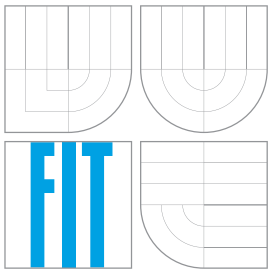
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

HLEDÁNÍ OBJEKTU VE SNÍMCÍCH

SEARCHING FOR OBJECTS IN PICTURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MATÚŠ MOTLÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. FRANTIŠEK V. ZBOŘIL, CSc.

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Motlík Matúš**
Obor: Informační technologie
Téma: **Hledání objektu ve snímcích**
Searching for Objects in Pictures
Kategorie: Umělá inteligence

Pokyny:

1. Prostudujte zadanou literaturu.
2. Navrhněte program pro hledání jednoduchých objektů v černobílých snímcích.
3. Navržený program implementujte.
4. Proveďte experimenty spočívající v hledání zadaného objektu ve skupině nepřekrývajících se objektů.
5. Zhodnoťte dosažené výsledky.

Literatura:

- Davies, E. R.: Machine Vision, Elsevier, 2005, ISBN-13-9780-12-206093-9
- Digital Image Processing, Springer, 2008, ISBN-978-1-84628-379-6
- Nedoma, D.: Rozpoznávání objektů v obrazech, bakalářská práce, FIT VUT v Brně, 2013

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zbořil František V., doc. Ing., CSc.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Táto bakalárska práca sa zaoberá riešením vyhľadávania jednoduchých objektov v obraze. Cieľom bolo vytvoriť program schopný vyhľadávať jednoduché objekty v čiernobielych snímkoch. Postupne sú opísané jednotlivé kroky spracovania snímkov. V práci je stručne popísané predspracovanie obrazu, segmentácia, popis segmentovaného obrazu, klasifikácia a vyhľadávanie objektov v obraze.

Abstract

This bachelor's thesis deals with solving the problem of searching simple objects in pictures. The aim was to create a program that will be able to search simple objects in grayscale images. In this theses are described progressively steps of image processing. There are described preprocessing of image, segmentation, description of segmented data, classification and searching for objects in image.

Kľúčové slová

vyhľadávanie objektov v obraze, spracovávanie obrazu, segmentácia, prahovanie, adaptívne prahovanie, popis segmentovaných dát, dvoj-priechodový algoritmus, príznaky objektov, momenty, klasifikácia

Keywords

searching for objects in pictures, image processing, segmentation, thresholding, adaptive thresholding, description of segmented data, two-pass algorithm, object features, moments, classification

Citácia

MOTLÍK, Matúš. *Hledání objektu ve snímcích.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Zbořil František.

Hledání objektu ve snímcích

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Doc. Ing. Františka V. Zbořila, CSc. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Matúš Motlík
17. mája 2016

Poďakovanie

Chcel by som sa poďakovať pánovi Doc. Ing. Františkovi V. Zbořilovi, CSc. za odborné vedenie práce a cenné rady, návrhy a pripomienky týkajúce sa práce.

Obsah

Kapitola 1

Úvod

Počítačové videnie, detekcia objektov v obraze a spracovanie obrazu sa už niekoľko desaťročí aktívne rozvíja. Svoje uplatnenie nachádza v rôznych oblastiach. Príkladom môže byť detekcia tvárí v obraze, s ktorou sa môžeme bežne stretnúť vo fotoaparátoch a kamerách, prípadne na sociálnych sieťach a pod. Medzi ďalšie príklady využitia počítačového videnia patrí analýza a sledovanie dopravy. V tomto prípade sa môžu sledovať dopravné značky, chodci či ostatné vozidlá v premávke aby sa predišlo prípadným kolíziám a pod. V medicíne je možné použiť detekciu objektov v obraze pri spracovávaní medicínskych snímkov. Táto analýza snímkov môže pomôcť s diagnózou a liečbou rôznych zdravotných problémov. Počítačové videnie sa hojne využíva v priemysle, napríklad pri inšpekcii výrobkov na výrobnom páse, detekcii chýb a nedostatkov atď.

Spracovávanie obrazu bežne pozostáva z nasledujúcich krokov: Obraz je zachytený senzorom (kamera, fotoaparát a pod.) a následne je digitalizovaný. Pri digitalizácií obrazu môžu v obraze vzniknúť chyby ako napríklad šum, preto je snaha čo najviac eliminovať tieto negatívne javy. Táto fáza spracovania obrazu sa nazýva predspracovanie obrazu. Následne je potrebné oddeliť jednotlivé objekty od pozadia, prípadne od ostatných objektov. Tento proces sa nazýva segmentácia obrazu. Medzi najjednoduchšie techniky segmentácie obrazu patrí prahovanie, ktoré sa dá použiť na jednoduché problémy, akým je napríklad oddelenie neprekrývajúcich sa objektov od pozadia, prípadne rozpoznávanie znakov v tlačenom texte. Pre komplikovanejšie úlohy je potrebné použiť sofistikovanejšie techniky, príkladom týchto metód je napríklad detekcia hrán v obraze. Posledným štádiom spracovania obrazu je popis jednotlivých objektov a následná klasifikácia objektov do klasifikačných tried.

Cieľom tejto práce je vytvoriť program na vyhľadávanie jednoduchých objektov v čiernobielych snímkoch. Tento program by mal byť schopný vyhľadať jednotlivé objekty, určiť ich pozíciu v obraze, prípadne vypočítať ďalšie vlastnosti nájdených objektov ako napríklad ich ťažisko, celkovú plochu a pod. Aplikácia by mala umožňovať načítať trénovaciu sadu základných objektov, ktoré bude možné následne vyhľadávať v rôznych snímkoch. Mala by byť schopná úspešne segmentovať objekty od pozadia, vhodným spôsobom popísať ich vlastnosti na základe ktorých budú jednotlivé objekty klasifikované do tried určených trénovacou sadou. Funkčnosť implementovanej aplikácie by mala byť otestovaná na skupine snímkov neprekrývajúcich sa objektov.

Nasledujúca kapitola objasňuje problémy spojené s predspracovaním obrazu. Ďalšia kapitola sa venuje rôznym technikám segmentácie obrazu, teda oddelenia jednotlivých objektov od pozadia. Kapitola 4 sa venuje identifikácie jednotlivých objektov v segmentovanom obraze a následne sú popísané spôsoby získania rôznych príznakov týchto objektov. V ďalšej kapitole sú popísané možné spôsoby rozpoznávania jednotlivých objektov v obraze. Nasle-

dujúce dve kapitoly popisujú samotný návrh a implementáciu aplikácie na vyhľadávanie objektov v čiernobielom obraze. Nakoniec sú v poslednej kapitole popísané základné testy, ktorými bola aplikácia otestovaná.

Kapitola 2

Predspracovanie obrazu

Predspracovanie obrazu má za úlohu pripraviť obraz na následnú segmentáciu, tzn. eliminovať čo najviac chýb v obraze. Chyby v obraze vznikajú digitalizáciou obrazu. Príkladom takýchto chýb sú šum, skreslenie a iné. Existuje viacero metód na eliminovanie chýb v obraze, napríklad lineárne a nelineárne filtre, prípadne bodové operácie. Hlavný rozdiel medzi filtrami a bodovými operáciami je, že pri použití filtra sa na výpočet novej hodnoty pixelu berú do úvahy hodnoty jeho okolitých pixelov. [1].

V niektorých prípadoch môže byť farba v obraze redundantnou informáciou, preto je vhodné previesť obraz na čiernobiely. Hodnota jasu I daného pixelu sa z jeho farebných zložiek vypočíta podľa vzťahu [1]:

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

kde R, G a B označujú jednotlivé farebné zložky – červenú, zelenú a modrú. Táto rovnica berie v úvahu citlivosť oka na jednotlivé farebné zložky.

2.1 Lineárny filter

Lineárne filtre používajú na výpočet novej hodnoty pixelu hodnoty jeho okolitých pixelov v lineárnej funkcii, napríklad ako vážený priemer. Efekt lineárneho filtru je jasne definovaný hodnotami koeficientov v jeho matici, napríklad:

$$H = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.2}$$

Matica filtru definuje počet susedných pixelov a ich váhu akou sa podieľajú na novej hodnote pixelu. Formálne sa výpočet novej hodnoty pixelu použitím matice H dá vyjadriť vzťahom [1]:

$$f(x,y) = \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} f(x+i, y+j) * H(i,j) \tag{2.3}$$

Podľa toho, aké hodnoty nadobúda matica filtru, môžeme ich rozdeliť na vyhladzovacie (smoothing), tie majú všetky koeficienty kladné, a rozdielové (difference) filtre, kde niektoré ich koeficienty sú záporné. Medzi prvý typ filtrov patria box filter a Gaussov filter, k rozdielovým sa dá zaradiť Laplaceov filter. Značnou nevýhodou použitia lineárnych filtrov na odstránenie šumu je rozmazanie celého obrazu rovnomerne a teda napríklad aj hrán, čo môže mať negatívny dopad pri následnej segmentácií [1].

2.2 Nelineárny filter

Na rozdiel od lineárnych filtrov, je nová hodnota pixelu vypočítaná pomocou nelineárnej funkcie. Najjednoduchším nelineárnym filtrom je minimum a maximum filter. V tomto prípade je nová hodnota nastavená na hodnotu minima, prípadne maxima okolitých pixelov. Ďalším príkladom je mediánový filter. Hodnota každého pixelu je v tomto prípade určená ako medián hodnôt susedných pixelov. Tento filter eliminuje veľmi malé štruktúry v obraze (menšie ako polovica veľkosti filtra), avšak všetky ostatné štruktúry ostávajú prevažne nezmenené. Veľmi podobným príkladom nelineárneho filtra je vážený mediánový filter, ktorý priradí individuálne váhy okolitým pixelom [1].

Kapitola 3

Segmentácia

Segmentácia je proces rozdelenia obrazu na časti, ktoré sú významné pre špecifický problém – objekty a pozadie obrazu. Segmentáciu do značnej miery ovplyvňujú vstupné dáta. Negatívne vplývajú javy ako šum alebo nerovnomerné rozloženie jasu v obraze. Segmentačné metódy sa rozdeľujú na dve hlavné skupiny – metódy založené na oblastiach, kde sa detegujú podobné vlastnosti regiónov, a metódy založené na detekcii hrán, kde sa detegujú náhle zmeny jasu v obraze (hrany) [4].

Výsledok segmentácie je zjednodušený binárny obraz, ktorý jasne oddeľuje objekty od pozadia a pri tom zachováva ich veľkosť a tvar. Pixely pozadia majú zvyčajne hodnotu 0, pixely objektov hodnotu 1.

3.1 Segmentácia prahovaním

V prípade, že pozadie v obraze je prevažne jednotné a objekty sú v kontraste s pozadím, segmentácia sa dá jednoducho dosiahnuť pomocou prahovania. Je to výpočtovo nenáročná a rýchla metóda, využívaná hlavne tam, kde sa dajú zaručiť vhodné podmienky obrazu. Prahovanie je transformácia vstupného obrazu f na binárny obraz g [7]:

$$g(i,j) = \begin{cases} 1 & \text{pre } f(i,j) \geq T \\ 0 & \text{pre } f(i,j) < T \end{cases} \tag{3.1}$$

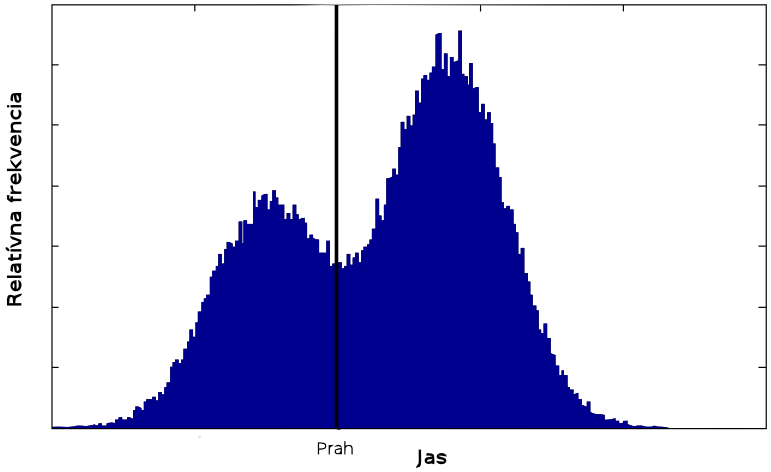
kde T je hodnota prahu, $g(i,j) = 1$ pre hodnoty objektov, $g(i,j) = 0$ pre hodnoty pozadia (prípadne opačne). Správne nájdenie hodnoty prahu je kritické pre úspešnú segmentáciu prahovaním.

3.1.1 Nájdenie vhodného prahu

Pri vhodných podmienkach, akými sú nepatrný šum, konštantné rozloženie jasu objektov a pozadia sa dá určiť pevná hodnota prahu, podľa ktorej sa následne prahuje vstupný obraz. Táto technika sa využíva vtedy, ak je možné garantovať dané podmienky.

Najviac používanou technikou je však analýza histogramu intenzít v obraze. V prípade, že je nájdené jednoznačné lokálne minimum (viď 3.1), toto minimum je interpretované ako hodnota prahu. Hodnoty napravo od tohto minima korešpondujú s tmavými objektmi, hodnoty naľavo reprezentujú svetlé pozadie (príp. naopak). Pri použití tejto metódy však môžu nastať nasledujúce problémy [3]:

1. Údolie môže byť príliš široké a preto je náročné zistiť jednoznačné minimum.



Obr. 3.1: Histogram intenzít v obraze, k označuje prah, prevzaté z [6]

2. V histograme sa nachádza viac lokálnych miním a preto môže byť ťažké určiť ktoré je najvýznamnejšie.
3. Šum v údolí histogramu môže brániť nájdeniu optimálneho prahu.
4. Údolie nie je rozpoznateľné kvôli značnému šumu, alebo nerovnomernému jasu pozadia v obraze.
5. Jeden z vrcholov v histograme môže byť výrazne väčší než druhý, čo ovplyvní polohu minima.
6. Histogram môže mať viac lokálnych maxím, vďaka čomu je náročné určiť polohu relevantného prahu.

Medzi ďalšie postupy zisťovania prahu patrí metóda Otsu pomenovaná po jej objaviteľovi Nobuyuki Otsu. V tejto metóde sa iteratívne hľadá prah, podľa ktorého je histogram rozdelený na dve časti tak, aby boli optimalizované kritéria založené na medzitriednom rozptyle (between-class variance) hodnôt histogramu [3].

Uvažujme šedo-tónový obraz, ktorý má L odtieňov šedej. Počet pixelov, ktoré majú odtieň šedej i je označený ako n_i. Celkový počet pixelov je potom $N = n_1 + n_2 + \cdots + n_L$. Pravdepodobnosť, že pixel ma odtieň i je [3]:

$$p_i = \frac{n_i}{N} \tag{3.2}$$

Pre rozsahy intenzít nad a pod hodnotou prahu k, môžeme vypočítať medzitriedny rozptyl σ_B^2 podľa vzťahu [3]:

$$\sigma_B^2 = \pi_0 \pi_1 (\mu_1 - \mu_0)^2, \tag{3.3}$$

kde

$$\pi_0 = \sum_{i=1}^{k} p_i \qquad \pi_1 = \sum_{i=k+1}^{L} p_i = 1 - \pi_0 \tag{3.4}$$

$$\mu_0 = \sum_{i=1}^{k} \frac{i p_i}{\pi_0} \qquad \mu_1 = \sum_{i=k+1}^{L} \frac{i p_i}{\pi_1} \tag{3.5}$$

Pre nájdenie vhodného prahu k musí byť vypočítaná hodnota medzitriedneho rozptylu pre všetky možné hodnoty prahu v histograme. Výsledný prah je následne určený maximálnou hodnotou medzitriedneho rozptylu σ_B^2 [5].

3.1.2 Používane metódy

Výber metódy prahovania je pre správnu segmentáciu kritický. Globálne prahovanie, kedy je použitý jeden prah, podľa ktorého sa segmentuje celý obraz, sa dá použiť iba za veľmi špecifických podmienok, akými sú jednotné nasvietenie scény, nepatrný šum, tiene objektov a pod. Niekedy používanou technikou je tzv. poloprahovanie, ktoré odstraní pozadie obrazu, pritom objekty zostanú neporušené [7].

$$g(i,j) = \begin{cases} f(i,j) & \text{pre } f(i,j) \geq T \\ 0 & \text{pre } f(i,j) < T \end{cases} \tag{3.6}$$

V prípade, že scéna je osvetlená nerovnomerne, používa sa tzv. adaptívne prahovanie. Hodnota prahu sa v tomto prípade mení dynamicky v jednotlivých častiach obrazu. Existujú rôzne spôsoby adaptívneho prahovania, napríklad metóda *Chow a Kaneko* [3]. Ďalším postupom je tzv. lokálne prahovanie [3]. V tomto prípade sa pre každý pixel v obraze analyzujú hodnoty intenzít susedných pixelov pre získanie optimálneho prahu. Hodnota prahu sa môže určiť rôznymi spôsobmi, napríklad ako priemer intenzít okolitých pixelov prípadne ako medián hodnôt susedných pixelov. Tieto metódy však pracujú správne iba v prípade, že veľkosť okolia použitého pre výpočet príslušného prahu je aspoň tak veľká, aby bolo možné zachytiť dostatočné množstvo pixelov pozadia a objektov.

3.2 Iné segmentačné metódy

Segmentácia založená na detekcii hrán

Segmentácia založená na detekcii hrán predstavuje množstvo metód, ktorých základ tvoria informácie o hranách. Výsledkom tejto segmentácie je binárny obraz zobrazujúci hrany jednotlivých objektov. Hrana je miesto v obraze, kde dochádza k výraznej lokálnej zmene intenzity. Tieto zmeny intenzity zvyčajne vznikajú na hranici dvoch objektov. Pre nájdenie týchto zmien v obraze sa používajú hranové operátory. Príkladom týchto operátorov sú Sobelov, Prewittov, Robinsonov, Laplaceov gradientný operátor [7].

Segmentácia narastaním oblastí

Jednotlivé body obrazu, ktoré majú podobnú intenzitu, prípadne inú vhodnú vlastnosť, sú postupne zlučované do väčších oblastí. Je nutné definovať pravidlá, kedy môžu byť susedné pixely zlúčené do jednej oblasti. Príkladom týchto techník sú [7]:

- Spojovanie oblastí
- Spojovanie a štiepenie oblastí
- Zaplavovanie oblastí

Tieto techniky sú odolnejšie voči šumu. Nevýhodou môže byť ich vyššia výpočtová náročnosť. Zvyčajne pracujú iteratívne, jednotlivé pixely sú posudzované viac krát a v priebehu výpočtu sa postupne môžu meniť hypotézy ohľadom pixelov ktoré patria do jednotlivých oblastí [3].

Kapitola 4

Popis segmentovaných oblastí

Napriek tomu, že binárne obrazy obsahujú omnoho menej informácií ako ich čiernobiele varianty, zachovávajú tvar a veľkosť objektov (príp. hranice objektov). Rozpoznávanie jednotlivých objektov v obraze a získanie presného popisu ich tvaru, vo vhodnej forme pre klasifikátor, je dôležitým krokom k pochopeniu obrazových dát. Tento popis by mal generovať číselný vektor, alebo nečíselnú slovnú charakteristiku, ktorá jasne popisuje vlastnosti daného objektu. K získaniu takéhoto popisu je často nevyhnutné identifikovať jednotlivé objekty v binárnom obraze. Následne je vhodné odstrániť nežiadúce objekty, ako napríklad menšie alebo väčšie objekty než určený limit, neúplné objekty na hranici obrazu, a pod.

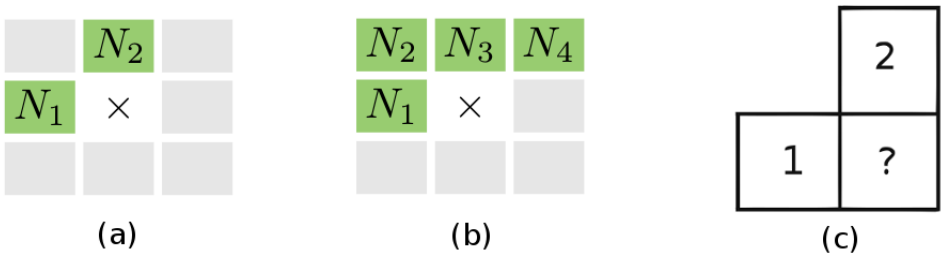
4.1 Identifikácia oblastí

Identifikácia oblastí je nevyhnutným krokom pre získanie ich popisu. Jedným z mnohých prístupov je označiť každý objekt jedinečným číselným indexom. Celkový počet objektov v obraze potom zvyčajne určuje najvyššie použitý index. Je nutné definovať, kedy jednotlivé pixely tvoria jeden objekt(tzn. sú susedné), a kedy oba patria do dvoch disjunktných oblastí. Typicky sa používajú dve definície – 4-okolie a 8-okolie.

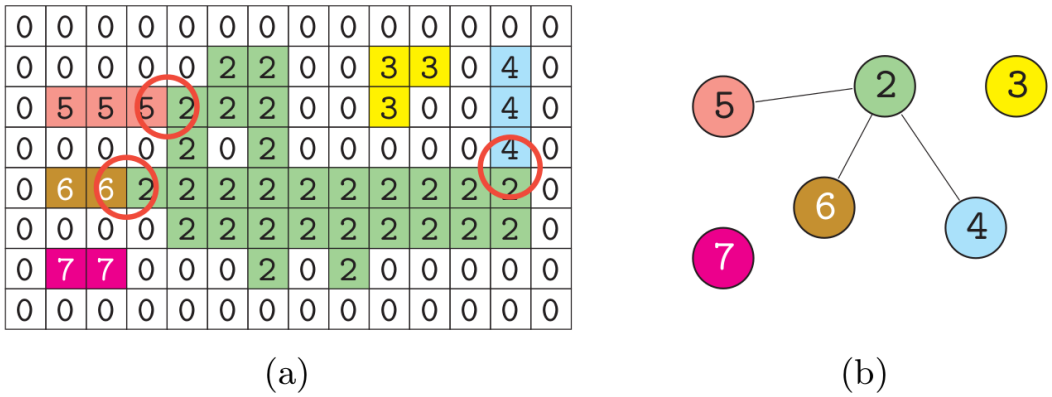
Algoritmus 4.1.1. popisuje sekvenčný prístup k označeniu objektov v segmentovanom obraze. Vstupom algoritmu je binárny obraz, príp. šedo-tónový obraz, kde pozadie je reprezentované nulovými pixelmi a objekty majú nenulové hodnoty pixelov. [7].

1. Prejdi celý obraz riadok po riadku a nenulovým hodnotám pixelov $f(i, j)$ priraď číselný identifikátor v. Hodnota v je vybratá v závislosti na hodnotách identifikátorov susedných pixelov podľa definovaného okolia pixelu (viď 4.1). Hodnoty pixelov ležiacich mimo obrázok sa neuvažujú.
   (a) V prípade, že všetky susedné pixely sú pixely pozadia (tzn. majú hodnotu nula), pixelu $f(i, j)$ je priradený nový identifikátor v.
   (b) Ak práve jeden susedný pixel má priradený identifikátor, jeho hodnota sa priradí aj pixelu $f(i, j)$.
   (c) Ak je viac susedných pixelov s priradeným identifikátorom, priraď ktorýkoľvek z nich pixelu $f(i, j)$. V prípade, že sa identifikátory susedných pixelov líšia, tzn. nastala kolízia identifikátorov (label collision) znázornená na obrázku 4.1, ulož tento pár ako ekvivalentný. Kolizné páry sú uložené v oddelenej datovej štruktúre, tzv. tabuľke ekvivalentov.

2. Všetkým nenulovým pixelom boli priradené identifikátory. Niektoré objekty ale obsahujú pixely s rôznymi hodnotami identifikátorov, preto je celý obraz prejdený znova a pixely sú preznačené podľa informácií z tabuľky ekvivalentov.



Obr. 4.1: 4-okolie (a), 8-okolie (b) a kolízia identifikátorov susedných pixelov (c), prevzaté z [1]



Obr. 4.2: Dočasný stav obrazu po prvom kroku algoritmu (a). V kružniciach sú znázornené kolízie hodnôt identifikátorov pixelov. Uzly grafu (b) naznačujú ekvivalentné hodnoty identifikátorov, ktoré tvoria jeden objekt. Prevzaté z [1]

4.2 Príznaky

Príznakom objektu je myslená špecifická číselná alebo kvalitatívna hodnota, vypočítaná z hodnôt a súradníc pixelov daného objektu. Medzi najjednoduchšie príznaky patrí plocha objektu – teda počet pixelov, ktoré tvoria popisovaný objekt. Jeden príznak pre presný popis objektov väčšinou nestačí, preto sa používajú rôzne príznaky kombinované do tzv. *vektoru príznakov*. Tieto vektory príznakov sa dajú zobraziť ako body v tzv. *priestore príznakov* (feature space). Pre n príznakov je tento priestor n-dimenzionálny. Objekty z jednej triedy by sa mali zhlukovať dokopy a zároveň by mali byť jasne oddeliteľné od ostatných tried. V následnom procese klasifikácie je cieľom priradiť každý vektor do jednej triedy. Vo všeobecnosti by príznaky mali mať určité vlastnosti [4]:

- **Robustnosť** – mali by byť invariantné voči translácií, rotácií a zmene miery.
- **Diskriminabilita** – rozsahy hodnôt objektov iných tried by mali byť rozdielne a pokiaľ možno izolované a neprelínajúce sa.
- **Spoľahlivosť** – všetky objekty jednej triedy by mali mať podobne hodnoty.
- **Nezávislosť** – hodnoty príznakov by mali byť navzájom nezávislé.

Výber vhodných príznakov záleží hlavne na danom probléme, avšak vo všeobecnosti je cieľom čo najviac zredukovať počet príznakov. Problémom je, že s rastúcim počtom príznakom, rastie dimenzia príznakového priestoru a tým sa výrazne komplikuje následná klasifikácia, či už z pohľadu časovej alebo pamäťovej náročnosti. Cieľom návrhu je teda vybrať minimálny počet „dobrých“ príznakov. Jedným z postupov je na začiatku vybrať väčší počet príznakov než je nevyhnutné a následne zredukovať ich počet použitím [4]:

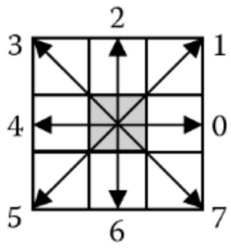
- **Selekcie príznakov** (feature selection) – vyberie sa podmnožina čo najviac informatívnych príznakov a teda redundantné príznaky sa odstránia.
- **Extrakcie príznakov** (feature extraction) – existujúci vektor sa pretransformuje na menší vektor, obsahujúci nové príznaky ktoré vznikli kombináciou pôvodných. Najviac používanými metódami extrakcie príznakov sú *Principal Components Analysis (PCA)* a *Linear Discriminant Analysis (LDA)*.

4.2.1 Príznaky založené na hraniciach

1. **Reťazový kód (Chain code)** – popisuje objekt pomocou postupnosti znakov s určeným smerom. Vyberie sa jeden začiatočný hraničný pixel objektu, následne sa prejdú všetky hraničné body objektu a v každom kroku sa uloží symbol reprezentujúci smer pohybu. Výsledkom je teda postupnosť čísel, tzv. Freemanov kód [7].
2. **Jednoduché geometrické hraničné deskriptory** – sú založené prevažne na geometrických vlastnostiach popisovaných objektov, ako napríklad dĺžka hranice, zakrivenie, signatúra a iné. Kvôli diskrétnemu charakteru digitálnych obrazov sú citlivé na rozlíšenie obrazu [7].
3. **Fourierové deskriptory** – Fourierové deskriptory môžu byť použité na popis uzavretej hranice objektu. Hranica objektu je reprezentovaná postupnosťou hraničných bodov. Tie môžu byť považované za vzorkovacie hodnoty spojitej krivky f, ktorá môže byť rozšírená na funkciu s periódou 2π. Na ňu je následne aplikovaná diskrétna fourierová transformácia, z ktorej sa získajú koeficienty približne popisujúce tvar objektu. Z tých sa zostavia Fourierové deskriptory. Použitím ďalších matematických operácií je možné získať deskriptory invariantné voči zmene pozície, veľkosti a otočenia [2].
4. **Reprezentácia B-spline** – hranica objektu je reprezentovaná pomocou koeficientov krivky B-spline, ktorá príbližne popisuje tvar hranice objektu. Najčastejšie sa používajú krivky tretieho rádu [7].
5. **Iné** – Houghova transformácia, hraničné deskriptory používajúce postupnosť segmentov, a pod.

Algoritmus 4.2.1. Algoritmus pre zistenie vnútornej hranice objektu. Prevzaté z [9]:

```
s = 0;                                          // aktuálny index hraničného pixelu
dir = 7;                  // smer hľadania ďalšieho pixelu hrany (obr. 4.3)
repeat
    // počiatočný smer hľadania
    if dir je nepárne then
        dir = (dir + 6) mod 8;
    else
        dir = (dir + 7) mod 8;
    end
    while aktuálny susedný pixel nie je pixelom objektu do
        dir = (dir + 1) mod 8 ;
    end
    s = s + 1;
    borderPixel[s] = nájdený susedný pixel hrany objektu;
until borderPixel[s] == borderPixel[1] and borderPixel[s-1] == borderPixel[0];
```



Obr. 4.3: Číslicami sú znázornené smery hľadania hraničného pixelu objektu použité v algoritme 4.2.1. Prevzaté z [9]

4.2.2 Príznaky založené na oblastiach

Jednoduché skalárne deskriptory:

1. **Plocha** – udáva počet pixelov ktoré tvoria objekt. Je preto zrejmé, že sa bude výrazne líšiť v závislosti na rozlíšený obrazu.

2. **Eulerovo číslo** – popisuje jednoduchú, topologickú vlastnosť objektu. Ide o rozdiel počtu spojitých častí objektu od počtu dier v objekte. Objekt môže pozostávať z viac ako jednej spojitej časti, v opačnom prípade sa počet spojitých častí objektu rovná jednej [7].

3. **Projekcia** – Projekcia obrazu je jedno-dimenzionálna reprezentácia obrazových komponent, zvyčajne vypočítaná paralelne k súradnicovým osiam. Horizontálna a vertikálna projekcia $p_h(i)$ a $p_v(i)$ sú definovaneé vzťahmi [7]:

$$p_h(i) = \sum_j f(i,j), \quad p_v(j) = \sum_i f(i,j).$$

   Horizontálna projekcia je teda súčet hodnôt pixelov v jednotlivých riadkoch, vertikálna udáva súčet hodnôt pixelov v stĺpcoch.

4. **Výstrednosť** (Eccentricity) – pomer hlavnej a vedľajšej osi objektu [7].

5. **Pretiahnutie** (Elongatedness) – pomer dĺžky a šírky opísaného obdĺžnika. V prípade zakriveného objektu sa pretiahnutie vypočíta ako pomer plochy a druhej mocniny hrúbky objektu. Hrúbka objektu môže byť zistená pomocou počtu krokov erózie aplikovaných na daný objekt, kým úplne nezmizne. V prípade, že počet krokov erózie je d, pretiahnutie objektu je potom [7]:

$$pretiahnutie = \frac{plocha\_objektu}{(2d)^2}$$

6. **Pravouhlosť** – je to maximálna hodnota F_k, kde F_k je pomer plochy objektu a obsahu opísaného obdĺžnika so smerom k [7].

7. **Orientácia** – dáva zmysel iba u pretiahnutých objektov. Je to orientácia dlhšej hrany minimálneho opísaného obdĺžnika [7].

8. **Kompaktnosť** – vyjadruje mieru podobnosti objektu ku kruhu. Je daná vzťahom [7]:

$$\text{kompaktnost} = \frac{(\text{obvod\_objektu})^2}{\text{plocha\_objektu}}$$

.

Medzi ďalšie spôsoby popisu objektu patria momenty, viď. 4.3. Konvexný obal (Convex hull) objektu môže byť použitý pre popis vlastností tvaru objektov, prípadne pre zostrojenie stromovej štruktúry popisujúcej konkávnosť daného objektu. Konvexný obal objektu R je najmenší konvexný región H, pre ktorý platí podmienka $R \subseteq H$ [7]. Ďalším spôsobom ako popísať tvar objektu je dekompozíciou objektu na tzv. tvarové primitívy – najjednoduchšie elementy, ktoré tvoria daný objekt. Následne sa zostrojí graf, kde uzly grafu tvoria tvarové primitívy a hrany medzi nimi popisujú ich vzájomné vzťahy [7].

4.3 Momenty

Momenty môžu byť použité pre popis binárneho, alebo šedo-tónového objektu. Výpočet vychádza z tvaru a jasových hodnôt pixelov daného objektu. Všetky momenty sú závislé na hodnote jasu pixelov, preto pre popis tvaru sa pracuje s binárnymi obrazmi, kde $f(i,j) = 1$ pre všetky pixely objektu. Moment rádu $(p+q)$ je daný vzorcom [7] [8]:

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p j^q, \tag{4.1}$$

kde i a j sú súradnice bodov objektu. Momenty definované vzťahom 4.1 sú závislé od súradníc bodov objektu, preto nie sú invariantné voči geometrickým transformáciám – posunu, zmene mierky a rotácií. Invariantnosť voči posunu môže byť dosiahnutá použitím centrálnych momentov [7] [8]

$$\mu_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p (j - y_c)^q, \tag{4.2}$$

kde x_c a y_c sú súradnice ťažiska objektu, dané vzťahom [7] [8]:

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}}. \tag{4.3}$$

V prípade binárneho obrazu, m_{00} reprezentuje plochu objektu.

Normalizované centrálne momenty, ktoré sú nezávislé od posunu a zmene mierky sú dané vzťahom [7] [8]:

$$\vartheta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \quad \gamma = \frac{p+q+2}{2} \tag{4.4}$$

Pri rotácií obrazu sa hodnota normalizovaných centrálnych momentov mení. Aby sa tomuto javu predišlo, v praxi sa používa kombinácia jednoduchších príznakov, známa ako sada siedmych momentov – *Hu's Moments*. Tieto momenty sú známe aj ako momentové invarianty, keďže sú invariantné voči zmene pozície, mierky a rotácie obrazu. Sú popísané rovnicami [1] [7] [8]:

$$\begin{aligned}
\varphi_1 &= \vartheta_{20} + \vartheta_{02}, && (4.5)\\
\varphi_2 &= (\vartheta_{20} - \vartheta_{02})^2 + 4\vartheta_{11}^1, && (4.6)\\
\varphi_3 &= (\vartheta_{30} - 3\vartheta_{12})^2 + (\vartheta_{21} - 3\vartheta_{03})^2, && (4.7)\\
\varphi_4 &= (\vartheta_{30} + \vartheta_{12})^2) + (\vartheta_{21} + \vartheta_{03})^2), && (4.8)\\
\varphi_5 &= (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{30} + \vartheta_{12})((\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2) \\
&\quad + (3\vartheta_{21} - \vartheta_{03})(\vartheta_{21} + \vartheta_{03})(3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2), && (4.9)\\
\varphi_6 &= (\vartheta_{20} - \vartheta_{02})((\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2) \\
&\quad + 4\vartheta_{11}(\vartheta_{30} + \vartheta_{12})(\vartheta_{21} + \vartheta_{03}), && (4.10)\\
\varphi_7 &= (3\vartheta_{21} - \vartheta_{03})(\vartheta_{30} + \vartheta_{12})((\vartheta_{30} + \vartheta_{12})^2) - 3(\vartheta_{21} + \vartheta_{03})^2) \\
&\quad - (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{21} + \vartheta_{03})(3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2), && (4.11)
\end{aligned}$$

kde hodnoty ϑ môžu byť vypočítané zo vzťahu 4.4. Prvých šesť je invariantných aj voči reflexií, teda nemenia hodnotu pri zrkadlovom obraze objektu, kým φ_7 mení znamienko. V praxi sa počíta s logaritmom ich absolútnych hodnôt aby sa predišlo problémom s presnosťou desatinných čísel [1] [8].

V prípade, že potrebujeme momenty invariantné voči všeobecnej afinnej transformácií, môžeme použiť sadu štyroch momentových invariantov odvodených z druhých a tretích momentov [7].

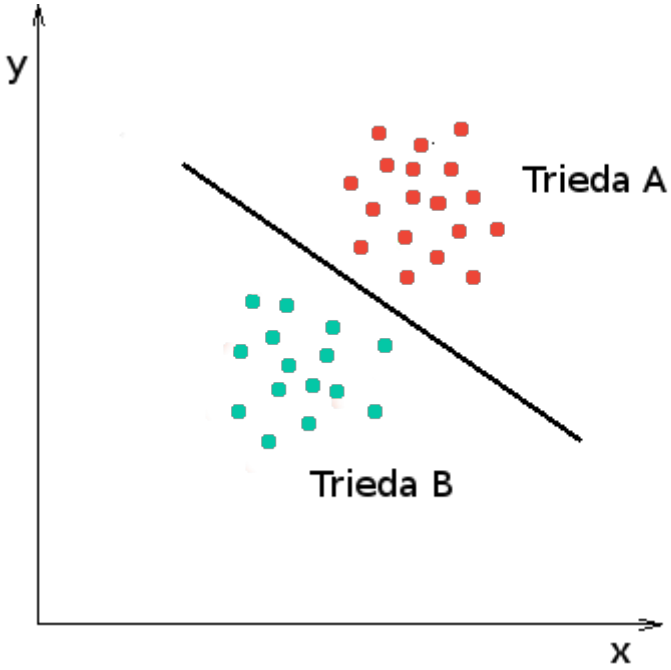
$$\begin{aligned}
I_1 &= \frac{\mu_{20}\mu_{02} - \mu_{11}^2}{\mu_{00}^4}, && (4.12)\\
I_2 &= \frac{\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}\mu_{12}^3 + 4\mu_{21}^3\mu_{03} - 3\mu_{21}^2\mu_{12}^2}{\mu_{00}^{10}}, && (4.13)\\
I_3 &= \frac{\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \mu_{02(\mu_{30}\mu_{12} - \mu_{21}^2)}}{\mu_{00}^7}, && (4.14)\\
I_4 &= \left(\begin{array}{c} \mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^2\mu_{02}\mu_{12}^2 + \\ 12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} + 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} \\ -8\mu_{11}^3\mu_{30}\mu_{03} - 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} + 9\mu_{20}\mu_{02}^2\mu_{21}^2 \\ +12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} + \mu_{02}^3\mu_{30}^2 \end{array} \right) \Big/ \mu_{00}^{11}, && (4.15)
\end{aligned}$$

Kapitola 5

Rozpoznávanie objektov

Rozpoznávanie objektov je proces, kedy klasifikátor priradí jednotlivé objekty do tried. V prípade, že jednotlivé triedy sú známe dopredu z tzv. trénovacej sady (training set), hovoríme o tzv. kontrolovanom (supervised recognition) rozpoznávaní. V opačnom prípade hovoríme o tzv. učení bez učiteľa (unsupervised recognition). V tomto prípade nie sú dopredu známe žiadne informácie o jednotlivých triedach, klasifikátor musí sám rozhodnúť o podobnostiach jednotlivých objektov a zoskupiť tieto podobné objekty do tried [8].

Klasifikátor rozhodne o tom, že daný objekt patrí do špecifickej triedy na základe jeho vlastností popísanými príznakovým vektorom. Samotný príznakový vektor je špecifický pre daný problém, popisuje charakteristické vlastnosti objektu. Proces získania tohoto popisu bol popísaný v kapitole 4. Tieto vektory príznakov sa dajú znázorniť v tzv. priestore príznakov (feature space), viď. obrázok 5.1.



Obr. 5.1: Dvoj-dimenzionálny priestor príznakov x a y. Vektory príznakov sa zhlukujú do skupín reprezentujúcich jednotlivé triedy. V tomto prípade čiara jasne oddeľuje jednotlivé triedy.

Vektory reprezentujúce objekty, ktoré patria do jednej triedy, by sa mali zhlukovať do oddeliteľných skupín. Jednotlivé triedy sa potom dajú rozdeliť tzv. diskriminačnou funkciou. Rovná čiara na obrázku 5.1 znázorňuje diskriminačnú funkciu, ktorá v tomto prípade reprezentuje úlohu klasifikátora. Tento klasifikátor rozdeľuje priestor príznakov na dva oblasti reprezentujúce triedy A a B. V prípade, že vektor objektu X spadá do oblasti triedy A, tento objekt je klasifikovaný ako objekt triedy A.

Klasifikátor je teda zariadenie, ktoré má n vstupov a jeden výstup. Každý vstup je použitý pre vstup informácie o jednom z n príznakov $x_1, x_2, \ldots, x_n$ klasifikovaného objektu. Výstupom klasifikátoru je jeden z R identifikátorov $\omega_1, \omega_2, \ldots, \omega_R$ jednotlivých tried.

5.1 Bayesovo pravidlo

Pre zlepšenie schopnosti rozpoznávania klasifikátoru je niekedy vhodné kombinovať informácie získané z príznakov a apriórnu pravdepodobnosť aplikovaním Bayesovho pravidla. Pre jeden vektor príznakov x má pravidlo vzorec [3] [8]:

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)}, \tag{5.1}$$

kde

$$p(x) = \sum_j p(x|\omega_j)P(\omega_j) \tag{5.2}$$

je hustota pravdepodobnosti vektoru príznakov x, $P(\omega_i)$ je apriórna pravdepodobnosť triedy ω_i, $p(x|\omega_i)$ je podmienená pravdepodobnosť výskytu príznakového vektoru x v triede ω_i – to je pravdepodobnosť, že vektor príznakov x sa objaví pre triedu ω_i, občas sa označuje aj ako vierohodnosť (likelihood). $P(\omega_i|x)$ označuje posteriórnu pravdepodobnosť – pravdepodobnosť, že objekt patrí do triedy ω_i ak má vektor príznakov hodnotu x.

Pri klasifikácií sa následne porovnávajú hodnoty všetkých posteriórnych pravdepodobností $P(\omega_j|x)$. Objekt je potom klasifikovaný do triedy ω_i ak platí vzťah [3]:

$$P(\omega_i|x) > P(\omega_j|x) \qquad \text{pre všetky } j \neq i \tag{5.3}$$

Bayesovo pravidlo teda tvrdí, že pre určenie triedy daného objektu potrebuje poznať dva typy informácií o objektoch. Prvým typom informácie je pravdepodobnosť $P(\omega_i)$ udávajúca, že objekt patrí do trieda ω_i. Druhým typom potrebnej informácie je rozloženie hodnôt príznakového vektoru x pre každú triedu. Oba druhy informácií sa dajú získať analýzou objektov z trénovacej sady.

5.2 Diskriminačná funkcia

Klasifikátor môže byť reprezentovaný sadou diskriminačných funkcií $g_i(\vec{x})$ pre každú triedu $\omega_1, \omega_2, \ldots, \omega_R$, kde $\vec{x}$ je príznakový vektor. Vektor $\vec{x}$ je potom klasifikovaný do triedy ω_i, ktorej funkcia $g_i(\vec{x})$ má maximálnu hodnotu [4]. Pre Bayesov klasifikátor sú jeho diskriminačné funkcie posteriórne pravdepodobnosti (viď. 5.1). Použitím diskriminačných funkcií je priestor príznakov rozdelený na jednotlivé regióny, oddelené rozhodovacími hranicami (*decision boundaries*). Rozhodovacie hranice susedných regiónov sú definované [8]:

$$g_{ij}(\vec{x}) \equiv g_i(\vec{x}) - g_j(\vec{x}) = 0, \tag{5.4}$$

kde $i, j = 1, 2, \ldots R$, $i \neq j$.

Najčastejšie používanou funkciou hustoty pravdepodobnosti je Gaussovské rozloženie hustoty pravdepodobnosti [8]:

$$\mathcal{N}(\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\vec{x} - \mu)^T \Sigma^{-1} (\vec{x} - \mu)\right), \tag{5.5}$$

kde μ je stredná hodnota vektoru $\vec{x}$ a Σ označuje $n \times n$ kovariantnú maticu. Diskriminačná funkcia je daná vzťahom [4]:

$$g_i(\vec{x}) = -\frac{1}{2}(\vec{x} - \mu_i)^T \Sigma_i^{-1} (\vec{x} - \mu_i) + \ln P(\omega_i) + c_i, \tag{5.6}$$

kde c_i je konštanta $c_i = -\frac{n}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma_i|$.

Predpokladajme, že všetky príznaky tvoriace príznakový vektor sú nezávislé a majú rovnaký rozptyl. Kovariantná matica je $\Sigma = \sigma^2 \mathbf{I}$, kde $\mathbf{I}$ označuje n-dimenzionálnu maticu identity. Expandovaním vzťahu 5.6 dostaneme [4]:

$$g_i(\vec{x}) = \frac{1}{2\sigma^2}[\vec{x}^T \vec{x} - 2\mu_i^T \vec{x} + \mu_i^T \mu_i] + \ln P(\omega_i) \tag{5.7}$$

Výraz $x^T x$ a c_i sú konštanty rovnaké pre všetky i, takže môžu byť vypustené, čím dostaneme vzťah [8]:

$$g_i(\vec{x}) = w_i^T \vec{x} + w_{i0}, \tag{5.8}$$

kde

$$w_i = \frac{1}{\sigma^2} \mu_i \quad \text{a} \quad w_{i0} = -\frac{1}{2\sigma^2} \mu_i^T \mu_i + \ln P(\omega_i) \tag{5.9}$$

Vzťah 5.8 definuje lineárnu diskriminačnú funkciu [4].Príslušná rozhodovacia nadrovina (decision hyperplane) môže byť vyjadrená ako [8]:

$$g_{ij}(\vec{x}) \equiv g_i(\vec{x}) - g_j(\vec{x}) = w^T(\vec{x} - \vec{x}_0) = 0 \tag{5.10}$$

kde

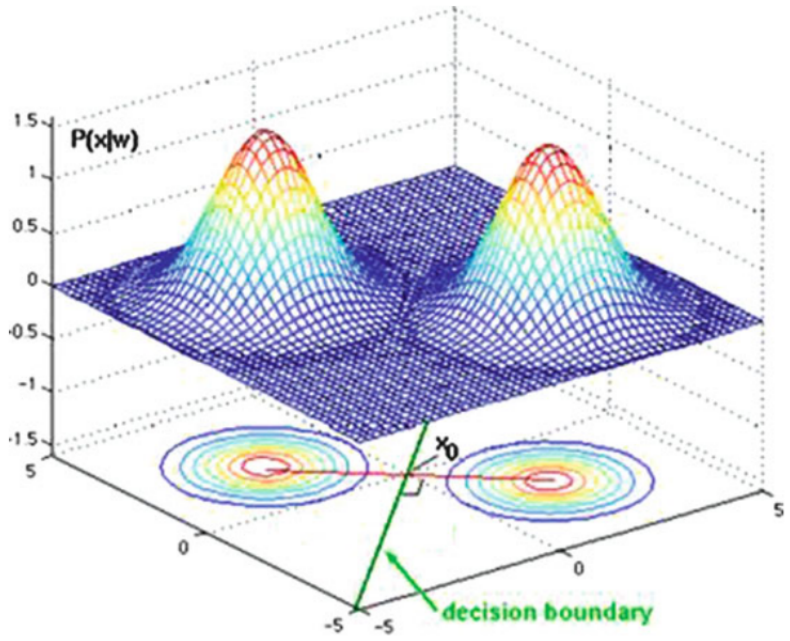
$$w = \mu_i - \mu_j \tag{5.11}$$

a

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \sigma^2 \ln\left(\frac{P(\omega_i)}{P(\omega_j)}\right) \frac{\mu_i - \mu_j}{||\mu_i - \mu_j||^2} \tag{5.12}$$

Tento výraz definuje nadrovinu prechádzajúcu bodom x_0. V prípade, že $P(\omega_i) = P(\omega_j)$, potom $x_0 = \frac{1}{2}(\mu_i + \mu_j)$, tzn. nadrovina prechádza stredom medzi μ_i a μ_j a je kolmá na ich spojnicu (znázornené na obrázku 5.2) [4].

Ak sú apriórne pravdepodobnosti všetkých tried $P(\omega_i)$ rovnaké a kovariantná matica $\Sigma = \sigma^2 \mathbf{I}$, výpočet maximálnej hodnoty výrazu 5.6 je ekvivalentný výpočtu minimálnej Euklidovskej vzdialenosti vektoru $\vec{x}$ od stredu triedy ω_i [4]. Tento klasifikátor je teda založený na princípe minimálnej vzdialenosti.



Obr. 5.2: Dva normálne distribúcie reprezentujúce triedy, ktoré majú rovnaké apriórne pravdepodobnosti. Rozhodovacia hranica (decision boundary) je kolmá na spojnicu stredov tried a prechádza bodom x_0, ktorý leží v strede tejto spojnice. Prevzaté z [4].

5.3 Princíp minimálnej vzdialenosti

Klasifikátor založený na princípe minimálnej vzdialenosti je špeciálny príklad klasifikácie s diskriminačnou funkciou [7]. Majme R vektorov definovaných v priestore príznakov, $v_1, v_2, \ldots, v_R$, ktoré reprezentujú vzory jednotlivých tried $\omega_1, \omega_2, \ldots, \omega_R$. Klasifikátor potom klasifikuje objekt s vektorom x do triedy, ktorej vzor je k nemu najbližšie.

$$d(\mathrm{x}) = \omega_r \iff |v_r - x| = \min_{s=1,\ldots,R} |v_s - x| \tag{5.13}$$

Algoritmus 5.3.1. popisuje princíp klasifikácie podľa minimálnej vzdialenosti [7].

1. **Učenie:** Pre všetky triedy vypočítaj vzor $\mathbf{v}_i$ podľa trénovacej sady

   $$\mathbf{v}_i(k_i + 1) = \frac{1}{k_i + 1} \left(k_i \mathbf{v}_i(k_i) + \mathbf{x}_i(k_i + 1)\right),$$

   kde $\mathbf{x}_i(k_i + 1)$ sú objekty z triedy ω_i a k_i udáva počet objektov triedy ω_i použitých pre učenie klasifikátora.

2. **Klasifikácia:** Pre vektor $\mathbf{x}$ objektu vypočítaj vzdialenosť $\mathbf{x}$ od vektoru triedy v_i. Objekt klasifikuj do triedy ω_i v prípade, že vzdialenosť $\mathbf{x}$ od vektoru v_j je najmenšia (rovnica 5.13).

Kapitola 6

Návrh aplikácie

Cieľom návrhu aplikácie je navrhnúť základný koncept aplikácie, pomocou ktorej bude možné vyhľadávať jednoduché objekty v čiernobielych snímkoch. V návrhu sú popísané jednotlivé logické časti tejto aplikácie.

Návrh užívateľského rozhrania

Grafické užívateľské rozhranie má poskytnúť prehľadné a intuitívne ovládanie aplikácie. Aplikácia bude ovládaná myšou a bude pozostávať z jedného hlavného okna. V okne aplikácie bude náhľad spracovávaného obrázku, tlačítka pre vykonávanie požadovaných akcií a panel s podrobnými informáciami o nájdenom objekte. Pre načítanie obrazových súborov bude možné použiť kontextové okná.

Predspracovanie obrazu

Po načítaní obrazu bude potrebné opraviť prípadné nedostatky ako napríklad šum, prípadne previesť farebný obraz na čiernobiely podľa vzťahu 2.1. Pre redukovanie šumu a iných nedostatkov je možné použiť techniku rozostrenia obrazu pomocou lineárnych alebo nelineárnych filtrov uvedených v kapitole 2.

Segmentácia obrazu

Úlohou segmentácie je oddeliť pozadie obrazu od jednotlivých objektov. Existuje mnoho techník segmentácie, najjednoduchšou je segmentácia prahovaním, ktorá prevedie vstupný čiernobiely obraz na obraz binárny. Výpočtovo najefektívnejšou technikou prahovania je globálne prahovanie, ktoré používa jeden globálny prah pre celý obraz, preto sa binárny obraz získa jediným prechodom celého obrazu. Určenie správneho prahu je kritické pre úspešnú segmentáciu prahovaním, preto bude prah vypočítaný automaticky analýzou histogramu intenzít v obraze, pomocou metódy Otsu popísanej v podkapitole 3.1.1. Napriek tomu, že táto metóda dosahuje uspokojivé výsledky pri určovaní vhodného prahu, v niektorých prípadoch je potrebné upraviť vypočítanú hodnotu prahu. Užívateľ preto bude môcť ručne nastaviť prah.

Globálne prahovanie dosahuje dobré výsledky za optimálnych podmienok, akými sú napríklad rovnomerné rozloženie jasu v obraze. Pre eliminovanie vplyvu nerovnomerného rozloženia jasu bude implementovaná technika adaptívneho prahovania. Úspešnosť tejto techniky však závisí od viacerých faktorov, akými sú veľkosť použitej masky, určenie lokálneho prahu, prípadne veľkosť objektov v obraze. Veľkosť masky v tomto prípade označuje

veľkosť okolia pixelu, z ktorého je vypočítaný lokálny prah. Vhodná veľkosť masky sa líši pre obrazy s rôznym rozlíšením a rôzne veľkými objektami. Musí byť dosť veľká na to aby zachytila dostatočné množstvo pixelov pozadia a objektov pre určenie správneho lokálneho prahu. Túto veľkosť bude preto zadávať užívateľ. Každý pixel obrazu bude následne prahovaný podľa vypočítanej hodnoty lokálneho prahu. Hodnota tohto prahu bude určená ako priemer maximálnej a minimálnej hodnoty okolitých pixelov.

Detekcia objektov

Následne bude potrebné detegovať jednotlivé objekty v binárnom obraze. Keďže predpokladáme, že jednotlivé objekty sa nebudú prekrývať, je možné použiť jednoduchý dvojpriechodový algoritmus pre identifikáciu objektov (viď. 4.1.1). Tento algoritmus označí všetky pixely objektov jedinečným identifikátorom. Pre vizuálne znázornenie jednotlivých objektov sa ako identifikátor použije náhodne vygenerovaná farba. Týmto spôsobom bude každý objekt znázornený inou, jasne rozlíšiteľnou farbou.

Po úspešnej detekcii objektov bude potrebné vypočítať ich príznaky. Niektoré vlastnosti, akými sú napríklad plocha objektu, dĺžka hrany, pozícia prípadne ťažisko objektu nie sú vhodnými kandidátmi pre príznaky použité pri klasifikácii objektu. Napriek tomu poskytujú užitočné informácie o objekte v obraze, preto budú zobrazené ako doplňujúce vlastnosti daného objektu. Príznaky použité pri klasifikácii musia byť vybrané tak, aby boli čo najmenej ovplyvnené zmenou pozície objektov, ich otočením, či rôznou veľkosťou, zároveň ale musia jasne charakterizovať objekt. Tieto vlastnosti spĺňajú momentové invarianty popísané v podkapitole 4.3. Pre jednoznačnú identifikáciu bude potrebné vypočítať viacero príznakov a následne ich predať klasifikátoru vo vhodnej forme – vektore príznakov. Vektor príznakov bude reprezentovaný ako pole hodnôt jednotlivých príznakov.

Klasifikátor

Predtým ako začne proces klasifikácie objektov bude nutné natrénovať klasifikátor trénovacími dátami. Tie môžu byť reprezentované vhodnou databázou objektov rozdelených do jednotlivých tried. Z týchto objektov sa pre každú triedu extrahuje vzor (viď. 5.3.1). Proces trénovania klasifikátora je zvyčajne časovo náročný, preto je častokrát výhodnejšie mať klasifikátor natrénovaný dopredu a uložený vo vhodnej forme pre efektívne načítanie programom. Tento prístup ale znemožňuje dodatočnú zmenu trénovacej sady a tým pádom aj vlastností klasifikátora. Preto budú implementované oba prístupy získania klasifikátora.

Klasifikácia a vyhľadávanie objektov

Klasifikácia objektov v obraze bude prebiehať postupným porovnávaním príznakového vektoru každého objektu s príznakovými vektormi tried klasifikátora. Výsledkom týchto porovnaní bude určenie podobnosti príznakových vektorov, resp. podobnosti objektu a danej triedy. Táto podobnosť môže byť vyjadrená ako vzdialenosť dvoch vektorov v príznakovom priestore. V prípade, že pre danú triedu bude podobnosť jej príznakového vektora a vektora objektu najväčšia (v prípade počítania vzdialenosti vektorov najväčšiu podobnosť určuje najmenšia vzdialenosť vektorov, viď. 5.3.1), bude objekt klasifikovaný do danej triedy. Hlavným problémom tohto prístupu bude vhodná testovacia sada klasifikátora. V prípade neúplnej alebo chybnej testovacej sady sa chybovosť klasifikátora môže výrazne zvýšiť.

Vyhľadávanie objektov bude po klasifikácií prebiehať jednoduchým hľadaním objektov klasifikovaných do hľadanej triedy. V prípade nájdeného objektu sa o ňom vypíšu na obra-

zovku základné informácie a vlastnosti. Každý nájdený objekt bude v obraze zvýraznený opísaným obdĺžnikom. V prípade, že sa v obraze bude nachádzať viac hľadaných objektov, užívateľ bude môcť kliknutím myšou na konkrétny objekt zobraziť o ňom informácie.

Návrh tried aplikácie

Aplikácia bude rozdelená do troch hlavných balíkov tried. Balík `Window` bude obsahovať triedy pre zobrazenie grafického užívateľského rozhrania. Ovládacie prvky aplikácie budú použité z balíčku `java.swing`. Balík `Classificator` bude implementovať metódy pre natrénovanie klasifikátora a metódy pre klasifikáciu a vyhľadávanie objektov podľa ich príznakových vektorov. Balík `ImgProc` bude obsahovať triedy implementujúce metódy pre spracovanie obrazu a prácu s jednotlivými objektami. Posledný balík `Controller` bude reprezentovať takzvanú medzivrstvu medzi užívateľským rozhraním a dátami aplikácie. na základe stlačených tlačidiel vykoná potrebné akcie a pripravý dáta ktoré sa potom zobrazia používateľovi aplikácie.

Kapitola 7

Implementácia

V tejto kapitole budú stručne popísané najdôležitejšie časti implementácie programu. Ako implementačný jazyk bol zvolený jazyk Java. Aplikácia bola vyvíjaná vo vývojovom prostredí Netbeans.

7.1 Spracovanie obrazu

Vstupné obrazové dáta sú načítané pomocou dialógového okna pre výber súboru a uložia sa do inštancie triedy `BufferedImage` balíka `java.awt.image`. Pre jednotnosť načítaných dát z rôznych formátov ako napríklad formát obrazu *.png*, ktorý obsahuje informácie o priehľadnosti narozdiel od iných formátov ako napríklad *.jpg*, bola implementovaná metóda `convertImageToArgb()`. V tejto metóde sa načítané obrazové dáta prevedú na jednotný formát vytvorením inštancie triedy `Graphics2D` balíka `java.awt` a volaním metód `drawImage()` a `dispose()`. Po načítaní obrazu je potrebné farebný obraz previesť na čiernobiely. To sa uskutoční pomocou metódy `grayscaleConversion()` triedy `ImgProc`.

7.1.1 Histogram

Po načítaní obrazu je potrebné vypočítať jeho histogram. Výpočet histogramu prebieha v metóde `computeHistogram()` triedy `ImgProc`, kde sa spočíta početnosť hodnôt pixelov $0 - 255$. Aby sa mohli hodnoty histogramu zobraziť na stupnici od 0 po 100, prepočítajú sa podľa vzťahu:

$$\frac{počet\ hodnoty[i] * 100}{maximálna\ početnosť}$$

Následne sa z hodnôt histogramu vytvorí inštancia triedy `BufferedImage` v metóde `showHistogram()`. Pre znázornenie hodnoty prahu sa v obraze vykreslí vertikálna čiara. Aby mohol užívateľ upraviť hodnotu prahu, boli implementované metódy `mouseDragged()` a `mousePressed()` triedy `MouseAdapter` pre panel zobrazujúci histogram. Pre automatické určenie hodnoty prahu bola implementovaná metóda `computeThreshold()` podľa Otsu metódy popísanej v podkapitole 3.1.1.

7.1.2 Prahovanie

Pre segmentáciu prahovaním boli implementované dve metódy – globálne a adaptívne prahovanie. Užívateľ má možnosť zvoliť, ktorá metóda bude použitá. Prahovanie je implementované tak, že sa prejde celý čiernobiely obrazok a podľa toho či je hodnota aktuálne

spracovávaného pixelu väčšia ako stanovený prah, zmení sa jeho hodnota na bielu farbu, teda farbu pozadia, v opačnom prípade na čiernu farbu. Týmto spôsobom je vstupný obraz prevedený na binárny, kde pixely objektov majú čiernu farbu.

Adaptívne prahovanie bolo implementované v metóde `adaptiveSegmentation()`. V tomto prípade sa opäť prechádza celý obraz, ale každý pixel sa prahuje podľa vypočítaného lokálneho prahu. Prah je určený ako medián hodnôt jeho susedných pixelov.

7.1.3 Identifikácia objektov

Pre identifikáciu objektov bol implementovaný algoritmus 4.1.1 v metóde `labelObjects()` triedy `imgProc`. V prvom prechode celého obrazu sa priradia počiatočné identifikátory pixelom objektov. V prípade, že nastane tzv. kolízia identifikátorov (obrázok c) 4.1), kolízne páry sa ukladajú do tzv. tabuľky ekvivalencií reprezentovanej triedou `CollisionsTable`. Pre vloženie nového páru ekvivalentných identifikátorov bola implementovaná metóda `add()`. Pre optimalizáciu prehľadávania tabuľky je vhodné pamätať si niekoľko naposledy vložených párov identifikátorov.

Po prvom kroku algoritmu sú všetky pixely objektov označené identifikátormi, avšak je potrebné vyriešiť ekvivalentné páry identifikátorov (viď. 4.2). Tabuľka ekvivalentných párov identifikátorov je prevedená na asociatívne pole kvôli optimalizácií vyhľadávania. Pre všetky ekvivalentné identifikátory sa následne vygeneruje jedinečná farba objektu. Toto asociatívne pole je vytvorené tak, že kľúče tohto poľa reprezentujú identifikátory pixelov a ich hodnoty sú vygenerované farby objektov. Všetky ekvivalentné identifikátory reprezentujúce jeden objekt teda majú uloženú rovnakú vygenerovanú farbu.

V druhom prechode obrazu sa pre všetky pixely objektov vyhľadá vygenerovaná farba v asociatívnom poli podľa ich identifikátoru, a tá sa následne uloží ako nová hodnota pixelu. Týmto spôsobom sa v obraze jednotlivé objekty zafarbia odlišnými farbami.

7.1.4 Výpočet príznakov objektov

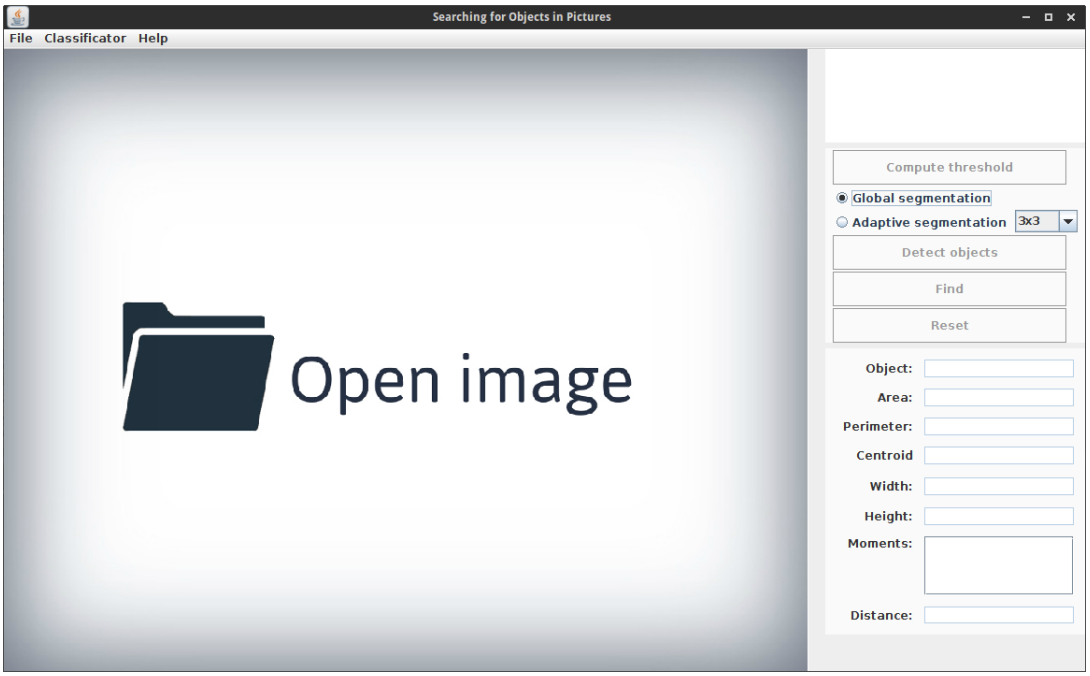
Pre každý objekt sa vytvorí inštancia triedy `PictureObject`, ktorá obsahuje pole všetkých bodov objektu, jeho príznakový vektor, súradnice opísaného obdĺžnika objektu a ďalšie informácie o objekte. Príznakový vektor predstavuje inštanciu triedy `FeatureVector`, kde hodnoty príznakov, ktoré sa použijú pri následnej klasifikácii objektu sú uložené ako `ArrayList` hodnôt typu `Double`. Okrem toho boli implementované metódy pre zistenie celkovej plochy objektu, jeho ťažiska, výšky a šírky objektu v obraze. Taktiež bol implementovaný algoritmus 4.2.1 pre výpočet dĺžky vnútornej hrany objektu.

7.1.5 Vyhľadávanie objektov v obraze

Klasifikácia a následné vyhľadávanie objektov bolo implementované v triede `Classificator`. Objekty sa klasifikujú do jednotlivých tried podľa minimálnej vzdialenosti (viď. 5.3.1). Každá trieda je reprezentovaná príznakovým vektorom. Klasifikátor teda vypočíta vzdialenosť medzi príznakovým vektorom objektu a jednotlivých tried a následne objekt klasifikuje do triedy, ktorej vzdialenosť v priestore príznakov je najmenšia, resp. jeho príznakový vektor sa najviac podobá na vektor príznakov danej triedy.

7.2 Grafické užívateľské rozhranie

Grafické užívateľské prostredie bolo navrhnuté aby poskytlo jednoduché a intuitívne ovládanie. Aplikácia sa skladá z jedného hlavného okna (obrázok 7.1) a niekoľkých dialógových okien. Z menu aplikácie je možné načítať nový obrázok, natrénovať klasifikátor vybratím trénovacej sady, prípadne načítať už natrénovaný klasifikátor a taktiež uložiť natrénovaný klasifikátor. V hlavnom okne sa zobrazuje náhľad práve spracovávaného obrázku. V pravej časti okna aplikácie sa nachádza panel, kde sa zobrazuje histogram načítaného obrazu, tlačítka pomocou ktorých sa vykonávajú jednotlivé akcie a výpis vypočítaných vlastností nájdeného objektu.



Obr. 7.1: Náhľad hlavného okna aplikácie

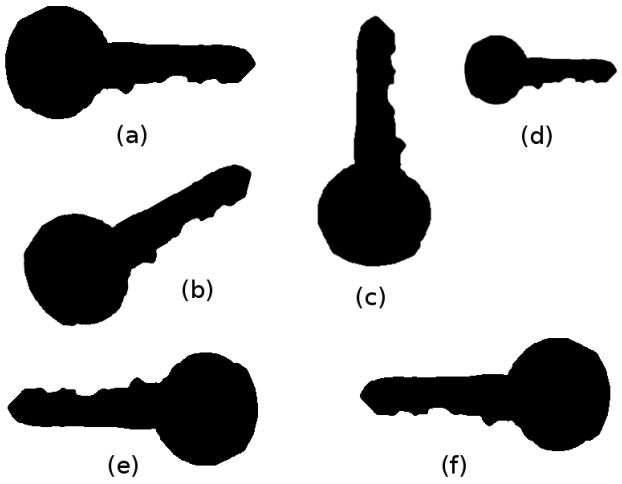
Kapitola 8

Testovanie

Funkčnosť aplikácie bola otestovaná experimentálne pomocou vytvorenej sady testovacích obrázkov. Bola vytvorená databáza jednoduchých objektov základných tvarov, pomocou ktorej bol následne natrénovaný klasifikátor. Testovacia sada pozostávala z obrázkov objektov v rôznom rozlíšení. V testoch boli objekty v rôznych veľkostiach, rozlične otočené a posunuté. Boli spravené pokusy s prekrývajúcimi a neprekrývajúcimi sa objektami, ako aj experimenty s nerovnomerným osvetlením obrazovej scény.

8.1 Základné testy

Základné testy prebehli na jednoduchých objektoch, ktoré boli rôzne natočené a v rôznych veľkostiach. Testy prebehli na segmentovaných obrazoch objektov (viď. 8.1), aby sa predišlo nežiadúcim vplyvom šumu a chybnej segmentácie v dôsledku nerovnomerného osvetlenia, či prípadných vlastných tieňov objektov. Cieľom teda bolo zaznamenať čo najlepšie tvar objektov a následne porovnať hodnoty jednotlivých príznakov. Hodnoty príznakov sú znázornené v tabuľke 8.1.



Obr. 8.1: Príklad objektov tvaru kľúča v rôznych otočeniach a veľkostiach. Originálny objekt (a), otočený o 30° (b), 90° (c), zmenšený objekt (d), otočený o 180° (e), Zrkadlový obraz pôvodného objektu (f)

Príznaky	0°	30°	90°	Zmenšené	180°	Zrkadlový obraz
φ_1	0.3297	0.3293	0.3309	0.3276	0.3296	0.3296
φ_2	0.0647	0.0646	0.0653	0.0636	0.0646	0.0646
φ_3	0,0191	0,0188	0,0196	0,0188	0,0192	0,0191
φ_4	-0,0001	0,0023	0,0095	-0,0001	0,0001	0,0001

Tabuľka 8.1: Tabuľka znázorňujúca hodnoty príznakov objektov z obr. 8.1. Hodnoty boli zaokrúhlené. Vybrané boli príznaky 4.5, 4.6, 4.7 a 4.8 uvedené v podkapitole 4.3

Z tabuľky 8.1 je jasne vidieť, že hodnoty príznakov sa líšia len minimálne a teda sú nezávislé na zmene otočenia, pozície či veľkosti objektu. Tieto príznaky teda je možné použiť pre rozpoznávanie tvaru rôznych objektov. Následná klasifikácia objektov a vyhľadanie objektov, v tomto prípade tvaru kľúča, prebehla úspešne. Týmito základnými testami bol overený vhodný výber príznakov.

8.2 Nerovnomerné osvetlenie objektov

Následne boli vykonané testy na vyhľadávanie neprekrývajúcich sa objektov pri nerovnomernom osvetlení snímanej scény. V týchto obrazoch bol hlavným problémom správna segmentácia objektov. V prípade, že objekty neboli správne oddelené od pozadia, ich následné vyhľadávanie sa vyznačovalo zvýšenou chybovosťou. Segmentácia pomocou globálneho prahu nedokázala detegovať všetky objekty v obraze aj napriek automatickému určeniu hodnoty prahu (viď. 3.1.1). Kvôli týmto problémom bola implementovaná metóda adaptívneho prahovania.

Použitím adaptívneho prahovania boli úspešne segmentované objekty v obraze s relatívne malým rozlíšením. Doba spracovania však bola oproti globálnemu prahovaniu vyššia, závisela na veľkosti použitej masky, teda veľkosť okolia pixelu. Adaptívne prahovanie však dosahovalo výrazne horších výsledkov v obrazoch s vysokým rozlíšením, pri ktorých bolo výrazne pomalšie už pri použití relatívne malého okolia pixelu. V týchto prípadoch neoddelilo celé objekty od pozadia, ale iba ich hranice. Aby bola segmentácia aspoň čiastočne úspešná, bolo potrebné brať do úvahy výrazne väčšie okolie pixelu, čo sa veľmi negatívne podpísalo na efektívnosti celej segmentácie. V týchto prípadoch preto poskytovala lepšie výsledky technika globálneho prahovania.

Výsledný tvar objektov môže byť taktiež ovplyvnený ich vlastnými tieňmi čo je v tomto prípade nežiadúce. Testovaním bolo zistené, že adaptívne prahovanie podľa očakávaní tieto tiene väčšinou dokáže eliminovať narozdiel od techniky globálneho prahovania.

8.3 Prekrývajúce sa objekty

Boli taktiež uskutočnené testy vyhľadávania objektov v prípade prekrývajúcich sa alebo neúplných objektov v obraze. V týchto prípadoch segmentácia a následná identifikácia objektov algoritmom 4.1.1 identifikovala prekrývajúce sa objekty ako jeden objekt. To následne výrazne ovplyvnilo hodnoty jednotlivých príznakov a teda aj následnú klasifikáciu a vyhľadávanie objektov. Aby sa predišlo zisteným problémom, bolo by potrebné zmeniť princíp segmentácie objektov.

V prípade neúplných objektov boli rovnako ovplyvnené hodnoty príznakov a teda aj celkový výsledok klasifikácie a vyhľadania objektov. Úspešnosť vyhľadania objektu v tomto prípade závisela hlavne na celkovej veľkosti chýbajúcej časti objektu – čím bola táto časť menšia, úspešnosť rozpoznania bola lepšia.

8.4 Zhodnotenie výsledkov testovania

Testovaním bolo overené, že navrhnutá aplikácia dokáže úspešne vyhľadávať jednoduché neprekrývajúce sa objekty v čiernobielom obraze. Z uskutočnených testov bolo zistené, že jedným z najdôležitejších krokov je správna segmentácia objektov. V prípade, že objekty boli od pozadia správne oddelené, vo väčšine prípadov boli správne klasifikované. Objekty sa však nesmeli prekrývať a nesmela žiadna ich časť chýbať. Boli otestované oba implementované techniky segmentácie obrazu. Adaptívne prahovanie oproti tomu globálnemu dosahovalo výrazne lepších výsledkov v prípade nerovnomerného osvetlenia snímanej scény, avšak pri vysokom rozlíšení obrazu bolo výrazne pomalé, a teda prakticky nepoužiteľné. V prípade vhodných podmienok bolo globálne prahovanie výrazne efektívnejšie.

Testovaním bolo taktiež overené, že na správnosť klasifikácie má výrazný vplyv výber vhodných príznakov. Vhodnými príznakmi sú v tomto prípade tie, ktoré sa nemenia, prípadne svoje hodnoty menia minimálne v závislosti na zmene pozície, otočenia či veľkosti objektu.

Kapitola 9

Záver

Cieľom tejto práce bolo vytvoriť program pre vyhľadávanie jednoduchých objektov v čiernobielych snímkoch. V práci som postupne popísal jeden z možných prístupov riešenia tohto problému. Postup bol rozdelený na tri hlavné kroky. Prvým krokom bola segmentácia obrazu. Zvolil som techniku prahovania a implementoval som techniky globálneho a adaptívneho prahovania. Ďalším krokom bol popis jednotlivých segmentovaných oblastí, teda objektov v obraze. Pre identifikáciu objektov bol implementovaný dvoj-priechodový algoritmus. Pre popis jednotlivých objektov pomocou príznakov som zvolil momentové invarianty. Posledným krokom bola klasifikácia a vyhľadávanie objektov. Bol zvolený klasifikátor podľa minimálnej vzdialenosti, ktorý klasifikuje všetky objekty v obraze do klasifikačných tried. Následne po tom prebieha vyhľadávanie požadovaných objektov.

Navrhnutý program je schopný úspešne vyhľadať neprekrývajúce sa objekty, určiť ich pozíciu v obraze, ťažisko, rozmery nájdeného objektu, počet jeho hraničných pixelov a ďalšie príznaky. Program dokáže natrénovať klasifikátor z trénovacej sady objektov ako aj načítať už natrénovaný klasifikátor.

Následne bol navrhnutý program otestovaný sadou testov popísaných v kapitole 8. Testami bolo overené, že program dosahuje uspokojivých výsledkov pri vyhľadávaní neprekrývajúcich sa objektov v snímkoch. Najčastejšie problémy boli spojené s nesprávanou segmentáciou obrazu, spôsobenou hlavne nerovnomerným osvetlením obrazu. Pre elimináciu tohto nežiadúceho javu bola implementovaná metóda adaptívneho prahovania, ktorá sa osvedčila pri obrazoch s relatívne malým rozlíšením, avšak pri vyššom rozlíšení bolo zistené, že je prakticky nepoužiteľná kvôli vysokej výpočtovej náročnosti. Pre správne vyhľadávanie prekrývajúcich sa objektov by bolo potrebné zmeniť techniku detegovania objektov v obraze.

V tejto bakalárskej práci je možné pokračovať a ďalej ju rozširovať. Bolo by vhodné doplniť implementáciu tak, aby užívateľ mohol rozšíriť natrénovaný klasifikátor o nové objekty. Tak isto by bolo vhodné experimentovať s rôznymi technikami detekcie objektov v obraze, aby bolo možné detegovať prekrývajúce sa objekty. V neposlednej rade by bolo možné aplikáciu vylepšiť tým, že by sa zvolil komplikovanejší prístup klasifikácie objektov.

Literatúra

[1] Burger, W.; Burge, M. J. *Digital image processing : an algorithmic introduction using Java*. 1st ed. New York: Springer, 2008. ISBN 978-1-84628-379-6.

[2] Dalitz, C.; Brandt, C.; Goebbels, S.; aj. : Fourier Descriptors for Broken Shapes. *EURASIP Journal on Advances in Signal Processing*, roč. 2013, 2013: s 161–, ISSN 1687-6180, 10.1186/1687-6180-2013-161. Dostupné z: <http://asp.eurasipjournals.com/content/2013/1/161>

[3] Davies, E. *Machine vision : theory, algorithms, practicalities*. 3rd ed. Boston: Elsevier, 2005. ISBN 0-12-206093-8.

[4] Dougherty, G. *Pattern recognition and classification : an introduction*. New York: Springer Science+Business Media, 2012. ISBN 978-1-4614-5322-2.

[5] Greensted, A. : Otsu Thresholding. online. Dostupné z: <http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html>

[6] Maljovec, D. : CS6640 - Project 1. online. Dostupné z: <http://www.cs.utah.edu/~maljovec/CS6640/project1/index.html>

[7] Sonka, M.; Hlavac, V.; Boyle, R. *Image processing, analysis, and machine vision*. 3rd ed. Toronto: Thomson, 2008. ISBN 978-0-495-08252-1.

[8] Theodoridis, S.; Koutroumbas, K. *Pattern recognition*. 2nd ed. Boston: Academic Press, 2003. ISBN 01-268-5875-6.

[9] Wang, M.; Lai, C.-H. *A concise introduction to image processing using C++*. Boca, Raton: CRC Press, 2009. ISBN 15-848-8897-0.

Prílohy

Zoznam príloh

Príloha A

Obsah CD

- `Build/` – adresár so spustiteľnou verziou programu
- `Data/` – adresár s ukážkovými dátami
- `Source/` – adresár so zdrojovými súbormi
- `Theses/` – adresár s textom technickej správy

Príloha B

Manuál

Spustenie aplikácie

Pre spustenie aplikácie je nutné mať k dispozícií:

- JAVA SE 8

V adresáry `Build/` sa nachádza:

- `IBP.jar` – spustiteľný jar archív
- `classificator.socl` – natrénovaný klasifikátor

Aplikácia následne spustíme príkazom:

- `java -jar IBP.jar`

Preloženie aplikácie

Pre preklad je nutné mať k dispozícií:

- JAVA SE 8
- Aplikáciu ant

V zdrojovom adresári `Source/` sa nachádza:

- `build.xml` – build súbor pre aplikáciu ant
- `src/` – adresár so zdrojovými súbormi

V zdrojovom adresári `Source/` spustíme príkazom `ant compile` preklad. Preklad vytvorí nové adresáre:

- `build/` – adresár s preloženými class súbormi a spustiteľným `.jar` archívom
- `doc/` – adresár s vygenerovanou dokumentáciou programu

Následne je možné aplikáciu spustiť príkazom `ant run`.

Ovládanie aplikácie

Aplikácia sa ovláda pomocou grafického rozhrania. Po spustení sa načíta súbor `classificator.socl`, ktorý sa hľadá v adresári, z ktorého bola aplikácia spustená. V prípade, že tento súbor neexistuje, je potrebné natrénovať nový klasifikátor z databázy vzorových objektov. O tejto skutočnosti program informuje užívateľa textom *Classificator was not found* v pravom dolnom rohu hlavného okna.

Natrénovanie klasifikátoru novou trénovacou sadou

Pre vytvorenie novej trénovacej sady je potrebné vytvoriť adresár so snímkami jednotlivých objektov tried. Adresár musí mať nasledujúcu štruktúru:

- `adresár/`
  - `trieda1/` - zložka so snímkami objektov triedy
  - `trieda2/` - zložka so snímkami objektov triedy
  - ...

Zložky `triedaX/` obsahujú 1 až N snímkov objektov danej triedy. Názov triedy reprezentuje názov zložky `triedaX/`.

Ak máme takto pripravený adresár, tak v aplikácií vyberieme v menu položku `Classificator -> Train classificator`. V prieskumníku vyberieme požadovaný adresár. Aplikácia následne natrénuje nový klasifikátor podľa trénovacej sady a natrénovaný klasifikátor uloží do súboru `classificator.socl`. V prípade, že bol úspešne natrénovaný klasifikátor, program o tom informuje užívateľa dialógovým oknom.

Užívateľ má taktiež možnosť načítať už natrénovaný klasifikátor. V menu vyberie položku `Classificator -> Load classificator` a následne vyberie súbor s uloženým klasifikátorom.

Vyhľadávanie objektov

Po spustení aplikácie je možné načítať obrázok kliknutím na panel `Open image`, ďalšie obrázky je možné načítať z menu v položke `File -> New image`. Po načítaní obrázku sa v pravom hornom rohu zobrazí histogram, kde zvislá čiara znázorňuje hodnotu prahu. Zmeniť hodnotu tohto prahu je možné pomocou myši pohybom tejto zvislej čiary na požadovanú hodnotu. Tlačítko `Compute threshold` vypočíta automaticky optimálnu hodnotu prahu.

Pomocou tlačítka `Detect objects` sa načítaný obraz prahuje podľa zvolenej metódy a dvoj-priechodovým algoritmom sa označia jednotlivé objekty. Užívateľ má na výber z dvoch metód prahovania – globálneho a adaptívneho. Pri výbere adaptívneho je možné zmeniť veľkosť okolia pixelu, z ktorého sa počíta lokálny prah (veľké okolie spôsobí dlhú dobu výpočtu, najme pri snímkoch vo vysokom rozlíšení).

Po stlačení tlačítka `Find` sa zobrazí dialógové okno, kde je možné vybrať hľadaný objekt. Zoznam možných hľadaných objektov je určený jednotlivými triedami načítaného klasifikátora. Po potvrdení výberu sa v načítanom snímku hľadá zadaný objekt. V prípade, že sa tento objekt nenájde, užívateľ je o tom informovaný dialógovým oknom. V opačnom prípade sú nájdené objekty v obraze zvýraznené obdĺžnikom. Pre výber konkrétneho objektu

je potrebné myšou kliknúť na daný objekt. Po kliknutí sa v pravom dolnom rohu zobrazia vypočítane vlastnosti tohto objektu.

Behom vykonávania rôznych krokov je možné obrázok znova načítať stlačením tlačítka `Reset`.