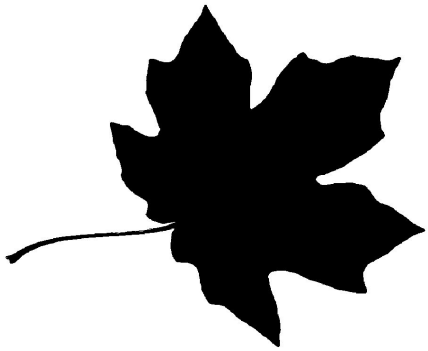
Representatie en beschrijving van randen en regio's

Kristof Teelen

Labo beeldverwerking / 2007-08

1 Inleiding

Er bestaan veel methodes om een representatie en een beschrijving van objecten te creëren. Randen en aaneensluitende regio's zijn op zich al een beschrijving van het object. In dit labo zien we verschillende methodes om randen en regio's te representeren/beschrijven. Bekijk eerst het voorbeeld van het berekenen van een signatuur, en implementeer dan zelf één van de volgende beschrijvingen, ofwel door kettingcodes of ofwel met een polygonale benadering.



Figuur 1: Binair beeld van een blad.

Om de randpunten van het blad te berekenen, kan je gebruik maken van de volgende code:

```
im = imread('leaf.bmp');
[B,L] = bwboundaries(1-im2bw(im),'noholes');
figure, imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
for k = 1:length(B),
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end
hold off
```

Nu heb je alle punten van het blad beschikbaar in een cell-array. Om deze randpunten dan te herschikken zodat je een lijst met de punten in kloksgewijze volgorde krijgt, moet je volgende stuk code uitvoeren:

```
B = bwtraceboundary(1-im2bw(im),B{1}(1,:),'N',8,Inf,'clockwise');
```

Een andere manier om objecten te beschrijven is aan de hand van het skelet, zoals we al eerder hebben gezien in het labo morfologie. Een klein voorbeeldje:

```
BWm = bwmorph(1-im2bw(im),'skel',Inf);
figure, imshow(BWm)
```

2 Signaturen

Een signatuur is een 1D functionele beschrijving van een objectrand. Een simpele manier om een rand te beschrijven is de straal uit te zetten in functie van de hoek ten opzichte van de horizontale. Bereken eerst een oorsprong voor het assenstelsel waarin de hoek en straal berekend worden: kies hiervoor het zwaartepunt van alle randpunten van het object dat je wil beschrijven:

```
cengrav = sum(boundary,1)/size(boundary,1)
figure, hold on,
plot(boundary(:,2),boundary(:,1),'k')
plot(cengrav(1,2),cengrav(1,1),'r*-')
hold off
```

en bereken dan de beschrijving in functie van de afstand tot het middelpunt (= rho) en de hoek t.o.v. de horizontale (= theta).

```
rhotheta = zeros(size(boundary,1),2);
boundarypol = boundary;
boundarypol(:,1) = boundarypol(:,1) - cengrav(1,1);
boundarypol(:,2) = boundarypol(:,2) - cengrav(1,2);
[boundrho,boundtheta] = cart2pol(boundarypol(:,1),boundarypol(:,2));
figure, plot(boundrho,boundtheta,'k.')
```

3 Kettingcodes

Kettingcodes worden gebruikt om objectranden voor te stellen door een geconnecteerde sequentie van segmenten bestaande uit rechte lijnen met een bepaalde richting en lengte. Bereken een kettingcode voor het beeld **leaf.bmp**: stel een algoritme op om een kettingcode voor deze rand te creëren. Zorg ervoor dat dit algoritme zowel 4- en 8-connectiviteit aankan.

Door de originele rand te coderen krijg je een kettingcode die even lang is als de rand. Wat zijn nog andere nadelen van deze aanpak?

Implementeer een veel kortere kettingcoderepresentatie door een soort resampling van de rand te bepalen. Leg een resample rooster over het beeld met een vooraf bepaalde afstand (bv. een tiental pixels) tussen de opeenvolgende samplepunten. Gebruik de minimale Manhattan- of de cityblock-distance als maat om originele randpunten toe te wijzen aan een samplepunt op het rooster. Let erop dat je de volgorde van doorlopen van de rand niet verbreekt. Door de gesamplede versie van de rand te normaliseren met de roosterafstand, kan de ontwikkelde code voor het kettingalgoritme opnieuw gebruikt worden.

4 Polygonale benaderingen

Een polygonale benadering voor een objectrand is gemakkelijk te berekenen met het volgende algoritme:

1. Zoek eerst de 2 punten a en b op de objectrand die het verst van elkaar verwijderd zijn. Creëer een lijst polygon_app met deze 2 punten: $\{a, b\}$.
2. herhaal de volgende stappen totdat polygon_app niet meer aangroeit.
3. Zoek in alle randpunten tussen 2 opeenvolgende punten p_i en p_j uit polygon_app het punt p_k dat het verste van de lijn tussen p_i en p_j afligt.
4. Als de afstand tussen p_k en de lijn $p_i - p_j$ kleiner is dan een vooropgelegde drempelwaarde T_d (bv. 25 % van de lengte tussen a en b), voeg dan het punt p_k toe aan polygon_app tussen p_i en p_j.

Bekijk het resultaat van dit algoritme voor het beeld **leaf.bmp** voor verschillende waardes van de drempelwaarde T_d.

5 Bespreking van de representatie

Wat is belangrijk voor de representatie van een object? Stel dat je een gelijkaardig, maar niet hetzelfde, object wil gaan herkennen, waarmee moet je dan rekening houden? Hou in de bespreking rekening met translatie, rotatie, schaling en kleine vormafwijkingen van het te herkennen object.

6 Randbeschrijving

Bereken enkele eenvoudige descriptoren voor de randen, zoals de lengte of de diameter (= de afstand tussen de 2 verst gelegen punten op een rand) wat de grote as van het object beschrijft. Daaruit kan de kleine as van het object berekend worden als de loodrechte op de grote as op de plaats waar het object het breedste is. Op basis van deze inforamtie kan dan ook de minimaal omsluitende rechthoek en de eccentriciteit (komt ongeveer overeen met de verhouding tussen lengte en breedte) berekend worden.

Een andere manier om de rand te beschrijven is de kromming of de snelheid waarmee dat de helling veranderd. Dit kan je gemakkelijk toepassen op de de polygonale representatie van de objectrand: bereken voor de lijst polygon_app (gegeven in de file **polygon_app.mat** of als resultaat van je eigen implementatie) telkens het verschil in helling tussen 2 opeenvolgende lijnsegmenten. Daaruit kan je conclusies trekken over het recht zijn van opeenvolgende segmenten (de hoekverandering ertussen is kleiner dan 10^o) of dat 2 segmenten een duidelijk hoek vormen (hoekverandering $> 80^o$). Pas dit toe en duidt de hoekpunten aan op de rand van het blad.

7 Beschrijving van de regio's

Voor de beschrijving van regio's kunnen gelijkaardig kenmerken bepaald worden. Gebruik de functie **regionprops** om de gesegmenteerde regio's in het gelabelde beeld van de verschillende munten te beschrijven. Denk je dat je op deze manier de verschillende munten kan herkennen?

7.1 Textuur

Een andere manier om gebieden van objecten te beschrijven is aan de hand van textuureigenschappen. Gebruik de code gegeven in de file **textureanalysis.m** om regio's met verschillende textuur in het beeld **campus.jpg** te analyseren. De input van deze file is gewoon de naam van het beeld. De uitvoer geeft een vector met volgende waardes terug: [gemiddelde grijswaarde, gemiddeld contrast, relatieve smoothness, genormaliseerd derde moment, uniformiteit, entropie]. Hoe kan je de verschillende regio's onderscheiden a.d.h.v. deze parameters? Kijk vooral naar de regio's met kleine en grote contrastverschillen, regio's met een bepaald herhaling in de intensiteitsovergangen, ...

Figuur 2: campus.jpg

8 Principal Components Analyse (PCA)

Door een PCA kan je de belangrijkste elementen van je dataset onderscheiden. Pas dit eens toe op het kleurenbeeld **lena_color.jpg** en bekijke de resultaten van de onderstaande code. Geef een bespreking van deze resultaten in functie van de berekende eigenwaarden en -vectoren.

```
% PCA op een kleurenbeeld
im = im2double(imread('lena\_color.jpg'));
imresh = reshape(im,size(im,1)*size(im,2),3);
featurevecs = imresh;
covfeat = cov(featurevecs);
% Compute eigenvalues en eigenvectors
[V,D] = eig(covfeat); % V = vectors, D = values
d = diag(D); % rearrange
[d,idx] = sort(d);
d = flipud(d);
idx = flipud(idx);
D = diag(d);
V = V(:,idx);
% Compute the Hotelling transform
A = V';
mx = sum(featurevecs,1)/size(featurevecs,1);
Mx = repmat(mx,size(featurevecs,1),1);
Y = A*((featurevecs - Mx)');
imres = zeros(size(im));
for i = 1:3,
    imres(:,:,i) = reshape(Y(i,:),size(im,1),size(im,2));
end
figure,
subplot(221), imshow(im),
subplot(222), imshow(imres(:,:,1))
subplot(223), imshow(imres(:,:,2))
subplot(224), imshow(imres(:,:,3)) }
```

Figuur 3: lena_color.jpg en de beelden voor de respectievelijke eigenwaarden, gerangschikt van groot naar klein.

Pas nu zelf PCA toe op de randpixels van de het blad uit het beeld **leaf.bmp**. Gebruik de coördinaten van de randpixels als featurevectoren waarvoor je dan de eigenwaarden- en vectoren berekend. Wat is dan het effect van Hotelling transformatie en wat is het voordeel daarvan?