VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

EVALUACE TEXTURNÍCH PŘÍZNAKŮ

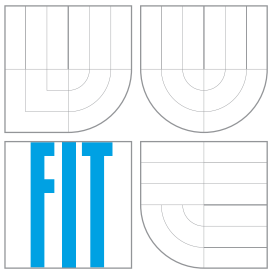
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR PRŮŠA

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

EVALUACE TEXTURNÍCH PŘÍZNAKŮ

EVALUATION OF TEXTURE FEATURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR PRŮŠA
AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ ŠILER
SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářská práce se zabývá statistickým popisem textur metodami LBP a maticemi souslednosti. Obsahem je také vyhodnocení matic souslednosti pomocí Haralickových příznaků. Implementace je v jazyce C++ a byla využita knihovna CImg.h

Abstract

The bachelor's thesis deals with statistical description of textures by methods LBP and co-occurence matrix. The co-occurence matrix is evalueted by Haralick's texture features. The implementation is in C++ language with using CImg.h library.

Klíčová slova

textura, příznak textury, popis textury, LBP, lokalní binární vzory, matice souslednosti, kookureční matice, histogram kookurence, Haralickovy příznaky.

Keywords

texture, texture feature, texture description, LBP, local binary patterns, co-occurence matrix, co-coherence matrix, Haralick texture feature

Citace

Petr Průša: Evaluace texturních příznaků, bakalářská práce, Brno, FIT VUT v Brně, 2009

Evaluace texturních příznaků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Ondřeje Šilera. Další informace mi poskytl pan Ing. Miroslav Švub. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Petr Průša
20.05.2009

Poděkování

Chtěl bych poděkovat mému vedoucímu Ing. Ondřeji Šilerovi za odbornou pomoc v průběhu celé práce. Dále děkuji panu Ing. Miroslavovi Švubovi za užitečné rady.

Obsah

Kapitola 1

Úvod

V posledních letech se především díky neustálému zvyšování výkonu počítačů do popředí zájmu informatiků dostává zpracování obrazu. Dnes již není problém zpracovat velký obraz v reálném čase a tím se tento obor informatiky stává v praxi velmi dobře využitelným. Proto se metody zpracování obrazu stále více uplatňují v medicíně, v dopravě, při odhalování vad výrobků, v klasifikaci atd. V medicíně se používají pro identifikování různých nádorů a poškození vnitřních orgánů, zatímco v dopravě se jedná hlavně o rozpoznání dopravních značek, vodorovných pruhů na silnici a předmětů pohybujících se v okolí vozovky. Tyto systémy pak pomáhají lékařům určit diagnózu pacienta a řidičům jsou nápomocné zejména v problémových situacích(špatná viditelnost, mikrospánek atp.). V klasifikaci se jedná o automatické zařazení obrazu do předem určených skupin zpravidla podle trénovacích dat.

Při zpracování obrazu se velice často pracuje s pojmem textura. Ač tento pojem nemá přesnou definici, je jedním ze základních stavebních kamenů při rozpoznávání objektů v obraze. Textury se používají, aby obraz vypadal více realisticky. Dodávají určité vlastnosti povrchu a struktuře objektů. Tedy textura bývá část obrazu s určitým barevným tónem a strukturou. Chceme-li rozpoznat nějaký objekt v obraze, je dobré zjistit, jaké textury se v něm vyskytují.

Pokud má textura výraznou prostorovou strukturu, pak ji lze popsat pomocí množiny opakujících se primitiv a jejich geometrickým uspořádáním. V případě, že textura má malá primitiva a různá prostorová uspořádání, pak je vhodné ji popsat statisticky. Zde se jedná zejména o získání histogramů specializovanými metodami a jejich následné porovnání. Histogramy lze porovnávat buď pomocí různých vzdáleností jednotlivých bodů, nebo spočítáním např. Haralickových příznaků.

Tato práce se zabývá evaluací(vyhodnocováním) textur na základě jejich statistického popisu. Pro tyto účely byly zvoleny metody LBP(viz 3.5) a matice souslednosti(viz 3.4). Pro vyhodnocení získaných matic byly použity Haralickovy příznaky(viz 3.4.1).

Kapitola 2 rozebírá teorii potřebnou k pochopení toho, co je to obraz a jak je na počítači reprezentován. Je zde tedy vysvětleno, jak obraz získáváme a jakým způsobem ho upravujeme pro práci na počítači. Dále v kapitole najdeme stručné pojednání o barevných modelech využívaných v číslicové technice.

kapitola 3 nás uvede do toho, co je to textura a k čemu se využívá. Dále zde je rozebráno dělení textur a způsoby jejich popisu. Nakonec tu jsou objasněny metody statistického popisu textur, které jsou předmětem této práce.

V kapitole 4 jsou sepsány základní požadavky na výsledný program a je zde nastíněn jeho návrh.

Pátá kapitola se zabývá tím, jak byl návrh uvedený v předchozí kapitole v praxi realizován. Je zde popsána knihovna, která byla využita k načtení obrázků. Následně je tu popsána implementace celého programu a princip jeho činnosti. Dále zde najdeme odstavec zabývající se testy a způsobem jejich vyhodnocování.

Kapitola šestá uvádí vzorek výsledků programu a popisuje na příkladech způsoby jejich vyhodnocování.

V kapitole sedmé je samotná práce zhodnocena a jsou v ní uvedeny návrhy na další vylepšení stávajícího programu.

Kapitola 2

Obraz

Zde bude vysvětlen pojem obraz. Jak je obraz definován matematicky,jak ho intuitivně vnímá člověk, jak jej získáváme, jak se zpracovává pro použití na počítači a jaké vlastnosti má reprezentovaný obraz na počítači.

2.1 Definice obrazu

Obraz je ve své podstatě signál, který lze matematicky popsat jako spojitou funkci. Tato funkce může být jednodimenzionální (tj. závislá na čase), dvoudimenzionální (tj. obraz popsán dvěma souřadnicemi v rovině), třídimenzionální(tj. těleso v prostoru) i vyšší. Skalární funkce může popisovat monochromatický obraz, zatímco vektorová funkce se používá pro reprezentaci barevných obrazů. Intuitivně je obraz to, co zachytí naše sítnice nebo třeba videokamera. Zpravidla je reprezentován spojitou 2D funkcí $f(x, y)$, kde x a y jsou souřadnice plochy. Někdy lze obraz pojmout jako 3D, v tomto případě je třetí souřadnicí čas. Více informací v [8].

2.2 Získání digitálního obrazu

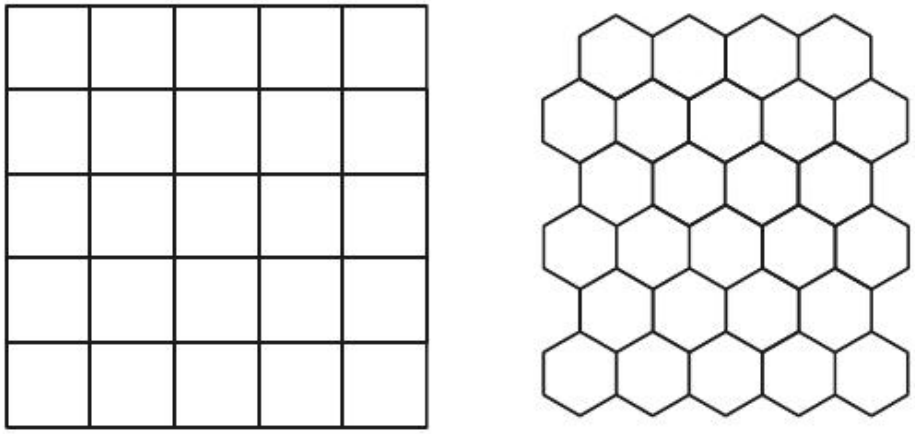
Jelikož je obraz spojitá funkce, kterou nejsme schopni na počítači reprezentovat, musíme ho zde popsat jinak. Proto je třeba obraz nejdříve převést na jeho diskrétní ekvivalent.

2.2.1 Digitalizace obrazu

Diskretizace obrazu z původní spojité funkce na funkci s diskrétními hodnotami se nazývá digitalizace obrazu. Spojitou funkci $f(x, y)$ tedy navzorkujeme na matici o velikosti M řádků a N sloupců a to tak, že spojité osy x a y rozdělíme na K intervalů. Čím větší K zvolíme, tím více se digitální obraz blíží původnímu spojitému obrazu. Větší K tedy zvyšuje kvalitu obrazu, avšak znamená také větší početní náročnost. Počet intervalů K se volí podle Shannonova teorému, ze kterého vyplývá, že velikost vzorkovacího intervalu by měla být menší než polovina nejmenšího požadovaného detailu obrazu. Když máme zvolenou vzorkovací frekvenci, zbývá už jen zvolit vzorkovací mřížku. V praxi se používá buď čtvercová, nebo hexagonální (viz obrázek 2.1). Jeden bod vzorkovací mřížky se nazývá pixel.

2.2.2 Kvantizace

Aby byla číslicová reprezentace ekvivalentní se svou spojitou funkcí, musí obraz projít ještě tzv. kvantizací. Počet kvantizačních úrovní musí být dostatečně velký, aby lidské vnímání dokázalo



Obrázek 2.1: Čtvercová a hexagonální mřížka (převzato z [2]

rozpoznat jemné stíny v obraze. Většina zařízení na zpracování číslicového obrazu používá k stejných intervalů. Jestliže k vyjadření jasu obrazu je použito 8bitů, pak počet urovní jasu bude $k = 2^8$. U barevných obrázků se většinou používá 8bitů pro jeden kanál v pixelu (kanály jsou červený,zelený, modrý). Je však možné nalézt i systémy používající jiný počet bitů. Více informací v [8] a [2].

2.3 Reprezentace barev v obraze

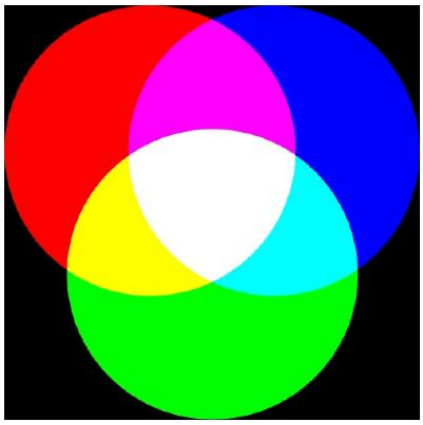
Jelikož člověk vnímá svět barevně, je třeba, aby i barva byla na počítači nějakým způsobem reprezentována. Většinou se k tomu využívá míchání barev. Pak nám stačí 3 základní barvy k získání dostatečného množství barev. Míchání barev může být aditivní a subtraktivní.

2.3.1 Barevný model RGB

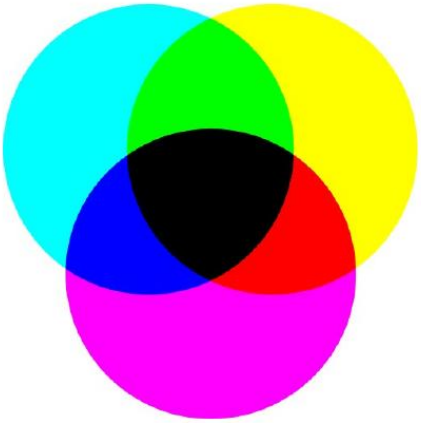
Model RGB je založen na aditivním míchání barev,které se provádí ze tří základních barev: červená, zelená a modrá. Výsledná barva je pak součtem intenzit jednotlivých barevných složek. Při smíchání všech tří složek maximálních možných intenzit vznikne bílá barva. Tento model je základním barevným modelem v počítačové grafice a v počítačové technice obecně. V počítačové grafice se většinou používá pro každý barevný kanál 8bitů. Tedy každá barevná složka má rozsah hodot 0–255. Tento barevný model se užívá především v monitorech, projektorech, kamerách atd.

2.3.2 Barevný model CMY(K)

Model CMY je založen na subtraktivním míchání těchto základních barev: žluté, azurové, purpurové a někdy i černé barvy (zejména v tiskárnách). Zde výslednou barvu získáme narozdíl od modelu RGB ubíráním původního světla pomocí barevných filtrů nebo mícháním pigmentových barev. Při složení základních barev získáme černou barvu. Na základě tohoto modelu míchání barev fungují tiskárny, plotry, ofsety a další zařízení.



Obrázek 2.2: Ukázka aditivního míchání barev (převzato z [4])



Obrázek 2.3: Ukázka subtraktivního míchání barev (převzato z [4])

2.3.3 Další barevné modely

Barevných modelů existuje podstatně více než jenom RGB a CMY. Např. model HSV, který narozdíl od předchozích není založen na míchání základních tří barev, ale na jejich vlastnostech. HSV (hue, saturation, value) je tedy systém, kde je barva vyjádřena pomocí odstínu, sytosti a jasu. Tyto složky se míchají pomocí procent v rozsahu 0–100. Pro člověka je tato práce s barvami více intuitivní.

Dalším modelem je např. YUV, kde barva je reprezentována složkami U a V. Y potom vyjadřuje jas. Výhodou tohoto modelu je, že nebarevné informace jsou uloženy pouze ve složce V. Využívá se v televizní technice.

2.3.4 Reprezentace ve stupních šedi

Někdy je pro nás informace o barvě nadbytečná. V takovém případě je vhodné tuto informaci z obrazu odstranit a neplýtvat tak zbytečně pamětí. Avšak potřebujeme zachovat podstatu obrazu. To lze učinit pomocí převodu barevného obrazu na stupně šědi. Zde se zpravidla jedná o převod z 3 bytů pro reprezentaci barev na byte jeden, který reprezentuje stupně šedi. Máme tedy k dispozici pouze 256 odstínů. Při použití vhodného vztahu, tak lze pomocí tohoto rozsahu převést barevný

obrázek na černobílý se zachováním jeho původních obrysů. Pro tento převod byl empiricky zjištěn vzorec 2.1.

$$jas = 0.299R + 0.587G + 0.114B \tag{2.1}$$

Koeficienty u jednotlivých barevných složek jsou různé, protože naše oko je vnímá s odlišnou intenzitou. Na obrázku 2.4 vidíme, jak může takový převod barevného obrázku na obraz ve stupních šedi dopadnout. Je patrné, že i přes ztrátu barevné informace jsme schopni rozpoznat obrysy jednotlivých objektů. Tedy vyjadřovací schopnost obrazu zůstala zachována. Více informací o barevných modelech a převodu do stupní šedi naleznete v [5] a [4].

Obrázek 2.4: Příklad transformace obrazu na stupně šedi. Vlevo původní barevný obraz. Vpravo tentýž obraz ve stupních šedi. (Převzato z [5])

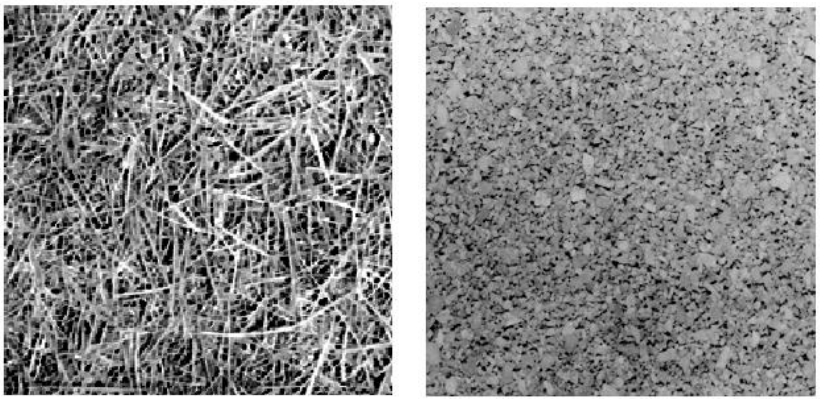
Kapitola 3

Textura

Tato kapitola pojednává o textuře, jak ji lze definovat, k čemu se používá. Dále bude vysvětleno složení textury a jakými způsoby lze textury dělit a popsat.

3.1 Definice textury

Textury se využívají pro realizaci vlastností povrchu nebo struktury objektu. Textura je široce využívaný pojem, a proto pro ni neexistuje jednotná a přesná definice. Často se textura definuje jako opakující se struktura primitiv. Primitivu v textuře se též někdy říká texel. Primitiva mají různou velikost. Pro trávu může být primitivem skupina pixelů korespondující ke stéblu trávy, zatímco např. u korku může být každý pixel primitivem. (viz obrázek 3.1).



Obrázek 3.1: Příklad textur (převzato z [12]).

Z toho vyplývá, že popis textur je závislý na měřítku. Hlavním cílem zpracování textur zpravidla bývá rozpoznání textur a na texturách založené rozpoznávání tvarů v obraze. Při sestavování zařízení pro rozpoznávání textur potřebujeme přesné rysy, podle kterých se dají jednotlivé textury odlišit. Tyto rysy mohou vycházet z barevného tónu nebo ze struktury textury. Zatímco barevný tón je závislý na intenzitách pixelů v primitivu, struktura je závislá na prostorových vztazích mezi texely. Každý pixel může být charakterizován svojí polohou a barevností. Takže primitivum může být charakterizováno jako množina sousedních pixelů s nějakou společnou barevnou charakteristikou.

V případě struktury bude textura popsána počtem a typem primitiv a jejich prostorovou závislostí. Stejný počet a typ primitiv nemusí nutně dávat stejnou texturu. Ty se mohou lišit svým prostorovým

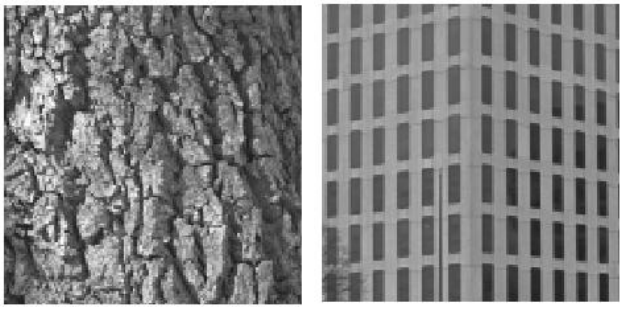
uspořádáním. Stejně tak textury se stejným prostorovým uspořádáním nemusí být stejné. Barevnost a struktura textury však nejsou na sobě nezávislé. Většinou jedna či druhá převažuje. Pokud jsou primitiva malá a jejich barevná odlišnost výrazná, pak se jedná o jemnou texturu. Pokud jsou primitiva velká, složená z několika pixelů, jde o hrubou texturu. Proto je potřeba k popisu používat jak strukturu, tak barevný tón textur.

Při zpracování textur je většinou informace o barvě nadbytečná, a proto se obraz převádí na jeho ekvivalent ve stupních šedi, což bylo popsáno zde: 2.3.4. Pak je výše zmiňovaný barevný tón nahrazen pojmem jas pixelu nebo stupeň šedi. Více informací v [8], [12] a [11].

3.2 Dělení textur

Textury se dělí na slabé a silné. Slabé textury jsou nedekomponovatelné a mají nepravidelnou strukturu. Jejich primitiva jsou blízko u sebe. Tyto textury lze adekvátně nahradit frekvencemi primitivních typů vycházejicích z nějakého okolí. Proto je mnoho statistických vlastností textur vyčíslováno právě na slabých texturách.

Silné textury jsou naopak dekomponovatelné a mají výraznou strukturu. Mezi primitivy jsou nějakým způsobem stejné prostorové vztahy. K popisu silných textur postačuje frekvence výskytu a páry primitiv v prostorovém vztahu. Silné textury bývají rozpoznávány podle primitiv a jejich prostorového uspořádání. (viz obrázek 3.2). Více informací v [8], [12] a [11].



Obrázek 3.2: Příklad slabé textury vlevo a silné textury vpravo (obrázek fasády převzat z [12]).

3.3 Popis textur

Popis textur se dělí na statistický a strukturní podle toho, jestli se při popisu využívá struktura texturních primitiv, nebo se získávají statistická data na základě výskytu primitiv v textuře.

3.3.1 Statistický popis textur

Statistický popis textur je vhodný zejména pro slabé textury bez výraznějších geometrických vztahů mezi primitivy. Velikost primitiv se zde zpravidla blíží velikosti pixelu. Zde texturu popíšeme tak, že spočítáme určité vlastnosti z barevných tónů pixelů. Z těchto spočítaných hodnot sestavíme takzvaný příznakový vektor. Následně se snažíme najít nějaká pravidla pro zařazení textury podle příznakového vektoru do určité třídy. Tato metoda popisu většinou využívá data získaná z předem připravené testovací sady obrázků.

3.3.2 Strukturní popis textur

Druhou možností popisu textur je popis strukturní. Používá se tam, kde primitiva jsou podstatně větší než velikost pixelu a zároveň jsou nějakým způsobem geometricky uspořádána. Tedy zde je popis možný pomocí více vlastností než těch, které vycházejí z barevné informace v obrazu. Tyto metody nejsou používané tak často jako metody statistické. Při jejich aplikaci se využívá formálních jazyků a gramatik těchto jazyků. Více informací o popisu textur v [8] a [11].

3.4 Matice souslednosti

Matice souslednosti (anglicky co-occurence matrix) se používají pro statistický popis textur. Metoda je založena na zjišťování, jak často se v obraze objevuje konfigurace pixelů se stejným stupněm šedi, vzdálených o určitý vektor. Tato konfigurace rychle roste se vzdáleností u jemných textur, zatímco u hrubých textur roste pomalu. Předpokládejme, že budeme počítat matice souslednosti pro obraz obdelníkového tvaru o rozměrech $M \times N$. Pak $P_{\phi,d}(a,b)$ udává, jak často se objevuje ve vzdálenosti d ve směru ϕ od pixelu se stupňem šedi a pixel se stupňem šedi b. Následně nám pro celý obraz vyjde čtvercová matice, jejíž velikost je dána počtem stupňů šedi.Obvykle se matice souslednosti počítají pro 4 směry a lze je definovat takto:

$$P_{0^\circ,d}(a,b) = |\{[(k,l),(m,n)] \in D : k-m=0, |l-n|=d, f(k,l)=a, f(m,n)=b\}| \tag{3.1}$$

$$P_{45^\circ,d}(a,b) = |\{[(k,l),(m,n)] \in D : (k-m=d, l-n=d) \vee (k-m=-d, l-n=d), f(k,l)=a, f(m,n)=b\}| \tag{3.2}$$

$$P_{90^\circ,d}(a,b) = |\{[(k,l),(m,n)] \in D : |k-m|=d, l-n=0, f(k,l)=a, f(m,n)=b\}| \tag{3.3}$$

$$P_{135^\circ,d}(a,b) = |\{[(k,l),(m,n)] \in D : (k-m=d, l-n=d) \vee (k-m=-d, l-n=-d), f(k,l)=a, f(m,n)=b\}|, \tag{3.4}$$

kde $|\{\dots\}|$ udávají četnost konfigurace v obraze a $D = (M \times N) \times (M \times N)$. k a l jsou x-ová a y-ová souřadnice pixelu se stupňem šedi a a m, n jsou souřadnice pixelu b. Tedy např. $P_{90^\circ,d}(a,b)$ v praxi znamená, že se posuneme po ose x o vzdálenost d doleva nebo doprava a y-ová souřadnice zůstane nezměněna. Takto definované matice jsou symetrické.

Pro objasnění zde uvedu příklad výpočtu matic souslednosti pro 4 stupně šedi(viz tabulka 3.1).

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

Tabulka 3.1: Příklad obrazu se čtyřmi stupni šedi (převzato z [8])

Výsledné matice souslednosti pro obraz z tabulky 3.1:

$$P_{0^\circ,1} = \begin{vmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{vmatrix}$$

$$P_{135^\circ,1} = \begin{vmatrix} 2 & 1 & 3 & 0 \\ 1 & 2 & 1 & 0 \\ 3 & 1 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{vmatrix}$$

Např. pro $P_{0^\circ,1}(0,0)$ je výsledek 4, protože v uvedeném obraze celkem 4-krát nalezneme ve směru 0° a vzdálenosti 1 od pixelu s hodnotou 0 pixel s hodnotou 0. Stejně tak např. $P_{0^\circ,1}(3,2) = 1$ a z důvodu symetrie, tak musí platit, že: $P_{0^\circ,1}(2,3) = 1$ Postup sestavování matic pro jiné směry a vzdálenosti je obdobný.

Výhodou matic souslednosti je popis prostorových vztahů mezi barevnými pixely. Na druhou stranu matice souslednosti neuvažují primitivní okraje a nejsou vhodné pro textury s velkými texely. Další jejich nevýhodou je velká početní náročnost, která se však v praxi snižuje redukováním stupňů šedi na 32 nebo 64. To má za následek určitou nepřesnost ve výpočtu, avšak pro praxi zpravidla nepodstatnou. Matice souslednosti dávají dobré výsledky pro rozpoznání textur.

Matice souslednosti se dále zpracovávají pomocí Haralickových příznaků. Více viz [8], [12] a [11].

3.4.1 Haralickovy příznaky

Haralickovy příznaky se využívají pro výpočet jistých charakteristik z matic souslednosti. Dohromady je těchto příznaků 14. Zde představime pouze ty, které jsou implementovány v našem projektu. Aby se dosáhlo určité nezávislosti na směru textury, je vhodné udělat pro každý příznak průměr ze čtyř směrů matic. Více viz [8], [11] a [1].

- Energie (někdy též v angličtině nazývaná jako angular second moment):

$$\sum_{a,b} P^2_{\phi,d}(a,b). \tag{3.5}$$

- Entropie:

$$\sum_{a,b} P_{\phi,d}(a,b) \log_2 P_{\phi,d}(a,b). \tag{3.6}$$

- Kontrast:

$$\sum_{a,b} (a-b)^2 P_{\phi,d}(a,b). \tag{3.7}$$

- Homogenita (anglicky inverse difference moment):

$$\sum_{a,b;a\neq b} \frac{P_{\phi,d}(a,b)}{(a-b)^2} \tag{3.8}$$

- Korelace:

$$\frac{\sum_{a,b}[(ab)P_{\phi,d}(a,b)] - \mu_x\mu_y}{\sigma_x\sigma_y}, \tag{3.9}$$

kde μ_x, μ_y jsou průměry a $\sigma_x, sigma_y$ jsou standardní odchylky.

$$\mu_x = \sum_a a \sum_b P_{\phi,d}(a,b), \qquad \sigma_x = \sum_a (a-\mu_x)^2 \sum_b P_{\phi,d}(a,b),$$

$$\mu_y = \sum_b b \sum_a P_{\phi,d}(a,b), \qquad \sigma_y = \sum_b (b-\mu_y)^2 \sum_a P_{\phi,d}(a,b).$$

Pomocí těchto příznaků, lze z matic souslednosti získat čísla, která se využijí k zařazení textur do předem připravených tříd. Haralickovy příznaky jsou nejčastěji využívaným způsobem pro vyhodnocení matic souslednosti.

3.5 LBP

Zkratka LBP pochází z anglického local binary patterns, což lze do češtiny přeložit jako lokální binární vzory. Opět se jedná o statistickou metodu popisu textur. Tato metoda byla vynalezena teprve nedávno na univerzitě ve finském Oulu. Jedná se o velice rychlou metodu, kterou lze snadno realizovat i v hardwaru. Často se užívá pro detekci objektů v obraze. Její velkou výhodou je také její invariantnost vůči změnám osvětlení, což v praxi znamená, že metoda dává podobné výsledky v průběhu celého dne. Existuje více druhů metod LBP, ale všechny vycházejí ze základního algoritmu. Více informací v [3]

3.5.1 Základní algoritmus LBP

Při výpočtu LBP pro určitý pixel zpravidla vycházíme z jeho osmiokolí. Nejdříve zjistíme intenzitu prostředního pixelu, kterou následně porovnáme s hodnotami intenzity pixelů v okolí.

Tedy vlastně provedeme prahování osmiokolí podle centrálního pixelu. Tzn. pixel s větší intenzitou než prostřední pixel má hodnotu 1 a naopak, pixel s menší intenzitou je 0. Výsledkem tohoto počínání je, že v osmiokolí se nachází pouze hodnoty 1 a 0.

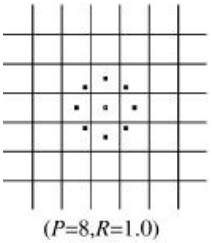
Dalším krokem při výpočtu je určení si směru, podle kterého budeme pixelům v osmiokolí přiřazovat koeficienty. Pro první prvek ve směru je tak koeficient $2^0 = 1$, pro druhý $2^1 = 2$ atd. Výsledný součet koeficientů je 255, což je maximální možná hodnota LBP pro jeden pixel, kalkulujeme-li s osmiokolím. Tedy výsledný LBP centrálního pixelu je součet koeficientů vynásobených hodnotami pixelů v osmiokolí.

Tento postup aplikujeme na každý pixel obrazu. Následně zjistíme četnost jednotlivých hodnot LBP v obraze a sestavíme z nich histogram pro hodnoty 0-255. Ten se v praxi zpravidla normuje. Výsledný histogram je u LBP zároveň příznakovým vektorem. Při určování podobnosti textur se porovnávají histogramy např. pomocí eukleidovské vzdálenosti, která je pro body $x = [x1, \ldots, xn]$ a $y = [y1 \ldots, yn] \in \mathbb{R}^n$ definována takto:

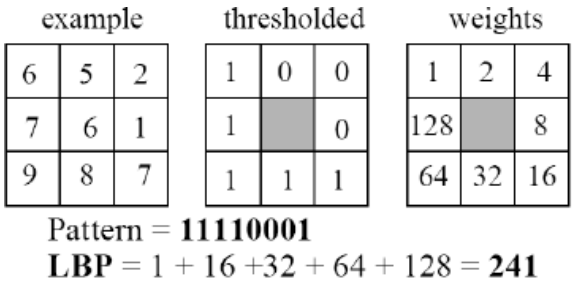
$$p(x,y) = \sqrt{(y1-x1)^2 + \cdots + (yn-xn)^2} \tag{3.10}$$

Matematicky lze výpočet LBP vyjádřit takto:

$$LBP_{P,R} = \sum_{p=0}^{p-1} s(g_p - g_c)2^p, \tag{3.11}$$



Obrázek 3.3: Ukázka osmiokolí pixelu (převzato z [6])



Obrázek 3.4: Ukázka výpočtu LBP pro jeden pixel (převzato z [6])

$$kde \quad s(x) = \begin{cases} 1, x \geq 0 \\ 0.x < 0 \end{cases}$$

P ve vzorci představuje velikost počítaného okolí. Pro náš případ tedy $P = 8$. R je vzdálenost od centrálního pixelu. Pro osmiokolí $R = 1$. Ve vzorci p přestavuje pořadí pixelu ve vybraném směru, g_p je intenzita pixelu, g_c intenzita centrálního pixelu a s je hodnota po prahování. Více informací v [11], [7] a [6].

3.5.2 Rotačně invariantní LBP

Základní metoda LBP je citlivá k rotaci. To znamená, že pokud je v osmiokolí stejná konfigurace pixelů, avšak o něco posunuta vůči centrálnímu pixelu, pak vyjde různá hodnota LBP(viz obrázek 3.5).

Tímto dochází k tomu, že základní LBP nerozeznají různě natočená, stejná primitiva. Proto je třeba výsledek klasického LBP upravit např. podle vzorce 3.12, který vyjadřuje bitovou rotaci vpravo. Tedy např. číslo 00110100 převede na 00001101.

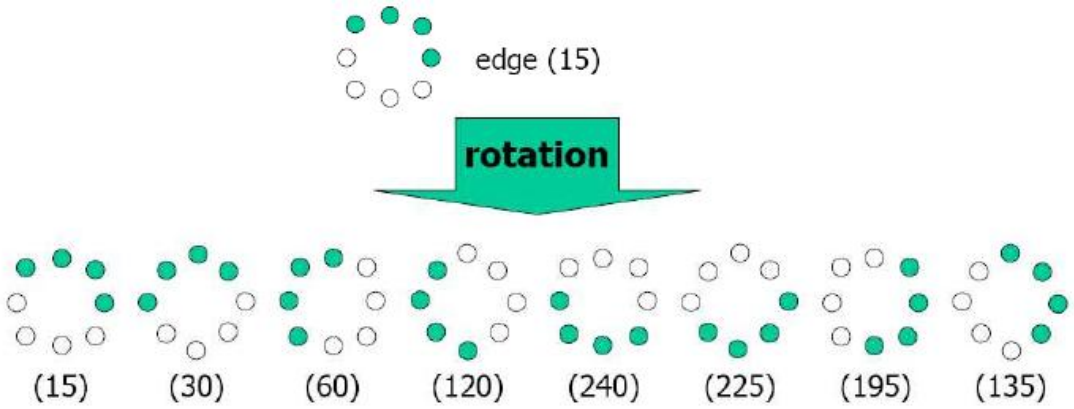
$$LBP^{ri}_{P,R} = min\{rot(LBP_{P,R}, i)|i = 1, 2..., P - 1\} \tag{3.12}$$

Více informací v [11], [7] a [6].

3.5.3 Uniformní vzory

Uniformní vzory jsou specifická podmnožina LBP. Všech 256 vzorů zakódovaných do histogramu totiž není stejně důležitých. Některé vzory jsou více stabilní vůči geometrické transformaci obrazu a ty jsou pro klasifikaci výhodnější.

Najít takové vzory však může být časově velmi náročné. Lze pro to použít paprskové vyhledávání (anglicky beam search) anebo využít podstatně lepší metodu uniformních vzorů. Výhodou



Obrázek 3.5: Problém rotace u základní metody LBP (převzato z [11])

uniformních vzorů je, že si dopředu definujeme množinu vhodných vzorů, a tak jejich časově náročné vyhledávání není potřeba.

Nejdřív je třeba si definovat příznak neuniformity $U(LBP)$. Ten reprezentuje počet přechodů mezi 1 a 0 v čísle LBP. Tedy např. čísla 00000000 a 11111111 mají $U = 0$. Stejně tak např. čísla 10010010 a 01011011 mají stejné $U = 5$, jelikož obsahují celkem 5 přechodů z 0 do 1 a naopak.

Pro nás jsou ale nejvíce zajímavé vzory, jejichž příznak neuniformity dosahuje nejvýšše hodnoty 2. Tedy počet přechodů mezi 1 a 2 je maximálně 2. To jsou tato čísla: 00000000, 00000001, 00000011, 00000111, 00001111, 00011111, 00111111, 01111111 a 11111111 a jejich rotované verze pro zajištění rotační invariantnosti. Ostatní vzory mají $U \geq 4$. Tedy obsahují více přechodů a nejsou tolik stabilní vůči geometrické transformaci.

Námi vybrané vzory nejsou tolik náchylné k naklonění a rotaci. Dohromady se svými rotačními variantami dávají pouhých 58 z celkových 256 originálních nerotovaných vzorů, které vytvoříme z osmiokolí. Uniformní vzory lze formálně definovat takto:

$$LBP^{riu2}_{P,R} = \begin{cases} \sum_{p=0}^{p-1} s(g_p - g_c) & \text{if } U(LBP_{P,R}) \leq 2 \\ P+1 & jinak \end{cases}, \tag{3.13}$$

kde

$$\begin{aligned} U(LBP_{P,R}) &= |s(g_{p-1} - g_c) - s(g_0 - g_c)| \\ &+ \sum_{p=1}^{p-1} |s(g_p - g_c) - s(g_{p-1} - g_c)|. \end{aligned} \tag{3.14}$$

V rovnici 3.13 vyjadřuje horní index $riu2$, že se používá uniformní invariantní vzor, který má $U \leq 2$. $P + 1$ je maximální počet uniformních binárních vzorů, které můžou vzniknout v okolí o velikosti P pixelů. První řádek této rovnosti obsahuje vzorec pro uniformní matice s $U \leq 2$ a druhý řádek odpovídá neuniformním vzorům.

Počítání s necelou pětinou možných čísel sice nevypadá, že by mělo být příliš přesné. Avšak opak je pravdou, především díky dobře zvoleným vzorům. Více informací v [11], [7] a [6].

Kapitola 4

Návrh systému

V této kapitole bude uvedeno, jak by měl vypadat výsledný program. Jak byl navrhnut a jaká je jeho struktura.

4.1 Popis cílového systému

Cílem bylo vytvořit konzolovou aplikaci v jazyce C++, která bude porovnávat textury pomocí metody LBP(viz 3.5) a matic souslednosti(viz 3.4). Program měl pracovat s barevnými obrázky ve formátu .jpg(.jpeg). Výstupem programu měly být soubory naplněné hodnotami, které charakterizují podobnost zadaných textur.

4.2 Návrh aplikace

I přes implementaci v objektově orientovaném jazyce C++, jehož použití bylo nevyhnutelné, jelikož ho vyžadovala námi vybraná knihovna pro načítání obrázků, jsme zvolili návrh klasický.

Program byl navrhnut tak, aby vzorové textury načítal z konfiguračního souboru, který mu bude předán jako první argument z příkazového řádku. Dále hned po spuštění vyzve uživatele k zadání textury, kterou bude porovnávat se vzory.

Následně přejde k výpočtu LBP. Jakmile jsou spočteny potřebné histogramy, porovná je a výsledek zapíše do souboru.

Dalším krokem programu je převod obrazu na jeho reprezentaci ve stupních šedi(viz 2.3.4). Pro takto upravený obraz jsou spočítány matice souslednosti, které jsou dále vyhodnoceny pomocí Haralickových příznaků, jež jsou následně uloženy do souboru.

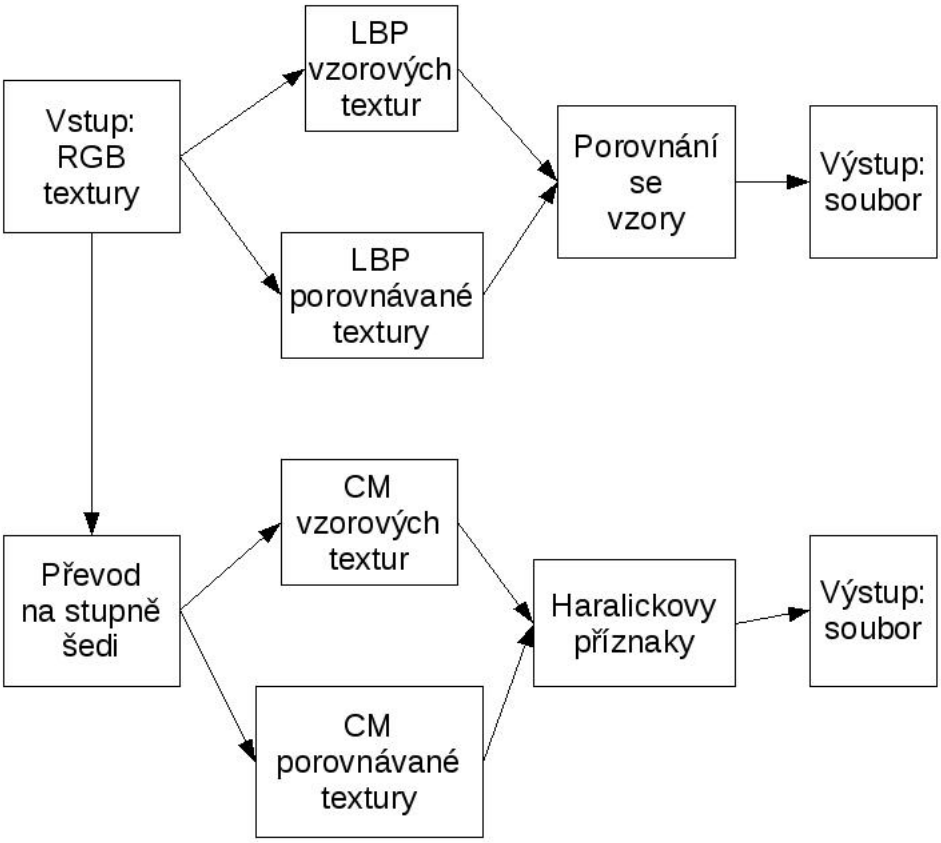
Výstupem programu tedy budou soubory, které v případě LBP budou obsahovat míru podobnosti k jednotlivým vzorům. Zatímco u matic souslednosti zde budou uloženy spočtené Haralickovy příznaky, které je třeba dále zpracovat, aby bylo možné určit podobnost textur.

4.3 Návrh testů

Pro otestování programu rozdělíme textury do tříd podle jejich podobnosti. Na těchto texturách zjistíme, jaké hodnoty dostaneme pro navzájem si podobné textury a naopak pro různé textury. Na základě těchto hodnot, pak budeme moci určit, zda textury patří do stejné třídy či nikoliv. U metody LBP budou porovnávané hodnoty v rozsahu 0–100%.

U matic souslednosti budou nejdříve spočteny hodnoty Haralickových příznaků. Následně určíme jejich průměr a směrodatnou odchylku. Dále budeme předpokládat, že textura může být

texturou dané třídy v případě, že její příznak vyjde v rozmezí průměr $+-$ směrodatná odchylka tohoto příznaku pro onu třídu.



Obrázek 4.1: Schéma programu

Kapitola 5

Implementace

V této kapitole bude popsáno, jak byl program podle návrhu (viz předešlá kapitola 4) realizován. Jaké knihovny a algoritmy byly využity. Jaké testy byly navrženy a podle čeho je podobnost vyhodnocována.

5.1 Knihovna CImg.h

Pro práci s barevnými obrázky byla vybrána knihovna `CImg.h` [9]. Ta je licencována jako open source. Jejím autorem je David Tschumperlé a jeho spolupracovníci.

Knihovna definuje jednoduché třídy a metody pro práci s obrazem v jazyce C++. Umožnuje načítání a ukládání obrázků, přístup k jednotlivým pixelům a jejich barevným hodnotám, kreslení různých grafických primitiv a grafů, zmenšování, zvětšování obrazu, jeho rotace, počítání statistik atd. Zvládá i interakci s uživatelem.

Velkou výhodou knihovny je její jednoduchost. Je založena na jediném hlavičkovém souboru `CImg.h` a po jeho vložení do kódu s ní můžeme začít okamžitě pracovat. Knihovnu je možné dále rozšiřovat pomocí různých plug-inů a lze ji využít ve spojení s dalšími knihovnami, např. pro načítání obrázků různých formátů.

Knihovna se skládá z pouhých čtyř tříd a funguje na různých platformách(Unix, Windows, MacOS X, *BSD) s různými kompilátory jazyka C++(Visual C++, GNU g++, Borland bcc . . .).

V naší práci byla knihovna využita ve spolupráci s knihovnou `libjpeg`, pro načtení obrazu ve formátu .jpg(.jpeg). Obrázek je načítán pomocí metod třídy `cimg_library::CImg`. Dále nám byla užitečná makra cyklů pro průchod obrázkem pixel po pixelu a odchytávání chyb při načítání obrázků zabezpečila třída `cimg_library::CImgException`.

5.2 Program BP

Program BP je tedy těžištěm této práce. Jsou v něm v praxi uplatněny námi získané poznatky o LBP (3.5), maticích souslednosti (3.4) a Haralickových příznacích (3.4.1).

Aplikace se skládá z šesti souborů zdrojového kódu, plus knihovna `CImg.h`. Přičemž v souboru `mian.cpp` nalezneme pouze deklarace potřebých polí, která budou předávána jednotlivým funkcím a v nichž se budou ukládat výsledky těchto funkcí. Dále je zde implementován výstup výsledků do souboru. Jednotlivé metody pro porovnání obrazu a funkce k nim potřebné jsou zapsány v souborech `cm.cpp`, `haralick.cpp`, `functions.cpp` a `lbp.cpp`. Hlavičkový soubor `functions.h` obsahuje deklarace funkcí a maker, které jsou volány z jiných souborů, než kde jsou definována.

Program po spuštění z konzole načte jména vzorových textur z kofiguračního souboru do globálního pole a určí jejich počet(též globální proměnná). V konfiguračním souboru je na každém řádku uveden jeden vzor. Pak je uživatel vyzván k zadání textury, kterou chce porovnat se vzory.

Následně se postupně vypočítávají LBP pro všechny vzory a vyšlé histogramy jsou ihned zapsány do souborů. Pak se spočte LBP pro porovnávanou texturu a následně je vyhodnocena její podobnost s histogramy uloženými v souborech.

Přičemž metoda LBP byla implementována přesně podle 3.5.1. Tedy k získání hodnoty LBP jednoho pixelu bylo použito jeho osmiokolí. Následné porovnání histogramů bylo založeno na eukleidovské vzdálenosti viz 3.10. Výsledky porovnání jsou uloženy do souboru `vysledky_lbp`, který je po opětovném puštění programu přepisován. Výsledkem LBP je číslo v rozsahu 0-100%, přičemž čím vyšší číslo, tím větší podobnost.

Po porovnání histogramů následuje převod obrazu do stupňů šedi(2.3.4) podle vzorce 2.1 a výpočet matic souslednosti pro směry uvedené v 3.4. Výpočet matic zde není implementován přesně podle definičních vztahů 5.1, 3.2, 3.3. a 3.4. Pro jednoduchost námi počítané matice nejsou symetrické. Tedy např. vztah 5.1 by byl upraven takto:

$$P_{0^\circ,d}(a,b) = |\{[(k,l),(m,n)] \in D : k - m = 0, l - n = d, f(k,l) = a, f(m,n) = b\}|. \tag{5.1}$$

Tedy výsledná matice vzniká pouze porovnáním s pixely vpravo vzdálenými o d. V našem případě se jedná o porovnání pixelů sousedních, tzn. $d = 1$. Pro vzory se opět výsledné matice souslednosti ukládají do souborů, odkud jsou po výpočtu matic pro porovnávanou texturu opět načteny a dojde k jejich vyhodnocení pomocí Haralickových příznaků, které jsou uvedeny v 3.4.1. Výsledky tohoto zpracování se uloží do souboru `vysledky_CM` pro vzory a `vysledky_akt_CM` pro porovnávanou texturu.

Program BP tedy vyhodnocuje podobnost metodou LBP a vypočítává z matic souslednosti Haralickovy příznaky, které po dalším zpracování vedou ke zjistění podobnosti obrazů.

5.3 Aplikace průměr

Aplikace průměr je pomocná aplikace k programu BP, která pomáhá při zpracování čísel vyšlých z Haralickových příznaků. Program se spouští bez parametrů a počítá aritmetický průměr a směrodatnou odchylku z dat uvedených v souboru `vysledky_CM`.

5.4 Ovládání programu BP

Příklad spuštění:

`./BP -h` –spuštění nápovědy

`./BP config_file` –spuštění programu pro vzorové textu uvedené v `config_file`

Textury, se kterými chceme pracovat, musí být ve stejném adresáři jako aplikace.

5.5 Testovací galerie

Pro testovací účely byla sestavena galerie textur o rozměrech 100×150 px. Ty byly rozřazeny na základě své podobnosti do pěti tříd. Jedná se o textury zobrazující kůru stromu, písek, skálu, trávu a vodu. Každá třída obsahuje 10 textur. Textury byly pořízeny ze serveru cgtextures.com viz [10].

Kapitola 6

Výsledky a testování

Tato kapitola má za cíl ukázat vzorek výsledků, které byly získány naší aplikací na základě testovací galerie.

6.1 Výsledky metody LBP

Nejdříve je třeba zjistit, jaké hodoty získáme metodou LBP při porovnání textur ze stejné třídy. To bylo provedeno tak, že do konfiguračního souboru se daly všechnny textury jedné třídy a byly porovnávány s texturami téže třídy.

Např. pro texturu `kura1.jpg` při porovnání s obrázky stejné třídy (viz 6.1) vyšly tyto hodnoty:

100.00	98.39	98.02	98.31	97.89	98.06	97.43	98.22	98.27	97.99

Hodnoty jsou natěsnány v rozmezí 97–100%. Z toho plyne velká podobnost zvolených textur. Velice podobná čísla vycházela i při porovnání s jinými texturami z dané třídy.

Obrázek 6.1: Vlevo textura `kura1.jpg`. Zbylé textury jsou zástupci stejné třídy kůra.

Obrázek 6.2: Vlevo textura `trava1.jpg`. Zbylé textury jsou zástupci stejné třídy tráva.

Např. pro podobnost textury `trava1.jpg` s obrázky z její třídy tráva (viz obrázek 6.2) vychází tyto výsledky:

100.00	95.75	95.66	93.84	93.02	93.15	92.61	93.06	90.34	89.82

Zde se již podobnost pohybuje od 89 do 100%. Při testování byla naměřena nejnižší hodnota pro textury jedné třídy přes 88%. Z toho lze usuzovat, že pokud vyjde hodnota podobnosti větší než 88%, jedná se pravděpodobně o textury stejné třídy. Hodnota 100% by tedy měla znamenat totožnost textur. Během testování, však byl zaznámenán i případ, kdy se nejednalo o úplně stejné textury.

Obrázek 6.3: Vlevo textura `voda3.jpg`. Zbylé textury jsou zástupci třídy tráva.

Obrázek 6.4: Vlevo textura `pisek4.jpg`. Zbylé textury jsou zástupci třídy voda.

Hodnoty pro podobné textury máme zjistěny a nyní je třeba zjistit, jaké hodnoty vyjdou pro textury různých tříd. To bylo pozorováno tak, že vzory byly obrázky jedné třídy a byly porovnávány s obrázky pocházejícími z jiné třídy. Příklady zjištěných hodnot jsou uvedeny zde:

75.89	75.33	77.38	79.51	80.83	81.14	80.56	80.15	80.69	82.76
86.47	88.10	87.31	88.56	88.35	89.47	89.02	90.81	90.78	90.81

Nepodobné si textury tedy dosahují hodnot v rozsahu od 75% do 91%. Pro ilustraci můžeme vidět některé textury z prvního porovnání na obrázku 6.3 a z druhého porovnání na obrázku 6.4.

Obrázek 6.5: Příklad textur z ruzných tříd, které si však jsou velice podobné.

Tedy z výšše uvedených výsledků plyne, že pravděpodobnost, že by textury s hodnotou podobnosti v rozsahu 0–88% pocházely ze stejné třídy, je téměř nulová. Naopak problém nastává pří zjišťování, jestli textury patří do jedné třídy. Zde se u námi zvolenými obrázky projevila velká barevná a strukturní (zejména velikost texturních primitiv) podobnost tříd písek, skála a kůra viz textury na obrázku 6.5. Pro podobnost textur z těchto tříd vycházely hodnoty nerozeznatelné od vyčíslení podobnosti textur stejné třídy:

96.40	96.59	96.26	94.55	95.80	95.45	95.66	96.05	96.14	96.62

V tabulce 6.1 je znázorněn příklad, kdy jako vzory byly zvoleny textury z různých tříd viz obrázek 6.6. Pokud tyto textury porovnáme s texturou ze třídy voda nebo tráva, vyjdou výsledky

podobné prvnímu řádku tabulky 6.1, přičemž poslední sloupec v tomto řádku reprezentuje texturu ze stejné třídy. Zde lze zcela jistě prohlásit, že pro texturu byla nalezena třída, do které patří.

Naopak druhý řádek v tabulce ukazuje, jak dopadne porovnání se stejnými vzory pro texturu skály (viz obrázek 6.6). Zde vidíme již dříve zmíněný závěr, že textury ze tříd písek, skála a kůra nelze naším programem od sebe rozpoznat.

Obrázek 6.6: Různé textury, které byly využity jako vzory k příkladu 6.1.

kůra	písek	skála	tráva	voda
89.81	89.02	87.57	80.99	97.24
96.59	96.73	94.86	86.37	92.90

Tabulka 6.1: V prvním řádku výsledek pro texturu `voda7.jpg` a v druhém pro texturu `skala2.jpg`

Tedy pro námi zvolené textury lze pomocí metody LBP zjistit, že textury nepatří do stejné třídy. U těchto obrázků však metodou LBP nejde jednoznačně určit, jestli obrázky pochází z jedné třídy. Touto metodou u naší testovací množiny jsme schopni odlišit pouze, jestli textury patří do třídy voda, nebo tráva, a nebo do jedné ze tříd skála, písek, kůra.

6.2 Výsledky matic souslednosti a haralickových příznaků

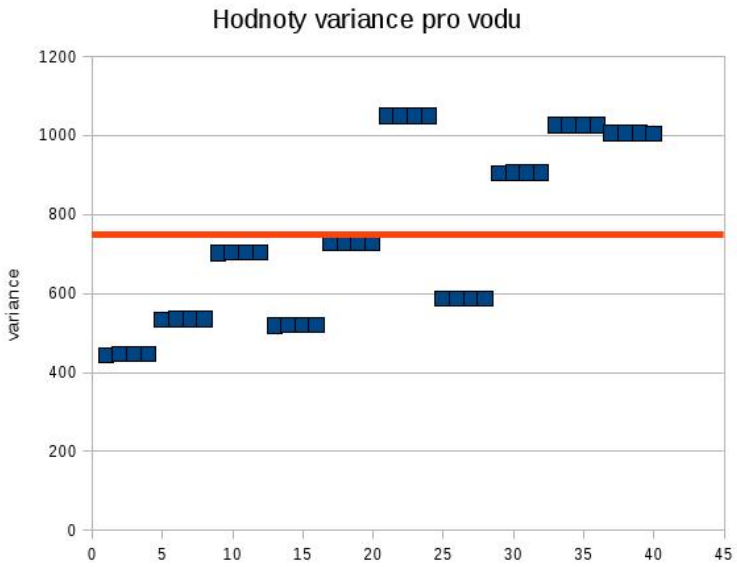
Abychom mohli vyhodnocovat podobnost na základě matic souslednosti a Haralickových příznaků, je potřeba nejdříve zjistit nějaké charakteristické hodnoty pro každou třídu. Za charakteristické hodnoty zde byl zvolen aritmetický průměr všech vzorů a jeho směrodatná odchylka pro jednotlivé příznaky.

Na obrázku 6.7 je zobrazen graf hodnot variance pro vodu. Na grafu je znázorněno celkem 40 hodnot pro 4 matice různých směrů a 10 vzorů. Červenou čarou je zde vyznačen jejich průměr. Na grafu je vidět, že hodnoty variance se se směrem matice příliš neliší. Nejnižší hodnota je 443 a nejvyšší 1048. Přičemž průměr byl spočten na 750. Směrodatná odchylka vyšla 218. Z toho plyne, že pokud variance vyjde v rozmezí 532–968, bude textura podle variance zařazena do třídy voda. Podíváme-li se na hodnoty znázorněné na grafu, do daného rozmezí padne zhruba 60% hodnot.

V tabulce 6.2 jsou ukázány rozsahy jednotlivých příznaků pro každou třídu. Po podrobnějším prostudování tabulky zjistíme, že se rozsahy příznaků často překrývají, a proto nelze určit druh textury podle jediného příznaku. Z námi počítaných příznaků dokáže od sebe nejvíce tříd rozeznat entropie, pak homogenita, energie, variance, kontrast a nejhůře si vede korelace.

V tabulce 6.3 jsou uvedeny hodnoty příznaků pro texturu `trava5.jpg`. Nyní tyto hodnoty porovnáme s rozsahy pro jednotlivé třídy, které jsou uvedeny v tabulce 6.2.

Porovnáme-li např. energii, zjistíme, že námi získaná hodnota 0.002530 odpovídá rozsahu třídy voda a písek. V tomto porovnávání dále pokračujeme pro každý příznak. Pro kontrast nám vyjdou



Obrázek 6.7: Hodnoty příznaku variance pro vodu.

	Kůra	Skála	Tráva	Voda
Energie	0.0009 – 0.0014	0.0010 – 0.0022	0.0026 – 0.0051	0.00197 – 0.0044
Kontrast	3244.7 – 5650.17	2519.7 – 5156.0	1481.8 – 2387.8	1815.4 – 3492.3
Variance	895.6 – 1277.7	413.6 – 741.4	704.9 – 1197.8	531.2 – 967.9
Homogenita	0.0989 – 0.1340	0.1224 – 0.1807	0.2271 – 0.3021	0.1450 – 0.2149
Korelace	–398.8 – –180.6	–323.4 – –131.7	–593.5 – –141.2	–1735.8– –463.0
Entropie	9.908 – 10.477	9.447 – 10.330	8.728 – 9.303	8.558 – 9.322

Tabulka 6.2: Příklad rozsahů příznaků pro některé třídy.

třídy tráva, písek a voda, pro varianci kůra a tráva, pro homogenitu tráva a písek, pro korelaci tráva,písek a voda a pro entropii budeme uvažovat třídy kůra, tráva, písek a voda.

Tedy celkem pět příznaků nám potvrdilo, že textura `trava5.jpg` skutečně patří do třídy tráva. Tímto tvrzením si však nemůžeme být zcela jistí, neboť stejný počet příznaků vyšel v rozsahu třídy písek. Navíc voda nám vyšla u čtyř příznaků. Naopak si můžeme být úplně jistí, že textura `trava5.jpg` není zástupcem třídy skála, jelikož ani jeden příznak nebyl z jejího rozsahu hodnot. Do třídy kůra, by byla textura `tráva5.jpg` zařazena pouhými dvěma příznaky.

Energie	Kontrast	Varince	Homogenita	Korelace	Entropie
0.002530	2158,815	1194,65	0,231329	-560,309250	9,293245

Tabulka 6.3: Hodnoty Haralickových příznaků pro texturu `trava5.jpg`.

Z uvedených příkladů vyplývá, že žádný příznak není tak silný, aby dokázal od sebe odlišit všechny třídy. Dokonce nám většinou po porovnání všech příznaků vyjde více možných tříd, kam by daná textura mohla patřit. Avšak rozhodovací schopnost Haralickových příznaků by se dala

zvýšit určením pravidel pro jejich porovnávání na základě výsledků větší testovací množiny.

Kapitola 7

Závěr

Cílem této práce bylo prostudovat teorii ohledně statistického popisu textur, ze kterého jsme vybrali dvě metody (LBP a matice souslednosti). Tyto dvě metody jsme si následně vyzkoušeli v praxi, kdy jsme se snažili zjistit, s jakou mírou účinnosti dokáží správně určit třídu textury.

Výsledný program nakonec dokázal pomocí metody LBP určit podobnost textur a matice souslednosti byly zpracovány Haralickovými příznaky, které by po následném vhodném zpracování, vedly ke správnému určení třídy textury.

Dalším vylepšením stávajícího programu by mohlo být použití dokonalejších metod LBP jako jsou rotačně invariantní LBP nebo uniformní vzory. Dalším rozšířením aplikace by mohl být triviální klasifikátor textur založený na maticích souslednosti a jejich vyhodnocení pomocí Haralickových příznaků. Pokud by se povedlo nalézt vhodné kombinace příznaků pro jednotlivé třídy textur, jednalo by se jistě o program s dobrou úspěšností při zařazování tříd.

Literatura

[1] Boland, M. V.: Haralick texture features [online]. Dostupné na URL: http://murphylab.web.cmu.edu/publications/boland/boland_node26.html, 1999-09-18 [cit. 2009-5-14].

[2] Fiřt, J.; Holota, R.: Digitalizace a zpracování obrazu [online]. Dostupné na URL: <home.zcu.cz/ holota5/publ/DigZprO.pdf> , [cit. 2009-05-16].

[3] Jelínek, T.: *Detekce pohybujících se objektů ve video sekvenci*. FIV VUT v Brně, 2007, diplomová práce.

[4] Kršek, P.; Španěl, M.: *Barvy a barevné modely – přednáška z předmětu IZG*. FIT VUT v Brně, 2007.

[5] Kršek, P.; Španěl, M.: *Redukce barevného prostoru – přednáška z předmětu IZG*. FIT VUT v Brně, 2007.

[6] Maenpaa, T.; Ojala, T.; Pietikainen, M.; aj.: Robust Texture Classfication by Subsets of Local Binary Patterns [online]. Dostupné na URL: www.mediateam.oulu.fi/publications/pdf/3.pdf, [cit. 2009-05-14].

[7] Ojala, T.; Pietikainen, M.; Maenpaa, T.: Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns [online]. Dostupné na URL: www.mediateam.oulu.fi/publications/pdf/94.pdf, 2007-07 [cit. 2009-05-14].

[8] Sonka, M.; Hlavac, V.; Boyle, R.: *Image Processing, Analysis, and Machine Vision, Third Edition*. Thomson, 2008, ISBN 978-0-495-08252-1.

[9] Tschumperlé, D.: The CImg library [online]. Dostupné na URL: http://cimg.sourceforge.net/, 2004-10 [cit. 2009-05-16].

[10] Vijfwinkel, M.: CGTextures [online]. Dostupné na URL: http://www.cgtextures.com/, 2009-05-18 [cit. 2009-5-18].

[11] Španěl, M.: *Analýza a extrakce příznaků z textur – přednáška z předmětu POV*. FIT VUT v Brně, 2007.

[12] Šára, R.: Analýza textury [online]. Dostupné na URL: cmp.felk.cvut.cz/cmp/courses/ZS1/slidy/lecture_texture_sara.pdf, 1999 [cit. 2009-05-16].