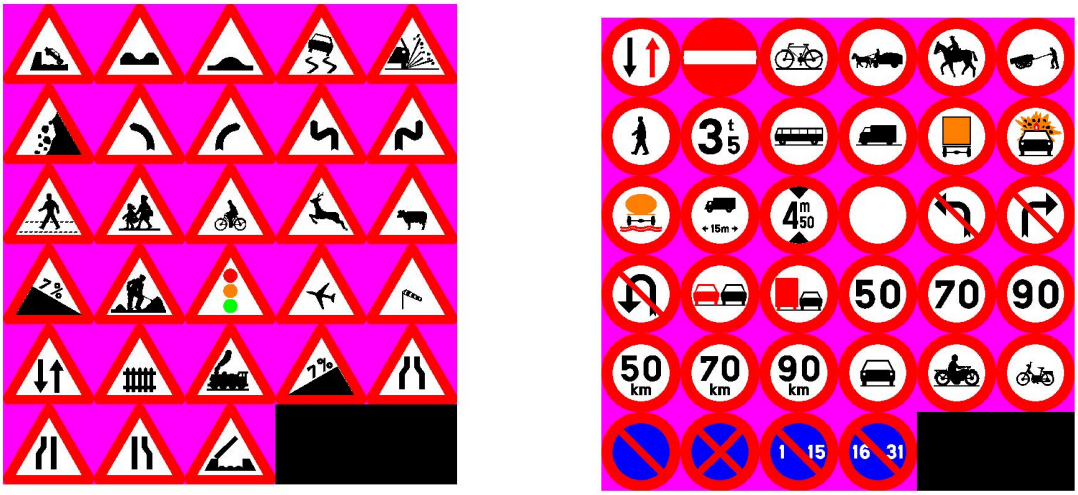
Objectherkenning

Kristof Teelen

Labo beeldverwerking / 2007-08

1 Minimum Distance Classifier

Herkenning gebeurt vaak op basis van enkele specifieke kenmerken van de te herkennen objecten. Een verzameling of vector van desciptoren of features vormt dan het te herkennen patroon. Automatische machineherkenning moet dan de patronen toewijzen aan de correcte klasse, liefst zonder inmening van een menselijke operator.



Figuur 1: De beide klassen van verkeersborden gebruikt in het eerste voorbeeld.

Patronen worden dikwijls samengevat in een featurevector, een lijst van eigenschappen, specifiek voor dat patroon. De verschillende features moeten natuurlijk zo gekozen worden dat ze een goed onderscheid vormen tussen de verschillende objectklassen. In het eerste voorbeeld van dit labo moeten ronde rode van driehoekige rode verkeersborden onderscheiden worden. Dus moeten we features kiezen die een duidelijk verschil tussen beide klassen geven. Alle borden zijn hoofdzakelijk rood, dus kleur op zich is geen optie, een andere mogelijkheid is de verhouding van het aantal rode pixels ten opzichte van het totaal aantal pixels in ee nverkeersbord. Dat wodt het eerste feature. Voor het tweede feature kijken we naar de vorm: een rond verkeersbord is het breedst in het midden, terwijl dat voor een driehoek boven of onderaan is. Als tweede feature kiezen we dan de verhouding tussen de verticale positie van het breedst stuk tot de totale hoogte van het bord. Deze 2 features vormen dan een vector voor elk bord. De verzameling van vectoren, i.e. de feature vector, is gegeven in de file **data_trafficsigns.mat**.

Herkenning op basis van matching vereist een prototype patroonvector voor elke klasse. Voor de classificatiemethode op basis van minimale afstand nemen we de gemiddelde vector voor elke klasse.

```
load data_trafficsigns.mat
% compute prototypes
```

```
protot(1,:) = sum(roundred,1)/size(roundred,1);
protot(2,:) = sum(trianglered,1)/size(trianglered,1);
```

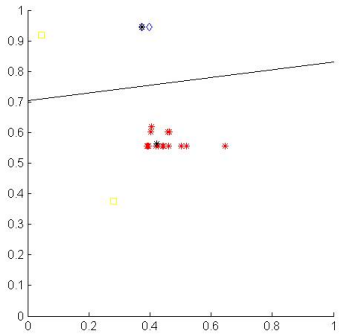
Figuur 2: Onbekende objecten die geclassificeerd moeten worden.

Om nu nieuwe, onbekende patronen toe te wijzen aan een klasse moeten we berekenen bij welke klasse dit patroon het dichtste aansluit. Voor minimale afstandsclassificatie moeten we dus een metriek definiëren waarmee we kunnen aanduiden hoe dicht dit patroon bij elke prototype vector ligt. We kiezen in dit voorbeeld voor de Euclidische of 2-norm afstand, die we berekenen in een 2D-ruimte, want we hebben natuurlijk 2 features. Er kan dan een decisiegrens opgesteld worden tussen verschillende klasses voor de minimale afstandsclassificatie:

```
d12 = dataval*(protot(1,:)'-protot(2,:)')
- ((protot(1,:)'-protot(2,:)')'*(protot(1,:)'+protot(2,:)'))/2
```

Het teken van $d12$ duidt aan tot welke klasse een onbekend patroon behoord. Probeer dit uit voor de featurevectoren gegeven in de variabele dataval in **data_trafficsigns.mat**, die verkregen werden voor de beelden in figuur 2. Bereken de ligging van de nieuwe featurevectoren ten opzichte van de decisiegrens.

```
colorplot = {'r*','bd'}
figure, hold on,axis equal, axis([0 1 0 1])
for i = 1:size(datavalues,1),
    plot(datavalues(i,1),datavalues(i,2),colorplot{datavalues(i,3)})
end
plot(protot(:,1),protot(:,2),'k*')
plot(dataval(:,1),dataval(:,2),'ys')
hold off
```



Figuur 3: Minimum distance classifier.

In de resulterende figuur 3 zien we dat de twee reële borden juist geclassificeerd worden door de classifier die we opgesteld hebben aan de hand van de databank van de verkeersborden. Welk van de features is het best gekozen en waarom?

2 K-means clustering

Clustering is een methode om data in verschillende disjuncte subverzamelingen van objecten te verdelen of te 'clusteren'. K-means clustering behandelt ieder object door enkele parameters (van een featurevector) uit te zetten als een locatie in het spatiale domein. Dan zoekt het algoritme een verdeling van die ruimte zodat objecten in een cluster zo dicht mogelijk bij elkaar liggen, en zo ver mogelijk van objecten in andere clusters terecht komen. K-means clustering vereist 2 inputs: het aantal clusters (dit moet dus op voorhand gekend zijn) en een afstandsmetriek om de afstand tussen 2 objecten in de ruimte te beschrijven.

Ondanks enkele beperkingen wordt dit algoritme vrij veel gebruikt omdat het snel en eenvoudig te implementeren is. Maak zelf een implementatie op basis van de volgende beschrijving. Het algoritme doorloopt iteratief een procedure om de verschillende clusters te schatten:

1. Initieel worden alle datapunten random verdeeld over de k clusters.
2. Stap 1: bereken de centroïd of het centrum voor elke cluster als $\frac{\sum_{i=1}^{n}}{n}$ met n het aantal objecten in die cluster, i.e. het 'gemiddelde' object.
3. Stap 2: wijs alle objecten toe aan die cluster waarvoor de afstand van het object tot de cluster centroïd het kleinste is.
4. Herhaal stap 1 en 2 totdat voldaan is aan een vooraf opgegeven stopcriterium. Kies bijvoorbeeld om te herhalen totdat er geen verschil meer is in de clusterverdeling.

Implementeer zelf het K-means clustering algoritme voor de dataset van featurevectoren gegeven in **datavalues.mat**. Hierin worden 2 features beschreven voor 3 verschillende sets van bladeren: de eerste 15 elementen komen overene met waardes voor *garryana*, de volgende 15 met *circinatum*, en de laatste 15 met *negundo*. De eerste kolom geeft de verhouding tussen de oppervlakte van het gebinariseerde blad en de oppervlakte van zijn bounding box, terwijl de tweede kolom de verhouding tussen de lengte van de grote en de kleine as van het gebinariseerde blad voorstelt. Deze parameters zijn berekend met de functie **regionprops**, die we in het labo over representatie ook al gebruikt hebben. Waarom gebruiken we hier verhoudingen tussen kenmerken als parameters in de featurevector voor elk blad?

Figuur 4: De 3 klasses van bladeren: *garryana*, *circinatum* en *negundo*. Het vierde blad is een nieuw blad, onbekend voor ons systeem en dus nog te classificeren blad.

De invoer voor je algoritme is duidelijk: er zijn $k = 3$ klassen van bladeren, dus ook drie clusters. Gebruik als afstandsmaat de 2-norm of de euclidische afstand voor de waarden van de parameters, die je kan uitzetten in een 2D vlak.

Bekijk de resultaten van de clustering? Werk het algoritme optimaal? Indien niet, waarom niet en wat zouden we kunnen aanpassen om de werking te verbeteren?

Indien we een nieuw beeld van het gebinariseerde blad krijgen, hoe kunnen we dat dan toewijzen aan één van de drie clusters? Probeer dit voor het blad gegeven in figuur 4 en de gebinariseerde versie in **unknownleaf.bmp**. Bereken de features voor dit blad en wijs het toe aan een van de drie klasses.

Wat is de relatie van K-means clustering met segmentatietechnieken? En met PCA?

3 Correlatie

Template matching gebeurt in beeldverwerking dikwijls door correlatietechnieken toe te passen. Kies een template of sub-beeld van het object dat je wil terug vinden in een gegeven beeld en vergelijk de template het gegeven beeld elkaar door de correlatie met elk deel van het beeld te berekenen.

Eén toepassing waarin deze techniek dikwijls toegepast wordt is stereovisie. Bij stereovisie wordt een scène opgenomen door 2 verschillende camera's C en C', op korte afstand van elkaar, zoals getoond in figuur 5[1]. Een punt x in het ene beeld kan je dan enkel relateren aan een lijn in het andere beeld omdat je de diepte van het 3D-punt X in de scène/wereld niet kent. Het punt ligt op 1 welbepaalde straal tussen het 3D wereldpunt X en het cameracentrum C van de eerste camera. Die straal komt overeen met een lijn l' in het tweede beeld: de epipolaire lijn voor het punt x in het eerste beeld. De relatie tussen homogene positieparameters voor x en de lijnparameters l' wordt gegeven door de fundamentele matrix F als $l' = Fx$. Deze relatie wordt geïllustreerd door de m-file **fundamental.m** (die de fundamentele matrix voor het stereobeeldenpaar uit figuur 6 inlaadt uit de mat-file **fundmat.mat**).

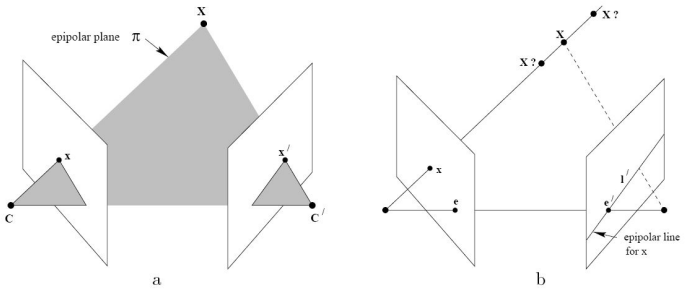


Fig. 8.1. **Point correspondence geometry.** *(a) The two cameras are indicated by their centres* **C** *and* **C**′ *and image planes. The camera centres, 3-space point* **X**, *and its images* **x** *and* **x**′ *lie in a common plane* π. *(b) An image point* **x** *back-projects to a ray in 3-space defined by the first camera centre,* **C**, *and* **x**. *This ray is imaged as a line* **l**′ *in the second view. The 3-space point* **X** *which projects to* **x** *must lie on this ray, so the image of* **X** *in the second view must lie on* **l**′.

Figuur 5: Stereovisie: epipolaire lijnen en de fundamentele matrix.

De correlatie wordt gebruikt om het gekozen punt uit het linkerbeeld toch te kunnen relateren aan een pixel uit het rechterbeeld. Vul de code gegeven in **fundamental.m** aan met de implementatie van de berekening van de correlatie. Selecteer eerst een regio rond het gekozen punt in het linkerbeeld en vergelijk die met alle subbeelden van dezelfde grootte uit het rechterbeeld die met hun centrum op de epipolaire lijn l' liggen. Bereken de correlatiecoëfficiënt met de matlabfunctie **corr2** en onthoud waar op l' de maximale waarde voorkomt. Duidt deze lokatie aan op een resultaatbeeld. Probeer dit voor verschillende punten uit beeldregio's met verschillende karakteristieken en vergelijk de resultaten.

Figuur 6: Stereovisiebeeldenpaar: **left1.jpg** en **right1.jpg**.

[1] Figuur uit *Hartley and Zisserman - Multiple View Geometry in Computer Vision, 2nd Ed., Cambridge University Press, 2003.*