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

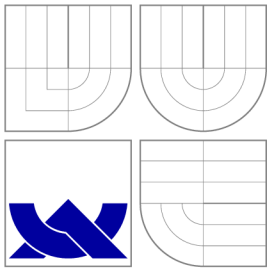
SEGMENTACE OBRAZU PODLE TEXTURY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

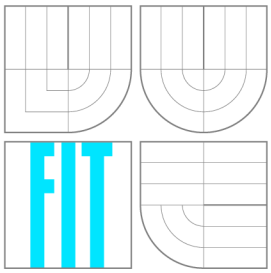
AUTOR PRÁCE
AUTHOR

Bc. VÁCLAV PASÁČEK

BRNO 2012



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

SEGMENTACE OBRAZU PODLE TEXTURY

TEXTURE-BASED IMAGE SEGMENTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VÁCLAV PASÁČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL, Ph.D.

BRNO 2012

Abstrakt

Segmentace obrazu je důležitým krokem zpracování obrazu a textura je jednou z obrazových informací, na jejichž základě lze segmentaci provádět. K popisu textury slouží texturní příznaky, přičemž existuje mnoho způsobů, jak je získat. Zde budou k reprezentaci textury využity Local Binary Patterns neboli LBP. Texturním příznakem u LBP není její hodnota, ale histogram četnosti výskytu v určité oblasti. Hlavním cílem této práce je porovnání vhodnosti několika variant extrakce texturních příznaků pomocí LBP a metod jejich následného shlukování za účelem segmentace obrazu. Ke shlukování texturních příznaků bude použita metoda Fuzzy C-Means.

Abstract

Image segmentation is an important step in image processing. A traditional way how to segment an image is a texture-based segmentation that uses texture features to describe image texture. In this work, Local Binary Patterns (LBP) are used for image texture representation. Texture feature is a histogram of occurences of LBP codes in a small image window. The work also aims to comparison of results of various modifications of Local Binary Patterns and their usability in the image segmentation which is done by unsupervised clustering of texture features. The Fuzzy C-Means algorithm is finally used for the clustering in this work.

Klíčová slova

Textura, texturní příznak, Local Binary Patterns, segmentace obrazu, Fuzzy C-Means.

Keywords

Texture, texture feature, Local Binary Patterns, image segmentation, Fuzzy C-Means.

Citace

Václav Pasáček: Segmentace obrazu podle textury, diplomová práce, Brno, FIT VUT v Brně, 2012

Segmentace obrazu podle textury

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Španěla Ph.D.

.....
Václav Pasáček
21. května 2012

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Michalu Španělovi Ph.D. , bez jehož pomoci a vedení by tato práce nemohla vzniknout. Také bych chtěl poděkovat své rodině a přátelům za jejich morální podporu, která pro mě byla velmi duležitá.

Obsah

Kapitola 1

Úvod

Tato diplomová práce se zabývá jednou z nejdůležitějších oblastí počítačového vidění, a to segmentací obrazu. Segmentace je rozdělení obrazu do několika regionů tak, aby bylo snazší nalezené regiony sémanticky interpretovat, respektive oddělit podstatná data od pozadí. Segmentace se využívá například ve zpracování medicínských dat, leteckých a satelitních snímků, automatické kontrole kvality, detekci otisku prstu a v mnoha dalších oblastech.

Segmentace obrazu může být prováděna prahováním, shlukovou analýzou, metodami orientovanými na regiony obrazu, hybridními metodami nebo na základě detekce hran. Tato práce se zabývá segmentací obrazu pomocí shlukování příznakových vektorů získaných na základě textury.

Hlavním cílem je porovnání jednotlivých variant získání texturních příznaků a metod jejich následného shlukovaní za účelem automatické segmentace obrazu. Toto porovnání realizuji vyhodnocením úspěšnosti segmentace obrazu provedené na jejich základě.

K popisu textury se používají texturní příznaky. K jejich získání bude použita metoda Local Binary Patterns neboli LBP. Texturním příznakem zde není přímo její hodnota, ale histogram četnosti výskytu jednotlivých hodnot v okně určité velikosti se středem v aktuálním bodě. Kromě základní varianty LBP budou zkoumány i modifikace pracující také s lokálním kontrastem. Dále pak LBP modifikované pro práci s barevnými obrazy. Součástí proto bude i vyhodnocení vlivu použitého barevného prostoru.

Ke shlukování příznakových vektorů se použije metoda Fuzzy C-Means neboli FCM. Je to iterativní metoda počítající příslušnost vstupních bodů do jednotlivých shluků a středy těchto shluků. Její výsledek je však závislý na počáteční inicializaci, a proto představíme i meta-heuristiky, které pro ní lze použít. Dále bude testován vliv metriky vzdálenosti užité v FCM na výsledek segmentace. Poslední věcí spojenou s FCM je defuzzifikace jejích výsledků. Tu lze provést několika metodami, přičemž zde bude zkoumáno jakým způsobem ovlivní výsledek segmentace.

Po této úvodní kapitole následuje druhá část, která popisuje teorii analýzy textur, získávání texturních příznaků a Local Binary Patterns. Třetí část si klade za cíl seznámit čtenáře s jednotlivými druhy segmentačních algoritmů, jejich vlastnostmi a metodikami hodnocení uspěšnosti segmentace. Čtvrtá kapitola nejdříve pojednává obecně o technikách shlukování a poté podrobněji rozebírá FCM. Pátá část obsahuje návrh výpočtu texturních příznaků a systému, který bude tyto příznaky využívat k segmentaci obrazu. Šestá část popisuje implementaci testovacího frameworku pro segmentaci obrazu. Sedmá kapitola pojednává o experimentálních výsledcích dosažených pomocí implementovaného frameworku. V poslední osmé kapitole je obsaženo závěrečné zhodnocení dosažených výsledků a důsledky, které z nich vyplývají.

Kapitola 2

Analýza textur

Tato část si klade za cíl seznámit čtenáře s teorií nezbytnou k analýze textur.

2.1 Textura

Protože povrch většiny reálných těles není jednobarevný, je nutné ho nějakým způsobem popsat. K popisu povrchu objektu slouží textura. Neexistuje jednotná definice textury, ale obecně se tímto pojmem označuje specifická povrchová struktura, barva a optické vlastnosti povrchu [46].

Textura se skládá z grafických primitiv zvaných texely. Texely se pravidelně opakují a pokud je jejich počet větší než jejich variabilita, lze o textuře říci, že má výrazné statické vlastnosti. Textura má jedno nebo více charakteristických měřítek.

Textury se dají dělit podle rozměru na plošné, prostorové nebo prostorové měnící se v čase. Také mohou být rozděleny na silné (s pravidelnou strukturou) a slabé (s nepravidelnou strukturou). Textury se obvykle skládají ze dvou složek. Vlastní část je neměnná a závisí pouze na zabarvení povrchu, zatímco nevlastní část je zavislá na intenzitě a směru osvětlení. Textury lze reprezentovat datovou mapou nebo matematickou funkcí.

Obrázek 2.1: Příklad silných(vrchní tři obrázky) a slabých(spodní tři obrázky) textur.

2.2 Popis textur

Tato část vychází z [31]. K popisu textury slouží takzvané texturní příznaky. Existuje mnoho metod pro jejich extrakci, které se navzájem liší nejen principem, efektivností nebo časem výpočtu, ale i oblastí, pro kterou se takto získané texturní příznaky používají.

Existují 2 základní přístupy k texturním příznakům:

- Strukturní - Jestliže jsou texely dostatečně velké, aby bylo možné odlišit je od pozadí, mohou být popsány jejich jednotlivé typy a vzájemné geometrické uspořádání. Vlastní popis lze realizovat polygonální sítí nebo gramatikou.
- Statický - Pokud je počet primitiv mnohem větší než jejich variabilita, mají textury výrazné statické vlastnosti. Popisuje rozložení intenzit v textuře. Textura je pak popsána množinou čísel zvanou příznakový vektor.

2.3 Gáborovská analýza

Tato metoda využívá banku filtrů, které aplikuje postupně na všechny body v obraze. Každý filtr představuje hranový detektor s jiným natočením a velikostí. Tyto filtry se na obraz aplikují pomocí konvoluce. Konvolucí s těmito fitry je získán příznakový vektor.

Matematická definice

Ve zpracování obrazu se používá dvourozměrná Gáborova funkce, která se také nazývá "Gáborova vlnka".

Rovnice této základní Gáborovy funkce [30] je:

$$G_{jkl}(x,y) = e^{\frac{x^2+m^2*y^2}{2*o^2}} * cos(2*Pi*\frac{x}{j}+l) \tag{2.1}$$

Kde o je standardní odchylka, j je délka vlny, m je posun konvolučního jádra, k je úhel natočení gáborovy vlnky.

Těmito funkcemi pak lze generovat celou banku Gáborových filtrů:

$$x = x*cos(k) + y*sin(k) \tag{2.2}$$

$$y = -x*sin + k*cos(k) \tag{2.3}$$

Obrázek 2.2: Dvě dvoudimenzionální Gáborovy vlnky, se stejnou odchylkou, ale s rozdílnou délkou vlny. Převzato z [30].

Gáborovská analýza je jednou z nejpoužívanějších a nejefektivnějších metod popisu textury. Tato metoda je bohužel poměrně výpočetně náročná a není robustní vůči změnán osvětlení.

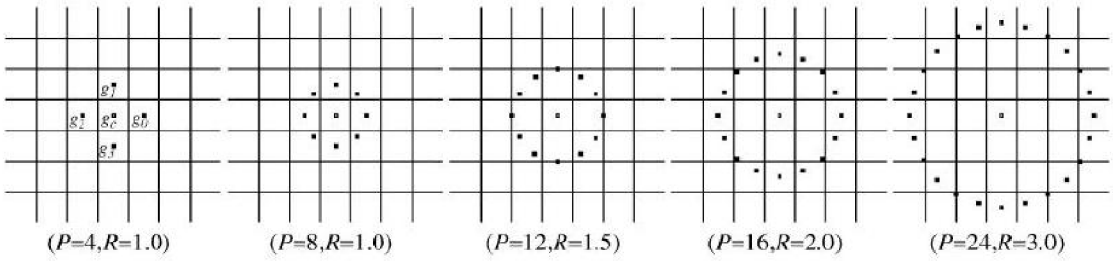
Obrázek 2.3: Energie spektra dvou dvourozměrných Gáborových funkcí. Převzato z [30].

2.4 Local binary patterns

Pro tuto metodu se již vžila zkratka LBP [29] [28] . Tato zkratka bude používáná i v následujícím textu.

LBP kombinuje statický a strukturní přístup k získání texturních příznaků. Její základní myšlenkou je průchod celým obrázkem po jednotlivých bodech a ohodnocení okolí každého bodu. Příznakovým vektorem je potom histogram četnosti výskytu různých druhů okolí pro všechny body v obraze.

Hodnocení okolí bodu probíhá umístěním P bodů ve vzdálenosti R od aktuálního bodu. Umísťěny jsou tak, aby tvořili kruhově symetrické okolí. Hodnota bodů, které přesně nezapadají do mřížky, se bilineárně interpoluje z okolních bodů.



Obrázek 2.4: LBP pro různý počet bodů a různou vzdálenost. Převzato z [29].

Pro nejpoužívanější variantu LBP, kde P = 8 a R = 1, se však nepočítá poloha okolních bodů, z kterých bude LBP extrahován, ale použijí se body 8-okolí.

Hodnoty okolních bodů se následně prahují hodnotou dobu středového:

$$x_t = \begin{cases} 1 & x_o \geq x_c \\ 0 & \text{jinak} \end{cases} \tag{2.4}$$

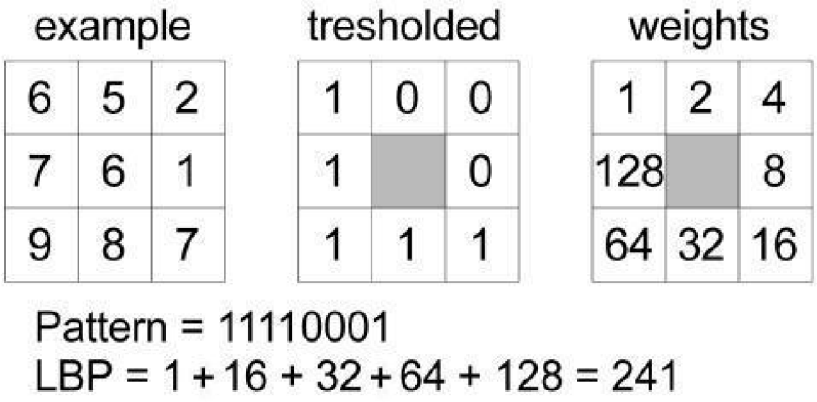
Kde x_c je hodnota středového bodu, x_o je hodnota okolního bodu a x_t je výsledná prahovaná hodnota. Tímto postupem vznikne P prahovaných hodnot x_{t0} až $x_{t(P-1)}$. Každý z okolních bodů bude mít váhu:

$$x_n = 2^n \tag{2.5}$$

Prahované hodnoty jsou následně vynásobeny příslušnou váhou a sečteny. Výsledná hodnota, binární číslo o maximální délce P, je hledaný LBP příznak.

$$LBP = \sum_{n=0}^{P-1} x_n * x_{tn} \tag{2.6}$$

Uvedený postup je znárorněn na obrázku 2.5.



Obrázek 2.5: Postup výpočtu LBP pro P = 8 a R = 1.

Příznakový vektor je histogram četnosti výskytu jednotlivých LBP v celé textuře.

Tento histogram bude mít 2^P položek a je vhodné ho normalizovat:

$$lbp'_n = \frac{lbp_n}{S} \tag{2.7}$$

Kde lbp'_n je normalizovaná hodnota v *n-tém* sloupci histogramu, lbp_n je původní hodnota v *n-tém* sloupci a S je celkový počet LBP příznaků v histogramu.

$$S = \sum_{n=0}^{N-1} lbp_n \tag{2.8}$$

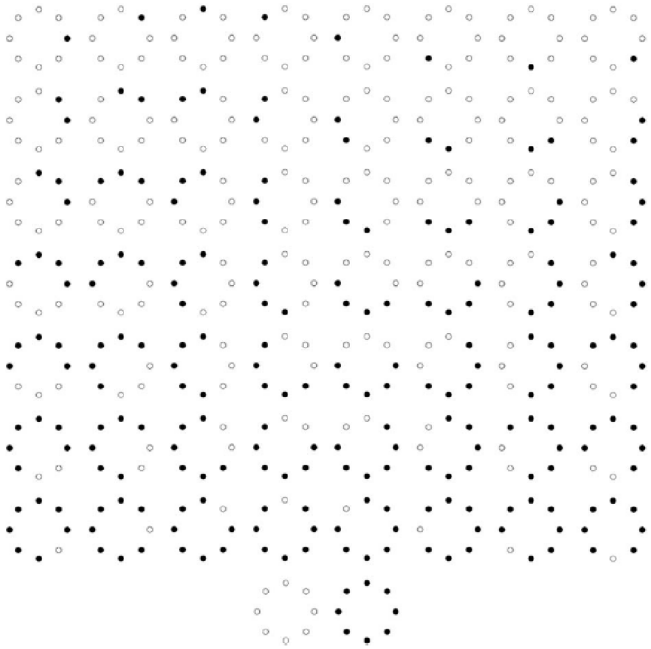
Toto je základní a nejjednodušší varianta LBP, která má však několik nevýhod. Tou největší je především rotační neinvariantnost. Dalším problémem je, že některé LBP vzory se v texturách prakticky nevyskytují, a proto jsou sloupce histogramu, které je představují, tak nízké, že jsou statisticky zanedbatelné a pouze ztežují klasifikaci. Je tedy třeba nějakou modifikací zajistit rotační invariantnost a snížení počtu sloupců histogramu při zachování stejného počtu bodů, z nichž budou LBP příznaky počítány.

Uniformní LBP

Vzhledem k tomu, že výskýt některých LBP vzorů v textuře je prakticky zanedbatelný, lze použít pouze jejich jistou podmnožinu. K výpočtu optimální podmnožiny slouží metoda zvaná *beam search*, která je však výpočetně velmi náročná, a proto se v praxi nepoužívá. Místo ní se použijí pouze LBP vzory, v nichž je počet přechodů mezi *0* a *1* menší nebo roven dvěma.

Počtem přechodů je myšlen počet změn hodnoty v LBP reprezentované binárním číslem. Například v osmibitovém kódu *00001111* jsou právě dvě změny.

Uniformní LBP ponechává právě jen takovéto kódy a jejich cyklicky rotované varianty. Všechny ostatní hodnoty se sečtou do posledního sloupce histogramu.



Obrázek 2.6: Uniformní vzorky LBP pro P = 8. Převzato z [19].

Rotačně invarinatní LBP

Další modifikací základního LBP je jeho rotačně invariantní varianta. Tato varianta zohledňuje to, že při reálné práci s texturami nebudou vstupní obrázky vždy stejně natočeny. Při použití základní varianty LBP na jeden obrázek s různou rotací, budou se získané histogramy poměrně významně lišit. Přitom se LBP příznak při použití na natočený obrázek také pouze kruhově pootočí, jak je znázorněno na obrázku 2.7.

Z toho vyplývá, že rotační invariance LBP, lze docílit pouze bitovou rotací LBP. Dosahuje se cyklickou bitovou rotací původního vzorku pro všechna možná natočení a následným výběren toho s nejnižší hodnotou.

Protože tato metoda mapuje několik původních texturních vzorů na jeden, snižuje se počet sloupců histogramu.

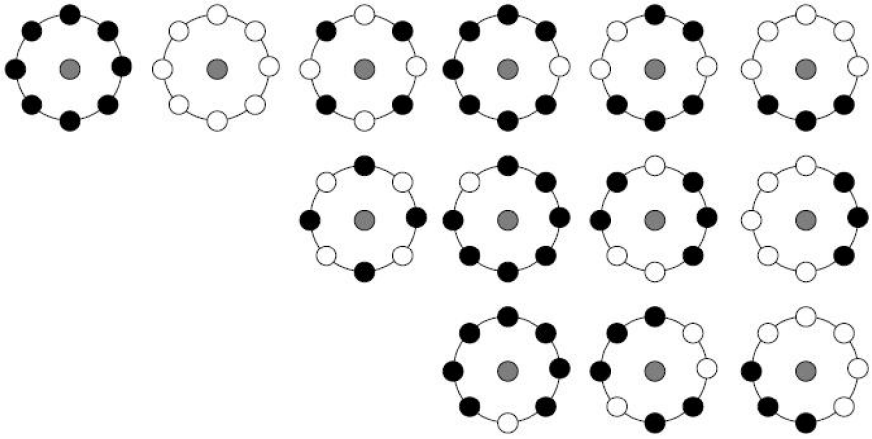
Barvy v LBP

LBP se ve své základní formě počítají z intenzity jasu, kterou lze vypočítat následovně:

$$I = 0.299 * R + 0.587 * G + 0.114 * B \tag{2.9}$$

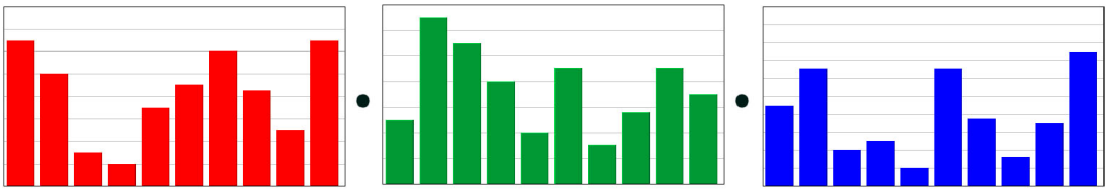
Kde I je nová intenzita jasu pixelu, R je intenzita červeného subpixelu, G intenzita zeleného subpixelu a B intenzita modrého subpixelu.

Tímto postupem ale přijdeme o část informace, a to konkrétně o barvu, která by mohla být také využita. Z tohoto důvodu vznikly CLBP, což jsou LBP, které nepracují s intenzitou



Obrázek 2.7: Šest různých LBP vzorů a jejich možné rotace. Převzato z [29].

jasu, ale přímo s barvou pixelu. CLBP zpracovávají pro každý barevný kanál LBP odděleně. To znamená, že pro každý barevný kanál vznikne histogram četnosti výskytu jednotlivých LBP vzorů. Výsledný příznakový vektor se potom vytvoří konkatenací dílčích histogramů jednotlivých barevných kanálů [43].



Obrázek 2.8: Histogramy četnosti výskytu jednotlivých LBP počítané pro jednotlivé barevné kanály zvlášť. Výsledný histogram vznikne jejich konkatenací.

Další možností rozšíření LBP o barvu jsou OCLBP. Které vytvářejí nejen pro každý barevný kanál samostatný příznakový vektor, ale navíc i příznakové vektory, které berou středový bod z jednoho barevného kanálu a okolní body z jiného. V tříkanálovém barevném prostoru vznikne tedy devět kombinací jak vytvořit příznakový vektor. V barevném prostoru RGB s kanály označenými R, G a B lze příznakový vektor se středovým bodem z R a okolními body z G označit jako $P(R,G)$. Možné kombinace pro RGB barevný prostor jsou znázorněny v tabulce 2.1.

		okolní body		
		R	G	B
středový bod	R	$P(R,R)$	$P(R,G)$	$P(R,B)$
	G	$P(G,R)$	$P(G,G)$	$P(G,B)$
	B	$P(B,R)$	$P(B,G)$	$P(B,B)$

Tabulka 2.1: Všechny možné kombinace příznakového vektoru OCLBP v barevném prostoru RGB.

Lze si však všimnou, že vždy dvojice příznakových vektorů typu $P(R,G)$ a $P(G,R)$

nese redundantní informace, takže z ní stačí pouze jeden. Po eliminaci redundantních dvojic zůstane pouze šest příznakových vektorů na místo původních devíti při minimální ztrátě přenášené informace. Těchto šest kombinací může odpovídat například 2.2.

		okolní body		
		R	G	B
středový bod	R	$P(R,R)$	$P(R,G)$	$P(R,B)$
	G		$P(G,G)$	$P(G,B)$
	B			$P(B,B)$

Tabulka 2.2: Možné kombinace příznakového vektoru OCLBP v barevném prostoru RGB bez redundantních informací.

Obdobně lze tyto příznakové vektory vytvořit i pro ostatní vícekanálové barevné prostory. Pro všechny barevné prostory je však nutné, aby hodnoty obsažené v jednotlivých kanálech byly ze stejné intervalu. U některých barevných prostorů (například HSV nebo HSL) tomu tak implicitně není, a proto je nutné hodnoty nejdříve normalizovat. Získané příznakové vektory jsou samozřejmě závislé na použitém barevném prostoru.

Kontrast v LBP

Jednou z podstatných vlastností LBP je, že nenese žádnou informaci o kontrastu. Důvod je patrný už z principu extrakce LBP, kdy se hodnoty okolí prahují hodnotou středu, ale to, o kolik jsou nižší respektive vyšší, už nijak nepostihují. Z tohoto důvodu vznikla metoda nesoucí název *Rotation invariant variance measures* neboli VAR [19]. Tato metoda si klade za cíl postihnou právě lokální kontrast daného bodu. Její hlavní myšlenkou je kruhově symetrické rozmístění určitého počtu bodů do okolí a následný výpočtet rozptylu jejich hodnot. Má stejně jako LBP dva parametry: počet bodů okolí P a vzdálenost těchto bodů od středu R. Mějme P okolních bodů $g_1, \cdots, g_P$, potom můžeme jejich průměr značený u vypočítat:

$$u = \frac{\sum_{n=1}^{P} g_n}{P} \tag{2.10}$$

Následně lze VAR vyjádřit takto:

$$VAR_{P,R} = \frac{\sum_{n=1}^{P} (g_n - u)^2}{P} \tag{2.11}$$

Výsledkem VAR je tak reálné číslo a nikoli jako u LBP jedna z X hodnot. Proto je nutné výsledek VAR kvantovat tak, aby byla kvantovaná hodnota co nejpřesnější, ale zároveň musí VAR nabývat co nejmenšího počtu hodnot kvůli délce histogramu četnosti. LBP a VAR se vzájemně doplňují a příznakový vektor tvořený zároveň LBP i VAR může lépe charakterizovat lokální texturu. Pokud se k popisu okolí bodu využívá příznakový vektor tvořený právě LBP společně s VAR volí se obvykle u těchto dvou metod stejné P i R, aby bylo možné konkrétní kombinaci LBP/VAR přiřadit sémantický význam.

Další možností, jak spojit LBP s kontrastem, je metoda LBPV. Ta je založena na předpokladu, že oblasti s vyšším kontrastem mají vyšší šanci rozlišovat rozdílné regiony, a proto by měly mít větší váhu, než oblasti s nižším kontrastem. Při běžné extrakci LBP se po

vypočtení LBP hodnoty pro daný bod přičte jedna do sloupce histogramu, který odpovídá dané hodnotě. Naproti tomu v LBPV se po vypočtení LBP přičte do odpovídajícího sloupce VAR, čímž dosáhneme větší váhy pro oblasti s vyšším kontrastem. Pokud i respektive j jsou souřadnice bodu v obraze a $LBPV_{P,R}(k)$ označíme k-tý sloupec histogramu, můžeme LBPV zapsat následovně:

$$LBPV_{P,R}(k) = \sum_{i=1}^{N} \sum_{j=1}^{M} w\left(LBP_{P,R}(i,j), k\right) \ , \ k \in [0, K] \tag{2.12}$$

$$w\left(LBP_{P,R}(i,j), k\right) = \begin{cases} VAR_{P,R}(i,j) & LBP_{P,R}(i,j) = k \\ 0 & \text{jinak} \end{cases} \tag{2.13}$$

Takto získaný příznakový vektor je opět nutné normalizovat.

Vlastnosti LBP

LBP je na výpočet poměrně nenáročná, přičemž její úspěšnost je velmi dobrá. Ostatní metody pro získání texturních příznaků s obdobnou úspěšností jsou časově daleko náročnější. Navíc některé varianty LBP jsou robustní vůči natočení, zpracovávají i kontrast nebo pracují s barvou. Z těchto důvodů jsem si vybral k získávání texturních příznaků v této práci právě LBP.

2.5 Využití textury

V počítačové grafice existuje hned několik problémů spojených s texturou.

Prvním z nich je klasifikace obrazu. Klasifikací obrazu se myslí přiřazení významu nějakým obrazovým datům. Obvykle se to provádí zařazením do určité informační třídy. Typickým využitím klasifikace obrazu na základě textury je zpracováním medicínských dat.

Textura se také používá k syntéze povrchu objektu na základě texturního vzoru, kdy se z malého vzorku textury vytvoří dostatečně velká plocha k pokrytí celého povrchu požadovaného tělesa.

Další využitím může být určení geometrie povrchu tělesa. Pokud je na povrchu tělesa nanesena pravidelná textura a tento povrch je zdeformován, tak lze na základě změny textury určit geometrii povrchu.

Pomocí textury můžeme také obraz segmentovat, respektive ho rozdělit na části, které mají stejnou texturu. A právě na segmentaci obrazu na základě textury se bude tato práce specializovat.

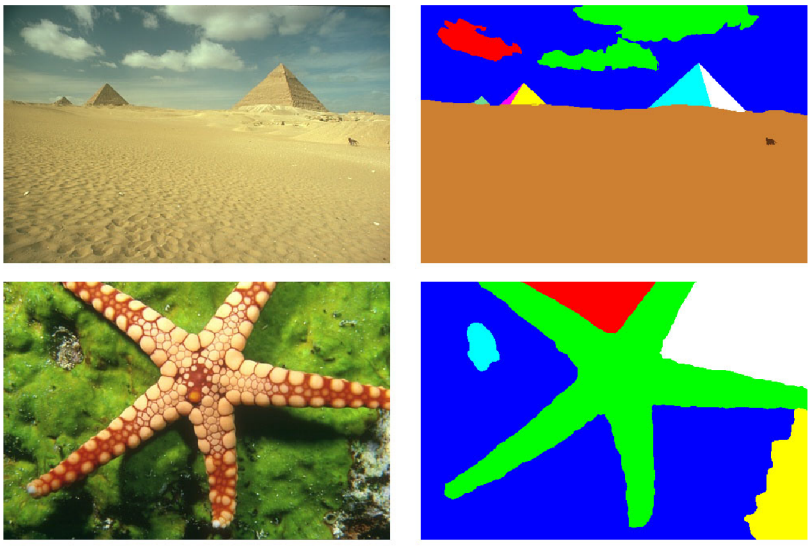
Kapitola 3

Segmentace obrazu

Segmentace obrazu je jedním ze základních a nejfrekventovanějších problémů počítačového vidění. Je to jeden z prvních kroků analýzy obrazu. Využívá se v mnoha oborech lidské činnosti, jako je automatická kontrola kvality výrobků, zpracování leteckých nebo satelitních snímků používaných v zemědělství, meterologii nebo vojenství. Další oblastí, kde se využívá segmentace obrazových dat, je medicína. Tady je důležitou podporou diagnózy srdce, ledvin a mnoha dalších orgánů.

Segmentací obrazových dat se myslí rozdělení obrazu na několik nepřekrývajících se částí, přičemž body z jedné části spadají do jednoho sémantického celku, zatímco body ze dvou různých oblastí nikoliv. Segmentace obrazu se také využívá k oddělení důležité informace od pozadí. Potom ale může nastat problém s interpretací obrazu jako celku, protože při různých účelech je onou důležitou informací něco jiného[44].

Segmentace může být buď úplná nebo částečná. Při úplné segmentaci jednotlivým částem obrazu připadají přímo konkrétní objekty reálného světa. Zatímco částečná segmentace rozdělí obraz pouze na významně části, které jsou poté dále analyzovány.



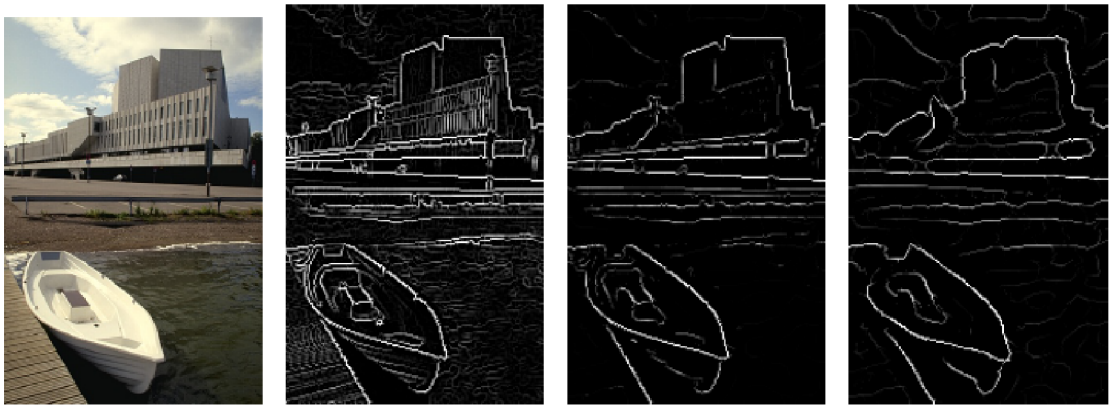
Obrázek 3.1: Ukázka segmentace obrazu. Vpravo původní obrázky ze sady *BSDS300* a vlevo jejich segmentace.

3.1 Základní přístupy

Existuje mnoho metod segmentace obrazu, které lze rozdělit do několika skupin:

Metody vycházející z detekce hran

Tyto metody využívají skutečnosti, že hrany většinou oddělují jednotlivé objekty nebo alespoň jejich části. Jejich nevýhodou je možnost přerušení nebo poškození důležité hrany, takže jiné nepodstatné hrany mohou být daleko výraznější. Při takové segmentaci se nejprve použije hranový detektor a poté se ztenčí tlusté hrany. Následuje prahování hran, při kterém se odstraní všechny velmi slabé hrany a také středně silné, pokud nejsou spojeny s nějakou velmi silnou hranou. "Opraví"se porušené hrany a nakonec se hrany spojí. Příkladem metody založené na detekci hran jsou takzvané *active contures*.



Obrázek 3.2: Postup segmentace založené na detekci hran. Zdroj [36]

Metody založené na regionech obrazu

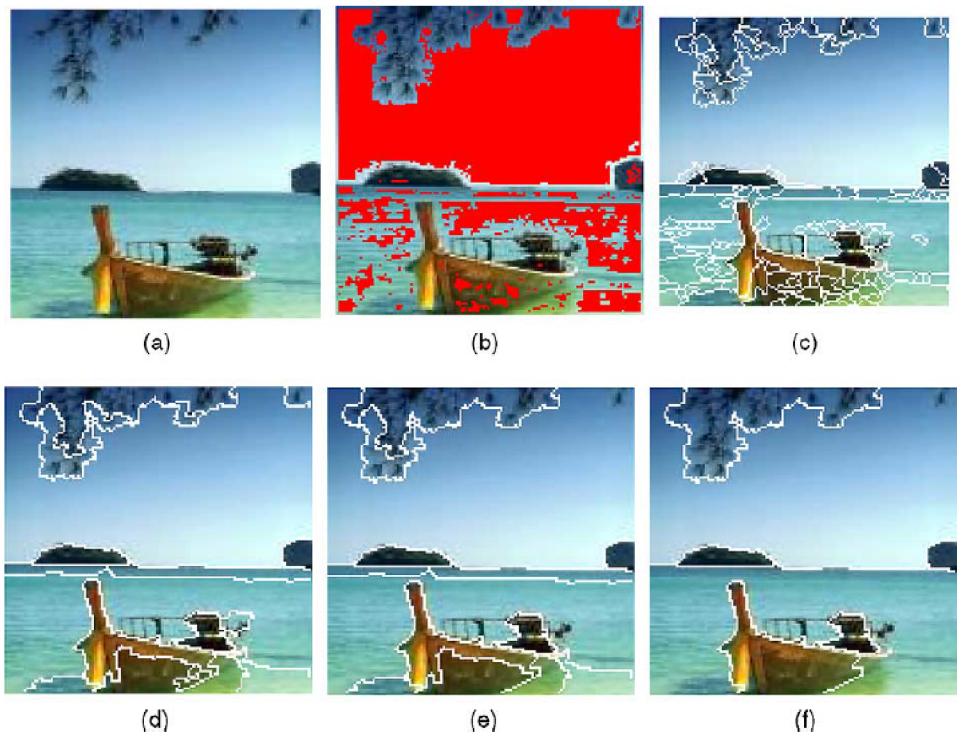
Fungují tak, že je stanoveno jisté kritérium homogenity, které musí splňovat všechny body patřící do jedné části(regionu) obrazu. Nejdříve se použije jedna z metod na rozdělení obrazu na regiony, například Region Growing nebo Blob Coloring a poté se podobné regiony sloučí. Obraz rozdělený do jednotlivých regionů se reprezentuje maticově, hierarchicky nebo pomocí grafu.

Statické metody

Jsou například prahování nebo shluková analýza. Shlukování se také nazývá **Clustering**. Na techniky shlukování se celá tato práce zaměřuje, a proto jsou podrobně rozebrány v kapitole 4.

Další metody

Existuje ještě několik dalších metod segmentace, které nelze zařadit do žádné z předchozích kategorií (někdy se také nazývají hybridní). Samostatnou kategorii pak také tvoří *Neuronové sítě*.



Obrázek 3.3: Postupné kroky segmentace pomocí Region Growing, kde a) je originální obrázek a f) výsledná segmentace. Převzato z [35]

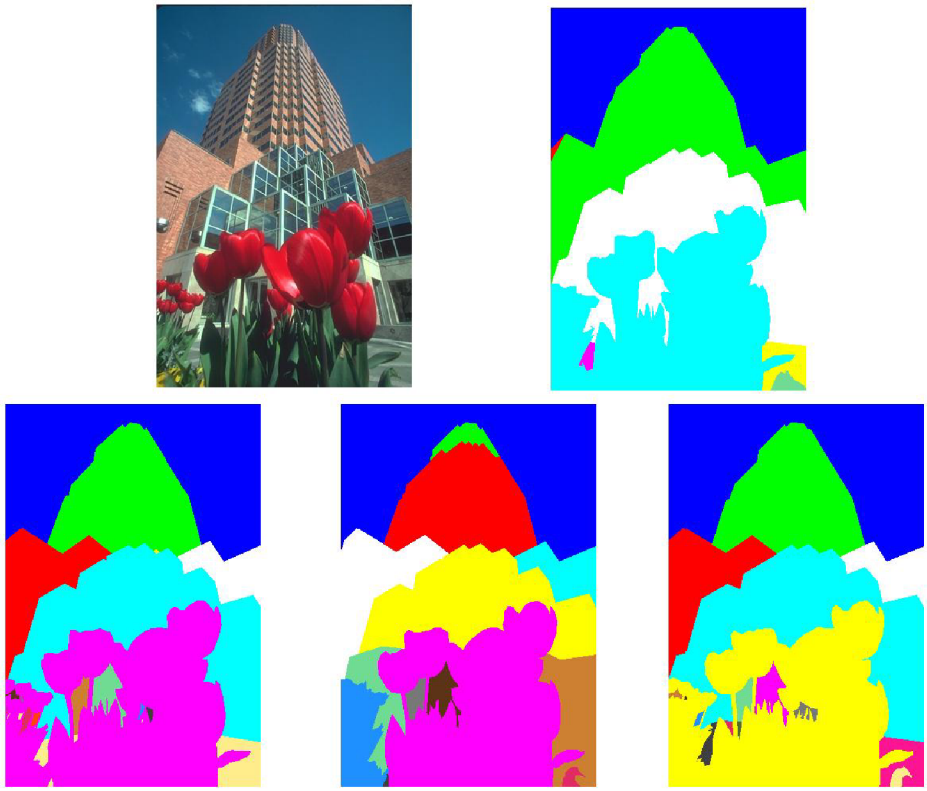
3.2 Metodiky hodnocení úspěšnosti segmentace

Ačkoliv existuje mnoho metod segmentace obrazu, metodik hodnocení její úspěšnosti je mnohem méně. Hodnocení úspěšnosti segmentace probíhá často pouze intuitivně pomocí několika segmentovaných obrázků, což je velice subjektivní. Hlavním problémem hodnocení úspěšnosti segmentace reálných obrázku je neexistence unikátního správného řešení, se kterým by se dal výsledek segmentace porovnat [39]. Pokud ručně anotujeme původní obrázek, dostaneme *ground truth*, které ale může být pouze subjektivní. Protože když ten samý obrázek bude anotovat nezávisle několik různých lidí, dostaneme pouze "částečně podobné", nikoliv však shodné výsledky. To souvisí s faktem, že obrázek si může interpretovat každý člověk jinak. Proto je nejlepší možností, jak hodnotit úspěšnost segmentace, metrika pracující s několika mírně se lišícími ručně anotovanými *ground truth* každého obrázku.

Existují čtyři kategorie hodnocení úspěšnosti segmentace [38]:

- **Odlišení regionů** - Tyto metody počítají míru překrytí regionů segmentovaného obrázku s regiony *ground truth*. Důležitým aspektem těchto metod je, jakou měrou budou penalizovány rozdíly nepřekrývajících se regionů. Mezi takovéto metody patří *Global Consistency Error* (GCE) a *Local Consistency Error* (LCE) [26].

- **Hledání hranic** - Problém segmentace obrazu na jednotlivé regiony lze převést na hledání hranic mezi nimi. Uspěšnost segmentace se pak počítá jako míra překrytí hranic. Tyto metody jsou však velmi citlivé na malé rozdíly, a proto mohou špatně hodnotit i segmentace, které se zdají vizuálně správné.



Obrázek 3.4: Originální obrázek ze sady *BSDS300* (vlevo nahoře) a jeho čtyři rozdílné ručně vytvořené ground truth.

- **Informační teorie** -Vypočítají míru informačního obsahu v segmentovaném obraze a v *ground truth*. Poté zjistí, kolik informací poskytuje jedna o druhé. Příkladem takové metriky je *variation of information* (VI), která za míru informačního obsahu považuje podmíněnou entropii labelů jednotlivých pixelů v obraze.
- **Neparametrické testy** - Statistické testy počítající páry pixelů, jejichž labely k sobě mají stejný vztah. Následně vyhodnocují, jestli k sobě má daná dvojice pixelů stejný vztah v segmentovaném obraze a v *ground truth*. Vztahem může být myšleno například to, zda jsou pixely v jednom regionu či nikoliv. Mezi takovéto metody patří *Cohen's Kappa* nebo *Rand Index*.

Následující metody počítají s tím, že každý pixel segmentovaného obrazu náleží do jednoho regionu. Přičemž jednotlivé regiony *ground truth* nemají sémantický význam, a proto jsou jejich labely vzájemně zaměnitelné.

Cohen's Kappa

Cohen's Kappa je metrika měřící podobnost mezi dvěma hodnotiteli. Přičemž se předpokládá ,že každý hodnotitel rozděluje N objektů do M vzájemně se vylučujících tříd [13]. Pokud za objekty považuji N pixelů v obraze, které chci rozdělit do M regionu, prvním hodnotitelem bude *ground truth* a druhým výsledek segmentace, lze *Cohen's Kappa* použít jako metriky hodnotící úspěšnost segmentace.

Mějme M tříd $C_0,\ C_1, \ldots, C_M$. Nechť $p_{i,j}$ je poměr objektů, které jsou prvním hodnotitelem zařazeny do třídy C_i a druhým hodnotitelem do třídy C_j. Poměr objektů zařazených do třídy i prvním hodnotitelem lze zapsat jako:

$$p_{i*} = \sum_{j=1}^{M} p_{i,j} \tag{3.1}$$

Obdobně poměr objektů zařazených do třídy j druhým hodnotitelem bude:

$$p_{*j} = \sum_{i=1}^{M} p_{i,j} \tag{3.2}$$

Poměr objektů zařazených oběma hodnotiteli do stejné třídy p_0.

$$p_0 = \sum_{i=1}^{M} p_{i,i} \tag{3.3}$$

A očekávaná šance na shodu p_c:

$$p_c = \sum_{i=1}^{M} p_{i*} p_{*j} \tag{3.4}$$

Dostaneme výsledný vzorec pro výpočet *Cohen's Kappa*:

$$\kappa = \frac{p_0 - p_c}{1 - p_c} \tag{3.5}$$

Přičemž κ je reálné číslo od nuly do jedné a za velmi dobrý výsledek se považuje $\kappa > 0.75$.

Rand Index

Převádí problém dvou různých rozdělení N bodů do M regionů na problém porovnání vztahů dvojice pixelů v těchto rozděleních [34]. Výsledek segmentace nazvu S a *ground truth* G. N bodů označím jako $x_1, x_2, \ldots, x_N$, jejich labely vzniklé segmentací S budou označeny $l_{S1}, l_{S2}, \ldots, l_{SN}$ respektive $l_{G1}, l_{G2}, \ldots, l_{GN}$, pokud se bude jednat o labely *ground truth*. *Rand Index* pak lze zapsat:

$$R\left(S,G\right) = \frac{1}{\binom{N}{2}} \sum_{i=1,\ j=1, i\neq j}^{M} \left[\prod(l_{Si} = l_{Sj} \bigwedge l_{Gi} = l_{Gj}) + \prod(l_{Si} \neq l_{Sj} \bigwedge l_{Gi} \neq l_{Gj})\right] \tag{3.6}$$

$\prod$ je funkce identity, která nabývá hodnoty jedna v případě, že je její podmínka splněna, jinak nula.

$$\prod(true) = 1 \quad , \quad \prod(false) = 0 \tag{3.7}$$

Pokud je n_{uv} počet bodů majících label u v S a zároveň label v v G, lze počet bodů mající label u v S označit jako n_{u*}, který se vypočte:

$$n_{u*} = \sum_{v} n_{uv} \tag{3.8}$$

Obdobně n_{*v} značí počet bodů mající label v v G:

$$n_{*v} = \sum_{u} n_{uv} \tag{3.9}$$

Díky tomu lze *Rand Index* přepsat do výpočetně mnohem méně náročné podoby:

$$R\left(S,G\right)=1-\frac{0.5*\left(\sum_u {n_{u*}}^2+\sum_v {n_{*v}}^2\right)-\sum_{u,v}{n_{uv}}^2}{\binom{N}{2}} \tag{3.10}$$

Výstupem *Rand Indexu* je reálné číslo od nuly do jedné, přičemž vyšší číslo znamená větší podobnost porovnávaných rozdělení. V případě, kdy je jedním z porovnávaných rozdělení *ground truth*, značí vyšší číslo větší podobnost požadovaného výsledku s výstupem segmentace a tedy její vyšší úspěšnost.

3.3 F-measure

F-measure je jedna z nejpoužívanějších metod pro měření uspěšnosti klasifikace objektů [24]. Měření úspěšnosti klasifikace probíhá porovnáním s referenčním řešením, o kterém předpokládáme, že je správné a chceme se mu tedy co nejvíce přiblížit. Pokud klasifikuji objekty jedné třídy, rozhoduji pouze, zda daný objekt do této třídy patří. Celkově lze dosáhnou jen čtyř možných stavů [21]. Dvou, pokud referenční řešení do dané třídy patří, a další dvou, pokud nikoliv. Tyto stavy lze označít následovně:

- **True positive** - Pokud je objekt klasifikován do dané třídy, přičemž v referenčním řešení do této třídy také patří.
- **False positive** - Pokud je objekt klasifikován do dané třídy, přičemž v referenčním řešení do této třídy nepatří.
- **True negative** - Pokud objekt není klasifikován do dané třídy, přičemž do ní nepatří ani v referenčním řešení.
- **False negative** - Pokud objekt není klasifikován do dané třídy, přičemž v referenčním řešení do této třídy patří.

Tyto stavy jsou také znázorněny v tabulce 3.1.

		referenční rešení	
		true	false
výsledek klasifikace	true	true positive	false positive
	false	false negative	true negative

Tabulka 3.1: Tabulka stavů vzniklých porovnáním výsledku klasifikace s referenčním řešením.

Precision (preciznost) P je definována jako poměr objektů, které jsou do konkrétní třídy správně klasifikovány ke všem do této třídy klasifikovaným objektům. Bude P zapsáno následovně:

$$P=\frac{true\ positive}{true\ positive+false\ positive} \tag{3.11}$$

Recall (úplnost) R je definována jako poměr objektů, které jsou do konkrétní třídy správně klasifikovany ke všem objektům do této třídy patřícím.

$$R=\frac{true\ positive}{true\ positive+false\ negative} \tag{3.12}$$

Z [12] vyplývá, že obecný vzorec pro *F-measure* je:

$$F_\beta = \frac{(\beta^2 + 1) * P * R}{\beta^2 * P + R} \quad \beta \in < 0, \infty > \tag{3.13}$$

Přičemž parametr β určuje, jestli bude výsledné F více ovlivňovat *precision* nebo *recall*. Pokud je $\beta > 1$ výsledek ovlivňuje větší měrou *recall*, zatímco $\beta < 1$ znamená, že je F závislejší na *precision*. Jestliže $\beta = 1$, stane se *F-measure* harmonickým průměrem P a R. V tomto případě se také nazývá F_1-*measure* a může být zapsán:

$$F = \frac{2 * P * R}{P + R} \tag{3.14}$$

F-measure v segmentaci obrazu

Protože v segmentaci rozdělujeme obraz obecně do M regionu, nelze *F-measure* na výsledek použít přímo. Existují však dva způsoby, jak výsledek segmentace upravit respektive interpretovat, tak abych mohl vypočítat *F-measure*.

1. **Mapa hranic regionů** - Výsledek segmentace je převeden na mapu hranic regionů [25]. Tato mapa má stejnou velikost jako segmentovaný obrázek a obsahuje reálná čísla od nuly do jedné, přičemž vyšší číslo značí vyšší pravděpodobnost výskytu hranice mezi regiony. *Ground truth* je také převedeno na mapu hranic, která ale obsahuje jen hodnoty jedna, pokud na příslušném místě leží hranice, nebo nula pokud nikoli. Jestliže máme více *ground truth*, jsou do mapy zahrnuty hranice ze všech. Mapu hranic výsledku segmentace poté prahujeme, aby obsahovala pouze nuly a jedničky jako je tomu u mapy *ground truth*. Z těchto dvou map lze poté snadno vypočítat *precision* respektive *recall* a následně také požadované *F-measure*. Jediným problémem je zvolení optimální hodnoty pro prahování mapy výsledku segmentace. Proto se tato mapa prahuje K-krát s různými hodnotami prahu, čímž nám vznikne K hodnot *F-measure*, z kterých se vybere ta nejvyšší jako výsledná úspěšnost segmentace.

2. **Dvojice pixelů** - Při této interpretaci počítáme dvojice pixelů v obraze, které patří do stejného regionu [16].

   Pokud je výsledek segmentace nazván S,*ground truth* G a to, že daná dvojice pixelů patří do stejného regionu, se považuje za *true*:

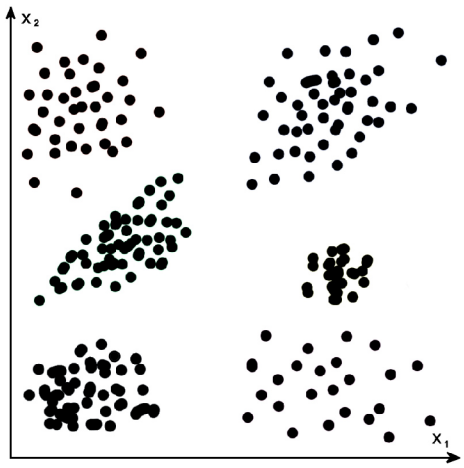
   - **True positive** - Dvojice pixelů klasifikována do stejného regionu v S i G.
   - **False positive** - Dvojice pixelů klasifikována do stejného regionu v S, ale do jiného v G.
   - **True negative** - Dvojice pixelů klasifikována do jiného regionu v S i G.
   - **False negative** - Dvojice pixelů klasifikována do jiného regionu v S, ale do stejného regionu v G.

   Dosazením do vzorce 3.11, lze vypočítat *precision*. *Precision* je možné v tomto případě interpretovat jako pravděpodobnost, že dvojice pixelů, která je klasifikována do stejného regionu v segmentovaném obraze, do stejného regionu skutečně patří. Obdobně lze dosazením do vzorce 3.12, vypočítat *recall*. Který v tomto případě značí pravděpodobnost, že dvojice pixelů patřící do stejného regionu bude do stejného regionu klasifikována i v segmentovaném obraze. Z *precision* a *recall* již není problém dopočítat pomocí vzorce 3.14 výslednou *F-measure*.

Kapitola 4

Techniky shlukování

Shlukování neboli clustering je obecná metoda k slučování objektů na základě podobnosti. Segmentaci obrazu si lze také představit jako klasifikační problém, kdy je obraz složen z **R** regionů, přičemž každý pixel patří právě do jednoho z těchto regionů [45] [23]. Pixel je přitom reprezentován N-rozměrným vektorem čísel, neboli příznakovým vektorem. Hodnoty v něm mohou nést například informace o barvě, jasu nebo okolí aktuálního pixelu. Typ a způsob získání těchto informací by měl být volen s ohledem na způsob použití tak, aby byly příznakové vektory pro pixely z jednoho regionu "podobnější", než vektory pro pixely z regionů odlišných. Čísla v jednotlivých vektorech mohou být považována za souřadnice bodů v N-rozměrném prostoru. Každému pixelu z původního obrazu tedy odpovídá jeden bod v tomto N-rozměrném příznakovém prostoru. Body patřící pixelům z jednoho regionu vytvářejí shluky, přičemž by měl jeden shluk bodů reprezentovat právě jeden region obrazu.

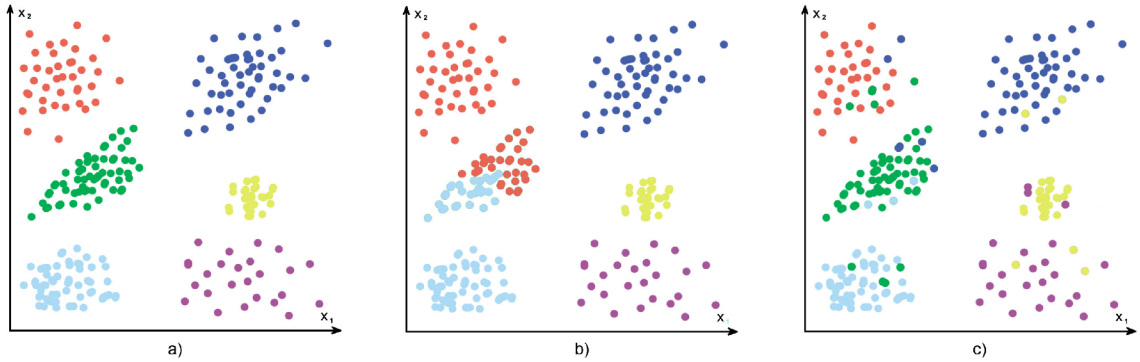


Obrázek 4.1: Body v dvourozměrném příznakovém prostoru rozmístěné tak, že shluky, které tvoří, jsou jednoznačně oddělitelné.

Výsledky shlukování

Segmentace obrazu pomocí shlukování příznakových vektorů pixelů má dvě hlavní úskalí - nalezení optimálního počtu shluků a správné rozřazení bodů do těchto shluků. Různé situace, které mohou nastat, budou demonstrovány na následujících obrazcích. Pro názornost

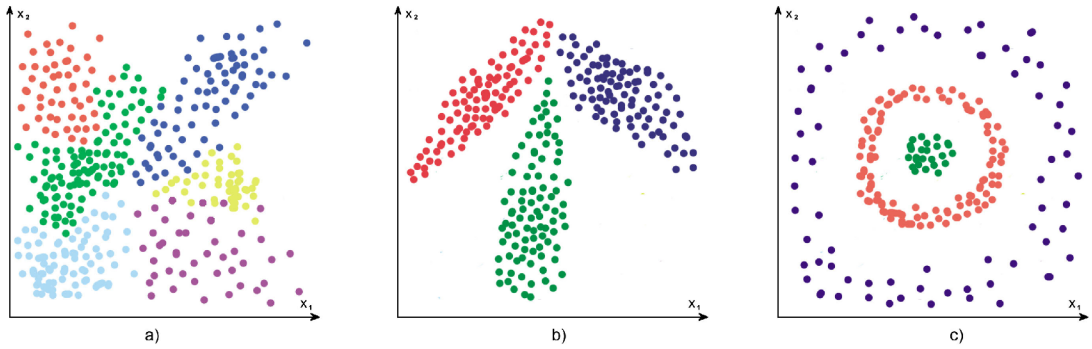
jsou zvoleny body v dvourozměrném příznakovém prostoru, přičemž body náležící pixelům klasifikovaným do jednoho regionu mají stejnou barvu.



Obrázek 4.2: Různé výsledky shlukování pro: a) správný počet shluků a všechny body správně klasifikovány b) špatný počet shluků c) správný počet shluků, ale některé body nesprávně klasifikovány

Na obrázcích 4.2 byly od sebe shluky snadno odlišitelné. Bohužel toto je pouze ideální stav, ke kterému se chceme vhodným výběrem extrakce příznaků co nejvíce přiblížit.

Při extrakci reálných příznaků se většinou setkáváme s tím, že shluky nemají stejnou velikost, tvar, nebo jsou zatíženy šumem. Velmi často také nastává situace, při níž nejsou dva blízko sebe ležící shluky jednoznačně oddělitelné, takže klasifikace bodů ležících okolo této hranice může být nejednoznačná. Některé z těchto případů jsou zobrazeny na obrázku 4.3.



Obrázek 4.3: Možné rozložení bodů v prostoru: a) nejednoznačně oddělitelné shluky b) podlouhlý tvar shluků c) shluky se stejným středem

Metriky vzdálenosti

Základním prvkem všech shlukovacích metod je metrika měřící "rozdílnost"příznakových vektorů, respektive vzdálenost bodů v prostoru příznaků(feature space). Pokud tuto metriku označíme D a příznakové vektory $\vec{x}$ respektive $\vec{y}$, bude výsledná vzdálenost $D(\vec{x}, \vec{y})$. Takováto metrika musí splňovat následující podmínky:

$$D(\vec{x}, \vec{y}) \geq 0 \ \ \forall \vec{x}, \forall \vec{y} \tag{4.1}$$

$$D(\vec{x}, \vec{y}) = D(\vec{y}, \vec{x}) \ \ \forall \vec{x}, \forall \vec{y} \tag{4.2}$$

$$D(\vec{x}, \vec{x}) = 0 \ \ \forall \vec{x} \tag{4.3}$$

To znamená, že je vždy nezáporná (záporná vzdálenost bodů by neměla smysl), symetrická a vzdálenost bodu od sebe samého je nulová.

Takovýchto metrik je mnoho, proto je v tabulce 4.1 uvedeno pouze několik vybraných:

Metriky vzdálenosti	
Euclidean	$D(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
Bhattacharyya	$D(\vec{x}, \vec{y}) = -\ln\left(\sum_{i=1}^{n} \sqrt{x_i * y_i}\right)$
Hellinger	$D(\vec{x}, \vec{y}) = \sqrt{1 - \sum_{i=1}^{n} \sqrt{x_i * y_i}}$
Normalized Euclidean	$D(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{{s_i}^2}}$
Manhattan	$D(\vec{x}, \vec{y}) = \sum_{i=1}^{n} \mid x_i - y_i \mid$
χ^2	$D(\vec{x}, \vec{y}) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i}$

Tabulka 4.1: Přehled metrik, které je možné použít k výpočtu vzálenosti dvou bodů. s_i je i-tá položka vektoru s, který je směrodatnou odchylkou množiny vstupních bodů X.

4.1 K-Means

Základním shlukovacím algoritmem je *k-means* [5]. Tento algoritmus potřebuje znát počet shluků před svým spuštěním - to znamená, že není sám schopný tento počet určit(narozdíl

třeba od Mean-Shiftu algoritmu).

Pokud je příznakový vektor považován za souřadnici bodu v d-rozměrném prostoru, může být každý shluk zastoupen svým středem, tzn. opět bodem v tomto prostoru. Pokud je počet shluků k, lze jejich středy označit $w_1, w_2, ...w_k$, vstupní body $i_1, i_2, ..., i_n$,množinu C_j všech bodů patřící do shluku w_j, bude se chybová funkce E počítat následovně:

$$E = \sum_{j=1}^{k} \sum_{i_p \in C_j} \mid i_p - w_j \mid^2 \tag{4.4}$$

Minimální hodnota chybové funkce označuje nejlepší shlukování bodů. Tato chybová funkce je vlastně součet vzdáleností nadruhou, všech bodů od středů svých shluků (within-cluster sum of squares).

Na začátku algoritmus inicializuje středy shluků na náhodně vybrané vstupní body:

$$w_j = i_p \; j \in \langle 1, k \rangle \; p \in \langle 1, n \rangle \tag{4.5}$$

Poté se pro všechny vstupní body vypočte vzdálenost $D(i_p, w_j)$ od jednotlivých středů shluků a přiřadí se do toho, ke kterému je vzdálenost nejnižší.

$$min(D(i_p, w_j)) \; j \in \langle 1, k \rangle \Rightarrow i_p \in C_j \tag{4.6}$$

Následně vypočteme novou polohu středů shluků ze všech bodů, které do něj náleží, přičemž $\mid C_j \mid$ je počet vstupních bodů v shluku C_j, následnovně:

$$w_j = \frac{\sum\limits_{i_p \in C_j} i_p}{\mid C_j \mid} \tag{4.7}$$

Rozřazení bodů do shluků a následný výpočet jejich nových středů se opakuje, dokud rozdíl chybových funkcí ve dvou po sobě následujících krocích není nižší něž požadovaná přesnost ϵ.

$$\mid E_l - E_{l+1} \mid < \epsilon \tag{4.8}$$

Chybová funkce E je monotonicky klesající [6], a tak k-means konverguje pouze do lokálního minima, které však nemusí být i minimem globálním. Výsledek je tedy závislý na počáteční inicializaci středů. Proto se běžně algoritmus spouští mnohokrát s různými počátečními inicializacemi a vybírá se pouze nejlepší dosažený výsledek (s nejnižší hodnotou E).

k-means++

Algoritmus *k-means++* představený v [6] je vylepšením klasického *k-means*. Zatímco v původním *k-means* se při inicializaci středů shluků vybírá ze vstupních bodů s uniformní pravděpodobností, *k-means++* nejdříve pro každý bod i_p stanoví jeho pravděpodobnost výběru za střed shluku jako ϕ_p. Pokud se vzdálenost bodů i_p od nejbližšího středu shluku označí $D(i_p)$, lze pravděpodobnost výběru tohoto bodu za střed shluku vyjádřit následnovně:

$$\phi_p = \frac{D(i_p)^2}{\sum\limits_{j=1}^{n} D(i_j)^2} \tag{4.9}$$

Celý algoritmus potom probíhá tak, že se vybere první střed shluku ze všech vstupních bodů s uniformní pravděpodobností. Další středy jsou potom vybrány již podle pravděpodobnosti ϕ_p, dokud jich není předem požadovaný počet. ϕ_p se samozřejmě při výběru každého dalšího středu shluku mění, protože pokud přibude nový shluk, změní se pro část vstupní bodů, vzdálenost k nejbližšímu středu. Po inicializaci se tento algoritmus neliší od původního *k-means*.

Neuniformní výběr středů trvá sice déle než uniformní, zato se ale významně sníží počet iterací nutných k nalezení lokálního minima, a tak se i sníží doba celkového běhu algoritmu. Podle [6] se také zlepší úspěšnost shlukování.

4.2 Fuzzy C-Means

Tento algoritmus, který byl představen v [15] a později vylepšen [9], je ze skupiny fuzzy algoritmů, jejichž hlavní vlastností je to, že každý vstupní bod může patřit do více shluků najednou (soft-clustering). Narozdíl od většiny ostatních metod, kde každý vstupní bod patří právě do jednoho shluku (hard-clustering). *Fuzzy C-Means* neboli *FCM* je velmi často používaný shlukovací algorimus pro segmentaci obrazových dat, který ale před svým spuštěním potřebuje znát počet shluků, do kterých má vstupní body rozřadit.

Pokud N vstupních bodů označených jako x_i , $i \in \langle 1, N \rangle$ budeme shlukovat do C shluků, jejichž středy jsou pojmenovány c_j , $j \in \langle 1, C \rangle$, lze příslušnost bodu x_i do shluku c_j zapsat jako u_{ij}, čímž vznikne matice příslušností U o rozměrech N x C.

Principem *FCM* je minimalizace objektové funkce definované:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} {u_{ij}}^m * \|x_i - c_j\|^2 \quad m \in \langle 1, \infty \rangle \tag{4.10}$$

Kde m je reálné číslo označované jako fuzzifikační koeficient, zatímco $\|x - y\|$ je normovaná vzdálenost mezi body x a y.
Přičemž hodnoty v matici příslušností mají následující dvě omezení:

$$u_{ij} \geq 0 \quad \forall j \ , \ j \in \langle 1, C \rangle \ ; \ \forall i \ , \ i \in \langle 1, N \rangle \tag{4.11}$$

$$\sum_{j=1}^{C} u_{ij} = 1 \quad \forall j \ , \ j \in \langle 1, C \rangle \tag{4.12}$$

Algoritmus pak probíhá tak, že se inicializují středy shluků na náhodně vybrané vstupní body a z těchto středů se vypočítají jednotlivé prvky matice příslušnosti.

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}} \tag{4.13}$$

Následně jsou z matice příslušnosti vypočteny nové středy shluků.

$$c_j = \frac{\sum_{i=1}^{N} {u_{ij}}^m * x_i}{\sum_{i=1}^{N} {u_{ij}}^m} \tag{4.14}$$

Tento postup se opakuje, dokud je rozdíl mezi maticemi příslušnosti ve dvou po sobě jdoucích krocích, menší než požadovaná přesnost ϵ.

Celý postup lze tedy shrnout do následujících bodů:

1. Náhodná inicializace středů shluků C^0, $i = 0$
2. Vypočtení matice příslušnosti U^0 z aktuálních středů C^0
3. Výpočet nové matice středů C^{i+1} z matice příslušnosti U^i
4. Výpočet nové matice příslušnosti U^{i+1} z matice středů C^{i+1}
5. Pokud je $\|U^{i+1} - U^i\| > \epsilon$ inkrementuj i a vrať se na bod 3, jinak algoritmus skončí a vrací U^{i+1} a C^{i+1} jako výsledek

Kde i je počítadlo kroků algoritmu, zatímco U^i a C^i jsou matice příslušnosti respektive matice středů shluků v i-tém kroce.

Jak je dokázáno v [32] FCM konverguje pouze do lokálního minima, které však nemusí být zároveň globálním extrémem. To, v jakého lokálním extrému algoritmus skončí, závisí na počáteční inicializaci středů shluků.

Počet shluků a stabilita FCM

Jak již bylo řečeno, algoritmus FCM není stabilní, to znamená, že pokud ho se spustí vícekrát pro stejná vstupní data, nemusí vždy vyjít stejný výsledek. Další nevýhodou FCM je, že pro svůj chod potřebuje znát počet shluků, do kterých má rozdělit vstupní data. Obě tyto nevýhody jdou řešit pomocí *Cluster validity indexu*, nebo vhodnou inicializací(bude popsána níže).

Cluster validity index

Cluster validity index je metrika hodnotící správnost shlukování pouze na základě jeho výstupů a vstupních dat. To znamená, že pracuje bez znalosti dalších informací, jakými by bylo u segmentace obrazu například ground truth. Existují i *Cluster validity indexy* pracujících s fuzzy hodnotami. Některé z nich, jako například *Partition coefficient* [7] nebo *Partition entropy* [8], používají pouze hodnoty z matice příslušnosti. Jsou však i další metody využívající kromě výstupu shlukování(v případě FCM jsou jimi matice příslušnosti a matice středů shluků) i vstupní data. Mezi ně patří Xie-Beni index [42], Fukuyama-Sugeno index [17], PCAES index [41] nebo CWB index [33] index. Pro všechny uvedené metriky platí, že čím je jejich hodnota nižší, tím lepší je shlukování, s výjimkou *Partition coefficientu* a PCAES indexu, kde je lepší vyšší hodnota.

Přehled vzorců pro uvedené metriky je v tabulce 4.2.

Cluster validity indexy	
Partitoin coefficient	$\frac{1}{N}\left(\sum_{i=1}^{n}\sum_{j=1}^{C} {u_{ij}}^2\right)$
Partition entropy	$\frac{1}{N}\left(\sum_{i=1}^{n}\sum_{j=1}^{C} {u_{ij}}^2 * \log_a u_{ij}\right)$
Xie-Beni	$\frac{\left(\sum_{i=1}^{n}\sum_{j=1}^{C} {u_{ij}}^2 * \|x_i - c_j\|\right)}{N * min\,(c_j - c_i)}$
Fukuyama-Sugeno	$\sum_{i=1}^{n}\sum_{j=1}^{C} {u_{ij}}^2 * (\|x_i - c_j\| - \|c_j - \vec{x}\|)$
$PCAES$	$\sum_{j=1}^{C} \sum_{i=1}^{n}\sum_{j=1}^{C} \frac{{u_{ij}}^2}{u_M} - \exp\left(\frac{-min\{\|c_i - c_k\|^2\}}{\beta_T}\right)$
CWB	$\alpha * Scat\,(c) + Disc\,(c)$

Tabulka 4.2: Přehled *Cluster validity indexů* pracujících s fuzzy množinami. $\vec{x}$ je průměr středů shluků, u_M je maximální velikost shluku na druhou, β_T je průměrná vzdálenost středů shluků od středu všech bodů . $Scat$ značí průměrný rozptyl bodů uvnitř shluků, $Disc$ je celkový rozptyl mezi shluky a α odpovídá hodnotě $Disc$ pro C_{max}.

Pokud tedy pomocí nějakého *Cluster validity indexu* můžeme vypočítat míru správnosti shlukování, můžeme pomocí ní také porovnat dva rozdílné výsledky shlukování. Algoritmus pro nalezení správného počtu shluků pak vypadá následovně:

1. Stanovíme maximální a minimální počet shluků C_{max} respektive C_{min}
2. Spustíme $C_{max} - C_{min}$ instancí FCM, přičemž jednotlivé instance budou mít rozdílný počet shluků v intervalu od C_{min} do C_{max}
3. Vypočítáme *Cluster validity index* pro každou instanci FCM
4. Počet shluků, který má instance s nejlepším *Cluster validity indexem*, je považován za výsledek.

Nestabilitu FCM lze řešit obdobně, tzn. že se spustí několik instancí FCM se stejným počtem shluků, ale s rozdílnou počáteční inicializací. Výstup instance s nejlepším *Cluster validity indexem* je považována za správný.

Inicializace

Klasický FCM algoritmus je inicializován na náhodně vybrané vstupní body. A právě to je důvod jeho nestability. Znamená to ale, že se zde také nachází prostor k odstranění tohoto nedostatku. Navíc lze vhodnou inicializací zvýšit rychlost FCM, snížením počtu iterací algoritmu nutných k dosažení lokálního minima.

Předem musím říci, že neexistuje univerzální řešení, které by vždy zaručovalo optimální inicializaci, ale existuje několik meta-heuristických algoritmů, jež se snaží dosáhnout řešení, které by se optimálnímu co nejvíce přibližovalo. Tyto algorimy pracují na principu projití celého prostoru všech možných řešení a nalezení toho nejlepšího, přičemž se snaží uniknout z lokálních extrémů.

Mezi tyto meta-heuristiky patří *Simulované žíhání*, které je pro inicializaci FCM použito v [2] a [40]. *Genetické algoritmy* jsou využity ve spojení s FCM v [20], [3] nebo [27]. Dalšími použitelnými meta-heuristikami jsou *tabu search* [1], *fuzzy ants colony*[22] nebo *harmonic search* [4].

Všechny výše uvedené metody dokáží, dle uvedených materiálů, zvýšit stabilitu FCM a tím i jeho celkovou úspěšnost. Přičemž algoritmy [40] a [27] dokonce mohou prohledávat prostor řešení obsahující různý počet shluků a jejich výsledkem je tedy i správný počet shluků.

Výkonnost všech uvedených meta-heuristických algoritmů je samozřejmě velmi ovlivněna heuristickou funkcí, kterou používají. Touto funkcí je zpravidla *Cluster validity index*. Pro různé datové sady se ale může optimální *Cluster validity index* lišit, a tak je jeho výběr klíčový.

Meta-heuristiky, které budou použity v rámci této práce, budou rozebrány ještě v kapitole 5.2.

Inicializace pomocí Simulovaného žíhání

Metoda Simulovaného žíhání byla využita k inicializaci FCM v [40]. Simulované žíhání je meta-heuristická metoda procházející prostor řešení a hledající nejlepší možné. Pracuje s aktuálním řešením, ze kterého náhodně zvolenou upravou vytvoří kandidátní řešení. Kandidátní řešení je přijato za nové aktuální řešení, pokud je jeho energie nižší než energie současného řešení. Pokud je energie kandidátního řešení vyšší než aktuální, je nové řešení přijato s pravděpodobností závislou na rozdílu jejich energií a aktuální teplotě. Teplota se postuně snižuje a s ní se snižuje i pravděpodobnost přijetí řešení s vyšší energií.

Tento postup lze upravit a použít k nalezení optimální sady středů shluků a tím i stanovení správného počtu shluků pro FCM. Jestliže je řešením nazvána sada středů shluků, lze jeho energii vypočítat pomocí *Cluster Validity Indexu*. Tato metoda používá metriku *Xie-Beni* k výpočtu energie řešení. *Xie-Beni* používá k výpočtu matici příslušnosti U a sadu středů shluků C. *Xie-Beni* lze vypočítat následovně:

$$XB = \frac{\left(\sum_{i=1}^{n} \sum_{j=1}^{C} {u_{ij}}^2 * \|x_i - c_j\| \right)}{N * min\,(c_j - c_i)} \tag{4.15}$$

Inicializace probíhá tak, že se nejdříve náhodně vygeneruje počet shluků v intervalu od dvou do $\sqrt{N}$ a středy těchto shluků jsou inicializovány na náhodně vybrané vstupní body, čímž vznikne první aktuální řešení C_a. Následně je toto řešení ohodnoceno pomocí *Xie-Beni* indexu a označeno F_a. Nejlepší dosažené řešení C_n se nastaví na aktuální řešení C_a a ohodnocení nejlepšího řešení F_n na hodnotu F_a. Maximální teplota T_{max} je nastavena na aktuální teplotu T_a.

Dalším krokem je výpočet velikosti jednotlivých shluků v aktuálním řešení. Velikost j-tého shluku se značí $\mid c_j \mid$ a lze ji vypočítat:

$$\mid c_j \mid = \sum_{i=1}^{N} u_{ij} \tag{4.16}$$

Poté se vygeneruje kandidátní řešení z aktuálního. Protože řešením je sada středů, jejichž počet se může měnit, je provedeno generování kandidátního řešení C_k náhodným výběrem jednoho z těchto kroků:

- **Posunutí středu** - Pozice středu nejmenšího shluku z C_a je posunuta:

  $$v_k[d] = v_a[d] + \alpha \tag{4.17}$$

  Kde $v_k[d]$ je d-tá dimenze nového středu a $v_a[d]$ je d-tá dimenze původního středu. α je posunutí definované:

  $$\alpha = \beta * rand\ ,\ \beta \in < -0.007, 0.007 >\ ,\ rand \in < 0, 1 > \tag{4.18}$$

- **Rozdělení shluku** - Největší shluk z C_a je rozdělen na dva. Polohu nových středů lze získat přičtením respektive odečtením vzdálenosti vybraného středu v_a od referenčního bodu x_r:

  $$v_k[d] = v_a[d] \pm \mid v_a[d] - x_r[d] \mid \tag{4.19}$$

  Podle aktuální situace je zvolen referenční bod. První situace nastane, pokud není největší shluk jednoznačně oddělen od ostatních. To znamená, že existují body s příslušností k největšímu shluku okolo hodnoty 0.5. V takovéto situaci je považován za referenční bod jeden ze vstupních bodů, který má příslušnost k danému shluku co nejblíže 0.5 ale přitom nižší než tato hodnota. Druhá situace nastane, pokud je největší shluk jednoznačně oddělen od ostatních. To znamená, že existují body s příslušností do největšího shluku výrazně větší než 0.5 a ostatní mají příslušnost do tohoto shluku menší než 0.4.

- **Odstranění shluku** - Odstraníme nejmenší shluk z C_a.

Toto kandidátní řešení je ohodnoceno pomocí *Xie-Beni* a hodnotu indexu označíme F_k. Pokud je kandidátní řešení lepší než aktuální ($F_k < F_a$), bude nové řešení přijmuto za aktuální. Poté se ověří, zda je nové aktuální řešení lepší než nejlepší prozatím dosažené řešení C_n, pokud tomu tak opravdu je, nahradí se nejlepší dosaženené řešení aktuálním.

Pokud je však kandidátní řešení horší než aktuální ($F_k > F_a$), je přijato za aktuální s pravděpodobností $p(C_k)$:

$$p(C_k) = \exp\left(-\frac{F_k - F_a}{T_a}\right) \tag{4.20}$$

Upraví se aktuální teplota:

$$T = T * \vartheta \ , \ \vartheta \in < 0, 1 > \tag{4.21}$$

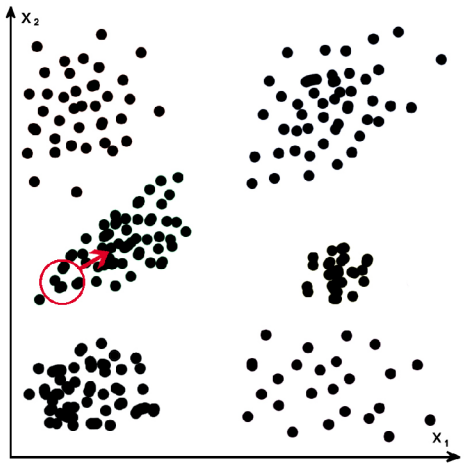
Kde ϑ je koeficient udávající, jak prudce se bude snižovat teplota. Poté se zkontroluje, zda aktuální teplota T_a neklesla pod minimální teplotu T_{min} nebo není překročen maximální počet iterací algoritmu. Obě tyto události by vedly k jeho ukončení. Pokud však ani jedna z těchto událostí nenastane, je vygenerováno nové kandidátní řešení a celý cyklus se znovu opakuje.

Po skončení poslední iterace je k inicializaci FCM použito nejlepší dosažené řešení C_n.

4.3 Mean-Shift

Mean-Shift je neparametrický algoritmus sloužící k iterativnímu nalezení lokálního maxima hustoty bodů v prostoru [14]. Základním principem je zjištění hustoty bodů nalézejících se v okolí aktuálně zpracovávaného bodu. Za předpokladu, že hustota bodů vzrůstá směrem ke středu shluku, lze pomocí gradientu hustoty zjistit, jakým směrem se posunout blíže lokálnímu maximu hustoty. Po provedení posunu je znovu vypočten gradient hustoty a takto algoritmus dokonverguje až do místa s maximální lokální hustotou. Toto místo je potom střed shluku pro bod, v němž jsme začali.

Pokud je tento algoritmus spuštěn samostatně pro každý bod v obraze, získáme střed shluku pro všechny body v obraze. Všechny body, pro které algoritmus dokonvergoval do stejného lokálního maxima, patří do jednoho shluku. Už z principu tedy vyplývá, že *Mean-Shift* nepotřebuje pro své fungování dopředu znát počet shluků, což je jednou z jeho velkých výhod. Protože je tedy shluk v podstatě množina bodů, které dokonvergovaly do stejného maxima, mohou mít shluky různý tvar. Což u některých jiných metod, které používají explicitní reprezentaci shluku, není možné.



Obrázek 4.4: Okolí bodu a směr gradientu.

Hustota bodů v $f(x)$ v okolí velikosti h, pro N bodů x_1, x_2 až x_n v d-rozměrném prostoru, lze vyjádřit pomocí *kernelu* $K(x)$:

$$f(x) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \tag{4.22}$$

Radiálně symetrický kernel lze definovat pomocí *profilu kernelu* k(x) pro nezáporné x následnovně:

$$K(x) = c_k * k\left(\|x^2\|\right) \tag{4.23}$$

c_k je kladná normalizační konstanta. Lokální extrémy rovnice jsou tam, kde je její první derivace rovna nule. Pokud do rovnice 4.22 dosadíme 4.23, derivujeme a výsledek upravíme, dojdeme k:

$$'f_{h,K}(x) = \frac{2c_{k,d}}{nh^{d+2}}[\sum_{i=n}^{N} g(||\frac{x-x_i}{h}||^2)][\frac{\sum_{i=1}^{N} x_i g(||\frac{x-x_i}{h}||^2)}{\sum_{i=n}^{N} g(||\frac{x-x_i}{h}||^2)} - x] \tag{4.24}$$

Zde nás bude zajímat hlavně druhá část tohoto výrazu, což je vektor směřující vždy k lokálnímu maximu hustoty bodů v prostoru:

$$m_{h,G}(x) = \frac{\sum_{i=1}^{N} x_i g(||\frac{x-x_i}{h}||^2)}{\sum_{i=n}^{N} g(||\frac{x-x_i}{h}||^2)} - x \tag{4.25}$$

Kapitola 5

Návrh algoritmu segmentace obrazu

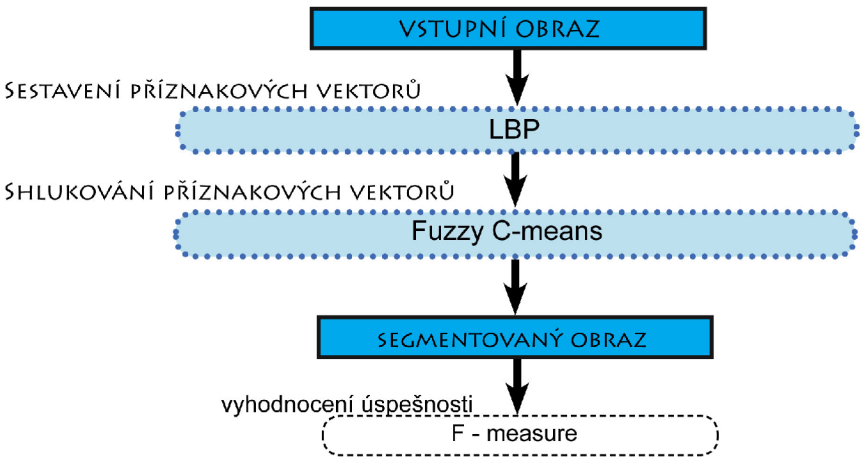
Cílem této práce je vybrat vhodné texturní příznaky a na jejich základě provést segmentaci neznámého obrazu. V podstatě jde o dvě dílčí podúlohy. Těmi je získání optimální sady texturních příznaků a jejich následné shlukování. Tato práce si také klade za cíl srovnat několik druhů texturních příznaků a metod shlukování použitých pro segmentaci obrazu.

Pro získání texturních příznaků byla vybrána metoda Local Binary Patterns (LBP). Tato metoda má několik parametrů, variant a vylepšení, jejichž vhodnost pro segmentaci obrazu bude jedním z výsledků této práce.

Pro účely shlukování byly zvoleny algoritmy *Fuzzy C-Means*, *Mean-Shift*, *K-means* a *K-means++*. Přičemž se podrobně se zaměřím na *Fuzzy C-Means*, pro který navrhnu několik modifikací, jejichž úspěšnost je předmětem testování.

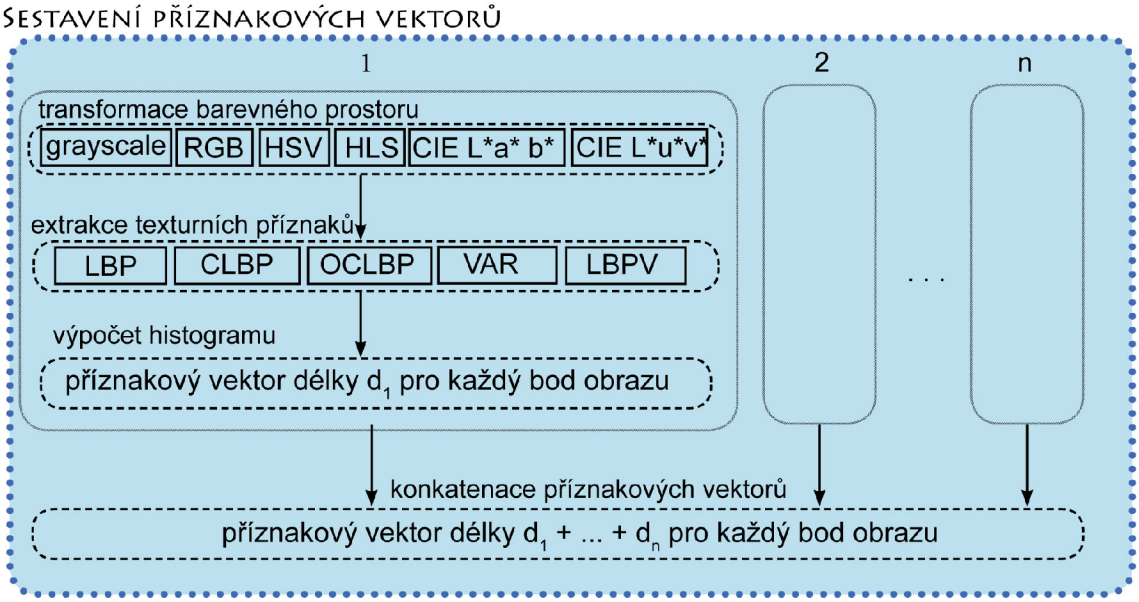
Následuje vyhodnocení úspěšnosti segmentace pro daný obrázek, pomocí porovnání s ručně anotovaným vzorem. Na závěr se vyhodnotí úspěšnost segmentace pro celou datovou sadu.

Součástí práce je samozřejmě i implementace programu provádějící segmentaci obrazu na základě texturních příznaků.



Obrázek 5.1: Celkový diagram navrženého systému.

5.1 Extrakce texturních příznaků



Obrázek 5.2: Schéma extrakce příznakových vektorů. Příznakový vektor je sestaven konkatenací dílčích příznakových vektorů. V diagramu označuje čárkovaný rámeček krok postupu, který bude realizován pro jednu z variant v plném rámečku, které obsahuje.

Jak již bylo řečeno v předchozím textu, k extrakci texturních příznaků byla vybrána metoda Local Binary Patterns. Jako základní metoda extrakce bude použita její uniformní a rotačně invariantní varianta. Základními parametry všech variant LBP jsou vzdálenost okolních bodů (R) a jejich počet(P). Konkrétním nastavením těchto parametrů se zabývá kapitola 7.2.

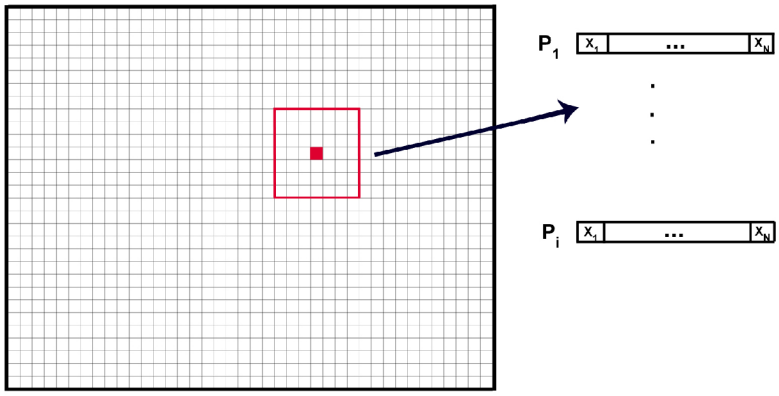
Základní postup

Příznakovým vektorem u LBP je histogram četnosti výskytu jednotlivých LBP vzorů v celé textuře. Protože však realizuji segmentaci obrazu shlukováním příznakových vektorů, potřebuji příznakový vektor pro každý bod obrazu.

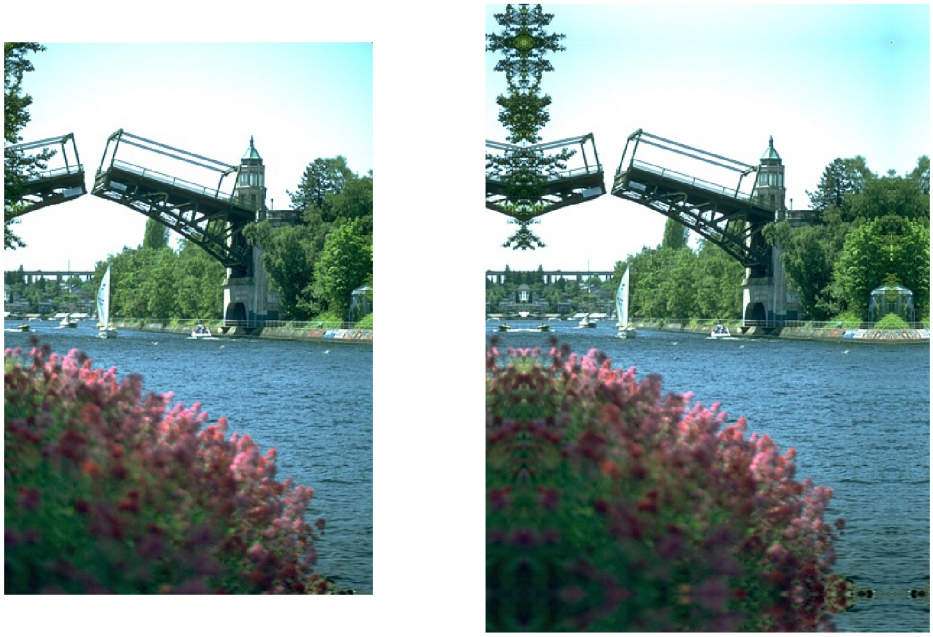
Toho lze dosáhnout vytvořením extrakčního okna velikosti d, v jehož středu bude aktuálně zpracovávaný bod. Následným vypočtením histogramu četnosti výskytu LBP pro body v tomto okně vznikne příznakový vektor aktuálního bodu. Posunem extrakčního okna a opakováním předchozího postupu vzniknou příznakové vektory pro všechny body obrazu.

Pro i pixelů v obraze bude i příznakových vektorů označených P_0 až P_i, kdy každý příznakový vektor má n položek označených x_1 až x_n. Postup extrakce ilustruje obrázek 5.3.

Problém nastane na krajích obrazu, kdy se část okna dostane mimo obraz. Chybějící body získám ozrcadlením okrajů. To znamená, že obraz je na každé straně zvětšen o polovinu velikosti extrakčního okna ($\frac{d}{2}$) a až poté se v něm počítají příznakové vektory. Ozrcadlení obrazu je zobrazeno na obrázku 5.4.



Obrázek 5.3: Postup extrakce texturních příznaků z obrazu.



Obrázek 5.4: Originální obraz ze sady *BSDS300* a jeho druhá verze s ozrcadlenými okraji.

Dále je příznakový vektor normalizován celkovým počtem bodů které obsahuje:

$$x'_n = \frac{x_n}{\sum_{j=1}^{n} x_j} \tag{5.1}$$

Kde x_n je je n-tá položka původního příznakového vektoru, x'_n je n-tá položka normalizovaného příznakového vektoru.

Tímto postupem jsem vytvořil sadu normalizovaných příznakových vektorů délky n pro jedno nastavení parametrů P a R. Délka příznakového vektoru n je pro, v této práci používanou, uniformní a rotačně invariantní variantu LBP:

$$n = P + 2 \tag{5.2}$$

Velikost sady příznakových vektorů samozřejmě odpovídá počtu klasifikovaných bodů v obraze.

Multiresolution LBP

Nevýhodou takto extrahované sady texturních příznaků je, že obsahuje pouze informace o okolí bodů ve vzdálenosti R. Tento problém řeší *Multiresolution LBP*, které předcházející postup několikrát zopakují pro různé vzdálenosti R a dílčí výsledky konkatenují. Tímto postupem sice získám informace o okolí bodu v různých vzdálenostech, ale zvětší se i délka příznakových vektorů. Protože zároveň s ní roste časová i paměťová náročnost celé segmentace obrazu a navíc může příliš velká délka příznakových vektorů negativně ovlivnit výsledek shlukování, je nutné při výběru jednotlivých parametrů *Multiresolution LBP* dbát i na výslednou délku příznakového vektoru. Konkrétní parametry použité při extrakci *Multiresolution LBP* obsahuje kapitola 7.2.

Velikost a tvar extrakčního okna

Velikost extrakčního okna je důležitým parametrem extrakce texturních příznaků. Příliš velké okno totiž poškodí detaily, zatímco příliš malé okno může vést k přesegmentaci obrazu. Navíc body ze dvou různých textur mohou mít při nevhodné velikosti velikost extrakčního okna velmi podobné příznakové vektory, obdobně body ze stejné textury mohou mít pro nevhodou velikost okna velmi rozdílné příznakové vektory. Podobností respektive rozdílností příznakových vektorů je myšlena vzdálenost v konkrétní zvolené metrice.

Protože velikost extrakčního okna nelze dopředu spolehlivě určit, budeme v této práci používát alespoň experimentálně zjištěnou horní mez velikosti okna:

$$d < \frac{a}{10} \tag{5.3}$$

a je délka kratší strany vstupního obrazu.

U každého experimentu bude uvedena použitá velikost extrakčního okna.

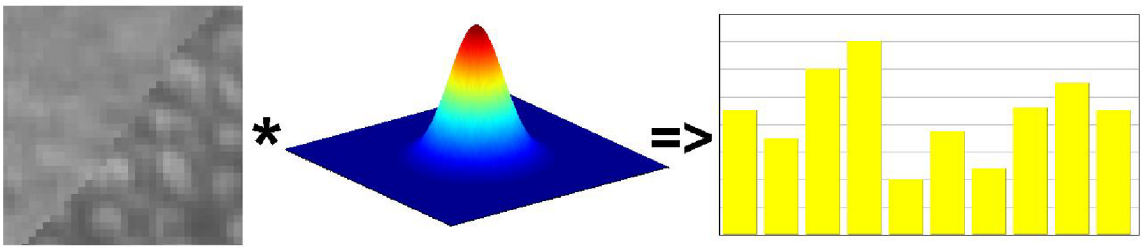
Tato práce se mimojiné zabývá také dvěma modifikacemi týkajícími se extrakčního okna. První z nich zkoumá jeho tvar, zatímco druhá váhou jednotlivých bodů.

Kruhové extrakční okno

Příznakový vektor je u LBP tvořen histogramem četnosti jednotlivých LBP vzorů v extrakčním okně se středem v aktuálním bodě. Běžně se pro extrakční okno používá čtvercový tvar. Pokud ale vyjdeme z předpokladu, že okolí bodu by mělo být tvořeno všemi body se vzdáleností nižší než d, bude mít extrakční okno kruhový tvar. Ostatní kroky extrakce zůstanou stejné.

Váhování 2D Gaussovou funkcí

Druhou navrženou modifikací týkající se extrakčního okna je váhování bodů. Hlavní myšlenkou je, že body umístěné blíže aktuálnímu bodu by měly mít vyšší váhu než ty vzdálenější. Pro váhování bodů jsme zvolili 2D Gaussovu funkci posunutou do středu extrakčního okno. Váhování bude provedeno pomocí předpočítané matice koeficientů. kterou se budou body v extrakčním okně násobit.Do odpovídajícího sloupce výsledného histogramu se přičte koeficient z předpočítané matice namísto jedničky. Zbytek extrakce pak zůstane nezměněn.



Obrázek 5.5: Vynásobení LBP hodnot v extrakčním okně 2D Gaussovou funkcí, čímž vznikne upravený histogram četnosti výskytu jednotlivých druhů LBP v gaussovském okolí.

Barvy v LBP

Základní metoda LBP pracuje s obrazem ve stupních šedi. Proto se práce zabývá i variantami LBP modifikovanými pro práci s barevným obrazem. Konkrétně půjde o metody CLBP a OCLBP, popsané v kapitole 2.4.

Další věcí spojenou s využitím barvy v LBP je barevný prostor. Vliv použitého barevného prostoru na výslednou segmentaci obrazu bude předmětem testování pro obě varianty LBP pracující s barvou.

Konkrétně se budeme zabývat barevnými prostory *RGB*, *HSV*, *HLS*, *CIE L* a* b**,*CIE L* u* v**, *CIE XYZ* a *YCrCb*.

LBPV a VAR

Poslední testovanou modifikací LBP je jejich rozšíření informací o lokálním kontrastu. První z testovaným možností je zkombinování LBP hodnoty a VAR hodnoty. Druhou pak použití LBPV namísto LBP. Metody LBPV i VAR jsou podrobněji rozebrány v kapitole 2.4.

Protože je výsledkem VAR reálná hodnota, je k vytvoření histogramu četnosti jejich výskytu nutné VAR hodnoty kvantovat. Zde je však nutné stanovit optimální počet kvantovacích hladin. Kvantování na vyšší počet kvantovacích hladin je sice přesnější, ale s počtem hladin roste i počet sloupců histogramu VAR, což je pro následné shlukování nevýhoda. Proto bude součástí testů i stanovení vhodného počtu těchto hladin.

Kvantování na k kvantizačních hladin provedu následovně:

1. Vypočítám VAR hodnotu pro každý bod v obraze.
2. Všechny VAR hodnoty vzestupně seřadím do pole P, které bude mít N prvků $P[1]$ až $P[N]$.
3. Kvantizační hladinu L_x potom stanovím:

$$L_x = P\left[x * (\frac{1}{k} * N)\right] \quad x \in < 1\ ,\ k > \tag{5.4}$$

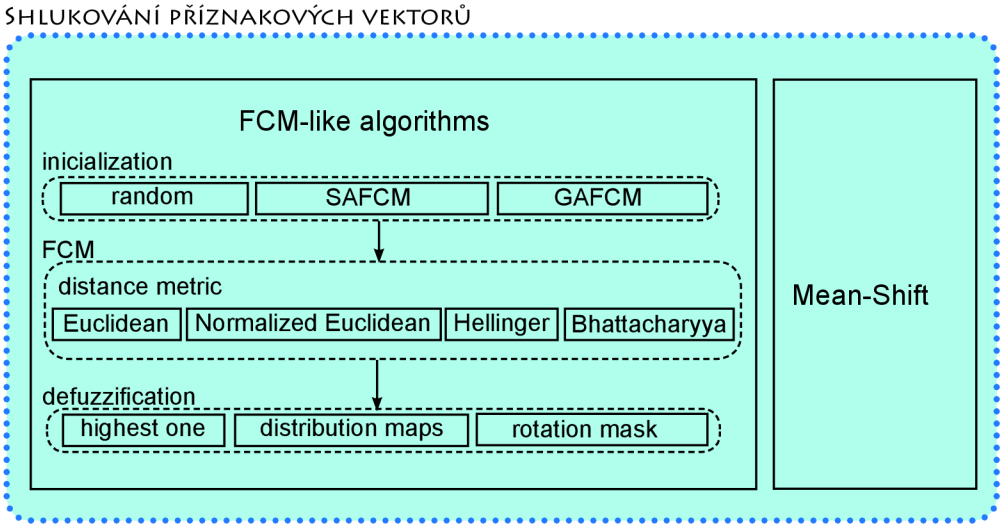
4. Všechny na začátku vypočtené VAR převedu na nejbližší hladinu, vyšší než je původní VAR hodnota.

Zkombinování LBP a VAR budeme provádět konkatenací obou histogramů. Při kombinování LBP a VAR vždy použiji pro obě metody stejné P a R.

U LBPV není kvantování díky jejímu principu nutné.

Obě zmíněné metody budou testovány a jejich úspěšnost porovnána se základní variantou LBP.

5.2 Shlukování



Obrázek 5.6: Diagram shlukování příznakových vektorů.

Shlukování příznakových vektorů je druhým krokem postupu segmentace. Jeho výsledek by měl být takový, že jeden shluk příznaků odpovídá jednomu regionu v segmentovaném obraze.

Ke shlukování texturních příznaků použiji především algoritmus Fuzzy C-Means a několik jeho navržených modifikací. Dále budou pro srovnání uvedeny i výsledky dosažené pomocí K-Means, K-Means++ a Mean-Shift. Všechny tyto metody jsou podrobně popsány v kapitole 4.

Fuzzy C-Means

Fuzzy C-Means je hlavním shlukovacím algoritmem použitým v této práci. Při jeho zkoumání jsem se zaměřil především na možnosti inicializace FCM, použitou metriku vzdálenosti a defuzifikaci výsledků FCM.

Původní algoritmus FCM potřebuje dopředu znát správný počet shluků, do kterého má příznakové vektory roztřídit. Protože tento počet nemusí být dopředu známý, je nutné ho nejprve odhadnout. Jako první určím interval, do kterého správný počet shluků spadá. V

této práci použijeme následující interval počtu shluků:

$$< 2\ ,\ 10 > \tag{5.5}$$

První z možností, jak dále pokračovat ,je provedení FCM pro každý počet shluků z předchozího intervalu. Následně ohodnotit pomocí *Cluster Validity Indexu* 4.2 všechny dosažené výsledky a nejlepší z nich považovat za správný. Tento postup dále značím jako *All Cluster Count*.

Druhou možností je inicializace počtu shluků pomocí nějaké meta-heuristiky. V rámci této práce jsem se rozhodl použít *Simulované žíhání* a *Genetické algoritmy*. Metoda založená na *Simulovaném žíhání* ponese nazev *SAFCM*, zatímco metoda pracující s *Genetickým algoritmem* budem označena jako *GAFCM*. Obě tyto metody budou podrobněji rozebrány dále.

Metrika vzdálenosti v FCM

První zkoumanou věcí v souvislosti s FCM je použitá metrika vzdálenosti. Vzdálenostní metrika se v FCM používá při výpočtu matice příslušnosti ve vzorci 4.13. Několik základních metrik vzdálenosti již bylo uvedeno v tabulce 4.1. Všechny testy týkající se vhodnosti texturních příznaků budou při výpočtu FCM používat *Normalizovanou Euklidovskou* vzdálenost.

Mezi testované metriky patří *Euklidovská vzdálenost*, *Normalizovaná Euklidovská vzdálenost*, *Bhattacharyyova vzdálenost*, *Hellingerova vzdálenost* a metrika χ^2.

Při použití jiné než *Euklidovské vzdálenosti* v FCM, je nutné upravit použitý *Cluster Validity Index*, aby ve svých výpočtech používal stejnou metriku vzdálenosti jako FCM.

SAFCM

K inicializaci FCM budeme mimojiné využívat i metodu Simulovaného žíhání. Inicializace pomocí ní byla představena v [40] a je popsána v kapitole 4.2. Její modifikovanou verzi, použitou v této práci budeme nazývat *Simulated annealing Fuzzy C-Means* neboli zkráceně *SAFCM*.

Prvním změnou oproti původní metodě je hodnotící metrika. Zde bude možné kromě původní hodnotící metriky *Xie-Beni*, použít i *Fukuyama-Sugeno*, *PCAES* nebo *CWB* index. *CWB* index ale nemusí ve spojení s touto metodou nebo GAFCM pracovat zcela správně. To je zapříčiněno mechanismem výpočtu *CWB*, který používá jako koeficient α hodnotu vypočtenou pro "konečné řešení"s nejvyšším možným počtem shluků. Ta však nemusí být optimální pro "částečná řešení", s kterými pracují tyto dvě metody inicializace.

Další modifikací je, že nebude pracovat s počtem shluků respektive jejich středů od dvou do $\sqrt{N}$ ale od $minClusterCount$ do $maxClusterCount$.

Poslední změnou je, že po vygenerování kandidátního řešení se vypočte matice příslušnosti a poté nová pozice středů shluků. Čímž se v podstatě provede jeden krok FCM, který tak "vylepší"kandidátní řešení a zvedne jeho šance na přijetí.

GAFCM

Navrhnu metodu používající k inicializaci FCM genetický algoritmus, kterou budu nazývat *Genetic algorithm Fuzzy C-Means* neboli zkráceně *GAFCM*. K inicializování FCM pomocí genetického algoritmu jsem byl inspirován v [27].

Algorithm 1 Pseudokód inicializace SAFCM:

```
l = randInt(minClusterCount, maxClusterCount)
ca = chooseKFromX(X,l)
fa = countClusterValidityIndex(ca)
t = tmax
for k = 1 →maxiter do
    if t > tmin then
        smallestCenter = getSmallestCenter(ca)
        biggestCenter = getBiggestCenter(ca)
        nextstep = randInt(1, 3)
        if nextstep = 1 then
            ck = perturbCenter(ca, smallestCenter)
        end if
        if nextstep = 2 then
            ck = splitCluster(ca, biggestCenter)
        end if
        if nextstep = 3 then
            ck = deleteCluster(ca, smallestCenter)
        end if
        ck = oneStepFCM(ck)
        fk = countClusterValidityIndex(ck)
        if fk < fa then
            ca = ck
            fa = fk
            if fn < fa then
                cn = ca
                fn = fa
            end if
        else
            pAccept = exp(−(fk − fa)/t)
            pRand = randReal(0,1)
            if pAccept > pRand then
                ca = ck
                fa = fk
            end if
        end ift = t * theta
    end if
end for
```

ca, ck a cn jsou aktuální řešení, kandidátní řešení a nejlepší řešení
fa, fk a fn jsou jejich ohodnocení
ϑ je koeficient snižování teploty
X je množina všech vstupních bodů

Základním principem genetických algoritmů je vytvoření populace jedinců, kdy každý jedinec reprezentuje přípustné řešení. Všichni jedinci se poté ohodnotí cílovou funkcí. Na základě ohodnocení se vyberou jedinci, kteří budou přeneseni do další generace. Na ně se

poté aplikují genetické operátory křížení a mutace. Křížení dvou jedinců musí být přitom provedeno tak, aby nově vzniklý jedinec reprezentoval přípustné řešení. Z takto vytvořených jedinců poté vznikne nová generace. Tento proces se opakuje, dokud není dosažena ukončující podmínka například ve formě maximálního počtu iterací nebo minimálního požadovaného ohodnocení.

V tomto případě bude jedinec reprezentován sadou středů shluků. Křížení realizuji sloučením vybraných středů shluků dvou jedinců předchozí generace a mutaci posunem středu shluku. Jako stop-kriterium použiji maximální počet iterací.

Nejdříve se vypočte velikost populace. Tu na začátku tvoří stejný počet jedinců pro každý počet shluků z předem daného intervalu $< c_{min}, c_{max} >$. Pokud počet jedinců v počáteční populaci pro každý počet shluku označím c_{ind}, můžu celkový počet jedinců S v populaci P vypočítat:

$$S = (c_{max} - c_{min} + 1) * c_{ind} \tag{5.6}$$

Konkrétního jedince s počtem shluků c inicializuji tak, že střed jeho prvního shluku nastavím na náhodně vybraný vstupní bod z množiny X. Ostatní středy shluků tohoto jedince poté nastavím na další vstupní body z množiny X. Tyto vstupní body však nebudou vybrány s uniformní pravděpodobností, ale s pravděpodobností ϕ_p pro bod i_p.

$$\phi_p = \frac{D(i_p)^2}{\sum_{j=1}^{n} D(i_j)^2} \tag{5.7}$$

$D(i_p)$ značí vzdálenost bodu i_p od nejbližšího již vybraného středu shluku. Tento algoritmus výběru podle nejkratší vzdálenosti vychází z [6] a zde ho budeme nazývat *Shortest Distance Initialization*. Následně provedu pro každého jedince výpočet matice příslušnosti 4.13 a z té přepočítám nové souřadnice středů shluků 4.14. Každého jedince P_i v populaci ohodnotím pomocí *Fukuyama-Sugeno*, *Xie-Beni*, *PCAES* nebo *CWB* indexu a dosaženou hodnotu označím F_i. V případě *PCAES* indexu se F_i ještě násobí -1, aby bylo kompenzováno to, že tento index hodnotí vyšším číslem lepší uspořádání bodů, narozdíl od ostatních indexů, které to dělají naopak. Hodnotu aktuálního řešení F_i porovnám s nejlepším dosaženým F_{best} a pokud je aktuální nižší, tak jím to nejlepší nahradím.

Dále bude provedeno vybrání jedinců, kteří budou přeneseni do další generace. Tento krok realizuji metodou *Linear Ranking Selection* [10]. Nejdříve seřadím jedince v populaci podle hodnotící funkce tak, že index jedna bude mít nejhorší jedinec a index S nejlepší. Index vybraného prvku j pak dostanu ze vztahu:

$$j = \frac{S}{2 * (c - 1)} * \left(c - \sqrt{c^2 - 4 * (c - 1) * \chi}\right) \tag{5.8}$$

χ je náhodně vygenerované reálné číslo od nuly do jedné a parametr c určuje strmost klesání pravděpodobnosti výběru. Ve všech experimentech bude použito $c = 2$.

Po vybrání dostatečného počtu jedinců nutných ke vzniku další generace křížením tyto jedince projdu a sestavíme tabulku udávající počet vybraných řešení s určitým počtem shluků. Vydělením hodnot této tabulky celkovým počtem vybraných řešení dostanu pravděpodobnost výběru řešení s určitým počtem shluků. Tuto tabulku budu nazývat T_p a použiji jí při křížení dvou jedinců k určení počtu shluků, které bude mít nově vzniklý jedinec.

Tím se dostanu do situace, kdy nová generace bude obsahovat méně jedinců s počtem shluků, který byl v předchozí generaci špatně hodnocem, a místo nich budou jedinci s

Algorithm 2 Pseudokód inicializace GAFCM:

```
S = ( c_max - c_min + 1) * c_ind
for i = 1 → S do
    clusterCount = i div c_ind + c_min + 1)
    P[i] = shortestDistanceInit( X, clusterCount)
end for
for iteration = 1 → maxIterationCount do
    for i = 1 → S do
        P[i] = oneStepFCM(P[i])
    end for
    for i = 1 → S do
        F[i] = countClusterValidityIndex(P[i])
        F_best = isBest( F[i], P[i])
    end for
    for i = 1 → S  do
        crossoverFirst[i] = linearRankingSelection(P, F)
        crossoverSecond[i] = linearRankingSelection(P, F)
    end for
    T_p = makeOccuranceTable(crossoverFirst, crossoverSecond)
    for i = 1 → S  do
        if acceptProb < p_cross then
            crossedClusterCount = chooseClusterCount( T_p )
            firstAndSec = crossoverFirst[i].clusterCount + crossoverSecond[i].clusterCount
            if crossedClusterCount > firstAndSec then
                crossedClusterCount = firstAndSec
            end if
            crossed(1) = getBestCenterPCAES(crossoverFirst[i], crossoverSecond[i])
            for j = 2 → crossedClusterCount do
                crossed(j) = getNextCenter(crossoverFirst[i], crossoverSecond[i])
            end for
            P_next[i] = crossed
        else
            P_next[i] = crossoverFirst[i]
        end if
    end for
    for i = 1 → S  do
        for j = 1 → P_next[i].clusterCount do
            if acceptProb < p_mut then
                P_next[i](j) = perturb( P_next [i](j))
            end if
        end for
    end for
    for i = 1 → S  do P[i] = P_next[i]
    end for
end for
```

počtem shluků, který byl v předchozí generaci hodnocen lépe. Při zvolení optimální hodnotící funkce takový postup zaručí, že během několika generací algoritmu vymizí z populace jedinci s nevhodným počtem shluků.

Křížení jedinců proběhne s pravděpodobností p_{cross}. V rámci této práce bude použito:

$$p_{cross} = 0.8 \tag{5.9}$$

Vlastní křížení probíhá tak, že se s pravděpodobností určenou tabulkou $T_p$vygeneruje počet shluků nového jedince. Poté pomocí PCAES indexu [41] ohodnotím shluky obou vybraných jedinců. Tuto hodnotu pro shluk j vypočítám:

$$P_j = \sum_{i=1}^{n} \sum_{j=1}^{C} \frac{{u_{ij}}^2}{u_M} - \exp\left(\frac{-min\{\|c_i - c_k\|^2\}}{\beta_T}\right) \tag{5.10}$$

Ten, který bude ohodnocem nejlépe, použiji jako první shluk nově vzniklého jedince. Další shluky budou vybrány ze dvou původních jedinců na základě pravděpodobnosti vypočítané vztahem 5.7, dokud jich nebude požadovaný počet.

Po vytvoření S jedinců nové generace přichází na řadu mutace. Ta proběhne pro každý shluk jedince samostatně s pravděpodobností p_{mut} a bude v rámci této práce nastavena:

$$p_{mut} = 0.01 \tag{5.11}$$

Samotná mutace proběhne posunem polohy středu mutovaného shluku.

$$v_m[d] = v_p[d] + \alpha * rand\ ,\ rand \in < 0, 1 > \tag{5.12}$$

Kde $v_m[d]$ je d-tá dimenze mutovaného středu a $v_p[d]$ je d-tá dimenze původního středu. α je parametr udávající velikost posunutí:

$$\alpha \in < -\sigma_d, \sigma_d > \tag{5.13}$$

Přičemž σ_d je d-tá dimenze směrodatné odchylky nad množinou X. Tímto postupem byla vytvořena kompletní nová generace se stejnou velikostí jako měla předchozí. Celý postup bude opakován, dokud se dosáhne požadovaného počtu iterací.

Nakonec použiji nejlepší dosažené řešení uložené v P_{best} k inicializaci pozice středů pro FCM.

Subsamplování příznakových vektorů

Subsamplováním příznakových vektorů je založené na myšlence, že k určení správného počtu shluků není potřeba pracovat s celou sadou vstupních vektorů, ale jen s jejich jistou aproximací. Pokud bude tato sada aproximací menší než původní sada, sníží se výrazně čas nutný k nalezení optimálního počtu shluků. Aproximací je v tomto případě myšlen průměr příznakových vektorů odpovídajících bodům, které ve vstupním obraze tvoří souvislou čtvercovou oblast. Délka hrany čtvercové oblasti bude nazvána *faktor subsamplování* a označena s. Například pro $s = 2$ se sníží počet příznakových vektorů na jednu čtvrtinu. *Subsamplování příznakových vektorů* bude použito pouze k určení správného počtu shluků. Vlastní shlukování pak už proběhne s původní sadou příznakových vektorů. *Subsamplování* lze použít pro všechny tři navržené metody odhadu správného počtu shluků, a proto není jejich alternativou, ale možným zrychlením.

Stop-kriterium FCM

FCM v každé iteraci aktualizuje matici příslušnosti a středy shluků dokud se jejich hodnoty mění. V praxi se jako *stop-kriterium* používá porovnání rozdílu mezi dvěma po sobě následujícími kroky a požadované přesnosti.

$$\|U^{k+1} - U^k\| < \epsilon \tag{5.14}$$

Tento rozdíl bude v počítán jako hodnota maximálního prvku v rozdílové matici mezi dvěma kroky:

$$\epsilon = max\left(abs\left({U_{ij}}^{k+1} - {U_{ij}}^{k}\right)\right) \quad \forall U_{ij}\ ,\ U_{ij} \in U^k \tag{5.15}$$

Hodnota požadované přesnosti ϵ byla experimentálně nastavena na:

$$\epsilon = 0.001 \tag{5.16}$$

Vyhodnocení výstupu FCM

Výstupem algoritmu FCM je matice příslušnosti U a matice středů shluků C. Pro účely segmentace obrazu je ale nutné určit, do jakého shluku jednotlivé vstupní body patří.

První a nejjednodušší způsob je pro každý vstupní bod projít odpovídající řádek matice příslušnosti a o shluku, do kterého bude hodnota příslušnosti nejvyšší prohlásit, že do něj tento bod patří.

$$x_i \in c_j | u_{ij} = max(u_{ik}\ ,\ k \in < 1, C >) \tag{5.17}$$

Toto je základní metoda defuzzifikace výstupu FCM, kterou lze použít pokud nemáme o vstupních bodech žádné další informace. V textu ji budu označovat *Highest one*. Protože nevyužívá žádné informace o okolních bodech, může nastat situace, ve které bude aktuální bod zařazen do jednoho shluku, zatímco všechny okolní body budou zařazeny ve shluku jiném.

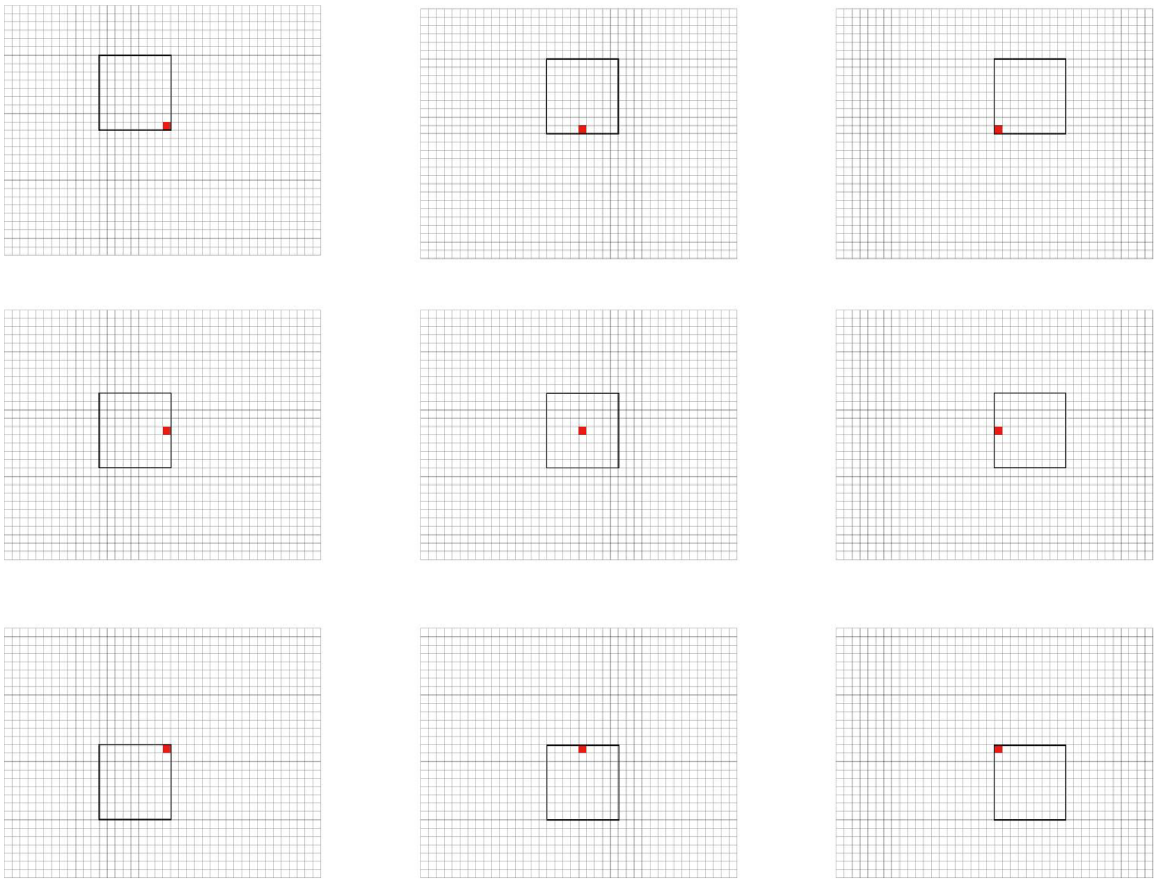
Díky tomu, že používám FCM ke shlukování příznakových vektorů odpovídajících bodům ve vstupním obraze, je navíc při defuzifikaci jejího výstupu k dispozici informace o jejich geometrickém uspořádání. Toho využívají následující dvě metody.

Metoda nesoucí název *Rotation mask* využívá rotační masku tak, že bod, pro který je počítán histogram četnosti výskytu LBP, již nemusí ležet ve středu okna, ale v jednom z jeho rohů případně hran. Celkem existuje devět různých poloh aktuálního bodu vůči extrakčnímu oknu. Všechny jsou znázorněny na obrázku 5.7.

Příznakový vektor aktuálního bodu s takto posunutým extrakčním oknem je vlastně totožný s příznakovým vektorem bodu posunutého vzhledem k aktuálnímu o polovinu délky okna v jednom respektive obou směrech.

Toho využiji při vyhodnocování výstupu FCM tak, že pro každý vstupní bod projdu kromě jemu odpovídajícího řádku v matici příslušnosti, také osm dalších řádků. Tyto řádky budou odpovídat příznakovým vektorům bodů vstupního obrazu posunutých o polovinu délky extrakčního okno oproti bodu aktuálnímu. Pokud polovinu délky extrakčního okna zaokrouhlenou na celé číslo označím Δ a souřadnice aktuálního bodu A ve vstupním obraze (x, y), můžeme souřadnice bodů, jejichž příznakové vektory použijeme při výpočtu shluku, do kterého patří aktuální bod, zapsat jako $A(1), \cdots, A(9)$:

$$A(k) = A_{i,j}\ ,\ i \in x - \Delta, x, x + \Delta\ ,\ j \in y - \Delta, y, y + \Delta \tag{5.18}$$



Obrázek 5.7: Ukázka všech devíti možností posunutí extrakční okna vůči aktuálnímu bodu.

Projitím těchto devíti řádků zjistíme maximální hodnotu příslušnosti do jednoho ze shluků a do něj aktuální bod přiřadíme.

$$A \in c_j | u_{kj} = max(u_{A(k)j}\ ,\ k \in < 1, 9 >\ ,\ j \in < 1, C >) \tag{5.19}$$

Tento postup přináší zlepšení hlavně na rozhraní dvou a více regionů v obraze, kdy původní příznakový vektor, kterým je v našem případě histogram četnosti výskytu LBP, může nést část informací z jednoho regionu a další část z jiného, čímž se stane výsledný příznakový vektor obtížně zařaditelným. Oproti tomu příznakový vektor, vzniklý posunutím extrakčního okna, může do některého z regionů zapadat mnohem lépe.

Třetí metoda, kterou budu testovat v rámci vyhodnocení výstupu FCM, se nazývá *Distribution Maps*. Jejím hlavním principem je, že hodnoty v matici příslušnosti, by neměli mít vliv pouze na příslušný bod výstupního obrazu, ale v menší míře i na okolní body. Přitom se vzrůstající vzdáleností od aktuálního bodu tento vliv klesá.

Nejprve vytvořím C distribučních map o velikosti vstupního obrazu. Tyto mapy označím D_1 až D_C. Poté vytvořím masku M s hranou o délce v, kde v je kladné liché číslo a $\frac{v-1}{2}$ označím v_h. Tato maska bude obsahovat koeficienty 2D Gaussovy funkce.

$$M(x, y) = \exp - \left(\frac{x - v_h}{(2 * \sigma)^2} + \frac{y - v_h}{(2 * \sigma)^2} \right) \tag{5.20}$$

$$\sigma = 0.3 * v_h + 0.8 \tag{5.21}$$

Poté vynásobením této masky s u_{ij} vytvořím masku M_{ij}.

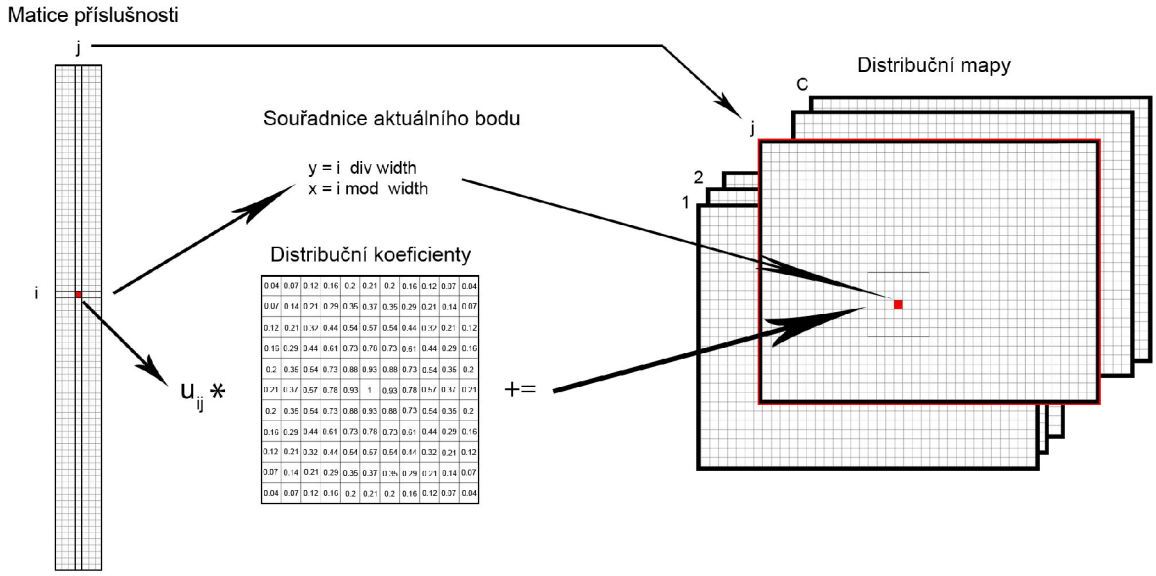
$$M_{i,j}(x,y) = M(x,y) * u_{ij} \quad x,y \in <1,v> \quad \forall u_{ij} \in U \tag{5.22}$$

Střed této masky $M_{i,j}(v_h, v_h)$ potom přiložím na pozici odpovídající aktuálnímu bodu i v distribuční mapě D_j a všechny její hodnoty do této mapy přičtu. Část masky, která se dostane mimo distribuční mapu, bude zanedbána.

$$D_j(x - v_h + k, y - v_h + l) += M_{i,j}(k,l) \quad k,l \in <1,v> \tag{5.23}$$

Tímto postupem jsem získal pro každý shluk jednu distribuční mapu, která ukazuje, jak je tento shluk v obraze zastoupen. Pro každý bod A vstupního obrazu na souřadnicích (x,y) zjistím odpovídající hodnotu $D_j(x,y)$ v každé z distribučních map. Vstupní bod potom zařadím do shluku, jehož distribuční mapa obsahuje pro tento bod nejvyšší hodnotu.

$$A(x,y) \in c_j | D_j(x,y) = max(D_k(x,y)) \ , \ k \in <1,C> \tag{5.24}$$



Obrázek 5.8: Vyhodnocení výstupu FCM metodou *Distribution Maps*.

Tato základní varianta *Distribution Maps*, která používá všechny hodnoty z matice příslušnosti, bude nazývána *Distribution Maps - all values*. Kromě ní budou testovány ještě další varianty *Distribution Maps - best value* a *Distribution Maps - best two values*.

Tyto dvě varianty se líší od původní pouze v tom, že k výrobě distribučních map nepoužívají všechny hodnoty matice příslušnosti, ale pouze jejich část. Konkrétně je to nejvyšší hodnota příslušnosti v každém řádku u *Distribution Maps - best value* respektive dvě nejvyšší hodnoty pro *Distribution Maps - best two values*.

To znamená, že vztah a 5.22 se u *Distribution Maps - best value* změní na:

$$M_{i,j}(x,y) = M(x,y) * u_{ij} \quad u_{ij} \in U | u_{ij} = max(u_{ik}) \ , \ k \in <i,C> \tag{5.25}$$

A u *Distribution Maps - best two values* na:

$$M_{i,j}(x,y) = M(x,y) * u_{ij} \quad u_{ij} \in U | u_{ij} = maxTwo(u_{ik}) \ , \ k \in <i,C> \tag{5.26}$$

Toto vyhodnocení výstupu FCM by mělo ve výsledné segmentaci obrazu zaručit lepší souvislost hranic regionů,a odstranění bodů, které jsou klasifikovány jinak než všechny okolní.

Mean-Shift

Algoritmus Mean-Shift se skládá ze dvou částí. První je Mean-Shift filtrace vstupních bodů, při které je každý bod posouván ve směru gradientu hustoty bodů do lokálního maxima hustoty. V ideálním případě by všechny body z jednoho shluku dokonvergovaly ke stejnému lokálnímu maximu hustoty a počet těchto maxim by se rovnal výslednému počtu shluků. Protože však počet lokálních maxim hustoty, do kterých dokonvergoval nějaký bod, může být ve vícerozměrném prostoru velmi vysoký a i body, které dosáhly stejného maxima, lze klasifikovat do stejného shluku jen s určitou tolerancí, je nutné provést druhou část algoritmu - následné vyhodnocení.

V této práci bude využita pouze první část - filtrace. K následnému vyhodnocení použijeme FCM. Mean-Shift filtrace bude tedy využita jako krok předcházející konečnému shlukování pomocí FCM, čímž se snažím dosáhnout stavu, kdy budou jednotlivé shluky před FCM kompaktnější a zároveň navzájem lépe oddělitelné, což by mělo pozitivně ovlivnit úspěšnost výsledné segmentace obrazu.

Pro výpočet Mean-Shiftu použiju Epanechnikův kernel s profilem [14]:

$$k_e(x) = \begin{cases} 1 - x & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases} \tag{5.27}$$

Filtrování bude provedeno Adaptive Mean-Shiftem [18], který nepoužívá šířku kernelu, ale počet nejbližších okolních bodů k, které do něj budou zahrnuty.

$$m_{h,G}(x) = \frac{\sum_{i=1}^{N} x_i g(||\frac{x-x_i}{h_x}||^2)}{\sum_{i=n}^{N} g(||\frac{x-x_i}{h_x}||^2)} - x \tag{5.28}$$

Kde h_x je vzdálenost k-tého nejbližšího bodu od x. Pokud k vztáhnu k celkovému počtu bodů, bude mít takto upravený Mean-Shift pouze jeden parametr p. Tento parametr určuje podíl celkového počtu bodů, který se při výpočtu vektoru posunu použije.

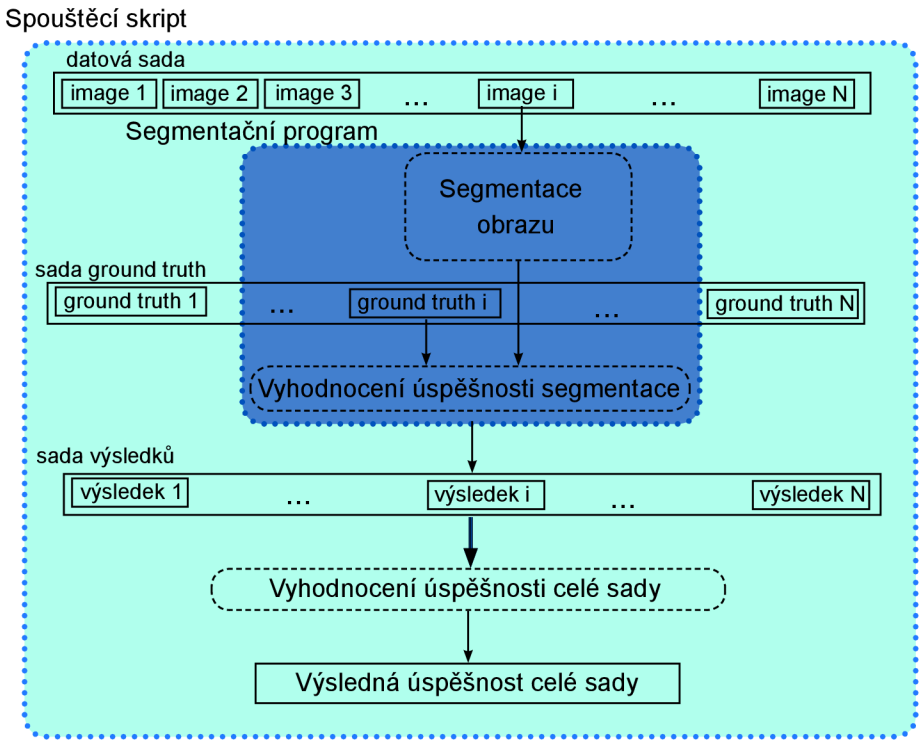
$$k = p * N \tag{5.29}$$

Protože by takovéto filtrování mohlo trvat dlouho, bude možné ho realizovat s *Subsamplováním příznakových vektorů.*

Kapitola 6

Implementace testovacího frameworku

V této kapitole budou popsány prostředky použité k vytvoření testovacího frameworku pro segmentaci obrazu na základě textury. Také se zabývá implementačními problémy jednotlivých algoritmů, případně problémy souvisejícími s využitými nástroji. Dále pak popisuje implementovaná zrychlení několika použitých metod.



Obrázek 6.1: Schéma znázorňující implementační kroky testovacího frameworku pro výpočet úspěšnosti segmentace obrazu pro celou datovou sadu.

6.1 Framework pro vyhodnocení úspěšnosti

Testovací framework je primárně určen k vyhodnocení úspěšnosti segmentace pro celou sadu obrázků, proto se skládá ze tří logických částí. V té první se provádí vlastní segmentace vstupního obrazu. Druhá hodnotí úspěšnost segmentace vzniklé v předcházející části porovnáním s příslušným *ground truth*. Třetí část pak spouští první dvě pro každý obrázek z datové sady a následně vyhodnocuje úspěšnost segmentace pro celou sadu.

V této práci bude první a druhá část testovacího frameworku implementována společně tak, aby tvořila jeden celek. Tento celek nazývám *segmentační program*. Třetí část, která spouští tento *segmentační program* a poté vyhodnocuje jeho výsledky, jsem nazval *spouštecí skript*. Schéma implementace je na obrázku 6.1.

K implementaci *segmentačního programu* jsem zvolil jazyk *C++*. A to z důvodů jeho platformní nezávislosti a také kvůli existenci velkého množství knihoven zabývají se zpracováním obrazových dat, které jsou k dispozici. *Segmentační program* je vytvořen jako konzolová aplikace, aby bylo možné jeho spouštění pomocí skriptů pracujících s celou sadou obrázků.

Hlavní knihovnou použitou pro práci s obrazem je *OpenCV 2.2* [11]. Pomocí jejich struktur a metod budou reprezentována respektive zpracovávána obrazová data.

Protože extrakce texturních příznaků i shlukovací metody mají mnoho parametrů, budou se všechna jejich nastavení načítat po startu *segmentačního programu* z konfiguračního souboru *config.xml*. Každý parametr má navíc defaultní nastavení, které se použije v případě, že ho konfigurační soubor neobsahuje.

Spouštecí skript je realizován pomocí skriptu v jazyce *Python*. Ten vždy spustí *segmentační program* pro konkrétní vstupní obrázek a čeká, než jeho běh skončí, a uloží jím vypočítanou úspěšnost segmentace tohoto obrazu. Skript postupně spouští *segmentační program* pro všechny obrázky v datové sadě. Poté co skončí běh posledního, zpracují se všechny výsledky a vyhodnotí se úspěšnost pro celou datovou sadu. Skript také počítá svou celkovou dobu běhu, aby bylo možné testované metody ohodnotit i z hlediska časové náročnosti.

Framework byl vytvořen a testován pod operačním systémem *MS Windows XP Professional* ve vývojovém prostředí *MS Visual Studio 2010 (32-bit)*. Aby byl testovací framework platformně nezávislý, je jeho součástí i soubor *Makefile* sloužící k vytvoření binárního souboru na Unixových operačních systémech. Pro jeho použití je však nutné mít správně nastavený *pkg-config*.

6.2 Extrakce texturních příznaků

Framework je vytvořen tak, aby bylo možné sestavit výsledný příznakový vektor konkatenací libovolného množství dílčích příznakových vektorů. Přičemž každý dílčí příznakový vektor se extrahuje samostatně, a proto také musí mít svoje vlastní nastavení druhu extrakce(VAR nebo LBP, případně druh LBP), použitého barevného prostoru a LBP parametrů P a R. Počet okolních bodů P může být nastaven na libovolné kladné celé číslo. Aby však okolní body mohly být kruhově symetricky rozmístěny, je nutné, aby jejich počet byl násobkem čtyř. Vzdálenost R musí být kladné reálné číslo.

Sestavení dílčího příznakového vektoru pak probíhá následovně:

- **Vstup** - Matice obsahující vstupní obraz s ozrcadlenými okraji.

- **Transformace barevného prostoru** - Transformace obrazové matice do příslušného barevného prostoru. V případě vícekanálového barevného prostoru následuje i normalizace hodnot do stejného intervalu ve všech kanálech.
- **Sestavení Lookup table** - Používá se pouze pro LBP. Při popisu okolí pomocí P okolních bodů základní variantou LBP lze získat 2^P možných LBP vzorů. Ostatní varianty LBP generují nižší počet vzorů. Například rotačně invariantní a uniformní varianta pouze $P + 2$. Přitom je postup výpočtu obou zmiňovaných variant stejný, až na to, že při té druhé se ještě převádí hodnoty z intervalu P^2 do $P + 2$. Tento převod realizujeme pomocí předpočítané jednorozměrné *Lookup table* o délce P^2
- **Kruhový sampler** - Pro každý bod získá hodnotu bodů kruhového okolí s parametry P a R. V případě, že okolní bod přesně neodpovídá konkrétnímu prvku obrazové matice, vypočítá jeho hodnotu bilineární interpolací.
- **Extrakce LBP/VAR kódu**
  - *LBP* - Hodnoty získané kruhovým samplerem se porovnají s hodnotou středového bodu. Pokud je hodnota vyšší než středová, přičte se příslušná váha do akumulátoru. Po porovnání všech hodnot získaných kruhovým samplerem se obsah akumulátoru použije jako index do *Lookup table*. Hodnota na této pozici se pak uloží do matice LBP kódů *Point values* na pozici odpovídající umístění aktuálního bodu ve vstupním obraze.
  - *VAR* - Nejdřív se vypočte VAR hodnota jako rozptyl hodnot získaných kruhovým samplerem a uloží se do matice *VAR matrix*. Po získání VAR hodnoty pro každý bod v obraze jsou tyto hodnoty vzestupně seřazeny a rozděleny do k itervalů, přičemž do každého z těchto intervalů musí spadat $\frac{N}{k}$ hodnot z matice *VAR matrix*. Číslo intervalu, do kterého spadá hodnota v matici *VAR matrix*, je uloženo na odpovídající pozici v matici VAR kódů *Point values*. Takto se převede kvantováním matice reálných čísel na matici celých čísel.
- **Extrakce příznakového vektoru** - Na začátek řádku matice kódů *Point values* je přiloženo extrakční okno. V něm se vypočte histogram četnosti výskytu jednotlivých kódů. Ten je normalizován počtem bodů v extrakčním okně, čímž vznikne příznakový vektor bodu ležícího ve středu okna, který je uložen na odpovídající místo v matici příznakových vektorů. Extrakční okno se posune po řádku a do nenormalizovaného histogramu jsou přičteny hodnoty bodů, které se posunem do tohoto okna dostaly, respektive odečteny ty, které v něm již nejsou. Vzniklý histogram je poté normalizován a uložen do matice příznakových vektorů. Takto se postupuje, dokud není dosažen konec řádku. Jestliže jsou tímto způsobem zpracovány všechny řádky, vznikne příznakový vektor pro každý bod vstupního obrazu.
- **Výstup** - Matice obsahující dílčí příznakový vektor pro každý bod vstupního obrazu.

Barevné LBP

V této práci jsou použity také dvě varianty LBP pracující s informacemi o barvě. Jsou jimi CLBP a OCLBP. Díky tomu, že je *Kruhový sampler* implemetován tak, aby mohl vzít středový bod z jednoho kanálu a okolní body z jiného, můžeme extrakci CLBP a OCLBP provést pouze zopakováním základního postupu pro příslušné kombinace barevných kanálů a následnou konkatenací.

6.3 Shlukování

Ke shlukování příznakových vektorů bude použita především metoda *Fuzzy C-means* a několik jejich variant. Její výsledky posléze porovnám s výsledky dosaženými pomocí *K-means*, *K-means++* a *Mean-Shift*. Protože *K-means* respektive *K-means++* jsou součástí knihovny *OpenCV*, bude použita tato jejich implementace v testovacím frameworku. K realizaci metody *Mean-shift* použiji součást knihovny *OpenCV* nesoucí název *flann*.

Fuzzy C-means

Hlavním shlukovacím algoritmem, kterým se tato práce zabývá je *Fuzzy C-means*. FCM je iterativní algoritmus, jehož stav bude reprezentován maticí středů shluků a maticí příslušnosti.

- **Vstup** - Matice příznakových vektorů a požadovaný počet shluků.
- **Inicializace středů shluků** - Každý střed shluku obsažený v matici středů shluků je nastaven na náhodně vybraný příznakový vektor.
- **Krok FCM** - Každý krok FCM se skládá z vypočtení matice příslušnosti na základě matice středů shluků a následného přepočtení matice středů shluků z nově vzniklé matice příslušnosti.
  - **Zálohování aktuální matice příslušnosti** - Uložení aktuální matice příslušnosti, aby bylo možné ji později použít pro výpočet rozdílu mezi dvěma kroky FCM.
  - **Výpočet nové matice příslušnosti** - V matici příslušnosti, o velikosti $N * C$, odpovídá řádek číslu příznakového vektoru a sloupec číslu shluku. To znamená, že na řádku jsou postupně uloženy hodnoty příslušnosti jednoho příznakového vektoru do všech shluků. Výpočet všech hodnot realizujeme postupně, průchodem této matice po řádcích. Na každém řádku provedeme C výpočtů příslušnosti. Přitom ve vzorci pro výpočet příslušnosti vektoru do konkrétního shluku 4.13 jsou použity všechny vzdálenosti tohoto vektoru od jednotlivých středů shluků. Při výpočtu matice příslušnosti bychom na každém řádku C-krát počítali všechny vzdálenosti aktuálního vektoru od jednotlivých středů shluků. Výpočet FCM proto urychlíme vytvořením tabulky vzdáleností aktuálního vektoru od středů shluku.

    Celý postup vypadá tak, že na každém řádku matice příslušnosti je naplněna tabulka vzdáleností aktuálního vektoru od jednotlivých středů shluků. Pomocí ní se poté vypočtou všechny hodnoty v tomto řádku matice příslušnosti.
  - **Přepočet matice středů shluků** - V matici středů shluků o velikosti $N * d$ je v každém řádku uložen jeden střed shluku. Každý střed shluku se přitom počítá samostatně. Pro výpočet lze použít vzorec 4.14.
- **Porovnání s předcházejícím krokem** - Do rozdílové matice je uložena matice vzniklá odečtením matice příslušnosti z předcházejícího kroku od té aktuální. Poté se vybere z této matice prvek s nejvyšší absolutní hodnotou a ta je považována za rozdíl mezi dvěma kroky FCM. Pokud je tato hodnota vyšší než požadovaná přesnost, provede se další krok FCM, jinak se přejde na vyhodnocení výstupu.

- **Výstup** - Výstupem je aktuální matice příslušnosti a matice středů shluků. Obě se poté ještě dále zpracovávají.

FCM s proměnlivým počtem shluků

Protože při segmentování obrazu nemusí být dopředu znám počet regionů v obraze, nezná ani FCM počet shluků, s kterým má pracovat. Jak se s tímto problémem vyrovnat bylo rozebráno v kapitole 4.2. Dále jsem navrhl tři možnosti řešení 5.2, které budou v rámci této práci použity. Jde o metodu *All Cluster Count*, procházející celý prostor řešení a tedy v podstatě o řešení "hrubou silou". Další dvě metody inicializace jsou založené na různých meta-heuristikách. Protože však všechny tři mohou běžet velmi dlouho, navrhnul jsem další zrychlení *subsamplováním příznakových vektorů*, které je možné použít u kterékoliv z nich.

Vyhodnocení výstupu FCM

Vyhodnocení výstupu FCM probíhá vždy, ať už byl počet shluků dopředu známý nebo se nějakým způsobem odhadoval. Na základě matice příslušnosti a matice středů shluků vytvoří matici o velikosti vstupního obrazu, jež obsahuje celá kladná čísla určující číslo regionu, do kterého příslušný bod patří.

6.4 Výpočet úspěšnosti segmentace

Celková úspěšnost segmentace pro datovou sadu lze vyjádřit pomocí průměru a směrodatné odchylky úspěšností segmentace jednotlivých obrázků z této sady.

Úspěšnost segmentace konkrétního obrázku bude vypočtena porovnáním s příslušným *ground truth*. Pokud datová sada obsahuje pro tento obrázek více než jedno *ground truth*, vypočte se úspěšnost segmentace porovnáním námi dosaženého výsledku s každým *ground truth* samostatně a nejlepší dosažená úspěšnost pak bude úspěšnost segmentace pro tento obrázek.

Porovnání segmentovaného obrazu s *ground truth* se provede pomocí *f-measure* vypočítaného z dvojic pixelů, tak jak bylo popsáno v kapitole 3.3. Vezmou se všechny dvojice pixelů v obraze a poté je porovnán počet těch, kde jsou jejich body ve stejné relaci (ze stejného regionu/z různých regionů) v segmentovaném obraze a v *ground truth*.

Protože je obrazových bodů typicky velmi mnoho a při výpočtu musí být projity všechny dvojice, kterých je $\binom{n}{2}$, mohlo by vyhodnocení úspěšnosti trvat délé než samotná segmentace obrazu. Z toho důvodu bylo nutné implementoval nějaké zrychlení.

- **Vstup** - Dvě matice velikosti vstupního obraz, přičemž první reprezentuje segmentovaný obraz a druhá příslušné *ground truth*. Obě matice obsahují nezáporná celá čísla, která značí číslo regionu, do kterého příslušný bod obrazu patří.

- **Vytvoření tabulky překrytí** - Zjistí se počet regionů v segmentovaném obrazu C_s a v *ground truth* C_g. Vytvoří se *tabulku překrytí* K o velikostí C_s x C_g. Následně projitím obou matic současně a zjištěním regionu i, do kterého patří aktuální bod v segmentovaném obraze, a region j, do kterého patří v *ground truth*. Inkrementuje se buňka *tabulky překrytí* na pozici (i, j). Tímto postupem vznikne tabulka, jejíž buňky udávají kolik bodů z regionu i v segmentovaném obraze se překrývá s regionem j v *ground truth*.

- **Výpočet kombinačních proměnných** - Počet dvojic vytvořených z množiny o velikosti n lze vyjádřit jako $\binom{n}{2}$. Díky tomu lze počet dvojic bodů, které leží ve stejném regionu jak v segmentovaném obraze tak i v *ground truth*, vypočítat:

$$true\ positive = \sum_{i=1}^{C_s} \sum_{j=1}^{C_g} \binom{K(i,j)}{2} \tag{6.1}$$

Počet dvojic pixelů klasifikovaných do stejného regionu v segmentovaném obraze, což je součet *true positive* a *false positive*, může být vyjádřen:

$$A = \sum_{i=1}^{C_s} \binom{\sum_{j=1}^{C_g} K(i,j)}{2} \tag{6.2}$$

Počet dvojic pixelů klasifikovaných do stejného regionu v *ground truth*, což je součet *true positive* a *false negative*, lze vyjádřit:

$$B = \sum_{j=1}^{C_g} \binom{\sum_{i=1}^{C_s} K(i,j)}{2} \tag{6.3}$$

- **Výpočet statistických proměnných** - Za pomoci kombinačních proměnných je možné vypočítat Precision:

$$P = \frac{true\ positive}{A} \tag{6.4}$$

Recall:

$$R = \frac{true\ positive}{B} \tag{6.5}$$

Posledním krokem je výpočet f-measure podle vzorce 3.14.

- **Výstup** - Reálné číslo od nuly do jedné, přičemž nula značí absolutní neshodu segmentovaného obrazu s *ground truth*, zatímco jedna naprostou shodu.

Načítání ground truth

V této práci budou použity obrazové i textové soubory k uchování *Ground truth*. V obou případech se soubory po načtení převedou na matici o rozměrech obrazu, která obsahuje kladná celá čísla značící číslo regionu, do kterého tento obrazový bod patří.

Obrazové *ground truth* je 8-bitový jednokanálový obrazový soubor, který přímo obsahuje hodnoty odpovídající číslu regionu, do kterého spadá.

Naproti tomu textová *ground truth* mají nejprve hlavičku obsahující metadata. Tyto informace jsou načteny jako první a ověří se jejich správnost. Pokud vše vyhovuje, přejde se na druhou část obsahující vlastní informace o regionech obrazu.

Ta se skládá z textových řádků, přičemž každý řádek reprezentuje segment obrazového řádku. Segmentem obrazového řádku je jeho souvislá část, ve které všechny body patří do jednoho regionu. Segment je reprezentován čtyřmi celými čísly. První označuje číslo regionu do kterého patří, druhé je číslo obrazového řádku, jehož část reprezentuje. Třetí a čtvrté pak počáteční respektive koncovou pozici na příslušném obrazovém řádku.

Program je implementován tak, že se každé *ground truth* snaží nejprve načíst jako obrazové. Pokud to nelze realizovat, vyzkouší textovou variantu.

Kapitola 7

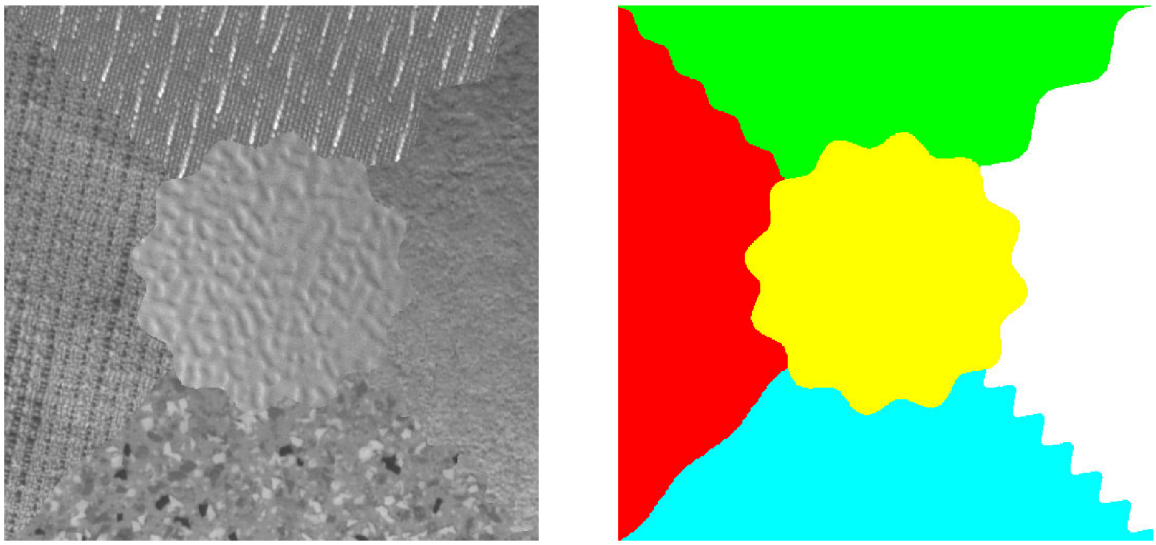
Provedené experimenty

V této kapitole budou nejdříve popsány použité datové sady a návrh všech experimentů. Další část pak prezentuje výsledky dosažené testovacím frameworkem.

7.1 Databáze obrazů pro experimenty

Porovnání dvou a více typů segmentace obrazu lze provést pouze sporovnáním jejich úspěšnosti nad nějakou datovou sadou. V této práci budou k vyhodnocení úspěšnosti použity dvě datové sady.

První z nich je sada syntetických obrázků *Outex US 00000*. Tato sada převzatá z [37] je zaměřena právě na segmentaci obrazu na základě textury. Obsahuje celkem sto šedotónových obrázků s rozměry 512 x 512. Každý je složený z pěti různých textur, přesně podle *ground truth*. To znamená, že obsahuje pět regionů, přičemž každý má svou jednotnou texturu. Textury byli foceny, v kontrolovaných podmínkách s homogenním osvětlením. Každá z nich může být ve výsledném obrázku natočena o 0, 5, 10, 15, 30, 45, 60, 75, nebo 90 stupňů a může mít rozdílné měřítko. Obrázek 7.1 obsahuje vzorek z této sady s jeho *ground truth*.

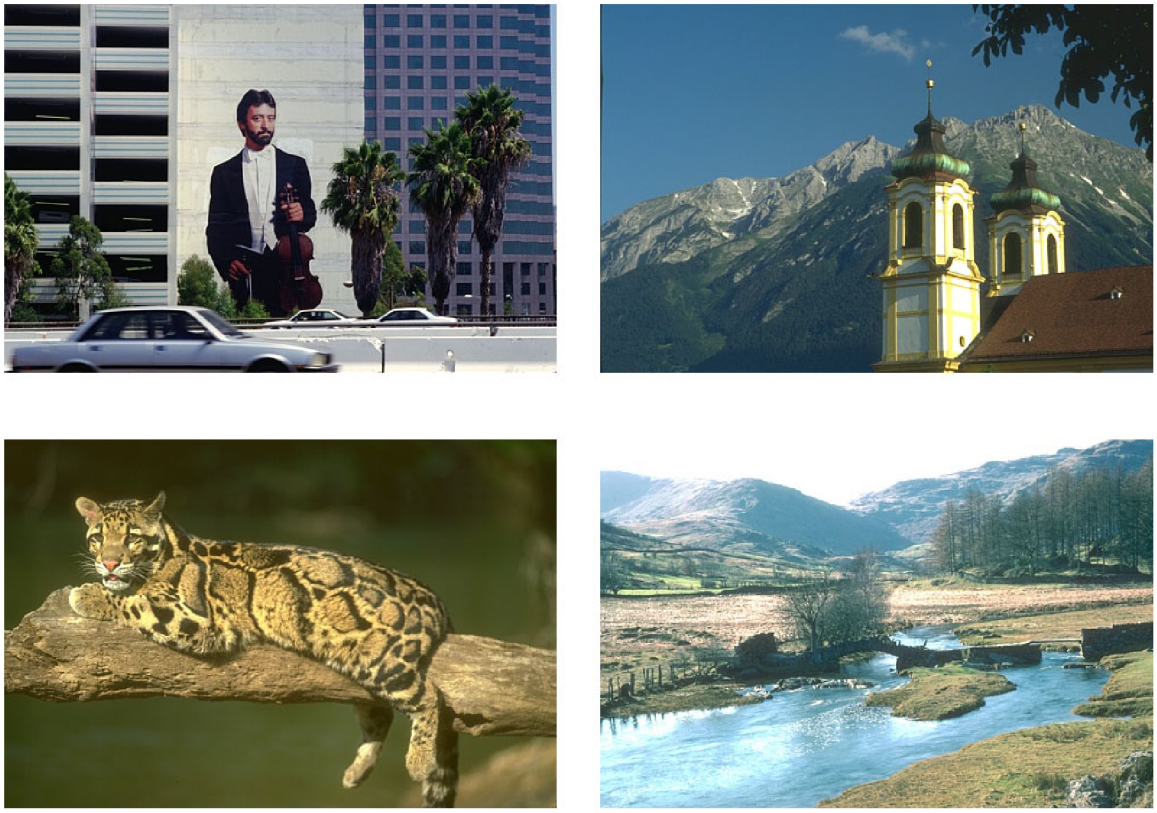


Obrázek 7.1: Obrázek ze sady *Outex US 00000* a jeho *ground truth*.

Díky tomu, že je tato sada synteticky vytvořená, odpovídá její *ground truth* jediné správné segmentaci obrazu. Narozdíl od reálných obrázků, jejichž *ground truth* je závislé na sémantické interpretaci obrazu člověkem, který ho vytváří. Proto bude tato sada použita převážně k porovnání parametrů extrakce texturních příznaků nebo parametrů shlukovacích algoritmů.

Datová sada byla původně tvořena obrázky ve formátu RAS. Aby je použité knihovny bez problémů zpracovaly, byly všechny obrázky převedeny do formátu PNG.

Druhá použitá sada obsahuje pouze reálné obrázky a nazývá se *BSDS300*. Je z *The Berkeley Segmentation Dataset and Benchmark* [25]. Skládá se z tří set obrázků s různým rozlišením. Každý z nich byl několikrát nezávisle na sobě ručně anotován a má tedy několik *ground truth*. Tuto sadu jsem se rozhodl upravit a vzniklou modifikaci nazvanou *BSDS300*



Obrázek 7.2: Ukázka obrázku ze sady *BSDS300*.

modified použiji ve všech testech. Původní *BSDS300* je rozdělena na trénovací a testovací část. Protože však tato práce nepracuje s žádným algoritmem používajícím učení, nemá rozdělení pro navržené experimenty smysl. Prvním krokem k vytvoření *BSDS300 modified* z *BSDS300* je tedy sloučení trénovací a testovací části.

BSDS300 obsahuje dvě sady *ground truth*. První vytvořenou ručním anotováním obrázků ve stupních šedi a druhou vytvořenou anotováním barevných obrázků. Tím však vzniká problém ve chvíli, kdy chci porovnat výsledky LBP pracujících s šedotónovým obrazem oproti variantám LBP, které využívají barvu, protože každá používá rozdílnou sadu *ground truth*. Z tohoto důvodu budu pro všechny varianty LBP používat pouze jednu sadu. Tou budou *ground truth* vzniklé anotováním barevných obrázků, protože by měly být přesnější, díky tomu, že při jejich vytváření byly k dispozici i informace o barvě.

Poslední významnou změnou *BSDS300 modified* oproti *BSDS300* je odstranění obrázků, které nemají *ground truth* odpovídající segmentaci na základě textury. Tento krok je velmi důležitý, protože *BSDS300* je určena k segmentaci obrazu, respektive detekci hranic regionů. Proto část jejich *ground truth* rozděluje obraz na regiony na základě hran i v případě, kdy mají tyto regiony stejnou, nebo velice podobnou texturu. Takováto *ground truth* jsou pro naše použití nevhodná, protože neodpovídají požadovanému chování nástroje na segmentaci obrazu. Vybrání obrázků, ke kterým existuje *ground truth* odpovídající požadovanému chování námi testovaného nástroje, je finálním krokem při tvorbě *BSDS300 modified*.

Tato sada bude použita především k vyhodnocení variant LBP zaměřených na barvu (CLBP a OCLBP). Dále pak také ke zjištění zda jsou vybrané metody extrakce a shlukování nad syntetickou sadou vhodné i pro reálné obrázky.

7.2 Návrh experimentů

Tato sekce se zabývá konkrétním nastavením testovaných metod respektive jejich parametrů.

Experimenty budou prováděny postupně, to znamená, že při některých se použije i část výsledků z předcházejících experimentů. Pokud je na každý test pohlíženo jako na hledání optimálního nastavení určitého parametru, lze také napsat, že při vyhodnocení nastavení parametru A se zafixují všechny ostatní parametry B, C, D, ... aby se co nejvíce omezil jejich vliv na daný experiment.

Všechny prováděné experimenty lze rozdělit do dvou skupin - extrakce optimálního příznakového vektoru a nalezení nejvhodnější metody jejich shlukování pro účely segmentace obrazu.

Experimenty týkající se extrakce optimálního příznakového vektoru budou prováděny pouze se sadou *Outex US 00000*, aby bylo možné určit, jaké parametry extrakce jsou pro popis textury nejvhodnější. Ke shlukování je v tomto případě použito FCM s pevně nastaveným počet shluků, aby nebyly výsledky testů ovlivněny metodami určujícími tento počet. Pouze experimenty týkající se variant LBP pracujících s informacemi o barvě budou naopak testovány na sadě *BSDS300 modified*. Naopak experimenty zabývající se shlukováním budou porovnávat jednotlivé metody k určení správného počtu shluků.

Všechny experimenty jsou prováděny s rotačně invariantní a uniformní variantou LBP. Při shlukování příznakových vektorů pomocí FCM je použita ve všech experimentech *Normalizovaná Euklidovská vzdálenost*. Pro sadu *Outex US 00000* bude ve všech experimentech použita velikost extrakčního okna 48 a pro sadu *BSDS300 modified* 24. Pokud není u testu explicitně uvedeno něco jiného, použije se vždy FCM s vyhodnocením výstupu metodou *Highest one*.

Nastavení parametrů P a R

Prvním z testů bude nalezení optimálních hodnot LBP parametrů **P** a **R**, kde **P** určuje počet budů popisujících lokální okolí bodu, zatímco **R** je jejich vzdálenost od tohoto bodu. Lze tedy říct, že jde pouze o nalezení optimálního počtu bodů pro popis okolí v určité vzdálenosti. Protože okolí bodu ve vzdálenosti jedna se popisuje vždy všemi osmi hodnotami okolních bodů, nebudem pro tuto vzdálenost žádný jiný počet testovat. Celý experiment bude hodnotit vhodný počet bodů okolí pro vzdálenosti jedna až čtyři.

Všechna testovaná nastavení jsou zapsána v tabulce 7.1, přičemž LBP pracující s parametry $P = x$ a $R = y$ bude označeno $PxRy$.

Vzdálenost	Testované varianty
1	$P8R1$
2	$P8R2$, $P12R2$, $P16R2$
3	$P8R3$, $P12R3$, $P16R3$, $P24R3$
4	$P8R4$, $P12R4$, $P16R4$, $P24R4$

Tabulka 7.1: Všechna testovaná nastavení parametrů P a R.

Cílem toho experimentu je zjistit optimální P pro jednotlivé R, čímž vzniknou čtyři varianty nastavení parametrů, které budou použity v dalších experimentech.

Ke shlukování se použije FCM s pevně nastaveným počtem shluků na 5.

Barevné LBP

V tomto testu bude hodnocena úspěšnost jednotlivých barevných prostorů pro Color LBP a Opponent color LBP. Otestuji přitom barevné prostory: RGB, HSV, HLS, CIE L*a*b, CIE L*u*v, CIE XYZ a CIE YCrCb . Testy proběhnou s nastavením $P8R1$ a vyhodnotí se i rozdíl výsledků dosažený těmito dvěma variantami od výsledku základních LBP.

Testy budou prováděny s datovou sadou *BSDS300 modified*. Shlukování pak pomocí FCM inicializovaným metodou *All Cluster Count* pracující s CWB *Cluster validity indexem* a zrychlením pomocí *Subsamplování*. Při shlukování pomocí FCM použiji namísto *Normalizované Euklidovské vzdálenosti* metriku chi^2. A to protože při extrakci texturních příznaku pomocí OCLBP vzniknou v histogramu četnosti i statisticky nevýznamné sloupce, kterým by při použití *Normalizované Euklidovské vzdálenosti* byla dána stejná relativní váha jako všem ostatním, a tak by zanášely do shlukování chybu.

Multiresolution LBP

Tento experiment navazuje na ten první, protože Multiresolution LBP je vlastně kombinací několika LBP s různými parametry P a R. Jelikož nemá význam kombinovat LBP se stejnou hodnotou R, bude pro každé testované R použito pouze jedno P. Tím bude optimální hodnota nalezená v předcházejícím testu. Pokud pro konkrétní $R = x$ nazveme optimální hodnotu parametru P $\sigma(x)$ lze celé nastavení označit $P\sigma(x)Rx$. Multiresolution LBP kombinující $P\sigma(x)Rx$ a $P\sigma(y)Ry$ označím $P\sigma(x)Rx|P\sigma(y)Ry$.

Všechny testované kombinace Multiresolution LBP jsou zapsané v tabulece 7.2.

Počet variant	Testované kombinace
2	$P8R1\|P\sigma(2)R2$, $P8R1\|P\sigma(3)R3$, $P8R1\|P\sigma(4)R4$, $P\sigma(2)R2\|P\sigma(3)R3$, $P\sigma(2)R2\|P\sigma(4)R4$, $P\sigma(3)R3\|P\sigma(4)R4$
3	$P8R1\|P\sigma(2)R2\|P\sigma(3)R3$, $P8R1\|P\sigma(2)R2\|P\sigma(4)R4$ $P8R1\|P\sigma(3)R3\|P\sigma(4)R4$, $P\sigma(2)R2\|P\sigma(3)R3\|P\sigma(4)R4$
4	$P8R1\|P\sigma(2)R2\|P\sigma(3)R3\|P\sigma(4)R4$

Tabulka 7.2: Všechna testovaná nastavení Multiresolution LBP.

Cílem toho experimentu je vytvoření Multiresolution LBP za účelem otestováním všech kombinací variant nastavení vzešlých z prvního experimentu.

Ke shlukování se použíje FCM s pevně nastaveným počtem shluků na 5.

Experimenty s extrakčním oknem a kontrastem v LBP

Tento experiment má dvě části. Nejprve se otestuje vhodnost zkombinování LBP a kontrastu. Otestováno bude, jak ovlivňuje počet hladin, na který je VAR kvantovát, úspěšnost segmentace. Optimální počet hladin bude testován pro Euklidovskou vzdálenost a pro normalizovanou Euklidovskou vzdálenost. Poté se otestuje i úspěšnost metody LBPV. Kombinaci LBP a VAR s VAR kvantovaným na z hladin bude značena $LBP + VAR(x)$. Celkový souhrn testů v tomto experimentu obsahuje tabulka 7.3.

Druhá část bude testovat vhodnost kruhového extrakčního okna navrženého v kapitole 5.1 a váhování extrakčního okna 2D Gaussovou funkcí z kapitoly 5.1 oproti klasickému postupu extrakce LBP texturních příznaků.

	Varianta
1	$LBP + VAR(8)$ Euclidean
2	$LBP + VAR(16)$ Euclidean
3	$LBP + VAR(32)$ Euclidean
4	$LBP + VAR(64)$ Euclidean
5	$LBP + VAR(8)$ Normalized Euclidean
6	$LBP + VAR(16)$ Normalized Euclidean
7	$LBP + VAR(32)$ Normalized Euclidean
8	$LBP + VAR(64)$ Normalized Euclidean
9	Kruhové extrakční okno
10	Váhování extrakčního okna 2D Gaussovou funkcí
11	LBPV

Tabulka 7.3: Všechna testovaná nastavení třetího experimentu.

Všechny testy tohoto experimentu proběhnou s nastavením parametrů $P8R1$. Ke shlukování se použije FCM s pevně nastaveným počtem shluků na 5.

Multiresolution LBP + VAR

Protože zkombinování LBP a VAR by mělo přinést zlepšení výsledků, otestuji, jak bude fungovat spojení Muliresolution LBP a VAR. Přitom vždy použijeme LBP a VAR nastavené na stejné hodnoty P a R. Testy budou prováděny pouze pro dva nejlepší počty hladin pro kvantování VAR, které jsem zjistil v předchozím testu. V testu bude použito jen několik nejlepších kombinací nastavení pro Multiresolution LBP zjištěných v předchozích testech. Zkombinování $P8R1\#LBP+VAR(16)$ a $P12R2\#LBP+VAR(16)$ budeme značit $P8R1|P12R2\#LBP + VAR(16)$.

Ke shlukování se použijeme FCM s Normalizovanou Euklidovskou vzdáleností, pevně nastaveným počtem shluků na 5.

Metriky vzdálenosti v FCM

Cílem tohoto experimentu je zjistit, jaký vliv na celkový výsledek má metrika výpočtu vzádlenosti v FCM. Testována bude Euklidovská vzdálenost, Normalizovaná Euklidovská vzdálenost, Hellingerova vzdálenost, Bhattacharyyova vzdálenost a metrika χ^2. Normalizovaní Euklidovské vzdálenosti se provede vydělením směrodatnou odchylkou nad vstupními daty. Vzorce pro výpočet vzdálenosti ve všech zmíněných metrikách obsahuje tabulka 4.1.

Testy použijí nastavení parametrů $P8R1$, namísto nejlepší dosažené varianty Multiresolution LBP + VAR, aby se zabránilo rozdílnému vlivu počtu kvantizačních hladin VAR na jednotlivé metriky. FCM použije pevně nastavený počet shluků na 5.

Metody pro neznámý počet shluků

Tento experiment se zabývá metodami určení optimálního počtu shluků v FCM. Základní metodu, která vypočítá FCM pro všechny přípustné počty shluků, z kterých potom vybere nejlepší, je nazvána *All Cluster Count - XXX*, kde XXX značí použitý *Cluster validity index*. Prvním testem tohoto experimentu bude nalezení nejvhodnějšího *Cluster validity indexu*. Vzorce několika vybraných lze najít v tabulce 4.2. Testy proběhnou pro *Fukuyama-Sugeno*, *Xie-Beni*, *PCAES* a *CWB* index.

Druhá část otestuje pro nejlepší nalezený *Cluster validity index* metody inicializace pomocí *SAFCM* a *GAFCM*. Nejlepší nalezený *Cluster validity indexem* se použije, aby byl z tohoto testu zřejmý vliv použitého algoritmu nikoliv hodnotící funkce.

U všech tří metod bude kromě úspěšnosti sledována také délku běhu. Ta bude pouze relativní, vztažená k délce běhu metody *All Cluster Count* s nejlepším *Cluster validity indexem*.

Úkolem třetí částí experimentu je zjistit, jak ovlivňuje rychlost a kvalitu segmentace *Subsamplování příznakových vektorů* popsané v kapitole 5.2. Testovaným parametrem je tomto případě velikost *faktoru subsamplování*.

Vyhodnocení výstupu FCM

Všechny předchozí experimenty používaly vyhodnocení výstupu FCM metodou *Highest one* 5.2. Tato metoda je nejjednodušším univerzálním způsobem defuzzifikace výsledků FCM, která však nepoužívá žádné informace o prostorovém rozložení bodů v obraze. Proto budou otestovány další dvě metody nazvané *Rotation mask* a *Distribution maps*, které s informacemi o uspořádání bodů ve vstupním obraze pracují, a tudíž by měly úspěšnost výsledné segmentace obrazu zvýšit.

Tento experiment také otestuje vliv zařazení mediánového filtru za defuzzifikaci výstupu FCM na celkovou úspěšnost segmentace obrazu.

Porovnání shlukovacích algoritmů

Přestože jedním z hlavních cílů této práce je zkoumání různých variant, vylepšení a modifikací shlukovacího algoritmu FCM, budou kromě něj ke shlukování příznakových vektorů použity i další metody. Konkrétně půjde o K-Means, K-Means++ a Mean-Shift filtering. V případě metody *Mean-Shift filtring* bude následné shlukování provedeno pomocí FCM s

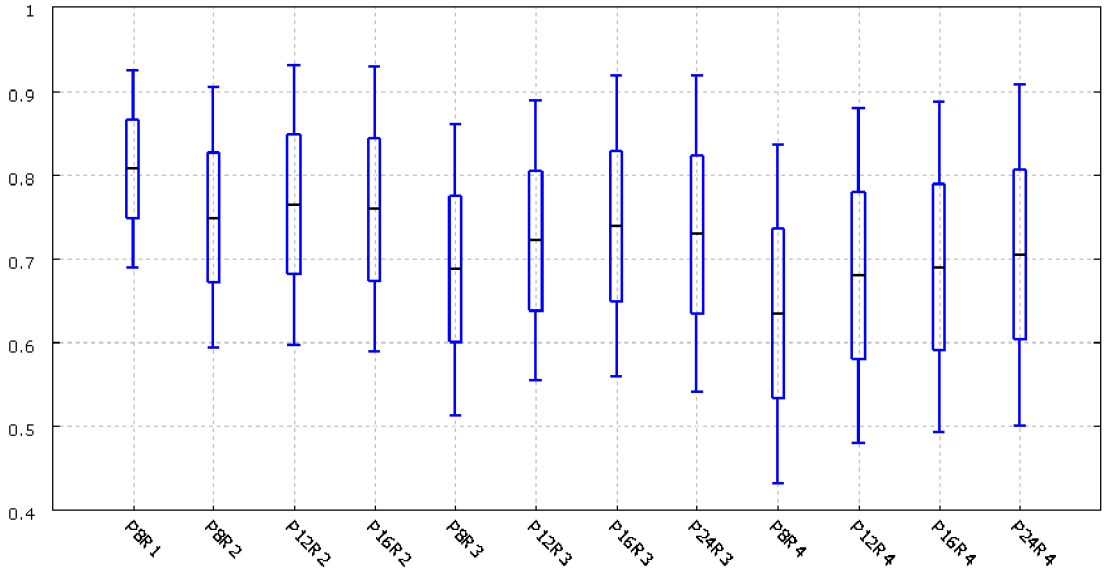
normalizovanou Euklidovskou vzdáleností. V případě FCM i Mean-Shift filteringu použiji defuzzifikaci *Distribution maps 2 - 36*. Algoritmy *K-Means* a *K-Means++* znají dopředu, narozdíl od ostatních, správný počet shluků.

7.3 Vyhodnocení extrakce texturních příznaků

Následující dvě kapitoly obsahují výsledky testů zabývajících se optimální sadou texturních příznaků respektive jejich následným shlukování. Vhodnost jednotlivých metod nebo nastavení je hodnocena průměrem a směrodatnou odchylkou úspěšnosti segmentace celé datové sady. Do grafů je pak vynášen průměr plusmínus jedna respektivě dvě směrodatné odchylky.

Nastavení parametrů P a R

Experiment hledající optimální počet bodů k popisu okolí v dané vzdálenosti metodou LBP. Všechny výsledky jsou zobrazeny v grafu 7.3 a tabulce 7.4.



Obrázek 7.3: Graf závislosti úspěšnosti segmentace na parametrech P a R.

Z grafu je jasně vidět, že optimální počet bodů k popisu okolí je závislý na jejich vzdálenosti od středového bodu, protože pro každý testovaný poloměr R nám vyšel jako nejvhodnější jiný počet P.

Také se potvrdilo, že se vzrůstajícím poloměrem R roste i optimální počet okolních bodů P. Je to dáno principem LBP, které popisuje kruhové okolí bodu rozmístěním P bodů na kružnici s poloměrem R a středem v popisovaném bodě. Protože rozmístěním bodů na tuto kružnici v podstatě aproximujeme průběh intenzity obrazu na celém jejím obvodu, je logické, že se vzrůstajícím obvodem je potřeba i více bodů k aproximaci, a proto se vzrůstajícím poloměrem R narůstá i optimální počet bodů P.

V dalších testech budu tedy používat zjištěné optimální kombinace parametrů P a R, označené : $P8R1$, $P12R2$, $P16R3$, $P24R4$.

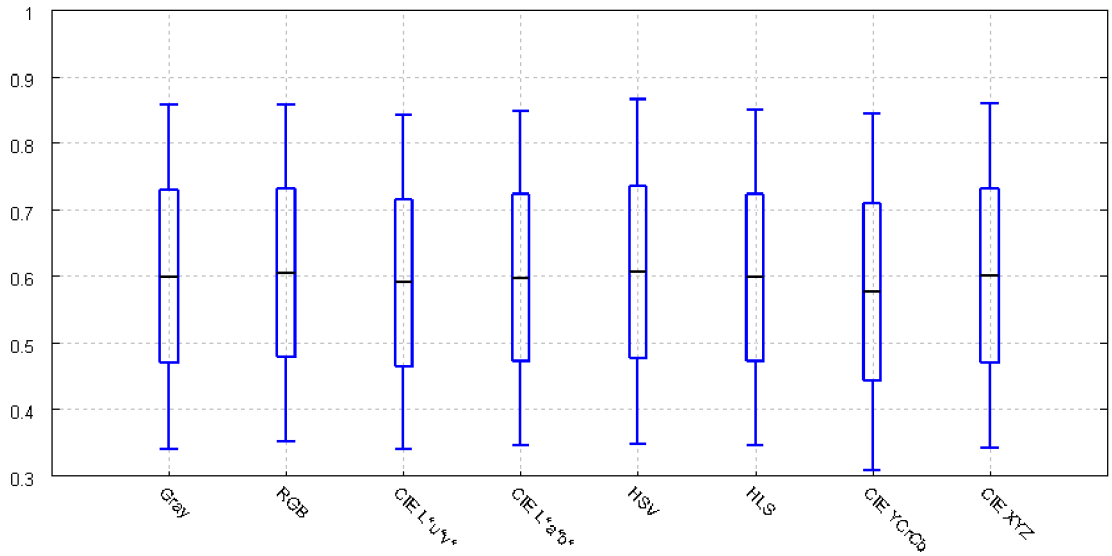
P	R	Průměr ± směrodatná odchylka
8	1	0.8065 ± 0.0589
8	2	0.7486 ± 0.0776
12	2	0.7640 ± 0.0833
16	2	0.7587 ± 0.0851
8	3	0.6869 ± 0.0870
12	3	0.7211 ± 0.0833
16	3	0.7383 ± 0.0887
24	3	0.7288 ± 0.0851
8	4	0.6339 ± 0.1010
12	4	0.6789 ± 0.0998
16	4	0.6894 ± 0.0985
24	4	0.7038 ± 0.1016

Tabulka 7.4: Tabulka výsledků úspěšnosti segmentace pro různé LBP parametry P a R. Pro každý poloměr je barevně zvýrazněn nejvhodnější počet bodů.

Úplně nejlépe dopadla základní varianta $P8R1$, a to nejen díky, tomu že má nejvyšší průměrnou úspěšnost, ale zároveň má i nejmenší směrodatnou odchylku, což znamená, že je ze všech testovaných variant nejstabilnější.

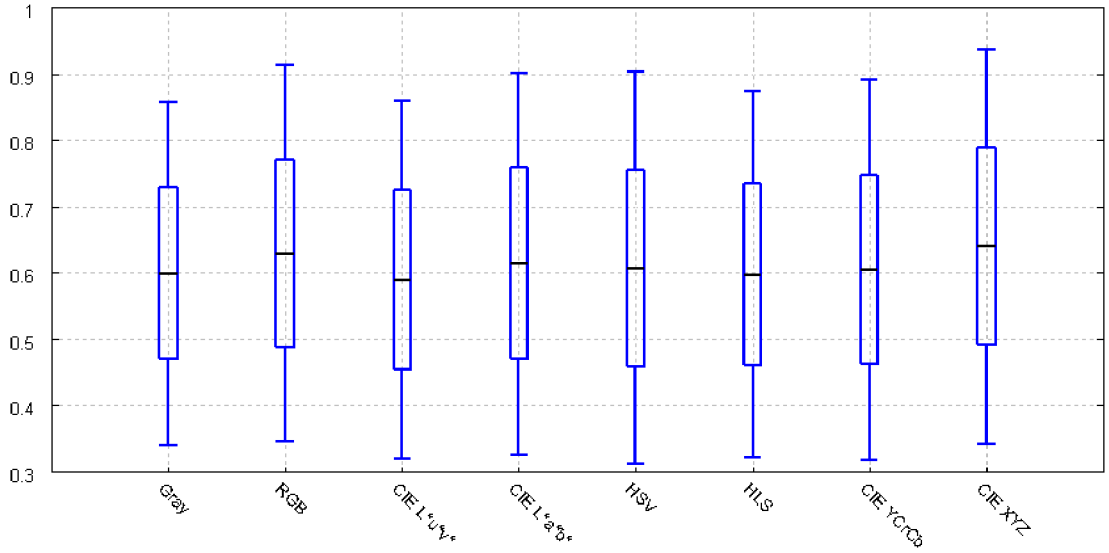
Barevné LBP

Tento experiment bude testovat na sadě *BSDS300 modified* vliv použitého barevného prostoru u metod Color LBP 7.4 a Opponent Color LBP 7.5.



Obrázek 7.4: Graf znázorňující úspěšnost segmetnace obrazu za použití color LBP pro jednotlivé barevné prostory.

Z 7.5 je vidět, že oponent color LBP dosáhly na testované sadě průměrně vyšší úspěšnosti, než color LBP a až na dva barevné prostory byly jejich výsledky lepší, než základní



Obrázek 7.5: Graf znázorňující úspěšnost segmetnace obrazu za použití oponent color LBP pro jednotlivé barevné prostory.

Barevný prostor a varianta	Průměr ± směrodatná odchylka
Gray LBP	0.5995 ± 0.1298
color RGB	0.6048 ± 0.1299
color CIE L*u*v*	0.5908 ± 0.1258
color CIE L*a*b*	0.5976 ± 0.1257
color HSV	0.6065 ± 0.1297
color HLS	0.5981 ± 0.1263
color CIE YCrCb	0.5765 ± 0.1338
CIE XYZ	0.6007 ± 0.1299
oponent color RGB	0.6293 ± 0.1419
oponent color CIE L*u*v*	0.5896 ± 0.1353
oponent color CIE L*a*b*	0.6145 ± 0.1440
oponent color HSV	0.6071 ± 0.1481
oponent color HLS	0.5978 ± 0.1377
oponent color CIE YCrCb	0.6048 ± 0.1432
oponent color CIE XYZ	0.6398 ± 0.1487

Tabulka 7.5: Tabulka výsledků úspěšnosti segmentace obrazu pro CLBP a OCLBP s různými barevnými prostory.

varianta LBP, nepracující s barvou. Rozdíl mezi nejlepším a nejhorším barevným prostorem v případě oponent color LBP je necelých 10%. Naproti tomu color LBP dosáhly ve čtyřech ze sedmi testovaných barevných prostorů horších výsledků než základní varianta LBP. Rozdíl mezi nejlepším a nejhorším barevným prostorem v tomto případě činí zhruba

5%. Z těchto výsledků vyplývá, že použitý barevný prostor má na výslednou úspěšnost segmentace značný vliv. Nejlepší testovanou variantou bylo Oponent color LBP pro barevný prostor CIE XYZ.

Na všech těchto výsledcích je vidět, že pro datovou sadu *BSDS300 modified* je průměrná úspěšnost nižší než u první testované sady *Outex US 00000*. Navíc její směrodatná odchylka je mnohem vyšší, což značí vyšší míru nestability. Po manuálním projití segmentovaných obrazů bylo zjištěno, že na části obrázků ze sady *BSDS300 modified* segmentační program selhává. Příčinnou byl vždy jeden z těchto důvodů:

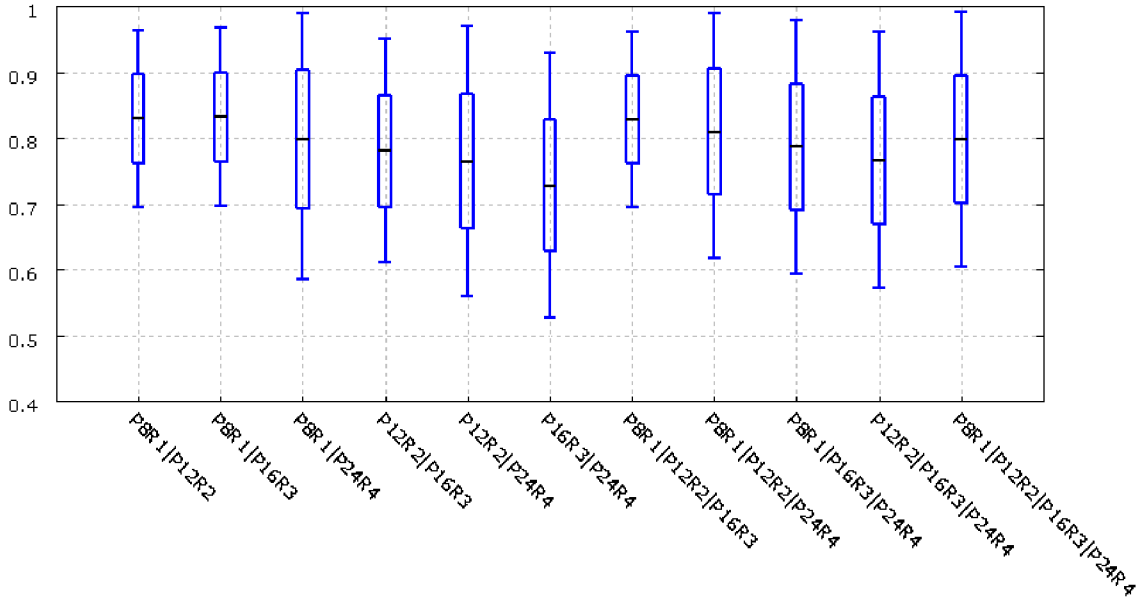
- *Rozostření části obrázku* - Některé obrázky ze sady *BSDS300 modified* mají část obrazu rozostřenou. Díky tomu mohou být příznakové vektory bodů, které jsou v rozostřené části obrazu, ale patří do různých regionů, velmi podobné a příznakové vektory bodů ze stejného regionu obrazu, jehož jedna část je rozostřená a druhá nikoliv, mohou být velmi odlišné. Takováto situace může vést až k rozdělení obrazu na ostrou a rozostřenou část. Příklad tohoto jevu je na obrázku A.4, kde je vidět, že výsledné regiony, do kterých je obraz rozdělen, poměrně přesně odpovídají ostré respektive rozostřené části obrazu. Rozostření části obrazu je pro segmentaci na základě LBP texturních příznaků velký problém.

- *Špatný počet shluků* - U některých obrázků je špatně vyhodnocen optimální počet shluků i v případě, že shlukování pro jiný počet shluků by dopadlo daleko lépe. Příklad tohoto jevu je na obrázku A.5, kde je zobrazen vstupní obraz, jeho segmentace pro manuálně nastavený počet regionů a segmentace, při které se zvolí optimální počet regionů pomocí *Cluster validity indexu*. Tento jev nastává převážně u obrázků, ve kterých se vyskytuje velká jednobarevná plocha bez textury. Příznakový prostor pak vypadá tak, že je v něm část vektorů reprezentujících body jednobarevné oblasti velmi blízko u sebe a zároveň je tato skupina vzdálená od všech ostatních. To zapříčiní, že rozdělení příznakových vektorů na ty, které reprezentují onu jednobarevnou oblast a "zbytek"je hodnoceno *Cluster validity indexem* jako nejlepší.

- *Nehomogenní osvětlení* - Sada *BSDS300 modified* je složena z reálných obrázků, které nemají homogenní osvětlení. A díky tomu, že color LBP a oponent color LBP jsou na osvětlení závislé, může segmentace na základě takovýchto příznaků, v případě že je míra nehomogenity osvětlení značná, selhat. Pro část těchto obrázků funguje lépe základní varianta LBP, která sice nepoužívá informace o barvě, ale je částečně invariantní vůči nehomogenímu osvětlení. Vliv nehomogenního osvětlení jsme se snažili omezit zařazením gamma korekce a a equalizace histogramu kontrastu. Tento krok sice zlepšil u části testovaných obrázku úspěšnosti segmentace A.6, ovšem u jiných vedl k nežádoucí přesegmentaci jednotlivých regionů A.7. Výsledná úspěšnost pro celou sadu tím klesla zhruba o 5%, a proto jsem ho v testech nepoužíval. V testovacím frameworku je však toto předzpracování obrazu možné zapnout.

Multiresolution LBP

V prvním experimentu byly nalezeny optimální kombinace LBP parametrů P a R pro segmentaci obrazu. Tento test se zaměří na to, jak spojení výsledků několika dílčích extrakcí, realizované konkatenací jejich příznakových vektorů, ovlivňuje úspěšnost segmentace. Otestovány proto budou všechny příznakové vektory vzniklé spojením dvou, tří nebo všech čtyř dílčích příznakových vektorů s optimální kombinací parametrů P a R. Dosažené výsledky jsou znázorněny v grafu 7.6 respektive v tabulce 7.6.



Obrázek 7.6: Graf znázorňující úspěšnost Multiresolution LBP vzniklých z optimálních kombinací P a R.

Z výsledků je jasně vidět, že příznakový vektor vzniklý spojením několika dílčích příznakových vektorů, může zvýšit úspěšnost segmentace obrazu. Toto zvýšení úspěšnosti však není u Multiresolution LBP pravidlem. Pět z jedenácti testovaných variant mělo totiž průměrnou úspěšnost nižší než nejlépe hodnocená základní varianta, z které vznikly. Tyto varianty Multiresolution LBP, které jsou horší než některá z variant, jejichž spojením vznikly, proto nebudou v rámci této práce dále používány.

Druhým aspektem hodnocení je stabilita segmentace obrazu, reprezentovaná směrodatnou odchylkou úspěšnosti pro celou datovou sadu. Všechny testované varianty Multiresolution LBP mají směrodatnou odchylku vyšší než jedna ze základních variant, jejichž složením vznikly, a tudíž jsou méně stabilní.

Pro další experimenty byly vybrány Multiresolution varianty $P8R1|P12R2$, $P8R1|P16R3$, $P8R1|P12R2|P16R3$.

Experimenty s extrakčním oknem a kontrastem v LBP

První část tohoto experimentu se zabývá možností zkombinování LBP příznaků s lokálním kontrastem obrazu pomocí VAR respektive LBPV příznaků. Druhá pak zkoumá úspěšnost kruhového a gaussovského extrakčního okna.

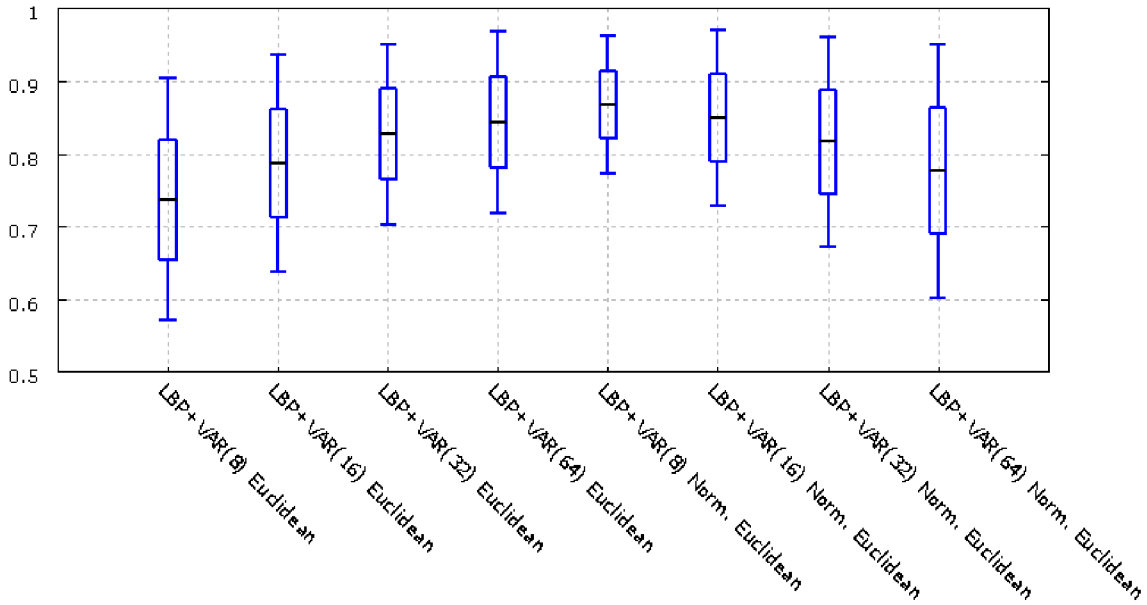
Nejdříve otestuji, jak při spojení LBP a VAR závisí úspěšnost segmentace obrazu na počtu kvantovacích hladin VAR. Tento vztah bude zkoumán v souvislosti s použitou metrikou vzdálenosti v FCM, protože s ní bezprostředně souvisí. Kromě v této práci používané Normalizované Euklidovské vzdálenosti otestuje i nenormalizovanou Euklidovskou vzdálenost, která se v FCM obvykle používá.

Výsledky experimentů obsahuje tabulka 7.7 a jsou také zaneseny v grafech 7.7 a 7.8.

První věcí, která je z výsledků 7.7 zřejmá, je, že se zvyšujícím se počtem kvantizačních hladin dochází při použití Euklidovské metriky ke zvýšení úspěšnosti segmentace. Toto chování je očekávané, protože se zvyšujícím se počtem hladin klesá kvantizační chyba, a

Název varianty	Průměr ± směrodatná odchylka
$P8R1\|P12R2$	0.8300 ± 0.0677
$P8R1\|P16R3$	0.8324 ± 0.0677
$P8R1\|P24R4$	0.7979 ± 0.1057
$P12R2\|P16R3$	0.7811 ± 0.0852
$P12R2\|P24R4$	0.7649 ± 0.1025
$P16R3\|P24R4$	0.7285 ± 0.1002
$P8R1\|P12R2\|P16R3$	0.8288 ± 0.0665
$P8R1\|P12R2\|P24R4$	0.8097 ± 0.0959
$P8R1\|P16R3\|P24R4$	0.7871 ± 0.0969
$P12R2\|P16R3\|P24R4$	0.7668 ± 0.0973
$P8R1\|P12R2\|P16R3\|P24R4$	0.7985 ± 0.0965

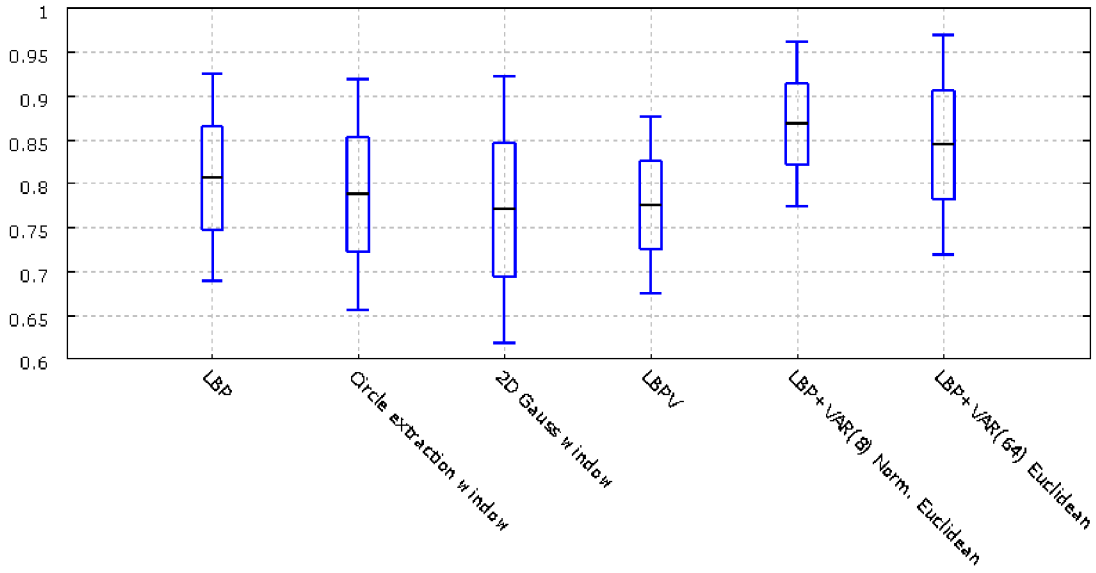
Tabulka 7.6: Tabulka výsledků úspěšnosti segmentace pro různé LBP parametry P a R. Pro každý poloměr je barevně zvýrazněn nejvhodnější počet bodů.



Obrázek 7.7: Úspěšnost segmentace obrazu na základě příznakových vektorů složených z LBP a VAR(b) a následně shlukovaných pomocí FCM s Normalizovanou a nenormalizovanou Euklidovskou vzdáleností.

tak stoupá přesnost popisu lokálního kontrastu pomocí VAR.

Naopak při použití Normalizované Euklidovské vzdálenosti se s vzrůstajícím počtem kvantizačních hladin úspěšnost segmentace snižuje. Tento jev je způsoben povahou Normalizované Euklidovské vzdálenosti. Pokud je totiž počítána vzdálenost dvou bodů touto metrikou, je dílčí rozdíl v každé dimenzi normován směrodatnou odchylkou nad vstupními daty. Což má za následek, že každá dimenze ovlivňuje vypočtenou vzdálenost stejnou vahou. Jestliže je VAR kvantován na vyšší počet hladin, zvýší se samozřejmě i dimenze prostoru



Obrázek 7.8: Experimenty s tvarem a váhováním extrakního okna a kontrastem v LBP.

Název varianty	Průměr ± směrodatná odchylka
$LBP + VAR(8)$ Euclidean	0.7370 ± 0.0844
$LBP + VAR(16)$ Euclidean	0.7865 ± 0.0748
$LBP + VAR(32)$ Euclidean	0.8270 ± 0.0620
$LBP + VAR(64)$ Euclidean	0.8440 ± 0.0624
$LBP + VAR(8)$ Normalized Euclidean	0.8676 ± 0.0472
$LBP + VAR(16)$ Normalized Euclidean	0.8493 ± 0.0603
$LBP + VAR(32)$ Normalized Euclidean	0.8165 ± 0.0722
$LBP + VAR(64)$ Normalized Euclidean	0.7768 ± 0.0873
Kruhové extrakční okno	0.7873 ± 0.0658
Váhování extrakčního okna 2D Gaussovou funkcí	0.7700 ± 0.0762
LBPV	0.7985 ± 0.0965

Tabulka 7.7: Tabulka výsledků závislosti úspěšnosti segmentace obrazu na počtu kvantizačních hladin VAR texturních příznaků. Také obsahuje výsledek pro LBPV a kruhové respekrive gaussovské extrakční okno. Všechny tyto experimenty proběhly s nastavením $P8R1$.

příznakových bodů. Následkem toho klesne vliv LBP na vypočtenou vzdálenost z 50% pro osm hladin na 8% pro šedesát čtyři hladin (pro nastavením $P8R1$). A proto klesne i celková úspěšnost segmentace.

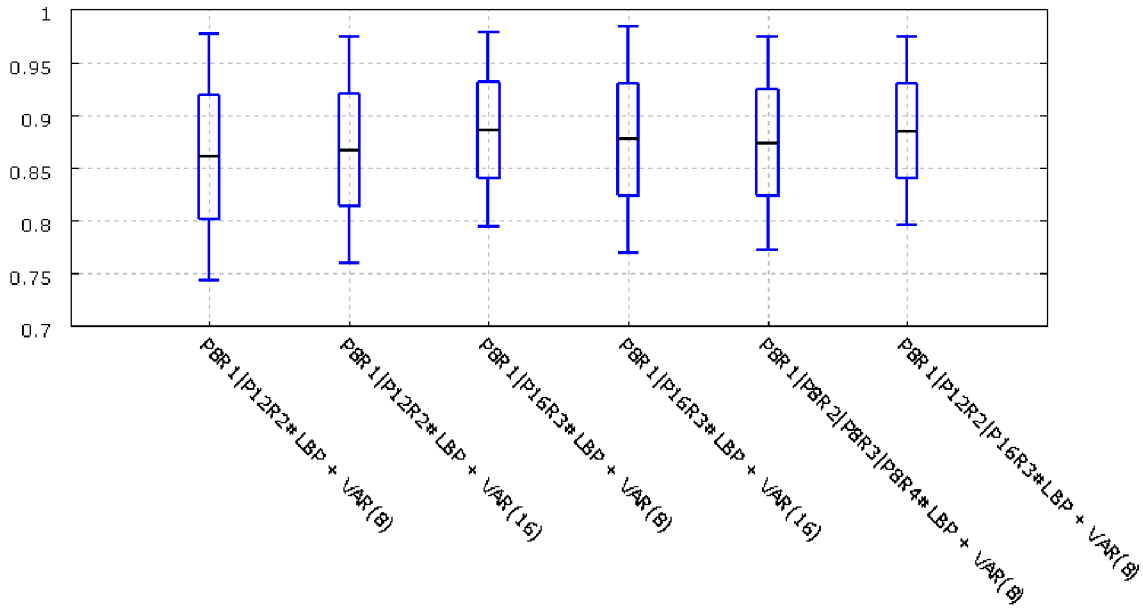
Z 7.8 je patrné, že úspěšnost segmentace obrazu se při použití gaussovského okna nebo kruhového extrakčního okna nezvýší, ale naopak sníží. LBPV sice narozdíl od VAR nepotřebuje kvantování, ale jeho úspěšnost je také nižší. Proto ani jedna z těchto tří variant nebude dále testována.

Ze všech testovaných variant pak měla nejvyšší průměrnou úspěšnost a zároveň nejnižší směrodatnou odchylku *LBP + VAR(8) Normalized Euclidean*.

Multiresolution LBP + VAR

V předchozích experimentech jsem ověřil, že Multiresolution LBP i LBP zkombinované s VAR příznaky zlepšují výslednou úspěšnost segmentace obrazu. Proto je opodstatněná domněnka že spojením těchto dvou vylepšení lze dosáhnout ještě dalšího zvýšení úspěšnosti segmentace obrazu, což je hlavním cílem tohoto experimentu. Dosažené výsledky jsou v grafu 7.9 nebo tabulce 7.8.

Ve všech těchto testech je při shlukování pomocí FCM použita Normalizovaná Euklidovská vzdálenost.



Obrázek 7.9: Porovnání všek kombinací P a R pro Multiresolution LBP.

Název varianty	Průměr ± směrodatná odchylka
$P8R1\|P12R2\#LBP + VAR(8)$	0.8602 ± 0.0587
$P8R1\|P12R2\#LBP + VAR(16)$	0.8669 ± 0.0536
$P8R1\|P16R3\#LBP + VAR(8)$	0.8860 ± 0.0459
$P8R1\|P16R3\#LBP + VAR(16)$	0.8768 ± 0.0538
$P8R1\|P8R2\|P8R3\|P8R4\#LBP+VAR(8)$	0.8735 ± 0.0505
$P8R1\|P12R2\|P16R3\#LBP + VAR(8)$	0.8848 ± 0.0445

Tabulka 7.8: Úspěšnost segmentace obrazu při zkombinovaní Multiresolution LBP a VAR texturních příznaků.

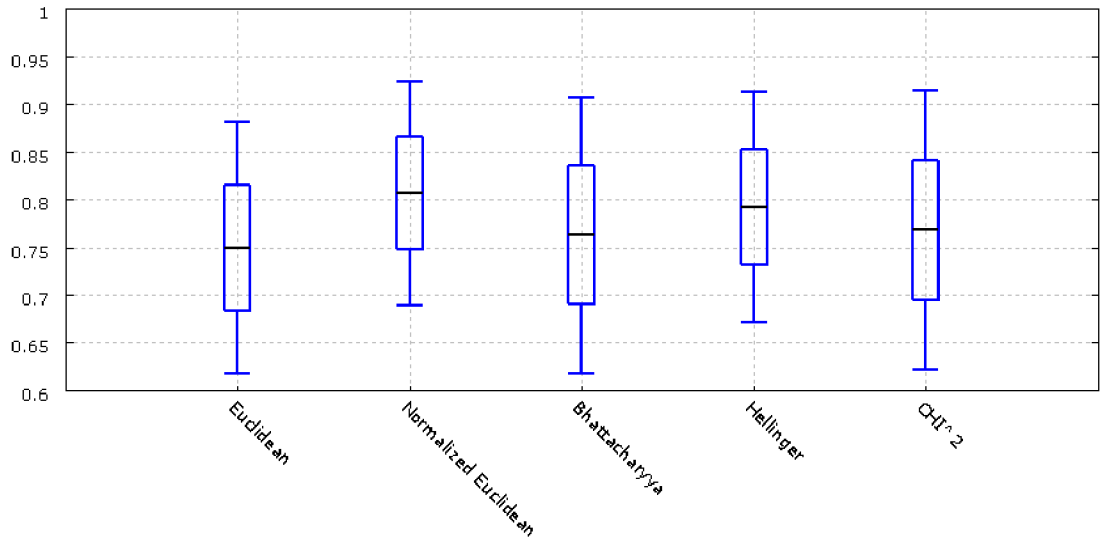
Ve všech testech vedlo spojení Multiresolution LBP a VAR ke zvýšení průměrné úspěšnosti segmentace a snížení směrodatné odchylky. To znamená, že zlepšuje úspěšnost i stabilitu výsledné segmentace obrazu. Toto zlepšení je však mírnější než zlepšení přinášející

samotné Multiresolution LBP nebo LBP + VAR oproti základním LBP. Jako nejlepší nalezenou variantu jsem vyhodnotil $P8R1|P16R3\#LBP+VAR(8)$. Ukázka segmentace obrazu s takovýmto nastavením je na obrázku A.1.

7.4 Srovnání shlukovacích algoritmů

Metriky vzdálenosti v FCM

Algoritmus FCM používá k výpočtu matice příslušnosti vzdálenost $\|*\|$. Pro různé metriky pak vzniknou různé matice příslušnosti. Proto se liší i celková úspěšnost segmentace, jak je vidět v grafu 7.10 respektive tabulky 7.9.



Obrázek 7.10: Porovnání FCM pro různé metriky vzdálenosti.

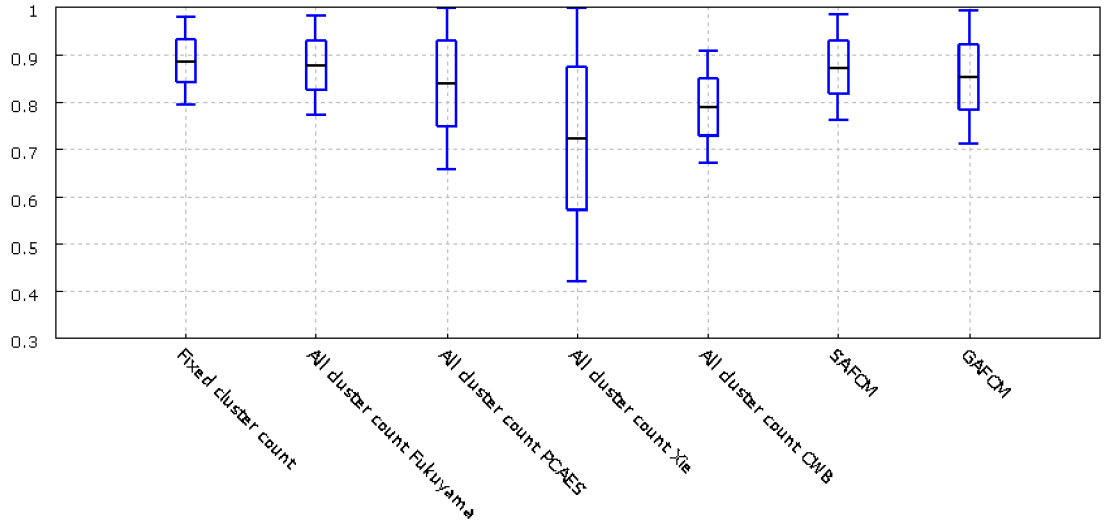
Metrika vzdálenosti FCM	Průměr ± směrodatná odchylka
Euklidovská vzdálenost	0.7495 ± 0.0657
Normalizovaná Euklidovská vzdálenost	0.8065 ± 0.0589
Bhattacharyyova vzdálenost	0.7630 ± 0.0724
Hellingerova vzdálenost	0.7915 ± 0.0603
χ^2	0.7681 ± 0.0733

Tabulka 7.9: Vliv použité metriky vzdálenosti v FCM na celkovou úspěšnost segmentace obrazu.

V tomto testu nejlépe dopadla Normalizovaná Euklidovská vzdálenost, která dosáhla nejvyšší průměrné úspěšnosti a zároveň nejnižší směrodatné odchylky. Naproti tomu nenormalizovaná Euklidovská vzdálenost, která se v FCM nejběžněji používá, má ze všech pěti testovaných metrik nejnižší průměrnou úspěšnost a z hlediska stability je až třetí. To může být způsobeno částečně tím, že FCM založené na Euklidovské vzdáleností vytváří shluky sférického charakteru. Přičemž shluky bodů v příznakovém prostoru takovýto charakter mít nemusí.

Metody pro neznámý počet shluků

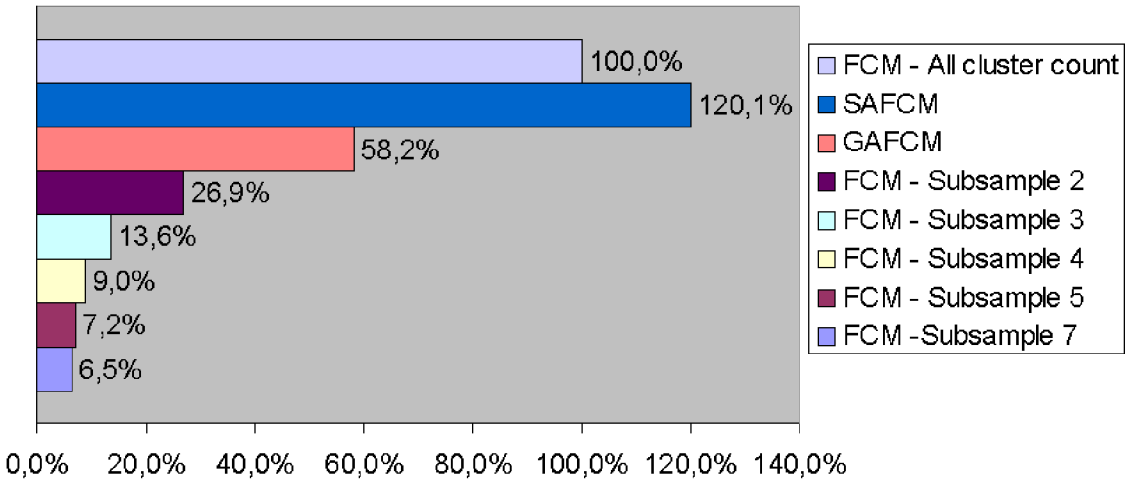
Tento experiment má tři části, nejdříve bude nalezen nejvhodnější *Cluster validity index* pro FCM. Poté se vyhodnotí úspěšnost a relativní doba běhu metod SAFCM a GAFCM založených na nejvhodnějším indexu. Na závěr pak otestuji jak *Subsamplování příznakových vektorů* ovlivňuje úspěšnost a délku běhu programu. Všechny testy proběhnou pro nastavení $P8R1|P16R3\#LBP+VAR(8)$. Jejich uspěšnost je znázorněna v grafu 7.11 a vyhodnocení délky jejich běhu v 7.12 respektive tabulce 7.11.



Obrázek 7.11: Srovnání úspěšnosti segmentace obrazu pro známý a neznámý počet shluků. V případě neznámého počtu shluků je použito několik *Cluster validity indexů* a několik různých metod nalezení optimálního počtu shluků.

Varianta shlukování	Průměrná úspěšnost	Relativní doba běhu
Fixed cluster count	0.8860 ± 0.0459	4.8%
All cluster count - Fukuyama	0.8772 ± 0.0524	99.8%
All cluster count - Xie Beni	0.7224 ± 0.1515	99.7%
All cluster count - PCAES	0.8377 ± 0.0906	96.2%
All cluster count - CWB	0.7889 ± 0.0598	98.3%
SAFCM	0.8726 ± 0.0561	120%
GAFCM	0.8521 ± 0.0700	58.2%
Subsample 2	0.8769 ± 0.0505	26.9%
Subsample 3	0.8761 ± 0.0553	13.6%
Subsample 4	0.8759 ± 0.0534	9.0%
Subsample 5	0.8749 ± 0.0558	7.2%
Subsample 7	0.8430 ± 0.0993	6.5%

Tabulka 7.10: Výsledná úspěšnost segmentace obrazu a relativní délka běhu se všemi testovanými shlukovacími algoritmy.



Obrázek 7.12: Srovnání relativní doby běhu segmentace obrazu pro různé shlukovací algoritmy.

Z grafu 7.11 je jasně vidět, že pro sadu *Outex US 00000* a námi testované nastavení je nejlepším *Cluster validity indexem Fukuyama-Sugeno*. Je totiž nutné vzít v úvahu, že texturní příznaky získané ruznými druhy extrakce, nebo pro rozdílné datové sady mohou vytvářet shluky s různým prostorovým uspořádáním. A protože každý *Cluster validity index* hodnotí "úspěšnost"shlukování na základě jiných kritérií, nelze žádný *Cluster validity index* označit za nejlepší a univerzálně ho používat, ale lze pouze prohlásit konkrétní index za nejvhodnější pro dané užití. Z tohoto důvodu bude použit pro sadu *Outex US 00000* *Fukuyama-Sugeno* index a pro *BSDS300 modified CWB* index. Ukázku toho, jak ovlivní segmentaci obrazu výběr nevhodného *Cluster validity indexu* je na obrázku A.2.

Úspěšnost segmentace obrazu metodou *All cluter count - Fukuyama* je jen o necelé procento nižší, než v případě, kdy je správný počet shluků dopředu známý, což značí, že byl *Fukuyama-Sugeno* index vybrán pro tento účel správně. Průměrná úspěšnost metody *SAFCM* je pak s přihlédnutím k možné nestabilitě FCM prakticky stejná jako *All cluter count - Fukuyama*. Naproti tomu úspěšnost *GAFCM* je o tři procenta nižší než *All cluter count - Fukuyama*.

Relativní doba běhu je vztažená k délce běhu metody *All cluter count - Fukuyama*. Tato doba se vždy skládá ze dvou částí - v té první probíhá hledání optimálního počtu shluků a ve druhé shlukování pomocí FCM pro tento počet. Protože však shlukovaní do většího počtu shluků trvá déle, může se lišit i doba běhu metody *All cluter count* pro různé *Cluster validity indexy* v závislosti na tom, jaký počet shluků vyhodnotí jako optimální. Doba hledání optimálního počtu shluků se zvyšuje s rostoucí délkou intervalu možného počtu shluků a tak narůstá i podíl této části na celkové době běhu. Doba běhu *Fixed cluster count* je vlastně doba trvání druhé části, a tak se pod tuto hodnotu nemůže žádna metoda pracující s neznámým počtem shluků dostat.

Relativní doba běhu *SAFCM* je 120%. To znamená, že při stejné úspěšnosti jako *All cluter count - Fukuyama* přináší pouze prodloužení doby o 20%. Na druhé straně doba běha *GAFCM* je pouze 58.2%. Tato metoda má sice o tři procenta nižší úspěšnost než referenční, ale její zrychlení je značné, což je velmi podstatné a činí to z *GAFCM* variantu použitelnou v případě, že záleží i na rychlosti segmentace obrazu.

Poslední součástí tohoto experimentu je otestování vlivu faktoru subsamplování příznakových vektorů na výslednou úspěšnost segmentace obrazu a relativní dobu jejího běhu. Subsamplování příznakových vektorů lze použít na všechny tři testované metody nalezení optimálního počtu shluků (*All cluter count*, *SAFCM* i *GAFCM*). V těchto testech ho budeme používat v kombinaci s metodou *All cluter count*. Z tabulky 7.11 je vidět, že průměrná úspěšnost segmentace obrazu pro faktor subsamplování od dvou do pěti klesá jen velmi mírně a znatelný pokles nastane až s faktorem sedm. Naproti tomu relativní doba běhu klesá velmi značně 7.12. Díky tomu, že pro faktor čtyři a více je doba hledání optimálního počtu shluků srovnatelná s dobou trvání následného shlukování, která je vždy stejná, nepřináší další zvyšování faktoru subsamplování již tak značné snížení relativní doby běhu. Podíl doby hledání optimálního počtu shluků činí bez subsamplování 95.2%, zatímco při použití subsamplování s faktorem sedm je to pouze 26.2% celkové doby běhu.

Použití subsamplování příznakových vektorů ale má i svojí nevýhodu. Tou je nutnost dostatečného počtu zástupců každého shluku po subsamplování. Tento požadavek závisí na celkovém počtu příznakových vektorů a předpokládaném počtu shluků. Zaprvé je nutné vzít v úvahu, že počet příznakových vektoru kvadraticky klesá s faktorem subsamplování, a tak musíme volit pro obrázky s menším rozlišením nižší faktor subsamplování. Druhou věcí je předpokládaný počet shluků, který bezprostředně souvisí s účelem segmentace obrazu. Pokud je použita k detekci oblasti zájmu (např. detekce otisku prstu), bude předpokládaný počet shluků velmi nízký. Naproti tomu ve chvíli, kdy je použíto k rozdělení reálného obrazu na jednolité části, které budou dále na základě jiného postupu spojovány, je předpokládaný počet shluků velmi vysoký. Se vzrůstajícím předpokládaným počtem shluků pak musí faktor subsamplování příznakových vektorů klesat, aby bylo vyhověno podmínce, že po subsamplování musí zůstat dostatečný počet zástupců každého shluku.

Defuzzifikace výstupu FCM

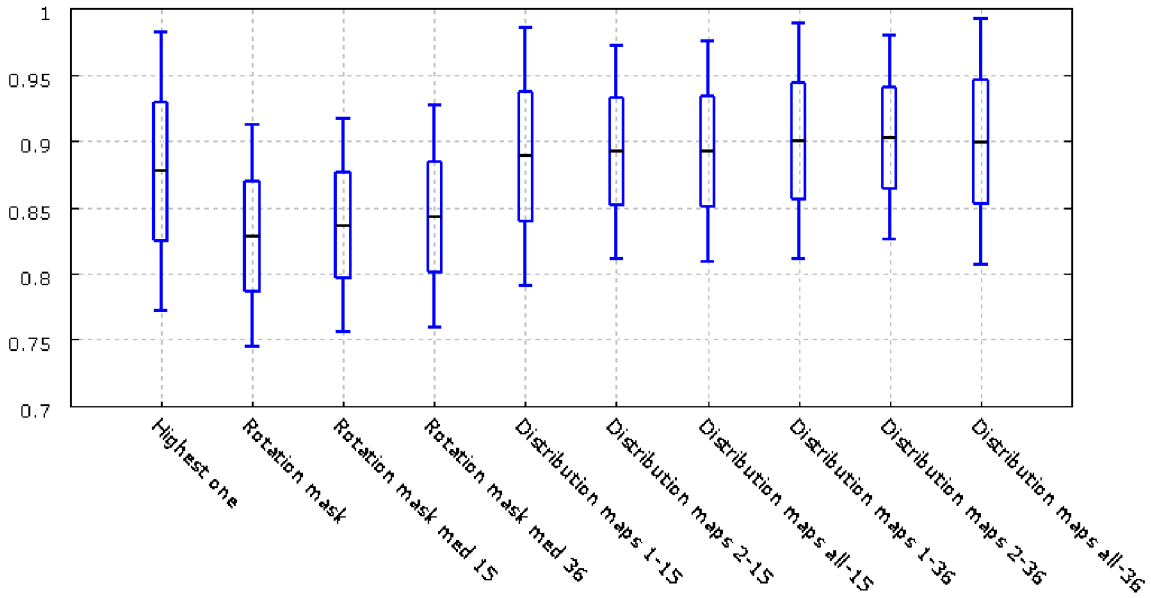
Výsledkem FCM je matice středů shluků a matice příslušnosti. Převod na výstupní obrázek, neboli defuzzifikaci, lze provést několika způsoby. Všechny tyto testy používají nastavení $P8R1|P16R3\#LBP+VAR(8)$. Označení "Rotation mask med X"znamená použití metody *Rotation mask* a následného aplikování mediánového filtru o velikosti X. "Distribution maps X - Y"pak X značí počet použitých hodnot z matice příslušnosti (jedna, dvě nejlepší nebo všechny) a Y velikost okolí v příslušné distribuční mapě, do kterého se každá hodnota promítne.

Z 7.13 je vidět, že nejlepšího výsledku dosáhla metoda defuzzifikace, která zapisuje do distribučních map dvě nejvyšší hodnoty z matice příslušnosti pro každý bod pracující s okolím velikosti 36. U metody *Distribution maps* se také ukázalo, že počet hodnot z matice příslušnosti, které se zapisují do distribučních map, nemá na výsledek takový vliv jako velikost okolí.

Metoda *Rotation mask* má výslednou úspěšnost nejnižší ze všech testovaných. S použitím mediánového filtru se sice její úspěšnost mírně zvyšuje, ale i v tomto případě je její úspěšnost nižší než u klasického vyhodnocení *Highest one*, oproti kterému měla podle předpokladů přinést zlepšení.

Porovnání shlukovacích algoritmů

Tento experiment byl prováděn s nastavením $P8R1$, aby byl omezen vliv počtu kvantovacích hladin pro VAR na různé metriky, které tyto shlukovací algoritmy používají. Výsledky jsou zobrazeny grafu 7.14 a tabulce 7.12.



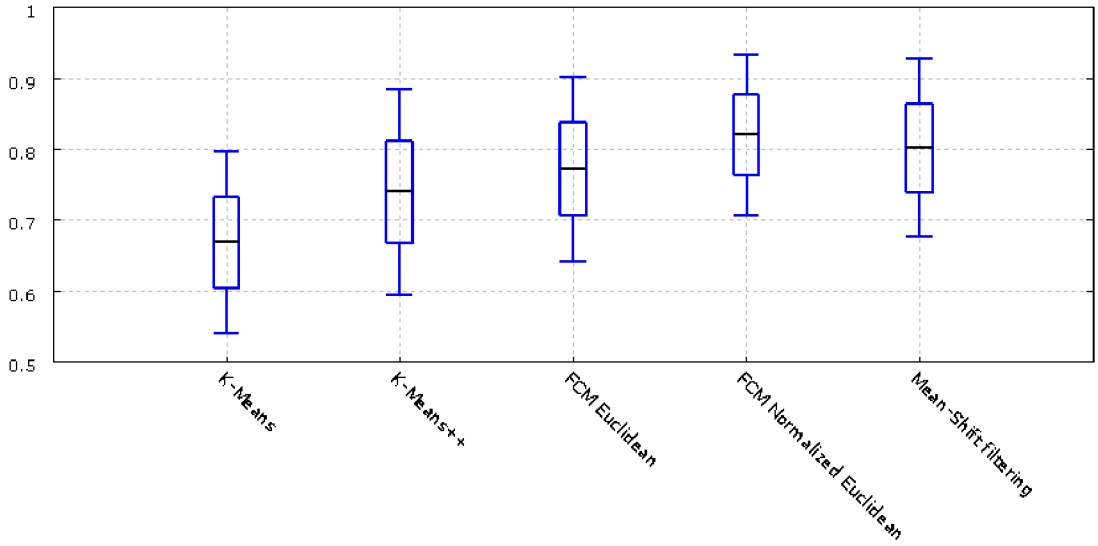
Obrázek 7.13: Graf úspěšnosti segmentace pro různé varianty vyhodnocení výstupu FCM.

Metoda defuzzifikace	Průměr ± směrodatná odchylka
Highest one	0.8772 ± 0.0524
Rotation mask	0.8284 ± 0.0419
Rotation mask med 15	0.8366 ± 0.0404
Rotation mask med 35	0.8430 ± 0.0420
Distribution maps 1 - 15	0.8885 ± 0.0485
Distribution maps 2 - 15	0.8922 ± 0.0402
Distribution maps all - 15	8925 ± 0.0417
Distribution maps 1 - 36	0.9002 ± 0.0443
Distribution maps 2 - 36	0.9026 ± 0.0286
Distribution maps all - 36	0.8996 ± 0.0463

Tabulka 7.11: Tabulka úspěšnosti segmentace obrazu pro různé metody defuzzifikace výstupu FCM.

Shlukovací algoritmus	Průměr ± směrodatná odchylka
K-Means	0.6680 ± 0.0646
K-Means++	0.7391 ± 0.0723
FCM Euclidean	0.7714 ± 0.0652
FCM Normalized Euclidean	0.8203 ± 0.0569
Mean-Shift filtering	0.8016 ± 0.0629

Tabulka 7.12: Tabulka úspěšnosti segmentace obrazu pro různé metody shlukování příznakových vektorů.



Obrázek 7.14: Graf úspěšnosti segmentace pro shlukovací algoritmy.

Z těchto výsledků je zřetelně vidět, že nejlepší z testovaných shlukovacích algoritmů je Fuzzy C-means založené na Normalizované Euklidovské vzdálenosti, který v rámci této práce používáme ve většině experimentů. Zajímavé je, že Mean-Shift filtrování, které mělo učinit shluky lépe oddělitelné, nesplnilo předpoklady a výslednou úspěšnost segmentace nezvýšilo.

7.5 Shrnutí výsledků

Ve všech experimentech je pomocí úspěšnosti segmentace hodnocena vhodnost testovaných metod nebo jejich parametrů k tomuto účelu. Některé experimenty využívají část výsledků předcházejících experimentů.

Prvním experimentem bylo nalezení vhodných parametrů LBP. Toto hledání optimálních parametrů bylo koncipováno jako hledání optimálního počtu bodů k popisu okolí pomocí LBP pro vzdálenost jedna až čtyři. Nalezenými optimálními kombinacemi pro sadu *Outex US 00000* jsou $LBP^{riu_2}_{8,1}$, $LBP^{riu_2}_{12,2}$, $LBP^{riu_2}_{16,3}$, $LBP^{riu_2}_{24,4}$, což potvrzuje předpoklad, že optimální počet bodů se s rostoucí vzdáleností zvyšuje.

Dále jsem testoval Multiresolution LBP realizované konkatenací příznakových vektorů vzniklých extrakcí základních LBP s optimálním nastavením nalezeným v předcházejícím experimentu. Ze všech těchto kombinací byla nadpoloviční většina úspěšnější než nejlepší z dílčích variant, ze kterých vznikly. To znamená, že při tvorbě Multiresolution LBP se musí správně zvolit dílčí varianty, aby bylo dosaženo zlepšení výsledku. Nejlépe v tomto testu dopadla kombinace $LBP^{riu_2}_{8,1}$ a $LBP^{riu_2}_{16,3}$.

Testování LBP pracujících s barvou proběhlo na sadě *BSDS300 modified*. Bylo zjištěno, že transformace prostoru má značný vliv na výslednou úspěšnost segmentace. Tento vliv je větší na Oponent color LBP než na Color LBP. Oponent color LBP měly ve většině testovaných barevných prostorů vyšší úspěšnost než Color LBP. V tomto experimentu byla jako nejlepší vyhodnocena varianta Oponent color LBP pracující s barevným prostorem CIE XYZ. Tato sada se skládá z reálných obrázků s nehomogenním osvětlením, kde mohou LBP pracující s barvou selhat. Tento problém jsem se snažil vyřešit normalizací osvětlení pomocí

gamma korekce a equalizace histogramu. Tento krok vedl u některých obrázků ke zlepšení A.6, ale u jiných naopak výsledek zhoršil A.7. Pro celou sadu se při použití normalizace osvětlení úspěšnost segmentace mírně zhoršila, a proto jsem jí při testech nepoužíval. Další problém tvořily obrázky, jejichž část byla rozostřená. S tímto problémem si neporadila ani základní varianta LBP a segmentační program je většinou rozdělil pouze na rozostřenou a nerozostřenou část.

Jeden z experimentů se zabýval extrakčním oknem u LBP. Extrakční okno má obvykle čtvercový tvar. Prvním návrhem na zlepšení bylo váhování bodů uvnitř tohoto okna 2D Gaussovou funkcí, druhým pak použití kruhového extrakčního okna. Obě tyto modifikace ovšem dosáhly nižší úspěšnosti segmentace obrazu než klasické čtvercové okno. A díky tomu, že pro klasické čtvercové okno je i jednodušší postup extrakce, rozhodl jsem se obě testované modifikace v rámci této práce dále nepoužívat.

Dalším důležitým faktem vyplývajícím z výsledků je možnost velmi výrazného zvýšení úspěšnosti segmentace při zkombinování LBP s informací o kontrastu. Jako kombinaci LBP a kontrastu jsem testoval texturní příznak LBPV a konkatenaci LBP a VAR příznaků. Pro LBPV bylo dosaženo ještě o trochu horších výsledků než pro samotné LBP. Naproti tomu konkatenace LBP a VAR přinesla výrazné zlepšení. V tomto případě má pro úspěšnost segmentace rozhodující význam počet kvantovacích hladin VAR. Při použití Normalizované Euklidovské metriky v FCM se jako optimální ukázalo osm kvantovacích hladin, zatímco pro Euklidovskou vzdálenost je to 64 a více.

Zkombinování Multiresolution LBP a VAR bylo pak jen dalším logickým krokem vedoucím ke zvýšení úspěšnosti segmentace obrazu. Nejlépe v tomto testu dopadla varianta $P8R1|P16R3\#LBP + VAR(8)$.

Zkombinováním LBP a VAR lze dosáhnout optimálního popisu textury, který při segmentaci obrazu s homogenním osvětlením vede k velmi dobrému oddělení oblastí s různou texturou. Při nehomogenním osvětlení ale může tento popis vést k nežádoucím rozdělení oblastí s jednotnou texturu na různě osvětlené části.

Jeden z experimentů se zabýval vhodností použité metriky vzdálenosti při výpočtu matice příslušnosti ve Fuzzy C-Means. Ze všech pěti testovaných metrik dopadla nejlépe Normalizovaná Euklidovská vzdálenost, zatímco běžná Euklidovská vzdálenost skončila až na posledním místě. S použitou metrikou souvisí bezprostředně i optimální počet kvantovacích hladin VAR, diskutovaný výše.

Další experiment se zabýval správným počtem shluků v FCM. Základním řešením tohoto problému "hrubou silou"je realizovat shlukování pro každý přípustný počet shluků a z jejich výsledků vybrat nejlepší, který se použije. První část experimentu se zabývala metodou hodnocení úspěšnosti segmentace, neboli *Cluster validity indexem*. Pro sadu *Outex US 00000* jsem jako nejlepší vyhodnotil *Fukuyama-Sugeno* index, který pro ní funguje téměř bezchybně. Pro sadu *BSDS300 modified* jsem jako nejlepší vyhodnotil *CWB* index, který však občas zvolí nižší počet shluků, než by bylo třeba.

Druhá část porovnává navržené alternativy řešení v podobě inicializace pomocí Simulovaného žíhání (SAFCM) nebo Genetického algoritmu (GAFCM) se základním řešením "hrubou silou", za použití *Fukuyama-Sugeno* indexu ve všech třech metodách. Úspěšnost dosažená pomocí SAFCM byla stejná jako v případě základního řešení, ale relativní doba běhu tohoto algoritmu je o 20% vyšší. Naproti tomu GAFCM měla úspěšnost o několik procent horší, ale doba běhu je oproti základnímu řešení zhruba poloviční. To znamená, že GAFCM je výbornou alternativou k běžnému řešení v případě, že jde i o rychlost segmentace. Subsamplování příznakových vektorů, se ukázalo jako další možné zrychlení výpočtu optimálního počtu shluků které ale nemusí být vždy použitelné.

V neposlední řadě byla zkoumána i defuzzifikace FCM, kde se ukázalo, že metoda "Distribution maps"pracující i s informacemi o geometrickém uspořádání bodů v obraze dosahuje lepších výsledků, než pouhé vyhodnocení nejvyšší příslušnosti.

Možnosti budoucího vývoje

Vývoj této práce se může dále ubírat dvěma základními směry. Prvním je zkoumání dalších metod extrakce texturních příznaků, například LBP/C, Local Ternary Patterns nebo vytváření Multiresolution LBP namísto konkatenace Celulárním automatem. Další možností by bylo zaměřit se na LBP popisující prostorové textury ve 3D objemovým datech. V případě užití na obrázky s nehomogenním osvětlením by bylo klíčovým krokem vhodné předzpracování obrazu, které by zachovalo původní texturu a zároveň normalizovalo osvětlení.

Druhou možností budoucího vývoje je zaměřit se na shlukování příznakových vektorů. Zde je z výsledku testů zřejmé, že použitá metrika vzdálenosti má na úspěšnost segmentace značný vliv, ale přitom se jí nevěnuje dostatek pozornosti. Jednou z možností vývoje segmentace je i úmyslné přesegmentování obrazu na malé uniformní regiony, které by byly následně na základě podobnosti spojovány, čímž se zamezí možnosti výběru nesprávného počtu shluků. Dalším možným vylepšením by bylo využít k normalizaci osvětlení nějaký komplexnější systém předzpracování obrazu, jakým je například Retinex, aby bylo výsledné řešení robustnější.

Díky tomu, že je možné výpočet FCM snadno paralelizovat, mohl by také být implementován přímo v hardwaru, a tak provádět segmentaci obrazu v reálném čase.

Další možností vývoje by také mohlo být zpřesnění hranic mezi regiony. Takovéto zpřesnění by bylo možné realizovat například detekcí hran v blízkosti stavájících hranic regionů.

Kapitola 8

Závěr

Cílem této práce bylo zkoumání problematiky segmentace obrazu na základě textury. Teoretická část se nejdříve zabývala popisem textury pomocí několika různých texturních příznaků. Poté byly rozebrány shlukovací algoritmy, přičemž jsem se zaměřil na Fuzzy C-Means a navrhl dvě jeho nové varianty (jednu zcela původní) pracující s neznámým počtem shluků. V praktické části jsem vytořil framework sloužící k segmentaci obrazu, jehož funkčnost byla předvedena na dvou velmi rozdílných datových sadách. Framework byl implementován v jazycích C++ a Python a je vytvořen tak, aby šel snadno rozšiřovat a modifikovat. Navíc umožňuje použít příznakový vektor, sestavený z libovolného počtu dílčích příznakových vektorů různých druhů a nastavení.

Z experimentů vyplývá, že pro obrázky založené na textuře a s homogenním osvětlením je nejvhodnějším textutním příznakem Multiresolution LBP + VAR, konkrétně nastavení v textu označované $P8R1|P16R3\#LBP+VAR(8)$. Pro něj je úspěšnost segmentace velmi dobrá a jediným nedostatkem je drobná nepřesnost hranice regionů.

Při testování barevných LBP byl zjištěn značný vliv použitého barevného prostoru na úspěšnost segmentace. Jako nejlepší varianta se jeví Oponent color LBP pracující s barevným prostorem CIE XYZ. Na některých obrázcích s nehomogenním osvětlením mohou ovšem LBP selhat. K eliminaci tohoto jevu by bylo vhodné využít nějaký komplexnější systém předzpracování obrazu (to je i jeden z námětů pro případný další rozvoj této práce).

Z provedených testů je také jasně vidět, že použitá metrika vzdálenosti má na výsledek FCM značný vliv. Ze všech testovaných metrik dopadla nejlépe Euklidovská vzdálenost normalizovaná směrodatnou odchylkou nad vstupními daty.

Nová varianta Fuzzy C-Means, která nepotřebuje znát počet shluků (GAFCM), dosáhla zhruba stejné úspěšnosti jako provedení shlukování pro každý přípustný počet shluků a následné vybrání nejvhodnějšího. Hlavní rozdíl je v tom, že GAFCM běží přibližně o polovinu rychleji než řešení "hrubou silou", a tak lze tento algoritmus použít zejména v aplikacích, pro které je klíčová doba běhu.

Další vývoj této práce by mohl využít například toho, že výpočet FCM lze snadno paralelizovat, díky čemuž by bylo možné ho implementovat přímo v hardwaru, a tak provádět segmentaci obrazu v reálném čase.

Díky provedeným experimentům byly zjištěny zajímavé vlastnosti týkající se texturních příznaků a shlukovacích algoritmů, které se dají použít i v praxi. Vytvořený framework by mohl sloužit například pro automatickou kontrolu kvality, vyhodnocování satelitních snímků, detekci otisků prstů a mnoho dalších aplikacích.

Literatura

[1] Al-Sultan, K.; Fedjki, C.: A tabu search-based algorithm for the fuzzy clustering problem. *Pattern Recognition*, ročník 30, č. 12, 1997: s. 2023–2030.

[2] Al-Sultan, K.; Selim, S.: A global algorithm for the fuzzy clustering problem. *Pattern Recognition*, ročník 26, č. 9, 1993: s. 1357–1361.

[3] Alata, M.; Molhim, M.; Ramini, A.: Optimizing of fuzzy c-means clustering algorithm using GA. *Update*, ročník 1, č. 5, 2008.

[4] Alia, O.; Mandava, R.; Ramachandram, D.; aj.: Harmony search-based cluster initialization for fuzzy c-means segmentation of mr images. In *TENCON 2009-2009 IEEE Region 10 Conference*, IEEE, 2009, s. 1–6.

[5] Alsabti, S., Khaled; Ranka; Singh, V.: An efficient k-means clustering algorithm. Electrical Engineering and Computer Science. Paper 43., 2009.

[6] Arthur, D.; Vassilvitskii, S.: k-means++: the advantages of careful seeding. 2007.
URL http://surface.syr.edu/eecs/43

[7] Bezdek, J.: Cluster validity with fuzzy sets. Volume:3, Issue: 3, Publisher: Taylor, Francis, Pages: 58, 1974.

[8] Bezdek, J.: Mathematical models for systematics and taxonomy. In: Estabrook, G., Proc. 8th Internat. Conf. Numerical Taxonomy Freeman, San Francisco, CA, pp. 143, 1975.

[9] Bezdek, J. C.: Pattern recognition with fuzzy objective function algorithms. New York, 256. p., 1981.

[10] Blickle, T.; Thiele, L.: A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, ročník 4, č. 4, 1996: s. 361–394.

[11] Bradski, G.; Kaehler, A.: Learning OpenCV: Computer vision with the OpenCV library. 2008.
URL http://opencv.willowgarage.com/wiki/

[12] Chinchor, N.: MUC-4 evaluation metrics. In *Proceedings of the 4th conference on Message understanding*, MUC4 '92, Stroudsburg, PA, USA: Association for Computational Linguistics, 1992, ISBN 1-55860-273-9, s. 22–29, doi:http://dx.doi.org/10.3115/1072064.1072067.
URL http://dx.doi.org/10.3115/1072064.1072067

[13] Cohen, J.; aj.: A coefficient of agreement for nominal scales. *Educational and psychological measurement*, ročník 20, č. 1, 1960: s. 37–46.

[14] Comaniciu, D.; Meer, P.: Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, ročník 24, May 2002: s. 603–619, ISSN 0162-8828, doi:10.1109/34.1000236.

[15] Dunn, J. C.: A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics 3: 32-57, 1973.

[16] Fowlkes, C.; Martin, D.; Malik, J.: Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, ročník 2, IEEE, 2003, s. II–54.

[17] Fukuyama, Y.; Sugeno, M.: A new method of choosing the number of clusters for the fuzzy c-mean method. In: Proc. 5th Fuzzy Syst. Symp., pp.248, 1989.

[18] Georgescu, B.; Shimshoni, I.; Meer, P.: Mean shift based clustering in high dimensions: A texture classification example. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, IEEE, 2003, s. 456–463.

[19] Guo, Z.; Zhang, L.; Zhang, D.: Rotation invariant texture classification using LBP variance (LBPV) with global matching. *Pattern Recognition*, ročník 43, č. 3, 2010: s. 706–719.

[20] Hall, L.; Ozyurt, I.; Bezdek, J.: Clustering with a genetically optimized approach. *Evolutionary Computation, IEEE Transactions on*, ročník 3, č. 2, 1999: s. 103–112.

[21] Hripcsak, G.; Rothschild, A.: Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, ročník 12, č. 3, 2005: s. 296–298.

[22] Kanade, P.; Hall, L.: Fuzzy ants and clustering. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, ročník 37, č. 5, 2007: s. 758–769.

[23] M. Španěl, V. B.: *Obrazové segmentační techniky, Brno, FIT VUT v Brně*. Fit.vutbr.cz [online]. 12.10.2005, 19.1.2006 [cit. 2011-04-19], 2006. URL http://www.fit.vutbr.cz/~spanel/segmentace/

[24] Makhoul, J.; Kubala, F.; Schwartz, R.; aj.: Performance measures for information extraction. In *Proceedings of DARPA Broadcast News Workshop*, 1999, s. 249–252.

[25] Martin, D.; Fowlkes, C.; Tal, D.; aj.: A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. July 2001.

[26] Martin, D.; Malik, J.; Patterson, D.: *An Empirical Approach to Grouping and Segmentaqtion*. Computer Science Division, University of California, 2003.

[27] Maulik, U.; Bandyopadhyay, S.: Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification. *Geoscience and Remote Sensing, IEEE Transactions on*, ročník 41, č. 5, 2003: s. 1075–1081.

[28] Mäenpää, T.: *The local binary pattern approach to texture analysis extensions and applications*. Faculty of Technology University of Oulu, 2003, iSBN 951-42-7075-4.

[29] Mäenpää, T.; Ojala, T.; Pietikäinen, M.; aj.: *Robust Texture Classification by Subsets of Local Binary Patterns*. 2002, proc. 15th International Conference on Pattern Recognition, Barcelona, Spain, 3:947 - 950.

[30] P. Kruizinga, N. P.; Grigorescu, S.: *Comparison of texture features based on Gabor filters*. Institute of Mathematics and Computing Science, University of Groningen, 1999, iEEE Transactions on Image Processing.

[31] Pasáček, V.: Charakteristika 2D textur, bakalářská práce, Brno, FIT VUT v Brně. 2009.

[32] R. Hathaway, J. C. B.; Tucker, W.: An improved convergence theorem for the fuzzy c-means clustering algorithms. Vol. 3. CRC Press, Boca Raton, 1986.

[33] Ramze Rezaee, M.; Lelieveldt, B.; Reiber, J.: A new cluster validity index for the fuzzy c-mean. *Pattern recognition letters*, ročník 19, č. 3-4, 1998: s. 237–246.

[34] Rand, W.: Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 1971: s. 846–850.

[35] Shih, F.; Cheng, S.: Automatic seeded region growing for color image segmentation. *Image and Vision Computing*, ročník 23, č. 10, 2005: s. 877–886.

[36] Sumengen, B.; Manjunath, B.: Multi-scale edge detection and image segmentation. In *European signal processing conference (EUSIPCO)*, Citeseer, 2005.

[37] University of Oulu: Outex Texture Database - Outex US 00000 [online]. September 2007, , [cit. 2012-02-15].
URL http://www.outex.oulu.fi/index.php?page=segmentation

[38] Unnikrishnan, R.; Pantofaru, C.; Hebert, M.: A measure for objective evaluation of image segmentation algorithms. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, IEEE, 2005, s. 34–34.

[39] Unnikrishnan, R.; Pantofaru, C.; Hebert, M.: Toward objective evaluation of image segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ročník 29, č. 6, 2007: s. 929–944.

[40] Wang, X.; Garibaldi, J.: Simulated annealing fuzzy clustering in cancer diagnosis. *Informatica*, ročník 29, č. 1, 2005: s. 61–70.

[41] Wu, K.; Yang, M.: A cluster validity index for fuzzy clustering. *Pattern Recognition Letters*, ročník 26, č. 9, 2005: s. 1275–1291.

[42] Xie, X.; Beni, G.: A validity measure for fuzzy clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ročník 13, č. 8, 1991: s. 841–847.

[43] Zhu, C.; Bichot, C.; Chen, L.: Multi-scale color local binary patterns for visual object classes recognition. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, IEEE, 2010, s. 3065–3068.

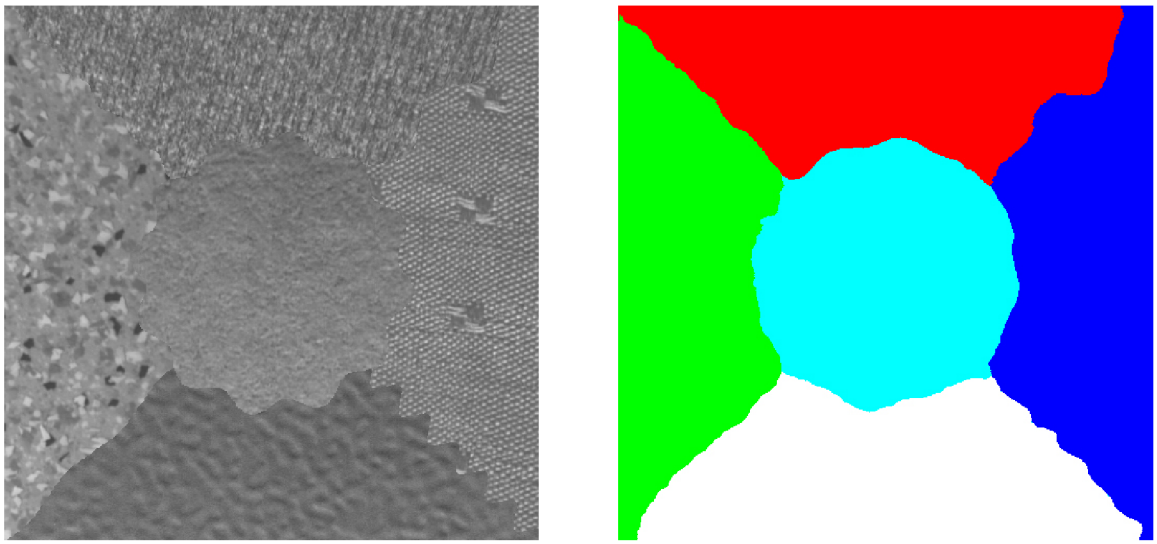
[44] Španěl, M.: *Počítačové vidění - Segmentace obrazu, analýza histogramu, analýza barev, shlukování, , Brno, FIT VUT v Brně*. Fit.vutbr.cz [online]. [cit. 2011-04-19], 2007, brno, FIT VUT v Brně.

[45] Španěl, M.: *Počítačové vidění - Statistické rozpoznávání a shlukování, Brno, FIT VUT v Brně*. 2007.

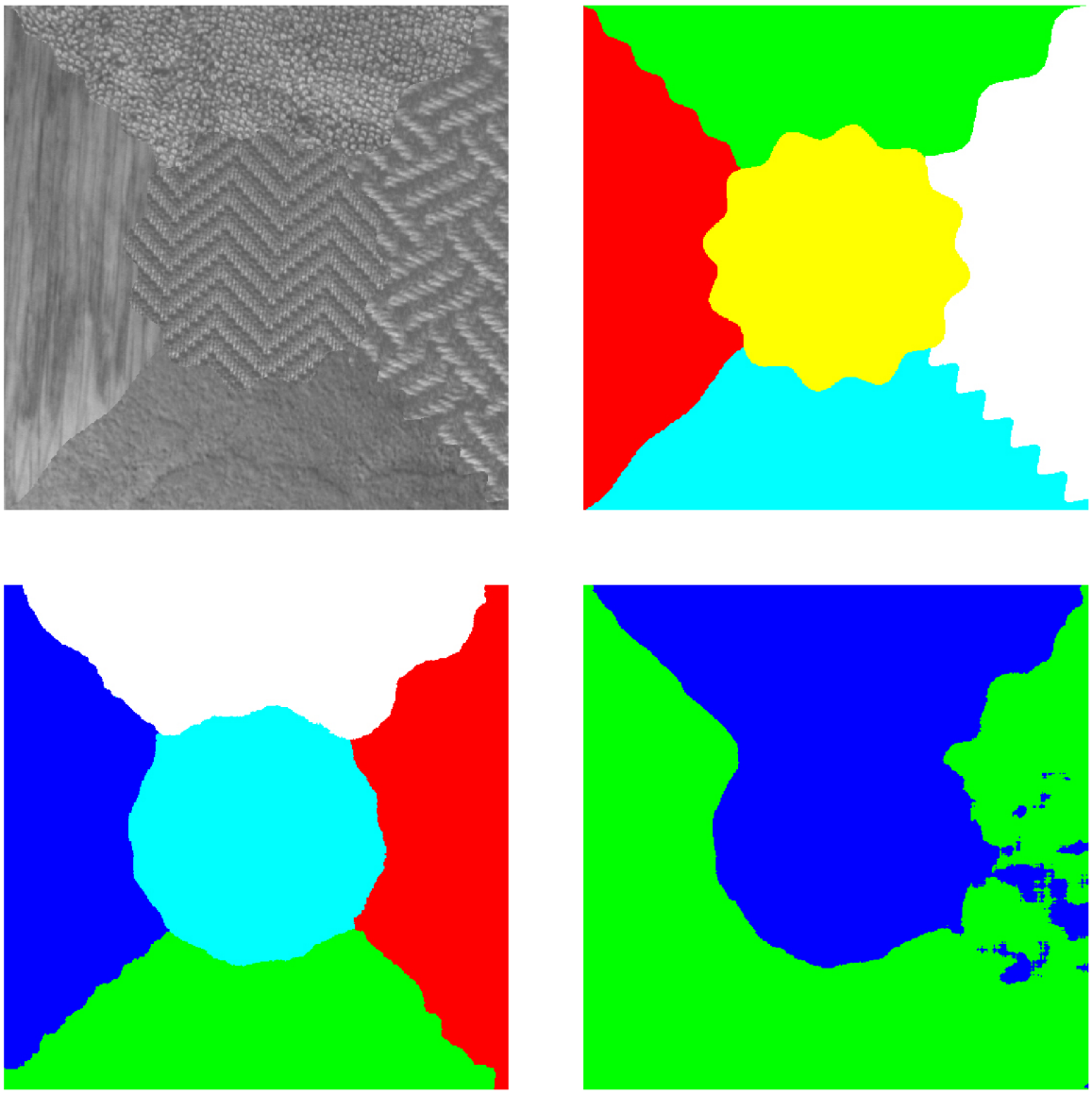
[46] Šára, R.: *Analýza textury*. Praha, FEL ČVUT, 1999, [cit. 2011-04-19].
URL
cmp.felk.cvut.cz/cmp/courses/dzo/resources/lecture_texture_sara.pdf

Příloha A

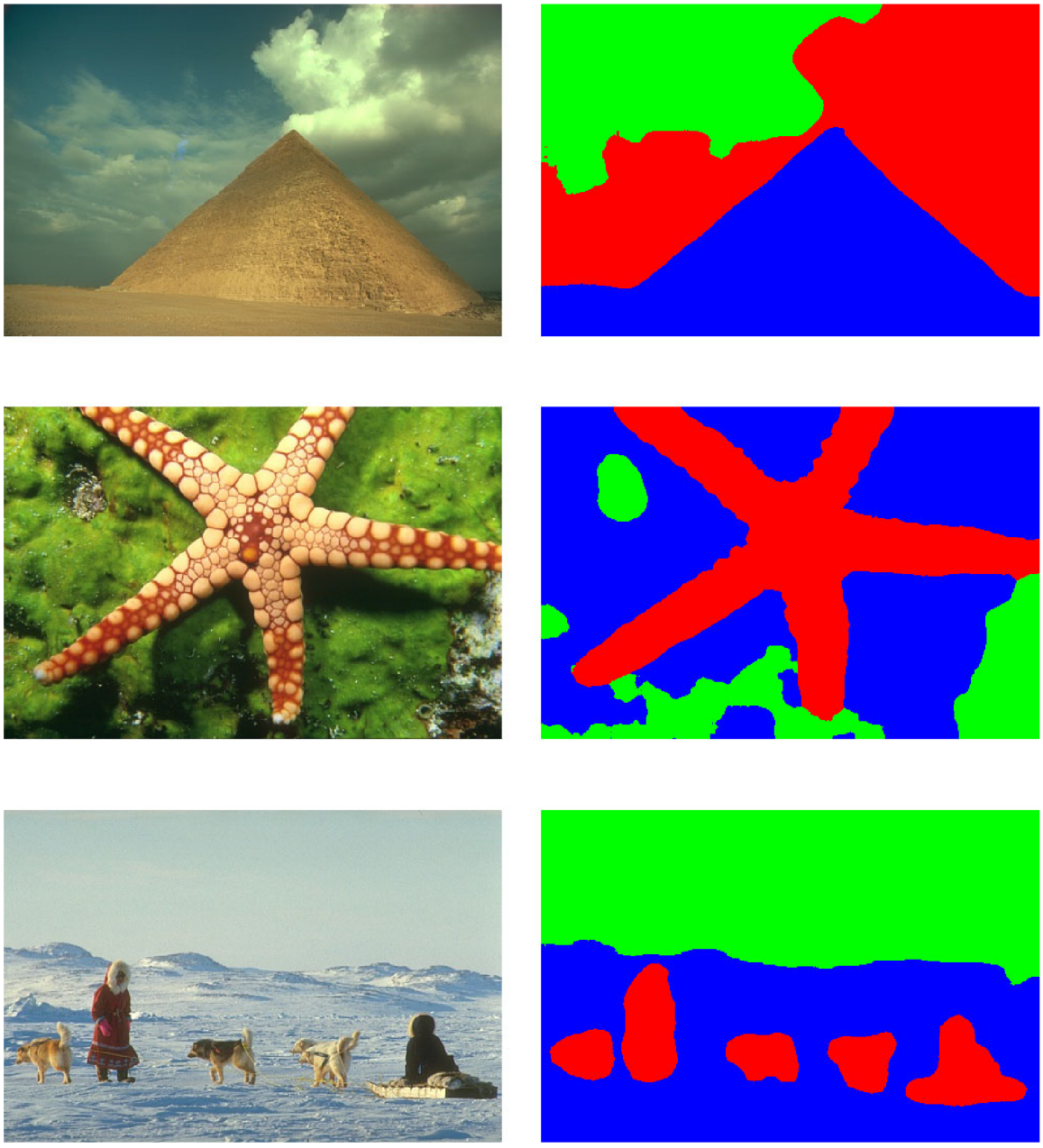
Ukázka segmentovaných obrazů



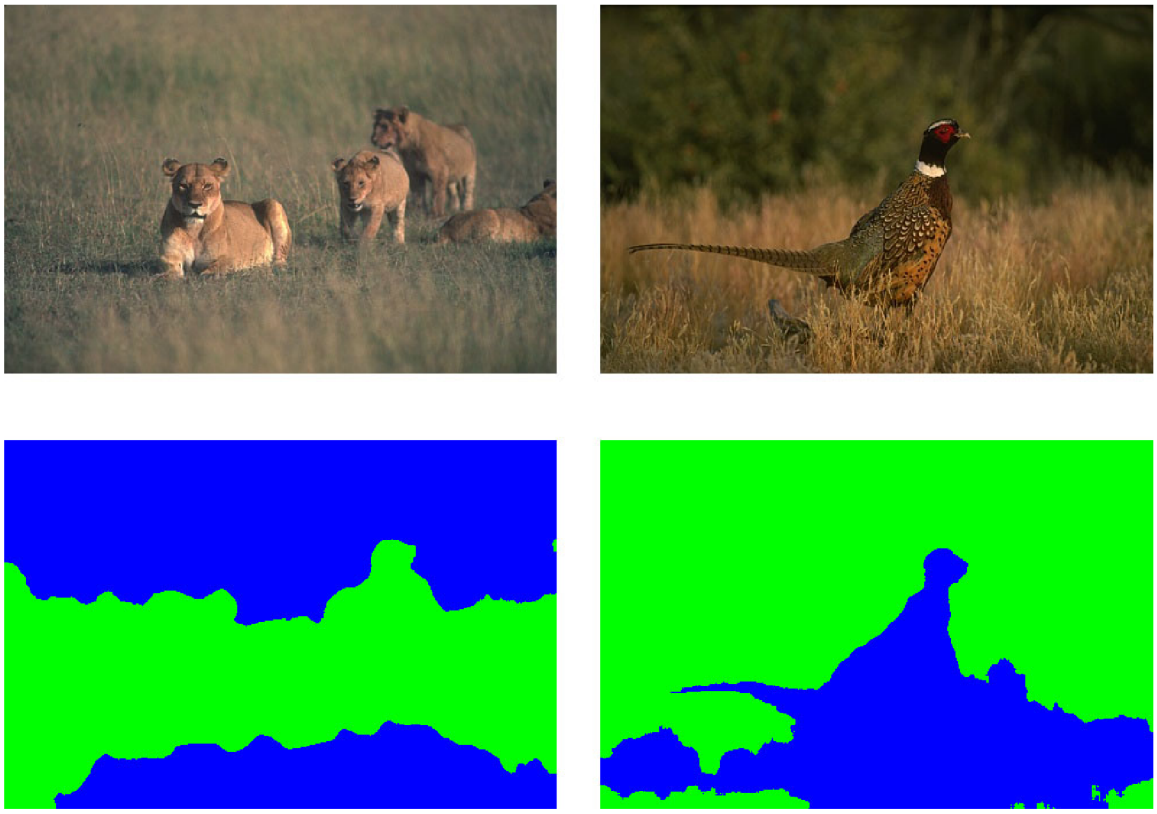
Obrázek A.1: Ukázka segmentace obrazů ze sady *Outex US 00000*. Segmentace proběhla na základě Multiresolution LBP + VAR s nastavením $P8R1|P16R3\#LBP + VAR(8)$. Shlukováním pomocí FCM používající *Normalizovanou Euklidovskou vzdálenost* a výběrem počtu shluků pomocí *Fukuyama-Sugeno* indexu.



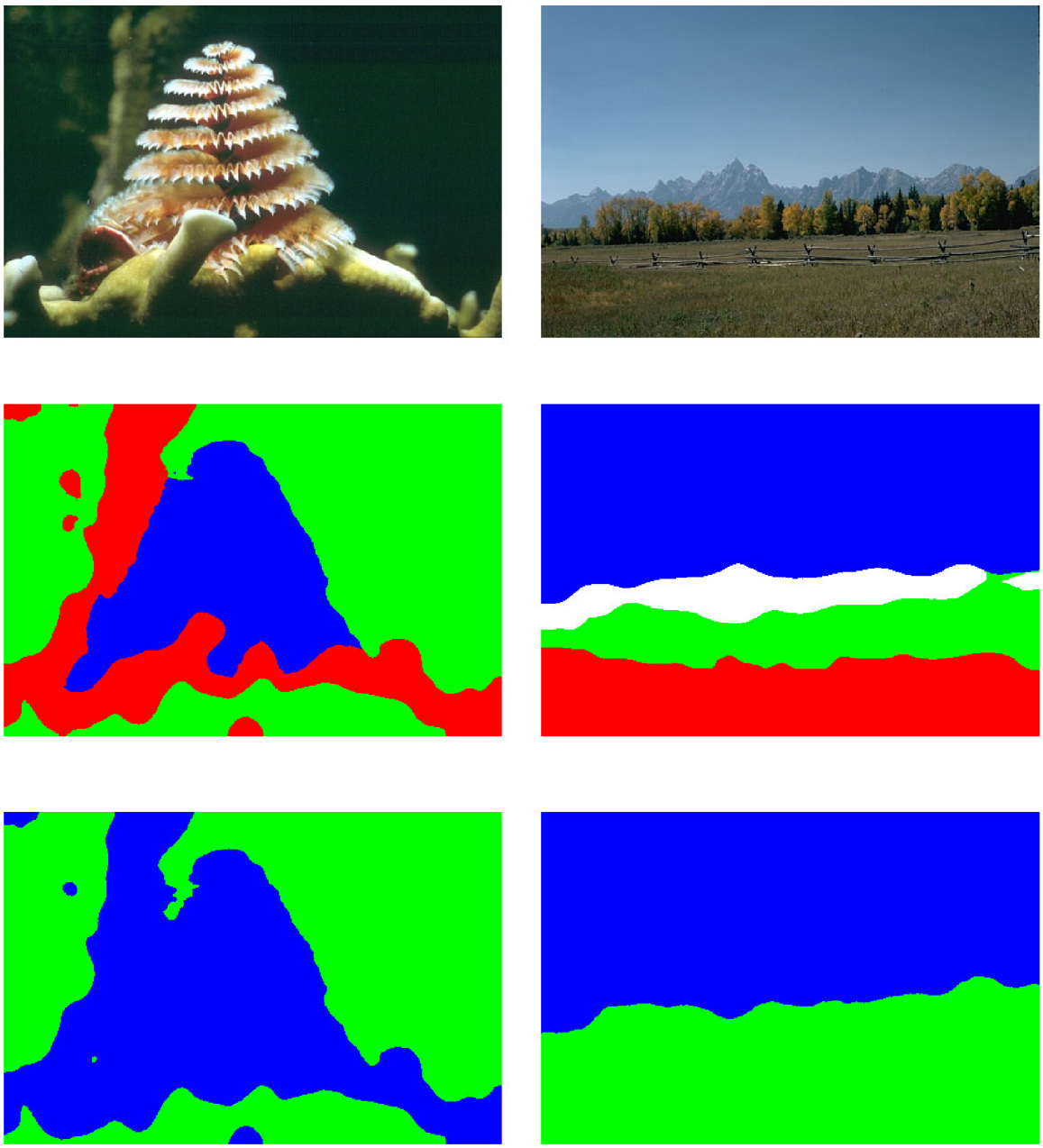
Obrázek A.2: Ukázka selhání PCAES indexu při výběru správného počtu shluků, pro obrázek ze sady *Outex US 00000*. Vlevo nahoře je vstupní obrázek a vpravo nahoře je jeho *ground truth*. Dole jsou výsledky segmentace toho obrazu s výběrem počtu shlůků pomocí Fukuyama-Sugeno(vlevo) a PCAES indexu(vpravo). Segmentace proběhla na základě Multiresolution LBP + VAR s nastavením $P8R1|P16R3\#LBP+VAR(8)$. Shlukováním pomocí FCM používající *Normalizovanou Euklidovskou vzdálenost*.



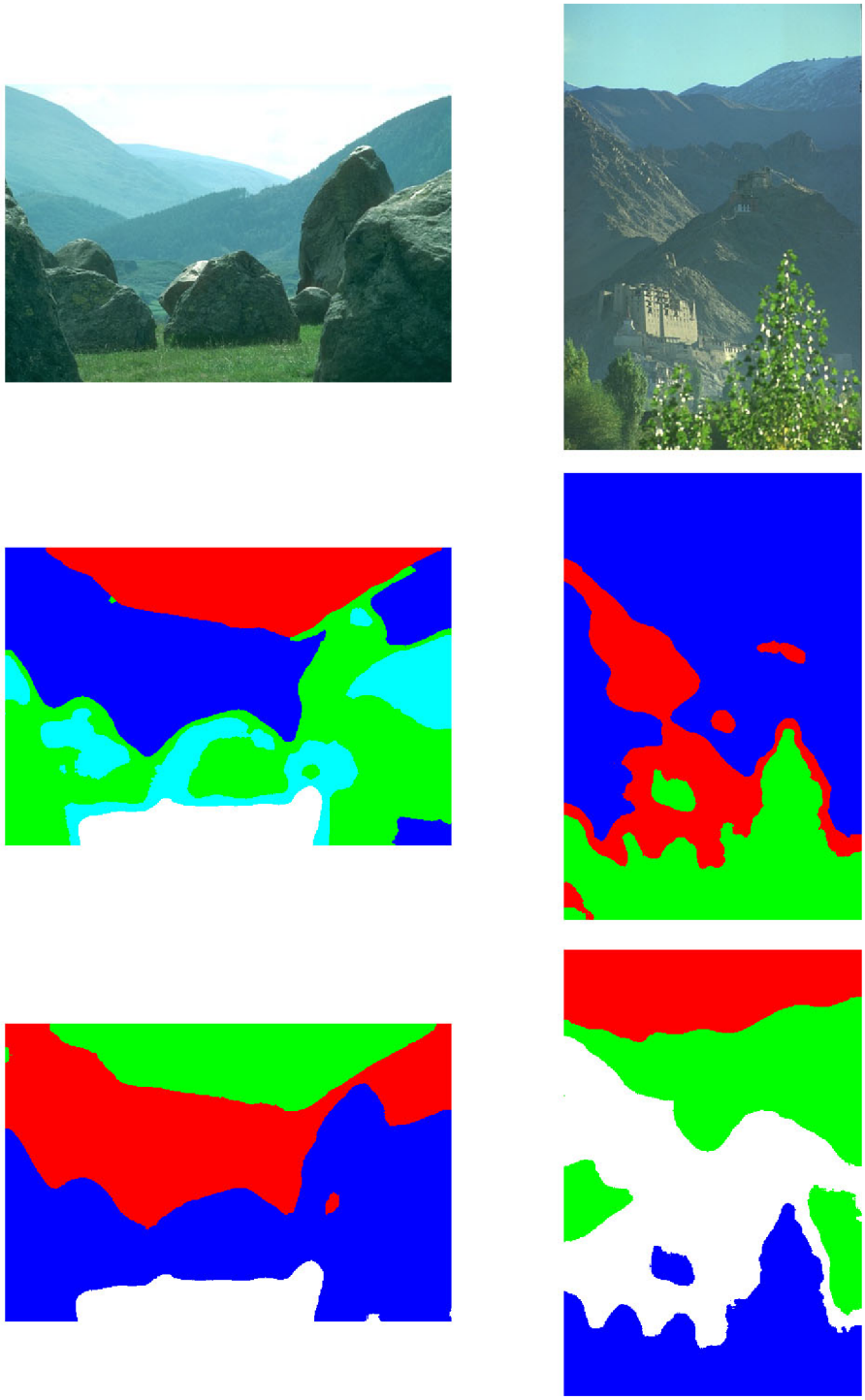
Obrázek A.3: Ukázka segmentace obrazů ze sady *BSDS300 modified*. Segmentace proběhla na základě OCLBP pro barevný prostor CIE XYZ a shlukováním pomocí FCM používající metriku vzdálenosti χ^2 s výběrem počtu shluků CWB indexem.



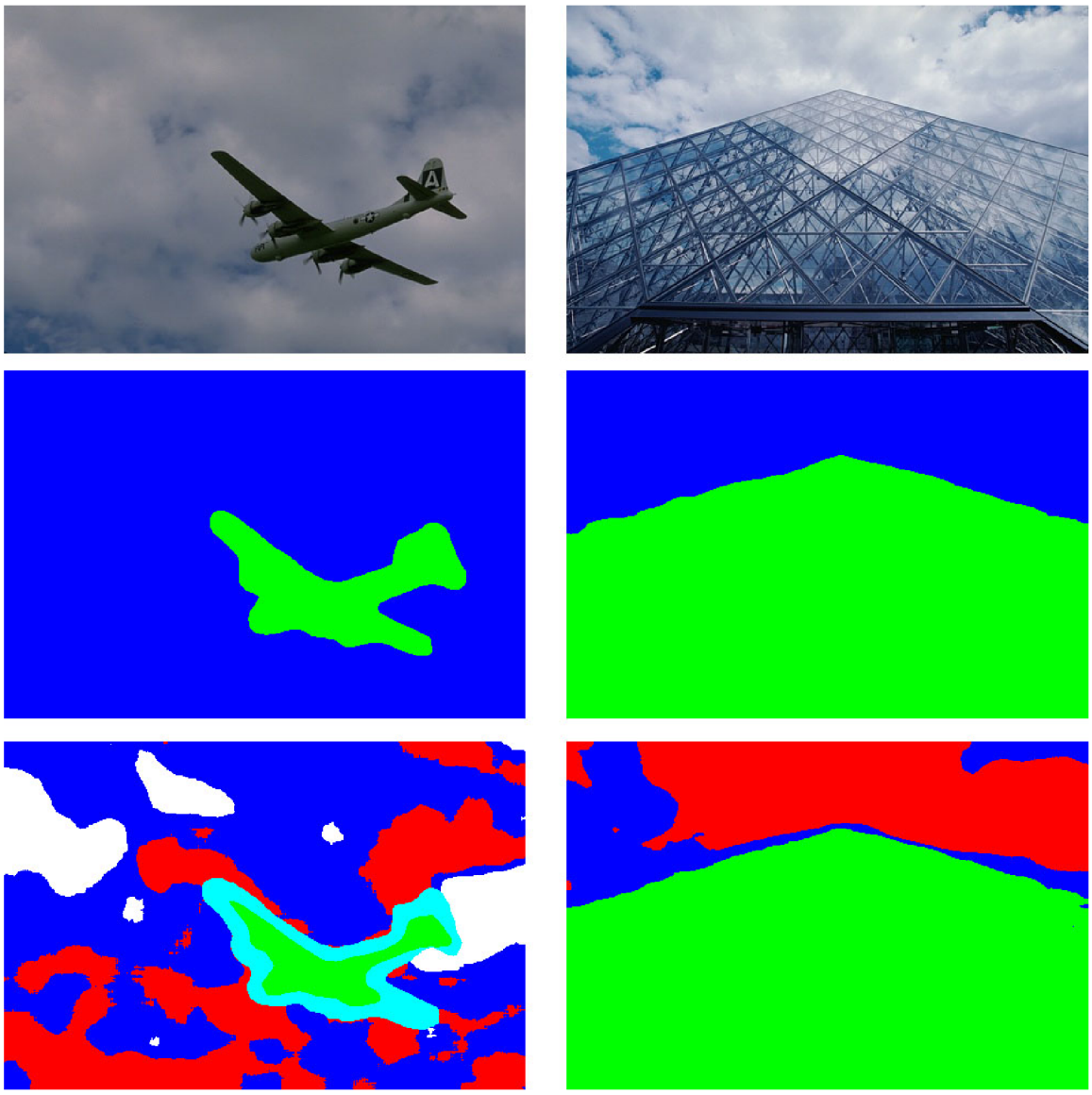
Obrázek A.4: Ukázka obrazů ze sady *BSDS300 modified*, jejichž část je rozostřená, což má za následek rozdělení obrazu na ostrou a rozostřenou část. Segmentace proběhla na základě OCLBP pro barevný prostor CIE XYZ. Shlukováním pomocí FCM používající metriku vzdálenosti χ^2 s výběrem počtu shluků CWB indexem.



Obrázek A.5: V horním řádku jsou vstupní obrazy ze sady *BSDS300 modified*. V prostředním jejich segmentované obrazy s manuálně nastaveným počtem regionů. Ve spodním je potom segmentovaný obraz, při kterém se chybně zvolí optimální počet regionů. Segmentace proběhla na základě OCLBP pro barevný prostor CIE XYZ. Shlukováním pomocí FCM používající metriku vzdálenosti χ^2 s výběrem počtu shluků CWB indexem.



Obrázek A.6: Ukázka úspěšné normalizace osvětlení. V horním řádku jsou vstupní obrazy ze sady *BSDS300 modified*, pod nimi jejich segmentace bez normalizace osvětlení a s ní (spodní řádek). histogramu. Normalizace osvětlení pomocí proběhla gamma korekcí a equalizací histogramu. Segmentace potom na základě OCLBP pro barevný prostor CIE XYZ. Shlukováním pomocí FCM používající metriku vzdálenosti χ^2 s výběrem počtu shluků CWB indexem.



Obrázek A.7: Ukázka neúspěšné normalizace osvětlení, která vede k nesprávnému přesegmentování obrazu. V horním řádku jsou vstupní obrazy ze sady *BSDS300 modified*, pod nimi jejich segmentace bez normalizace osvětlení a s ní (spodní řádek). Normalizace osvětlení byla realizována gamma korekcí a equalizací histogramu. Segmentace potom na základě OCLBP pro barevný prostor CIE XYZ. Shlukováním pomocí FCM používající metriku vzdálenosti χ^2 s výběrem počtu shluků CWB indexem.

Příloha B

Ovládání programu

Segmentační program je vytvořen jako konzolová aplikace, aby bylo možné využít ho v rozsáhlejších systémech pomocí skriptů. Protože je však jeho nastavení poměrně rozsáhlé, neprovádí se pomocí parametrů při spuštění, ale načítá se z konfiguračního souboru.

První parametr programu při spuštění slouží k zadání jména vstupního souboru, včetně cesty k němu. Tento parametr je povinný a program bez něj ihned s chybou skončí.

Druhý parametr obsahuje jméno souboru obsahujícího *ground truth*, pomocí kterého se vypočte úspěšnost segmentace. Pokud chceme k testovanému souboru použít více než jedno *ground truth*, uvedeme do tohoto parametru jejich seznam oddělený znakem *;* .Druhý parametr je nepovinný a při jeho zadání bude výstupem programu pouze vypočítaná úspěšnost segmentace. Pokud však tento parametr nepoužijeme, program namísto úspěšnosti segmentace uloží do souboru přímo segmentovaný obraz. Uložen bude do podadresáře *results* pod jménem, které odvodíme ze jména vstupního obrazu. Odvození proběhne přidáním koncovky *_out* za jméno souboru, bez uvedení cesty. To znamená že při vstupním souboru *folder1/folder2/image.png* vznikne soubor *results/image_out.png*.

Třetím a posledním parametrem může být přepínač *-s*. Pokud je tento přepínač uveden, bude výsledkem programu jak vypočítaná úspěšnost, tak i výstupní soubor se segmentovaným obrazem, který se vytvoří stejně jako v předchozím případě.

Jestliže se binární soubor jmenuje *seg_texture.exe*, můžeme spustit segmentační program pod *OS Windows* například následovně:

```
seg_texture.exe dataset/image23.jpg dataset/ground_truth/image23.seg -s
```

Takto spuštěný program vypíše na standardní výstup vypočtenou úspěšnost segmentace a segmentovaný obraz uloží do souboru *results/image23_out.png*.

Abychom mohli vypočítat i úspěšnost segmentace pro celou databázi, je součástí frameworku skript *image_set.py*. Tento skript potřebuje ke svému spuštění dva parametry. Prvním je jmého souboru segmentačního programu, který se bude spouštět. V našem případě *seg_texture.exe*. Druhým parametrem je jméno textového souboru obsahujícího seznam všech obrazových souborů a jejich *ground truth* v konkrétní datové sadě. Pokud se takovýto soubor bude jmenovat *bsds_filenames.txt*, můžeme skript spouštět například následovně:

```
python image_set.py seg_texture.exe bsds_filenames.txt
```

Tento textový soubor musí mít následující strukturu. Každý jeho řádek obsahuje nejprve jméno jednoho obrazového souboru včetně cesty k němu. Za ním následuje jméno souboru obsahujícího *ground truth*. Tato dvě jména musí být oddělena alespoň jedním bílým znakem. Řádky nemající takovouto strukturu se ignorují.

B.1 Konfigurační soubor

Všechna nastavení segmentačního programu probíhají pomocí konfiguračního souboru *config.xml*. Konfigurační soubor obsahuje tři nezávislé části, jejichž pořadí je zaměnitelné. Všechna nastavení mají své defaultní hodnoty, které se použijí v případě, že je konfiguračním soubor neobsahuje. První část obsahuje celkové nastavení segmentačního programu - vstupní filtrování, předzpracování obrazu, výstupní filtrování a metriku hodnotící úspěšnost segmentace. Ukázka této části je na obrázku B.1.

```
- <segmentation>
    <input_gauss_filtr size="5"/>
    <input_smoothing size="3"/>
    <preprocessing/>
    <output_med_filtr size="16"/>
    <evaluation_metric> f-measure </evaluation_metric>
  </segmentation>
```

Obrázek B.1: Část konfiguračního souboru, která obsahuje celkové nastavení segmentačního programu.

Všech pět položek je nepovinných a jejich pořadí je zaměnitelné. Pokud chybí v konfiguračním souboru položky odpovídající vstupnímu filtrování, předzpracování obrazu a výstupnímu filtrování, není ani jedna z těchto kompoment v segmentačním programu použita. K hodnocení úspěšnosti segmentace obrazu je defaultně použita *f-measure*. Kromě ní lze použít i *adjusted_rand_index* nebo *f-measure_per_segment*.

Druhá část obsahuje nastavení extrakce texturních příznaků - velikost extrakčního okna, počet kvantovacích hladin VAR a nastavení parametrů dílčích příznakových vektorů. Všechny

```
- <extraction>
    <extraction_window_size size="48"/>
    <var_bin_count count="8"/>
  - <feature_vector>
      <p point_count="8"/>
      <r radius="1"/>
      <color_space> gray </color_space>
      <lbp/>
    </feature_vector>
  - <feature_vector>
      <p point_count="8"/>
      <r radius="1"/>
      <color_space> gray </color_space>
      <var/>
    </feature_vector>
  </extraction>
```

Obrázek B.2:

položky jsou nepovinné a jejich pořadí je opět zaměnitelné, řičemž položka *feature_vector* může být přítomná několikrát. Každá tato položka obsahuje nastavení extrakce jednoho dílčího příznakového vektoru. Výsledný příznakový vektor je pak vytvořen konkatenací všech dílčích příznakových vektorů. Každý dílčí příznakový vektor má své nastavení typu texturního příznaku (*lbp*, *clbp*, *oclbp*, *lbpv*, *var*), barevného prostoru(*rgb*, *hsv*, *hls*, *rgb*, *gray*, *luv*, *lab*, *xyz*, *ycrcb*) a parametrů P a R.

Třetí část se zabývá nastavením shlukovacích algoritmů. Nejprve je druh shlukovacího algoritmu (*fcm*, *kmeans*, *kmeansplusplus*) a na něm závisí všechno ostatní.

```
- <clustering>
  - <fcm>
      <distance_metric> chi_2 </distance_metric>
    - <defuzzification>
        <rotation_mask/>
      </defuzzification>
    - <unknown_cluster_count>
        <cluster_validity_index> fukuyama </cluster_validity_index>
        <subsampling factor="2"/>
        <max_and_min_cluster_count min="2" max="10"/>
        <init_type> all_cluster_count </init_type>
      </unknown_cluster_count>
    </fcm>
  </clustering>
```

Obrázek B.3:

Pokud je tento druh *fcm*, následují položky udávající použitou metriku vzdálenosti (*euclidean*, *normalized_euclidean*, *hellinger*, *bhattacharyya*, *chi_2*), typ defuzzifikace (*highest_one*, *rotation_mask*, *distribution_maps-best_one*, *distribution_maps-best_two*, *distribution_maps-all*) a zda bude FCM pracovat se známým nebo neznámým počtem shluků (*known_cluster_count*, *unknown_cluster_count*). V případě neznámého počtu shluků lze pak ještě nastavit *cluster validity index* (*fukuyama*, *xie*, *pcaes*, *cwb*), faktor subsamplování, minimální a maximální počet shluků a typ inicializace (*all_cluster_count*, *gafcm*, *safcm*).

Pokud je jako druh shlukování zvolen *kmeans* nebo *kmeansplusplus* nastavuje se již pouze požadovaný počet shluků.

```
- <kmeans>
    <cluster count="5"/>
  </kmeans>
```

Obrázek B.4:

Příloha C

Plakát