VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Metody segmentace obrazu založené na teorii grafů

Graph-Based Image Segmentation Methods

2012 Petr Baroš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Petr Baroš**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Metody segmentace obrazu založené na teorii grafů
Graph-Based Image Segmentation Methods

Zásady pro vypracování:

Segmentace obrazu je významnou úlohou na cestě k analýze jeho obsahu. Hledání účinných metod segmentace je stále středem zájmu. Cílem diplomové práce je ověřit vlastnosti metod segmentace založených na teorii grafů. V diplomové práci proveďte následující:

1. Prostudujte metodu navrženou Felzenszwalbem a Huttenlocherem v článku "Efficient graph-based image segmention.
2. Metodu naimplementujte; implementaci proveďte v jazyce C++.
3. Výsledky segmentace porovnejte s výsledky jiných metod (např. meanshift).
4. Dosažené výsledky podrobně popište v textové části práce.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Dr. Ing. Eduard Sojka**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

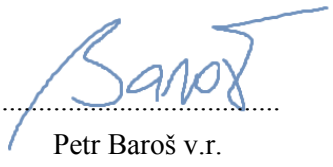
prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

30.4.2012

.....

Datum



.....

Petr Baroš v.r.

Tímto bych rád poděkoval vedoucímu své diplomové práce, doc. Dr. Ing. Eduardu Sojkovi, za odborné vedení a velmi podnětné rady, které mi v průběhu práce uděloval.

Abstrakt

Cílem diplomové práce bylo nastudovat metody pro segmentaci obrazu založené na teorii grafů, vybrat a implementovat jednu metodu, na závěr provést a vyhodnotit praktické testy. Text nejprve čtenáře seznamuje s problematikou segmentace obrazu a její úlohou při analýze obrazu. Následuje přehled několika algoritmů založených na teorii grafů, stručný popis jejich vlastností, vhodnost použití a zamyšlení nad odlišností oproti tradičním metodám. Ve střední části je detailněji popsán a vysvětlen jeden z grafových algoritmů a v krátké kapitole lze nahlédnout na specifika jeho implementace. V závěrečné kapitole je algoritmus podroben praktickým testům na sadě různých obrazů.

Klíčová slova

Segmentace obrazu, analýza obrazu, grafový algoritmus, dělení grafu

Abstract

The aim of this thesis was to study selected graph-based image segmentation method. The text begins with general description of image analysis and it specifies the role of image segmentation in that process. Since there are many graph-based image segmentation methods already published couple of them were selected and briefly described their thoughts. The “Efficient graph-based image segmentation” method was chosen for deep learning. Method was explained in detail and algorithm was implemented in C++. In the last chapter are commented several results of image segmentation producing by the algorithm.

Keywords

Image segmentation, image analysis, graph algorithm, graph partitioning

Seznam použitých symbolů a zkratek

HD – High Definition – vysoké rozlišení (obrazu)

Gestalt – celkový vnímavostní tvar

OCR – Optical Character Recognition – optické rozpoznávání znaků

Pixel – Picture element – jeden bod digitálního obrazu

Obsah

1 Úvod

Segmentace obrazu je část analýzy obrazu, počítačového vidění. Jejím úkolem je v obraze vyhledat oblasti, které tvoří jeden celek. Byť to na první pohled nevypadá, je tento úkol nelehký a někdy téměř nemožný. Následující text popisuje principy segmentace, její úlohu, vlastnosti a kvality jednotlivých metod. Jednou z možností řešení problému segmentace je využití teorie grafů. V krátkém přehledu několika metod, jež jako základ používají grafové algoritmy, se seznámíme s jejich principy a charakteristickými rysy.

Jako reprezentanta z široké škály grafových algoritmů jsme vybrali metodu Efektivní segmentace obrazu založená na teorii grafů od Pedra F. Felzenszwalba a Daniela P. Huttenlochera [1]. Jejich práci detailněji popíšeme ve střední části, ukážeme si použitý matematický základ a dokážeme, že lze s tímto algoritmem dosáhnout požadovaných výsledků. V implementačních poznámkách se zmíníme o některých specifických rysech a nápadech, kterými se podařilo algoritmus obohatit.

Na závěr jsem algoritmus podrobil testům na sadě obrazů, z jejichž výsledků vyplývá, že použití grafových algoritmů k řešení segmentace dává dobré výsledky.

2 Segmentace obrazu

V této kapitole se uvedeme do problematiky, vysvětlíme si, co obnáší analýza obrazu a jakou roli zde hraje segmentace. Popíšeme si několik druhů metod segmentace, jejich specifika a vhodnost použití.

2.1 Zpracování obrazu

Zpracování obrazu při počítačovém vidění bychom mohli rozdělit na dva základní obory, z nichž každý se skládá z několika obecných kroků:

- Získání obrazu
  - Snímání obrazu
  - Digitalizace obrazu pro počítačové zpracování
- Analýza obrazu
  - Předzpracování obrazu – filtrace
  - Segmentace obrazu
  - Zkoumání obrazu

Pro získání obrazu se dnes běžně používá několika zařízení, z nichž nejběžnější jsou fotoaparáty, kamery, scannery, či infračervené a ultrazvukové sondy. Z posledních uvedených je zřejmé, že obraz nemusí být nutně viditelný, tak jak jej vnímá lidské oko, ale může být pořízen v neviditelném frekvenčním pásmu. Digitalizace obrazu může probíhat již ve snímacím zařízení, např. digitální fotoaparát, nebo také pomocí SW v počítači, např. televizní kamera společně s televizní kartou, která analogový signál převádí v reálném čase na digitální.

2.2 Analýza obrazu

Analýza obrazu je důležitou součástí počítačového vidění neboli zpracování obrazu výpočetní technikou. Vstupem pro analýzu je obrazový signál, dvourozměrný nebo třírozměrný, statický, či pohyblivý. Výstup je různý a vždy záleží na zadání a účelu aplikace, pro kterou se analýza provádí. Jelikož se tento obor dramaticky vyvíjí, uvádím několik příkladů, které mohou být v dnešní době běžné a zajímavé.

Asi nejznámější úlohou je rozpoznávání textu, známého rovněž pod anglickou zkratkou jako OCR „Optical Character Recognition“. Vstupem pro analýzu obrazu je dvourozměrný statický obrázek – v libovolném formátu – tištěného, nebo rukou psaného textu. Nejběžnějším výstupem je pak text v požadovaném formátu a kódování. Již u tohoto jednoduchého příkladu lze ukázat různorodost výstupu. Např. na vstupu je několik obrázků tištěné dokumentace, výstupem může být soubor Open Office, formátovaný text, včetně tabulek a obrázků. Na vstupu je obrázek jedné

strany tištěného nevyplněného formuláře, požadovaný výstup pak může být formátovaná tabulka pro MS Excel. Na vstupu je ručně psaný dopis, výstupem je prostý textový soubor. Jako zajímavost můžeme uvést (již existující) aplikaci pro mobilní telefon, kde uživatel zaměří kameru telefonu na libovolný text (např. na velkou reklamu na fasádě domu či lépe informační tabulku na nádraží, v metru), aplikace vyhledá v obraze veškerý text a ten pak přeloží do předem zvoleného jazyka.

Analýza obrazu se také využívá v zabezpečovací technice. Dnes zcela běžné kamerové systémy zaznamenávají dění ve střeženém objektu. Aby pořizované záznamy nedosahovaly závratných velikostí, ukládají se na média pouze takové časové úseky, ve kterých je v obraze indikován nějaký pohyb. Vstupem pro analýzu obrazu jsou dva (nebo několik) obrazů z kamery v časové souslednosti, výstupem je pak jednoduchý ano / ne povel pro záznamová zařízení. Analýza může být jednoduchá, kdy se porovnává každý pixel obrazů, vhodná do uzavřených prostor, i složitější, kdy pohyblivý objekt musí mít určitou velikost, rychlost ap., aby se eliminovaly např. pohyby stromů ve větru, či pohyb stínu mraku při polojasném počasí. V případě, že systém obsahuje několik kamer zapojených v logickém celku, je možné ano / ne povely z jednotlivých kamer kombinovat a spouštět záznamy na kamerách, na kterých pohyb nebyl dosud detekován.

Zajímavou a určitě užitečnou úlohou je detekce volných parkovacích míst na rozlehlých parkovištích a v parkovacích domech. Jedna kamera, či vhodně zvolená soustava kamer nepřetržitě monitoruje celé parkoviště. Vstupem pro analýzu je jeden pořízený obraz, resp. kolekce obrazů soustavy kamer v jednom časovém okamžiku. Výstupem analýzy může být informace, které parkovací místo je volné a které obsazené. Prezentace výsledků analýzy již závisí na formě zadání.

Jako vstup pro analýzu obrazu ovšem nemusí být pouze obrazový signál pořízený kamerou. Ve zdravotnictví jsou dnes běžná zařízení, snímající části lidského těla pomocí elektromagnetických vln v pásmu 2,5 – 10 MHz tzv. ultrazvuku. Výstupem ze sondy je hloubková mapa. Asi nejznámějším použitím přístroje je vyšetření v průběhu těhotenství. Analýzou hloubkové mapy dokáže přístroj pomoci lékaři s nalezením plodu, či určením vyvíjejících se orgánů. Moderní přístroje umějí analyzovat vstup ze sondy tak, že lékaře upozorňují na možné vrozené vady plodu, apod.

Dnešní herní konzoly mají třírozměrné snímače pohybu hráčů. Senzor kombinuje několik vstupů, dvě kamery umístěné vodorovně a snímající prostor z různých úhlů a jeden infračervený pro hloubkovou mapu. Kombinací se získává třírozměrný obraz. Analýzou obrazu jsou detekováni jednotliví hráči, resp. jejich kostry neboli pozice hlavních kloubů v třírozměrném prostoru. Tyto senzory dokážou v reálném čase snímat pozice až 20 hráčů a dávají nový rozměr a požitek při hraní počítačových her.

Využití analýzy obrazu v běžných aplikacích je natolik rozmanité a postihuje téměř všechny oblasti lidské činnosti, že je nemožné uvést všechny. Každá analýza je specifická, dle řešeného problému a požadavku na výstup, že použité prostředky a algoritmy k jejímu zvládnutí jsou jedinečné. Přesto lze ve všech najít podobné rysy.

Při zpracování textu metodou OCR je vhodné rozdělit obrázek nejprve na sloupce, pak na odstavce, na řádky, na slova a nakonec na jednotlivá písmena. Při hledání pohybu, či auta na parkovišti bývají objekty a automobily odděleny od pozadí. A pro analýzu třírozměrného obrazu je nezbytné nejprve lokalizovat plod, či hráče v obrazovém signálu. Obecně lze říci, že v obraze rozeznáváme objekty zájmu od jiných objektů, či od nějakého obecného pozadí, takové činnosti nazýváme segmentace obrazu.

2.3 Segmentace obrazu

Segmentace obrazu je obecně chápána jako proces dělení obrazu na části, které jsou společné pro konkrétní objekty v obraze. Lze také říci, že každému pixelu obrazu je přiřazen index, který vyjadřuje příslušnost k nějakému objektu (či jeho části), neboli segmentu.

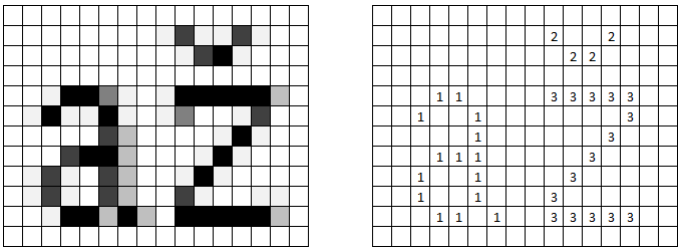
Definice 2.1 [10]:
Segmentace obrazu daného funkcí $f(x, y)$ je jeho dělení na podobrazy $R_1, R_2, \dots, R_n$ tak, že podobrazy splňují následující kritéria:

1. $\bigcup_{i=1}^{n} R_i = f(x, y)$
2. $R_i \cap R_j = \emptyset, i \neq j$
3. Každý podobraz splňuje nějaké tvrzení, popř. množinu tvrzení, např.:
   - Všechny pixely v podobraze R_i mají stejnou úroveň šedi.
   - Všechny pixely v podobraze R_i se neliší v úrovni šedi více než o předepsanou hodnotu.
   - Standardní odchylka úrovně šedi všech pixelů v podobraze R_i je dostatečně malá.
   - A podobně.

2.3.2 Objekty a pozadí

Jednoduché objekty jako kruh, obdélník, stěna domu a okno, obloha a na ní měsíc, i složitější objekty jako strom, automobil, či lidská postava mohou být předmětem zájmu při analýze obrazu.

Při zpracovávání tištěného textu metodou OCR by se daly dílčí kroky segmentace rozdělit například takto. Nejdříve jsou z obrazu ořízнuty okraje. Obraz se rozdělí na jednotlivé řádky, přičemž řádky jsou řazeny do odstavců podle velikosti mezer mezi řádky, či odsazením prvního řádku, kratším posledním řádkem v odstavci, podle velikosti písma (nadpisy), a podobně. Každý řádek je pak rozdělen na slova a slova na znaky. Segmentací obrazu pro rozpoznání znaků se každému pixelu přiřadí index, např. 0 pro pozadí a kladné číslo pro znak.



Obrázek 1: Příklad segmentace textu

Jedním z možných přístupů pro zpracování obrazu parkoviště, je obraz rozdělit na jednotlivá parkovací místa. K dělení můžeme volit z několika metod, či použít jejich kombinací, například detekce bílých dělících čár, nebo statické namapování parkovacího plánu na obraz ze stacionární kamery (předem víme, ve kterých pixelech obraz dělit), nebo výpočet vzdáleností podle záběru pohyblivé kamery a znalosti některých záchytných bodů. Výsledkem segmentace jsou opět malé obrázky každého parkovacího místa, které mohou být, pro další zpracování, transformovány na jednotný rozměr a jednotný úhel pohledu.

Obrázek 2: Ilustrativní obrázek parkoviště

2.3.3 Nástrahy segmentace

Analýza obrazu, potažmo segmentace nebývá obvykle jednoduchá a její úspěšnost závisí na několika faktorech. Některé lze eliminovat různými filtry, jiné nelze potlačit vůbec a tak je někdy segmentace úspěšná jen částečně.

Asi nejběžnější problém je osvětlení. Obraz parkoviště vypadá jinak, pokud je pořízen za slunečného letního dne a jinak o podzimním deštivém večeru. S použitím vhodných filtrů lze obraz částečně normalizovat, což je možné kombinovat s informacemi o aktuálních povětrnostních podmínkách. Rovněž volba umělého osvětlení hraje roli. Použití tzv. bílého světla, generující světlo v celém viditelném frekvenčním pásmu, dává předmětům pravé barvy, a kamery pořizují kvalitnější obrazy. Dalším problémem může být nevhodné umístění umělých světel a jejich intenzita, která hraje roli na vznik nežádoucích stínů.

Pro dobrou segmentaci hraje roli velikost objektu a barva, kterou má vůči pozadí. Velký jasný půlměsíc na tmavé obloze má konstantní žlutou barvu a velký kontrast vůči pozadí a tak je dobře rozeznatelný i výpočetní technikou. Na rozdíl tomu pokud objekt má nějaký grafický vzorek (například kostkovaná košile), či barevný přechod (osvětlená lesklá jednobarevná koule), dělá to segmentaci obtížnou. Velikost a počet objektů (kamenitá pláž) v obraze způsobí, že je obraz segmentací rozdělen na mnoho malinkých segmentů, které se obtížně rozeznávají a interpretují.

Obrázek 3: Fotografie jednobarevné koule, osvětlením vytvořený barevný přechod a silný odlesk

Jiným problémem bývá překrytí objektu zájmu. Buď je částečně schován za jiným objektem (jedoucí auto za jiným autem), či je dokonce rozdělen na části (dům je rozdělen na dvě poloviny stožárem elektrického vedení). Segmentace většinou tento problém neřeší, nebo jen částečně, a přenechává detekci na dalších algoritmech rozpoznávajících objekty v obraze.

2.4 Metody segmentace

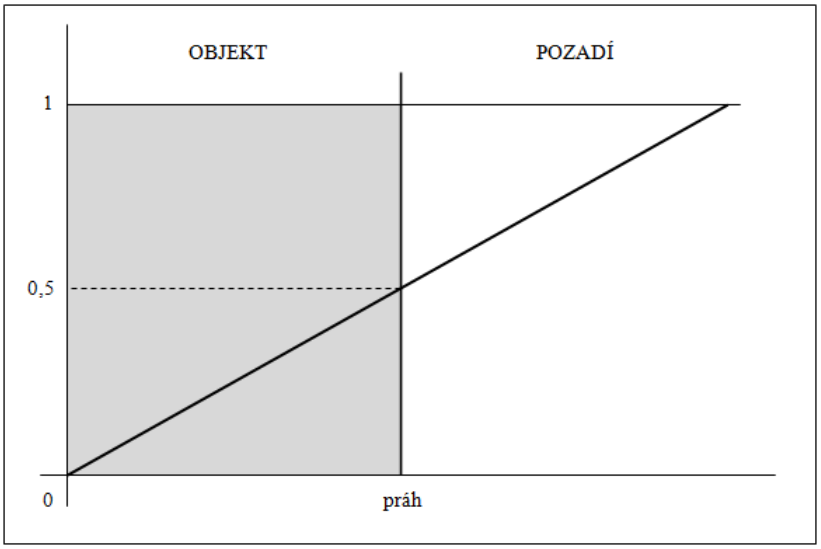
V průběhu let bylo vyvinuto a publikováno mnoho segmentačních metod, založených na různých principech. V této kapitole se velmi rychle seznámíme se základními typy algoritmů používaných pro tuto úlohu. Metody zde uvedené nejsou náplní této práce a je o nich pojednáváno jen pro ucelený pohled na problematiku segmentace.

2.4.1 Prahování

Velmi jednoduchá metoda segmentace, jejímž vstupem je většinou černobílý obraz s různým stupněm šedi, ale s vysokým kontrastem mezi pozadím a objekty. Typickým příkladem by mohl být obraz stránky textu, získaný pomocí scanneru. Papír, resp. pozadí, je bílý, či s malým stupněm šedi, a písmo, resp. objekty, jsou typicky černé, či s vysokým stupněm šedi. Ostatní odstíny se obvykle objevují na hraně mezi objektem a pozadím.

Obrázek 4: Různě šedé pixely na přechodu mezi objektem a pozadím

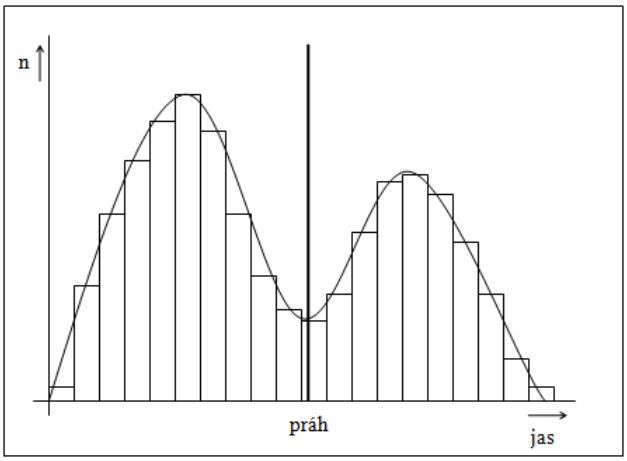
Z tohoto faktu těží metoda prahování. Stupeň šedi, či intenzitu jasu, budeme uvažovat v rozmezí hodnot 0 až 1. Parametrem metody je tzv. práh (v angličtině threshold), typicky uprostřed rozsahu tedy 0,5 (z některých důvodů může být posunut blíže k absolutní bílé – např. 0,65; či černé 0,35). Metoda jako pozadí označí každý pixel, jehož intenzita jasu je větší než práh. Jako objekt je označen naopak každý pixel, jehož intenzita jasu je menší, nebo rovna velikosti prahu.



Obrázek 5: Graf intenzity jasu a práh rozdělující pixely na objekt a pozadí

2.4.2 Metody založené na histogramu

Nevýhodou prahování je nutnost předem definovat hodnotu prahu. Metody založené na histogramu se snaží hodnotu prahu určit z obrazu. Tato metoda opět předpokládá, že v obraze existuje jednobarevné pozadí a jednobarevný objekt. Jejich barvy (resp. intenzity jasu) však nutně nemusí být bílá / černá.



Obrázek 6: Histogram intenzit jasů v obraze

Metoda nejprve určí histogram intenzit jasů, či barev. Rozsah intenzity je rozdělen na určitý počet menších podrozsahů (u digitalizovaného obrazu to bývá mocnina 2, podle velikosti obrazu a počtu pixelů, typicky 32/64). Každý pixel je dle své intenzity jasu přiřazen do patřičného podrozsahu. Každý podrozsah si pamatuje počet pixelů k němu příslušejících. Z těchto výsledků je pak sestaven sloupcový graf neboli histogram.

Pokud vrcholy sloupců proložíme křivkou, můžeme typicky pozorovat dva vrcholy (resp. dvě maxima) a mezi nimi údolí (resp. minimum), které je pak určeno jako práh. Pro vysvětlení lze říci, že jeden z vrcholů reprezentuje pixely pozadí a druhý pixely objektu. Ostatní sloupce z pravidla reprezentují pixely, jež se nacházejí na přechodu mezi objektem a pozadím.

2.4.3 Metody na bázi detekce hran

Tyto segmentační metody využívají detektorů hran. Detektory hran v obraze vyhledají různými algoritmy hrany, jichž následně využije segmentační metoda. Je zřejmé, že hrana dělí obraz na dva různé segmenty. K určení (resp. vyplnění) segmentu může být například použita tak zvaná semínková metoda, která vyplňuje oblast indexem segmentu do všech stran, dokud nenarazí na hranu. Jakmile je segment vyplněn, vyhledá se v obraze další neurčený pixel, zvýší se index segmentu a proces se opakuje do doby, než je všem pixelům přiřazen segment.

2.4.4 Metody s rostoucí oblastí

Vstupem pro tyto metody je kromě obrazu ještě řada (seznam) pixelů tzv. semínek neboli zárodků budoucích segmentů. Kvalita výsledné segmentace hodně závisí na počtu a výběru semínek. Obecný algoritmus lze popsat takto.

V nultém kroku jsou pixelům semínek přiřazeny indexy segmentů. V dalších krocích jsou kolem každého segmentu určeny nealokované pixely a rozdíly intenzit jasu δ mezi nealokovaným pixelem a průměrem intenzit jasu všech pixelů segmentu. Ke každému segmentu je přiřazen pixel s nejmenším δ. Krok se iteračně opakuje pro každý segment, dokud nejsou všechny pixely přiřazeny některému ze segmentů.

Modifikací metody, se pro δ stanovuje práh, neboli maximální hodnota, při které je ještě pixel přiřazen k segmentu. Takto mohou po ukončení základního algoritmu zůstat nepřiřazené pixely. Z nich se nově vyberou semínka dalších segmentů a algoritmus se opakuje.

2.4.5 Metody „Rozděl a spoj“

V zahraniční literatuře jsou tyto metody často označovány jako „Split-and-merge“, které budeme označovat jako „Rozděl a spoj“. Jsou definovány dvě operace:

a) rozdělení – pokud nějaký region obsahuje nesourodé pixely (např. rozdílné intenzity jasu), je rozdělen na čtyři stejné kvadranty.

b) spojení – sousedící dále nedělitelné obdélníky (resp. objekty), obsahující shodné pixely (podobné intenzity jasu), jsou sloučeny v jeden objekt (spojení nemusí bezpodmínečně vytvořit obdélník).

Metoda začíná s jedním obdélníkem představujícím celý obraz. Pak v opakujících se krocích jsou nejprve všechny obdélníky analyzovány a případně rozděleny na čtyři kvadranty. Další analýzou jsou některé z nich případně sloučeny v jeden objekt. Metoda pokračuje do té doby, dokud v jednom kroku došlo k alespoň jednomu rozdělení, či sloučení.

2.4.6 Metody založené na modelu

Metoda využívá znalosti segmentovaných objektů, jejich geometrických vlastností – tvaru, velikosti, apod. V obraze je hledán pravděpodobnostní model s přihlédnutím k možným odchylkám a deformacím. Úloha předpokládá znalost modelu a zahrnuje:

- Vytvoření průměrného jednotného modelu na základě sady trénovacích objektů.
- Vytvoření pravděpodobnosti rozdílů trénovacích objektů od průměrného modelu.
- Statistický koeficient mezi modelem a obrazem, neboli jak je pravděpodobné, že obraz představuje model.

2.4.7 Metody založené na neuronových sítích

Pro segmentaci obrazu lze také využít neuronových sítí. Jmenujme jednoho představitele neutronových sítí Pulzní spojování (v angličtině známé jako Pulsed-coupled), kde každý pixel představuje vstupní neutron a jeho intenzita jasu je počátečním vnitřním stimulem neuronu. Navzájem sousedící neurony jsou spojeny. Každý neuron dostává vnější stimuly od svých sousedů. Vnější a vnitřní stimuly se sčítají a tvoří interní aktivační systém, kdy se postupně zvyšuje stimul neuronu, až dosáhne definovaného prahu, jehož výsledkem je výstupní puls a dochází k „vybití“ neuronu. Opakující se výpočty zvyšují vnitřní stimuly a tak každý neuron produkuje sérii výstupních pulsů. Segmentace je založena na porovnávání frekvence výstupních pulsů všech neuronů.

3 Segmentace založené na teorii grafů

V této kapitole se seznámíme s metodami založenými na teorii grafů. Nejprve si nadefinujeme základní pojmy a pak si vysvětlíme princip vybraných algoritmů. Všechny metody mají společný postup. Na počátku je obraz převeden na neorientovaný ohodnocený graf, pak jsou uzly grafu spojovány do komponent či shluků dle některého kritéria a na závěr se z nalezených komponent určí segmenty obrazu. *Pozn.:* v následujícím textu se snažím popsat převážně hlavní myšlenky a nekladu důraz na podrobný a úplný rozbor uvedených algoritmů.

Definice 3.1:
Mějme neorientovaný ohodnocený graf $G = (V, E)$, kde každý uzel $v_i \in V$ koresponduje s právě jedním pixelem v obraze a každá neorientovaná ohodnocená hrana $(v_i, v_j) \in E$ spojuje dva sousední pixely. Hodnota hrany $w((v_i, v_j))$ pak vyjadřuje nějakou vlastnost spojených pixelů, jako například rozdíl intenzity jasu, barvy, pohybu, či jiných lokálních vlastností.

Definice 3.2:
Kostra grafu G je podgraf $K' = (V, E')$, kde $E' \subseteq E$, který má stejnou množinu uzlů, má stejný počet komponent, neobsahuje však kružnice.

Definice 3.3:
Minimální kostra grafu MST (anglicky Minimal Spanning Tree) je kostra grafu $K = (V, E')$ pro kterou platí, že součet ohodnocení všech hran

$$s = \sum_{e \in E'} w(e)$$

je minimální ze všech možných koster grafu G.

Definice 3.4:
Segmentace S je rozdělení množiny uzlů V do komponent, kde každá komponenta (region) $C \in V$ koresponduje s příslušnou komponentou v grafu $G' = (V, E')$, kde $E' \subseteq E$.

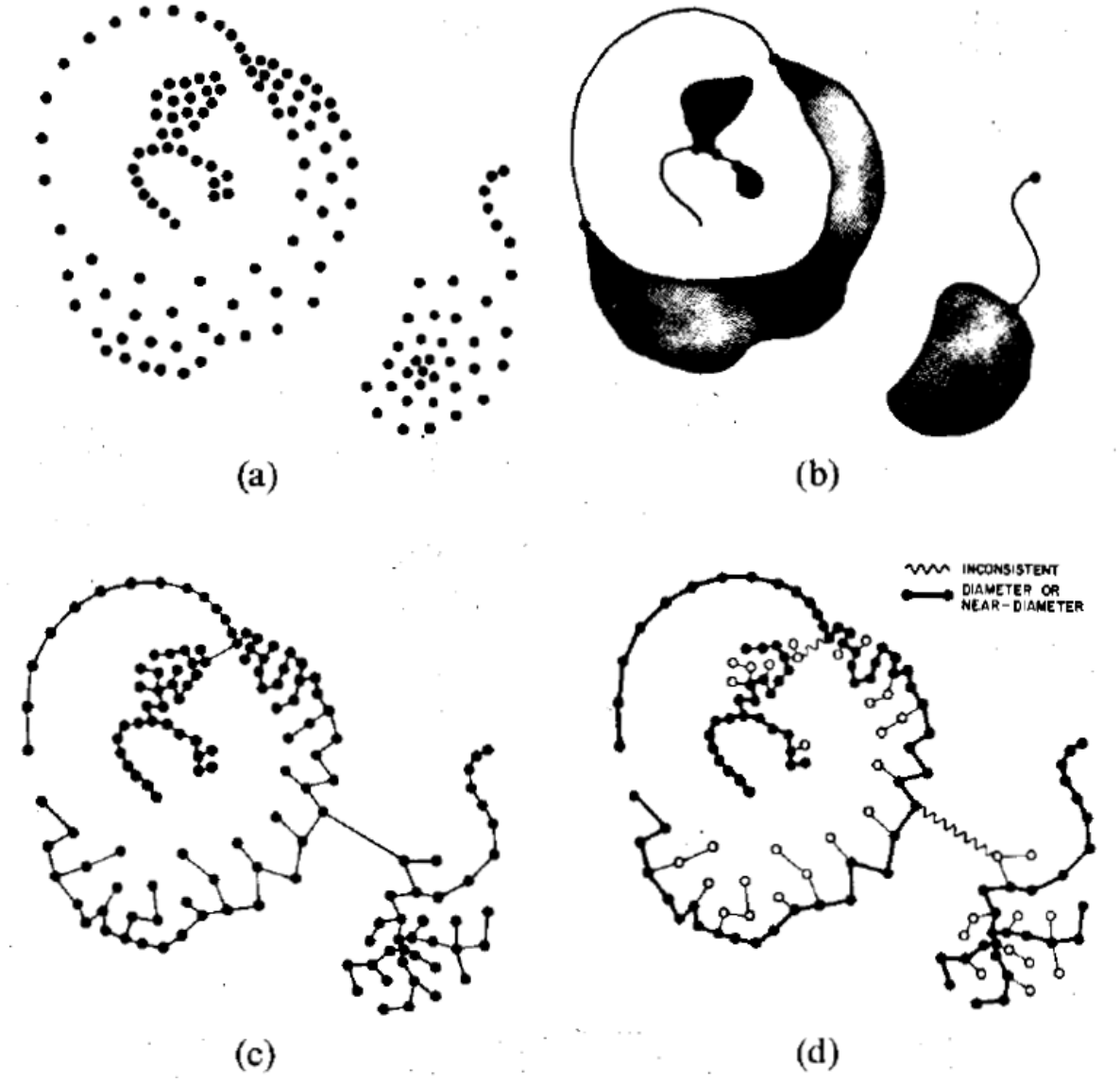
Pozn.: Jinými slovy cílem je, aby uzly v jedné komponentě si byly podobné a zároveň rozdílné od uzlů v jiných komponentách.

3.1 Metody pro detekci a popis gestalt shluků

V roce 1971 Charles T. Zahn vydal u organizace IEEE článek „Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters“ [2], kde popisuje rodinu algoritmů založených na teorii grafů, resp. na minimální kostře grafu. Algoritmy, detekující v grafu shluky, byly vyvíjeny na sadě bodů z dvourozměrného prostoru tak, aby kopírovaly lidské vnímání tvaru, či

seskupování bodů. Metody lze také aplikovat na úlohy z vícerozměrných prostorů. Přednostmi těchto metod jsou přesnost, jednoduchá interpretace výsledných shluků, shoda základních tvarů podle vnímání jejich organizace a neměnnost výsledků vzhledem k jednotné transformaci vzdáleností mezi body.

Vstupem pro všechny metody není klasický obraz (dvourozměrné pole pixelů s určitou hodnotou jasu, či barvy), ale sada bodů, resp. jejich souřadnice ve dvourozměrném prostoru. Tyto body zároveň tvoří uzly grafu. Hrany spojující uzly jsou pak ohodnoceny dle vzdálenosti uzlů v prostoru.



Obrázek 7: Množina bodů v dvourozměrném prostoru, vnímavostní gestalt a minimální kostra grafu

Základem všech metod je nalezení minimální kostry takového grafu, která představuje nejmenší vzdálenosti mezi body. V minimální kostře grafu jsou dalšími metodami hledány shluky, ze kterých se určuje vnímavostní gestalt. Jako příklad si uvedeme jednu z popisovaných metod Problém složeného shluku (v angličtině A Composite Cluster Problem). Na obrázku 7(a) je vstupní množina bodů v dvourozměrném prostoru, (b) vnímavostní gestalt, neboli jak bychom vnímali celkový obraz složený ze vstupních bodů. V našem příkladu se jedná o tři samostatné objekty. Na obrázku 7(c) je minimální kostra grafu. Obrázek 7(d) ukazuje tři druhy hran, které se v minimální kostře vyskytují. Vlnovkou jsou zobrazeny tzv. nekonzistentní hrany, které spojují různé objekty v obraze (v našem případě dvě hrany). Nekonzistentní hrany jsou algoritmem nalezeny a z minimální kostry odstraněny. Silnou čarou jsou zvýrazněny hrany, které představují obvod nebo průměr objektů. I ty jsou algoritmicky určovány. Ostatní hrany minimální kostry jsou zobrazeny tenkou černou čarou a ve výsledku jsou také odstraněny.

Další popisované metody pracují také s minimální kostrou grafu, pomocí které se snaží řešit některý problém na množině bodů z dvourozměrného prostoru. Z důvodu rozsahu práce však uvádím pouze strohý výčet jejich názvů:

- Popis dráhy, které body tvoří (anglicky Particle Track Description)
- Dotýkající se shluky (anglicky Touching Clusters)
- Dotýkající se gaussiánské shluky (anglicky Touching Gaussian Clusters)
- Detekce gradientu hustoty (anglicky Density Gradient Detection)

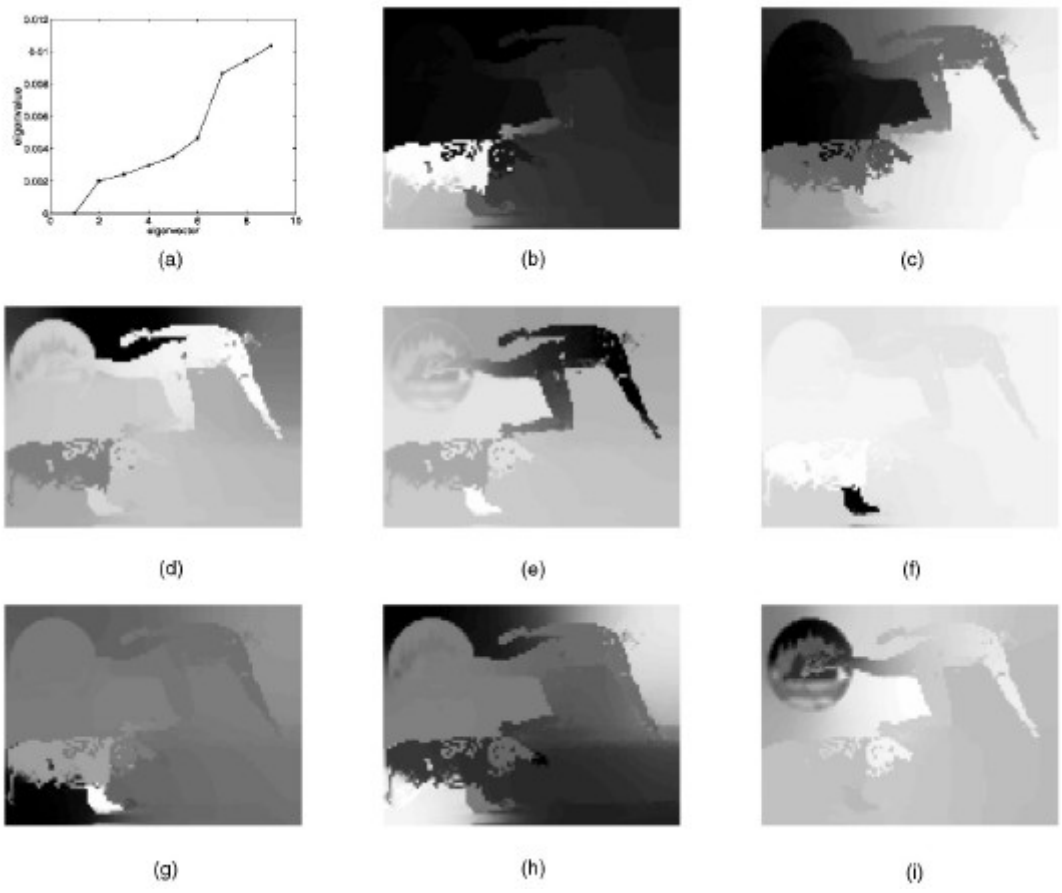
3.2 Segmentace obrazu pomocí normalizovaných řezů

V roce 2000 v oběžníku IEEE „Transactions on Pattern Analysis and Machine Intelligence“ vydali Jianbo Shi, Jitendra Malik článek “Normalized Cuts and Image Segmentation“ [3]. Popisují zde, jak sami říkají, originální přístup k řešení segmentace obrazu. Místo zaměření na lokální rysy a konzistenci obrazových dat, se snaží vytáhnout z obrazu jeho celkový dojem. Segmentaci obrazu převedli na problém dělení grafu a navrhli původní globální kritérium, *normalizovaný řez*, pro segmentaci grafu.

Obrázek 8: Fotografie dvou hráčů baseballu

Kritérium normalizovaného řezu poměřuje jednak celkovou odlišnost různých komponent, stejně dobře jako podobnost elementů uvnitř jedné komponenty. Je založen na hledání vlastních čísel a vektorů v matici sousednosti. Základní algoritmus se skládá z následujících kroků:

1. Sestavení neorientovaného ohodnoceného grafu z obrazu.
2. Řešení rovnice $(D - W)x = \lambda Dx$ pro vlastní vektory s nejmenšími vlastními čísly. Kde D je diagonální matice, na jejíž diagonále jsou součty vzdáleností $d(i) = \sum_j w(i,j)$ z uzlu v_i do všech ostatních. A W je symetrická matice sousednosti, jejíž prvky představují hodnoty hran spojující vrcholy $W(i,j) = W(j,i) = w(i,j)$.
3. Vlastní vektor s druhým nejmenším vlastním číslem se použije pro dělení grafu.
4. Kroky 2,3 se rekurzivně opakují, dokud lze některou z částí dále dělit.



Obrázek 9: Grafické zobrazení vlastních vektorů

V obrázku 9(a) je vykreslen graf nejmenších vlastních vektorů vůči hodnotám vlastních čísel, obrázky 9(b)-(i) pak zobrazují vlastní vektory korespondující s druhým nejmenším až devátým nejmenším vlastním číslem systému.

Autoři v publikaci dále uvedli další dvě modifikace základního algoritmu. První pojmenovali rekurzivní dvou-cestný normalizovaný řez, jehož vylepšení spočívá zejména v rozhodnutí, zda je možné podgraf dále dělit, a to podle kontroly stability řezu. Druhý se jmenuje simultánní k-cestný řez s násobnými vlastními vektory, jejichž popis je nad rámec této práce.

3.3 Efektivní segmentace obrazu pomocí řezu grafu

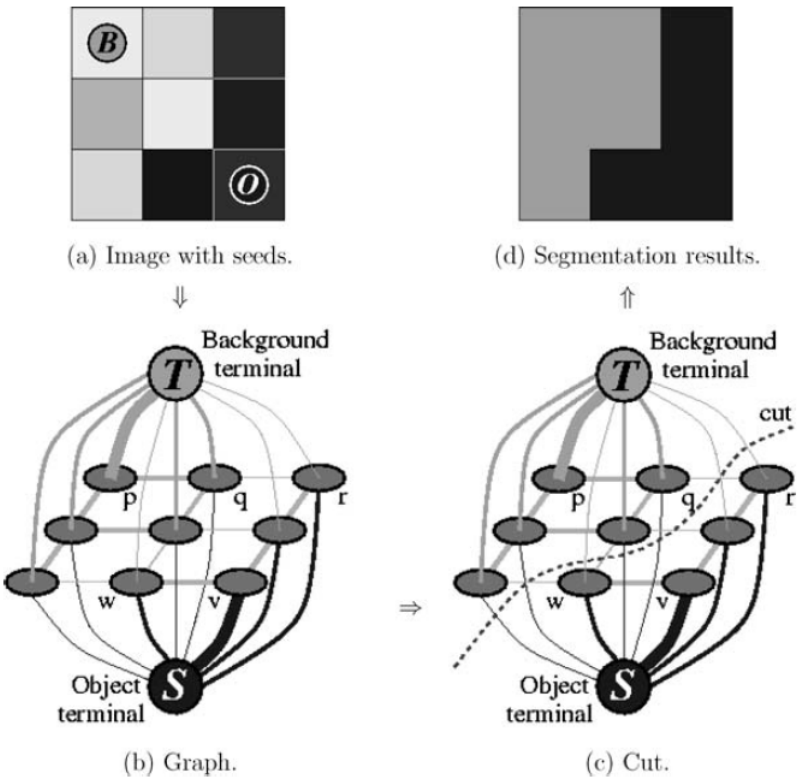
Další algoritmus založený na grafovém řezu uvedli Yuri Boykov a Gareth Funka-Lea v roce 2004 v publikaci „Graph Cuts and Efficient N-D Image Segmentation“ [4]. Výsledkem jejich práce je aplikace implementující metodu s typickými rysy kombinačních řezů grafu. Algoritmus má vlastnosti jako globální optimalizace, praktická výkonnost, numerická robustnost, schopnost zkombinovat široké spektrum vizuálních podnětů a omezení, neomezené topologické vlastnosti segmentů.

Podobně jako ostatní metody i zde každý pixel obrazu představuje jeden uzel grafu, které jsou mezi sebou spojeny hranou ohodnocenou podle podobnosti pixelů (čím více jsou si pixely podobné, tím větší ohodnocení). Vlastní metoda je založena na definování dalších dvou speciálních uzlů (obecně pólů) nazvaných S zdroj (anglicky source) reprezentující objekt a T spotřebič (anglicky sink) reprezentující pozadí. Všechny uzly (pixely) jsou spojeny s těmito dvěma speciálními póly pomocí ohodnocených hran. *Pozn.*: takto získáme v grafu dva druhy hran: n-hrany (z anglického neighbor – soused) spojující sousední pixely a t-hrany (z anglického terminal – pól) spojující pixely s póly.

V obraze jsou určité pixely označeny jako objekt, resp. pozadí, kterým se říká semínka. Jejich označení může být manuální, jako vstupní parametr segmentační metody, nebo mohou být určeny algoritmicky úvodní operací.

Obecně uvažujme jako vstupní parametr dvě semínka, kde jedno označuje objekt a druhé pozadí. Vezneme uzel (pixel, semínko) objektu a spojíme jej hranou s uzlem S, hranu ohodnotíme nekonečně velkou hodnotou. Pak vezneme uzel (pixel, semínko) pozadí a spojíme jej hranou s uzlem T, i tuto hranu ohodnotíme nekonečně velkou hodnotou. *Pozn.*: uzly „semínka“ nejsou spojeny se svými protipóly žádnou hranou.

Pak postupně spojujeme každý uzel grafu hranou s oběma póly S i T, ohodnocení patřičné hrany určujeme na základě podobnosti pixelu se „semínky“ (p je semínko pólu a q je spojovaný pixel s pólem). Typicky je hodnota hrany veliká, pokud jsou si pixely p a q podobné, na druhou stranu hodnota hrany je blízká 0, pokud jsou pixely p a q velmi odlišné. Hodnota hrany může být snížena libovolnou funkcí vzdálenosti těchto pixelů.



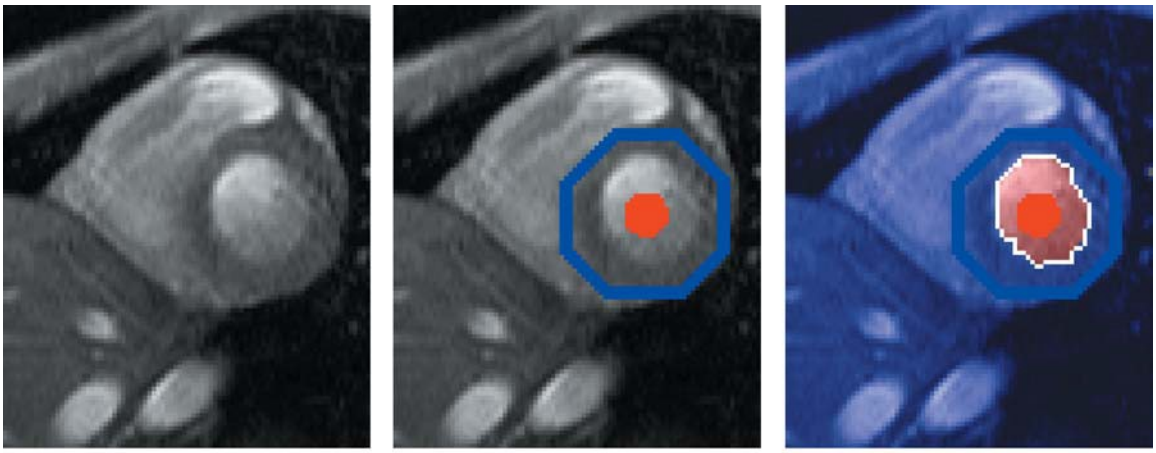
Obrázek 10: Ukázka řezu grafu

Na obrázku 10 vidíme modelový příklad, 10(a) představuje vstupní obraz s označenými „semínky“, 10(b) je generovaný graf, kde síla čáry představuje ohodnocení hrany, 10(c) je výsledný řez grafu a 10(d) výsledná segmentace.

Autoři definují tzv. s-t řez, který pro každý pixel spojený t-hranami s uzly S i T odstraní takovou t-hranu, která má menší ohodnocení (v konečném důsledku je po řezu každý pixel spojen pouze s uzlem S nebo T). Algoritmus vypočítá cenu (anglicky cost) řezu jednoduše tak, že sečte ohodnocení odstraněných t-hran. Při pohledu na obrázek 10(c) vidíme, že řez odstranil některé n-hrany. Algoritmus rovněž vypočítá cenu odstraněných n-hran sečtením jejich ohodnocení.

Je patrné, že odstraněné n-hrany jsou přesně na hranici segmentace, takže výsledná cena reprezentuje cenu hranice segmentace. Na druhou stranu odstraněné t-hrany mohou reprezentovat regionální vlastnosti samotných segmentů. A tak řez s minimální cenou může odpovídat segmentaci s vhodným vyvážením hraničních a regionálních vlastností. Cílem metody je nalézt takový řez s minimální cenou, který dává „optimální“ segmentaci. *Pozn.*: nekonečně velké ohodnocení t-hran umožňuje zavést pro segmenty tzv. pevná omezení.

V reálných obrazech většinou objekty nemají zřetelné hranice a tak je nezbytné před výpočtem více omezit prostor hledání. Proto autoři definují t pevná omezení, která určují co je pozadí a co objekt. Tato pevná omezení mohou být automaticky vypočtena, nebo manuálně nastavena.



Obrázek 11: Segmentace krevní sraženiny v srdeční komoře

Například v lékařské aplikaci je potřeba segmentovat krevní sraženinu v levé srdeční komoře obrázek 11(a). Jednoduchou metodou porovnání podle vzoru je možné lokalizovat levou srdeční komoru, např. podle jejího kruhového tvaru. Pak pevná omezení 11(b) jsou automaticky určena ihned, jak je nalezena krevní sraženina. Následně algoritmus grafových řezů určí hranice krevní sraženiny 11(c).

Pozn.: modrá barva je použita pro omezení pozadí a červená pro objekt.

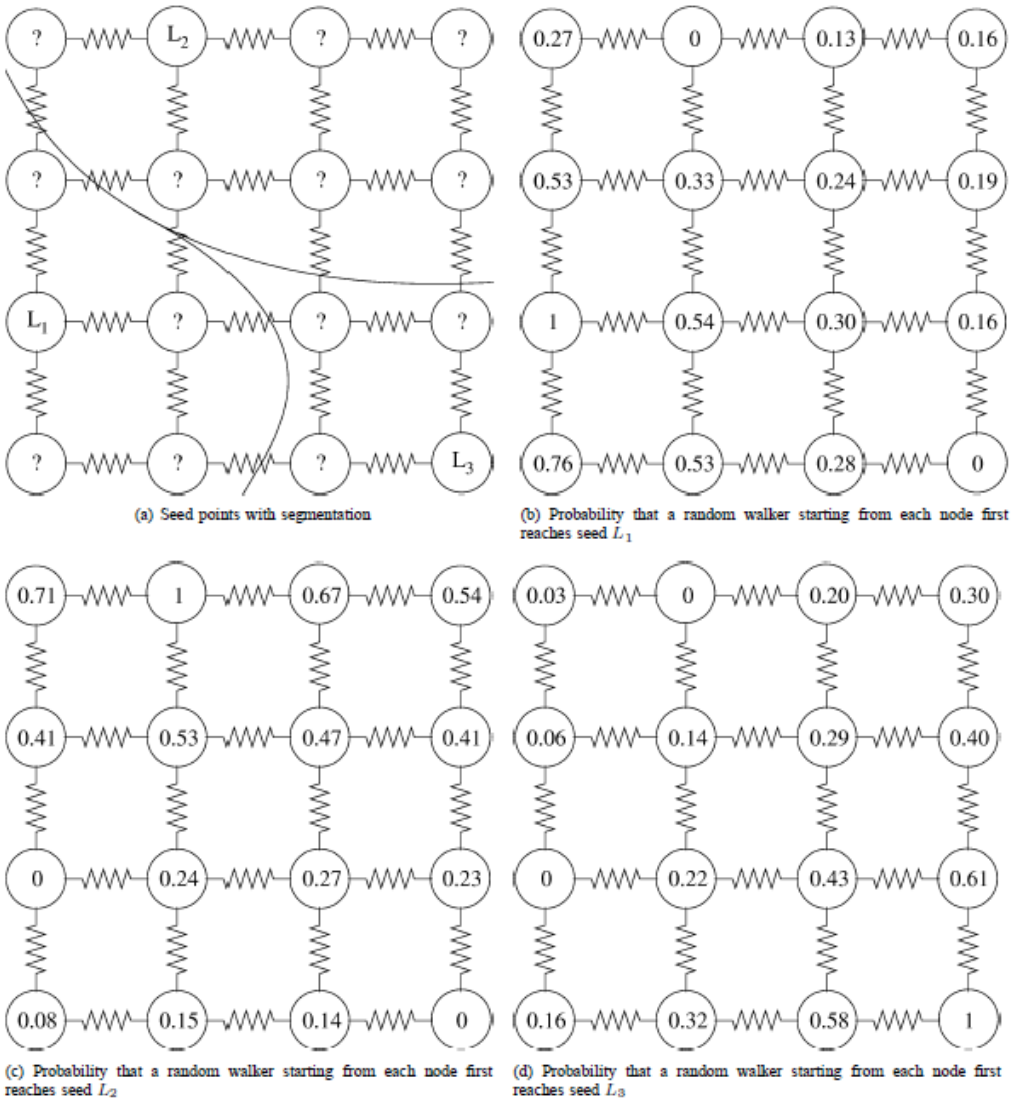
Autoři doporučují postup, kdy jsou nejprve pevná omezení nalezena automaticky, a následně mohou být manuálně přidávána, aby bylo možné výslednou segmentaci editovat.

3.4 Segmentace obrazu pomocí náhodné procházky

S poněkud originálním nápadem přišel Leo Grady v roce 2006, kdy publikoval článek „Random Walks for Image Segmentation“ [5]. Přestože se nejedná o typický grafový algoritmus, je v tomto seznamu uveden pro jeho matematický základ, vycházející právě z teorie grafů. Sousední pixely jsou mezi sebou propojeny ohodnocenými hranami a mezivýsledky představují pravděpodobnost náhodné procházky grafem, jak ukážeme níže.

Kromě samotného obrazu je dalším vstupem typicky malý počet pixelů s uživatelsky definovanými (nebo automaticky předdefinovanými) alespoň dvěma značkami. Jedna značka může označovat i více pixelů. Je vcelku snadné analyticky a rychle stanovit pravděpodobnost, s jakou náhodná procházka, začínající v neoznačeném pixelu, jako první dorazí do jednoho z pixelů označeného některou značkou.

Označením pixelů tou značkou, pro kterou byla vypočtena nejvyšší pravděpodobnost, může být dosaženo vysoké kvality segmentace. Algoritmus je formulován v diskrétním prostoru (na grafu) používající analogie standardních operátorů a principů z teorie nepřetržitého napětí a elektrických obvodů, aplikovatelný v libovolném rozměru na libovolný graf.



(a) Seed points with segmentation

(b) Probability that a random walker starting from each node first reaches seed L_1

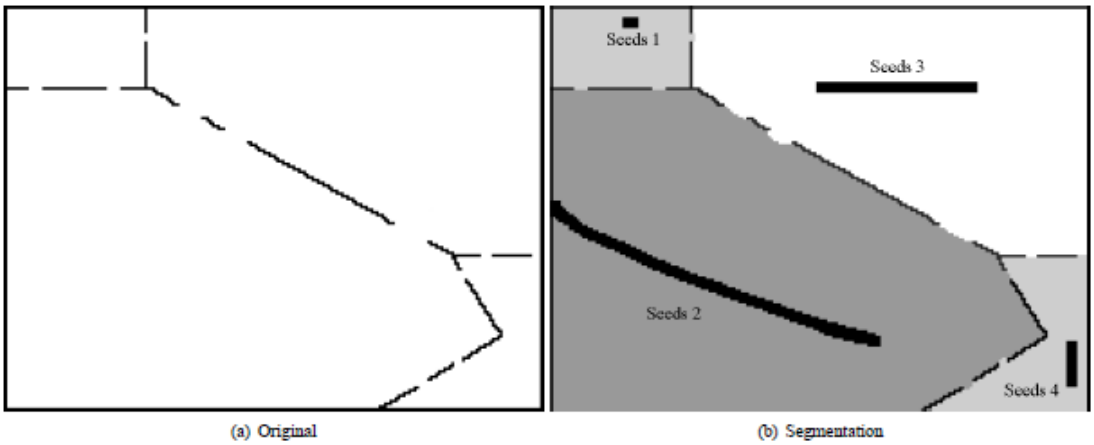
(c) Probability that a random walker starting from each node first reaches seed L_2

(d) Probability that a random walker starting from each node first reaches seed L_3

Obrázek 12: Ilustrace algoritmu

Na obrázku 12 tři uzly reprezentující tři různé značky (L_1,L_2,L_3). Stanovují se výsledná napětí mezi uzly pro všechny značky zvlášť. Na počátku každého dílčího výpočtu jsou uzly jedné značky nastaveny na hodnotu 1 (myšleno např. jako zdroj napětí) a uzly označené jinými značkami jsou nastavené na 0 (např. jako zem). Neoznačené uzly mají nastavené neznámé, nevypočtené hodnoty. Hrany mezi uzly lze vnímat jako rezistory a podle fyzikálních zákonů se vypočítá napětí v jednotlivých uzlech. Takto vypočtené elektrické napětí reprezentuje

pravděpodobnost, s jakou náhodná procházka začínající v každém uzlu dorazí jako první do uzlu se značkou právě nastaveného na 1. Obrázek 12(a) ukazuje počáteční značky a výslednou segmentaci vypočtenou na základě největší pravděpodobnosti. Součet pravděpodobností ze všech dílčích výpočtů je v každém uzlu roven 1. Pro tuto ilustraci byly všechny váhy hran (rezistory) nastaveny na 1. V případě obrazu mohou být tyto hodnoty funkcí např. intenzity jasu.



Obrázek 13: Segmentace obrazu s nejasnými hranicemi

Na obrázku 13(a) je demonstrace algoritmu ukazující syntetický obraz (pouze bílé a černé pixely) se 4 oblastmi oddělenými chabými dělenými hranicemi. Jednotlivá semínka jsou na 13(b) označena černou čarou a výsledná segmentace různou barvou šedé.

4 Efektivní segmentace obrazu založená na teorii grafů

Pro detailní nastudování, implementaci a praktické testy byl vybrán algoritmus od Pedro F. Felzenszwalba a Daniel P. Huttenlochera, který publikovali v roce 2004 v článku „Efficient Graph-Based Image Segmentation“ [1]. V této kapitole se podrobně seznámíme s vlastním principem a teoretickým základem metody.

4.1 Motivace

Hlavním cílem autorů bylo vyvinout segmentaci použitelnou v širokém spektru úloh počítačového vidění, podobně jako například detekce hran. Podle autorů je důležité, aby taková metoda měla následující vlastnosti:

1. *Zachycení důležitých oblastí a regionů, které se v obrazu nacházejí.* Z tohoto důvodu je důležité precizně definovat vlastnosti výsledné segmentace tak, aby umožnily porovnání různých přístupů a metod.
2. *Vysoká výkonnost, tj. běžet v reálném čase blízkém lineární závislosti na počtu pixelů.* Pro praktické využití je nezbytné, aby rychlost segmentační metody byla srovnatelná s rychlostí metod jako detekce hran, či jiných rychlých metod pro zpravování obrazu. Například segmentační metoda, která zvládne několik snímků za sekundu, může být použita pro zpracování video signálu.

Zde popisovaná segmentační metoda běží s maximálně $O(n \log n)$ časovou složitostí pro n pixelů s malými konstantami a tak je vhodná i pro zpracování pohyblivého obrazu. Časová složitost bude dále podrobněji objasněna.

Podobně jako algoritmy uvedené v předešlé kapitole, i tato metoda je založena na zpracování grafu, ve kterém je každý pixel uzlem grafu a je spojen se sousedním pixelem neorientovanou hranou. Každá hrana je ohodnocena dle odlišnosti spojených pixelů.

4.2 Princip metody

V této podkapitole si popíšeme principy a hlavní myšlenky metody, poukážeme na jednoduchost algoritmu a v závěru nastíníme některé modifikace pro konstrukci grafu, které nabízejí autoři.

Metodu samotnou lze pro zjednodušení rozdělit na dva kroky:

a) Vytvoření grafu
b) Zpracování grafu

Vytvoření grafu

Prvním úkolem metody je převést obraz na ohodnocený neorientovaný graf. Podobně jako metody jmenované v předešlé kapitole, i tato metoda ke každému pixelu v obraze vytváří uzel v grafu, jež daný pixel reprezentuje. Tedy počet uzlů v grafu je totožný s počtem pixelů v obraze. V základní verzi autoři spojují hranami uzly reprezentující sousední pixely v obraze, a to ve všech osmi směrech (*pozn.*: uzly na okraji grafu jsou spojeny pouze se svými pěti sousedy, resp. uzly v rozích grafu se třemi sousedy). Tímto vzniká pravidelný obdélníkový graf neboli mřížka.

Pro ohodnocení hran se používá některá metrika pro měření odlišnosti pixelů, například absolutní hodnota rozdílu intenzit jasu.

Zpracování grafu

Algoritmus je založený na spojování uzlů do komponent podle ohodnocení hrany, která je spojuje. Komponenta je skupina navzájem propojených uzlů tak, že z každého uzlu vede cesta do libovolného jiného uzlu komponenty. Hrana s malou váhou, resp. nulovou váhou znamená, že uzly jsou si velice podobné a patří do stejné komponenty. Naproti tomu hrana s vysokým ohodnocením spojuje natolik rozdílné uzly, že tvoří hranici mezi komponentami (*pozn.*: neexistující hranu mezi komponentami lze chápat jako hranu s nekonečně velkým ohodnocením). Na konci algoritmu je graf rozdělen do určitého počtu komponent a každý uzel je součástí pouze jedné z nich. Každá komponenta představuje výsledný segment obrazu.

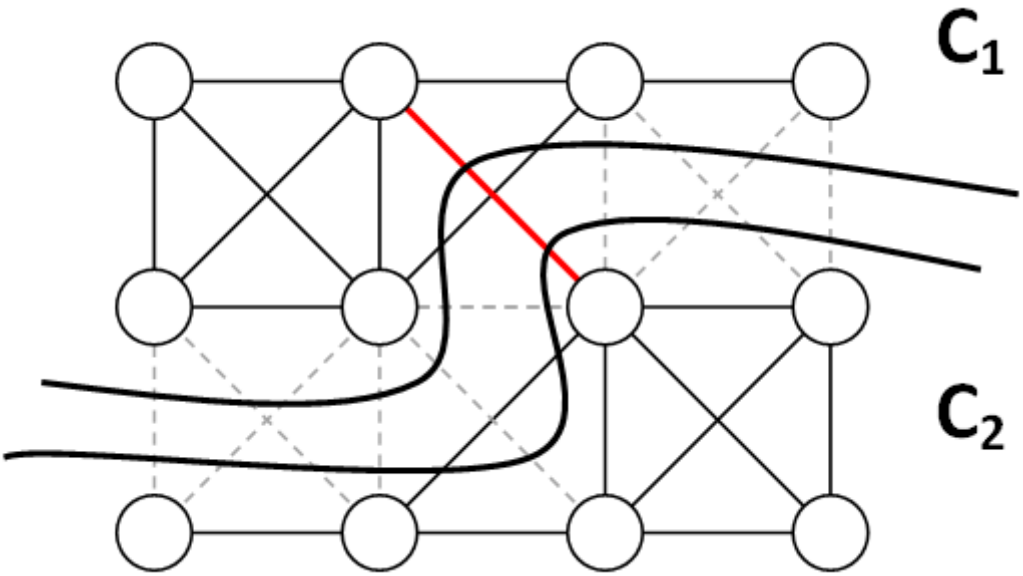
Algoritmus postupně prochází neklesající posloupnost hran a u každé hrany vyhodnotí, zda spojuje dvě komponenty, které jsou navzájem odlišné (neprovede nic), nebo spojuje dvě komponenty, které jsou si podobné, a ty pak sloučí v jednu komponentu. Jak vylývá z předešlé věty, algoritmus je založen na dvou základních myšlenkách. Za prvé setřídění hran do neklesající posloupnosti a za druhé podmínce pro porovnání podobnosti dvou komponent.

Pokud jsou hrany seřazeny do neklesající posloupnosti, je patrné, že na počátku algoritmu se zpracovávají hrany s nulovým či velmi malým ohodnocením, jež spojují velmi podobné uzly, dochází tak ke spojování komponent. Pokud hrana spojuje dvě rozdílné komponenty a algoritmus rozhodne o jejich nesloučení, tyto dvě komponenty již nikdy nebudou spojeny, neboť všechny další ještě nezpracované hrany mají stejné, nebo větší ohodnocení. Neklesající posloupnost má ještě jeden důsledek, jehož je využito při výpočtu vnitřní velikosti komponenty.

Dříve než si vysvětlíme podmínku, pomocí které algoritmus rozhoduje o sloučení dvou komponent, zavedeme si pojem vnitřní velikost komponenty. Vnitřní velikost komponenty je definována jako velikost její minimální kostry (definice 3.3). Nalezení minimální kostry v komponentě je NP těžký problém a tak autoři tuto definici upravili. Vnitřní velikost komponenty je hodnota hrany s největším ohodnocením, ze všech hran uvnitř komponenty. Pro komponenty s jedním uzlem, neboli žádnou hranou je vnitřní velikost komponenty 0. Praktickým

důsledkem této definice je rychlost výpočtu. Pokud víme, že postupně zpracováváme hrany z neklesající posloupnosti, víme také, že právě zpracovávaná hrana má největší ohodnocení, ze všech hran, které byly doposud zpracované. Jestliže v důsledku podmínky spojíme dvě komponenty v jednu, pak její hrana s největším ohodnocením má hodnotu rovnou hodnotě právě zpracovávané hrany. A tak hodnota právě zpracovávané hrany je rovna vnitřní velikosti nové komponenty.

Při zpracování hrany mohou nastat dvě situace. Hrana spojuje uzly z jedné komponenty, pak není třeba vykonat žádnou operaci, neboť se jedná o jednu a tutéž komponentu. V opačném případě, kdy hrana spojuje dva uzly z rozdílných komponent, autoři definují podmínku pro porovnání dvou komponent a rozhodnutí, zda se mají sloučit.



Obrázek 14: Část pravidelného mřížkového grafu s vyznačenými hranicemi dvou komponent

Dvě komponenty jsou si podobné (a budou sloučeny), pokud velikost hrany je menší nebo rovna vnitřní velikosti alespoň jedné z komponent. To znamená, že hrana spojuje uzly, které jsou si více či stejně podobné, jako uzly uvnitř komponenty samotné. A tak zjevně takové uzly patří do stejné komponenty. Na obrázku 14 vidíme část grafu reprezentujícího obraz. Algoritmus již zpracoval černé plné hrany a rozdělil tuto část grafu na dvě komponenty, jejíž hranice jsou znázorněny černými silnými křivkami. Ještě nezpracované hrany jsou šedé čárkované a právě zpracovávaná hrana je zvýrazněna červeně. Mohou nastat dvě situace. Buď je ohodnocení hrany větší, než vnitřní velikost obou komponent, pak tato hrana představuje přirozenou hranici mezi nimi a ke sloučení komponent nedojde ani zpracováním dalších čárkovaných hran, které mají stejnou nebo větší hodnotu. Nebo ohodnocení hrany je stejné, nebo menší než vnitřní vzdálenost alespoň jedné

z komponent, pak tato hrana není hranicí a ke sloučení komponent dojde, bez ohledu na ohodnocení zbývajících čárkovaných hran.

Výše definovaný výpočet vnitřní vzdálenosti má jednu slabinu. Při uvážení postupného zpracovávání hran v neklesající posloupnosti, podmínka by velmi brzy přestala komponenty slučovat (všechny hrany s ohodnocením 1 a více jsou určitě větší než vnitřní velikost 0 obou komponent – podle podmínky se jedná o hranici).

A tak autoři upravují výpočet vnitřní velikosti komponenty tak, že k původnímu výsledku přičítají hodnotu tzv. prahové funkce. V základní verzi je prahová funkce definovaná jako podíl libovolné kladné konstanty s počtem uzlů v komponentě. To má za následek, že pro malé komponenty dává prahová funkce velké hodnoty (blízké konstantě) a pro velké komponenty je její hodnota malá. Komponenty tak mají vnitřní velikost nepřímo úměrnou definované konstantě. Takto se malé komponenty snáze spojí s komponentami velkými.

Modifikace

Pro segmentaci barevných obrazů je nejrychlejší nejprve převést pixely na hodnotu intenzity jasu (neboli černobílý obraz), například jako jednoduchý průměr všech jejích barevných složek. Pak je algoritmus segmentace vykonán pouze jednou. Autoři také doporučují postup zpracování každé barevné složky zvlášť základním algoritmem a výslednou segmentaci pak generovat jako průnik dílčích segmentací.

Chování algoritmu lze ovlivnit také změnou prahové funkce. Například si představme, že bychom měli za úkol spojovat v obraze podlouhlé objekty. Vhodnou úpravou prahové funkce tak, aby pro podlouhlé komponenty vracela vysoké číslo a pro ostatní malé, je možné takový problém řešit.

Dalším příkladem, který autoři uvádějí, je květinový záhon. Základní metoda segmentace rozdělí květinky na mnoho segmentů, podobně jako jejich zelené stonky a trávník. Pro autory ideální segmentací takového obrazu je jen několik segmentů, jeden pro všechna červená kvítí, druhý pro jejich stonky a třetí pro trávník. K docílení takového výsledku definují následující způsob vytváření hran v grafu.

Pixely jsou imaginárně umístěny do pětirozměrného prostoru (x, y, r, g, b), kde kromě pozice v obraze jsou dalšími rozměry jednotlivé barevné složky pixelu. Dále je definovaná fixní vzdálenost d tak, že každý uzel (pixel) se spojí s jiným, pokud je v tomto pětirozměrném prostoru vzdálen maximálně d. To má za následek, že jsou hranou spojeny uzly (pixely), které mají stejnou, či podobnou barvu a přitom jsou od sebe vzdáleny i přes několik pixelů. Váha hrany je pak definovaná jako Euklidovská vzdálenost mezi sousedními uzly (dle potřeby připouštějí i jinak definovanou funkci). Takto generovaný graf je zpracován základním algoritmem.

4.3 Predikát pro porovnání dvou regionů

V následujících kapitolách si předešlý, trochu naivní popis ukážeme ještě jednou a to pomocí matematických definicí, vět a důkazů. Postupně si nadefinujeme jednotlivé pojmy a vzorce a dokážeme, že algoritmus je matematicky korektní. Začneme predikátem vyhodnocujícím, zda existuje v segmentaci hranice mezi dvěma komponentami (dvěma regiony v obraze). Tento predikát je založen na porovnání rozdílnosti mezi dvěma komponentami s rozdílností elementů uvnitř každé komponenty.

Nejprve si nadefinujme pomocné predikáty, které nám umožní komponenty porovnávat a měřit.

Predikát 4.1:
Vnitřní velikost komponenty $C \subseteq V$ je největší váha hrany v minimální kostře komponenty (anglicky Minimal Spanning Tree) $MST(C, E)$. Tedy:

$$Int(C) = \max_{e \in MST(C,E)} w(e)$$

Intuitivně lze říci, že vnitřní velikost komponenty $C \subseteq V$ je maximální váha hrany $Int(C)$, s jakou jsou elementy v dané komponentě spojeny.

Predikát 4.2:
Rozdíl mezi dvěma komponentami $C_1, C_2 \subseteq V$ je minimální váha hrany spojující tyto dvě komponenty. Tedy:

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w\left((v_i, v_j)\right)$$

Pokud neexistuje hrana spojující C_1 a C_2 pak $Dif(C_1, C_2) = \infty$.

Pozn.: tato definice rozdílu může být v principu problematická, jelikož bere v úvahu pouze hranu s nejmenší váhou mezi dvěma komponentami. Ve skutečnosti se však ukázalo, že tato míra dává dobré výsledky i navzdory zřejmé nepřesnosti. Mimo to, úprava definice na výpočet mediánu (či jiného kvantilu), aby byla odolná proti statisticky odlehlým hodnotám, učiní celý algoritmus NP těžkým. Tedy tato drobná změna na kritérium drasticky mění obtížnost řešení problému segmentace.

Predikát pro porovnání dvou regionů vyhodnocuje, zda existuje hranice mezi dvěma komponentami porovnáním, zda rozdíl mezi dvěma komponentami $Dif(C_1, C_2)$ je větší vůči vnitřní velikosti alespoň jedné komponenty $Int(C_1)$ nebo $Int(C_2)$.

Predikát 4.3:

$$D(C_1, C_2) = \begin{cases} true & pokud\ Dif(C_1, C_2) > MInt(C_1, C_2) \\ false & jinak \end{cases}$$

Kde funkce minimální vnitřní velikosti $MInt$ je definovaná jako

$$MInt(C_1, C_2) = min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$$

Prahová funkce τ udává minimální velikost, o kolik musí být rozdíl mezi komponentami větší než jejich vnitřní velikosti. Samotná funkce $Int(C)$ znevýhodňuje malé komponenty (ty by se pak nepřipojovaly k velkým komponentám) a v extrémním případě, kdy počet uzlů v komponentě $|C| = 1$ je hodnota funkce dokonce $Int(C) = 0$, jelikož neobsahuje žádnou hranu. Proto je prahová funkce odvozená od velikosti komponenty, tedy:

$$\tau(C) = k/|C|$$

kde $|C|$ je velikost komponenty a k je libovolná kladná konstanta.

Pozn.: z toho vyplývá, malé komponenty musí být spojeny s ostatními komponentami pouze hranami s vysokým ohodnocením, aby existovala hranice mezi nimi. Prakticky k udává stupeň segmentace, kdy vyšší k způsobí vznik větších komponent (nicméně k neznamená minimální velikost komponenty). Menší komponenty jsou přípustné pouze tehdy, pokud existuje dostatečný rozdíl mezi sousedícími komponentami.

Pozn.: jako funkce τ může být použita libovolná nezáporná funkce, což nemá žádný vliv na algoritmus popsaný níže. Například je možné mít segmentační metodu preferující komponenty určitých tvarů definováním funkce τ tak, aby dávala vysoké hodnoty pro komponenty neodpovídající konkrétním tvarům a naopak. To způsobí spojování komponent, které nejsou požadovaného tvaru, do velkých komponent představujících pozadí a komponenty požadovaného tvaru představující objekty zájmu. Výběr tvarů může být obecný, např. komponenty, které jsou dlouhé a tenké (např. poměr obvodu a plochy), nebo konkrétní, komponenty odpovídající konkrétnímu tvaru.

4.4 Algoritmus a jeho vlastnosti

V této kapitole si vysvětlíme a zanalyzujeme algoritmus generující segmentaci za použití rozhodovacího kritéria D zmíněného výše. Ukážeme, že tato segmentace splňuje vlastnosti – není příliš hrubá ani příliš jemná – podle následujících definic, které uvádějí autoři ve svém článku. Termíny příliš hrubá a příliš jemná se snaží popsat obecné vlastnosti segmentace.

Definice 4.1:
Segmentace S je *příliš jemná*, pokud v ní existují dva regiony $C_1, C_2 \in S$, pro které neexistuje zjevná hranice mezi nimi.

Zde stojí namístě si nadefinovat pojem *zjemnění* segmentace. Mějme dvě segmentace S a T, které byly vytvořeny ze stejného obrazu. Pak řekneme, že T je *zjemněním* segmentace S, pokud každá komponenta T je obsažena v (nebo je shodná s) komponentě segmentace S. Navíc říkáme, že T je *vlastním zjemněním* segmentace S, pokud $T \neq S$.

Pozn.: pokud T je vlastním zjemněním segmentace S, pak T může být získáno rozdělením jednoho, nebo více regionů segmentace S. Pokud T je vlastním zjemněním segmentace S říkáme, že T je jemnější než S a S je hrubší než T.

Definice 4.2:
Segmentace S je *příliš hrubá*, pokud existuje vlastní zjemnění S, které není příliš jemné.

Toto vystihuje intuitivní představa, že regiony segmentace mohou být rozděleny v místech, kde existují hranice mezi sousedními regiony, které původní segmentace sloučila dohromady.

Vyvstávají dvě přirozené otázky, které se týkají segmentace, která není příliš hrubá ani příliš jemná. Za prvé zda vždy existuje nějaká a pokud ano, zda je jedinečná. Na druhou otázku lze odpovědět, že obecně může existovat více segmentací, jež nejsou ani příliš hrubé ani příliš jemné, takže taková segmentace není jedinečná. Důkaz, že vždy existuje taková segmentace, prokáže následující vlastnost.

Vlastnost 4.1:
Pro každý (konečný) graf $G = (V, E)$ existuje nějaká segmentace S, která není ani příliš hrubá ani příliš jemná.

Mějme segmentaci, kde všechny elementy jsou v jediné komponentě. Pak taková segmentace není příliš jemná, protože má pouze jedinou komponentu. Pokud segmentace není ani příliš hrubá, jsme hotovi. Jinak, dle definice příliš hrubé segmentace, existují vlastní zjemnění, které nejsou příliš jemné. Vezmeme jedno takové zjemnění a proceduru opakujeme, dokud nezískáme segmentaci, která není příliš hrubá. V nejkrajnějším případě získáme segmentaci, kde každému pixelu odpovídá jedna komponenta. Ale taková segmentace již není příliš hrubá.

Algoritmus 4.1:
Vstupem je neorientovaný ohodnocený graf $G = (V, E)$ s n vrcholy a m hranami. Výstupem je segmentace vrcholů z množiny V do komponent, neboli $S = (C_1, \dots, C_r)$.

0. Setřiď hrany E grafu do $\pi = (o_1, \dots, o_m)$, podle váhy hrany, do neklesající posloupnosti.
1. Začni se segmentací S^0, kde každý vrchol je ve své vlastní komponentě.
2. Opakuj krok 3 pro $q = 1, \dots, m$.
3. Vytvoř segmentaci S^q vzhledem k S^{q-1} následovně. Mějme vrcholy v_i a v_j spojené q-tou hranou v pořadí, $o_q = (v_i, v_j)$. Pokud vrcholy v_i a v_j jsou v různých komponentách segmentace S^{q-1} a váha hrany $w(o_q)$ je menší v porovnání s vnitřní velikostí obou komponent, pak obě komponenty spoj, jinak nedělej nic. Více formálně, nechť C_i^{q-1} je komponentou segmentace S^{q-1} obsahující vrchol v_i a C_j^{q-1} je komponentou segmentace S^{q-1} obsahující vrchol v_j. Pokud $C_i^{q-1} \neq C_j^{q-1}$ a $w(o_q) \leq MInt\left(C_i^{q-1}, C_j^{q-1}\right)$, pak segmentaci S^q získáme ze segmentace S^{q-1} sloučením komponent C_i^{q-1} a C_j^{q-1}. Jinak je $S^q = S^{q-1}$.
4. Vrať segmentaci $S = S^m$.

Nyní si ukážeme, že segmentace získaná algoritmem 4.1 není příliš jemná ani příliš hrubá při použití porovnání D podle predikátu 4.3. To je dáno tím, že algoritmus dělá pouze taková rozhodnutí, která produkují segmentaci splňující tyto vlastnosti. Navíc si ukážeme, že setřídění hran do libovolné neklesající posloupnosti (v kroku 0), dává vždy stejnou výslednou segmentaci.

Lemma 4.1:
V třetím kroku algoritmu, kde díky hraně o_q jsou dvě odlišné komponenty ponechány a nejsou sloučeny, pak jedna z těchto komponent bude i ve finální segmentaci. Nechť C_i^{q-1} a C_j^{q-1} jsou dvě komponenty spojené hranou $o_q = (v_i, v_j)$. Pak buď $C_i = C_i^{q-1}$ nebo $C_j = C_j^{q-1}$, kde C_i je komponenta obsahující vrchol v_i a C_j je komponenta obsahujicí vrchol v_j ve finální segmentaci.

Důkaz:
Mohou existovat dva případy, díky kterým nedojde ke sloučení dvou komponent. Řekněme, že je to díky vlastnosti $w(o_q) > Int(C_i^{q-1}) + \tau(C_i^{q-1})$. Vzhledem ke skutečnosti, že hrany jsou seřazeny do neklesající posloupnosti, pak $w(o_k) \geq w(o_q)$ pro všechna $k \geq q + 1$. Tudíž nedojde k žádnému dalšímu sloučení s touto komponentou, tedy $C_i = C_i^{q-1}$. Analogicky pro druhý případ kdy $w(o_q) > Int\left(C_j^{q-1}\right) + \tau\left(C_j^{q-1}\right)$. ■

Pozn.: Lemma 4.1 naznačuje, že váha hrany způsobující sloučení dvou komponent je přesně rovna hraně s minimální vahou mezi komponentami. A tak hrany, které způsobí sloučení, jsou

přesně hranami, které mohou být vybrány Kruskalovým algoritmem pro konstrukci minimální kostry (MST) každé komponenty.

Věta 4.1:
Segmentace S generovaná algoritmem 4.1 není příliš jemná podle definice 4.1, při použití predikátu 4.3 pro porovnání regionu D.

Důkaz:
Podle definice, aby segmentace S byla příliš jemná, tak existuje nějaký pár komponent, pro který D neplatí. Musí tedy existovat alespoň jedna hrana mezi těmito dvěma komponentami, která již byla v kroku 3 zpracována a zároveň nezpůsobila sloučení komponent. Mějme $o_q = (v_i, v_j)$ jako první takovou hranu v pořadí. V tomto případě se algoritmus rozhodl nesloučit C_i^{q-1} a C_j^{q-1}, což znamená $w(o_q) > MInt\left(C_i^{q-1}, C_j^{q-1}\right)$. Podle lemma 4.1 víme, že buď $C_i = C_i^{q-1}$ nebo $C_j = C_j^{q-1}$. Také vidíme, že $w(o_q) > MInt(C_i, C_j)$ implikuje, že D platí pro C_i a C_j, což je rozpor. ■

Věta 4.2:
Segmentace S generovaná algoritmem 4.1 není příliš hrubá podle definice 4.2, při použití predikátu 4.3 pro porovnání regionu D.

Důkaz:
Podle definice, aby segmentace S byla přiliš hrubá, musí existovat vlastní zjemnění T, které není příliš jemné. Uvažujme hranu s minimální vahou, která je uvnitř komponenty $C \in S$, ale spojuje odlišné komponenty $A, B \in T$. Podle definice zjemnění $A \subset C$ a $B \subset C$.

Jelikož T není příliš jemná, buď $w(e) > Int(A) + \tau(A)$, nebo $w(e) > Int(B) + \tau(B)$. Bez ztráty obecnosti řekněme, že první je pravdivý. Podle konstrukce každá hrana spojující A s jinou pod-komponentou komponenty C má váhu alespoň tak velkou jako $w(e)$, která je následně větší než maximální váha hrany v minimální kostře $MST(A, E)$, protože $w(e) > Int(A)$. Proto algoritmus, který má hrany setříděné do neklesající posloupnosti dle váhy, musel zpracovat všechny hrany v $MST(A, E)$ před zpracováním libovolné hrany z A spojené s jinou částí z C. Takže algoritmus musel sestavit A před sestavením C a při sestavování C musel spojit A s některou jinou pod-komponentou komponenty C. Váhy hrany, která způsobila sloučení, musela být alespoň tak velká jako $w(e)$. Avšak algoritmus by neměl sloučit A v případě, že platí $w(e) > Int(A) + \tau(A)$, což je rozpor. ■

Věta 4.3:
Segmentace S generovaná algoritmem 4.1 nezávisí na pořadí hran seřazených do libovolné neklesající posloupnosti.

Důkaz:
Každá posloupnost může být změněna na jinou posloupnost pouze záměnou sousedních elementů. Proto je dostačující ukázat, že záměna pořadí dvou sousedních hran se stejnou vahou v neklesající posloupnosti nezmění výsledek generovaný algoritmem 4.1.

Nechť e_1 a e_2 jsou dvě hrany se stejnou vahou, které sousedí v nějaké neklesající posloupnosti. Je zřejmé, že pokud hrany spojují rozdílné páry komponent, nebo naopak přesně stejné páry komponent, na pořadí jejich zpracování algoritmem nezáleží. Případ, který musíme ověřit, je pokud hrana e_1 spojuje komponenty A a B a hrana e_2 spojuje některou z těchto komponent, řekněme B s jinou komponentou C.

Nyní ukážeme, že hrana e_1 způsobí spojení, pokud je zpracovaná po hraně e_2, stejně jako pokud je zpracovaná před ní. Nejprve předpokládejme, že hrana e_1 způsobí spojení před zpracováním hrany e_2. To implikuje $w(e_1) \leq MInt(A, B)$. Pokud by hrana e_2 byla místo toho zpracována před hranou e_1, buď by hrana e_2 nezpůsobila spojení a hrana e_1 by jednoduše stále způsobila spojení, nebo by hrana e_2 způsobila spojení a pak nová komponenta $B \cup C$ by měla mít vnitřní velikost $Int(B \cup C) = w(e_2) = w(e_1)$. Z toho plyne, že $w(e_1) \leq MInt(A, B \cup C)$, což implikuje, že hrana e_1 stále způsobí spojení. Nyní předpokládejme, že hrana e_1 nezpůsobí spojení před zpracováním hrany e_2. To implikuje $w(e_1) > MInt(A, B)$. Takže buď $w(e_1) > Int(A) + \tau(A)$ a v tom případě to bude stále platné i když hrana e_2 bude zpracovaná jako první (hrana e_2 není spojena s A), nebo $w(e_1) > Int(B) + \tau(B)$. V tom případě ani hrana e_2 nemůže způsobit spojení, protože $w(e_2) = w(e_1)$ a tak $w(e_2) > Int(B) + \tau(B)$, i když bude zpracovaná jako první. Proto i když je hrana e_1 zpracovaná až po hraně e_2 stále platí, že $w(e_1) > MInt(A, B)$ a hrana e_1 nezpůsobí spojení. ■

Algoritmus 4.1 lze rozdělit na dvě operace. První v kroku 0, je nezbytné všechny hrany setřídit do neklesající posloupnosti. Pokud jsou váhy hran celé hodnoty, je tato operace $O(m)$ lineárně závislá na počtu hran použitím přihrádkového třídění. Obecně pro reálné hodnoty vah je časová složitost $O(m \log m)$, kde m je počet hran.

Kroky 1-3 reprezentující samotný algoritmus má časovou složitost $O(m \propto (m))$, kde $\propto$ je velmi pomalu rostoucí inverzní Ackermanovou funkcí, jak uvádějí autoři a jejíž důkaz je nad rámec této práce. Pro hrubou představu. Zpracování každé hrany představují až tři samostatné operace. První je porovnání, zda hrana spojuje dvě komponenty, které mají být sloučeny. Operace se vykonává pro každou hranu a je časově konstantní. Další dvě operace jsou vykonávány pouze tehdy, pokud

dochází ke sloučení komponent. Časově konstantní výpočet vnitřní velikosti komponenty. A operace sloučení elementů dvou komponent.

Z předešlých úvah vyplývá, že celková časová náročnost algoritmu je závislá na typu ohodnocení hran potažmo na časové složitosti třídícího algoritmu. Obecná časová složitost algoritmu 4.1 je $O(n \log n)$, kde n je počet pixelů, neboť m je lineárně závislé na n. Pokud jsou hrany ohodnoceny celočíselnými hodnotami, časová složitost algoritmu je pouze $O(n \propto(n))$.

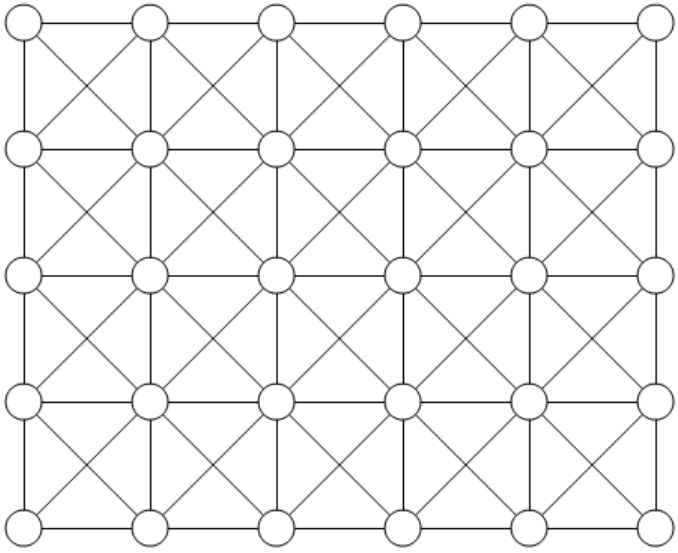
5 Implementační poznámky

Algoritmus jsme implementovali v jazyce C++, vývojovém prostředí Visual Studio 2010 pro operační systém Windows, zdrojové kódy jsou samostatnou přílohou této práce. V této kapitole bychom rádi vyzdvihli některá specifika implementace a seznámili čtenáře s vlastním přístupem k programování a řešení problému. Řekneme si, které základní operace metoda provádí a jak lze výslednou segmentaci modifikovat pomocí parametrů.

5.1 Operace metody

Autoři původní metody připouštějí a nabízejí reprezentace obrazu různými grafy, které více či méně respektují lokální a globální vlastnosti obrazu. Rovněž pro segmentaci barevného obrazu popisují několik jeho variant. Pro jednoznačnou implementaci metody jsme stanovili následující pravidla:

a) Graf, reprezentující obraz, tvoří dvourozměrnou mřížku, kde každý uzel představuje pixel v obraze. Každý uzel je spojen hranou se sousedním (dle topologie) ve všech 8 směrech.



Obrázek 15: Pravidelný mřížkový graf

b) Pro ohodnocení hrany se stanovuje rozdíl stupňů šedi mezi sousedícími uzly (resp. pixely), stupně šedi jsou definovány na rozsahu 0 až 255. Barevné obrazy jsou převedeny na stupně šedi jednoduchým průměrem hodnot všech jejich složek.

Implementovanou funkci lze rozdělit do následujících sekvenčních operací, které společně generují výslednou segmentaci:

1. Předzpracování obrazu
2. Převedení obrazu do grafu
    a. Vytvoření mřížkového grafu, kde každý pixel je spojen s 8 sousedními pixely
    b. Výpočet ohodnocení hrany
    c. Vložení hrany do setříděného seznamu
3. Zpracování grafu dle algoritmu 4.1
4. Přečíslování indexů výsledných segmentů

Jak si ukážeme dále v kapitole implementovaný algoritmus má celkovou prostorovou složitost $O(n)$, tedy lineární a celkovou časovou složitost Ackermanovskou $O(n \propto(n))$, kde $\propto$ je velmi pomalu rostoucí inverzní Ackermanovou funkcí (jak bylo napsáno v předešlé kapitole).

5.1.1 Předzpracování obrazu

V rámci předzpracování obrazu je nejprve obraz (černobílý i barevný) převeden do dvourozměrného jednobajtového pole stupňů šedi v rozmezí hodnot 0 až 255, kde 0 znamená absolutně černý pixel a 255 absolutně bílý pixel. Před převedením obrazu do grafu, je vhodné obraz zbavit nežádoucích příznaků digitalizace použitím některého filtru. Časová složitost operace je lineární $O(n)$.

Konvoluce

Důležitou operací při lineárním přístupu ke zpracování obrazu je konvoluce. Obecně konvoluce dvourozměrné spojité funkce signálu je definována pomocí dvojného integrálu. V našem případě, kdy zpracováváme digitální signál, budeme používat diskrétní konvoluci. Definice [7]:

Nechť je $\Omega = \{(m,n) | m = 0, \dots, M-1; n = 0, \dots, N-1\}$ je prostor diskrétních signálů a $f(m,n)$ a $h(m,n)$ jsou signály z uvažovaného prostoru:

$$(f * h)(m,n) = \sum_{r=0}^{M-1} \sum_{s=0}^{N-1} f(r,s) \cdot h(m-r, n-s)$$

kde $f(m,n)$ je diskrétní obraz a $h(m,n)$ se nazývá jádro konvoluce.

Gaussovská filtrace

Na filtraci lze nahlédnout jako na diskrétní konvoluci, přičemž konvoluční jádro definujeme jako pravoúhlé okolí. V našem případě však zvyšujeme váhu středového bodu konvolučního jádra, aby lépe aproximovalo vlastnosti šumu Gaussovským rozložením. Příkladem takového konvolučního jádra neboli filtru je:

$$h_{16} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, h_{10} = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

V algoritmu se velikost konvolučního jádra a rozložení jeho hodnot počítá a následně normalizuje dle vstupního parametru. Účinnost filtru tak lze ovlivnit nastavením parametru metody σ, kde 0 – znamená žádné vyhlazení a typicky používaná hodnota je 0.8, která nemá viditelný vliv na obraz a přitom pomáhá odstranit nežádoucí šumy.

5.1.2 Převedení obrazu do grafu

Z hlediska implementace jde operaci, která má významný vliv na celkovou časovou a prostorovou složitost metody. S ohledem na konkrétní konstrukci grafu a ohodnocování jeho hran je časová i prostorová složitost operace lineární tedy $O(n)$.

V implementované metodě segmentace jsou použity 3 datové struktury pro uzly, hrany a komponenty (ve zdrojovém kódu je použito anglických názvů). Jejich datové položky a význam je následující:

```
struct NODE {
  COMPONENT* comp;
  NODE* next;
}
```

Proměnná `comp` ukazuje na komponentu, ke které daný uzel právě přísluší. Proměnná `next` ukazuje na další uzel v dynamickém listu uzlů (každá komponenta má vlastní list uzlů, které k ní náleží).

```
struct EDGE {
  NODE* v1, v2;
  EDGE* next;
}
```

Proměnné `v1` a `v2` jsou ukazateli na uzly, které hrana spojuje. Proměnná `next` ukazuje na další hranu v dynamickém listu (posloupnosti) hran.

Pozn.: ohodnocení hrany není zaznamenáno datovou položkou, ale je řešeno vložením hrany do příslušného listu, jak bude uvedeno dále.

```
struct COMPONENT {
  int index, count;
  double intPtau;
  NODE* first, last;
}
```

Proměnná `index` se používá až v závěrečné operaci pro indexování segmentů výsledné segmentace, defaultní hodnota `-1` znamená nepřiřazený index. Do proměnné `count` se ukládá počet uzlů komponenty, na počátku algoritmu je nastaven na `1`. V proměnné `intPtau` se ukládá

výsledek operace $Int(C) + \tau(C)$, jejíž počáteční hodnota je rovna parametru k. Proměnné `first` a `last` jsou příslušnými ukazateli na dynamický list uzlů komponenty.

Pozn.: výsledek algoritmu (segmentace) lze ovlivnit druhým parametrem metody k, který je použitý v prahové funkci τ. Pro připomenutí uvádíme její definici $\tau(C) = k/|C|$, kde $|C|$ je aktuální počet uzlů v komponentě C.

Metoda, pro potřeby algoritmu, alokuje 3 statická pole výše uvedených datových struktur.

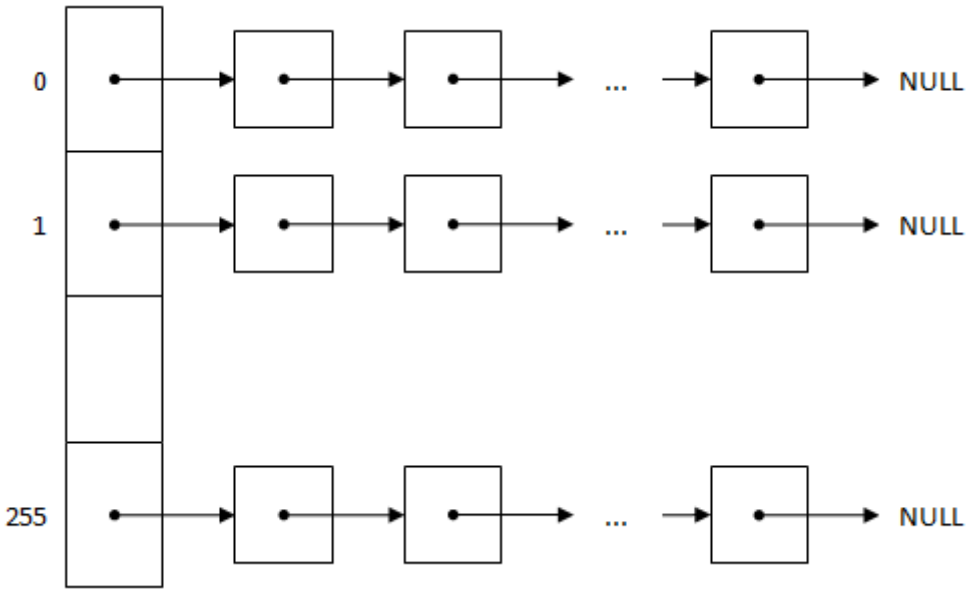
- Dvourozměrné pole uzlů o rozměrech $k \times l$, kde k je šířka (počet pixelů) obrazu a l je výška (počet pixelů) obrazu.
- Jednorozměrné pole komponent o velikosti n (počet všech pixelů $k \cdot l$).
- Jednorozměrné pole hran má velikost $4n$, což vyplývá z jednoduché úvahy, že uzel je spojen hranou se svými 8 sousedy, každá hrana má dva vrcholy, takže počet hran je ve výsledku poloviční. *Pozn.:* rovnice $4n$ není, co do počtu hran, matematicky přesná. Počet hran je ve skutečnosti o $[3(k + l) - 2]$ menší a to díky hraničním pixelům, které nejsou spojeny s 8 sousedními pixely, ale s 5 (resp. 3 pro rohové pixely). Pro zjednodušení indexace pole hran však toto malé množství nevyužité paměti akceptujeme.

Při výpočtu paměťové náročnosti takto reprezentovaného grafu musíme nejprve uvážit některé fakta. Každý pixel obrazu alokuje jeden objekt `NODE` a `COMPONENT` a 4 objekty `EDGE`. Dále předpokládejme spuštění segmentační metody na 32-bitovém operačním systému, kde velikost datového typu `int` je 4B (bajty), velikost obecného ukazatele je rovněž 4B a velikost datového typu `double` je 8B. Pak alokovaná paměť bude:

$$M = (8 + 24 + 4 * 12)n = 80n$$

, kde n je celkový počet pixelů. *Např.* obraz o rozměru 100×100 pixelů bude reprezentován grafem, který alokuje 800kB paměti.

Pro úplnost poznamenejme, že hrany jsou uspořádány, dle svého ohodnocení, v dynamických polích, pomocí tzv. přihrádkového řazení. Pro tato dynamická pole alokujeme jedno statické pole ukazatelů o 256 prvcích, kde nultý prvek ukazuje na první hranu s ohodnocením 0, první prvek ukazuje na první hranu s ohodnocením 1,…, poslední 255 prvek ukazuje na první hranu s ohodnocením 255. *Pozn.:* hrany v každém dynamickém poli jsou dosažitelné pomocí položky `next` datové struktury `EDGE`.



Obrázek 16: Pole dynamických listů

Postup vytvoření grafu

1. Alokuj potřebné množství paměti pro objekty typu `NODE`, `EDGE` a `COMPONENT`.
2. Pro všechny pixely obrazu opakuj následující kroky:
3. Inicializuj objekty `COMPONENT` a `NODE` aktuálního pixelu.
4. Pixel spoj hranami se sousedními pixely ve 4 směrech (vpravo nahoře, vpravo, vpravo dole a dole) a ohodnoť podle rozdílnosti pixelů. *Pozn.:* pokud pixel v daném směru neexistuje, hranu nevytvářej.
5. Každou hranu, dle ohodnocení, přidej na začátek příslušného dynamického listu.

5.1.3 Zpracování grafu

Tato operace implementuje algoritmus 4.1, přesněji řečeno kroky 1 – 3. Jak již bylo ukázáno v předešlé kapitole, časová složitost operace je $O(n \propto (n))$. Vzhledem k implementované reprezentaci grafu, lze původní algoritmus upravit následovně.

Implementovaný algoritmus

1. Jako aktuálně zpracovávané pole hran vezmi nultý prvek pole hran přihrádkového řazení, představující pole hran s ohodnocením 0.
2. Jako aktuálně zpracovávanou hranu vezmi hranu, na kterou ukazuje aktuální pole.
3. Pokud aktuální hrana je NULL pokračuj krokem 7.
4. Porovnej komponenty, do kterých patří uzly, jež spojuje aktuální hrana. Pokud komponenty mají být sloučeny, pokračuj krokem 5, jinak krokem 6.
5. Sluč komponentu s menším počtem uzlů B s komponentou s větším počtem uzlů A.
    a. Pro všechny uzly komponenty B nastav ukazatel na novou komponentu A.
    b. Spoj pole uzlů tak, že poslední uzel komponenty A ukazuje na první uzel komponenty B. Změň poslední uzel komponenty A na hodnotu posledního uzlu komponenty B.
    c. Vypočítej počet uzlů pro sloučenou komponentu A.
    d. Vypočítej novou hodnotu $Int(C) + \tau(C)$ pro sloučenou komponentu A.
6. Jako aktuálně zpracovávanou hranu vezmi hranu následující v dynamickém poli hran a pokračuj krokem 3.
7. Jako aktuálně zpracovávané pole hran vezmi další prvek ze statického pole přihrádkově řazených dynamických polí hran. Pokračuj krokem 2. Pokud jsou již zpracovány všechny hrany, algoritmus končí.

5.1.4 Přečíslování indexů

Metoda vrací pole indexů segmentů, které je stejně formátováno jako vstupní obraz. Místo barevné informace o pixelu je však uložena hodnota indexu segmentu. Časová složitost této operace je lineární, tedy $O(n)$.

Na počátku všechny segmenty nemají přiřazeny indexy. Při sekvenčním průchodu polem uzlů grafu, řádek po řádku, jsou do pole výsledné segmentace ukládány indexy segmentu, kterému uzel, potažmo pixel, přísluší. Pokud segment ještě nemá žádný index, je mu přiřazen další v pořadí. Segmenty se číslují kladnými čísly od 0.

6 Testování algoritmu

Testování algoritmu proběhlo na sadě obrazů získaných z webové stránky fakulty elektrotechniky a informatiky univerzity Berkeley v Kalifornii ve Spojených Státech Amerických [6]. Tyto jsem doplnil několika dalšími obrazy volně staženými z internetu.

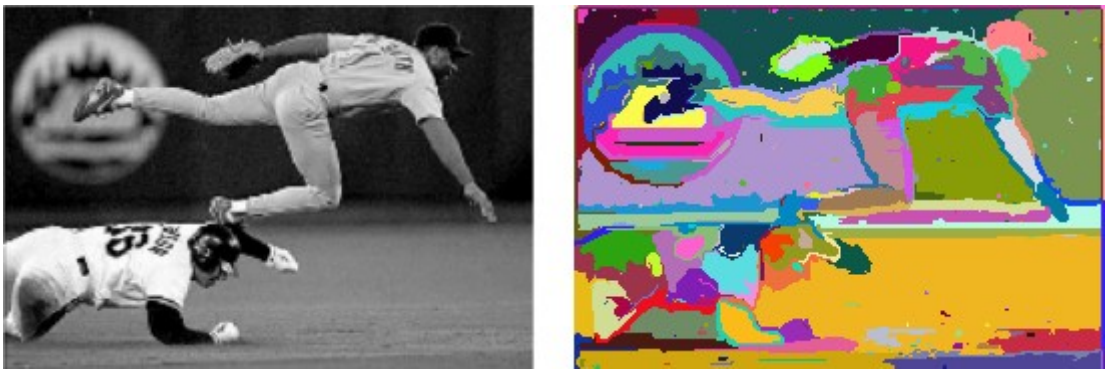
Detailní výsledky testů se nacházejí v samostatných souborech, které jsou přílohou této práce. Připravil jsem skript (konzolovou aplikaci) pro dávkové zpracování obrazů, který pro každý obraz vykonal algoritmus segmentace postupně s hodnotami parametru $\sigma = \{0; 0{,}4; 0{,}8; 1{,}2\}$ a hodnotami parametru $k = \{150; 300; 450\}$. Pro připomenutí parametr σ má vliv na filtrování nežádoucího šumu v obraze, kde hodnota 0 znamená žádná filtrace. Parametr k pak má vliv na připojování malých komponent k velkým komponentám grafu, čím větší hodnota, tím snadněji se malá komponenta spojí s velkou.

Pozn.: ke každé segmentaci byla vygenerován obraz, kde se jednotlivým segmentům přiřadila náhodná barva. Tyto obrazy jsem uložil do samostatných souborů. Proto opakované pokusy na stejné sadě obrazů nemusí vzniknout identicky stejné obrazy, co se barevnosti týká.

Test probíhal na počítači s dvoujádrovým procesorem Intel 2,8GHz, operační pamětí 4GB a operačním systémem Windows 7. Měřenými parametry byla rychlost segmentace a počet nalezených segmentů. Obecně lze říci, že nezanedbatelný podíl na rychlost segmentace měl parametr σ. Segmentace byla nejrychlejší při hodnotě $\sigma = 0$, kdy se filtrace obrazu neprovádí, naproti tomu při hodnotě $\sigma = 0{,}8$ byl čas segmentace přibližně dvojnásobný. Parametr k měl na rychlost segmentace vliv minimální.

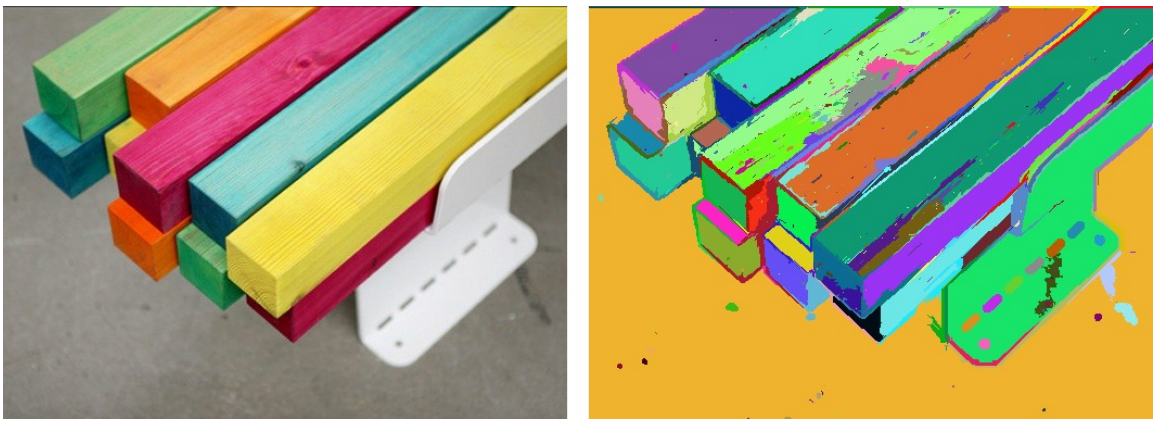
Všechny obrazy z Berkeley datového skladu byly barevné a měly stejný rozměr 481×321 pixelů. Rychlost segmentace těchto obrazů se pohybovala v rozmezí 40-60 ms při hodnotě parametru $\sigma = 0$. Tyto výsledky naznačují, že je algoritmus použitelný i pro segmentaci pohyblivého obrazu. Počet nalezených segmentů byl v závislosti na parametru σ a k velmi rozdílný a pohyboval se od řádu několika stovek, až blížící se k deseti tisícům.

V dalším textu se pokusím subjektivně zhodnotit algoritmus na několika obrazech, zda a jak dobře se podařilo nalézt důležité prvky obrazu. Různé aspekty segmentace se snažím zachytit vhodným výběrem segmentovaných obrazů.



Obrázek 17: Černobílá fotografie hráčů baseballu a její segmentace

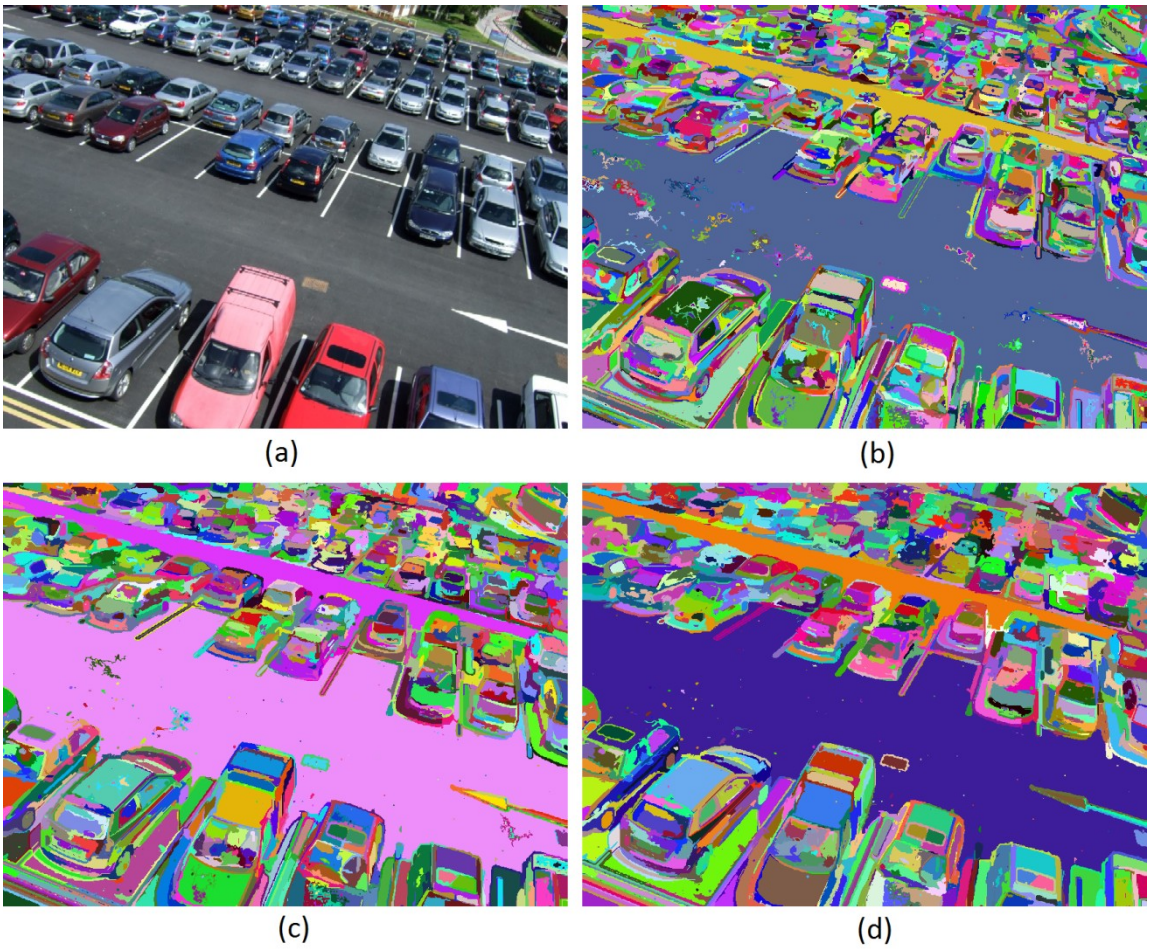
Jako první se podívejme na segmentaci fotografie, kterou uvádějí ve svém článku [1] autoři algoritmu. Obrázek 17 zobrazuje dva hráče baseballu, které můžeme vnímat jako objekty, travnatá plocha a hrazení s rozmazaným logem pak jako pozadí. Parametry segmentace jsou $\sigma = 0{,}8$ a $k = 300$. Největší segment představuje travnatou plochu a další čtyři velké segmenty jsou černé hrazení v pozadí. Úzký a podlouhlý stín pod ležícím hráčem způsobil vznik malého obdélníkového segmentu trávníku pod hráčem. Přechod mezi trávníkem a hrazením je složen z několika podlouhlých segmentů. Segmentace rozmazaného loga v pozadí je odpovídající. Hráči jsou rozděleni na mnoho malých segmentů zapříčiněno stíny na jejich oblečeních. U segmentace letícího hráče algoritmus našel celistvé segmenty jako tvář, přilba, ruce a rukavici na levé ruce. U ležícího hráče, zřejmě vlivem filtrace, došlo ke sloučení tváře s kšiltem přilby a odlesk na přilbě vytvořil segment připomínající oko.



Obrázek 18: Barevné trámy na bílém podstavci

Na obrázku 18 je fotografie několika dřevěných trámů natřených různými barevnými mořidly a posazených na bílém podstavci, pozadí tvoří šedivá podlaha. Při parametrech segmentace nastavených na $\sigma = 0{,}8$ a $k = 450$ dostáváme velmi pěkný výsledek. Šedé pozadí tvoří celistvý segment přerušený pouze několika drobnými komponentami. Bílý podstavec je také tvořen jedním segmentem se správně detekovanými otvory. Většina trámů je segmentována dobře. Žlutý

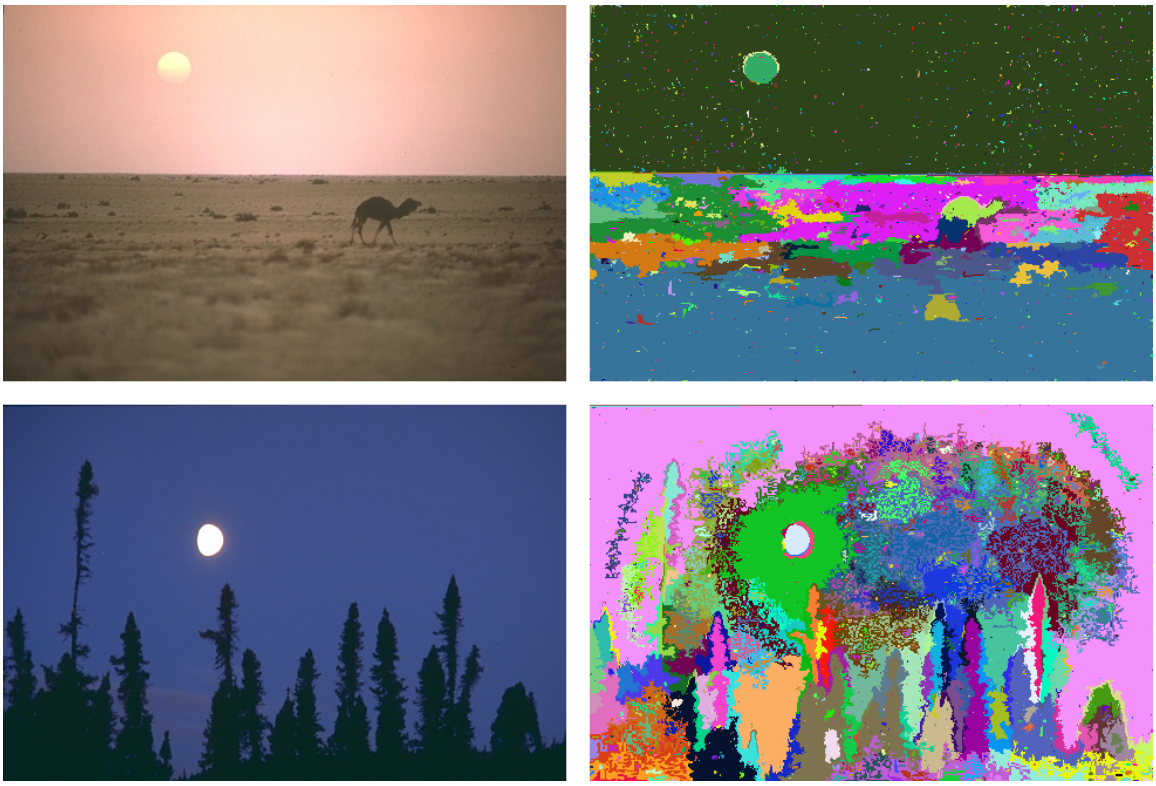
trám má na počátku menší „falešné“ podlouhlé segmenty, které se postupně vytrácejí. Při detailním pohledu na originální fotografii zjistíme, že jsou způsobené několika podélnými rýhami v trámu. U červeného trámu došlo rovněž k detekci „falešných“ segmentů, což tentokrát zapříčinil suk. I přes zjevnou existenci suku v modrém horním trámu nedošlo k výraznějšímu narušení jeho segmentace. Výsledek by bylo možné vylepšit zvětšením parametru k, což by vedlo ke sloučení se sousední hlavní komponentou trámu.



Obrázek 19: Parkoviště a segmentace s různým parametr k, postupně (b) = 150, (c) = 300 a (d) = 450

Na obrázku 19(a) máme fotografii téměř plně obsazeného parkoviště. Další obrázky představují jeho segmentaci se shodným parametrem $\sigma = 0{,}8$ a postupně 19(b) $k = 150$, 19(c) $k = 300$ a 19(d) $k = 450$. V testech se jedná o největší obrázek s rozměry 854×641 pixelů, počet nalezených segmentů byl pro 19(b) 4922, pro 19(c) 3110 a pro 19(d) 2465. Na tomto příkladu jde vidět, jak parametr k ovlivňuje kvalitu segmentace. Při bližším pohledu lze na obrázku 19(b) pozorovat drobné nežádoucí segmenty na cestě a kapotách vozů. Tyto segmenty s vyšší hodnotou parametru k mizí. Ke ztrátě detailů však nedochází a v obrázku 19(d) jde stále vidět segmentaci drobných objektů, jako kanálu na cestě, bílé šipky a bílých dělicích čar, zřetelná je i segmentace

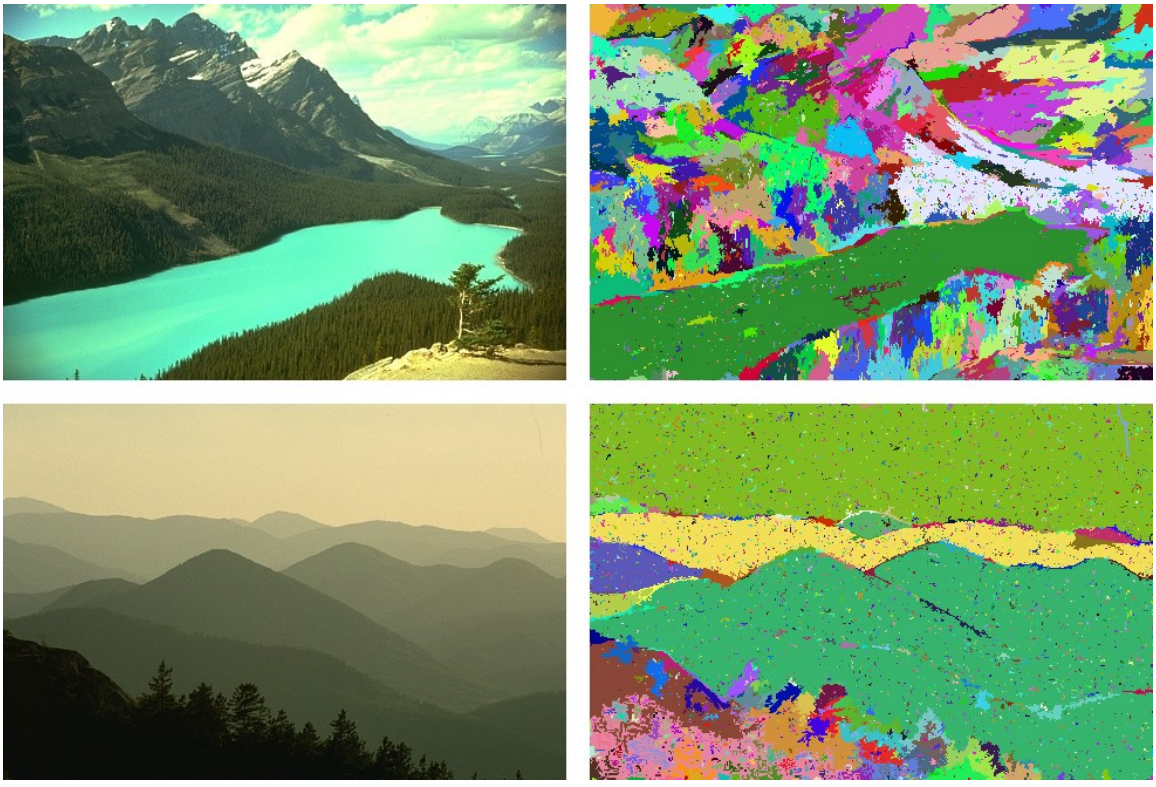
střešních oken aut stojících v první řadě. Velkým celistvým segmentem je také cesta za druhou řadou vozidel. Na druhou stranu je segmentace stále velmi detailní a činí následnou analýzu komplikovanou. V ideálním případě bychom rádi dostali jeden velký segment pro parkoviště a pro každé auto samostatný segment. Změna konstantního parametru k nepovede k cíli, zřejmě by nedošlo ke sloučení všech komponent jednoho auta (odlišná barevnost kapoty, oken a světlometů) a naopak by to mohlo vést ke sloučení komponent dvou sousedních aut s podobnou barvou. K dalším experimentům se nabízí prahová funkce τ. Jak bylo uvedeno v teoretickém textu, lze změnou její definice určovat, které komponenty se mohou sloučit a které ne. Takto by mohly bý sloučeny i vnitřní komponenty oken aut či jejich světlometů s vnější komponentou kapoty auta.



Obrázek 20: Segmentace denní a noční oblohy

Na obrázku 20 vlevo máme fotografie oblohy v různá denní období, vpravo jejich segmentace při nastavených parametrech $\sigma = 0{,}4$ a $k = 450$. Segmentace denní oblohy a to i přes vnímaný přechod od tmavě růžové přes světle růžovou opět k tmavě růžové vytvořil jeden velký segment. Naproti tomu noční obloha, která se jeví jako jednobarevná, byla segmentována na mnoho segmentů. Došlo k tomu jednak tím, že barevný obraz je převáděn na černobílý (prostý průměr všech barevných složek) a pak vlivem výpočtu pro ohodnocování hran a algoritmem samotným. Přestože na denní obloze pozorujeme barevný přechod, rozdílnost sousedních pixelů je zanedbatelná, ohodnocení těchto hran je malé a způsobí sloučení pixelů do jedné komponenty.

Převedením obrazu noční oblohy na stupně jasu a následné porovnání sousedních pixelů (konkrétně oblohy) zapříčiní, že velké množství hran má ohodnocení 0 a pouze v některých případech (což je to důležité) je ohodnocení jen velmi malé číslo. Podobným ohodnocováním „trpí“ i segmentace černých stromů v popředí. Pokud uvážíme algoritmus, resp. vliv prahové funkce, pochopíme k čemu dochází. V algoritmu jsou nejprve zpracovány hrany s nulovým ohodnocením – ty vždy způsobí sloučení komponent. Těch je ale v případě noční oblohy většina a tak vznikne několik komponent s velikým počtem uzlů. Prahová funkce v takovém případě vrací hodnotu limitně se blížící k 0. Díky tomu v momentě, kdy se začíná zpracovávat hrana s ohodnocením 1, ke sloučení velkých komponent již nedochází, jelikož hodnota hrany je větší než vnitřní velikosti obou velikých komponent a tak je hrana algoritmem nesprávně označena za hranici mezi komponentami. Prosté zvýšení hodnoty parametru k by nevedlo ke zlepšení, neboť by došlo k nežádoucímu spojení komponenty měsíce s komponentou oblohy. Proto doporučuji zaměřit další výzkum na princip dynamického chování hodnoty parametru k, které by vedlo k lepší segmentaci velkých téměř stejnobarevných ploch.



Obrázek 21: Fotografie krajin

Na obrázku 21 jsou fotografie krajin a jejich segmentace, které v sobě mají dominantní prvky, resp. objekty. Parametry pro segmentaci jsou shodně nastaveny na $\sigma = 0{,}4$ a $k = 300$. V horním obraze vidíme dominantní vodní plochu, která je algoritmem poměrně správně nalezena a je reprezentována největším segmentem. Ostatní aspekty obrazu jsou rozkouskovány do mnoha

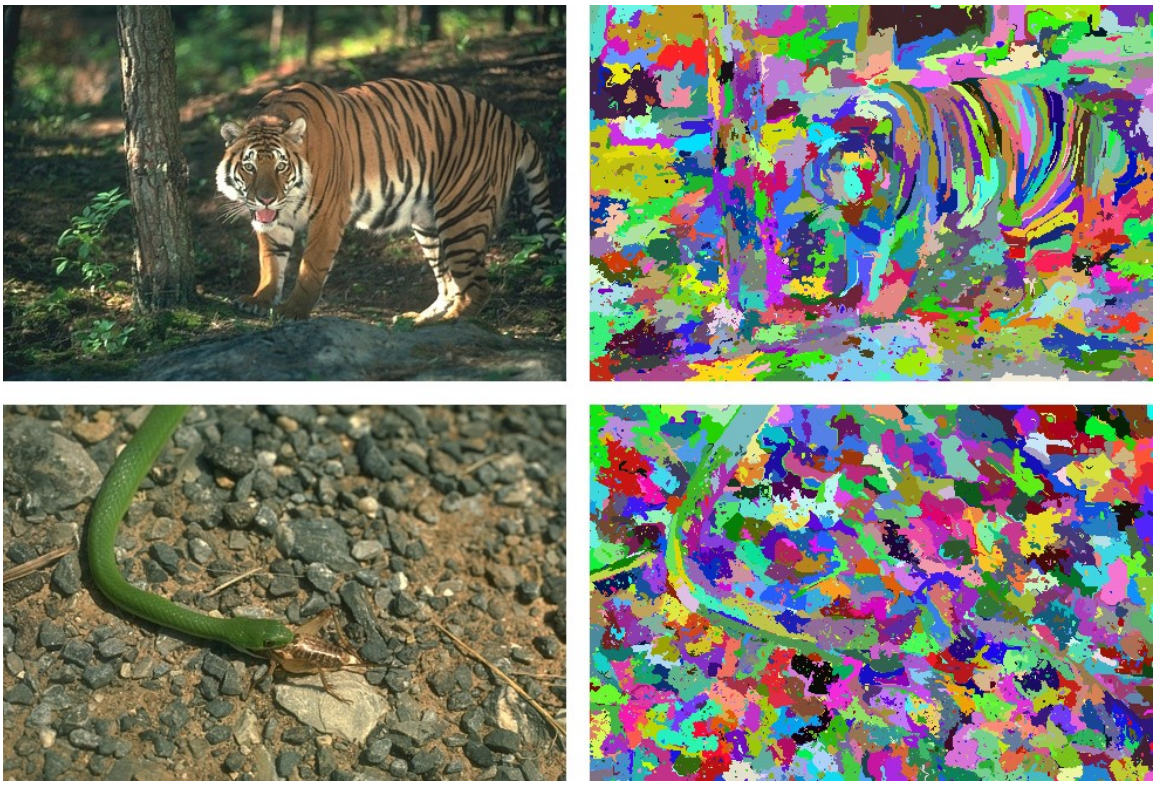
nesourodých segmentů, z nichž celistvějšími jsou jen segmenty oblačnosti. Dolní obraz zachycující kopcovitou krajinu, kdy jednotlivé kopečky mají poměrně stejnou barvu, dává výslednou segmentaci téměř dokonalou. Kvalitu kazí pouze segmentace černého popředí vlevo dole z podobných příčin, jako předchozí příklady. Pro tento případ by nepomohlo jednoduché zvýšení parametru k, které by mělo za následek i nežádoucí slučování komponent některých pohoří. Tento příklad vede k úvaze, zda by nebylo možné parametr k nějakým způsobem dynamicky upravovat, pro různé části obrazu na základě jejich vlastností. Tato úvaha je však nad rámec práce a tak ponecháme na čtenáři, jak s myšlenkou naloží.

Obrázek 22: Medvěd

Na obrázku 22 máme fotografii medvěda s hustě chlupatou srstí, celé pozadí je silně rozmazané, vysoká tráva v popředí, v dolní části obrazu je rozostřena jemně a některé její listy a stonky jsou zřetelné. Parametry segmentace jsou $\sigma = 0{,}4$ a $k = 450$. Výsledek je překvapivě dobrý, největší segment tvoří celé pozadí (louka, vodní plocha a tmavý les), s malou disharmonií v horní části obrazu, která opět ukazuje na problematické zpracování tmavých ploch. Algoritmus rovněž generoval správné segmenty pro trávu v popředí, všimněte si převážně podlouhlých segmentů odpovídajícím vysokému lučnímu býlí. Segmentace samotného medvěda vypadá subjektivně dobře, zejména lze vyzdvihnout segmenty odpovídající nosu, čumáku spolu s tlamou, vnitřní

strana uší a detekována je i světlejší srst na zádech. Najdeme i drobné chyby, například algoritmus při tomto nastavení nesprávně segmentoval obě oči, když je sloučil s oblastí lícních kostí.

Pokud bychom požadovali hrubější segmentaci pouze medvěd a pozadí, mohli bychom toho docílit zvýšením parametru k.

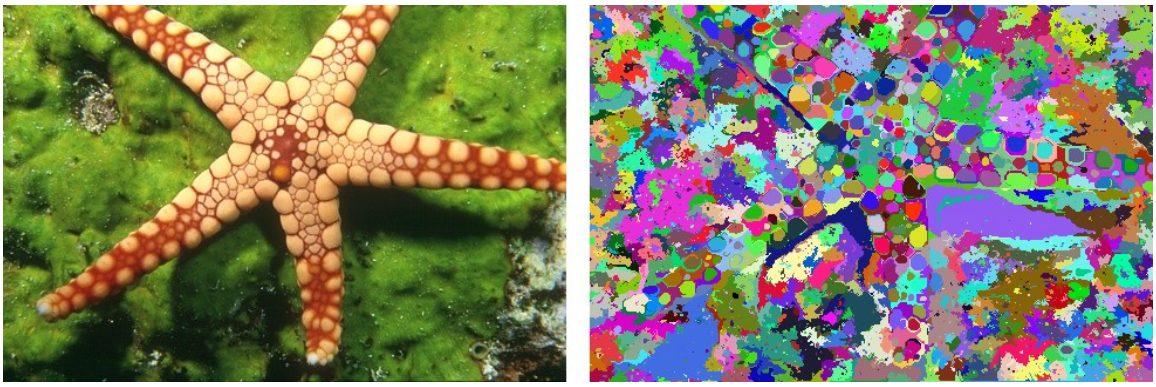


Obrázek 23: Divoká příroda

Na obrázku 23 máme vlevo fotografie z přírody, tentokrát však není pozadí kolem objektu tolik rozmazané a skládá se z mnoha drobných částí, vpravo je jejich segmentace při nastavených parametrech $\sigma = 0{,}4$ a $k = 300$. Lze pozorovat, že algoritmus se pro segmentování takových obrazů příliš nehodí. Při troše fantazie v segmentovaných obrazech můžeme vidět jak tygra a vedle stojící strom na prvém, tak hada na druhém obrázku. Objekty jsou však složeny z velkého počtu segmentů.

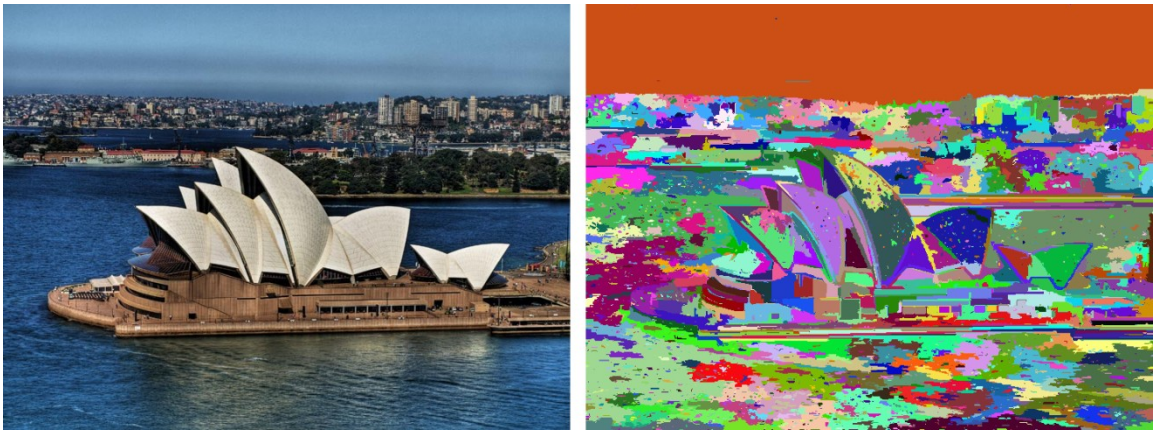
Při pohledu na fotografii hada si můžeme povšimnout faktu, že had je zelený a ostatní pozadí je šedivé a hnědožluté. Pokud bychom zpracovávali tento barevný obraz tak, že bychom při ohodnocování rozdílnosti pixelů vyzdvihli pouze zelenou barvu vůči všem ostatním, dosáhli bychom lepšího výsledku. Tento experiment nebyl z časových důvodů uskutečněn.

Ještě jeden záběr z přírody, jehož výsledek je podobný předešlým příkladům. Na obrázku 24 je fotografie červenožluté hvězdice na zeleném pozadí, pro segmentaci jsme nastavili parametry $\sigma = 0{,}4$ a $k = 150$. Ve výsledné segmentaci lze poměrně snadno vidět segmenty odpovídající žlutým přísavkám hvězdice. Nezanedbatelným je i největší segment představující stín pod hvězdicí. Zvýšení parametru k nevedlo k výrazně lepším výsledkům a tak i zde by bylo možné dosáhnout lepší segmentace, pokud bychom v algoritmu ohodnocování hran uvažovali červenožlutou barvu hvězdice versus sytě zelené pozadí.



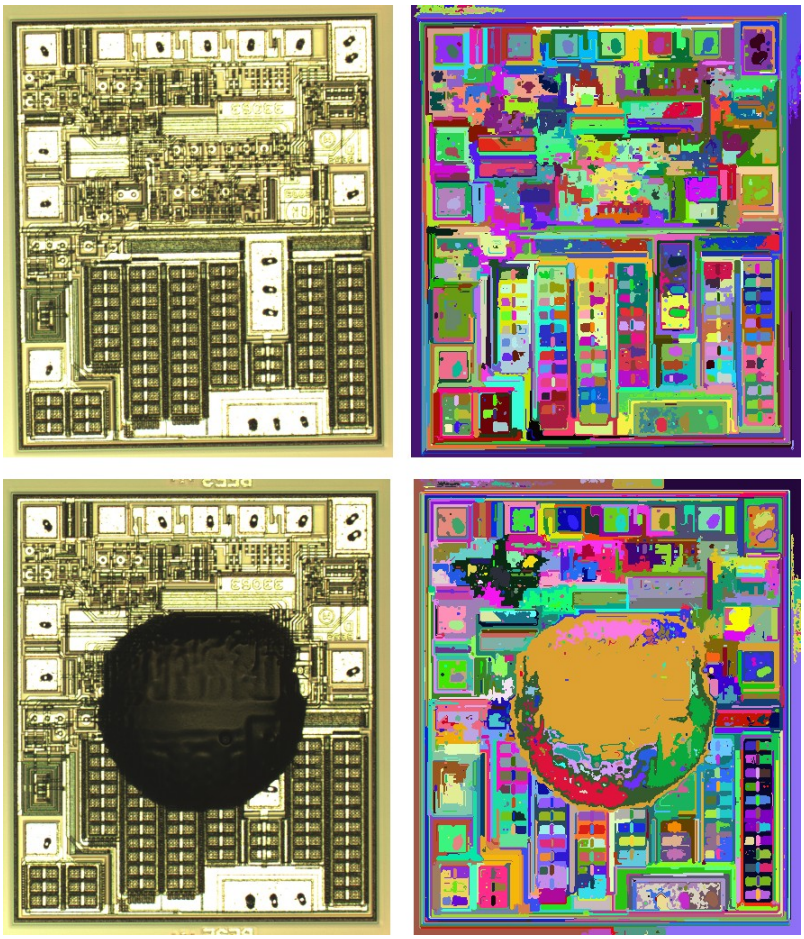
Obrázek 24: Hvězdice na mořském dně

Na obrázku 25 máme fotografii budovy opery v Sydney, vpravo je segmentace při nastavených parametrech $\sigma = 0{,}8$ a $k = 450$. Největší segment v horní části představuje oblačnou oblohu. Zřejmě druhým největším segmentem je oceán za Operou v pravé střední části obrazu. Segmentace budovy Opery je vcelku zdařilá, jsou patrné jednak segmenty typické střešní konstrukce, tak i segmenty budovy samotné věrně vyjadřují její jednotlivé části a dominantní prvky. Pro segmentaci oceánu platí to, co již bylo mnohokrát řečeno.



Obrázek 25: Budova opery v Sydney

A na závěr jeden příklad z praxe. Na obrázku 26 máme dvě fotografie elektronických čipů na křemíkové desce, pořízených kamerou připojenou k mikroskopu. V závěrečné fázi výrobního procesu se čipy, které nevyhovují parametrům kvality a požadavkům na elektrické vlastnosti, označí kapkou černého inkoustu (značka pro vadný čip). Jeden z úkolů bylo dodat SW pro automatickou kontrolu, zda byly správně označeny kapkou pouze vadné čipy a ostatní dobré čipy zůstaly čisté. Tato aplikace pak analyzovala pořízené fotografie. Tehdy jsem pro řešení použil odlišných algoritmů, které vracely návratovou hodnotu ANO je kapka, NE není kapka. Zkusíme nyní námi implementovaný grafový algoritmus a parametry segmentace nastavíme na $\sigma = 0{,}8$ a $k = 450$. Je zřejmé, že ve výsledné segmentaci čipu s kapkou lze pozorovat výskyt velkého segmentu uprostřed představujícího kapku samotnou. Taková komponenta však v segmentaci obrazu čipu bez kapky zjevně chybí. Pokud bychom navíc pro následnou analýzu využili již existující reprezentaci grafem, vyhledat největší komponentu (či komponenty) a prozkoumat její vlastnosti by bylo mnohem snadnější. Podle naměřených časů, bychom při použití našeho algoritmu zřejmě dosáhli rychlejší odpovědi, než v původní aplikaci. To však nebylo dokázáno, neboť algoritmus analýzy není předmětem této práce.



Obrázek 26: Neoznačený a označený elektronický čip na křemíkové desce

7 Závěr

V úvodu textu jsem uvedl, co je to analýza obrazu a jakou roli hraje segmentace obrazu. Popsal jsem několik klasických metod k řešení tohoto problému. Pak jsem se zaměřil na kategorii algoritmů založených na grafových algoritmech a jako reprezentanta z široké škály jsem vybral algoritmus Efektivní segmentace obrazu založená na teorii grafů od Pedra F. Felzenszwalba a Daniela P. Huttenlochera [1], se kterým jsem čtenáře seznámil detailněji.

V implementačních poznámkách jsem naznačil, jak jsem k algoritmu přistoupil. Z nabízených možností jsem vybral reprezentaci obrazu mřížkovým grafem, kde jsou sousední pixely spojeny hranou ve všech osmi směrech. Ohodnocení hrany jsem definoval jako rozdíl intenzity jasu neboli rozdíl stupňů odstínu šedi. Pro barevné obrazy jsem zvolil převod na intenzitu jasu jako průměrnou hodnotu všech jejich barevných složek. Tímto způsobem jsem implementoval algoritmus, který má lineární prostorovou složitost $O(n)$ a „Ackermanovskou“ časovou složitost $O(n \propto(n))$, kde $\propto$ je velmi pomalu rostoucí inverzní Ackermanovou funkcí a n je počet pixelů.

Při testování jsem algoritmus podrobil obrazům zachycujícím různé aspekty našeho světa. Časová výkonnost algoritmu je dobrá a pro obrazy s počtem pixelů v řádech stovek tisíců je metoda vhodná i pro zpracování video signálu. Výsledné segmentace obrazů, kde jsou objekty v kontrastu k pozadí, jsou přesné. Pro segmentaci fotografií krajin a přírody se takto implementovaná metoda hodí méně a opravdu záleží na konkrétním případu. Na závěr jsem ukázal použití algoritmu na fotografiích z praxe a srovnání výsledků s existující průmyslovou aplikací.

Pro dosažení lepších výsledků segmentace je vhodné zaměřit větší pozornost a další výzkum na části algoritmu jako předzpracování obrazu, alternativní výpočty pro ohodnocování hran, dynamické změně parametru k, nebo různé definici prahové funkce τ. Použitím důmyslnějších filtrů, které by lépe aproximovaly tmavé plochy, či barevné přechody, pomůže algoritmu generovat mnohem lepší výslednou segmentaci.

S ohledem na velký rozmach výzkumu v oblasti obecných grafových algoritmů a komplexních sítích doporučuji tuto oblast nadále sledovat. Vhodnost použití algoritmů pro úkol segmentace obrazu byla prací ověřena. Vzhledem ke vzrůstajícím velikostem obrazu ať statických (dnes běžné desítky miliónů pixelů), či pohyblivých (např. HD rozlišení 1920x1080) může představovat zpracování obrazu pomocí grafu stále efektivní metodu s ohledem na její časovou složitost.

Literatura

[1] FELZENSWALB, Pedro F., HUTTENLOCHER, Daniel P. *Efficient Graph-Based Image Segmentation*. 2004.

[2] ZAHN, Charles T. *Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters*. IEEE Transactions on Computing, 1971.

[3] SHI, Jianbo, MALIK, Jitendra. *Normalized Cuts and Image Segmentation*. IEEE Transactions on Pattern Analysis and Machine Inteligence, 2000.

[4] BOYKOV, Yuri, FUNKA-LEA, Gareth. *Graph Cuts and Efficient N-D Image Segmentation*. 2004.

[5] GRADY, Leo. *Random Walk for Image Segmentation*. IEEE Transactions on Pattern Analysis and Machine Inteligence, 2006.

[6] *The Berkley Segmentation Dataset and Benchmark*. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/.

[7] SOJKA, Eduard, GAURA, Jan, KRUMNIKL, Michal. *Matematické základy digitálního zpracování* obrazu. VŠB-TU Ostrava, 2011.

[8] OCHODKOVÁ, Eliška. *Grafové algoritmy*. VŠB-TU Ostrava, 2003.

[9] VEČERKA, Arnošt. *Grafy a grafové algoritmy*. UP Olomouc, 2007.

[10] ŠPANĚL, Michal, BERAN, Vítězslav. *Obrazové segmentační techniky*. VUT Brno, 2006. http://www.fit.vutbr.cz/~spanel/segmentace/