СУ „СВ. КЛИМЕНТ ОХРИДСКИ”

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

Автореферат

по

Разпознаване на образи

**Тема №7**
“ Методи за извличане на графични признаци. Оконтуряване. Скелетизация. Изтъняване. Нормализация. Векторизиране. Верижни кодове. Трансформация на разстоянията “

изготвил: Владимир Ивов Блажев

ФН: 23721

ИИ, 1 курс

София

2012

**1. Методи за извличане на графични признаци**

Извличането на графични признаци (feature extraction) включва използването на алгоритми за откриване и изолиране на различни желани части от изображението. Това могат да са линии, криви, окръжности, елипси, контур и т.н. Обикновено графичните признаци се представят чрез т. нар. вектори на признаците (feature vectors). В повечето случаи, извличането на графични признаци по подразбиране се асоциира и с редуцирането на размерността на тези вектори, но по такъв начин, че загубата на информация да е минимална. Един такъв метод е PCA (Principal Component Analysis) или метод на главните компоненти, но той е общ, а не насочен към извличане на специфични графични признаци. За разлика от него, например, трансформацията на Хав (Hough) се използва за намиране на контур, прави линии, елипси и окръжности, а обобщената Хав трансформация - за откриване на произволни форми. Линейният филтър по Sobel също може да бъде използван за намиране на контур, а също така и двойно диференциращият филтър на Лаплас. За оконтуряване могат да се използват и верижни кодове, алгоритъм на бръмбара и други. Можем да използваме Фурие или уейвлет трансформация, за да получим изображение (в спектралната област), което е независимо от транслацията на входното.

**2. Оконтуряване**

Контур на едно свързано множество от пиксели $R$ ще наричаме множеството от пиксели $K$ ($K⊂R$), всеки от които има най-малко един съсед, който е вътрешен за $R$.

Първият метод за откриване на контур, който ще разгледам е т.нар. „алгоритъм на бръмбара” [1]. За целта, изображението, което ще използваме, трябва да е бинаризирано. Алгоритъмът започва от един пиксел, който не е част от фона (най-често приемаме, че черните пиксели са част от съществената част от изображението, а белите от фона). За да намерим такъв пиксел, можем да започнем обходждаме изображението движейки се от горе-надолу и от ляво надясно. Алгоритъмът приключва когато следващият пиксел, който трябва да обходим е началният, т.е. получим затворен контур. Следва описание на стъпките му:

*Обозначения:* А – първата точка, която сме открили, че принадлежи на контура на *R*.
C – текущата точка, чиито околности изследваме в момента. S – посоката на търсене на следващия пиксел (Фиг. 1 дава разяснения каква е поредността на околните точки на някаква точка ***p***). ***първи* –** флаг, който има стойност „истина ” само в началния момент. ***намерен*** – флаг, който има стойност „истина”, само когато намерим поредната точка от контура.

1. Намираме такава точка А от контура, че нейният съсед на позиция 4 да не принадлежи на *R*, т.е. точката А се намира отляво на множестовото *R.*
2. На С се присвоява стойността на А, на посоката S се присвоява стойност 6, т.е. надолу, а на флага ***първи*** стойност „истина”
3. **While** C $\ne $ A или флагът ***първи*** има стойност „истина” **do** стъпки от 3 до 10:
 **Begin**
4. ***намерен*** = „лъжа”
5. **While** ***намерен*** = „лъжа” **do** стъпки от 5 до 9 (но не повече от 3 пъти):
 **Begin**
6. **If** В = (S - 1)-я съсед на C принадлежи на *R*, **then**:
 **Begin**
7. На C се присвоява стойността на В, S = 5, стойност S-2, а
флагът  ***намерен*** = „истина”
8. **Else If**В = S-я съсед на C, който принадлежи на множеството *R*, **then** на C се присвоява стойността на В, а нафлагът  ***намерен*** - „истина”
9. **Else If**В = (S + 1)-я съсед на C, който принадлежи на множеството *R*,
 **then** на C се присвоява стойността на В, а нафлагът  ***намерен*** - „истина”
10. **Else** стойносттна на S се увличава с 2
 **End**
11. На флага ***първи*** се присвоява стойност „лъжа”
 **End**
12. Край на алгоритъма


 Фиг. 1

След като сме открили един контур може да продължим да изграждаме т.нар. Tree of Contours (ToC) или дърво на конурите. За целта можем да подходим както в дълбочина, така и в широчина. Ако подходим в дълбочина, то първо трябва да изградим ново работно изображение от току-що намереното, в което да търсим съдържащи се контури. Ако подходим в широчина, то трябва да намерим друг, непресичащ се с текущо намерения контур. На Фиг. 2 е показано примерно дърво на контурите за съответната картинка.

Друг метод за оконтуряване е като се използва диференциращият филтър по Sobel. За контура може да се мисли като за мястото с най-голям скок в интензитета на изображението, т.е. максималният градиент. За целта нелинейният филтър по Sobel използва две 3х3 ядра, като едното се използва за диференциране по хоризонтала, а другото по вертикала. Изчислява се съответната конволюция на двете ядра с изображението ($f(x, y)$), при което се получават:

 $\vec{F\_{x}}=f\left(x, y\right)\* S\_{x}$

 $\vec{F\_{y}}=f\left(x, y\right)\* S\_{y}$

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -2 | 0 | 2 |
|  -1 | 0 | 1 |

Ядро за хоризонтално диференциране $S\_{x}$

|  |  |  |
| --- | --- | --- |
| -1 | -2 | -1 |
| 0 | 0 | 0 |
|  1 | 2 | 1 |

Ядро за вертикално диференциране $S\_{y}$

Максималният скок в интензитета, можем да изчислим по формулата:

 $\left|grad\left(f\left(x,y\right)\right)\right|= \sqrt{F\_{x}^{2}+ F\_{y}^{2}}$

**3. Скелетизация**

Скелетизация е процес на трансформация на даден компонент от едно дигитално изображение в някакво негово подмножество. За скелета на едни елемент от дадено изображение можем да си мислим като за „линията на гасене на пожара”, т.е. ако запалим границите на елемента, то скелетът е мястото където огънят ще се събере (и изгаси). Мотивите за намиране на „скелет” на даден компонент са няколко. По този начин намаляваме броят на точките, които трябва да обработим при евентуална манипулация с изображението. Например, извличането на характеристики (features) от избражението би било по-лесно, ако се извърши върху неговия „скелет”, вместо върху оригиналното изображение. Също така, по този начин могат да се елиминират локални ефекти на зашумяване на изображението.

1. **Базирани на трансформация на разстоянията.**Трансформациите на разстоянията ще бъдат разгледани по-подробно по-нататък в изложението. Идеята на метода се състои в изграждането на скелет на базата на множество от *симетрични точки* (*symmetric points*). В едно затворено Евклидово пространство една точка **р** се нарича симетрична тогава и само тогава, когато съществуват поне 2 точки на границата на това пространсто, които да са равно отдалечени от **р**. За всяка симтерична точка, максималният й диск е най-големият диск в това множество. Трансформацията на разстоянията се използва за намирането на тези симетрични точки. Множеството от точки, означени със съответните им максимални радиуси на максималните им дискове, съставляват скелета на множеството.

 
 Фиг. 2 Фиг. 3
2. **Базирани на „критичните точки”**Този алгоритъм определя средните точки на всички подмножества от свързани компоненти, като сканира пикселите на всеки ред. Нека *l* е броят на свързаните компоненти за даден ред на оригиналното изображение. Тогава резултатът от обхождането на ред *i* е множество от координата $(i, m\_{i}(l))$ от средни точки на свързаните компоненти в реда *i*. Множеството от всички точки за всички редове сформира т.нар. „скелет на критичните точки”. Методът е ефективен по сложност. Има опасност обаче резултатното множество от точки да не е свързано.
3. **Базирани на итеративно изтъняване**Изтъняването в обработката на изображения е техника, която запазва свързаността на компонентите. Обикновено се имплементира като итеративен процес, при който някои от точките от контура на компонента се превръщат в точки от фона. Резултатът от алгоритъма за изтъняване може да се дефинира като *идеално тънък*, когато нито една точка, която не е крайна, не може да бъде изтрита, без това да развали свързаността на компонента.


 Фиг. 4

**4. Нормализация**

Нормализацията е проста техника в обработката на изборажения, чиято цел е да подобри контраста в едно изображение като „разтегли” стойностите на интензитетите на неговата хистограма до някакви нови желани стойности. Например, ако едно полутоново изображение има стойности само в интервала [50, 180], то възможно е да пожелаем да нормализираме неговите стойности като ги „разтеглим” до максималния интервал [0, 255]. Нормализацията е алгоритъм, който е много сходен с алгоритъма за изравяване на хистограмата (histogram equalization), но за разлика от него тук можем да приложим само *линейна* скалираща функция към интензитетите на пикселите.

Нормализацията е алгоритъм, който изисква два входни параметър – минималната и максималната стойности на новия интервал, в който искаме да се разпростират стойностите на хистограмата – [newMin, newMax]. Нормализираното изображение можем да изчислим чрез формулата:

$$I\_{N}=\left(I-Min\right)\frac{newMax-newMin}{Max-Min}+ newMin$$

 $I\_{N}$ е резултатното изборажение, $I$ e входното изображение, а $[Min, Max]$ е интервалът, в който се намират интензитетие на входното изображение.

Един от проблемите на този алгоритъм е, че е силно зависим от единични много тъмно или много светли точки от входното изображение. Например, ако полутоновата хистограма на изображението се намира в интервала [50, 180], но има един пиксел (поради шум, например), който има стойност 240, то разликата $Max-Min$ няма да е 130, а 190, което значително ще „изкриви” нормализацията. Едно възможно решение на този проблем е да първо да изчислим хистограмата на входното изображение и след това да елиминираме някакъв процент от интензитетите, например 5 %. Това означава, че 5 % от пикселите, които формират хистограмата, ще има реално по-малка стойност от $Min$, а също така 5 % от пикселите ще има по-голяма стойност от $Max$.

**5. Векторизиране**

Векторизирането е процес, при който едно растерно изображение се превръща във векторно. Такъв процес можем да извършим, например, когато сме сканирали едно изображение и искаме да го превърнем от растерно във векторно. За целта трябва да се извършат няколко стъпки. Обикновено първо се започва с откриване на линиите в оригиналното изображение. Най-често това се извършва като се намери скелетът на растерното изображение. Това можем да извършим, като използваме описаните по-горе техники. След това трябва да апроксимираме откритите вече линии в някакво множество от вектори, като за целта можем да използваме някакъв метод за полигонално апроксимиране (т.е. разбиваме една двуизмерна линейна крива на дискретен брой отсечки). След тази стъпка опционално може да се извърши някаква оптимизация. Например, определени вектори могат да бъдат сляти, тъй като имат една и съща посока и т.н.

**6. Верижни кодове**

Верижните кодове се използват, за да представят определени компоненти от дадено полутоново изображение. Това представяне може да се използва както за компресия на изображението, така и за разпознаване на описания компонент. За целта, се избира една точка от границата на обекта и се запаметяват нейните координати. След това се обхожда целият контур на свързания компонент, като се запаметява само посоката, в която се намира следващият пиксел. Този метод е ефективен за изображения, които съдържат малък брой, но големи по площ свързани компоненти. Важна особеност на верижните кодове е, че те лесно могат да се получат като „вторичен продукт” при оконтуряването на дадения обект. При разпознаване обаче, за да се запази инвариантност по местоположение на обекта, координатите на началната точка не трябва да се запаметяват. Проблемът обаче е в това, че ако преместим началната позция с един пиксел дори, то цялата верига също ще се „завърти” с една позиция наляво или надясно (в зависимост от това кой от двата съседни пиксела на началния сме избрали). За целта, веригата от кодове трябва да се нормализира. Да разглеждаме веригата от кодове като единствено цяло число. Така за начална точка можем да изберем този пиксел, при който от веригата се получава най-малкото число. Един от недостатъците на верижните кодове е тяхната зависимост зашумяване и трансформации като ротацията или скалиране на свързаните компоненти.

**7. Трансформация на разстоянията**

Трансформацията на разстоянията е глобална техника, която открива най-краткото разстояние от даден пискел до някакъв елемент (или препятствие, в зависимост от контекста) от изображението. За да приложим алгоритъма, първо трябва да разполагаме с бинаризирано изображение, в което (например) черните точки са част от изображението, а белите – от фона. След това трябва да изберем структурен елемент, с който да обходим изображението. Центърът на тази маска се поставя върху всеки пиксел. Локалното разстояние във всеки пикел от маската се събира със стойността на пиксела „под” него (включително централната 0). Новата стойност на точката е минимумът от всички суми. Този алгоритъм се повтаря докато стойността на нито един пиксел не се промени, т.е. броят на итерациите е пропорционален на най-голямото растояние в изображението. На картинките по-долу са дадени примери за различните стойности на трансформацията на разстоянията при избор на различен структурен елемент.

 
 Фиг. 5 Фиг. 6

**Литература:**

[1] <http://en.wikipedia.org/wiki/Feature_extraction>

[2] Павлидис Т.: Алгоритмы машинной графики и обработка изображений. "Радио и связь", М., 1986.

[3] <http://www.inf.u-szeged.hu/~palagyi/skel/skel.html>

[4] Tombre K., Tabbone, S. : Vectorization in Graphics Recognition: To Thin or not to Thin

[5] Klette G. : Skeletons in Digital Image Processing

[6] [http://en.wikipedia.org/wiki/Normalization\_(image\_processing)](http://en.wikipedia.org/wiki/Normalization_%28image_processing%29)

[7] <http://cs.joensuu.fi/~koles/approximation/Ch3_1.html>

[8] <http://en.wikipedia.org/wiki/Chain_code>

[9] <http://en.wikipedia.org/wiki/Distance_transform>

[10] Borgefors G.: Distance Transformations in Digital Images