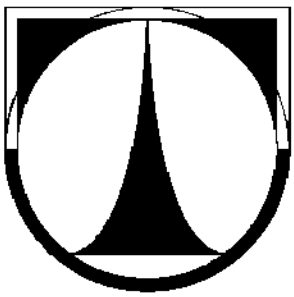
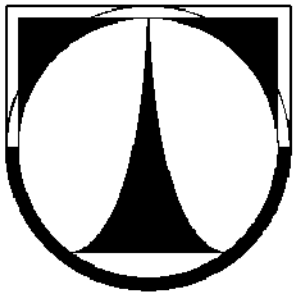


TECHNICKÁ UNIVERZITA
V LIBERCI

FAKULTA MECHATRONIKY A MEZIOBOROVÝCH
INŽENÝRSKÝCH STUDIÍ

DIPLOMOVÁ PRÁCE

2002 Ondřej Vacek



TECHNICKÁ UNIVERZITA V LIBERCI

FAKULTA MECHATRONIKY A MEZIOBOROVÝCH INŽENÝRSKÝCH STUDIÍ

Studijní program: 2612T – Elektotechnika a informatika

Katedra aplikované matematiky

Metoda klasifikačního stromu v analýze obrazové informace

The method of classification tree applicabled to analysis of visual information

Ondřej Vacek

Vedoucí DP: Doc. Petr Volf, Csc

Počet stran: 59
Počet obrázků: 27
Počet příloh: 04
Počet kódů programu: 20
Počet vzorců: 41

Tady bude zadání DP

Anotace

VACEK, Ondřej 2002 Ved. DP: Doc. Petr Volf, Csc

Metoda klasifikačního stromu s aplikací v analýze obrazové informace

Cílem diplomové práce je vytvořit algoritmus pro generování klasifikačního stromu a implementovat ho do vývojového prostředí MATLAB.

Práce se dále zabývá výpočty charakteristik prvního a druhého řádu. Tyto charakteristiky jsou vhodné pro popis texturních obrazů, uložených v digitální formě. Metoda klasifikačního stromu je pak následně aplikována na dvou příkladech s cílem klasifikace objektů (texturních obrazů) do správných tříd nebo vyhledávání poruch ve struktuře textury.

V přílohách jsou uvedeny výpisy vytvořených programů a vypočítané hodnoty z obou příkladů.

Abstract

VACEK, Ondřej 2002 Leading DP: Doc. Petr Volf, Csc

The method of classification tree applicabled to analysis of visual information

The aim of diploma work is formation of alogorithm for generation the classification tree and its implementation into the development environment "Matlab".

This diploma work occupies with calculations of characteristics the first and the second order. These characteristics are suitable for description of texture images, filed in the digital form. The method of classification tree is finally applicated to two examples with an aim of classification texture images into the right class or looking up defects in the structure of texture.

The notes of formed programmes and calculated data from both examples are presented in supplements.

Prohlášení

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

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce.

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

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

V Mladé Boleslavi 22. 5. 2002

Ondřej Vacek

Poděkování

Touto cestou bych chtěl vyslovit poděkování Doc. Petru Volfovi za odborné vedení při plnění úkolu diplomové práce a za půjčení odborné literatury. Dále bych chtěl poděkovat Mgr. Renatě Javůrkové za jazykovou úpravu diplomové práce.

Obsah

1. Úvod

S rozvojem výpočetní techniky se stále rozšiřují možnosti i v oblastech zpracování poměrně složité informace, jakou představují například obrazová data. Při praktické analýze obrazové informace je často potřeba řešit klasifikační problém, tj. rozpoznat, do jaké skupiny či třídy zkoumaný objekt patří, a to na základě měřitelných charakteristik tohoto objektu. Jednou z metod používanou na takovéto rozpoznávání je metoda klasifikačního stromu [1].

Cílem této práce bylo vytvořit a popsat algoritmus konstrukce klasifikačního stromu, implementovat ho do programového prostředí MATLAB a tuto proceduru využít k obrazové analýze textur, s cílem navrhnout počítačovou metodu rozpoznávání vzorku materiálu podle jeho digitalizovaného obrazu a následně ukázat i možnost detekce defektů materiálu. Programové prostředí MATLAB je stručně popsáno v kapitole 2.

Metoda klasifikačního stromu a vytvořené algoritmy jsou popsány v kapitole 3. Konstrukce stromu je prováděna na trénovací množině charakteristik objektů, u kterých známe zařazení do tříd. Vlastní klasifikace je pak založena na porovnávání číselných charakteristik klasifikovaných objektů s hodnotami uloženými ve struktuře stromu.

Dále se tato práce v kapitole 4 zabývá výpočty číselných charakteristik uvedených v [4], vhodných právě pro popis texturních obrazů.

V kapitole 5 jsou vytvořené algoritmy aplikovány na rozpoznávání textur pomocí klasifikačního stromu a na základě vytipovaných charakteristik. Jsou zpracovány dva příklady, první z nich je zaměřen na klasifikaci typu textury, na druhém příkladu je pak znázorněna možnost využití klasifikačního stromu pro klasifikaci poruch struktury textilního materiálu. Obrazové vzorky textur byly získány z tzv. Brodatzova alba textur [11] nebo byly využity snímky textilních materiálů získané z FT TUL [12].

V přílohách jsou výpisy všech vytvořených programů a tabulky vypočítaných hodnot k oběma příkladům. Součástí diplomové práce je i CD s vytvořenými programy.

2. Programové prostředí MATLAB

MATLAB (produkt firmy MathWorks) je integrované programové prostředí pro vědeckotechnické výpočty, modelování, simulace, analýzu a prezentaci dat, měření a zpracování signálů. MATLAB byl implementován na všech významných platformách, od osobních počítačů s operačními systémemy MS-Windows, Linux až po UNIXové pracovní stanice. Pro svoji otevřenost, jednoduchou syntaxi a kvalitní implementované algoritmy se stal velice oblíbeným nástrojem univerzitních laboratoří po celém světě.

Otevřená architektura MATLABu umožňuje vznik knihoven funkcí, nazývaných *toolboxy*, které rozšiřují použití programu v příslušných vědních a technických oborech. Proto byl MATLAB také využit k vytvoření několika algoritmů pro potřeby této práce.

Jedná se o klasifikační algoritmy na bázi klasifikačního stromu. Tyto algoritmy tvoří důležitou součást praktických (aplikačních) výsledků této práce a jsou podrobně popsány v kapitole 3. Pro tvorbu algoritmu klasifikačního stromu byly využity informace z literatury [2], [3] a zkušenosti s algoritmem implementovaným v programovém prostředí *S-Plus* (programový balík pro statistické výpočty).

Dále byl MATLAB použit k naprogramování jednoduchých funkcí pro výpočet texturních charakteristik, popsaných v kapitole 4. V těchto vytvořených programech byly použity některé funkce z toolboxu s názvem *Image Processing Toolbox* [10].

Jedním z důvodů pro práci na algoritmu klasifikačního stromu byla i skutečnost, že tato metoda je sice součástí některých statistických a analytických programových produktů (*S-Plus, XploRe,..*), ale zdrojový kód není zpravidla k dispozici a není také snadné v nich algoritmy modifikovat a přizpůsobovat řešenému problému. S ohledem na tento požadavek bylo zvoleno právě programové prostředí MATLAB.

3. Rozpoznávání a klasifikace

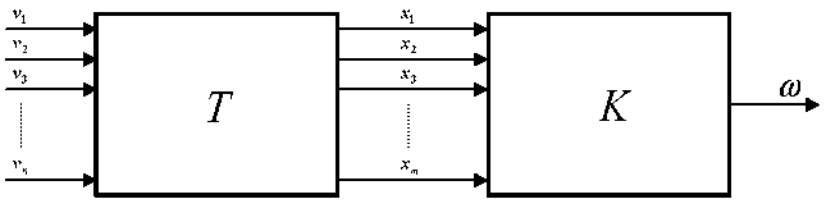
Nejprve bychom měli připomenout význam samotného pojmu rozpoznávání. Rozpoznávání [5] obvykle chápeme jako úlohu, při které objekty zařazujeme do tříd podle jejich společných vlastností tak, že objekty vzájemně si podobné zařazujeme do stejné třídy.

Rozlišujeme několik typů úloh rozpoznávání, v závislosti na tom, jak je problém zadán:

klasifikace - zařazujeme do předem známého, pevného počtu tříd. O těchto třídách máme většinou nějakou apriorní informaci (např.: ve formě trénovacích dat). Do této kategorie můžeme např. zařadit rozpoznávání znaků.

rozpoznávání - počet tříd není předem znám, výsledné třídy vznikají až v důsledku výskytu více či méně oddělených skupin v analyzovaných situacích. Do této kategorie patří např. rozpoznávání plynulé řeči.

Principiálně můžeme obecnou klasifikační úlohu znázornit tímto blokovým schématem:



Obr. 1

Nejprve podrobíme vstupní objekty v transformaci T, při které získáme charakteristické příznaky objektů x. Klasifikátor K pak na základě definovaného rozhodovacího pravidla zařazuje obrazy do klasifikačních tříd. Výstupem klasifikátoru je identifikátor třídy ω.

Nastavení klasifikátoru se provádí trénováním neboli učením. Rozlišujeme učení s učitelem (řízená klasifikace - „*supervised*"), kdy klasifikátoru předkládáme obrazy, u nichž známe jejich příslušnost k třídě, a učení bez učitele (neřízená klasifikace - „*unsupervised*"), kdy správné zařazení do klasifikačních tříd neznáme.

3.1 Základní rozdělení metod rozpoznávání

Podle toho, zda je rozhodovací pravidlo postaveno na porovnávání měřitelných veličin nebo strukturních vlastností objektů, dělíme metody na:

Příznakové - zde se klasifikátor obvykle zaměřuje na měřitelné (fyzikální) veličiny objektu.

Strukturální - jsou postaveny na identifikaci elementů objektu a analýze vztahů (relací) mezi nimi.

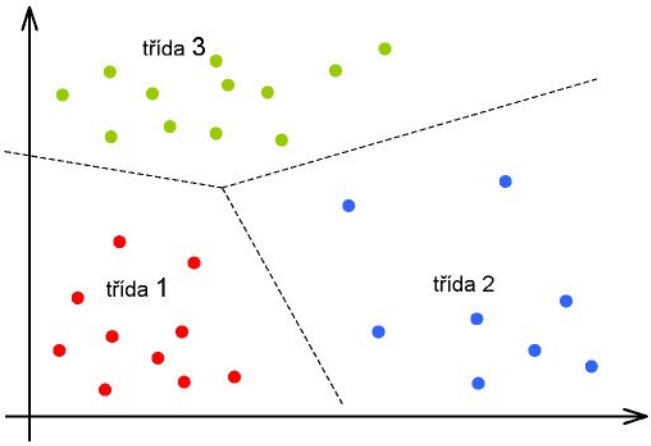
Hybridní - jsou kombinací obou předchozích přístupů

3.2 Příznakové metody

Základem příznakových metod je práce s měřitelnými fyzikálními veličinami - příznaky objektu. Příznaky je nutné volit tak, aby měly co nejvyšší informativní hodnotu a bylo jich co nejméně. Více příznaků znamená lepší výsledky klasifikátoru, ale také delší dobu vyhodnocení. Existuje ovšem hranice, kdy další zvyšování počtu příznaků už kvalitu klasifikace nezlepšuje.

Naopak zvýšení počtu příznaků může vést k nespolehlivosti klasifikace v důsledku zvýšení důrazu na individuální příznaky, než na typové. Proto je častou metodou ověřování optimality klasifikačních pravidel získaných z trénovací fáze jejich testování na testovacích datech, o kterých také víme, do jaké třídy spadají. Pokud je výsledek podstatně horší než na trénovacích datech, znamená to, že rozhodovací pravidla jsou příliš podrobná a nastavená právě jen na učící data (a zřejmě i počet příznaků použitých pro klasifikaci je zbytečně velký). Obvykle jde o nalezení kompromisu mezi bohatostí modelu a vysvětlovací schopností (flexibilitou) modelu.

Pokud si vektory příznaků představíme jako body v obecně n-rozměrném prostoru, pak tento prostor nazýváme obrazovým prostorem. Rozkladem obrazového prostoru vzniknou podprostory, které odpovídají klasifikačním třídám. Na obrázku 2 na následující straně je uveden příklad pro dvourozměrný prostor, kde barevné body odpovídají příznakovým vektorům.



Obr. 2

Podobnost objektů v rámci jedné třídy je vyjádřena blízkostí jejich obrazů v obrazovém prostoru. Obrazy jedné třídy vytváří shluky, které od sebe můžeme oddělit rozdělující nadplochou. V případě uvedeného obrázku je nadplochou trojice polopřímek, které rozdělují prostor na tři oblasti.

V realných případech je však situace zpravidla mnohem komplikovanější, shluky bodů se mohou prolínat. Je to důsledek přirozené variability objektů, resp. jejich příznaků, v každé ze tříd. Tuto variabilitu lze popsat například náhodnou složkou a celý shluk příslušející k jedné třídě pak sdruženým (n-rozměrným) rozdělením pravděpodobnosti. Záleží pak na klasifikátoru, a především na výběru příznaků, aby bylo možné klasifikační proceduru optimalizovat, tj. dosáhnout co nejmenší míry misklasifikace (ať už měřené na trénovacích datech, či např. odvozené ze zmíněného pravděpodobnostního popisu).

3.3 Strukturální metody

Strukturální metody jsou založeny na popisu objektu pomocí elementárních vlastností (tzv. primitiv) vyjadřujících strukturu objektu a relací mezi nimi. Pokud jsou zvolena primitiva tak, že odpovídají podstatným částem objektů, a pokud relace vyjadřují důležité vztahy mezi nimi, získaný popis dobře vystihuje strukturální vlastnosti objektu.

Základní myšlenka metod strukturálního rozpoznávání vychází z předpokladu, že obrazy objektů každé třídy jsou si strukturálně podobné a tvoří jazyk třídy objektů. Úloha se tak převádí na vyhodnocení podobnosti dvou jazyků. K tomu využíváme propracovaný

aparát metod syntaktické analýzy z teorie formálních jazyků. Na jednoduchém příkladu (viz. obrázek 3) je znázorněna práce s jazyky a gramatikami.

Obr. 3

Primitiva pro popis obrázků stromů budou: trojúhelník, obdélník, kruh a relace | bude znamenat "je pod". Můžeme tedy vytvořit gramatiky pro oba typy stromů:

G1 (Jedle):	G2 (Dub):
(1) S >>> Kmen | Koruna (2) Kmen >>> obdélník (3) Koruna >>> trojúhelník-Koruna | trojúhelník	(1) S >>> Kmen | Koruna (2) Kmen >>> obdélník (3) Koruna >>> kruh

Tab. 1

Z těchto gramatik, které jsme vytvořili, můžeme zpětně vytvořit oba původní stromy nebo i jejich nové varianty. Můžeme vytvářet jedle různých velikostí koruny tak, že je skládáme z jednotlivých pater. Měnit velikost koruny dubu ovšem není tak snadné, neboť dub má korunu pouze jedné velikosti. Jediné, co můžeme udělat, je mít v gramatice zásobu kruhů různých velikostí. Tímto způsobem se nám ale zvyšuje počet primitiv, což není žádoucí.

Tyto protichůdné požadavky vyřešíme použitím tzv. atributových gramatik, které pracují nejen s primitivy, ale i s jejich vlastnostmi. Tak můžeme např. každému kruhu přiřadit jeho poloměr, obdélníku šířku a výšku apod.

Gramatiku obvykle navrhuje konstruktér klasifikátoru. Dosud však neexistuje obecná metoda, jak ze slov jazyka odvodit gramatiku, která by jazyk vhodně aproximovala (musíme totiž předpokládat, že trénovací množina není úplná).

Při rozpoznávání stojíme před rozhodnutím, zda rozpoznávané slovo patří do jazyka třídy či nikoliv, tzn. zda je generováno gramatikou třídy. K tomu lze použít syntaktickou analýzu, kterou se podrobně zabývá teorie formálních jazyků.

3.4 Hybridní metody

V některých aplikacích nevystačíme s čistě příznakovým nebo čistě strukturálním rozpoznáváním, ale používáme oba přístupy v kombinaci. Tento přístup se nazývá hybridní. Rozpoznávání pak obvykle probíhá v následujících fázích:

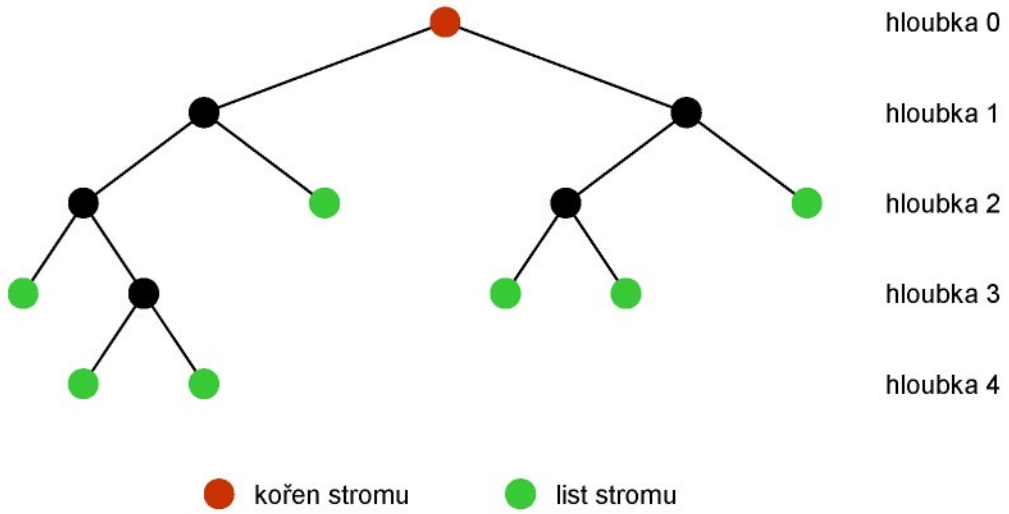
1. příznakovými metodami identifikujeme primitiva pro vytvoření symbolického popisu objektu
2. obraz objektu podrobíme syntaktické analýze s cílem zařadit ho do některé ze tříd
3. výsledný popis objektu je prověřen z hlediska významu a je doplněn o sémantickou (číselnou) informaci

3.5 Metoda klasifikačního stromu

Jak už bylo ukázáno na příkladu (viz. obrázek 2, str. 13), na klasifikaci se můžeme dívat též jako na proceduru rozkládající prostor Φ vysvětlujících proměných (příznaků, charakteristik) X na M navzájem disjunktních množin $A_1,.....,A_M$ takových, že jakmile $x \in A_k$, pak příslušný objekt zařadíme do k-té třídy. K takovému rozkladu můžeme dojít i postupným štěpením celého prostoru Φ v mnoha krocích, přičemž v každém kroku pouze rozdělíme určitou vybranou množinu (z těch, které jsou produktem předchozích kroků) na dvě podmnožiny [3].

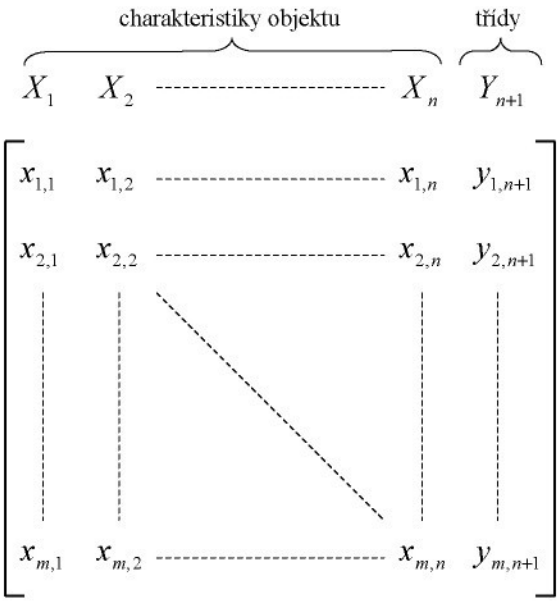
Tento rozklad může být proveden metodou binárního klasifikačního stromu. Podle předchozího rozdělení můžeme tuto metodu zařadit mezi metody příznakové. Klasifikace objektů u této metody je založena na porovnávání jejich příznaků (charakteristik) s hodnotami uloženými v uzlech binárního stromu. Binárním stromem rozumíme hierarchickou datovou strukturu, skládající se z uzlů, které uchovávají informace, a větví, které uzly spojují. Každý z uzlů buď nemá žádného následníka (pak se nazývá list), nebo má právě dva následníky (další uzly stromu). Můžeme též zavést pojem hloubka uzlu,

který udává vzdálenost uzlu od kořene stromu. Na obrázku 4 je znázorněn příklad struktury binárního stromu.



Obr. 4

Pro klasifikování objektů je nutné nejdříve klasifikační strom vytvořit. Strom konstruujeme na základě trénovací množiny objektů, u kterých známe zařazení do tříd. Trénovací data jsou obvykle udávána ve formě matice (viz. obrázek 5), kde řádky tvoří hodnoty charakteristik objektů X_n, poslední prvek řádku označuje zařazení do třídy (obvykle celé číslo).



Obr. 5

Protože pravidla dělení (tj. algoritmu) klasifikačního stromu se tvoří při analýze konkrétních trénovacích dat, dále uvedená kritéria pro tvorbu stromu mají empirický charakter (tj. jejich splnění sledujeme na konkrétních datech).

Pro sestrojení algoritmu klasifikačního stromu je nutné vzít v úvahu tyto aspekty:

1. V každém kroku uvažujeme dělení stromu typu: $x_j < a$ proti $x_j \geq a$ (je-li vstupní veličina X_j numerická) nebo $x_j = A, B, \ldots$ (výčet indikátorů – označení kategorií) proti $x_j = C, D, \ldots$, je-li veličina X_j kategoriální.

2. Vybrat vhodné kritérium měřící kvalitu separace tříd od sebe. Toto kritérium nazýváme *deviance* (zde má tento výraz význam „neurčitost" spíš než „odchylka"). Účelem je minimalizace této deviance. Obvykle se používají dvě různá kritéria, obě měří nehomogennost rozdělení pravděpodobnosti jednotlivých tříd pro objekty v koncových uzlech. Kritérium, které používá i výpočetní nástroj *S-Plus*, má tvar:

$$D_i = -2 \cdot \sum_{k=1}^{M} n_{ik} \cdot \log p_{ik} , \qquad (1)$$

kde M je celkový počet tříd, n_{ik} je počet objektů v i-tém koncovém uzlu, spadající do k-té třídy a p_{ik} lze odhadnout jako $\frac{n_{ik}}{n_i}$, n_i je celkový počet objektů v listu i.

Hodnota deviance je maximální pro rovnoměrné (tj. z našeho hlediska nejméně rozlišující) rozdělení tříd po jednotlivých listech. Toto kritérium přímo souvisí s entropií jako mírou neurčitosti.

Dalším kritériem je tzv. Ginniho index, který je pro i-tý list definován takto:

$$D_{Gi} = 1 - \sum_{k=1}^{M} p_{ik}^2 \qquad (2)$$

Maxima opět dosahuje pro rovnoměrná p_{ik}.

3. Stanovit kritérium ukončení dělení stromu. Můžeme použít několik kritérií a ta případně kombinovat (tj. alespoň jedno z nich by mělo být splněno). Kritériem může být například minimální změna deviance, při níž se ještě vyplatí další dělení, maximální celková misklasifikace (misklasifikace - poměrné vyjádření počtu špatně klasifikovaných objektů) nebo dosažení maximální hloubky stromu.

Dělení je také možné ukončit, když je strom již dostatečně podrobný, tj. jeho list, který uvažujeme pro další dělení, již obsahuje zvolený minimální počet objektů.

3.6 Algoritmus klasifikačního stromu

Pro potřeby naší analýzy byl vytvořen algoritmus implementovaný jako funkce s názvem *genstrom.m* v prostředí MATLAB [10]. Argumentem této funkce je matice trénovacích dat (struktura matice vychází z obrázku 5, str. 16), hodnota maximální misklasifikace pro ukončení dělení stromu a parametr maximální hloubky stromu. Výstupem je pak třírozměrná matice, ve které jsou uložena data struktury vytvořeného stromu.

Pro klasifikaci objektů je vytvořena další funkce s názvem *klasifikuj.m*, která na základě vygenerovaného stromu (prvním argumentem je výše zmíněná třírozměrná matice) klasifikuje objekty. Charakteristiky těchto objektů jsou stejně jako trénovací data uloženy v matici se stejným pořadím prvků, pouze chybí poslední sloupec s čísly tříd (tato matice je pak druhým argumentem této funkce). Funkce vrací matici, která je doplněnou vstupní maticí právě o poslední sloupec s čísly tříd, do kterých byly objekty podle stromu zaklasifikovány.

V následujících kapitolách jsou programy obou funkcí podrobně popsány.

3.6.1 Popis programu funkce *genstrom.m*

Funkce pro generování stromu je definovaná takto:

(kód 1)

```
function [Strom_vys] = genstrom(X,maxm,maxh),
```

do proměnné `X` vstupuje matice trénovacích dat, do `maxm` se ukládá hodnota maximální misklasifikace a do `maxh` se ukládá maximální hloubka stromu. Hodnota `maxm` nám určuje, kdy se větvení stromu zastaví (zadává se v intervalu $\langle 0, 0.5)$. Pokud zadáme 0, dělení stromu se ukončí tehdy, je-li strom vytvořen tak, že jsou všechny trénovací objekty klasifikovány do správných tříd, nebo při dosažení hloubky stromu omezené hodnotou v `maxh`.

Po definici funkce následuje příprava a inicializace proměnných.

(kód 2)

```
format short g;                %vystupni format cisel
```

```
Strom_klas=0;                    %hodnota klasifikaci stromu
Pracovni=X;                      %pracovni matice, v cyklu meni obsah
[rd_X,sl_X]=size(Pracovni);      %rozmer matice X, radky,sloupce
poc_trid=max(Pracovni(:,sl_X)); %pocet klasifikovanych trid
konec=0;
```

Vstupní matice `X` se uloží do matice `Pracovni`, se kterou se pracuje v celém programu. Dále je zjištěna velikost vstupní matice (proměnné `rd_X` a `sl_X`) a počet tříd `poc_trid` z posledního sloupce vstupní matice (uvažuje se, že třídy jsou označeny vzestupně celými čísly od *1* do M).

Dále následuje hlavní cyklus pro generování stromu.

(kód 3)

```
%Hlavni cyklus, maximalni pocet radku stromu je ulozen v promenne maxh
for radek_strom=1:maxh
   for sloupec_strom=1:2^(radek_strom-1)
```

Proměnná `radek_strom` nahrazuje veličinu hloubka uzlu, definovanou v kapitole 3.5 (str. 16), pouze je počítána od 1. Počet řádků (hloubka stromu) je omezena hodnotou ve vstupním argumentu `maxh`. Název proměnné `radek_strom` vychází z toho, že struktura stromu je uložena v matici. Podobně je tomu tak i u proměnné `sloupec_strom`. Oba cykly slouží k procházení stromu pro všechna možná dělení.

Následující podmínka rozhoduje, zda v daném uzlu stromu dojde k dalšímu dělení, nebo se stane uzel listem. Tato podmínka se uplatní, až když jsou již některá data klasifikována. V matici `Strom_klas` jsou uloženy hodnoty klasifikace ($klasifikace = 1 - misklasifikace$), pro listy je hodnota rovna 1.

(kód 4)

```
if Strom_klas(radek_strom,sloupec_strom)<1
   for sloupec=1:sl_X-1
      %vzestupne serazeni prac. matice podle prom. 'sloupec'
      Pracovni=sortrows(Pracovni,sloupec);
```

Cyklus proměnné `sloupec` je hlavním cyklem pro určení dělení stromu podle charakteristiky objektů, která se nachází v matici `Pracovni` právě ve sloupci `sloupec`. Dále se matice `Pracovni` vzestupně seřadí pomocí funkce *sortrows* podle sloupce `sloupec`.

Podmínka `radek_strom>1` zjišťuje, zda nejde o první dělení stromu (dělení v kořenu stromu). V případě, že je podmínka splněna, začne se generovat dvouřádková

matice `Cesta_zpet`, která je důležitá pro určení dat, které budou vstupovat do následujícího dělení.

(kód 5)

```
if radek_strom>1
   %1. radek - delici radky daneho sloupce
   %2. radek - sudy (=1) nebo lichy (=0) sloupec pro dany delici radek
   Cesta_zpet=[0 rd_X;1 0];
```

Do této matice jsou postupně zaznamenávány hodnoty dělících řádků rodičovských uzlů a informace o tom, zda uzel je vůči rodičovskému uzlu sudý (vpravo) nebo lichý (vlevo). Dělicí řádek je číslo řádku matice `Pracovni`, ve které došlo k dělení stromu podle dané charakteristiky. Příklad dělení je znázorněn na následujícím obrázku.

$$\begin{matrix} X_1 & X_2 & Y \end{matrix}$$
$$\begin{bmatrix} 0.32 & 2.56 & 1 \\ 0.45 & 1.8 & 3 \\ 0.5 & 2.32 & 2 \\ 0.54 & 2.41 & 2 \end{bmatrix}$$

Obr. 6

Pracovní matice příkladu obsahuje dvě charakteristiky X_1 a X_2, sloupec Y vyjadřuje zařazení do tříd. Uvažujme první dělení, tedy u kořene stromu. Algoritmus dělení (viz. str. 22) najde nejvýhodnější dělení stromu podle charakteristiky X_1. Matice je tedy seřazena vzestupně podle prvního sloupce a nalezne se optimální dělící hodnota. (viz. kód 9, str. 23) Hodnota dělícího řádku je pak pro tento příklad rovna 2 (dělení mezi druhým a třetím řádkem matice). Data matice nad červenou čárkovanou čarou jsou pro lichou větev stromu (levá větev od rodičovského uzlu) a data pod zelenou čárkovanou čarou jsou pro lichou větev stromu (pravá větev od rodičovského uzlu).

Hodnoty dělících řádků se ukládají do matice `Strom_rd`, čísla sloupců (charakteristik), ve kterých bylo provedeno dělení, se ukládají do matice `Strom_sl`. Následující cyklus prochází strom od daného uzlu ke kořeni stromu a ukládá postupně hodnoty dělících řádků do matice `Cesta_zpet`.

(kód 6)

```
for a=1:radek_strom-1
   rd=radek_strom-a;
   a2=2^a;
   sl=ceil(sloupec_strom/a2);
   sl_nasled=ceil(2*sloupec_strom/a2);
```

```
      if sloupec==Strom_sl(rd,sl)
         Cesta_zpet=cat(2,Cesta_zpet,[Strom_rd(rd,sl);
                                      sl_nasled/2==ceil(sl_nasled/2)]);
      end
   end
```

Podmínka v cyklu vybírá pouze dělící řádky, které byly provedeny jen pro danou charakteristiku (proměnná `sloupec`). Pomocí funkce *cat* se v každém cyklu k matici `Cesta_zpet` přidávají nové sloupce.

Z matice `Cesta_zpet` se vypočítávají dva řádkové vektory `Zacatek` a `Konec`. Jednotlivé hodnoty obou vektorů se stejným argumentem udávají počáteční a konečný index řádků matice `Pracovni` pro výběr dat. V případě, že jde o dělení stromu v kořeni (viz. nesplnění podmínky: `if radek_strom>1`, kód 5, str. 20), jsou pak vybrána všechna data z daného sloupce matice (`Zacatek(sloupec)=1` a `Konec(sloupec)=rd_X`).

(kód 7)

```
      Cesta_zpet(1,:)=Cesta_zpet(1,:)+Cesta_zpet(2,:);

      Zacatek(sloupec)=max(Cesta_zpet(1,:).*Cesta_zpet(2,:));
      Konec(sloupec)=min(not(Cesta_zpet(2,:)).*Cesta_zpet(1,:)+
                                              2*rd_X*Cesta_zpet(2,:));
   else % viz. podminka - if radek_strom>1
      %Tento pripad nastava pri deleni korene stromu
      Zacatek(sloupec)=1;
      Konec(sloupec)=rd_X;
   end  % viz. podminka - if radek_strom>1
```

V následující části programu se generuje matice `G`, podle které se rozhoduje dělení stromu pro daný uzel. Hodnoty matice `G` jsou vypočítávány pomocí dělícího kritéria 1 (uvedeného v kapitole 3.5, str. 17) pro všechny sloupce (charakteristiky) matice `Pracovni` a všechna možná dělení. Nejprve je aktuální sloupec matice `G` naplněn „velkými čísly“ (hledá se minimum a rozměr matice může být větší než počet zpracovávaných dat pro danou charakteristiku). Proměnná `i` v prvním cyklu je dělícím řádkem pro aktuální sloupec matice `Pracovni`. Hodnota `i` se mění v rozmezí vybraných řádků matice `Pracovni` dané indexy `Zacatek(sloupec)` a `Konec(sloupec)-1`.

(kód 8)

```
      G(1:rd_X-1,sloupec)=1e5;
      for i=Zacatek(sloupec):Konec(sloupec)-1
         Pom=zeros(poc_trid,2);
         for j=Zacatek(sloupec):i
            if Pracovni(j,sl_X)>0
               Pom(Pracovni(j,sl_X),1)=Pom(Pracovni(j,sl_X),1)+1;
            end
         end
         for k=i+1:Konec(sloupec)
```

```
            if Pracovni(k,sl_X)>0
               Pom(Pracovni(k,sl_X),2)=Pom(Pracovni(k,sl_X),2)+1;
            end
         end
         N=sum(Pom);
         if ~((N(1)==0)|(N(2)==0))
            Pom_p=~Pom+Pom;
            G(i,sloupec)=abs(sum(Pom(:,1).*log(Pom_p(:,1)/N(1)))+
                                    sum(Pom(:,2).*log(Pom_p(:,2)/N(2))));
         end
      end % konec cyklu promenne i
   end % konec cyklu promenne sloupec
```

V dalších dvou vnořených cyklech se vypočítávají hodnoty dvousloupcové matice `Pom`. Po doběhnutí obou vnořených cyklů máme v této matici uloženy počty objektů spadajících do *k*-té třídy, kde řádkový index této matice odpovídá příslušné třídě (1. sloupec je pro levou větev a 2. sloupec je pro pravou větev dělení stromu podle `i`). Matice `Pom` je základem pro výpočet hodnot p_{ik} (viz. dělící kritérium 1 kapitola 3.5, str. 17). Ve vnořených cyklech jsou ještě podmínky, které vynechávají objekty s označením třídy 0. Tyto objekty byly již v průběhu tvoření stromu zaklasifikovány a není třeba je dále uvažovat.

Po vnořených cyklech následuje výpočet hodnoty kritéria pro dané dělení `i`. Hodnoty vektoru `N` jsou součty prvního a druhého sloupce matice `Pom`. Protože uvažujeme, že $0 \cdot \log 0 = 0$, je matice `Pom` upravena na matici `Pom_p`, která místo nul obsahuje jedničky. Hodnota dělícího kritéria `G(i,sloupec)` je pak součtem kritérií pro levou a pravou větev stromu v absolutní hodnotě. Po skončení cyklu proměnné `i` máme hodnoty dělícího kritéria pro první sloupec (charakteristiku) matice `Pracovni`. Následovně se program vrací na začátek cyklu proměnné `sloupec` (kód 4, str. 19), kde je její hodnota zvýšena o jedničku (pokud matice `Pracovni` obsahuje více než jednu charakteristiku).

Pro výše uvedený příklad (obrázek 6, str 20) matice `Pracovni` bude matice `G` vypadat následovně:

$$\begin{bmatrix} 1.9095 & 1.9095 \\ 1.3863 & 2.7726 \\ 3.2958 & 1.9095 \end{bmatrix}$$

Obr. 7

V následujícím bloku programu se matice `G` vyhodnocuje. Pomocí funkce *find* nalezneme vektory indexů `del_rd` a `del_sl` minimálních hodnot matice `G` (pokud se minimální hodnota vyskytuje v matici vícekrát, bereme v úvahu první nalezenou). Následně matici

`Pracovni` setřídíme podle sloupce, ve kterém byla nalezena minimální hodnota dělícího kritéria, a vypočteme dělící koeficient charakteristiky `del_koef` pro daný uzel stromu. Tento koeficient se vypočítává jako průměrná hodnota z hodnoty na dělícím řádku a z následujícího řádku dělícího sloupce v matici `Pracovni`.

(kód 9)

```
[del_rd,del_sl]=find(G==min(min(G)));
del_rd=del_rd(1); del_sl=del_sl(1);
Pracovni=sortrows(Pracovni,del_sl);
posun=1;
while Pracovni(del_rd+posun,sl_X)==0
   posun=posun+1;
end
%Vypocet deliciho koeficientu.
del_koef=mean([Pracovni(del_rd,del_sl) Pracovni
                                    (del_rd+posun,del_sl)]);
Strom(radek_strom,sloupec_strom)=del_koef;
Strom_sl(radek_strom,sloupec_strom)=del_sl;
Strom_rd(radek_strom,sloupec_strom)=del_rd;
```

Cyklem `while` se zjišťuje velikost hodnoty, která se přičítá k indexu dělícího řádku z důvodu správného výpočtu dělícího koeficientu (přeskočí se řádky s označením třídy 0, viz. kód 12, str. 24). Dále jsou hodnoty dělícího koeficientu `del_koef`, dělícího sloupce `del_sl` a dělícího řádku `del_rd` uloženy do matic `Strom`, `Strom_sl` a `Strom_rd` na pozice dané aktuálním uzlem stromu.

Další blok programu řeší výpočet hodnoty klasifikace pro levou (lichou) i pravou (sudou) větev stromu. Ve dvou cyklech zjistíme počty objektů spadajících do různých tříd. Hodnoty jsou stejně jako při výpočtu dělícího kritéria uloženy do dvousloupcové matice `Pom`. Hodnotou klasifikace rozumíme poměrné vyjádření počtu správně klasifikovaných objektů do tříd. V tomto programu ji vypočítáváme jako poměr maximálního výskytu objektu jedné třídy a všech objektů vstupujících do společné větve stromu. Hodnoty pro obě větve jsou uloženy ve vektoru `klas_n`. Do proměnných `trida_l` a `trida_s` se ukládají čísla tříd, do kterých byly klasifikovány objekty s největší klasifikací.

(kód 10)

```
%V tomto bloku se zjistuje hodnota klasifikace
Pom=zeros(poc_trid,2);
for j=Zacatek(del_sl):del_rd
   if Pracovni(j,sl_X)>0
      Pom(Pracovni(j,sl_X),1)=Pom(Pracovni(j,sl_X),1)+1;
   end
end
for k=del_rd+1:Konec(del_sl)
   if Pracovni(k,sl_X)>0
      Pom(Pracovni(k,sl_X),2)=Pom(Pracovni(k,sl_X),2)+1;
   end
```

```
end

klas=max(Pom);
klas_n=(klas./sum(pom));
trida_l=find(klas(1)==pom(:,1));
trida_s=find(klas(2)==pom(:,2));
```

Následně jsou hodnoty klasifikace uloženy do výstupních matic této funkce.

(kód 11)

```
Strom_klas(radek_strom+1,2*sloupec_strom-1)=klas_n(1);
Strom_klas(radek_strom+1,2*sloupec_strom)=klas_n(2);
Strom_trida(radek_strom+1,2*sloupec_strom-1)=
                                    (klas_n(1)>=(1-maxm))*trida_l(1);
Strom_trida(radek_strom+1,2*sloupec_strom)=
                                    (klas_n(2)>=(1-maxm))*trida_s(1);
```

V následujícím bloku programu jsou označeny již zaklasifikované objekty. Do posledního sloupce matice `Pracovni` je všem těmto objektů z příslušné větve stromu zapsána 0 a pro další dělení stromu se tato data již nebudou uvažovat. Podmínky zajišťují, zda pro levou nebo pravou větev je hodnota klasifikace vypočítaná v předchozím bloku programu rovna 1.

(kód 12)

```
if Strom_klas(radek_strom+1,2*sloupec_strom-1)==1
   for j=Zacatek(del_sl):del_rd
      Pracovni(j,sl_X)=0;
   end
end
if Strom_klas(radek_strom+1,2*sloupec_strom)==1
   for j=del_rd+1:Konec(del_sl)
      Pracovni(j,sl_X)=0;
   end
end
```

Následující podmínka určuje, zda bude dělení stromu ukončeno. Touto podmínkou může být dělení stromu ukončeno dvěma způsoby:

1. Všechny objekty jsou zaklasifikovány do tříd, součet hodnot posledního sloupce matice `Pracovni` je roven 0.
2. Maximální hodnota misklasifikace pro danou hloubku (řádek) stromu je menší nebo rovna hodnotě `maxm`, která je zadána jako druhý argument této funkce. Podmínka v logickém součinu s touto podmínkou kontroluje, zda je hodnota misklasifikace počítána pro všechny uzly z řádku stromu.

(kód 13)

```
max_misklas=1-min(Strom_klas(radek_strom+1,:));
```

```
   if (sum(Pracovni(:,sl_X))==0)|((max_misklas<=maxm)&
                                  (sloupec_strom==2^(radek_strom-1)))
      konec=1;
      break
   end
```

Proměnná `konec` je příznakem pro ukončení funkce. Příkazem `break` se pak vyskočí z nejvnitřnějšího cyklu `for` (v tomto případě v cyklu proměnné `sloupec_strom`).

Dále pak následuje druhá část podmínky z úvodu funkce (kód 4, str. 19). Nastává případ, kdy uzel byl vyhodnocen jako list.

(kód 14)

```
      else
         %Dalsi vetveni v teto vetvi stromu neni nutne, vse je
         %jiz zaklasifikovano, zapisuji se hodnoty z rodicovskeho uzlu
         Strom_klas(radek_strom+1,2*sloupec_strom-1)=1;
         Strom_klas(radek_strom+1,2*sloupec_strom)=1;
         Strom_trida(radek_strom+1,2*sloupec_strom-1)=
                              Strom_trida(radek_strom,sloupec_strom);
         Strom_trida(radek_strom+1,2*sloupec_strom)=
                             Strom_trida(radek_strom,sloupec_strom);
      end
end %konec cyklu promenne sloupec_strom
```

Do výstupních matic `Strom_klas` a `Strom_trida` se pro dané uzly zapisují stejné hodnoty, jako v rodičovském uzlu.

Za cyklem proměnné `sloupec_strom` následuje tato část funkce. Podmínka hlídá příznak `konec`. Pokud je nastaven do 1, pak se všechny výstupní matice (`Strom`, `Strom_sl`, `Strom_trida` a `Strom_klas`) uloží do jedné třírozměrné matice `Strom_vys`. Matice `Strom` a `Strom_sl` mají menší rozměr než ostatní dvě, proto musíme upravit jejich velikost (doplnění o nuly). Příkaz `break` vyskočí z hlavního cyklu proměnné `radek_strom` a tím je úplně ukončena činnost funkce.

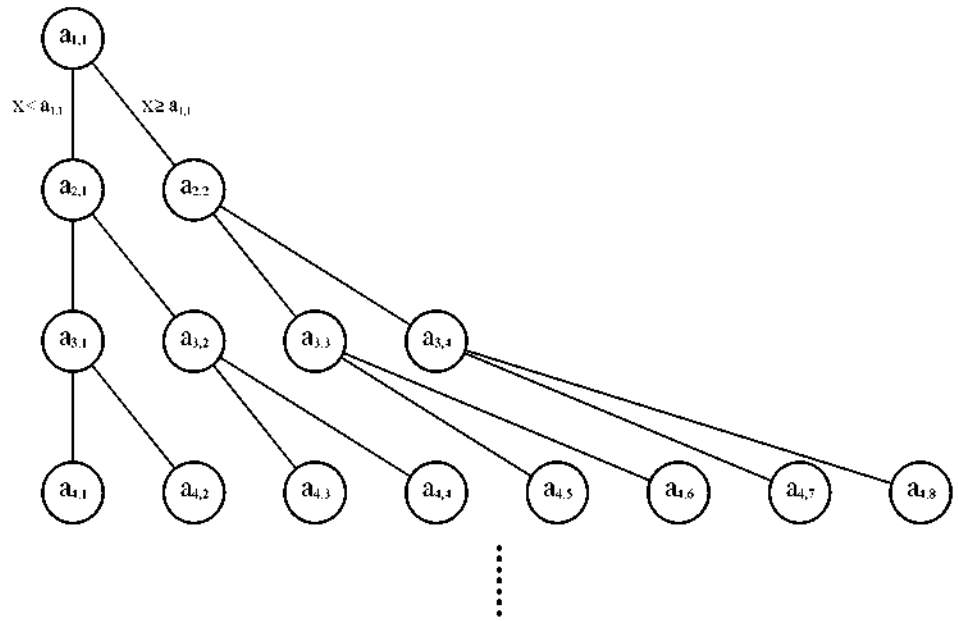
(kód 15)

```
      if konec==1
         %formovani vysledku
         [smr,sms]=size(Strom_klas);
         [sr,ss]=size(Strom);
         sp=zeros(smr,sms);
         sp(1:sr,1:ss)=Strom;
         spsl=zeros(smr,sms);
         spsl(1:sr,1:ss)=Strom_sl;
         Strom_vys(:,:,1)=sp;
         Strom_vys(:,:,2)=spsl;
         Strom_vys(:,:,3)=Strom_trida;
         Strom_vys(:,:,4)=Strom_klas;
         break
      end
   end
```

Pokud příznak `konec` není nastaven do jedničky, tělo podmínky je přeskočeno a program se vrací na začátek cyklu proměnné `radek_strom` (kód 3, str. 19), kde se zvětší o jedničku a program pokračuje v dělení stromu. Celý výpis programu je uveden v příloze A.

3.6.2 Popis struktury výstupní matice funkce *genstrom.m*

Výstupem funkce *genstrom.m* je, jak již bylo uvedeno, třírozměrná matice, která se skládá ze čtyř dvourozměrných matic. V první matici s indexem `(:,:,1)` jsou uloženy dělící hodnoty stromu. Struktura matice vychází z následujícího obrázku stromu.



Obr. 8

Hodnoty $a_{i,j}$ jsou dělící hodnoty stromu pro různé charakteristiky. Matice dělících hodnot pak vypadá následovně:

$$\begin{bmatrix} a_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ a_{2,1} & a_{2,2} & 0 & 0 & 0 & 0 & 0 & 0 & \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & 0 & 0 & 0 & 0 & \cdots\cdots \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} & \\ & & & \vdots & & & & & \end{bmatrix}$$

Obr. 9

Poslední řádek této matice je vždy nulový, z důvodu srovnání velikosti všech čtyř matic.

Druhá matice s indexem `(:,:,2)` udává číslo n charakteristiky X_n, pro kterou je příslušná dělící hodnota z první matice určena (n vyjadřuje číslo sloupce matice `Pracovni`, ve kterém bylo provedeno dělení v daném uzlu). Tato matice má totožný rozměr s první maticí, pro každou dělící hodnotu $a_{i,j}$ z první matice přísluší $n_{i,j}$ z matice druhé.

Další matice s indexem `(:,:,3)` obsahuje čísla tříd, do kterých byly objekty v listu nebo uzlu stromu zaklasifikovány. Čísla tříd jsou zapisována do této matice jen tehdy, pokud misklasifikace daného uzlu nebo listu je menší než zadaná maximální misklasifikace `maxm` jako argument funkce. Struktura matice je stejná s předchozími maticemi, první prvek na souřadnicích `(1,1,3)` této matice je vždy nulový (kořen stromu). Další hodnoty pak závisí na tom, zda je splněn požadavek maximální misklasifikace. Čísla tříd se vyskytují i na posledním řádku této matice.

Čtvrtou v pořadí je matice hodnot klasifikací pro jednotlivé uzly a listy stromu. Její index je `(:,:,4)`. Hodnoty se pohybují v intervalu $\langle 0,1 \rangle$ a vyjadřují míru úspěšnosti klasifikace v uzlu nebo listu. První prvek této matice je opět nulový.

3.6.3 Popis programu funkce *klasifikuj.m*

Funkce pro klasifikaci objektů podle vytvořeného stromu je definována takto:

(kód 16)

```
function [DATA_OUT]=klasifikuj(Strom,DATA_IN).
```

Prvním argumentem této funkce je třírozměrná matice generovaná funkcí *genstrom.m*. Druhým argumentem je matice charakteristik objektů, které chceme zaklasifikovat. Důležité je, aby charakteristiky objektů (sloupce matice) byly v matici uloženy ve stejném pořadí jako u trénovacích dat.

Nejprve se v programu připraví podle rozměrů vstupní matice `DATA_IN` výstupní matice `DATA_OUT`. Výstupní matice je o jeden sloupec větší než vstupní. Do tohoto sloupce se ukládají čísla tříd objektů.

(kód 17)

```
[r s]=size(DATA_IN);
v_strom=size(Strom);
%vytvoreni vystupni matice
DATA_OUT=zeros(r,s+1);
DATA_OUT(1:r,1:s)=DATA_IN;
```

Dále následuje cyklus pro procházení vstupní matice po řádkách. Pro každý řádek je strom procházen od kořene zvlášť, proto jsou indexy matice stromu nastaveny na první prvek (kořen stromu).

(kód 18)

```
for radek=1:r
   radek_strom=1;
   sloupec_strom=1;
```

Cyklus `while` prochází strom, dokud je hodnota čtvrté matice pro daný uzel menší jak 1 (hodnota klasifikace je menší než 1) a zároveň je hodnota v `radek_strom` menší než celková hloubka stromu (tato podmínka se uplatňuje v případě, když maximální misklasifikace stromu je větší jak 0). Podmínka v cyklu pak určuje cestu stromem porovnáním dělících hodnot v první matici a hodnot charakteristik ve vstupní matici. Prvek matice `DATA_IN` je dán indexy `(radek,Strom(radek_strom,sloupec_strom,2))`. Druhý index je prvek druhé matice stromu určující číslo charakteristiky.

(kód 19)

```
while (Strom(radek_strom,sloupec_strom,4)<1)&(radek_strom<v_strom(1))
   %rozhodovací podmínka
   if Strom(radek_strom,sloupec_strom,1)<=
                  DATA_IN(radek,Strom(radek_strom,sloupec_strom,2))
      sloupec_strom=sloupec_strom*2;
   else
      sloupec_strom=sloupec_strom*2-1;
   end
      radek_strom=radek_strom+1;
end
```

Pokud je rozhodovací podmínka splněna (hodnota charakteristiky objektu je větší než dělící hodnota stromu), pak cesta stromem povede pravou větví od rodičovského uzlu (`sloupec_strom=sloupec_strom*2`). Pokud není splněna, cesta stromem povede levou větví (`sloupec_strom=sloupec_strom*2-1`). Po podmínce se zvětší index řádku stromu o jedničku (hloubka stromu).

Po skončení cyklu známe indexy (`radek_strom` a `sloupec_strom`) konečného listu stromu. Číslo třídy je pak prvek třetí matice stromu s uvedenými indexy. Toto číslo se pak uloží do výstupní matice `DATA_OUT`.

(kód 20)

```
   DATA_OUT(radek,s+1)=Strom(radek_strom,sloupec_strom,3);
end %konec cyklu proměnne radek
```

Následně se program vrací zpět k cyklu `for` proměnné `radek` (viz. kód 18, str. 28) a následuje určení třídy pro další objekt. Po skončení tohoto cyklu máme ve výstupní matici u všech objektů přiřazeno číslo třídy, do které byl objekt zaklasifikován. Výpis celého programu je uveden v příloze A.

4. Texturní charakteristiky

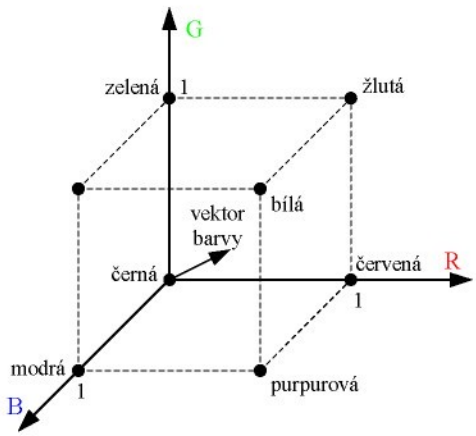
4.1 Digitální obraz a jeho reprezentace

Obrazem rozumíme dvourozměrnou reprezentaci (projekci) reálné třírozměrné scény. V ideálním případě ho můžeme popsat spojitou funkcí

$$z = f(x, y), \tag{3}$$

kde x a y jsou souřadnice plochy obrazu a z vyjadřuje barvu v daném bodě.

Pro počítačové zpracování je nutné obraz transformovat do digitálního tvaru. Tuto transformaci obvykle provádí snímací zařízení, které může být připojené k počítači (např.: skenr, digitální kamera apod.). Snímání je pak prováděno po bodech (pixelech) obrazu. Počet těchto bodů je dán rozlišovací schopností snímacího zařízení. Hodnota charakterizující bod v obraze může mít jen dvě úrovně při černobílém obrazu (binární obraz), nebo může vyjadřovat úroveň šedi bodu (monochromatický obraz), nebo může být reprezentována vektorem barvy bodu [8]. Pro barevné obrazy je většinou vektor barvy bodu dán třemi složkami RGB (R-červená, G-zelená, B-modrá) nebo čtyřmi složkami CMYK (C-azurová, M-purpurová, Y-žlutá, K-černá) [7]. Na obrázku 10 je znázorněn barevný prostor pro kódování RGB.



Obr. 10

V této práci se budeme ovšem dále výhradně zabývat monochromatickými obrazy, a to převážně s dvěma sty padesáti šesti odstíny šedi.

4.2 Texturní obrazy, textury

Textura je typ obrazu, jejímž hlavním znakem je určitá struktura, vzor (pattern). Předpokladem je, že struktura je homogenní a do určité míry pravidelná, tak aby bylo možné zachytit matematicky její základní vlastnosti. Typické pro texturu je, že popsáním lokálních vlastností a vztahů (pomocí charakteristik) získáme dostatečný popis pro celou homogenní texturu.

Většina texturních obrázků použitých v této práci je získána z uznávaného fotografického alba textur Phila Brodatze, které je volně šířeno na internetu [11].

4.3 Charakteristiky texturních obrazů

Protože textury mají zvláštní charakter, tj. jejich hlavním atributem je určitá struktura, odpovídají tomu i charakteristiky (příznaky) [4], [6], používané pro jejich popis a specifikaci. Sama textura (ostatně jako každý digitalizovaný obraz) je zadána jako diskrétní pole hodnot vyjadřující odstíny barev jednotlivých obrazových bodů (pixelů) nebo, jako v našem případě, pole odstínů (úrovní) šedi. Tyto hodnoty a jejich vzájemné vztahy (zejména v sousedství) jsou základem pro výpočet texturních charakteristik.

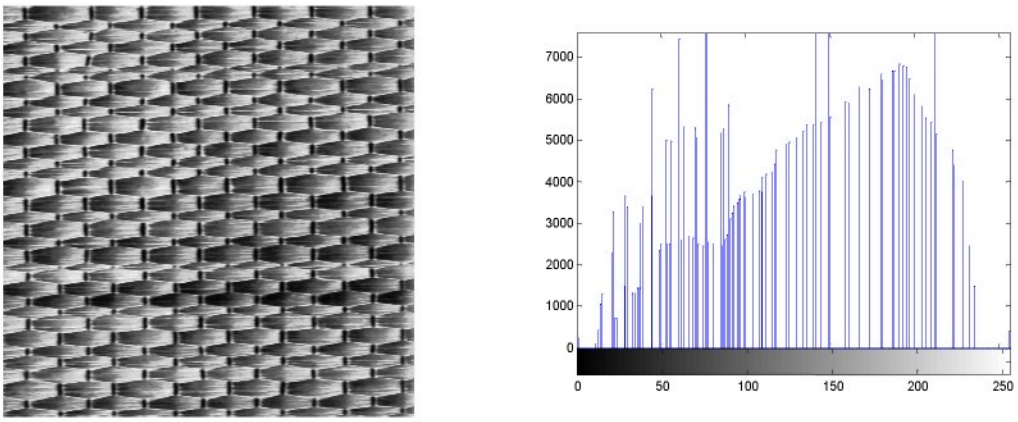
Charakteristiky texturních obrazů jsou často rozdělovány na charakteristiky prvního řádu, druhého řádu a vyššího řádu. Charakteristiky prvního řádu se vypočítávají ze zjištěných četností jednotlivých úrovní šedi v obraze. Charakteristiky druhého řádu jsou obvykle založeny na kookurenční matici, která popisuje četnost výskytu sousedních párů obrazových bodů pro různé úrovně šedi. Popisují tedy souvislosti v textuře v nejbližších bodech. Charakteristiky vyšších řádů jsou založeny na rozdělení tří a více obrazových bodů s různými úrovněmi šedi (např. v okolích řádu vyššího než jedna, ve vybraných oblastech textury apod.) Charakteristikami vyšších řádů se v této práci již dále nebudeme zabývat.

V následující kapitole je uveden přehled používaných charakteristik pro analýzu a klasifikaci texturních obrazů.

4.4 Charakteristiky prvního řádu

Charakteristiky obrazů textur vycházejí z histogramu úrovní šedi $\{H_i\}$. H_i vyjadřuje počet bodů v obraze s úrovní šedi i. Označíme-li N jako počet všech bodů v obraze a G jako počet použitých úrovní šedi (např. 16, 64, 256), potom $\sum_{i=0}^{G-1} H_i = N$. Pro normalizovaný histogram $\{h_i\}$ platí: $h_i = \frac{H_i}{N}$. Hodnoty h_i také vyjadřují pravděpodobnostní rozdělení úrovní šedi.

Na obrázku 11 je příklad textury (640×640 bodů, 256 odstínů šedi). Graf zobrazuje příslušný histogram obrázku textury$\{H_i\}$. Osa X tvoří stupnici úrovní šedi, osa Y vyjadřuje počet bodů v obraze s příslušnou úrovní šedi.



Obr. 11 - textura a histogram úrovní šedi

Jak už bylo uvedeno, normalizovaný histogram $\{h_i\}$ reprezentuje distribuci úrovní šedi i ve smyslu rozdělení pravděpodobnosti výskytu jednotlivých úrovní. Pokud si tedy konkrétní texturu představíme jako data-realizaci nějakého rozdělení pravděpodobnosti, tak pochopitelně vhodnými charakteristikami budou charakteristiky tohoto rozdělení. V následujících podkapitolách uvedeme základní z nich.

4.4.1 Odhad střední úrovně šedi

Pro střední úroveň šedi platí:

$$\mu = \sum_{i=0}^{G-1} i \cdot h_i \,. \tag{4}$$

Střední úroveň šedi μ nám dává informaci o průměrné intenzitě obrazu. Pro výpočet hodnoty střední úrovně šedi obrazu slouží v Matlabu funkce *stredhod.m*. Argumentem této funkce je matice obrazu.

4.4.2 Odhad rozptylu úrovně šedi

Pro rozptyl úrovně šedi platí:

$$\sigma^2 = \sum_{i=0}^{G-1} (i-\mu)^2 \cdot h_i \,. \tag{5}$$

Druhá odmocnina z rozptylu σ se nazývá směrodatná odchylka. Rozptyl a směrodatná odchylka nám dávají informaci o celkovém kontrastu obrázku. Hodnotu rozptylu vypočítáme pomocí funkce *rozptyl.m*. Argumentem této funkce je matice obrazu.

4.4.3 Odhad variačního koeficientu

Pro variační koeficient platí:

$$cv = \frac{\sigma}{\mu} \,. \tag{6}$$

Variační koeficient je invariantní vůči změně měřítka úrovní šedi $i' = A \cdot i$. K výpočtu hodnoty variačního koeficientu slouží funkce *varkoef.m*. Argumentem této funkce je opět matice obrazu.

4.4.4 Výběrová šikmost úrovně šedi

Pro výběrovou šikmost platí:

$$\gamma_1 = \frac{1}{\sigma^3} \cdot \sum_{i=0}^{G-1} (i-\mu)^3 \cdot H_i \,. \tag{7}$$

Výběrová šikmost je invariantní vůči lineární transformaci úrovní šedi $i' = A \cdot i + B$. Hodnotu výběrové šikmosti získáme pomocí funkce *vybsikm.m*. Argumentem je matice obrazu.

4.4.5 Výběrová špičatost úrovně šedi

Pro výběrovou špičatost platí:

$$\gamma_2 = \frac{1}{\sigma^4} \cdot \sum_{i=0}^{G-1} (i-\mu)^4 \cdot h_i - 3 . \tag{8}$$

Výběrová špičatost úrovně šedi udává míru podobnosti obrysu histogramu s Gaussovským rozdělením. Hodnota γ_2 je rovna 0, právě když jde o Gaussovské rozdělení. Výběrová špičatost je také invariantní vůči lineární transformaci úrovní šedi $i' = A \cdot i + B$. Pro výpočet hodnoty výběrové špičatosti slouží funkce *vybspic.m*. Argumentem této funkce je matice obrazu.

4.4.6 Energie úrovně šedi

Pro energii úrovně šedi platí:

$$e = \sum_{i=0}^{G-1} h_i^2 . \tag{9}$$

Hodnota energie se pohybuje v intervalu $G^{-1} \le e \le 1$. Energie nám udává míru neuspořádanosti histogramu. Funkce *energ.m* vrací hodnotu energie daného obrazu. Vstupní údaj této funkce je matice obrazu.

4.4.7 Entropie úrovně šedi

Pro entropii úrovně šedi platí:

$$s = -\sum_{i=0}^{G-1} h_i \cdot \log h_i . \tag{10}$$

Hodnota energie se pohybuje v intervalu $0 \le s \le \log G$. Tato veličina udává míru uspořádanosti histogramu. Často se používá v oblasti komprese obrazů. Hodnotu entropie získáme pomocí funkce *entrop.m*. Argumentem je matice obrazu.

4.5 Shrnutí charakteristik prvního řádu, ekvalizace

Charakteristiky prvního řádu jsou závislé na světelných podmínkách, proto je třeba tento vliv eliminovat. Obvykle se tento problém řeší pomocí transformace zvané *ekvalizace histogramu*, kde z původního obrazu vytvoříme nový. Tato transformace změní rozložení jednotlivých úrovní šedi tak, aby se v něm vyskytovaly pokud možno v co největším

rozmezí, a to přibližně se stejnou četností. Ekvalizace [9] umožňuje v obraze s celkově vysokým kontrastem zvýraznit špatně rozpoznatelné detaily s nízkým kontrastem. Výsledná intenzita obrazového bodu po ekvalizaci *I'(x,y)* se vypočítá z původní hodnoty úrovně šedi obrazového bodu *I(x,y)*, a to pomocí vztahu

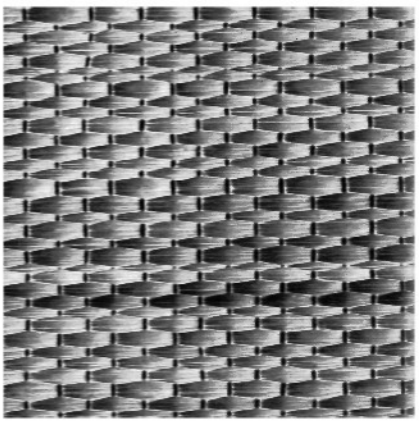
$$I'(x,y) = \frac{B-1}{N} \cdot \sum_{i=I_0}^{I(x,y)} H_i \,, \tag{11}$$

kde *x,y* jsou souřadnice bodu v obraze, *B* je počet úrovní šedi v novém obrazu, *N* je celkový počet bodů v obraze, I_0 je nejnižší úroveň šedi v původním obrazu a H_i je *i*-tá složka nenormalizovaného histogramu původního obrazu.

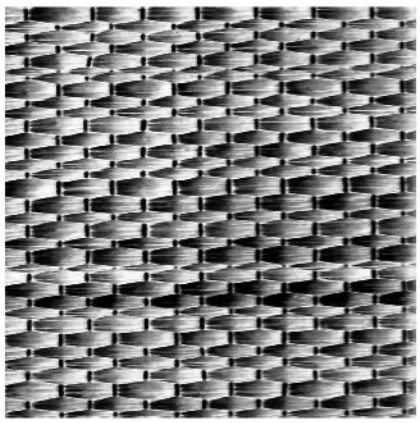
Výpočet ekvalizace obrazu můžeme provést v *Matlabu* pomocí funkce *ekvaliz.m*. Vstupním argumentem je matice obrazu, u kterého chceme ekvalizaci provést. Funkce pak vrací ekvalizovanou matici původního obrazu. Na následující straně je uveden příklad ekvalizace textury vypočtené pomocí této funkce.

4.5.1 Příklad použití ekvalizace

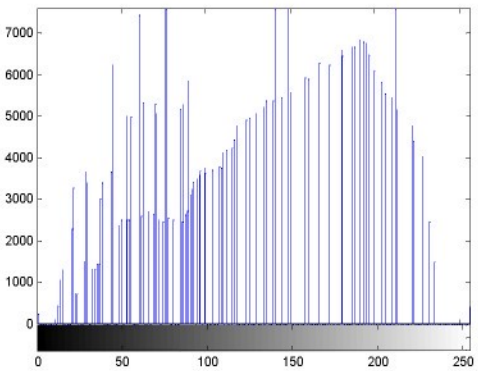
Pro ilustraci použití ekvalizace byla použita textura z obrázku 11 (str. 32).



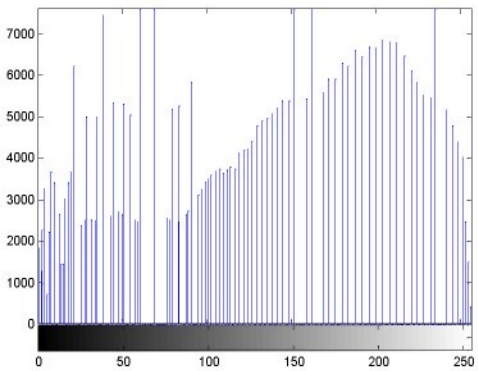
Obr. 12a - textura



Obr. 12c - ekvalizovaná textura



Obr. 12b - histogram textury



Obr. 12d - histogram ekvalizované textury

Na histogramu ekvalizovaného obrázku (obr. 12d, str. 35) je vidět, že jednotlivé jasové úrovně jsou rozloženy rovnoměrněji. V nově vytvořeném obrazu (obr. 12c, str. 35) se ekvalizace projevila zvýšením kontrastu.

4.6 Charakteristiky druhého řádu, kookurenční matice

Tyto charakteristiky vypovídají o vztahu úrovně šedi mezi sousedními body obrazu. Charakterizují vzájemnou závislost, korelaci, asociaci a lze je také chápat jako charakteristiky dvourozměrného rozdělení pravděpodobnosti.

Základem pro výpočet charakteristik druhého řádu je kookurenční matice [4]. Kookurenční matice $\{c_{ij}(r),i=0,....,G-1,j=0,...,G-1\}$ je čtvercová matice, kde počet řádků a sloupců je roven počtu úrovní šedi G. Je definovaná vzhledem k danému posunutí $r=(k,l)$ a obrazovému bodu (x,y), hodnota c_{ij} označuje počet výskytů obrazových bodů s úrovní šedi i a bodů, jež mají úroveň j a tyto body jsou od sebe vzdáleny právě o posunutí r. Označíme-li N_r jako celkový počet těchto dvojic, pak

$$C_{ij}(r) = \frac{c_{ij}(r)}{N_r} \tag{12}$$

je normalizovaná kookurenční matice. Pro lepší ilustraci je uveden příklad výpočtu kookurenční matice na zvětšeném obrázku 13 o velikosti 10x10 obrazových bodů, který má čtyři úrovně šedi.

0	0	0	0	1	1	0	0	0	0
0	0	0	1	3	3	1	0	0	0
0	0	0	2	3	3	2	0	0	0
0	0	1	3	2	2	3	1	0	0
0	0	2	3	1	1	3	2	0	0
0	1	3	2	1	1	2	3	1	0
0	2	3	3	3	3	3	3	2	0
1	3	2	1	1	1	1	2	3	1
2	3	1	0	0	0	0	1	3	2
1	1	0	0	0	0	0	0	1	1

Obr. 13

i, j	0	1	2	3
0	19	3	3	4
1	7	4	1	7
2	1	5	3	4
3	1	8	6	5

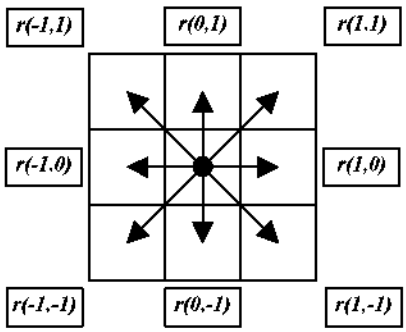
Tab. 2a

i, j	0	1	2	3
0	0,235	0,037	0,037	0,049
1	0,086	0,049	0,012	0,086
2	0,012	0,062	0,037	0,049
3	0,012	0,099	0,074	0,062

Tab. 2b

Pro výpočet kookurenční matice bylo použito relativní posunutí *r=(1,1)*, tedy dvojic bodů *(x,y)* a *(x+1,y+1)*. Takových dvojic je pro tento obrázek $N_r=81$. V tabulce 2a (str. 36) jsou uvedeny hodnoty kookurenční matice $c_{ij}(r)$, v tabulce 2b (str. 36) jsou uvedeny hodnoty normalizované kookurenční matice $C_{ij}(r)$.

Kookurenční matice obrazu se obvykle vypočítává pro různá relativní posunutí *r*. Pro jeden bod *(x,y)* existuje celkem osm bodů sousedních, tedy i osm posunutí *r* (viz. obrázek 14). Protože čtyři z nich jsou jen zpětnými posunutími k ostatním čtyřem, pak při předpokladu symetrie vzájemných vztahů sousedních bodů v textuře vystačíme se čtyřmi směry (např. *r(1,1), r(1,0), r(1,-1), r(0,-1)*).



Obr. 14

Pro kookurenční matici dále platí:

$$C_{ij}(-r) = C^T(r). \tag{13}$$

Této vlastnosti se využívá pro výpočet symetrické kookurenční matice $c_S(r)$, platí:

$$c_S(r) = c(r) + c^T(r) \tag{14}$$

a pro normalizovanou kookurenční matici

$$C_S(r) = \frac{1}{2} \cdot \left[C(r) + C^T(r)\right]. \tag{15}$$

Pro klasifikaci se často využívá tzv. izotropní kookurenční matice, která vznikne součtem kookurenčních matic s různým směrem posunutí *r*.

$$c_I(1) = c_S(0,1) + c_S(1,0) + c_S(1,1) + c_S(-1,1) \tag{16}$$

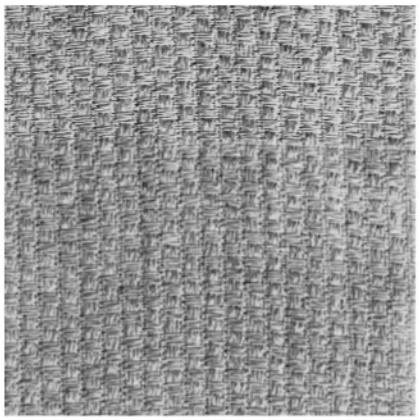
a

$$C_I = \frac{1}{4} \cdot \left[C_S(0,1) + C_S(1,0) + C_S(1,1) + C_S(-1,1)\right]. \tag{17}$$

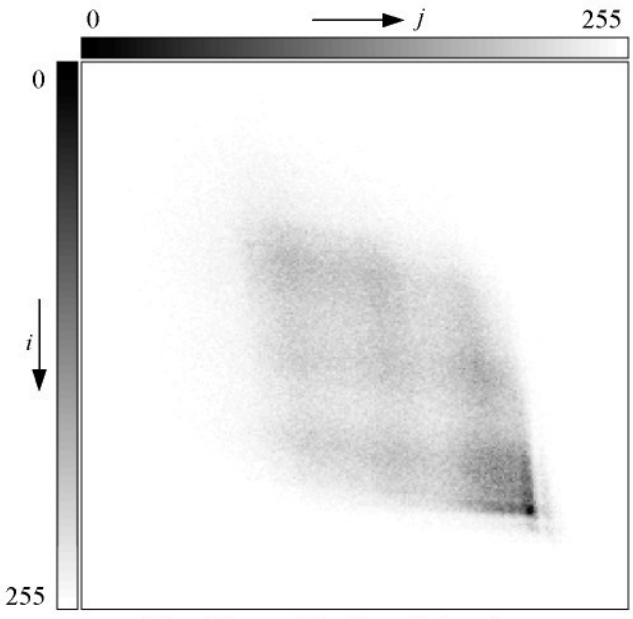
Jeden z problémů výpočtu kookurenčních matic je ten, že se pracuje s dosti rozsáhlými daty. Například pro obrázek s barevnou hloubkou dvou set padesáti šesti úrovní šedi má kookurenční matice 65536 prvků ($=256^2$).

Konkrétní příklad kookurenční matice je uveden na obrázku 15b. Kookurenční matice byla vypočítána z obrázku textury 15a o rozměrech 640×640 bodů s dvou set padesáti šesti odstíny šedi pro jedno relativní posunutí $r=(1,1)$. Kookurenční matice je zobrazena jako dvourozměrný graf, který pro dané souřadnice matice i, j znázorňuje četnost výskytu dvou bodů, vzdálených o relativní posunutí r s úrovněmi šedi i a j.

Četnost je znázorněna úrovní šedi, bílá barva → nulový výskyt, černá barva → maximální výskyt (v našem případě je roven hodnotě $c_{210,210}=138$, $C_{210,210}=3.3797\cdot 10^{-4}$, jde tedy o sousední body se stejnou úrovní šedi 210).



Obr. 15a - textura



Obr. 15b - graf kookurenční matice

Pro výpočet normalizované kookurenční matice můžeme použít funkci *kookur.m*. Pro výše uvedený příklad byla použita funkce *kookurabs.m*, která vrací nenormalizovanou kookurenční matici. Vstupní hodnoty pro obě dvě funkce jsou: matice obrazu, vektor posunutí *r* a nepovinný parametr počtu úrovní šedi (pokud není parametr zadán, uvažuje se hodnota *256*).

4.7 Výpočty charakteristik druhého řádu

4.7.1 Energie (Druhý angulární moment)

Pro energii platí:

$$\varepsilon = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1} C_{ij}^2 \,. \tag{18}$$

Hodnota ε se pohybuje v intervalu $G^{-2} \le \varepsilon \le 1$. ε je rovno hodnotě G^{-2} při rovnoměrném rozdělení C a hodnotě 1, právě když je jeden prvek nenulový. K výpočtu hodnoty energie kookurenční matice slouží funkce *energkoo.m*. Argumentem je normalizovaná kookurenční matice.

4.7.2 Entropie kookurenční matice

Pro entropii platí:

$$S = -\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} C_{ij} \cdot \log C_{ij} \,. \tag{19}$$

Hodnota S se pohybuje v intervalu $0 \le S \le \log G^2$. S je rovno hodnotě $\log G^2$ při rovnoměrném rozdělení C a hodnotě 0, právě když je jeden prvek nenulový. Pokud je prvek kookurenční matice $C_{ij} = 0$, pak uvažujeme, že i výraz $C_{ij} \cdot \log C_{ij}$ je též roven nule. Hodnotu entropie získáme pomocí funkce *entrkoo.m*. Jejím argumentem je opět normalizovaná kookurenční matice.

4.7.3 Maximální pravděpodobnost

Pro maximální pravděpodobnost platí

$$M = \max C_{ij} . \tag{20}$$

Tato pravděpodobnost vyjadřuje maximální výskyt dvojic bodů s úrovněmi šedi *i* a *j*. Pro výpočet této charakteristiky slouží funkce *maxpr.m*. Argumentem této funkce je normalizovaná kookurenční matice.

4.7.4 Mezivýpočet

Pro výpočet dalších charakteristik je třeba provést mezivýpočet.

Nechť

$$C_i^x = \sum_{j=0}^{G-1} C_{ij} , \tag{21, 22}$$

$$C_j^y = \sum_{i=0}^{G-1} C_{ij}$$

a μ_x , μ_y , σ_x a σ_y jsou střední hodnoty a směrodatné odchylky C_i^x a C_j^y, pro které platí:

$$\mu_x = \sum_{i=0}^{G-1} i \cdot C_i^x , \qquad \sigma_x = \sum_{i=0}^{G-1} (i - \mu_x)^2 \cdot C_i^x , \tag{23, 24, 25, 26}$$

$$\mu_y = \sum_{j=0}^{G-1} j \cdot C_j^y , \qquad \sigma_y = \sum_{j=0}^{G-1} (j - \mu_y)^2 \cdot C_j^y .$$

4.7.5 Autokorelace

Pro autokorelaci platí:

$$\rho = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \frac{(i - \mu_x) \cdot (j - \mu_y) \cdot C_{ij}}{\sigma_x \cdot \sigma_y} . \tag{27}$$

Hodnota ρ se pohybuje v intervalu $-1 \le \rho \le 1$. ρ nabývá hodnoty 1, právě když jsou jen na hlavní diagonále nenulové hodnoty. Hodnoty 0 nabývá tehdy, jsou-li hodnoty nekorelované. Hodnotu autokorelace získáme pomocí funkce *autokorel.m*. Argumentem je normalizovaná kookurenční matice.

4.7.6 Diagonální moment

Pro diagonální moment platí:

$$D = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1} |i-j| \cdot (i+j-\mu_x-\mu_y) \cdot C_{ij} . \tag{28}$$

Diagonální moment D udává rozdíl v korelaci mezi vyššími a nižšími úrovněmi šedi. Hodnotu diagonálního momentu získáme pomocí funkce *diagmom.m*. Argumentem této funkce je opět normalizovaná kookurenční matice.

4.7.7 Rozdělení absolutních hodnot rozdílů úrovní šedi dvojic bodů

Z kookurenční matice úrovní šedi obdržíme odhad vektoru rozdělení D_k:

$$D_k = \underbrace{\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} C_{ij}}_{|i-j|=k} , \; k = 0,\ldots,G-1 . \tag{29}$$

Vektor D_k popisuje rozdělení vzdálenosti od hlavní diagonály v kookurenční matici. Tento vektor získáme pomocí funkce *rozroz.m*, argumentem je normalizovaná kookurenční matice. Na tomto vektoru rozdělení jsou postaveny následující charakteristiky druhého řádu.

4.7.8 Energie rozdělení absolutních rozdílů úrovní šedi

Pro energii rozdělení platí:

$$\varepsilon_D = \sum_{k=0}^{G-1} D_k^2 . \tag{30}$$

Hodnota ε_D se pohybuje v intervalu $G^{-1} \le \varepsilon_D \le 1$. Hodnotu energie rozdělení získáme pomocí funkce *enerdk.m*. Argumentem této funkce je vektor D_k.

4.7.9 Entropie rozdělení absolutních rozdílů úrovní šedi

Pro entropii rozdělení platí:

$$S_D = -\sum_{k=0}^{G-1} D_k \cdot \log D_k . \tag{31}$$

Hodnota S_D se pohybuje v intervalu $0 \leq S_D \leq \log G$. K výpočtu hodnoty entropie rozdělení slouží funkce *entrdk.m*. Vstupním argumentem funkce je vektor D_k.

4.7.10 Variogram (Inerce, Kontrast)

Pro výpočet variogramu platí:

$$V_D = -\sum_{k=0}^{G-1} k^2 \cdot D_k = 2 \cdot \sigma^2 \cdot (1-\rho). \tag{32}$$

σ^2 je rozptyl úrovní šedi a ρ je korelace. Hodnotu získáme pomocí funkce *variog.m*. Vstupním argumentem je opět vektor D_k.

4.7.11 Lokální homogenita

Pro lokální homogenitu platí:

$$L_D = \sum_{k=0}^{G-1} \frac{D_k}{(1+k^2)}. \tag{33}$$

K výpočtu hodnoty lokální homogenity slouží funkce *lokhom.m*. Argumentem funkce je vektor D_k.

4.7.12 Rozdílový rozptyl

Pro rozdílový rozptyl platí:

$$R_D = V_D - \left(\sum_{k=0}^{G-1} k \cdot D_k \right)^2. \tag{34}$$

Hodnotu rozdílového rozptylu R_D získáme pomocí funkce *rrozdk.m*. Argumentem funkce je opět vektor D_k.

4.7.13 Rozdělení hodnot součtů úrovní šedi dvojic bodů

Z kookurenční matice úrovní šedi můžeme rovněž získat vektoru S_k:

$$S_k = \underbrace{\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} C_{ij}}_{i+j=k}, \quad k = 0,\ldots,2G-2 \tag{35}$$

a pro výběrový průměr tohoto rozdělení platí

$$a_S = \sum_{k=0}^{2G-2} k \cdot S_k = \mu_x + \mu_y . \tag{36}$$

Vektor S_k získáme pomocí funkce *rozsouc.m*. Na tomto vektoru rozdělení jsou založeny následující charakteristiky druhého řádu:

4.7.14 Energie rozdělení součtů úrovní šedi

Pro energii platí:

$$\varepsilon_S = \sum_{k=0}^{2G-2} S_k^2 . \tag{37}$$

Hodnota ε_S se pohybuje v intervalu $(2G-2)^{-1} \le \varepsilon_S \le 1$. Funkce pro výpočet hodnoty energie ε_S má název *enersk.m*. Vstupním argumentem je vektor S_k.

4.7.15 Entropie rozdělení součtů úrovní šedi

Pro entropii platí:

$$S_S = -\sum_{k=0}^{2G-2} S_k \cdot \log S_k . \tag{38}$$

Hodnota entropie S_S se pohybuje v intervalu $0 \le S_S \le \log(2G-2)$. Hodnotu entropie můžeme získat pomocí funkce *entrsk.m*. Vstupním argumentem je vektor S_k.

4.7.16 Odhad rozptylu rozdělení součtů úrovní šedi

Pro odhad rozptylu platí:

$$V_S = -\sum_{k=0}^{2G-2} (k - a_S)^2 \cdot S_k . \tag{39}$$

K výpočtu hodnoty rozptylu S_k slouží funkce *rrozsk.m*. Argumentem je opět vektor S_k.

4.7.17 Odhad třetího momentu rozdělení součtů úrovní šedi

Pro odhad třetího momentu platí:

$$H_S = \sum_{k=0}^{2G-2} (k - a_S)^3 \cdot S_k \,. \tag{40}$$

Hodnotu třetího momentu rozdělení S_k vypočítáme pomocí funkce *m3rozsk.m*. Argumentem této funkce je vektor S_k.

4.7.18 Odhad čtvrtého momentu rozdělení součtů úrovní šedi

Pro odhad čtvrtého momentu platí:

$$P_S = \sum_{k=0}^{2G-2} (k - a_S)^4 \cdot S_k \,. \tag{41}$$

Pro výpočet hodnoty čtvrtého momentu rozdělení S_k slouží funkce *m4rozsk.m*. Argumentem je opět vektor S_k.

4.8 Shrnutí charakteristik druhého řádu

Charakteristiky druhého řádu jsou založeny na prostorovém uspořádání bodů v obraze, a proto jsou vhodné pro popis textur. Samozřejmě pro výpočet charakteristik (části) textury předpokládáme, že textura (resp. část, ze které získáme charakteristiky) je homogenní, tj. že vypočítané hodnoty skutečně texturu reprezentují.

Naopak sledování změn charakteristik na různých částech texturního obrazu má význam pro detekci odchylek v homogenitě analyzované textury. V praxi slouží např. pro detekci změn vlastností materiálu, který opticky hodnotíme pomocí texturní analýzy. Názorný příklad detekce poruch ve struktuře textury je uveden v kapitole 5 (Příklad 2, str. 53).

4.9 Přehled vytvořených funkcí pro výpočet charakteristik

Všechny funkce jsou naprogramovány v jazyce MATLAB a jsou programovány tak, aby použitý algoritmus výpočtu byl proveden co nejrychleji a zároveň byl srozumitelný. Krátký popis funkce je k dispozici po zadání příkazu *help* a názvu funkce v příkazovém okně MATLAB (např.: *» help kookur*).

Je důležité, aby byla v MATLABu zaregistrována cesta k adresáři, ve kterém jsou uloženy tyto funkce (nastavení cesty v příkazovém okně *Matlab*: *File* → *Set Path* nebo ikona *Path Browser* ) [10].

4.9.1 Funkce pro výpočet charakteristik prvního řádu

stredhod.m	- výpočet odhadu střední úrovně šedi
rozptyl.m	- odhad rozptylu úrovně šedi
varkoef.m	- odhad variačního koeficientu
vybsikm.m	- výběrová šikmost úrovně šedi
vybspic.m	- výběrová špičatost úrovně šedi
energ.m	- energie úrovně šedi
entrop.m	- entropie úrovně šedi

4.9.2 Funkce pro výpočet charakteristik druhého řádu

kookur.m	- výpočet normalizované kookurenční matice
kookurabs.m	- výpočet nenormalizované (absolutní) kookurenční matice
energkoo.m	- energie kookurenční matice (druhý angulární moment)
entrkoo.m	- entropie kookurenční matice
maxpr.m	- maximální pravděpodobnost
autokorel.m	- autokorelace
diagmom.m	- diagonální moment
rozroz.m	- vektor odhadu rozdělení absolutních hodnot rozdílů úrovní šedi dvojic bodů
enerdk.m	- energie rozdělení absolutních rozdílů úrovní šedi
entrdk.m	- entropie rozdělení absolutních rozdílů úrovní šedi
variog.m	- variogram (inerce, kontrast)

lokhom.m - lokální homogenita

rrozdk.m - rozdílový rozptyl

rozsouc.m - vektor odhadu rozdělení součtů hodnot úrovní šedi dvojic bodů

enersk.m - energie rozdělení součtů úrovní šedi

entrsk.m - entropie rozdělení součtů úrovní šedi

rrozsk.m - odhad rozptylu rozdělení součtů úrovní šedi

m3rozsk.m - odhad třetího momentu rozdělení součtů úrovní šedi

m4rozsk.m - odhad čtvrtého momentu rozdělení součtů úrovní šedi

4.9.3 Funkce pro předzpracování obrazu

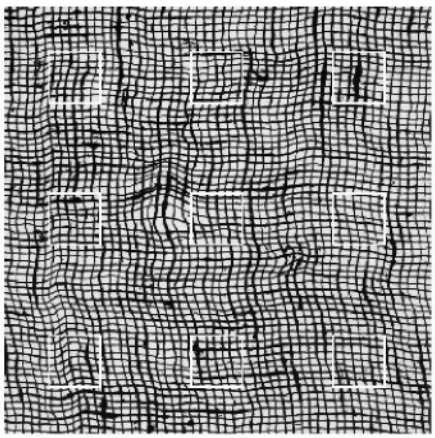
ekvaliz.m - provede ekvalizaci obrazu (vytvoří nový obraz)

Výpisy programů těchto funkcí jsou uvedeny v příloze B.

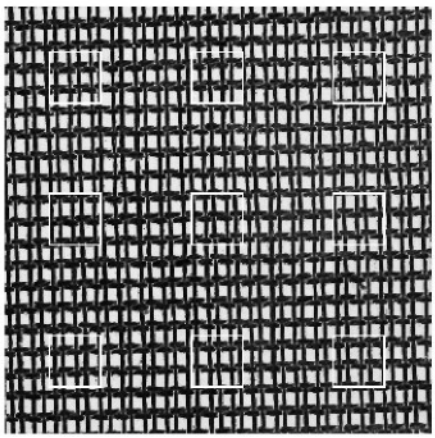
5. Příklady

5.1 Příklad 1

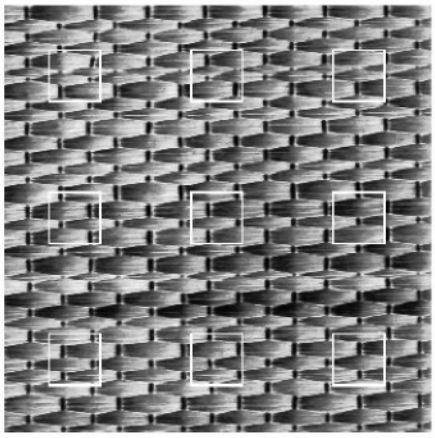
Konkrétní aplikace klasifikačního stromu je ukázána na příkladu, ve kterém se klasifikují náhodně vybraná okna (oblasti) textur z následujících šesti texturních obrázků. Trénovací data pro klasifikační strom byla vybrána ze stejných obrázků a jsou vyznačena bílým rámečkem.



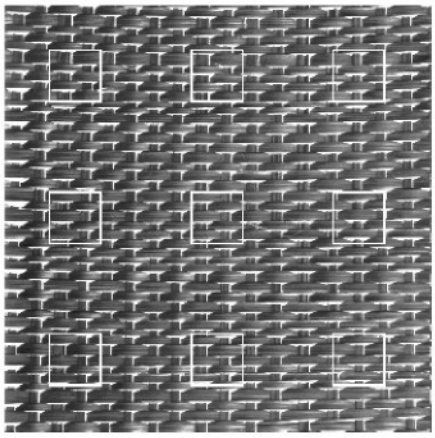
Obr. 16a (třída 1)



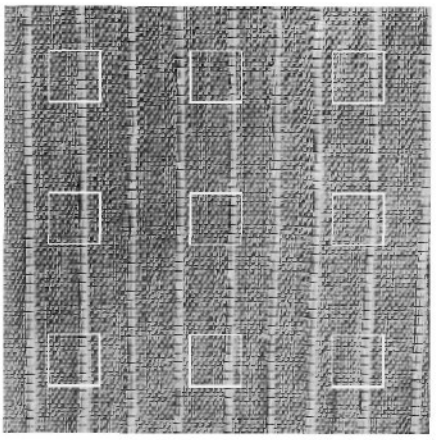
Obr. 16b (třída 2)



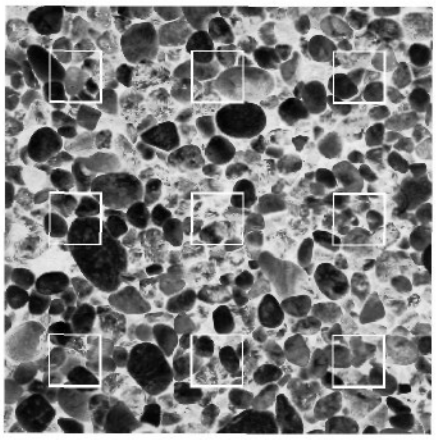
Obr. 16c (třída 3)



Obr. 16d (třída 4)

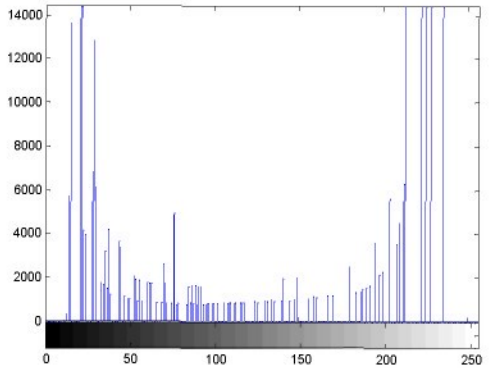


Obr. 16e (třída 5)

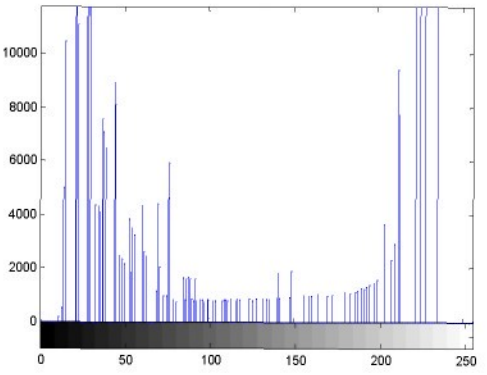


Obr. 16f (třída 6)

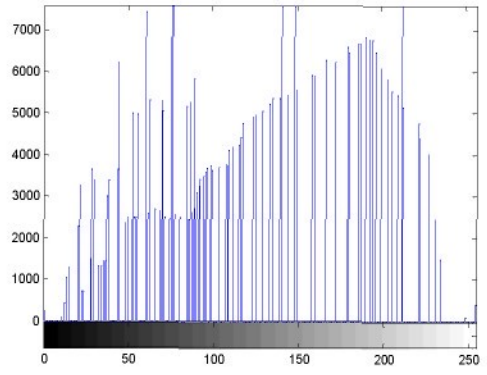
Na následujících obrázcích (obr. 17a ÷ 17f) jsou uvedeny grafy histogramů úrovní šedi pro výše uvedené texturní obrazy.



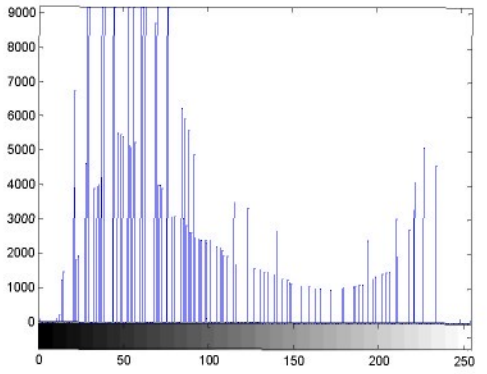
Obr. 17a - histogram textury (obr. 16a)



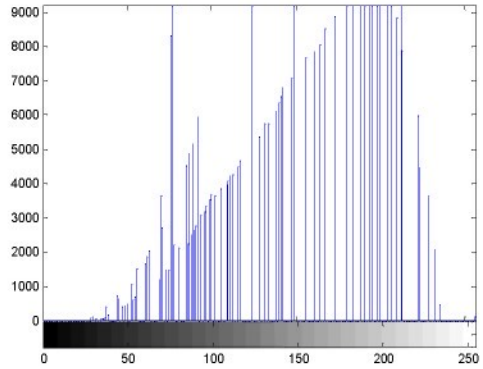
Obr. 17b - histogram textury (obr. 16b)



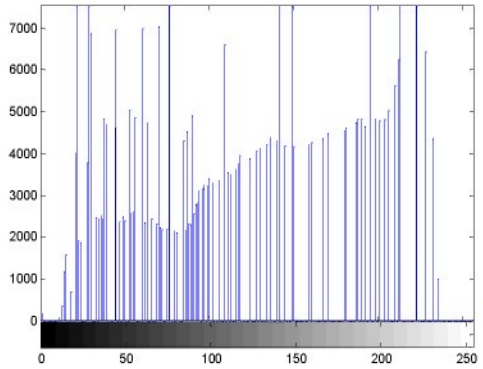
Obr. 17c - histogram textury (obr. 16c)



Obr. 17d - histogram textury (obr. 16d)



Obr. 17e - histogram textury (obr. 16e)



Obr. 17f - histogram textury (obr. 16f)

Obrázky výše uvedených textur mají velikost 640×640 bodů. Trénovací data charakteristik byla vypočítána z devíti oken (velikost okna byla zvolena 80×80 bodů) každého obrázku.

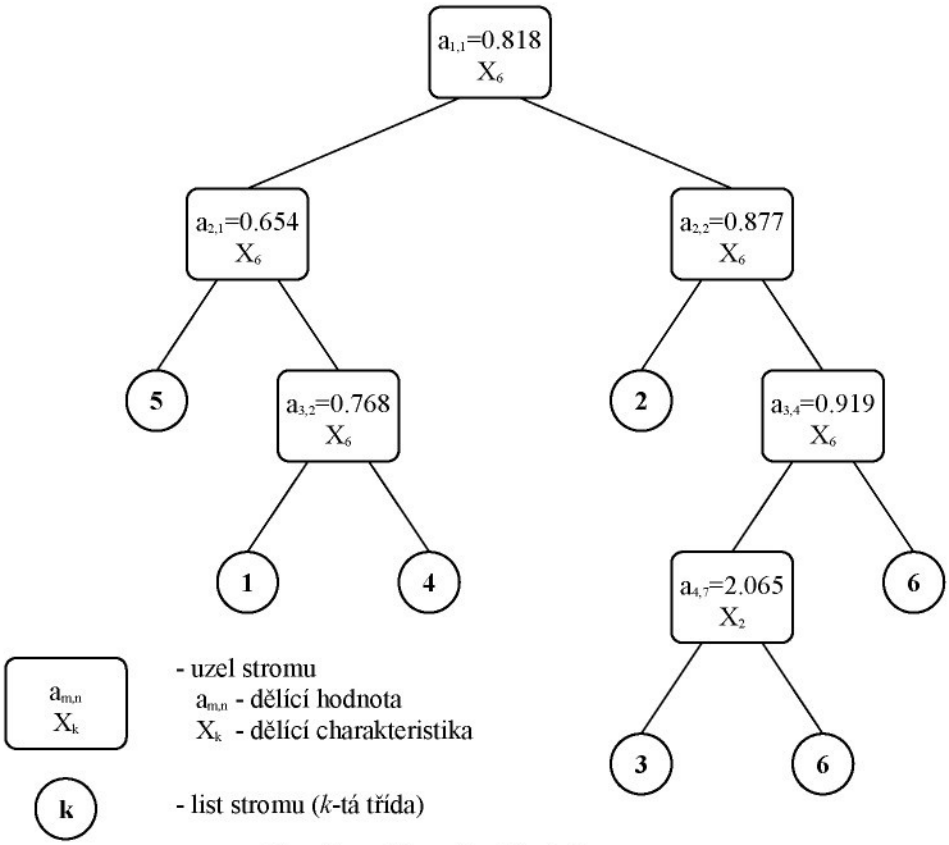
Typy charakteristik vybrané pro popis těchto objektů jsou uvedeny v následující tabulce:

Název charakteristiky		Označení
Odhad střední úrovně šedi μ	(str. 33)	X_1
Výběrová špičatost úrovně šedi γ_2	(str. 34)	X_2
Výběrová šikmost úrovně šedi γ_1	(str. 33)	X_3
Energie úrovně šedi e	(str. 34)	X_4
Energie kookurenční matice ε	(str. 39)	X_5
Autokorelace σ	(str. 40)	X_6
Maximální pravděpodobnost M	(str. 40)	X_7
Energie rozdělení absolutních rozdílů úrovní šedi ε_D	(str. 41)	X_8
Variogram V_D	(str. 42)	X_9
Energie rozdělení součtů úrovní šedi ε_S	(str. 43)	X_{10}

Tab. 3 - charakteristiky trénovacích dat

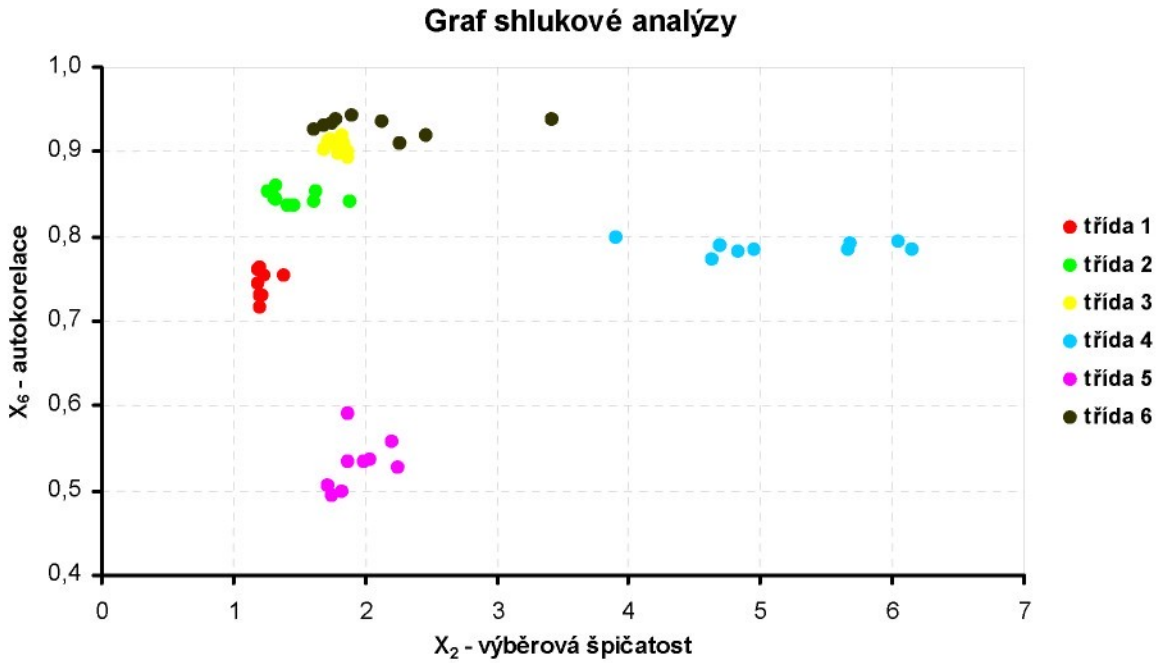
Vypočítané hodnoty charakteristik (trénovacích data) pro všechny obrázky jsou uvedeny v příloze C.

Struktura klasifikačního stromu vytvořeného pomocí funkce *genstrom.m* na základě těchto trénovacích dat je znázorněna na obrázku 18 na následující straně. Hodnota maximální misklasifikace byla zvolena 0 (druhý argument funkce *genstrom.m*, viz. kód 1, str. 18).



Obr. 18 - schéma klasifikačního stromu

Z obrázku klasifikačního stromu je zřejmé, že dělící pravidlo stromu vynechalo většinu charakteristik trénovacích dat. Z deseti charakteristik zbyly pouze dvě: autokorelace - X_6, která se na dělení stromu podílí nejvíce, a výběrová špičatost úrovně šedi - X_2. Závislost obou charakteristik nejlépe vystihuje následující graf 1.



Graf 1 - shluková analýza

Pro ověření funkčnosti vygenerovaného stromu byla opět použita data z uvedených obrázků. Nejprve bylo z každého obrázku náhodně vybráno dvacet tři oken o stejné velikosti (80×80 bodů) jako v případě trénovacích dat. Pro tato okna se pak dále vypočítaly hodnoty autokorelace a výběrové špičatosti.

Pomocí funkce *klasifikuj.m* se podle výše uvedeného stromu k těmto hodnotám přiřadila čísla tříd. Výsledek klasifikace je uveden v tabulce 3a.

Správná čísla tříd	1	2	3	4	5	6
Klasifikované oblasti obrázků	1	2	3	4	5	6
	1	2	6	4	5	6
	1	2	3	4	5	6
	1	2	3	4	5	6
	1	2	6	2	5	6
	1	2	3	4	5	6
	4	2	3	4	5	6
	4	2	6	4	5	6
	1	2	3	4	5	6
	1	2	3	4	5	6
	1	2	3	4	5	6
	1	2	3	4	5	6
	1	2	3	4	5	6
	1	2	6	4	5	6
	1	2	6	4	5	6
	1	2	3	4	5	3
	1	2	3	4	5	6
	1	2	3	4	5	3
	1	2	3	4	5	3
	1	2	3	4	5	6
	1	3	3	4	5	6
	4	2	3	4	5	3
	1	2	3	4	5	3
Počet chybně klasifikovaných oblastí	3	1	5	1	0	5

Tab. 3a

1	2	3	4	5	6
1	2	6	4	5	6
1	2	3	4	5	6
4	2	6	4	5	6
1	2	3	4	5	6
1	2	3	2	5	3
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	6	4	5	6
1	2	6	4	5	6
1	2	3	4	5	6
1	2	3	2	5	3
1	2	3	4	5	2
1	2	3	4	5	6
1	2	6	4	5	6
1	2	6	4	5	6
1	2	6	4	5	6
1	2	6	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
4	2	3	4	5	6
1	2	3	4	5	6
1	2	6	4	5	6
4	2	3	4	1	6
3	0	8	2	1	3

Tab. 3b

Z tabulky 3a je zřejmé, že ne všechna okna textur byla správně klasifikována. Celkem bylo špatně klasifikováno 15 oken ze 138. Celková úspěšnost klasifikace byla tedy 89.1%.

V druhém případě se u oken měnila náhodně i jejich velikost, a to v rozmezí 40×40 až 120×120 bodů. Počet náhodně vybraných oken zůstal stejný. Výsledek klasifikace je zobrazen v tabulce 3b. V tomto případě bylo špatně klasifikováno 17 oken textur. Výsledná úspěšnost klasifikace byla 87.7%.

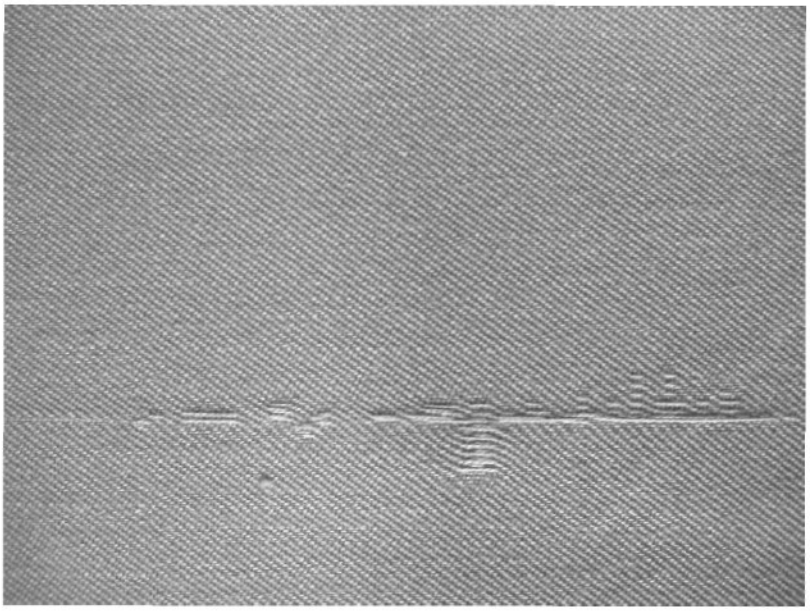
5.1.1 Shrnutí příkladu 1

Z tabulek je patrné, že nejvíce chyb klasifikace se objevilo v třetí a šesté třídě. Je to způsobeno tím, že shluky bodů pro tyto třídy jsou velice blízko u sebe a použité charakteristiky je nedokážou dostatečně oddělit. Z úspěšnosti klasifikace druhého případu lze usoudit, že na výslednou klasifikaci nemá velikost okna rozhodující vliv, avšak je potřeba zachovat nějakou minimální velikost okna, z důvodu zachycení struktury textury.

Na tomto příkladu je názorně ukázána výhoda klasifikačního stromu z hlediska výběru charakteristik. Charakteristiky, které jsou nejméně rozlišující pro třídy, nejsou brány v úvahu. Obrázky textur byly získány z [11].

5.2 Příklad 2

Následující příklad aplikace klasifikačního stromu je zaměřen na klasifikaci poruch v texturním obraze. Jako obraz textury byla vybrána fotografie textilního materiálu (viz. obrázek 19), na kterém jsou jasně zřetelné poruchy struktury materiálu (chybějící nebo přetržená vlákna).

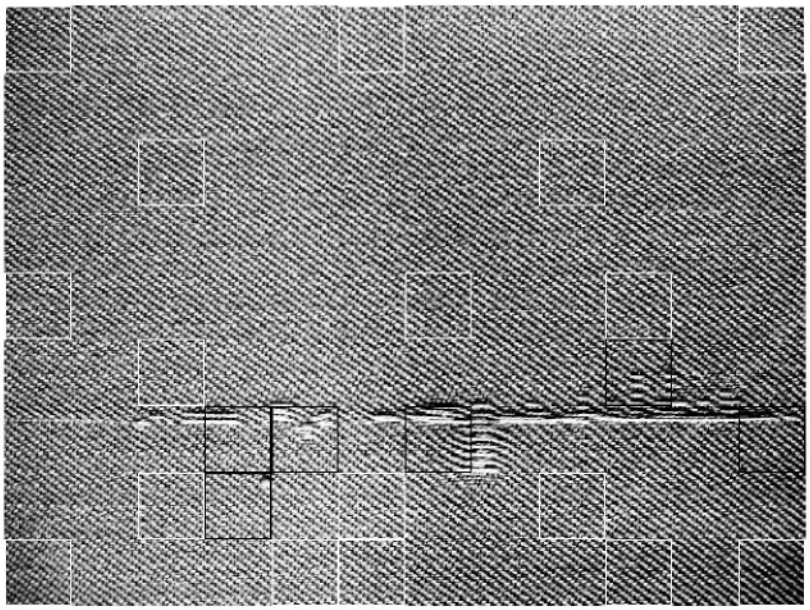


Obr. 19 - textura s poruchami

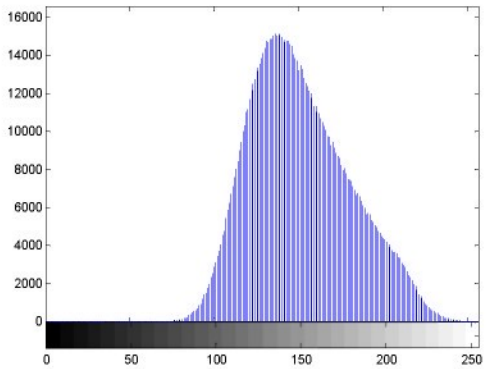
Cílem tohoto příkladu je zkonstruovat klasifikační strom, který rozlišuje, zda vybraná oblast (okno) obrazu obsahuje nebo neobsahuje poruchu struktury. Klasifikační strom tedy bude klasifikovat dvě třídy, do třídy 1 bude zařazovat okna bez poruchy struktury a do třídy 2 bude zařazovat okna, ve kterých se vyskytla porucha struktury.

Nejprve byla provedena ekvalizace obrázku textury pomocí funkce *ekvaliz.m*, při které došlo ke zvýšení kontrastu. Výsledek této ekvalizace je zobrazen na obrázku 20 (str. 54), na obrázku 21 (str. 54) jsou uvedeny histogramy úrovní šedi pro původní (viz. obrázek 21a, str. 54) a ekvalizovaný (viz. obrázek 21b, str. 54) obrázek.

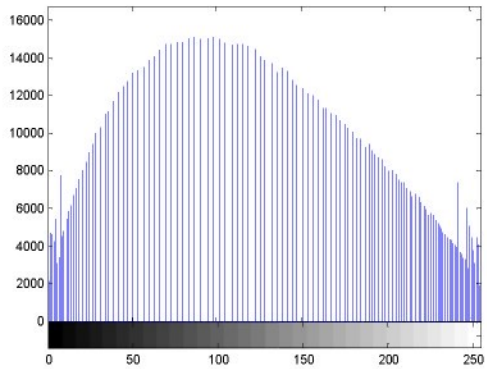
Dále pak byl obrázek rozdělen na sto osm oken o velikosti 100×100 bodů (rozměry obrázku jsou 1200×900 bodů, 12×9 oken). Pro výpočet trénovacích dat byla vybrána okna označená bílým (okna bez poruch) nebo černým rámečkem (okna s poruchami), zobrazená též na obrázku 20 (str. 54). Celkově bylo vybráno dvacet čtyři oken z třídy 1 a šest oken z třídy 2.



Obr. 20 - ekvalizovaná textura s označením oken trénovacích dat



Obr. 21a - histogram textury před ekvalizací



Obr. 21b - histogram textury po ekvalizaci

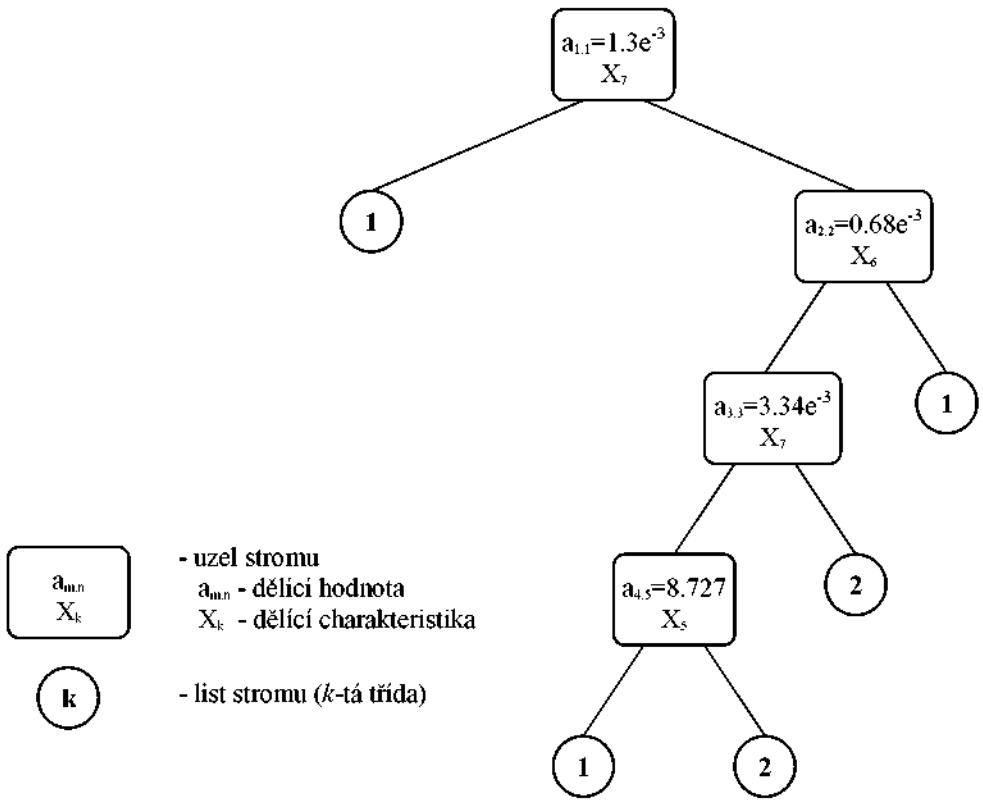
Trénovací data jednotlivých oken pro konstrukci klasifikačního stromu byla vypočítána z charakteristik uvedených v následující tabulce.

Název charakteristiky		Označení
Odhad rozptylu úrovně šedi σ^2	(str. 33)	X_1
Výběrová špičatost úrovně šedi γ_2	(str. 34)	X_2
Výběrová šikmost úrovně šedi γ_1	(str. 33)	X_3
Autokorelace σ	(str. 40)	X_4
Entropie kookurenční matice S	(str. 39)	X_5
Energie (Druhý angulární moment) ε	(str. 39)	X_6
Maximální pravděpodobnost M	(str. 40)	X_7

Tab. 4

Tabulka hodnot vypočítaných trénovacích dat z vybraných oken pro klasifikační strom je uvedena v příloze D.

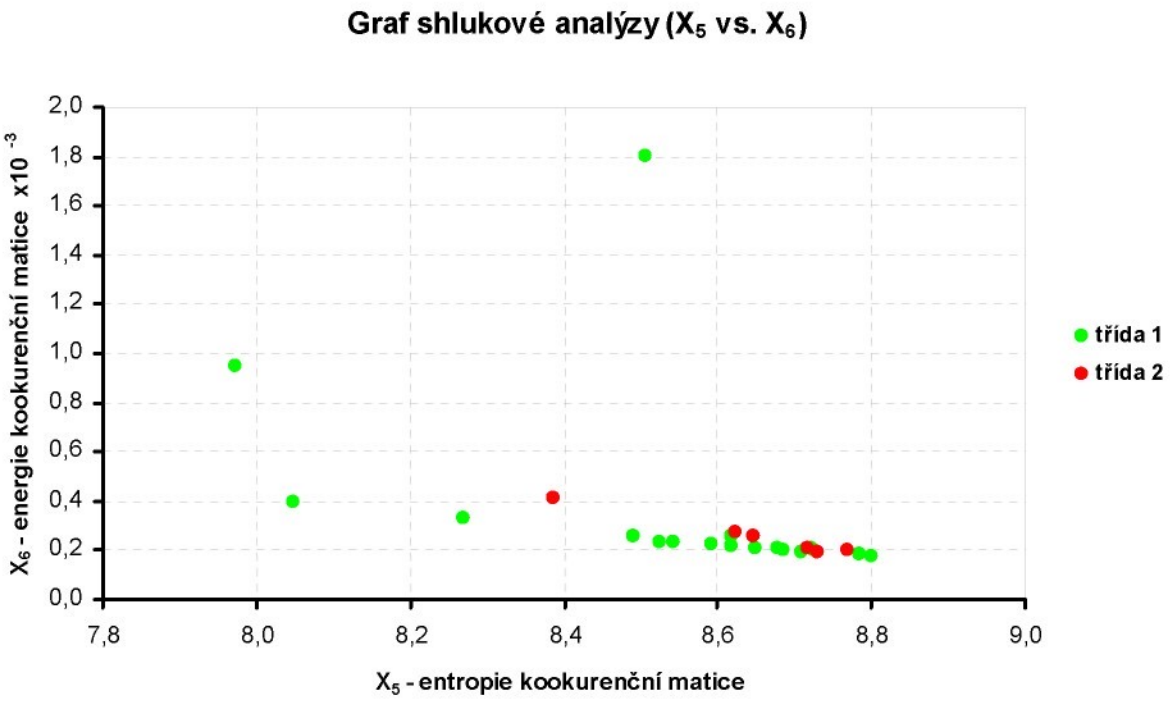
Na základě těchto trénovacích dat byl pomocí funkce *genstrom.m* vytvořen následující klasifikační strom (maximální hodnota misklasifikace byla zvolena 0).



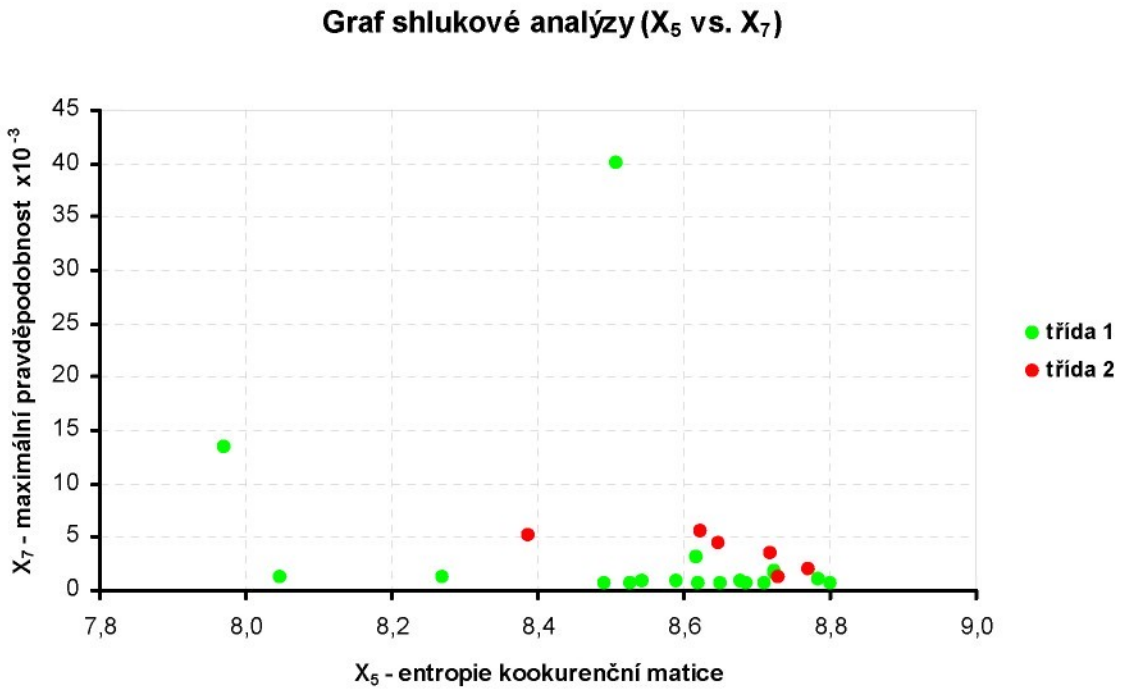
Obr. 22 - schéma klasifikačního stromu

Z obrázku klasifikačního stromu je vidět, že dělící pravidlo stromu vynechalo charakteristiky prvního řádu a autokorelaci. Zbyly tedy tyto charakteristiky: entropie kookurenční matice - X_5, energie kookurenční matice - X_6 a maximální pravděpodobnost - X_7.

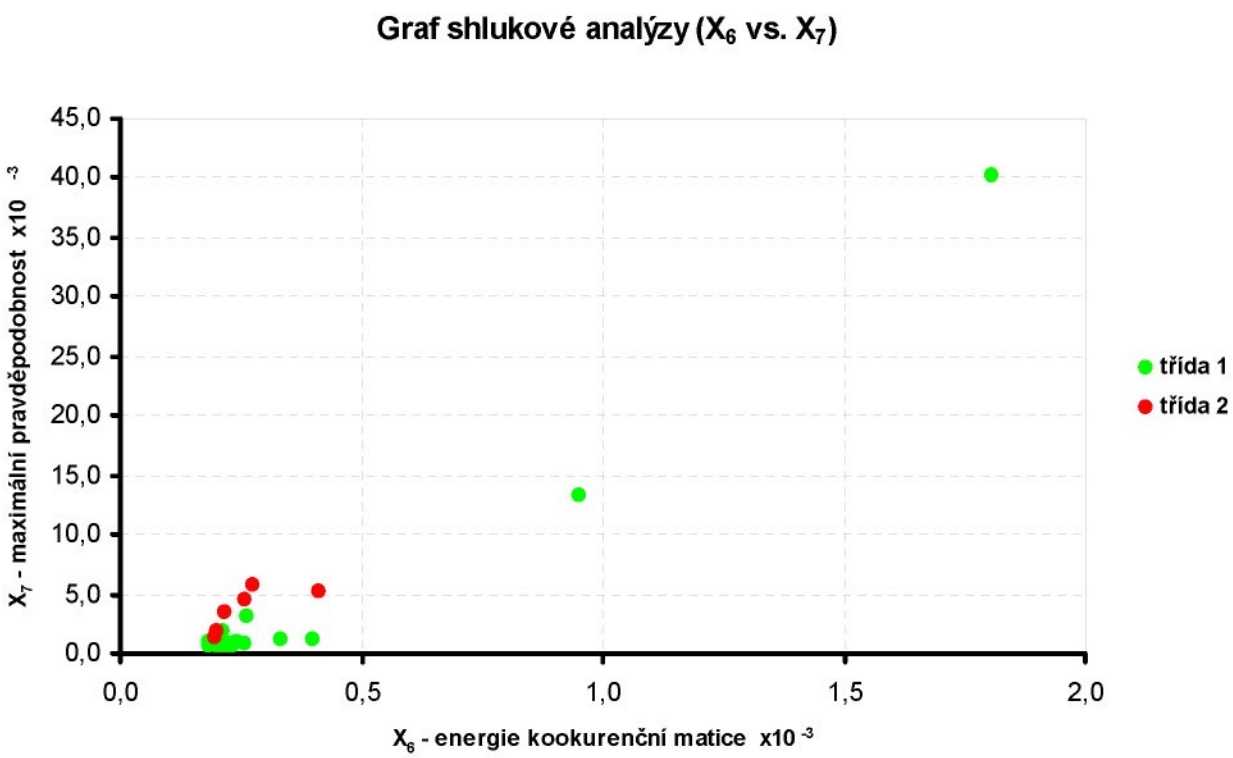
Vzájemné závislosti těchto charakteristik vystihují tři grafy (graf 2, 3 a 4) shlukové analýzy, uvedené na dalších dvou stranách.


Graf shlukové analýzy (X5 vs. X6)
X6 - energie kookurenční matice x10 -3
2,0
1,8
1,6
1,4
1,2
1,0
0,8
0,6
0,4
0,2
0,0
7,8
8,0
8,2
8,4
8,6
8,8
9,0
X5 - entropie kookurenční matice
třída 1
třída 2

Graf 2


Graf shlukové analýzy (X5 vs. X7)
X7 - maximální pravděpodobnost x10 -3
45
40
35
30
25
20
15
10
5
0
7,8
8,0
8,2
8,4
8,6
8,8
9,0
X5 - entropie kookurenční matice
třída 1
třída 2

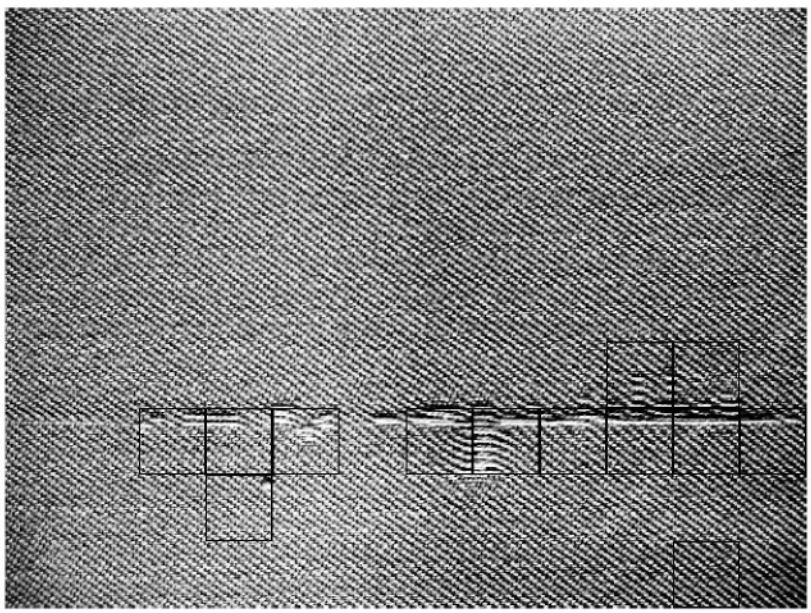
Graf 3



Graf 4

Ani na jednom z uvedených grafů nejsou zřetelné hranice mezi oběma třídami. Vždy některý objekt z jedné třídy splývá s objekty druhé třídy.

Pro ověření klasifikace byly vypočítány charakteristiky vybrané klasifikačním stromem pro všech sto osm oken obrázku textury. Funkcí *klasifikuj.m* se na základě vygenerovaného stromu a vypočtených charakteristik pro všechna okna obrázku textury (viz. obrázek 20, str. 54) přiřadila čísla tříd.



Obr. 23 - klasifikovaná okna s poruchou struktury

Výsledek klasifikace je zřejmý z obrázku 23 na předchozí straně. Černými rámečky jsou označena okna, která klasifikační strom zařadil do druhé třídy, tedy okna s poruchou struktury. Je vidět, že dvě okna nebyla správně klasifikována. Jedno okno bylo klasifikováno jako oblast bez poruchy (tedy do třídy 1, správně mělo být do třídy 2), ačkoli se v tomto okně porucha struktury nachází, a u druhého okna tomu bylo naopak.

Celková úspěšnost klasifikace byla 97,6% $\left(=100\cdot\left(1-\frac{2}{108-24}\right)\right)$, z hlediska klasifikace poruch pak 85.7% $\left(=100\cdot\left(1-\frac{1}{13-6}\right)\right)$.

5.2.1 Shrnutí příkladu 2

Na tomto příkladu byla ukázána další možnost využití klasifikačního stromu. Narozdíl od příkladu 1 zde nejsou třídy na shlukových grafech zřetelně odděleny, avšak relativně vysoká úspěšnost klasifikace potvrzuje schopnost klasifikačního stromu i tyto třídy oddělit. Pro lepší oddělení tříd by musela být speciálně pro tuto úlohu navržena charakteristika, která by lépe popisovala strukturu textury použitého obrázku.

Z grafů shlukové analýzy je patrné, že hodnoty trénovacích charakteristik jednoho okna z třídy 1 jsou dosti odlišné od ostatních. Jedná se o okno v pravém dolním rohu obrázku textury (obr. 20, str. 54), jehož celková průměrná úroveň stupně šedi je daleko menší než u ostatních oken z třídy 1 (je to pravděpodobně způsobeno nedokonalým nasvícením vzorku materiálu při vytváření jeho fotografie).

Pro praktické využití klasifikace v rozpoznávání defektů textilních materiálů je cílem analýzy klasifikačním stromem především vytipovat významné charakteristiky a ověřit jejich rozlišovací schopnost na datech. V další fázi by pak tyto charakteristiky (a jejich dělící hodnoty získané klasifikačním stromem) mohly být přímo použity k rozhodování o přítomnosti defektu (daného typu) na sledovaném vzorku materiálu, a to i v rámci sledování jakosti v reálném čase.

Obrázek textury použité v tomto příkladě byl získán z [12].

6. Závěr

Hlavním cílem této diplomové práce bylo vytvoření algoritmu pro generování klasifikačního stromu, tento algoritmus implementovat do programového prostředí MATLAB a navrhnout jeho využití pro analýzu a rozpoznávání texturních obrazů.

Výsledkem jsou dva programy *genstrom.m* a *klasifikuj.m*. Oba programy jsou implementovány v MATLABu jako funkce, což je výhodné pro použití v dalších aplikačních programech. První z uvedených funkcí generuje na základě trénovacích dat klasifikační strom. Struktura stromu je uložena v třírozměrné matici, která je též výstupem této funkce. Výhodou takto uložené struktury stromu je jednoduchá zpracovatelnost v ní uložených dat, nevýhodou může být velikost této matice, obzvláště pro klasifikační stromy s velkou hloubkou. Druhá uvedená funkce pak na základě vygenerovaného stromu a matice dat objektů, které chceme klasifikovat, přiřazuje čísla tříd, do kterých dané objekty náleží.

Dalším cílem této práce bylo získání (vytipování) charakteristik vhodných pro klasifikaci texturních obrazců. Tyto charakteristiky nám číselně popisují vlastnosti texturních obrazců a podle nich je pak můžeme klasifikovat. V této práci byly popsány charakteristiky prvního řádu, založené na histogramu úrovní šedi, a druhého řádu, založené na kookurenční matici. Pro výpočty hodnot těchto charakteristik byly vytvořeny funkce (též implentované do MATLABu), které zjednodušují následné aplikační programy.

Úlohu optimálního výběru charakteristik řeší výše zmíněný algoritmus klasifikačního stromu. Tento algoritmus má schopnost z více charakteristik (trénovací data) vybrat pouze ty nejvíce rozlišující.

Použití vytvořených programů je znázorněno na dvou příkladech. První příklad řeší úlohu klasifikace typu texturních obrazů, druhý příklad je zaměřen na vyhledávání poruch ve struktuře texturního obrazu. Na příkladech bylo zároveň ukázáno, jak lze tyto procedury využít pro počítačovou analýzu rozpoznávání vzorků opticky snímaných materiálů a detekci jejich defektů.

Použitá literatura

[1] Breiman, L. - Freidman, J. H. - Olshen, R. A. - Stone, C. J.: Classification and Regression Trees. Monterey, CA, 1984.

[2] Keprta, S.: Klasifikátory konstruované pomocí rekurzivních postupů. Diplomová práce. Univerzita Karlova, Praha, 1993.

[3] Volf, P.: O statistických metodách pro klasifikaci. Sborník Analýza dat. TriloByte, Pardubice, 1996.

[4] Carstensen, J.M.: Description and Simulation of Visual Textures. Thesis, TU Lyngby, 1992.

[5] Kotek, Z. - Mařík, V. - Hlaváč, V. - Psutka J. - Zdráhal, Z.: Metody rozpoznávání a jejich aplikace. Academia Praha, 1993.

[6] Linka, A. - Volf, P.: Statistické metody pro hodnocení homogenity materiálů. Sborník Robust 2000. JČMF Praha, 2000.

[7] Banks, S.: Signal Processing, Image Processing and Pattern Recognition. Prentice. Hall International, London, 1990.

[8] Sobotka, Z. - Sobotka, M.: Digitální zpracování a přenos obrazu II. Dům techniky ČSVTS Praha, 1987.

[9] Nouza, J.: Metody rozpoznávání. Přednášky, 2000.

[10] MATLAB. Function Reference, Image Processing Toolbox, Statistics Toolbox. The Math Works Inc., 1996.

Internetové odkazy

[11] Brodatz Textures - http://www.ux.his.no/~tranden/brodatz.html

[12] Textury textilií - http://siprint.utia.cas.cz/public/income/volf/obrazky_defect/Defects/

Přílohy

Příloha A

Zdrojový výpis programu funkce *genstrom.m*

```
%[STROM]=genstrom(DATA,MAXMISKLAS,MAXHLOUBKA) - Vystup: 3-dimensionalni
%  matice, STROM(:,:,1) zde jsou ulozeny delici hodnoty charakteristik,
%  STROM(:,:,2) zde je ulozeno poradi charakteristiky pro
%  danou delici hodnotu (cislo sloupce ve vstupnich datech),
%  STROM(:,:,3) zde je hodnota klasifikovane tridy pro dany list
%  (0 - jeste nezaklasifikovano) a STROM(:,:,4) zde jsou ulozeny hodnoty
%  klasifikace (misklasifikace=1-klasifikace; hodnota
%  klasifik. =1 -> vse je zaklasifikovano)
%
%  Vstup:   DATA - matice (m x n) trenovacich dat,
%  X1,X2,..,Xn - cherakteristiky, Y - trida (cela cisla 1..k).
%  MAXMISKLAS - hodnota maximalni misklasifikace (interval <0,0.5),
%     hodnota 0 -> vsechny prvky budou zaklasifikovany.
%  MAXHLOUBKA - omezeni maximalni hloubky stromu (cele cislo > 1),
%     doporucena max. hodnota je 16 (pro vetsi cisla zacne
%     neumerne rust velikost vystupni matice)
%
%Priklad matice vstupnich dat:
%
%           X1   X2   Y
%  DATA=[ 1.8   35   1 ;
%          2    15   2 ;
%         2.1   18   2 ;
%         3.6   28   3 ]
%
%  Priklad pouziti:  strom=genstrom(DATA,0,16);

function [Strom_vys] = genstrom(X,maxm,maxh)

format short g;
Strom_klas=0;                       %hodnota klasifikaci stromu
Pracovni=X;                         %pracovni matice, v cyklu meni obsah
[rd_X,sl_X]=size(Pracovni);         %rozmer matice X, radky,sloupce
poc_trid=max(Pracovni(:,sl_X));     %pocet klasifikovanych trid
konec=0;                            %konec==1 => konec tvoreni stromu

%Hlavni cyklus,maximalni pocet radku stromu je ulozen v promenne maxh
for radek_strom=1:maxh
   for sloupec_strom=1:2^(radek_strom-1)

%Podminka pro pokracovani vetveni stromu
%(nasled. list stromu musi mit hodnotu misklasifikace <1)
      if Strom_klas(radek_strom,sloupec_strom)<1

 %cyklus pres sloupce prac. matice(jednotlive charakteristiky X1,X2,..)
         for sloupec=1:sl_X-1
            %vzestupne serazeni prac. matice podle prom. 'sloupec'
            Pracovni=sortrows(Pracovni,sloupec);
            %podminka prvniho vetveni stromu
            %(tento blok se nevykonava pri prvnim vetveni)
            if radek_strom>1
%Cesta_zpet - 2 radkovy vektory pro zpetne generovani cesty, pro
%nalezeni indexu 'Zacatek' a 'Konec'
%1. radek - delici radky daneho sloupce
%2. radek - sudy (=1) nebo lichy (=0) sloupec pro dany delici radek
               Cesta_zpet=[0 rd_X;1 0];
```

```
                %cyklus hledani zpetne cesty k pocatku stromu
                for a=1:radek_strom-1
                   rd=radek_strom-a;
                   a2=2^a;
                   sl=ceil(sloupec_strom/a2);
                   sl_nasled=ceil(2*sloupec_strom/a2);
                   if sloupec==Strom_sl(rd,sl)
                      Cesta_zpet=cat(2,Cesta_zpet,[Strom_rd(rd,sl);
                                        sl_nasled/2==ceil(sl_nasled/2)]);
                   end
                end

                Cesta_zpet(1,:)=Cesta_zpet(1,:)+Cesta_zpet(2,:);

%Zacatek-index zacatku dat Pracovni matice pro dany sloupec a radek
%stromu
%Konec - index konce dat 'Pracovni' matice
                Zacatek(sloupec)=max(Cesta_zpet(1,:).*Cesta_zpet(2,:));
                Konec(sloupec)=min(not(Cesta_zpet(2,:)).*
                                  Cesta_zpet(1,:)+2*rd_X*Cesta_zpet(2,:));
             else  % viz. podminka - if radek_strom>1
                %Tento pripad nastava pri deleni korene stromu
                Zacatek(sloupec)=1;
                Konec(sloupec)=rd_X;
             end   % viz. podminka - if radek_strom>1

%Pomocne zaplneni velkymi cisly - hleda se minimum z matice 'G'
             G(1:rd_X-1,sloupec)=1e5;

%Cyklus pro urceni vetveni stromu pro prave pocitany sloupec 'Pracovni'
%matice.V tomto cyklu se generuje matice 'G', ktera obsahuje hodnoty
%deliciho kriteria.
             for i=Zacatek(sloupec):Konec(sloupec)-1
                pom=zeros(poc_trid,2);
                for j=Zacatek(sloupec):i
                   if Pracovni(j,sl_X)>0
                      pom(Pracovni(j,sl_X),1)=pom(Pracovni(j,sl_X),1)+1;
                   end
                end
                for k=i+1:Konec(sloupec)
                   if Pracovni(k,sl_X)>0
                      pom(Pracovni(k,sl_X),2)=pom(Pracovni(k,sl_X),2)+1;
                   end
                end
                N=sum(pom);
                if ~((N(1)==0)|(N(2)==0))
                   pom_p=~pom+pom;
                   G(i,sloupec)=abs(sum(pom(:,1).*log(pom_p(:,1)/N(1)))
                                   +sum(pom(:,2).*log(pom_p(:,2)/N(2))));
                end
             end
          end

%Nalezeni indexu 'del_rd' a 'del_sl' minimalni hodnoty s matice 'G'
          [del_rd,del_sl]=find(G==min(min(G)));
          %Pokud jich je vice, bereme prvni nalezenou hodnotu
          del_rd=del_rd(1); del_sl=del_sl(1);
          Pracovni=sortrows(Pracovni,del_sl);
```

```
%Vypocet pomocne hodnoty 'posun' pro preskoceni nulovych hodnot
%ve sloupci matice 'Pracovni', pro ziskani spravneho deliciho koeficintu.
%Nulove hodnoty v poslednim sloupci znamenaji jiz zatrideny radek
         posun=1;
         while Pracovni(del_rd+posun,sl_X)==0
            posun=posun+1;
         end
%Vypocet deliciho koeficientu.
         del_koef=mean([Pracovni(del_rd,del_sl)
                                        Pracovni(del_rd+posun,del_sl)]);
%Delici koeficient stromu pro dany list. 'Strom' je vystup funkce
         Strom(radek_strom,sloupec_strom)=del_koef;
%Ulozeni pomocnych hodnot. Pomocne hodnoty pro dalsi deleni stromu.
%'del_sl' - index sloupce-tez vystup funkce 'genstrom',
%'del_rd' - index radku -kde bylo provedeno deleni
         Strom_sl(radek_strom,sloupec_strom)=del_sl;
         Strom_rd(radek_strom,sloupec_strom)=del_rd;

%V tomto bloku se zjistuje hodnota klasifikace
         pom=zeros(poc_trid,2);
         for j=Zacatek(del_sl):del_rd
            if Pracovni(j,sl_X)>0
               pom(Pracovni(j,sl_X),1)=pom(Pracovni(j,sl_X),1)+1;
            end
         end
         for k=del_rd+1:Konec(del_sl)
            if Pracovni(k,sl_X)>0
               pom(Pracovni(k,sl_X),2)=pom(Pracovni(k,sl_X),2)+1;
            end
         end

         klas=max(pom);
         klas_n=(klas./sum(pom));
         trida_l=find(klas(1)==pom(:,1));
         trida_s=find(klas(2)==pom(:,2));

%Hodnota klasifikace 'Strom_klas' se pohybuje v intervalu <0,1).
%Pokud deleni stromu bylo takove, ze vsechny hodnoty jsou spravne
%zaklasifikovany,hodnota klasifikace je rovna 1
%V matici 'Strom_trida' se ukladaji cisla klasifikovane tridy pro dany
%list (jen pokud je hodnota klasifikace > 1-MAXMISKLAS)

         Strom_klas(radek_strom+1,2*sloupec_strom-1)=klas_n(1);
         Strom_klas(radek_strom+1,2*sloupec_strom)=klas_n(2);

         Strom_trida(radek_strom+1,2*sloupec_strom-1)=(klas_n(1)
                                           >=(1-maxm))*trida_l(1);
         Strom_trida(radek_strom+1,2*sloupec_strom)=(klas_n(2)
                                           >=(1-maxm))*trida_s(1);
%Zde se odeberou (cisla trid se nahradi 0) jiz zaklasifikovane radky
%'Pracovni' matice.
         if Strom_klas(radek_strom+1,2*sloupec_strom-1)==1
            for j=Zacatek(del_sl):del_rd
               Pracovni(j,sl_X)=0;
            end
         end
         if Strom_klas(radek_strom+1,2*sloupec_strom)==1
            for j=del_rd+1:Konec(del_sl)
               Pracovni(j,sl_X)=0;
            end
         end
```

```
%V pripade, ze je vse zaklasifikovano (soucet sloupce trid je roven 0)
%nebo max. misklasifikace je mensi nez pozadovana, je
%mozno vetveni ukoncit (tim i celou funkci -strom je vygenerovan)
         max_misklas=1-min(Strom_klas(radek_strom+1,:));
         if (sum(Pracovni(:,sl_X))==0)|((max_misklas<=maxm)
                              &(sloupec_strom==2^(radek_strom-1)))
            konec=1;
            break
         end
      else
%Dalsi vetveni v teto vetvi stromu neni nutne, vse je jiz
%zaklasifikovano, zapisuji se hodnoty z rodicovskeho uzlu.
         Strom_klas(radek_strom+1,2*sloupec_strom-1)=1;
         Strom_klas(radek_strom+1,2*sloupec_strom)=1;
         Strom_trida(radek_strom+1,2*sloupec_strom-1)
                              =Strom_trida(radek_strom,sloupec_strom);
         Strom_trida(radek_strom+1,2*sloupec_strom)
                              =Strom_trida(radek_strom,sloupec_strom);
      end
   end %konec cyklu promenne sloupec_strom
   if konec==1
      %formovani vysledku
      [smr,sms]=size(Strom_klas);
      [sr,ss]=size(Strom);
      sp=zeros(smr,sms);
      sp(1:sr,1:ss)=Strom;
      spsl=zeros(smr,sms);
      spsl(1:sr,1:ss)=Strom_sl;
      Strom_vys(:,:,1)=sp;
      Strom_vys(:,:,2)=spsl;
      Strom_vys(:,:,3)=Strom_trida;
      Strom_vys(:,:,4)=Strom_klas;
      break
   end
 end
```

Zdrojový výpis programu funkce *klasifikuj.m*

```
%[DATA_OUT]=klasifikuj(STROM,DATA_IN) - Klasifikuje data DATA_IN
%  podle vytvoreneho klasifikacniho stromu STROM funkci GENSTROM.
%  Vstupni data DATA_IN obsahuji charakteristiky obrazu. Je nutne%
%  zachovat stejne poradi charakteristik (sloupcu)jako u trenovacich
%  dat pro funkci GENSTROM. Vystupem teto funkce je pak matice
%  DATA_OUT ve ktere je prvnich n sloupcu schodnych z matici DATA_IN
%  a je pridan sloupec n+1, kde jsou cisla trid do kterych byl
%  dany radek zaklasifikovan.

function [DATA_OUT]=klasifikuj(Strom,DATA_IN)
   [r s]=size(DATA_IN);
   v_strom=size(Strom);
   %vytvoreni vystupni matice
   DATA_OUT=zeros(r,s+1);
   DATA_OUT(1:r,1:s)=DATA_IN;
   %hlavni cyklus pro projizdeni radku vstupni matice
   for radek=1:r
      radek_strom=1;
      sloupec_strom=1;
      %cyklus projizdi strom, dokud neni objekt zaklasifikovan
      %zaroven se porovnavaji vstupni charakteristiky s delicimi
      %hodnotami stromu
      while (Strom(radek_strom,sloupec_strom,4)<1)&
                                          (radek_strom<v_strom(1))
         if Strom(radek_strom,sloupec_strom,1)
                     <=DATA_IN(radek,Strom(radek_strom,sloupec_strom,2))
            sloupec_strom=sloupec_strom*2;
         else
            sloupec_strom=sloupec_strom*2-1;
         end
         radek_strom=radek_strom+1;
      end

%zapsat cislo tridy do posledniho radku vystupni matice pro dany radek

      DATA_OUT(radek,s+1)=Strom(radek_strom,sloupec_strom,3);

   end
```

Příloha B

Zdrojové výpisy programů funkcí pro výpočet charakteristik prvního řádu

Funkce *stredhod.m* - výpočet odhadu střední úrovně šedi

```
%[S]=stredhod(X) - Vraci odhad stredni urovne sedi S
%                  X je matice obrazku
function [strh]=stredhod(X)
      strh=mean(mean(X));
```

Funkce *rozptyl.m* - výpočet odhadu rozptylu úrovně šedi

```
%[R]=rozptyl(X) - Vraci odhad rozptylu urovne sedi R
%                 X - je matice obrazku
function [roz]=rozptyl(X)
   strh=mean(mean(X));
   [a,b]=imhist(uint8(X));
   b=b-1; n=sum(a); p=a/n;
   roz=sum(((b-strh).^2).*p);
```

Funkce *varkoef.m* - výpočet odhadu variačního koeficientu

```
%[VK]=varkoef(X) - Vraci hodnotu variacniho koeficientu VK
%                  X - je matice obrazku
function [vark]=varkoef(X)
   strh=mean(mean(X));
   [a,b]=imhist(uint8(X));
   b=b-1; n=sum(a); p=a/n;
   roz=sqrt(sum(((b-strh).^2).*p));
   vark=roz/strh;
```

Funkce *vybsikm.m* - výpočet výběrové šikmosti úrovně šedi

```
%[VS]=vybsikm(X) - Vraci vyberovou sikmost VS urovne sedi%
%  X - je matice obrazku
function [sik]=vybsikm(X)
   strh=mean(mean(X));
   [a,b]=imhist(uint8(X));
   b=b-1; n=sum(a); p=a/n;
   roz=sum(((b-strh).^2).*p);
   sik=sum(((b-strh).^3).*p)/roz^(3/2);
```

Funkce *vybspic.m* - výpočet výběrové špičatosti úrovně šedi

```
%[VB]=vybspic(X) - Vraci vyberovou spicatost VB urovne sedi%
%  X - je matice obrazku
function [spic]=vybspic(X)
   strh=mean(mean(X));
   [a,b]=imhist(uint8(X));
   b=b-1; n=sum(a); p=a/n;
```

```
   roz=sum(((b-strh).^2).*p);
   spic=sum(((b-strh).^4).*p-3)/roz^2;
```

Funkce *energ.m* - výpočet energie úrovně šedi

```
%[E]=energ(X) - Vraci hodnotu energie urovne sedi E%
%  X - je matice obrazku
function [energ]=energ(X)
   strh=mean(mean(X));
   [a,b]=imhist(uint8(X));
   n=sum(a); p=a/n;
   energ=sum(p.^2);
```

Funkce *entrop.m* - výpočet entropie úrovně šedi

```
%[EN]=entrop(X) - Vraci hodnotu entropie EN urovne sedi%
%     X - je matice obrazku
function [entr]=entrop(X)
   strh=mean(mean(X));
   [a,b]=imhist(uint8(X));
   n=sum(a); p=a/n; ppom=p==0;
   entr=-sum(p.*log(p+ppom));
```

Zdrojové výpisy programů funkcí pro výpočet charakteristik druhého řádu

Funkce *kookur.m* - výpočet normalizované kookurenční matice

```
%[K,POC]=kookur(X,R,N) - Vraci normalizovanou kookurencni matici K
%  pro dane posunuti R a celkovy pocet dvojic bodu POC
%  X - matice obrazku
%  R - vektor posunuti napr.: [0 1]
%  N - pocet urovni sedi v obraze (nepovinny parametr, standartne N=256)
function [C,poc]=kookur(X,r,n)
   Xd=double(X);
   if nargin==2
      v=256;
   else
      v=n;
   end
   if v<max(max(Xd))+1
      error('Obraz obsahuje vice urovni sedi, nez bylo zadano')
   end
   C=zeros(v,v);
   [rd,sl]=size(Xd);
   zaci=1+(r(2)>0);
   koni=rd-(r(2)<0);
   zacj=1+(r(1)<0);
   konj=sl-(r(1)>0);

   for i = zaci:koni
      for j = zacj:konj
       k = Xd(i,j)+1;
       l = Xd(i-r(2),j+r(1))+1;
       C(k,l) = C(k,l) + 1;
```

```
      end
   end
   poc=sum(sum(C));
   C=C/poc;
```

Funkce *kookurabs.m* - výpočet nenormalizované (absolutní) kookurenční matice

```
%[K]=kookurabs(X,R,N) - Vraci kookurencni matici K (absolutni hodnoty
vyskytu)
%  pro dane posunuti R
%  X - matice obrazku
%  R - vektor posunuti napr.: [0 1]
%  N - pocet urovni sedi v obraze (nepovinny parametr, standartne N=256)

function [C]=kookurabs(X,r)
   if nargin==2
      v=256;
   else
      v=n;
   end
   if v<max(max(X))+1
      error('Obraz obsahuje vice urovni sedi, nez bylo zadano')
   end
   Xd=double(X);
   C=zeros(v,v);
   [rd,sl]=size(Xd);
   zaci=1+(r(2)>0);
   koni=rd-(r(2)<0);
   zacj=1+(r(1)<0);
   konj=sl-(r(1)>0);

   for i = zaci:koni
      for j = zacj:konj
       k = Xd(i,j)+1;
       l = Xd(i-r(2),j+r(1))+1;
       C(k,l) = C(k,l) + 1;
      end
   end
```

Funkce *energkoo.m* - výpočet energie kookurenční matice (druhý angulární moment)

```
%[EK]=energkoo(K) - Vraci hodnotu energie EK normalizovane kookurencni
matice
%  K - norm. kookurencni matice
function [energ]=energkoo(K)
   [r s]=size(K);
   if (r~=s)|(abs(1-sum(sum(K)))>0.01)
      error('Chyba - kookurencni matice neni ctvercova nebo neni
                                                        normalizovana')
   end
   energ=sum(sum(K.^2));
```

Funkce *entrkoo.m* - výpočet entropie kookurenční matice

```
%[ENK]=entrkoo(K) - Vraci hodnotu entropie ENK norm. kookurencni matice
%  K - norm. kookurencni matice
function [entr]=entrkoo(K)
   [r s]=size(K);
   if (r~=s)|(abs(1-sum(sum(K)))>0.01)
      error('Chyba - kookurencni matice neni ctvercova nebo neni
                                                    normalizovana')
   end
   Kpom=K==0;
   entr=-sum(sum(K.*log(K+Kpom)));
```

Funkce *maxpr.m* - výpočet maximální pravděpodobnosti

```
%[MAXP]=maxpr(K) - Vraci hodnotu maximalni pravdepodobnosti MAXP
%  normalizovane kookurencni matice
%  K - norm. kookurencni matice
function [mx]=maxpr(K)
   [r s]=size(K);
   if (r~=s)|(abs(1-sum(sum(K)))>0.01)
      error('Chyba - kookurencni matice neni ctvercova nebo neni
                                                    normalizovana')
   end
   mx=max(max(K));
```

Funkce *autokorel.m* - výpočet autokorelace

```
%[KOR]=autokorel(K) - Vraci hodnotu autokorelace normalizovane
kookurencni matice
%  K - norm. kookurencni matice
function [kor]=autokorel(K)
   [r s]=size(K);
   if (r~=s)|(abs(1-sum(sum(K)))>0.01)
      error('Chyba - kookurencni matice neni ctvercova nebo neni
                                                    normalizovana')
   end
   hod=0:r-1;
   Cx=sum(K');
   Cy=sum(K);
   str_Cx=sum(Cx.*hod);
   str_Cy=sum  (Cy.*hod);
   rozpt_Cx=sum((hod-str_Cx).^2.*Cx);
   rozpt_Cy=sum((hod-str_Cy).^2.*Cy);
   jm=sqrt(rozpt_Cx*rozpt_Cy);
   kor=sum(sum((hod-str_Cx)'*(hod-str_Cy).*K))/jm;
```

Funkce *diagmom.m* - výpočet diagonálního momentu

```
%[DM]=diagmom(K) - Vraci hodnotu diagonalniho momentu DM normalizovane
kookurencni matice
%  K - norm. kookurencni matice
function [dm]=diagmom(K)
   [r s]=size(K);
   if (r~=s)|(abs(1-sum(sum(K)))>0.01)
```

```
      error('Chyba - kookurencni matice neni ctvercova nebo neni
                                                        normalizovana')
   end
   J=ones(r,1);
   P=0:r-1;
   A=abs(J*P-(J*P)');
   B=J*P+(J*P)';
   str_Cx=sum(sum(K').*P);
   str_Cy=sum(sum(K).*P);
   S=zeros(r,r);
   S(:,:)=str_Cx+str_Cy;
   dm=sum(sum(A.*(B-S).*K));
```

Funkce *rozroz.m* - výpočet vektoru rozdělení abs. hodnot rozdílů úrovní šedi dvojic bodů

```
%[Dk]=rozroz(K) - Vraci radkovy vektor Dk odhadu rozdeleni absolutnich
%  hodnot rozdilu urovni sedi
%  K - norm. kookurencni matice
function [Dk]=rozroz(K)
   [r s]=size(K);
   if (r~=s)|(abs(1-sum(sum(K)))>0.01)
      error('Chyba - kookurencni matice neni ctvercova nebo neni
                                                        normalizovana')
   end
   J=ones(r,1);
   A=0:r-1;
   H=J*A+(J*A)'+1;
   for i=0:r-1
      Indexy1(i+1,:)=(H(i+1,:)-1)*r+H(1,:);
      Indexy2(i+1,:)=(H(1,:)-1)*r+H(i+1,:);
   end
   J=fliplr(triu(ones(r)));
   Indexy2(1,:)=0;
   Indexy=[Indexy1.*J Indexy2.*J];
   Indexy=(Indexy+(~Indexy*(r*s+2)))';
   K(:,s+1)=0;
   Dk=sum(K(Indexy));
```

Funkce *enerdk.m* - výpočet energie rozdělení abs. rozdílů úrovní šedi

```
%[EDk]=enerdk(Dk) - Vraci hodnotu energie EDk rozdeleni absolutnich
%  rozdilu urovni sedi Dk
%  Dk - je vektor rozdeleni abs. rozdilu urovni sedi
function [energ]=enerdk(Dk)
   [r s]=size(Dk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
   energ=sum(Dk.^2);
```

Funkce *entrdk.m* - výpočet entropie rozdělení abs. rozdílů úrovní šedi

```
%[ENDk]=entrdk(Dk) - Vraci hodnotu entropie ENDk rozdeleni absolutnich
%  rozdilu urovni sedi Dk
%  Dk - je vektor rozdeleni abs. rozdilu urovni sedi
function [entr]=entrdk(Dk)
   [r s]=size(Dk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
   Dkpom=Dk==0;
   entr=-sum(Dk.*log(Dk+Dkpom));
```

Funkce *variog.m* - výpočet variogramu (inerce, kontrast)

```
%[VA]=variog(Dk) - Vraci hodnotu variogramu VA rozdeleni absolutnich
%  rozdilu urovni sedi Dk
%  Dk - je vektor rozdeleni abs. rozdilu urovni sedi
function [vari]=variog(Dk)
   [r s]=size(Dk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
   k=0:s-1;
   vari=sum((k.^2).*Dk);
```

Funkce *lokhom.m* - výpočet lokální homogenity

```
%[LH]=lokhom(Dk) - Vraci hodnotu lokalni homogenty LH rozdeleni
absolutnich
%  rozdilu urovni sedi Dk
%  Dk - je vektor rozdeleni abs. rozdilu urovni sedi
function [lh]=lokhom(Dk)
   [r s]=size(Dk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
   k=0:s-1;
   lh=sum(Dk./(1+k.^2));
```

Funkce *rrozdk.m* - výpočet rozdílového rozptylu

```
%[RD]=rrozdk(Dk) - Vraci hodnotu rozdiloveho rozptylu RD rozdeleni
absolutnich
%  rozdilu urovni sedi Dk
%  Dk - je vektor rozdeleni abs. rozdilu urovni sedi
function [rd]=rrozdk(Dk)
   [r s]=size(Dk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
   k=0:s-1;
   vari=sum((k.^2).*Dk);
   rd=vari-(sum(k.*Dk))^2;
```

Funkce *rozsouc.m* - výpočet vektoru rozdělení součtů hodnot úrovní šedi dvojic bodů

```
%[Sk]=rozsouc(K) - Vraci radkovy vektor Sk odhadu rozdeleni
%  souctu hodnot urovni sedi dvojic bodu
%  K - norm. kookurencni matice
function [Sk]=rozsouc(K)
   [r s]=size(K);
   if (r~=s)|(abs(1-sum(sum(K)))>0.01)
      error('Chyba - kookurencni matice neni ctvercova nebo neni
                                                    normalizovana')
   end
   n=2*r-1;
   N=zeros(n,s);
   for i=1:r
      N(i:r,i)=K(1:r+1-i,i);
      N(r+1:n+1-i,i)=K(r+1-i,1+i:s)';
   end
   Sk=sum(N');
```

Funkce *enersk.m* - výpočet energie rozdělení součtů úrovní šedi

```
%[ESk]=enersk(Sk) - Vraci hodnotu energie ESk rozdeleni
%  souctu hodnot urovni sedi Sk
%  Sk - je vektor rozdeleni souctu hodnot urovni sedi
function [energ]=enersk(Sk)
   [r s]=size(Sk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
   energ=sum(Sk.^2);
```

Funkce *entrsk.m* - výpočet entropie rozdělení součtů úrovní šedi

```
%[ENSk]=entrsk(Sk) - Vraci hodnotu entropie ENSk rozdeleni
%  souctu hodnot urovni sedi Sk
%  Sk - je vektor rozdeleni souctu hodnot urovni sedi
function [entr]=entrsk(Sk)
   [r s]=size(Sk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
   Skpom=Sk==0;
   entr=-sum(Sk.*log(Sk+Skpom));
```

Funkce *rrozsk.m* - výpočet odhadu rozptylu rozdělení součtů úrovní šedi

```
%[RRSk]=rrozsk(Sk) - Vraci hodnotu RRSk - odhad rozptylu rozdeleni
%  souctu urovni sedi Sk
%  Sk - je vektor rozdeleni souctu hodnot urovni sedi
function [rrsk]=rrozsk(Sk)
   [r s]=size(Sk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
```

```
   k=0:s-1;
   Sa=sum(k.*Sk);
   rrsk=sum((k-Sa).^2.*Sk);
```

Funkce *m3rozsk.m* - výpočet odhadu třetího momentu rozdělení součtů úrovní šedi

```
%[M3R]=m3rozsk(Sk) - Vraci hodnotu M3R - odhad tretiho momentu rozdeleni
%  souctu urovni sedi Sk
%  Sk - je vektor rozdeleni souctu hodnot urovni sedi
function [m3r]=m3rozsk(Sk)
   [r s]=size(Sk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
   k=0:s-1;
   Sa=sum(k.*Sk);
   m3r=sum((k-Sa).^3.*Sk);
```

Funkce *m4rozsk.m* - výpočet odhadu čtvrtého momentu rozdělení součtů úrovní šedi

```
%[M4R]=m4rozsk(Sk) - Vraci hodnotu M4R - odhad ctvrteho momentu rozdeleni
%  souctu urovni sedi Sk
%  Sk - je vektor rozdeleni souctu hodnot urovni sedi
function [m4r]=m4rozsk(Sk)
   [r s]=size(Sk);
   if (r~=1)
      error('Chyba - vstupni vektor musi byt radkovy')
   end
   k=0:s-1;
   Sa=sum(k.*Sk);
   m4r=sum((k-Sa).^4.*Sk);
```

Zdrojový výpis programu funkce *ekvaliz.m* pro předzpracování obrazu

```
%[I2]=ekvaliz(I) - Vytvori novou matici I2 ekvalizovaneho obrazu.
%  I - puvodni matice obrazu
function [I2]=ekvaliz(I)
   [H,B]=imhist(uint8(I));
   N=1/sum(H);
   I=double(I);
   [r s]=size(I);
   for i=1:r
      for j=1:s
         I2(i,j)=uint8(round(255*(N*sum(H(1:I(i,j))))));
      end
   end
```

Příloha C

Zdrojový výpis programu příkladu 1

```
clear all;
close all;
format short g;
map=([0:1/255:1;0:1/255:1;0:1/255:1])';

I(:,:,1)=imread('c:\diplomka\textury1\tkanina1-D104.tif');
I(:,:,2)=imread('c:\diplomka\textury1\tkanina2-D20.tif');
I(:,:,3)=imread('c:\diplomka\textury1\tkanina3-D56.tif');
I(:,:,4)=imread('c:\diplomka\textury1\tkanina4-D65.tif');
I(:,:,5)=imread('c:\diplomka\textury1\tkanina5-D85.tif');
I(:,:,6)=imread('c:\diplomka\textury1\kaminky-D23.tif');

vo=80;
okno=zeros(vo,vo,9,6);

%Vyber oken pro vypocet charakteristik trenovacich dat
for o=1:6

   [r,s]=size(I(:,:,o));
   rd=floor((r-3*vo)/3);
   rd2=floor((r-3*vo)/6);
   sd=floor((r-3*vo)/3);
   sd2=floor((r-3*vo)/6);

   k=1;
   for i=0:2
      for j=0:2
         okno(:,:,k,o)=I(i*(rd+vo)+rd2+1:i*(rd+vo)+rd2+vo,j
                                    *(sd+vo)+sd2+1:j*(sd+vo)+sd2+vo,o);
         I(i*(rd+vo)+rd2+1:i*(rd+vo)+rd2+vo,j
                                    *(sd+vo)+sd2+1:j*(sd+vo)+sd2+vo)=25*k;
         k=k+1;
      end
   end
end

clear I k
for o=1:6
   for w=1:9
      X1=stredhod(okno(:,:,w,o));
      X2=vybspic(okno(:,:,w,o));
      X3=vybsikm(okno(:,:,w,o));
      X4=energ(okno(:,:,w,o));

      K=0.5*(kookur(okno(:,:,w,o),[1 0])+kookur(okno(:,:,w,o),[1 -1]));
         X5=energkoo(K);
         X6=autokorel(K);
         X7=maxpr(K);

      Dk=rozroz(K);
         X8=enerdk(Dk);
         X9=variog(Dk);

      Sk=rozsouc(K);
         X10=enersk(Sk);
      trenovaci((o-1)*9+w,:)=[X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 o];
```

```
   end
end

%vygenerovani stromu na zaklade matice trenovaci
Strom=genstrom(trenovaci,0,16);

%vytvoreni matice klas pro naslednou klasifikaci (nahodne velikosti oken)
b=23;
for o=1:6
   for w=1:b
      vo=round(40+80*rand);
      okno2=zeros(vo,vo);
      rn=round((640-vo)*rand);
      sn=round((640-vo)*rand);
      okno2=I(1+rn:rn+vo,1+sn:sn+vo,o);

%      X1=stredhod(okno2);
       X2=vybspic(okno2);
%      X3=vybsikm(okno2);
%      X4=energ(okno2);

      K=0.5*(kookur(okno2,[1 0])+kookur(okno2,[1 -1]));
%        X5=energkoo(K);
         X6=autokorel(K);
%        X7=maxpr(K);

%      Dk=rozroz(K);
%        X8=enerdk(Dk);
%        X9=variog(Dk);

%      Sk=rozsouc(K);
%        X10=enersk(Sk);


      Klas((o-1)*b+w,:)=[0 X2 0 0 0 X6 0 0 0 0];
   end
end

Klasifikovano=klasifikuj(Strom,Klas);

Strom
Klasifikovano
```

Tabulka hodnot trénovacích dat z příkladu 1

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	třída
Obrázek 16a	133,35	1,206	-0,178	0,054	0,0094	0,731	0,045	0,054	4487,9	0,0161	1
	132,43	1,206	-0,151	0,054	0,0089	0,717	0,034	0,053	4694,0	0,0159	1
	122,55	1,179	-0,021	0,057	0,0112	0,760	0,050	0,062	3789,1	0,0173	1
	121,31	1,190	0,020	0,049	0,0074	0,744	0,031	0,050	4171,3	0,0135	1
	137,17	1,227	-0,216	0,057	0,0104	0,755	0,036	0,058	3918,7	0,0180	1
	148,13	1,389	-0,426	0,056	0,0104	0,753	0,043	0,057	3542,9	0,0186	1
	132,96	1,209	-0,157	0,056	0,0096	0,731	0,042	0,052	4119,7	0,0167	1
	127,50	1,179	-0,061	0,065	0,0122	0,763	0,048	0,065	3879,7	0,0203	1
	129,06	1,200	-0,100	0,052	0,0109	0,763	0,047	0,057	3508,4	0,0168	1
Obrázek 16b	109,70	1,303	0,344	0,042	0,0096	0,848	0,069	0,059	2519,1	0,0161	2
	116,16	1,255	0,259	0,043	0,0109	0,853	0,081	0,055	2374,0	0,0168	2
	105,68	1,453	0,488	0,031	0,0045	0,837	0,029	0,042	2261,4	0,0097	2
	87,99	1,888	0,804	0,038	0,0049	0,841	0,027	0,049	2215,0	0,0120	2
	98,83	1,606	0,611	0,035	0,0049	0,843	0,029	0,047	2264,8	0,0111	2
	111,83	1,327	0,359	0,037	0,0066	0,844	0,045	0,046	2146,6	0,0124	2
	106,52	1,317	0,389	0,045	0,0075	0,860	0,039	0,059	2244,6	0,0153	2
	102,92	1,412	0,465	0,042	0,0066	0,838	0,039	0,050	2449,3	0,0139	2
	98,07	1,624	0,626	0,034	0,0056	0,854	0,042	0,046	1875,7	0,0112	2
Obrázek 16c	143,09	1,681	-0,116	0,017	0,0014	0,902	0,010	0,040	713,3	0,0041	3
	121,18	1,863	0,164	0,015	0,0010	0,901	0,006	0,038	609,9	0,0036	3
	125,48	1,776	0,143	0,016	0,0012	0,909	0,008	0,041	542,9	0,0038	3
	128,14	1,725	0,074	0,015	0,0010	0,915	0,008	0,039	614,8	0,0035	3
	115,22	1,839	0,205	0,015	0,0009	0,910	0,007	0,040	597,6	0,0036	3
	111,97	1,829	0,225	0,014	0,0010	0,919	0,007	0,045	556,8	0,0035	3
	129,71	1,790	0,120	0,015	0,0010	0,898	0,007	0,039	680,3	0,0035	3
	122,60	1,873	0,128	0,014	0,0009	0,893	0,007	0,036	710,0	0,0033	3
	124,79	1,713	0,043	0,016	0,0011	0,912	0,008	0,041	601,1	0,0037	3
Obrázek 16d	68,61	5,665	1,791	0,025	0,0020	0,785	0,016	0,046	1138,0	0,0067	4
	80,54	4,825	1,555	0,022	0,0014	0,783	0,012	0,038	1126,1	0,0056	4
	86,90	4,630	1,481	0,022	0,0015	0,772	0,016	0,038	1136,7	0,0055	4
	64,15	6,154	1,915	0,025	0,0019	0,785	0,016	0,046	1073,2	0,0071	4
	80,71	4,955	1,607	0,022	0,0014	0,785	0,013	0,039	1082,9	0,0057	4
	81,27	5,672	1,755	0,025	0,0019	0,793	0,016	0,042	930,3	0,0065	4
	61,52	6,048	1,916	0,029	0,0028	0,795	0,022	0,053	1091,5	0,0084	4
	84,64	3,908	1,367	0,021	0,0014	0,800	0,012	0,040	1180,5	0,0054	4
	88,42	4,686	1,475	0,023	0,0017	0,790	0,018	0,040	953,6	0,0060	4
Obrázek 16e	136,11	1,710	-0,099	0,021	0,0007	0,505	0,008	0,015	2313,0	0,0040	5
	135,68	1,753	-0,046	0,019	0,0006	0,495	0,004	0,013	2478,0	0,0038	5
	164,03	1,994	-0,596	0,027	0,0013	0,534	0,012	0,017	2259,8	0,0050	5
	127,48	1,870	0,152	0,018	0,0006	0,536	0,007	0,014	2135,5	0,0039	5
	140,03	1,821	-0,116	0,021	0,0007	0,498	0,006	0,014	2209,7	0,0041	5
	169,51	2,240	-0,673	0,028	0,0015	0,527	0,012	0,019	2003,2	0,0053	5
	144,41	1,861	-0,210	0,020	0,0008	0,591	0,006	0,016	1947,2	0,0040	5
	159,14	2,039	-0,470	0,024	0,0010	0,537	0,006	0,017	1966,1	0,0045	5
	163,95	2,198	-0,618	0,025	0,0012	0,558	0,008	0,018	1944,2	0,0048	5
Obrázek 16f	99,03	2,131	0,403	0,015	0,0016	0,935	0,014	0,048	493,2	0,0043	6
	115,88	2,256	0,428	0,016	0,0011	0,910	0,014	0,040	524,6	0,0038	6
	114,58	1,783	0,282	0,014	0,0012	0,939	0,008	0,045	487,3	0,0036	6
	74,10	3,410	1,290	0,024	0,0025	0,938	0,018	0,055	468,2	0,0072	6
	144,00	1,681	-0,273	0,020	0,0023	0,932	0,016	0,040	611,1	0,0051	6
	126,02	1,746	0,209	0,015	0,0011	0,935	0,007	0,040	473,0	0,0036	6
	95,82	2,453	0,467	0,015	0,0013	0,919	0,013	0,046	496,2	0,0040	6
	119,69	1,613	0,084	0,015	0,0013	0,926	0,009	0,040	666,7	0,0037	6
	117,85	1,900	0,330	0,016	0,0014	0,944	0,013	0,048	367,7	0,0040	6

Příloha D

Zdrojový výpis programu příkladu 2

```
clear all;
close all;
format short g;
map=([0:1/255:1;0:1/255:1;0:1/255:1])';

%I=imread('c:\diplomka\textury2\vada288dpi.tif');
%I=ekvaliz(I);
%imwrite(I,map,'c:\diplomka\textury2\vada288dpi_ekv.tif',
%                                        'tif','resolution',[288 288]);

I=imread('c:\diplomka\textury2\vada288dpi_ekv.tif');
vo=100;
%matice ramecku, tl-tloustka v bodech
tl=2;
ramck=zeros(vo,vo);
ramck(1:tl,:)=1;
ramck(vo-tl+1:vo,:)=1;
ramck(:,1:tl)=1;
ramck(:,vo-tl+1:vo)=1;

%vektory souradnic oken bez poruch (ro-radek, so-sloupec)
ro=[1 3 1 3  1 5 6 5  5 9 8 9 8  9  9 9 8 6 ];
so=[1 3 6 9 12 1 3 7 10 1 3 6 9 12 10 5 6 3 ];
[a,b]=size(ro);
oknodobre=zeros(vo,vo,b);

%vyber oken pro trenovaci data - oknodobre
for t=1:b
    oknodobre(:,:,t)=double(I(((ro(t)-1)*vo)+1:
                              ro(t)*vo,((so(t)-1)*vo)+1:so(t)*vo));
    g=~ramck.*oknodobre(:,:,t)+255*ramck;
    I(((ro(t)-1)*vo)+1:ro(t)*vo,((so(t)-1)*vo)+1:so(t)*vo)=uint8(g);
end

%vektory souradnic oken s poruchami (ro-radek, so-sloupec)
ro=[7 7 8 7  6  7];
so=[4 5 4 7 10 12];
[a,d]=size(ro);
oknochyba=zeros(vo,vo,d);

%vyber oken pro trenovaci data - oknochyba
for t=1:d
   oknochyba(:,:,t)=double(I(((ro(t)-1)*vo)+1+:
                              ro(t)*vo,((so(t)-1)*vo)+1:so(t)*vo));
   g=~ramck.*oknochyba(:,:,t);
   I(((ro(t)-1)*vo)+1:ro(t)*vo,((so(t)-1)*vo)+1:so(t)*vo)=uint8(g);
end

imwrite(I,map,'c:\diplomka\textury2\vada288dpi_ram.tif','tif');
clear I;

%Vypocet charakteristik pro trenovaci data - z oknodobre
for o=1:b
      X1=rozptyl(oknodobre(:,:,o));
      X2=vybspic(oknodobre(:,:,o));
      X3=vybsikm(oknodobre(:,:,o));
```

```
      K=0.5*(kookur(oknodobre(:,:,o),[1 1])+
                                 kookur(oknodobre(:,:,o),[1 -1]));
         X4=autokorel(K);
         X5=entrkoo(K);
         X6=energkoo(K);
         X7=maxpr(K);
      trenovaci(o,:)=[X1 X2 X3 X4 X5 X6 X7 1];
end

%Vypocet charakteristik pro trenovaci data - z oknochyba
for o=1:d
      X1=rozptyl(oknochyba(:,:,o));
      X2=vybspic(oknochyba(:,:,o));
      X3=vybsikm(oknochyba(:,:,o));

      K=0.5*(kookur(oknochyba(:,:,o),[1 1])+
                           kookur(oknochyba(:,:,o),[1 -1]));
        X4=autokorel(K);
        X5=entrkoo(K);
        X6=energkoo(K);
        X7=maxpr(K);
   trenovaci(b+o,:)=[X1 X2 X3 X4 X5 X6 X7 2];
end

clear oknodobre,oknochyba;
%Mame vytvorenou matici trenovaci
%Vytvorime strom

Strom2=genstrom(trenovaci,0,16);

%Dale vypocitame charakteristiky pro ostatni data, ktere budeme
klasifikovat

I=imread('c:\diplomka\textury2\vada288dpi_ekv.tif');
vo=100;
okno=zeros(vo,vo);

m=round(900/vo);
n=round(1200/vo);

for r=1:m
   for s=1:n
      okno=zeros(vo,vo);
      okno=double(I(((r-1)*vo)+1:r*vo,((s-1)*vo)+1:s*vo));
%Z predchoziho zkouseni vim, ze strom vybere charakteristiky X5, X6, X7
% ostatni tedy nemusim uvazovat
%       X1=rozptyl(okno);
%       X2=vybspic(okno);
%       X3=vybsikm(okno);

      K=0.5*(kookur(okno,[1 1])+kookur(okno,[1 -1]));
%        X4=autokorel(K);
         X5=entrkoo(K);
         X6=energkoo(K);
         X7=maxpr(K);
   Klas2((r-1)*n+s,:)=[0 0 0 0 X5 X6 X7];
   end
   r
end
%Mame vytvoreny data pro klasifikovani Klas2 a strom v Strom2
%Nezbyva uz nic jineho nez pouzit funkce klasifikuj
```

```
Klasifikovana=klasifikuj(Strom2,Klas2);

%..a vysledky klasifikace ulozime do vysledneho obrazku (ramecky)

[a,b]=size(Klasifikovana);
d=Klasifikovana(:,b);
d=reshape(d,12,9)';
for r=1:9
   for s=1:12
      if d(r,s)==2
         okno=zeros(vo,vo);
         okno=double(I(((r-1)*vo)+1:r*vo,((s-1)*vo)+1:s*vo));
         g=~ramck.*okno;
         I(((r-1)*vo)+1:r*vo,((s-1)*vo)+1:s*vo)=uint8(g);
      end
   end
   r
end

figure 1;
imshow(I);
%vysledek klasifikace ulozime do obrazku
imwrite(I,map,'c:\diplomka\textury2\vada2_kl.tif','tif');
```

Tabulka hodnot trénovacích dat z příkladu 2

		X1	X2	X3	X4	X5	X6	Třída
Hodnoty charakteristik pro okna	- bez poruch	2,040	0,413	0,711	8,269	3,315E-04	1,275E-03	1
		1,846	0,046	0,733	8,542	2,397E-04	1,020E-03	1
		1,990	-0,273	0,757	8,525	2,328E-04	7,142E-04	1
		1,788	0,152	0,735	8,648	2,135E-04	7,652E-04	1
		1,836	0,077	0,725	8,618	2,209E-04	7,652E-04	1
		1,949	0,211	0,690	8,489	2,575E-04	8,162E-04	1
		1,862	-0,075	0,691	8,686	2,054E-04	7,652E-04	1
		1,799	0,182	0,754	8,590	2,285E-04	8,673E-04	1
		1,799	0,267	0,718	8,678	2,121E-04	8,673E-04	1
		2,162	0,265	0,749	8,047	3,979E-04	1,275E-03	1
		1,951	-0,226	0,663	8,710	1,977E-04	7,652E-04	1
		1,824	-0,194	0,687	8,784	1,843E-04	1,122E-03	1
		1,697	0,034	0,701	8,799	1,808E-04	7,142E-04	1
		2,356	0,754	0,782	7,971	9,519E-04	1,342E-02	1
		1,774	0,338	0,734	8,617	2,603E-04	3,214E-03	1
		2,021	-0,373	0,673	8,724	2,010E-04	1,684E-03	1
		1,949	-0,365	0,697	8,722	2,098E-04	1,939E-03	1
		1,812	-0,129	0,666	8,506	1,806E-03	4,020E-02	1
	- s poruchami	1,961	-0,313	0,708	8,717	2,153E-04	3,469E-03	2
		2,114	-0,445	0,709	8,623	2,726E-04	5,714E-03	2
		2,051	-0,404	0,665	8,730	1,966E-04	1,326E-03	2
		1,600	-0,019	0,763	8,769	1,998E-04	1,990E-03	2
		1,748	0,314	0,750	8,647	2,583E-04	4,540E-03	2
		2,053	0,592	0,757	8,386	4,124E-04	5,306E-03	2