FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

KIV/ZVI

PŘÍZNAKOVÉ ROZPOZNÁVÁNÍ OBJEKTŮ VE SNÍMKU

Martin Hamet

30. listopadu 2017

Obsah

1 Zadání

Nad zadanými obrazovými daty ve formátu BMP, 512 x 512 obrazových bodů, 256 úrovní šedé, v souborech IMG.BMP proveďte úlohy:

- Zvolenou metodou segmentace obrazu nalezněte všechny objekty ve snímku, určete jejich počet a jednotlivé objekty označte (např. obarvením plochy).
- Určete vnitřní hranice jednotlivých objektů, vypočítejte obvod a těžiště pro každý objekt.
- Metodami příznakového rozpoznávání nalezněte zadané objekty (písmena a číslice zadané z klávesnice), příznakový vektor může obsahovat např.: polohu těžiště, obvod, plochu opsaného čtyřúhelníka, počet děr, členitost hraniční čáry, ...
- Testování bude prováděno na natočených objektech a po provedení opakované dilatace, resp. eroze (do úrovně, kdy ještě nebude narušena topologie objektů) a po rozostření obrazu průměrováním v okolí 5 x 5.
- Procedury pro dilataci, erozi a průměrování snímku budou součástí programové realizace.

2 Analýza problému a návrh řešení

Úkolem je demonstrace jednotlivých metod a jejich použití pro klasifikaci objektů ve snímku. Úlohu lze rozdělit tři hlavní části segmentaci, parametrizaci a klasifikaci. Segmentací získámé jednotlivé prvky obrazu. Pomocí parametrizace zajistíme jednotný popis individuálních objektů, který bude možné dále klasifikovat pomocí klasifikátoru.

Protože se jedná o demonstrační úlohu, bylo by vhodné konstruovat ji s vědomím že by se mohla v budoucnu rozšiřovat o další metody (klasifikátory, způsoby segmentace etc.).

2.1 Segmentace

Segmentace se zabývá identifikací objektů ve snímků obvykle na základě nějaké vlastnosti, která je objektu vlastní. Objektem se v našem případě myslí znak (tvar) v obrazu, který se vyznačuje vysokou hodnotou jasu vůči svému okolí. Tedy světlé objekty na tmavém pozadí. Jednotlivé pixely můžeme tedy označit za součást objektu pokud jejich hodnota jasu překročí jistou hranici.

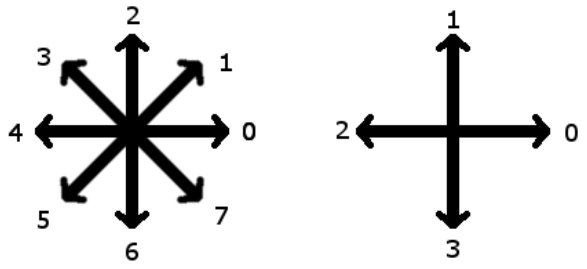
Pro učení prahové hodnoty lze použít automatické algoritmy, nicméně vzhledem k povaze našich vstupních dat, kde jsou obrázky stejného formátu s podobným jasovým rozložením, prahovou hodnotu určíme experimentálně a nebude ji třeba nadále upravovat.

K vlastnímu oddělení objektů od pozadí lze použít různě metody např. spojováním oblastí, maticí sousednosti, etc. V našem případě se jednotlivé objekty nekříží, ani nejsou vnořené (objekt obsahující jiný objekt). Nejsme tolik omezeni komplexností snímku a můžeme použít méně obvyklý způsob založený na Freemanově řetězovém kódu (viz 2.1.1). Tento způsob nám zajistí nalezení celého objektu podle jeho vnitřní hranice.

2.1.1 Freemanův řetězový kód a jeho využití pro segmentaci

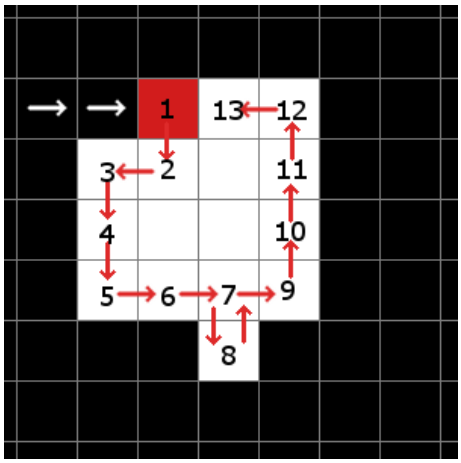
Freemanův řetězový kód Jedná se obecně o způsob popisu spojité křivky respektive posloupnosti jejích jednotlivý bodů. Lze uvažovat o sousednosti typu `D` (4 - okolí) nebo `DI` (8 - okolí). Protože nemáme žádné zvláštní požadavky na popis hranice použijeme `D` okolí pro hledání sousedních bodů.

Popis je založen na popisu směru ve kterém se nachází další sousední bod. Z praktického pohledu není třeba uchovávat informace o každém bodu hranice, ale pouze o počátku a dále jen informaci o směru k dalšímu bodu. Použitím omezeného počtu směrů (`D` okolí) se zabýváme pouze čtyřmi směry, které můžeme zakódovat jednocifernou číslicí (viz obr.1).



Obrázek 1: Popis směru ve Freemanově řetězovém kódu `DI`(vlevo) `D`(vpravo).

Využití pro segmentaci Pro nalezení objektu využijeme Freemanova kódu tak, že budeme procházet jednotlivé body obrazu a při nalezení bodu, který lze považovat za součást objektu (viz. 2.1), přejdeme na problém popisu vnitřní hranice objektu.



Obrázek 2: Pohyb po vnitřní hranici objektu.

Chceme-li popisovat vnitřní hranici musíme stanovit pravidla pro hledání následujícího bodu tak, aby tento bod byl také bodem hranice. Pokud tedy budeme procházet obrázkem zleva doprava a shora dolu tak nevyhnutelně prvním nalezeným bodem bude levý horní roh objektu 1(viz obr. 2). Budeme postupovat proti hodinovému směru. V průběhu průchodu si budeme držet informaci v jakém směru jsme našli aktuální bod. Pro 1 si určíme směr dolů (směr č. 3 viz obr. 1). Nový bod budeme hledat vždy v pořadí očíslování směrů a začneme o jeden směr zpět od směru ze kterého jsme nalezli aktuální bod. Například bod 3 jsme nalezli ve směru doleva (2) tedy kontrolujeme sousedy v pořadí (horní (1), levý (2), dolní (3) a pravý (4)) a označíme dolního souseda jako další hraniční bod 4 nalezený ve směru dolů (3). Tímto způsobem máme zajištěný průchod všech hraničních bodů nalezeného objektu a tím i definováním jeho umístění ve snímku.

2.1.2 Označení zpracovaných bodů

Při výše popsaném průchodu snímkem by docházelo k opakovanému průchodu již nalezených objektů, proto je nutné všechny body, které byli označeny jako součásti objektů (včetně vnitřních) označit jako již zpracované.

Pro nalezení vnitřních bodů daného objektu můžeme využít samotného průchodu po hranici a ukládat indexy nalezených hraničních bodů. Protože předpokládáme že nedochází ke křížení objektů, ani k jejich vnořování můžeme si dovolit ukládat pouze minimální a maximální index (Například pro třetí řádek v obr. 2 by jsme si uložili index prvku 3 při jeho průchodu a index prvku 11.

Takto získáme popis objektu podobný `RLC` s tím rozdílem že máme pouze počáteční a koncový index pro jednotlivé řádky. Všechny body mezi těmito indexy, které lze považovat za součást objektu, patří tomuto objektu a můžeme je tedy označit za nalezené.

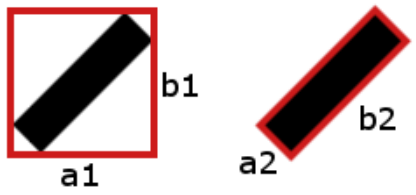
2.2 Parametrizace

Parametrizace slouží jako jednotný popis vybraných vlastností objektu, který se dá dále použít pro klasifikaci. Pro snadné uložení a zpracování těchto vlastností bude vhodné použít tzv. parametrizační vektor kde jeho složky jsou hodnoty odpovídající jednotlivým vlastnostem daného objektu.

Efektivita (popisová síla) jednotlivých vlastností se těžko určuje a je silně závislá na povaze objektů, které chceme popisovat. Protože objekty mohou mít náhodnou rotaci nemá smysl volit spojené s pozicí ve snímku, což značně sníží popisovací a tím i rozlišovací schopnost aplikace.

2.2.1 Minimální opsaný čtyřúhelník

Parametry opsaného obdélníku získáme snadno, pokud si uložíme minimální a maximální souřadnice respektive jejich složky `X, Y` jednotlivých bodů hranice. Body obdélníku budou: `[min X, min Y]`, `[min X, max Y]`, `[max X, max Y]`, `[min X, max Y]`.



Obrázek 3: Minimální opsaný obdélník.

Chceme-li minimální opsaný obdélník musíme vzít v úvahu úhel otočení objektu (viz obr. 3). Všechny údaje potřebné pro takový čtyřúhelník lze získat například rotací bodů na hranici se zvoleným krokem (nejlépe násobením transformační maticí) a hledání minimálních a maximálních souřadnic jak bylo popsáno dříve. Porovnáním obsahů takto získaných obdélníků získáme přibližně minimální s přesností podle zvoleného kroku.

Protože minimální opsaný čtyřúhelník není závislý na rotaci je to vhodný výchozí bod pro množství dalších parametrů objektu.

2.2.2 Podlouhlost

Z minimálního opsaného čtyřúhelníka (viz 2.2.1) můžeme snadno získat tzv. podlouhlost , která udává poměr jeho delší a kratší strany `b2/a2` dle obr. 3.

2.2.3 Obsah

Obsah objektu lze spočítat pomocí všech bodů, které danému objektu patří, respektive jejich jasů. Lze využít vážený obsah, kdy budeme brát v úvahu úroveň jasu v každém pixelu, nebo pouze pixelů patřících objektu. Protože jsou jasy vstupních dat víceméně uniformní budeme uvažovat druhou možnost. Aby jsme snížili závislost na velikosti objektu bude vhodnější získaný obsah vydělit obsahem minimálního obdélníku (viz 2.2.1). Získáme tím více univerzální popis než samotným obsahem.

2.2.4 Težiště

Těžiště lze počítat vážené, nebo brát všechny body patřící objektu se stejnou vahou podobně jako obsah (viz 2.2.3). Pokud vezmeme nevážené těžiště jedná se o průměrné souřadnice všech bodů patřících objektu. Aby jsme snížili závislost na pozici a rotaci objektu vztáhneme souřadnice těžiště vůči minimálnímu opsanému čtyřúhelníku. Toho docílíme rotací těžiště v úhlu rotace obdélníku (rotujeme podle stejného bodu jako při hledání obdélníku např. střed bitmapy) a odečtením hodnot souřadnic např. levého horního rohu.

Dále protože je možné najít minimální čtyřúhelník ve čtyřech možných úhlech rotace bude vhodné vztáhnout souřadnice těžiště ještě k rozměrům obdélníku ve smyslu kratší a delší strany v daném pořadí. Výsledné dvě hodnoty tedy získáme z rotovaného bodu těžiště:

$$d_1 = \frac{x - xr}{Min\{w, h\}} \qquad d_2 = \frac{y - yr}{Max\{w, h\}} \tag{1}$$

Kde $[x, y]$ jsou souřadnice rotovaného těžiště, $[xr, yr]$ jsou souřadnice levého horního rohu minimálního obdélníku, w a h jsou jeho šířka a výška.

2.2.5 Nejdelší spojnice

Jedná se o nejdelší rozměr mezi dvěma body hranice v objektu. Tento údaj lze získat porovnáním všech dvojic bodů na hranici. Aby tento údaj nebyl přímo závislý na velikosti objektu vztáhneme ho ke kratší (delší) straně minimálního obdélníku vydělením touto hodnotou.

2.2.6 Kruhovitost

Udává jakousi podobnost zkoumaného objektu a kruhu. Form factor spočítáme podle následujícího vztahu:

$$F = 4\pi \frac{S}{O^2} \tag{2}$$

Kde S je obsah objektu a O jeho obvod. Obvod získáme jednoduše, protože máme hranici objektu spojitou ve smyslu `D` okolí obvod je tedy počet prvků hranice.

2.2.7 Počet děr

Jako díru uvažujeme spojitou oblast, ve smyslu `D` okolí, která je plně obklopena objektem. Jinými slovy nedotýká se hranice snímku. K nalezení takových oblastí lze využít více způsobů. Protože zpracováváme relativně malé obrázky (objekty) můžeme využít rekurentního přístupu. Jeho hlavní výhodou je jednoduchost. Nevýhoda vzniká při zpracování většího snímku může nastat problém s nedostatečnou velikostí zásobníku.

Hledání spočívá v rekurzivním průchodu všemi pixely snímku, kde každý nově objevený pixel označíme jako nalezený (nebude se znovu zpracovávat) a pokud narazíme na pixel na hranici snímku zavrhneme celou nalezenou oblast, která tedy není dírou. Dále vybereme další nenalezený pixel a pokračujeme v průchodu. Pokud se tedy rekurentním pohybem ve smyslu `D` okolí z počátečního bodu nedá dostat na hranici snímku jedná se o díru.

Počet děr objektu je závislý pouze na spojité topologii objektu a je tedy ideálním parametrem pro popis. Nemá bohužel žádnou popisovací (rozlišovací) schopnost mezi znaky (objekty) bez děr.

2.3 Klasifikace

Při parametrizaci jsme získali vektor pro každý objekt, který ho podle zvolených parametrizačních metod popisuje. V klasifikační části potřebujeme určit do jaké klasifikační třídy zařadit vektor na základě trénovacích dat, kde jsme znali správný výsledek. Je tedy třeba klasifikátor připravit (vytvořit model) v trénovací části tak, aby s co největší přesností určil správnou třídu bez znalosti výsledku. Klasifikátor musí být také schopen takto vytvořený model uložit a případně později načíst, proto budeme volit jednoduché datové struktury. Pro účely ukládání bude vhodnější formát XML místo TXT vzhledem k přehlednosti uložených dat.

2.3.1 Etalonový klasifikátor

Tento klasifikátor nejprve načte všechna trénovací data a poté je zpracuje a vytvoří vzorové parametrizační vektory pro každou klasifikační třídu. Při testování už pouze počítá vzdálenost vzorů (etalonů) od testovaného vektoru (ze zkoumaného objektu). Podle etalonu s nejkratší vzdáleností od testovaného zařadí testovaný vektor do příslušné třídy. Jedná se o jednoduchý klasifikátor, kde ovšem záleží na správném vytvoření etalonů.

Nejjednodušší způsob určení etalonů je průměr v dané klasifikační třídě z vektorů vytvořených z trénovacích dat. Výhodou je že po trénovací fázi není nutné uchovávat všechna trénovací data, ale pouze výše zmíněné etalony a je tedy tento klasifikátor paměťově výrazně úsporný.

Pro vypočítání vzdálenosti v našem případě příznakových vektorů můžeme použít Euklidovskou vzdálenost. Je důležité aby se po celou dobu funkce tohoto klasifikátoru používal stejný parametrizační algoritmus, který produkuje vektory o stejné délce což je zajištěno parametrizací (viz 2.2).

3 Implementace (popis řešení)

V této sekci budou popsány hlavní moduly programu a jejich vzájemná funkce. Třídy a procedury nutné pro vykreslování a práci s GUI nebudou popisovány vzhledem k jejich nevýznamnosti vůči řešenému problému. Nicméně jsou nezanedbatelnou součástí aplikace.
(třídy `Form1` hlavní okno a `FragmentDetailForm` okno detailu objektu)

3.1 `Program`

Zajišťuje hlavní řízení programu.

`main()` Vstupní bod programu vytvoří příslušné prvky konzole, GUI a předá řízení hlavnímu oknu `Form1`.

`evaluateBitmap()` Vyhodnotí bitmapu, všechny objekty které obsahuje a vyplní jejich příslušné popisy. Výsledky jsou zobrazeny v hlavním okně aplikace.

3.2 Image

Jedná se o obalovou třídu, která shromažďuje veškeré informace o snímku a poskytuje funkce pro práci sním a objekty které daný snímek obsahuje.

`process()` Hlavní metoda volaná z konstruktoru zajistí průchod bitmapy snímku a nalezení objektů které obsahuje použitím metody `processByFreeman()`. Základní informace o jednotlivých objektech jsou uloženy do jejich instancí `Image.Fragment`.

`openBitmap()` Zajišťuje umožnění přístupu přímo k datům bitmapy. Je nutné po skončení práce volat metodu pro uzavření `closeBitmap()`.

3.2.1 Image.Fragment

Jedná se o obalovou třídu obsahující veškeré informace o nalezeném objektu (body vnitřní hranice, počátek, rozměry, etc.).

3.3 Description

Obalová třída pro detailní informace o objektu vyplněné především při generování parametrizačního vektoru. Slouží pro předání informací o objektu do GUI.

3.4 DefaultDescriptor

Slouží pro generování parametrizačního vektoru pro daný objekt.

`getDescription()` Vytvoří parametrizační vektor a nastaví všechny příslušné údaje objektu v obalové třídě `Description`. Vektor je dále vynásoben váhami jednotlivých parametrů po složkách.

`setAttributes()` Nastaví atributy minimálního opsaného čtyřúhelníka v `Description`, které budou potřeba pro generování parametrizačního vektoru.

3.5 DistanceClassifier

Reprezentuje etalonový klasifikátor. Přidávané trénovací vektory ukládá do seznamů ve slovníkové struktuře tak, že klíčem je správná klasifikační třída. Při prvním zavolání metody `evaluate()` se projdou všechny trénovací vektory metodou `compileReadings()` a vytvoří se příslušné etalony do seznamu `finalPatterns`. Při ukládání modelu zanikají informace o trénovacích vektorech a ukládá se pouze seznam etalonů.

`addTrainCase()` Přidá do klasifikátoru nový vektor se správnou třídou. Pomocí opakovaného volání této metody se trénuje klasifikátor.

`evaluate()` Vytvoří vektor (descriptorem klasifikátoru) z předaného objektu `Image.Fragment` a klasifikuje ho. Metoda vrací řetězec představující výsledek klasifikace (odpovídající klasifikační třídu).

`saveModel()` Uloží natrénovaný model do souboru (`Model.data`) ve formátu `XML` pro pozdější použití.

4 Uživatelská příručka

Pro správný běh programu je nutné, aby se spustitelný soubor nacházel ve stejném adresáři jako složka s testovacími daty, případně uložený soubor `Model.data`. Dále program vyžaduje nainstalovaný `Microsoft.NET Framework`.

4.1 Spuštění programu

Program je samostatně spustitelný a běžná komunikace probíhá pomocí uživatelského rozhraní (viz 4.1.2). Program také lze spustit z příkazové řádky pro některé stavové výpisy, ale není to z běžného pohledu nutné.

4.1.1 Formát uložených dat

Vstupní snímek Očekávaný velikost vstupních obrázků je `512x512` pixelů. Není nezbytně nutné tento parametr dodržet, ale s většími obrázky značně roste riziko spadnutí programu kvůli použití rekurzivního řešení a tedy vysokých nároků na paměť. Formát pixelů vstupního snímku by měl být `8bppIndexed` (8bitů na pixel s indexací).

Trénovací data Musí být uložena tak, že složka `TrainingData` obsahuje podsložky s názvy správných klasifikačních tříd, aby bylo jasné co za třídu klasifikátor právě trénuje (tyto názvy budou použity jako výsledky klasifikace). V těchto složkách se nacházejí jednotlivé snímky příslušící dané třídě.

Datový model Model je uložený ve formátu `XML` jako soubor `Model.data`. V souboru jsou uložené informace o použitém klasifikátoru a descriptoru. Dále následuje datový blok `<patterns>`, kde jsou uloženy jednotlivé ethalony parametrizačních vektorů pro každou klasifikační třídu. Parametrizační vektor je složen s hodnot které odpovídají parametrům vektoru pořadí: {obsah, podlouhlost, těžiště-1, těžiště-2, sppojnice-1, spojnice-2, kruhovitost, počet děr}. Jednotlivé hodnoty jsou vynásobeny složkami váhového vektoru {1, 2, 2, 2, 0.5, 0.5, 2.5, 3.5}.

4.1.2 Grafické uživatelské rozhraní GUI

Hlavní okno Po spuštění GUI je okamžitě možné sním pracovat. Nejprve je nutné natrénovat klasifikátor pomocí tlačítka `Train Classifier`, nebo načíst již hotový ze souboru `Load Classifier`.

Po vytvoření (načtení) klasifikátoru se zpřístupní možnost otevření vstupního snímku `Open Image`. Otevřený snímek bude zobrazen v hlavním panelu. S otevřeným snímkem je možné pracovat pomocí tlačítek:

- `Blur 5x5` Rozmazání průměrováním pomocí masky o rozměrech `5x5`.
- `Dilatation` Dilatace obrázku maskou `D` okolí.
- `Erosion` Eroze obrázku maskou `D` okolí.
- `Step back` Zajišťuje funkci návratu o jeden krok zpět.
- `Evaluate` Vyhodnotí obrázek a zobrazí v nalezené objekty jako výslednou klasifikační třídu a souřadnice levého horního rohu opsaného čtyřúhelníka.
- `Detail` Otevře nové okno s detailem vybraného objektu ze seznamu.

Okno detailu Okno je určené pro zobrazení detailů jednoho objektu ze snímku. Těchto oken je možné mít otevřeno více. V okně se nachází výpis některých důležitých parametrů objektu a pomocí tlačítek lze některé zobrazit přímo v objektu.

- `Show Original` Zobrazí originální výřez z původního snímku.
- `Show Minimal Rectangle` Vykreslí minimální opsaný čtyřúhelník oranžovou barvou.
- `Show Edge` Zvýrazní červeně vnitřní hranici objektu.
- `Show Center of Gravity` Oranžovým křížem naznačí těžiště objektu.

5 Testování

Testování jednotlivých parametrizačních metod a algoritmu segmentace není dokumentovaná vzhledem k jednoduchému ověření pomocí přiložených obrázků ve složce `TestIndividual` a zobrazením detailu v uživatelském rozhraní (viz 4.1.2).

5.1 Testování klasifikátoru

Testování klasifikace bylo provedeno na zadaných datech s využitím metod pro dilataci a rozmazání. Ve snímku se nachází 15 objektů úspěšnost klasifikace je tedy vztažená k tomuto počtu. Natočený snímek byl vždy klasifikován po použití funkce rozmazání.

`Úroveň dilatce`	`Úspěšnost`	`Úroveň eroze`	`Úspěšnost`
1	89%	1	93%
2	83%	2	86%
3	66%	3	73%
4	63%	4	56%
5	63%	5	49%
6	56%	6	49%

6 Závěr

Program je připraven pro další rozšiřování o klasifikátory a parametrizační algoritmy. Zvolené algoritmy byly vybrány tak, aby byly snadno implementovatelné a dali se dále rozšiřovat. Vzhledem k tomu že vstupní formát pixelů (`8bppIndexed` není standardně podporován základními funkcemi, bylo nutné pracovat na mnohem nižší úrovni abstrakce, přestože pracujeme s relativně malými snímky.

Jednotlivé parametrizační metody byli snadno ověřitelné pomocí uměle vytvořených snímků (o několika pixelech). Samotná klasifikace není příliš efektivní především kvůli jednoduchosti zvoleného klasifikátoru, absenci rozsáhlých trénovacích dat a pokročilejších trénovacích algoritmů. Přesto se podařilo dosáhnout alespoň částečné klasifikační schopnosti aplikace. Kvůli rotaci vstupních snímků nebylo možné využít vyzkoušené metody (např. histogram orientovaných gradientů), které silně závisí na pozici a jejich zobecněné verze se neprojevili jako příliš efektivní (alespoň v jejich jednoduché verzi).

Testování klasifikace bylo provedeno na zadaných datech s využitím metod pro dilataci a rozmazání.