladě aproximují průřezy vláken hedvábí.

Obrázek 30: RGB obrázek - řez hedvábím



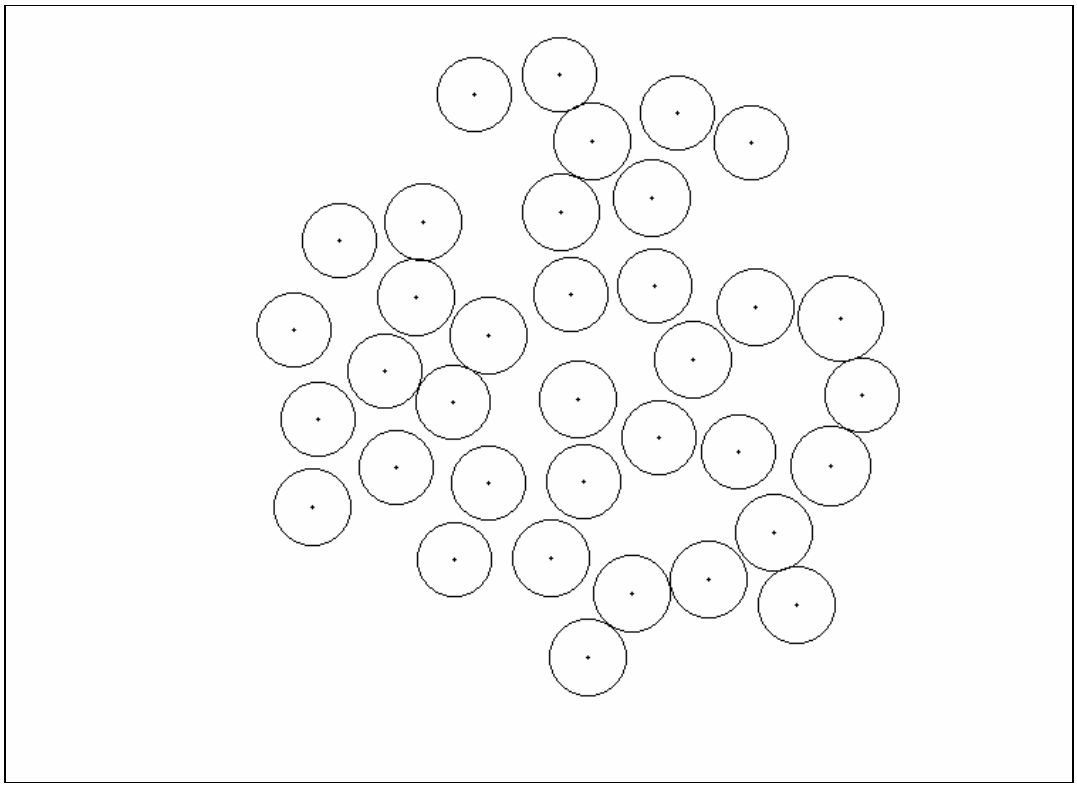
Obrázek 31: binární obrázek získaný úpravami z obrázku 29 (obrázek je invertován kvůli tisku)



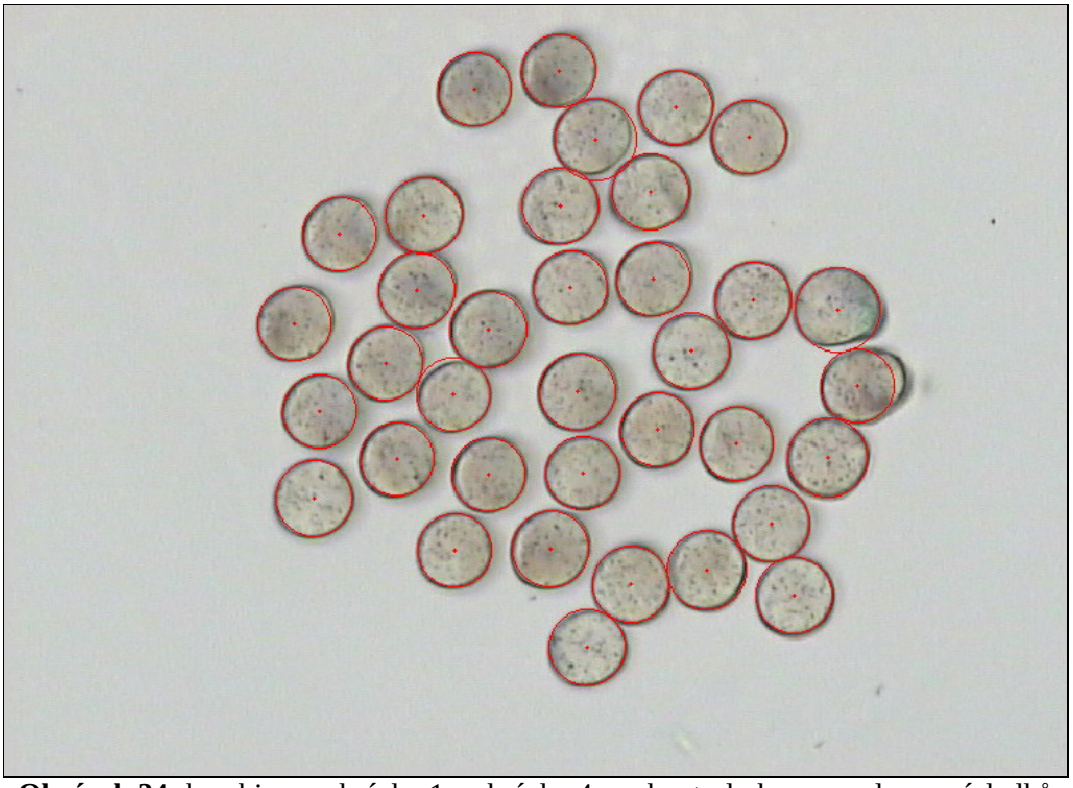
Obrázek 32: akumulátor Houghovy transformace odpovídající vstupnímu obrázku 2

datum a čas: 02-May-2005 17:41:25			
jméno zpracovávaného obrázku: hedvabi.bmp			
velikost obrázku: 548 x 752			
velikost akumulátoru: 548 x 752 x 11			
poloměr: 25 - 35, práh: 65 %, filtr: maska 3x3			
parametry kružnic: (a,b)... střed, r... poloměr, akum... hodnota akumulátoru			
a	b	r	akum
415	442	27	44
153	295	27	43
250	485	27	43
372	542	27	43
97	526	26	42
213	529	27	42
325	582	28	42
391	317	26	41
315	517	26	41
354	217	27	41
336	408	26	40
76	474	26	40
278	404	27	40
423	558	27	40
337	341	26	39
204	399	26	39
146	392	27	39
136	456	27	38
405	496	27	38
221	589	30	38
63	331	26	37
390	385	27	37
326	276	26	36
49	391	26	36
305	461	26	36
233	342	27	36
460	411	27	36
96	414	27	36
292	221	26	35
206	290	27	35
229	204	26	34
258	268	26	34
198	458	26	34
280	316	26	32
166	235	26	31
275	604	26	30

Tabulka 2: datový výstup z Houghovy transformace



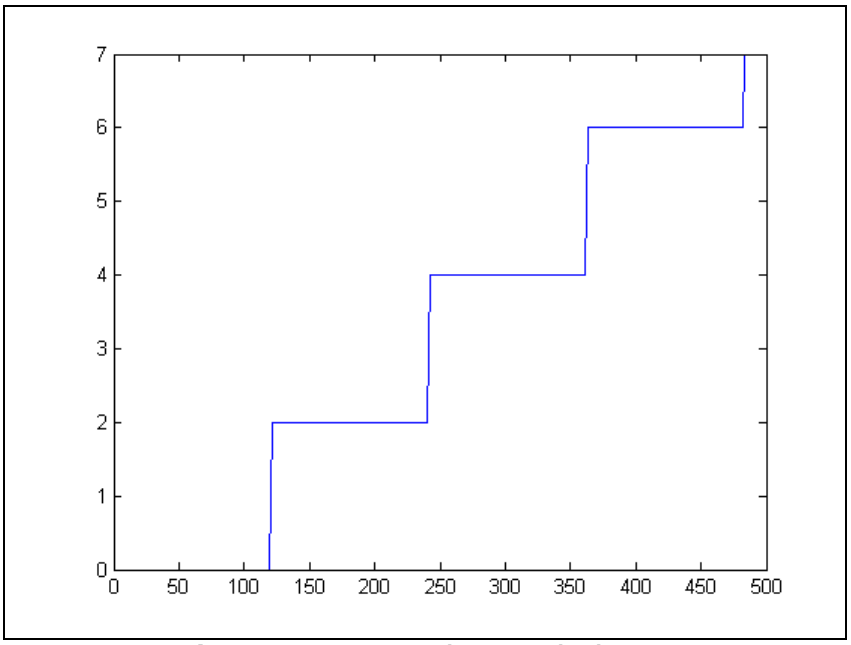
Obrázek 33: obrazový výstup Houghovy transformace (obrázek je invertován kvůli tisku)



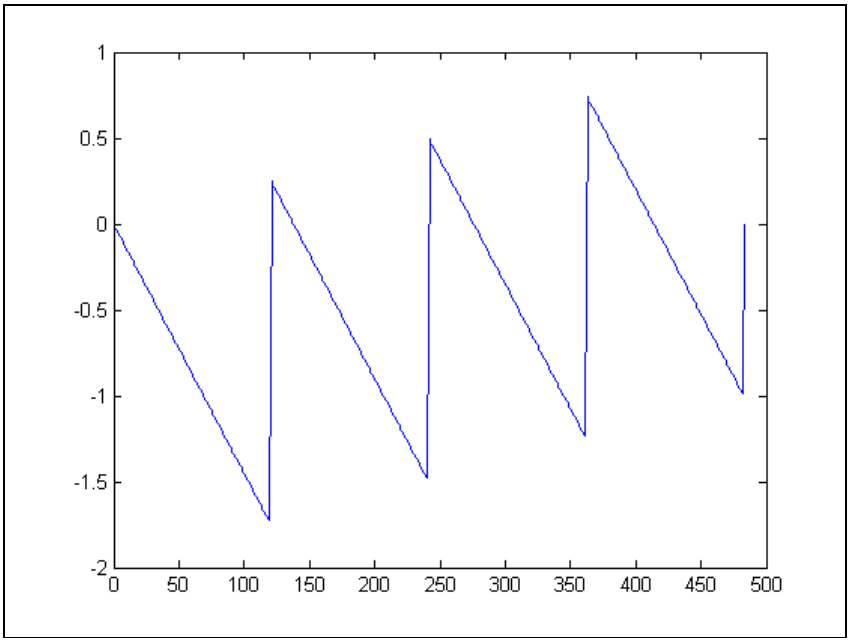
Obrázek 34: kombinace obrázku 1 a obrázku 4 pro kontrolu korespondence výsledků

Příloha 3: Řetězový kód a spektrum

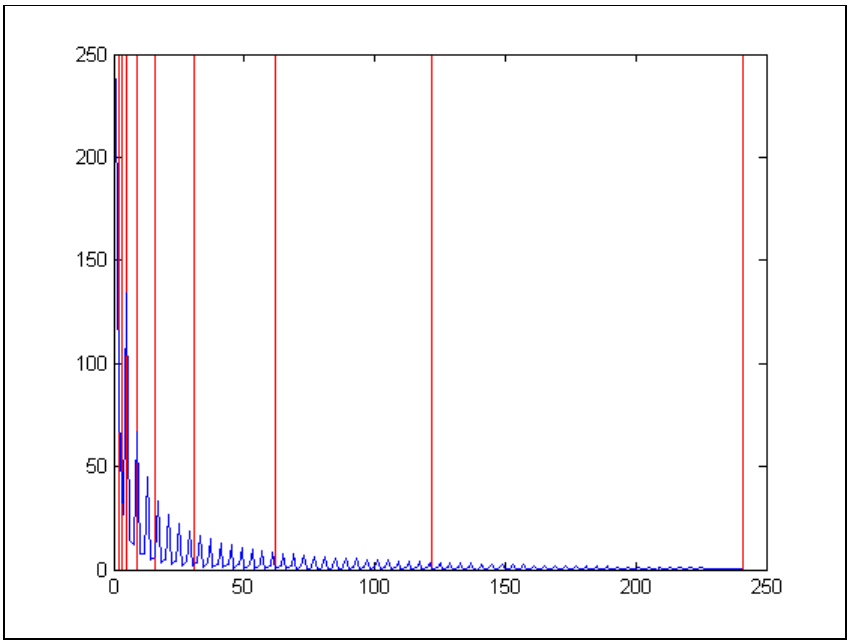
Následující dva příklady představují výstupy algoritmu pro získání spektrálních příznaků. Postupně jsou na obrázcích 35-40 ukázány kumulované řetězové kódy, upravené kumulované řetězové kódy a spektra pro čtverec a kružnici. Při určitých znalostech lze přibližně odhadnout vzhled objektu podle kumulovaného řetězového kódu nebo opačně.



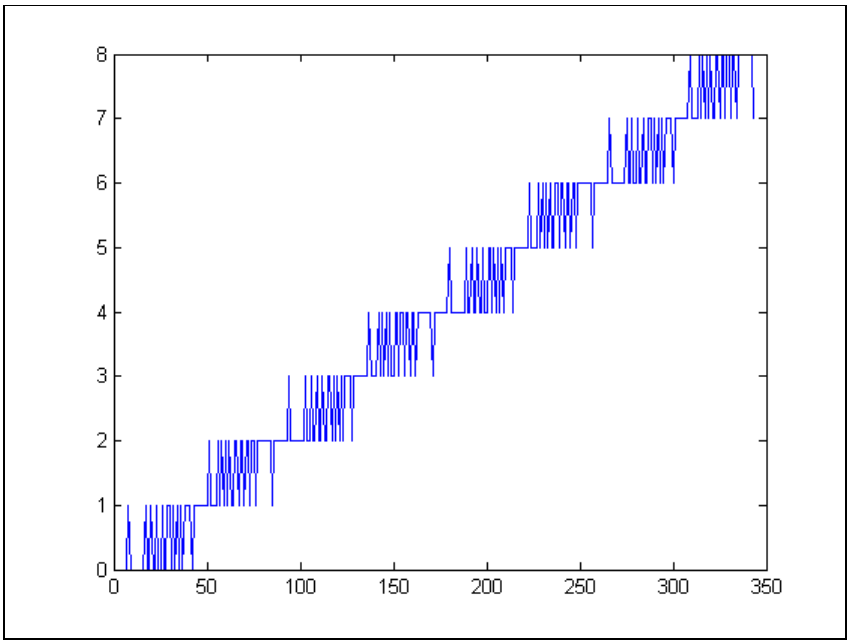
Obrázek 35: kumulovaný řetězový kód čtverce



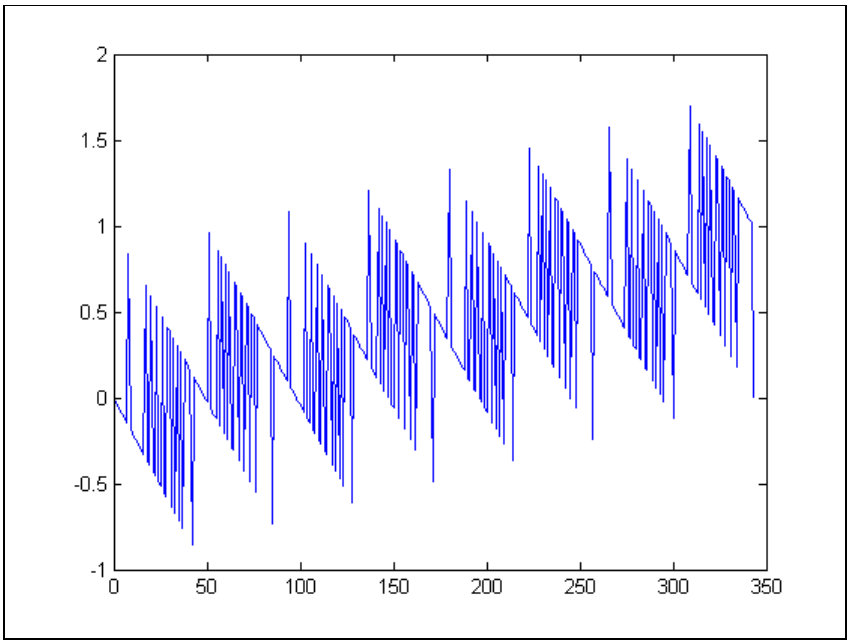
Obrázek 36: upravený kumulovaný řetězový kód čtverce



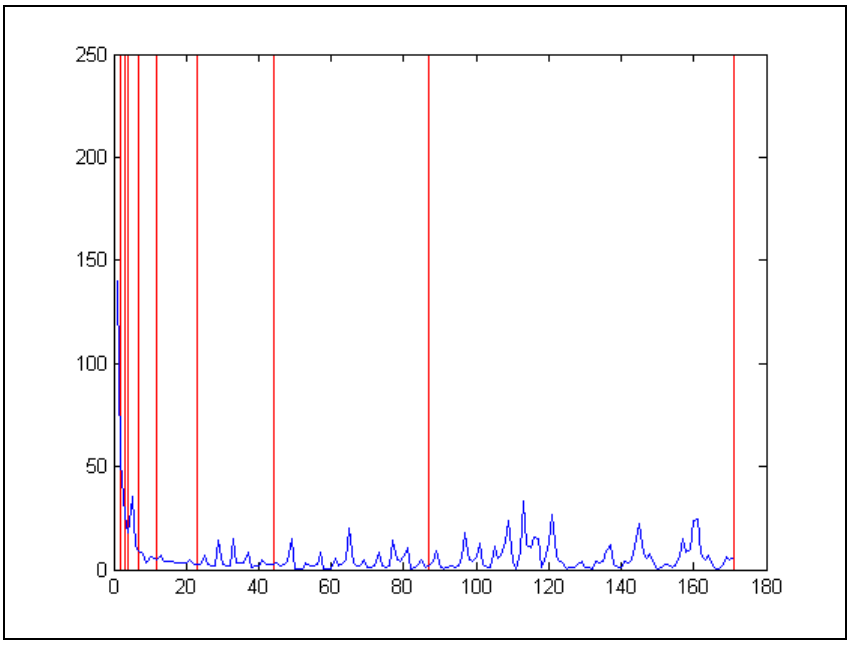
Obrázek 37: spektrum získané z upraveného kumulovaného kódu na obrázku 36 (je naznačeno logaritmické dělení pro tvorbu pásem ve kterých jsou hodnoty sčítané do spektrálních příznaků)



Obrázek 38: kumulovaný řetězový kód kružnice



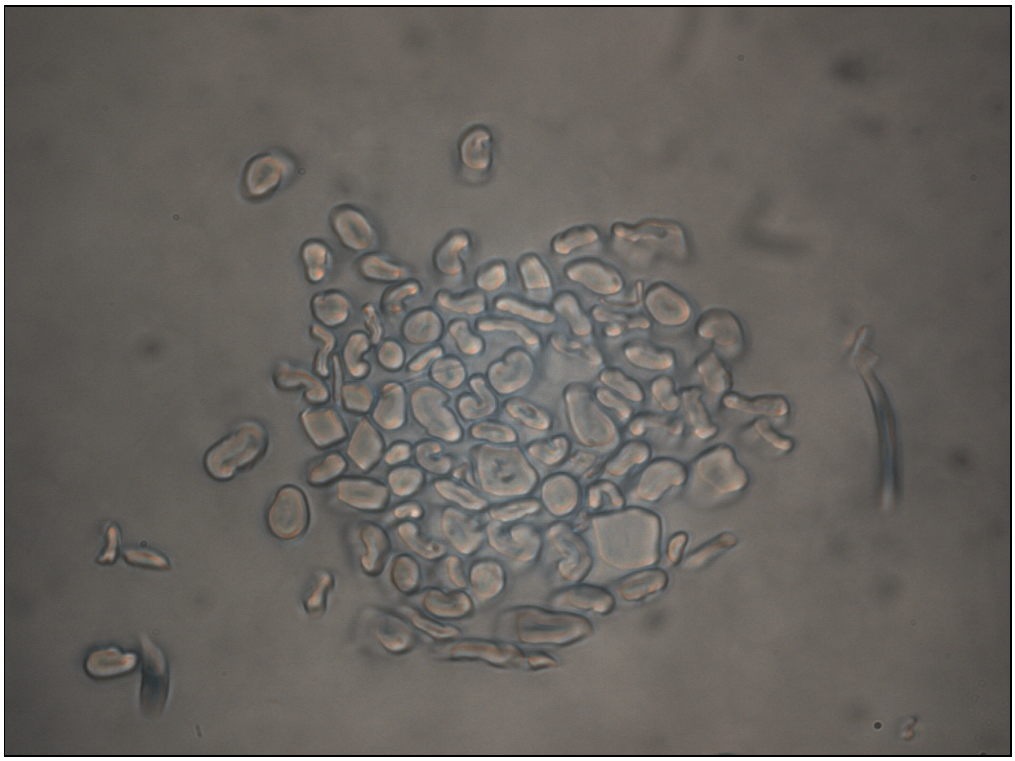
Obrázek 39: upravený kumulovaný řetězový kód kružnice



Obrázek 40: spektrum získané z upraveného kumulovaného kódu na obrázku 39 (je naznačeno logaritmické dělení pro tvorbu pásem ve kterých jsou hodnoty sčítané do spektrálních příznaků)

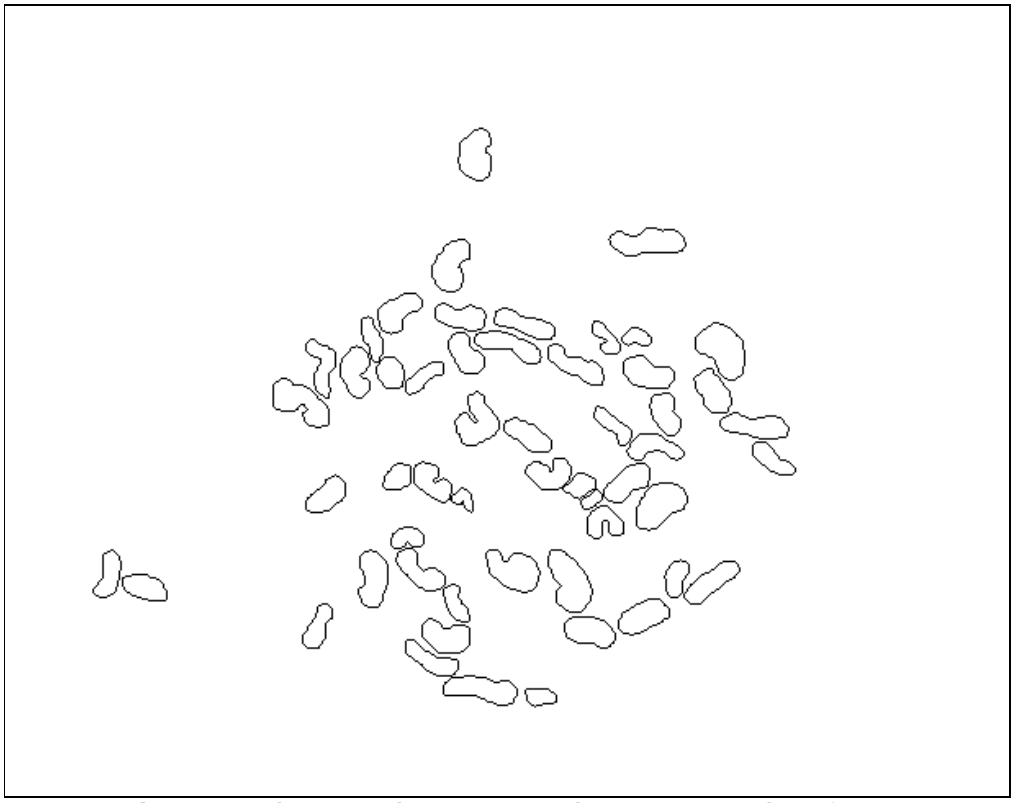
Příloha 4: Směsné příze

Ilustrační příklad řezu příze obsahující více typů vláken je na obrázcích 41-44. Na binárním obrázku 42 jsou vykreslena vlákna prvního typu, na obrázku 43 vlákna druhého typu. Obrázek 44 je již vytvořen vykreslením binárních obrázků 42 a 43 do vstupního RGB obrázku. Typy vláken jsou odlišeny zvolenými barvami jak je patrno z obrázku.

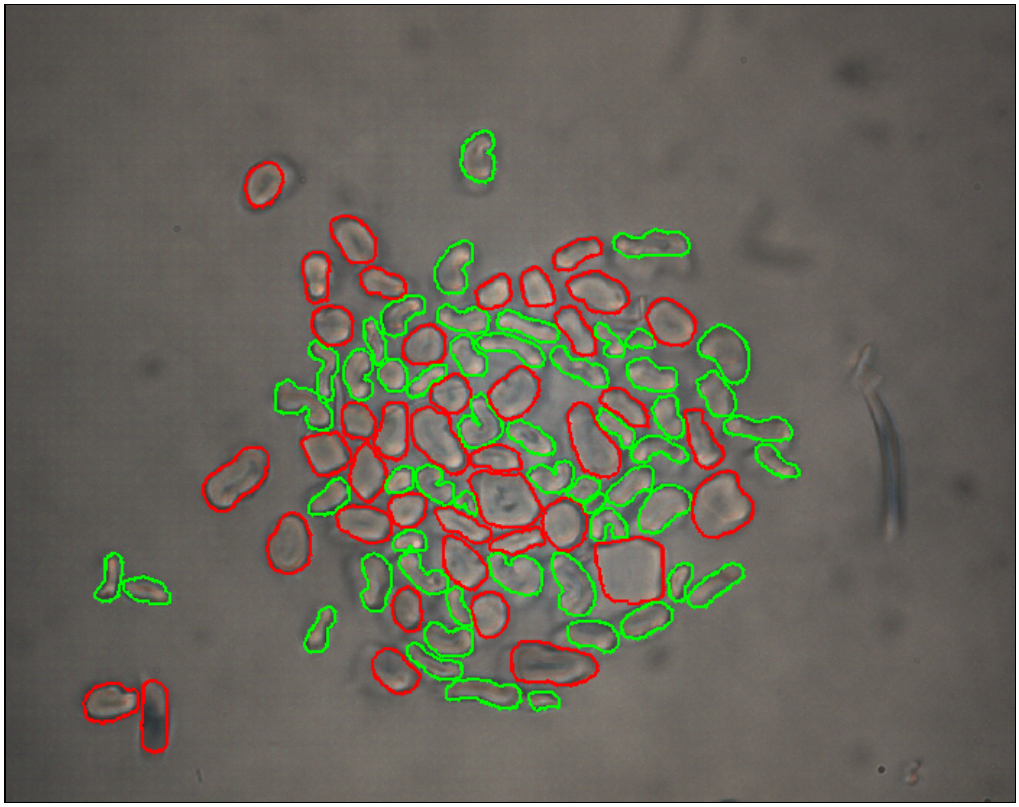


Obrázek 41: řez přízí s více typy vláken

Obrázek 42: vlákna prvního druhu (obrázek je invertován kvůli tisku)



Obrázek 43: vlákna druhého druhu (obrázek je invertován kvůli tisku)



Obrázek 44: obrázek vzniklý složením obrázku 41, 42 a 43 (každá barva odpovídá jednomu typu vlákna)