# Clustering in doelvolgalgoritmen van radarsystemen

Drs. B. Tesselaar

Dr. A. Volgenant
Universiteit van Amsterdam
*Faculteit der Economische Wetenschappen en Econometrie*
*Operations Research groep*

Prof. Dr. Ir. F.G.J. Absil
Koninklijk Instituut voor de Marine
*Vakgroep Informatie- en Wapentechnische Wetenschappen*

## Samenvatting

Clustering van doelen en metingen is een belangrijk onderdeel in doelvolgalgoritmen van radarsystemen, zoals het Multiple Hypotheses algoritme. Clustering is de bottleneck binnen een Multiple Target Tracking systeem als veel doelen real-time gevolgd moeten worden. Het is daarom van belang dat een efficiënt clusteringalgoritme in het systeem is geïmplementeerd. Verschillende clusteringalgoritmen worden beschreven evenals hoe deze zijn aangepast, geïmplementeerd en getest voor toepassing binnen een Multiple Target Tracking systeem.

---

A.Volgenant, Operations Research groep
Faculteit der Economische Wetenschappen en Econometrie
Roetersstraat 11, 1018 WB Amsterdam,
tel. 020 525 4219 (4217), Email: *tonv@fee.uva.nl*

# 1 Inleiding

Met Multiple Target Tracking (afgekort MTT) wordt hier bedoeld het volgen van meer luchtdoelen gelijktijdig met behulp van meetgegevens van een radarsysteem. De civiele toepassing concentreert zich op de luchtverkeersbegeleiding; in een militaire toepassing draagt dit bij aan de opbouw van het luchtbeeld in een luchtverdedigingsscenario.

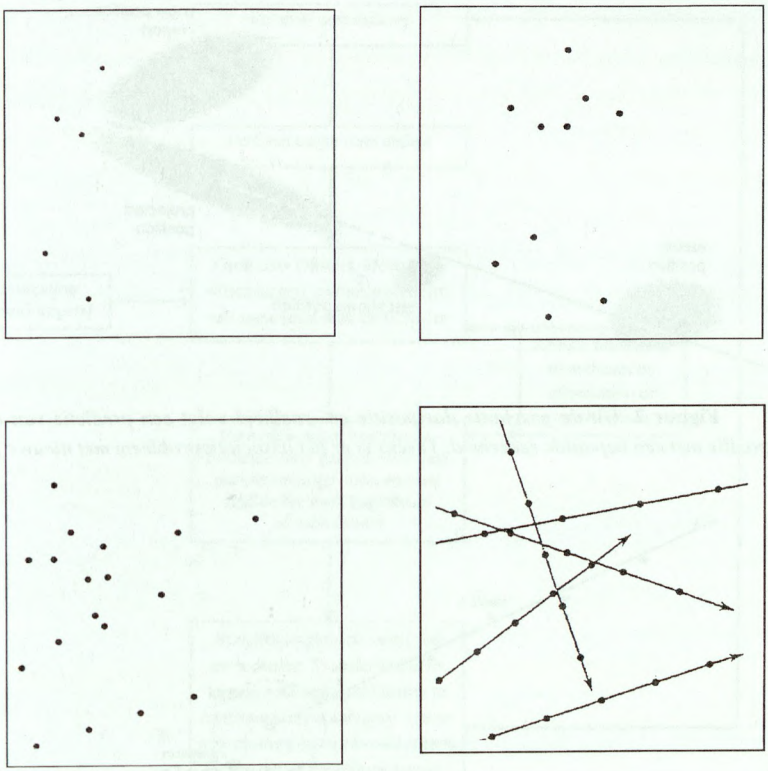
Een MTT-systeem genereert doelbanen (*tracks*), gebruikmakend van de *plots* van een rondzoekradar. Dit zijn de *metingen* (*measurements*) die na voorbewerking van het signaal als een geldige detectie overblijven. Bij een lange-afstandsradar (bereik > 100 km) vinden de metingen plaats in een 2D-ruimte in poolcoördinaten $r$ (de afstand tot de radar) en $\varphi$ (de azimut-hoek). De duur van een rondzoekslag, een *scan*, bedraagt 5 - 8 seconden.

Radarplots kunnen ook afkomstig zijn van niet-relevante doelen, zoals vogels, neerslag, bebouwing en zeegolven. Dit leidt tot *false alarms*. Tevens kunnen doelen tijdens een scan gemist worden, de zogenaamde *missed reports*. Het doelvolgalgoritme in de radar-processor moet dit type fouten verwerken.

Het schatten van de doelsparameters is veelal gebaseerd op de Kalman filteringstechniek. Voor een gedetailleerde beschrijving zij verwezen naar Bar-Shalom et al. (1995). Deze techniek bestaat uit een predictor-stap, gebruikmakend van een model van de doelsbeweging, en een corrector-stap, waarbij de meting op een zodanige wijze verwerkt wordt, dat de beste schatting (in kleinste kwadraten-zin) plaatsvindt. Fouten in de doelsparameters zijn het gevolg van proces- en meetruis en van onzekerheid omtrent de doelsbeweging.

Bij meervoudige doelen en metingen is het schattingsprobleem ingewikkelder. Alvorens de corrector-stap kan plaatsvinden moeten tracks en plots met elkaar geassocieerd worden; denk hierbij aan kruisende doelsbanen, formaties van doelen en manoeuvres.

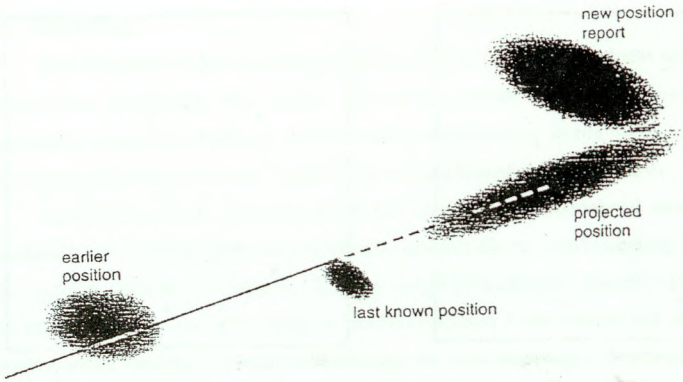
Een MTT-systeem dient een algoritme te bevatten dat zowel het associatie- als het schattingsprobleem oplost; dit is een actueel onderwerp in het vakgebied van de radar data-processing. De figuren 1, 2 en 3 - bron: Uhlmann (1992) - illustreren het probleem. In principe kunnen tijdens een scan alle plots met alle tracks geassocieerd worden. MTT-algoritmen leveren vaak pas na enige scans een oplossing voor de (meest waarschijnlijke) tracks. Elke associatie heeft zijn eigen waarschijnlijkheid die gebaseerd is op de predictie van de doelsparameters en hun onzekerheid, volgend uit een eerdere schatting. Figuren 2 en 3 illustreren dit: de onzekerheid wordt uitgedrukt in een foutenellips of validatie-regio. Figuur 3 toont dat de nearest-neighbor associatie niet noodzakelijkerwijs correct is.



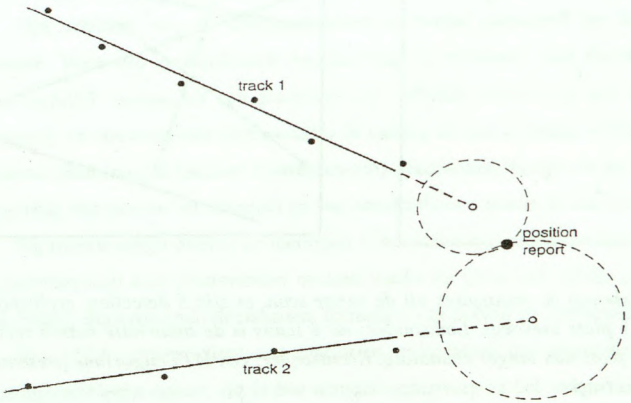
**Figuur 1.** *Linksboven de radarplots uit de eerste scan, er zijn 5 detecties; rechtsboven de tweede scan die 5 plots toevoegt. Linksonder: na 4 scans is de associatie tussen recente metingen en eerdere plots niet langer eenduidig. Rechtsonder: een MTT-algoritme presenteert na 5 scans een aantal tracks.*

## 2 Multiple Hypotheses Tracking

Om de metingen aan de doelen toe te wijzen, moeten de waarschijnlijkheden van mogelijke associaties tussen doelen en metingen worden berekend. Een combinatie van verschillende associaties leidt tot een hypothese met een bepaalde waarschijnlijkheid. Het aantal hypothesen stijgt exponentieel met het aantal doelen en / of metingen en dus ook de tijd om alle hypothesekansen te bepalen. Veel hypothesen zijn overbodig, omdat daarbinnen bijvoorbeeld een doel wordt geassocieerd met een verafgelegen meting.

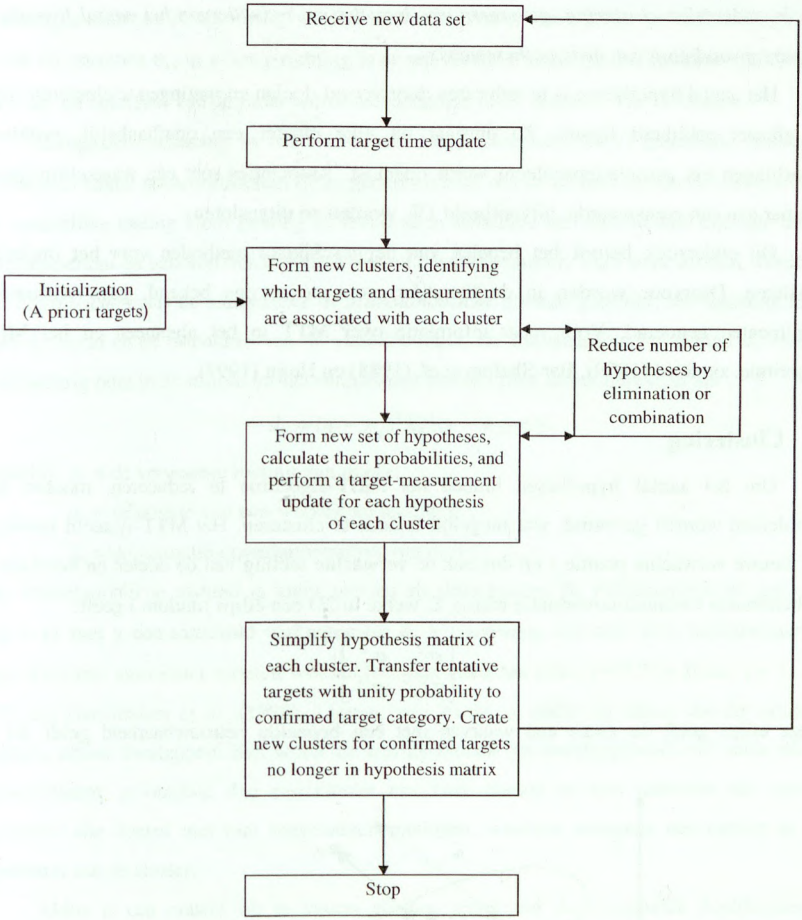



**Figuur 2.** *Uit de geschatte doelpositie en -snelheid volgt een predictie van de nieuwe positie met een bepaalde zekerheid. Tevens is er het associatieprobleem met nieuwe plots.*




**Figuur 3.** *De 2 tracks leveren elk een predictie. De foutenellipsen (validatie-regio's) representeren de onzekerheid van de predictie. De nieuwe plot valt binnen beide ellipsen en is dus hetzij afkomstig van een der doelen hetzij een false alarm.*

Het Multiple Hypotheses Tracking-algoritme (afgekort MHT), één van de ontwikkelde algoritmen voor MTT, werkt met een hypothese-boom, waarbij na iedere scan takken ontstaan en verdwijnen, zo dat bijvoorbeeld de 100 meest waarschijnlijke hypotheses bewaard blijven. Deze vormen een beeld van hoe doelen zich in de tijd waarschijnlijk verplaatst hebben. Heijn (1997) en Bar-Shalom *et al.* (1995) behandelen het toewijzingsprobleem.




**Figuur 4.** *Een stroomdiagram voor het MHT-algoritme — bron: Reid (1979).*

In het geval van grote aantallen doelen en metingen hebben veel doel-meting associaties een zeer kleine kans; ook is het vrijwel onmogelijk om real-time de hypothesekansen te bepalen. *Real-time* houdt in dat het MHT-algoritme doorlopen is voor een gegeven set geïnitialiseerde doelen en metingen alvorens de nieuwe set metingen beschikbaar komt. De tijd tussen twee opeenvolgende radarscans is uiteraard afhankelijk van het type radar dat gebruikt wordt.

Figuur 4 geeft het stroomdiagram van het MHT-algoritme, zie Reid (1979); het bestaat

uit de onderdelen *clustering, genereren van hypothesen, reductie van het aantal hypothesen* en *vereenvoudiging van de hypothesematrix.*

Het aantal hypothesen is te reduceren door vooraf doelen en metingen te clusteren welke in elkaars nabijheid liggen. Zo ontstaat uit elke cluster een onafhankelijk probleem, waarbinnen het associatieprobleem wordt opgelost. Associaties met een waarschijnlijkheid kleiner dan een grenswaarde, bijvoorbeeld 1%, worden zo uitgesloten.

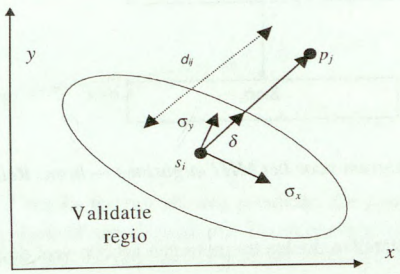
Dit onderzoek betreft het bepalen van de geschiktste methoden voor het onderdeel clusteren. Daarvoor worden in de literatuur, voor zover ons bekend, geen wiskundige argumenten genoemd. Voor meer informatie over MTT in het algemeen en het MHT-algoritme, zie Reid (1979), Bar-Shalom *et al.* (1988) en Heijn (1997).

## 3 Clustering

Om het aantal hypothesen binnen het MHT-algoritme te reduceren, moeten sub-problemen worden gevormd, wat mogelijk is door te clusteren. Het MTT-systeem voorspelt de nieuwe verwachte positie $s$ en dus ook de verwachte meting van de doelen en berekent de bijbehorende variantie-covariantie matrix $\Sigma$, welke in 2D een ellips rondom $s$ geeft:

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_y^2 \end{bmatrix}. \qquad (1)$$

Deze ellips geeft de grens aan waarvan met een bepaalde betrouwbaarheid geldt dat de




**Figuur 5.** *Ellips (validatie-regio of -gate) rond een verwachte doelpositie. Uit het Kalmanfilter ontstaat een voorspelling voor doelpositie $s_i$ en $\Sigma_i$. De covariantie-elementen uit $\Sigma$ bepalen de draaiing van de ellips, de threshold $\gamma$ bepaalt de grootte van de validatie-regio. De statistische afstand $d_{ij}$ wordt vergeleken met de threshold $\delta$ om meting $p_j$ aan doel $i$ toe te kunnen wijzen. Daar $d_{ij} > \delta$ vindt er geen toewijzing plaats.*

werkelijke radarplot $p$, afkomstig van het doel, zich binnen deze ellips bevindt (zie figuur 5). Door covarianties $\sigma_{xy}$ in $x$- en $y$-richting is er een draaiing mogelijk ten opzichte van de $x$- en de $y$-as. De meetfout van de radar wordt meegenomen in de matrix $\Sigma$ van de doelen.

Aangezien clustering in het Cartesisch coördinatenstelsel $(x, y)$ geschiedt, worden de radarplots vanuit poolcoördinaten (r, φ) geconverteerd. Als de afstand tussen de voorspelde en de werkelijke meting klein genoeg is, levert deze associatie een significante bijdrage aan de hypothesekans en behoren beide punten dus tot hetzelfde cluster. Voor deze afstand wordt niet de *Euclidische* maar de *statistische* of *Mahalanobische* afstand gebruikt, die rekening houdt met de vorm en de oriëntatie van de validatie-regio. De Mahalanobische afstand $d_{ij}$ van een willekeurig punt in de ruimte tot het middelpunt van de ellips wordt berekend als:

$$d_{ij} = (s_i - p_j)^T \Sigma_i^{-1} (s_i - p_j) < \gamma, \tag{2}$$

waarbij: $s_i$ = de verwachte meting van doel $i$;

$p_j$ = de positie van een werkelijke meting $j$;

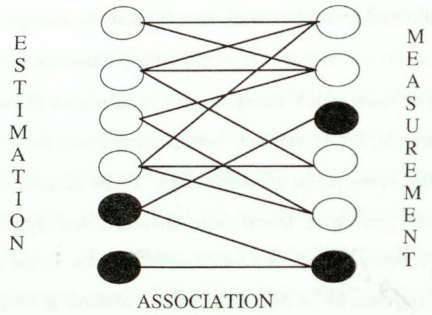
$\Sigma_i$ = de variantie-covariantiematrix van doel $i$.

De Mahalanobische afstand is klein genoeg als deze binnen de validatieregio of -gate valt ($d_{ij} < \gamma$) met $\gamma$ een threshold verkregen uit de $\chi^2$-verdeling. Als met 99% betrouwbaarheid alle relevante associaties moeten worden meegenomen, dan geldt $\gamma = 9.2$ in 2D en $\gamma = 11.4$ in 3D, zie Bar-Shalom *et al.* (1988). Liggen twee tracks zo dicht bij elkaar dat de validatie-regio's elkaar overlappen, dan wordt de meting binnen het overlapgebied met beide doelen geassocieerd; er ontstaat dan een cluster met twee doelen en met minstens één meting, alsmede alle doelen met hun toegewezen metingen, waarvan minstens één meting is toegewezen aan de cluster.

Aldus is een cluster op te vatten als een *graaf* met de voorspelde doelposities en verkregen metingen als de *knopen* en de toewijzingen van metingen aan doelen als de *kanten*.

Er kunnen meer clusters ontstaan. Een kenmerk van deze clusters is, dat deze (vrijwel) onafhankelijk van elkaar zijn (deze worden onafhankelijk verondersteld). Voor een meting uit de ene cluster kan met 99% betrouwbaarheid worden gezegd dat deze niet moet worden geassocieerd met een doel uit een andere cluster. Figuur 6 beeldt twee clusters (deelgrafen) af, zonder toewijzingen tussen de knopen uit een deelgraaf (een bipartite graaf).

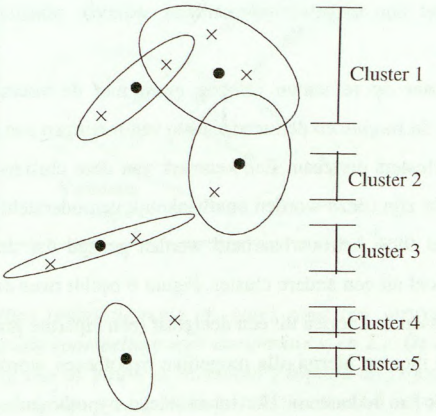
Voor elke cluster moeten hierna alle mogelijke hypothesen worden opgesteld, wat het aantal hypothesen sterk kan reduceren. Het totaal aantal hypothesen zonder clustering voor bijvoorbeeld $n$ = 10 doelen en $m$ = 10 metingen is gelijk aan het aantal verschillende




**Figuur 6.** *De toewijzingen tussen doelen (estimation) en metingen (measurements) vormen een bipartite graaf met als deelgrafen de cluster van de 'zwarte' en die van de 'open' knopen - bron: Collins et al. (1992).*

configuraties met $n \times m = 100$ associaties. Door clustering, bijvoorbeeld 3 clusters van respectievelijk 3, 4 en 3 doelen en metingen per cluster, ontstaan 34 associaties, een aanzienlijke reductie. Groeit het aantal real-time te volgen doelen, dan wordt clustering naar verwachting belangrijker.

Bij een missed report wordt er geen meting aan een doel toegewezen. Bij een false alarm valt de afstand tot een doel binnen de threshold, zodat het false alarm wordt toegewezen aan dat doel. Valt het false alarm buiten de validatie-regio van een voorspeld doel, dan wordt deze meting als een nieuw te initialiseren doel beschouwd in een eigen cluster. Figuur 5 illustreert




**Figuur 7:** *Een voorbeeld met 5 clusters (•: voorspelde doelpositie, ×: measurement).*

het bepalen van de statistische afstand doel-meting $d_{ij}$, hét criterium voor clustering in elke clusteringmethode. Figuur 7 illustreert het vinden van overlap tussen metingen en validatie-regio's en hoe clusters eruit gaan zien.

- Cluster 1 bestaat uit twee doelen die in één cluster zitten, omdat de doelen een gezamenlijke meting in hun validatie-regio bezitten; zo'n cluster wordt *supercluster* genoemd.
- Cluster 2 bestaat uit één doel en één meting; er is weliswaar overlap in de validatie-regio met een validatie-regio van een doel uit cluster 1, maar omdat deze zone geen meting bevat, wordt cluster 2 niet toegevoegd aan cluster 1.
- Cluster 3 heeft geen overlap met andere validatie-regio's en staat daarmee op zichzelf.
- Cluster 4 bestaat alleen uit een meting; deze valt buiten elke validatie-regio en wordt als een nieuw te initialiseren doel beschouwd in een eigen cluster, of ook wel '0'-cluster.
- Cluster 5 bestaat uit één doel zonder meting; het gaat hier om een missed report, of het doel was eerder geïnitialiseerd, terwijl het een false alarm was.

Het clusteren is in principe eenvoudig: alle doelen moeten met alle metingen worden vergeleken. Mahalanobische afstanden worden bepaald en vergeleken met de validatie-regio's rond de doelen. Maar deze 'brute-force'-methode (zie § 4) zal inefficiënt blijken te zijn.

In het vervolg beschrijven we methoden die Cox (1993) als efficiënter beschouwt: in § 5 de *binary tree* methode die een onderdeel is van de methode van Uhlmann (1992) (zie § 6), in § 7 de *bucketingmethode* beschreven door Zhang *et al.* (1992) en Edahiro *et al.* (1984) en in § 8 de *lower bound* methode volgens Orr *et al.* (1992).

## 4 De brute-force methode

De intuïtief eenvoudig te begrijpen brute-force methode leidt tot de juiste toewijzingen van metingen aan doelen, maar zal blijken niet efficiënt te zijn. Als met deze methode veel doelen gevolgd moeten worden, vormt volgens Cox (1993) het clusteren binnen MTT de bottleneck. Dit probleem was zeer relevant in een koude-oorlogsscenario waarbij massa-aanvallen met *intercontinental ballistic missiles* beschouwd werden en de doelen met infra-roodsensoren vanuit satellieten gevolgd werden.

Stel we hebben $m$ metingen en $n$ doelen. De brute force methode vergelijkt elke meting $p_j$ ($j = 1, ..., m$) met alle validiatie-regio's, die gebaseerd zijn op de berekende doelspositie $s_j$ en de (co)variantiematrix $\Sigma_i$ ($i = 1, ..., n$). Per doel $i$ wordt de Mahalanobische afstand $d_{ij}$ tot alle metingen berekend en vergeleken met de threshold $\gamma$ (zie § 3). Dit levert een aantal toe-wijzingen op; de meting is mogelijk afkomstig van dit doel.

De brute-force methode is niet efficiënt, omdat deze alle mogelijke doel-meting combinaties beschouwt. De rekentijd is O(*mn*) voor het bepalen van de overlap. Er vinden echter vele berekeningen plaats, die geen toewijzing zullen opleveren. Vanwege het real time aspect is het zinvol om a priori associatiemogelijkheden met een lage waarschijnlijkheid te elimineren. Dit zal tot een snellere methode leiden.

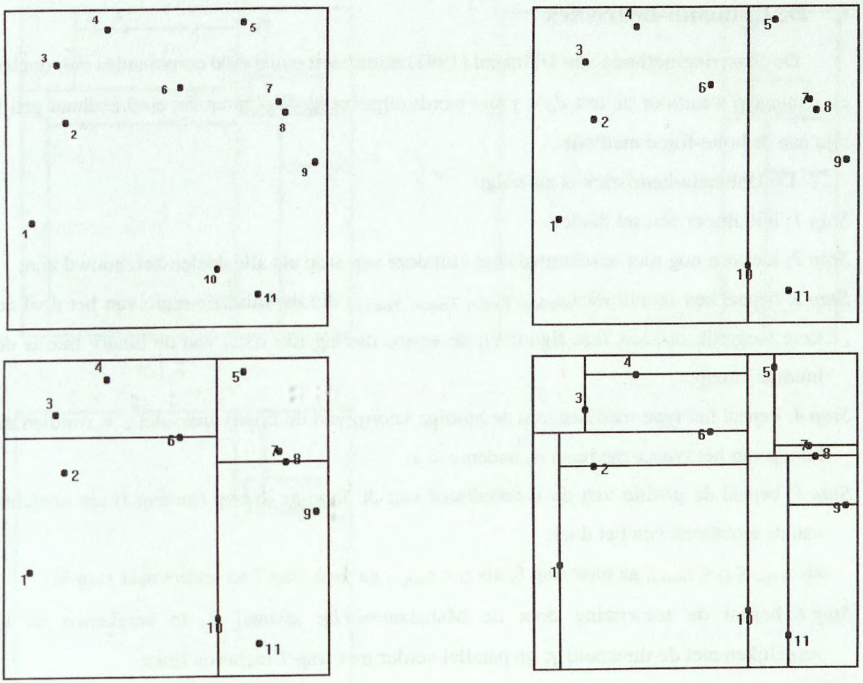
## 5 De binary tree methode

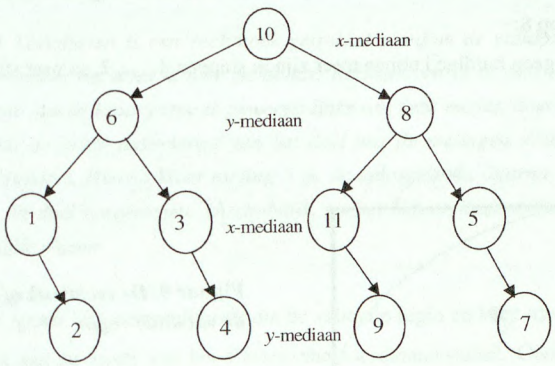
Uhlmann (1992) beschrijft voor het toewijzingsprobleem een heuristiek die gebaseerd is op een *binary tree* van metingen. We verklaren dit begrip alvorens op deze methode in te gaan. In het geval van 'binary' clustering van metingen en doelen in 2 dimensies spreekt men ook wel over een 2D-binary tree; meer over kD-binary trees onder andere in Aho *et al.* (1983) en Preparata *et al.* (1985).

Een *binary tree* van metingen bij clustering houdt in dat een boom wordt geconstrueerd, waarvan de takken verwijzingen naar metingen zijn. Om de tree te construeren moeten de metingen bij herhaling worden gesorteerd op achtereenvolgens $x$- en $y$-coördinaat, bijvoorbeeld met *quicksort*, een standaardmethode voor het sorteren van $n$ objecten in O($n$ log $n$) tijd.

De binary tree begint met de meting behorende bij de $x$-mediaan; dit is de meting, welke in $x$-richting gezien de mediaan is, dus de mediaan van de op de $x$-coördinaten gesorteerde metingen. Door die $x$-mediaan wordt dan in het $xy$-vlak een denkbeeldige lijn getrokken, loodrecht op de $x$-as; de ruimte wordt zo verdeeld in twee deelruimten. Van beide wordt afzonderlijk de $y$-mediaan bepaald; deze twee metingen worden toegewezen aan de binary tree en verbonden aan de bovenliggende $x$-mediaan in de tree. Tevens verdelen deze $y$-medianen de deelruimten zelf weer in deelruimten door lijnen parallel aan de $x$-as te trekken.

Voor elke onderliggende deelruimte wordt deze procedure herhaald tot alle metingen in de binary tree zijn opgenomen. Het construeren van de binary tree is een recursieve procedure; figuur 8 geeft daarvan een voorbeeld, waarbij de genummerde punten, in een $xy$-vlak radarplots voorstellen. De figuur toont verder hoe de mediaansplitsing verloopt; de lagen in de binary tree laten zien door welke mediaansplitsing ($x$- of $y$-richting) de meting in de tree is opgenomen, wat van belang is voor de heuristiek van Uhlmann. De eerste laag, de *root* van de tree genoemd, bevat altijd de meting afkomstig van de eerste $x$-mediaan, de lagen eronder alternerend $y$- en $x$-medianen.






**Figuur 8.** *De constructie van een (2D)-binary tree voor 11 metingen (linksboven). De eerste x-mediaan is meting 10 en verdeelt het xy-vlak in 2 deelruimten (rechtsboven). Middenlinks worden de twee deelruimten gesplitst door de y-medianen door metingen 6 en 8; middenrechts het eindresultaat, als alternerend x- en y-medianen zijn gezocht. Onder de bijbehorende binary tree (nummers corresponderen met die van de metingen uit het voorbeeld).*

## 6 De Uhlmann-heuristiek

De clusteringmethode van Uhlmann (1992) elimineert een aantal combinaties van doelen en metingen, waarvoor de test $d_{ij} < \gamma$ niet wordt uitgevoerd. Wel moet het eindresultaat gelijk zijn aan de brute-force methode.

De Uhlmann-heuristiek is als volgt:

*Stap 1:* initialiseer een set doelen;

*Stap 2:* kies een nog niet beschouwd doel $i$ uit deze set, stop als alle doelen beschouwd zijn;

*Stap 3:* bepaal een rechthoek ($x_{min,i}$, $y_{min,i}$, $x_{max,i}$, $y_{max,i}$) die de validatie-regio van het doel zo krap mogelijk omsluit (zie figuur 9); de eerste meting (de root) van de binary tree is de huidige knoop;

*Stap 4:* bepaal het type mediaan van de huidige knoop van de binary tree; stel $z = x$ indien de knoop van het type $x$-mediaan is, anders $z = y$;

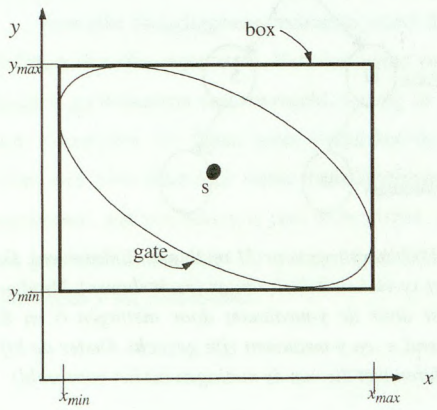
*Stap 5:* bepaal de positie van de $z_j$-coördinaat van de huidige knoop (meting $j$) ten opzichte van de rechthoek van het doel:

als $z_{min,i} \leq z_j \leq z_{max,i}$, ga naar stap 6; als $z_j < z_{min,i}$, ga naar stap 7 en anders naar stap 8;

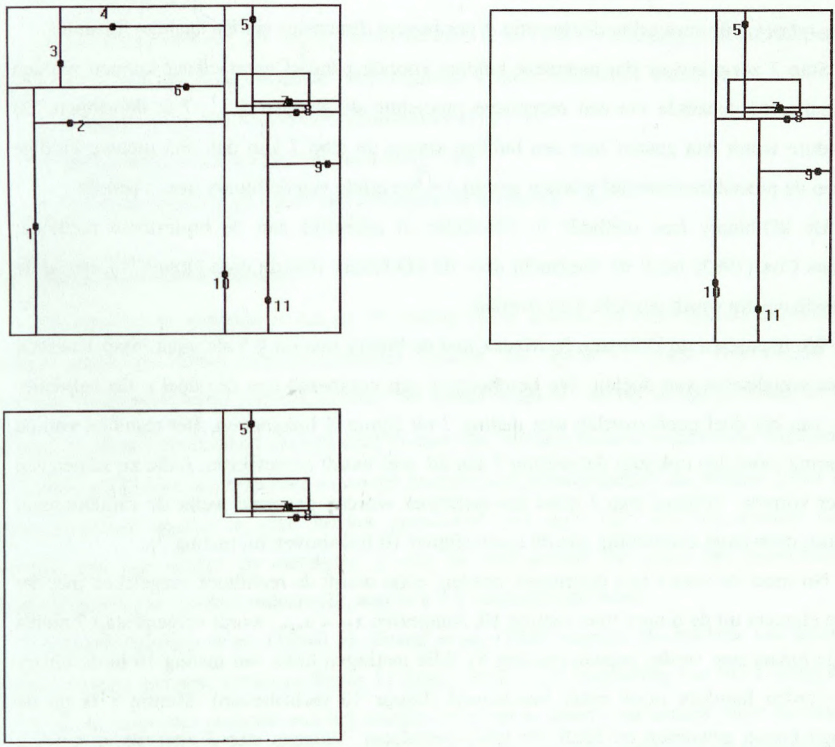
*Stap 6:* bepaal de toewijzing door de Mahalanobische afstand $d_{ij}$ te berekenen en te vergelijken met de threshold $\gamma$; ga parallel verder met stap 7 rechts en links;

*Stap 7:* (rechts / links) beschouw de binary tree; indien rechts / links onder huidige knoop nog een andere knoop komt, merk deze knoop dan als nieuwe huidige knoop en ga naar stap 4, anders naar stap 8;

*Stap 8:* indien er geen huidige knopen meer zijn in stappen 4, ..., 7, ga naar stap 2.




**Figuur 9.** *De rechthoek of box omsluit de validatie-regio.*




**Figuur 10.** *Linksboven is een rechthoek getrokken rondom de validatie-regio van een doel; de ellips rondom het doel is niet zichtbaar. Rechtsboven is te zien dat meting 10 de eerste (x-)mediaan van de binary tree is geweest; links van deze meting doet niet meer mee in de zoektocht naar de juiste associaties van het doel met de metingen evenals de metingen onder meting 8 (onder). Hierna komt meting 5 in de oplossing en daarna meting 7; alleen deze wordt aan dit doel toegewezen. Uiteindelijk, na het 'superclusteren' zitten het doel en meting 7 in dezelfde cluster*

In stap 3 is sprake van een rechthoek die de validatie-regio zo krap mogelijk omsluit en evenwijdig loopt aan de assen van het Cartesische coördinatenstelsel. Collins *et al.* (1992) geven de formules voor de zijden van zo'n rechthoek als:

$$x_{min}, x_{max} = x_i \pm \sqrt{\gamma \Sigma_{11}} \tag{3}$$

$$y_{min}, y_{max} = y_i \pm \sqrt{\gamma \Sigma_{22}} \tag{4}$$

met $s_i = (x_i, y_i)$ de verwachte doelpositie. Voor hogere dimensies gelden analoge formules.

Stap 7 zorgt ervoor dat meerdere huidige knopen parallel naast elkaar kunnen worden geïnitialiseerd, teneinde via een recursieve procedure de stappen 4, ..., 7 te doorlopen. De procedure wordt dan gestart met een huidige knoop. In stap 7 kan dan met nieuwe huidige knopen de procedure recursief worden gestart, tot het einde van de binary tree is bereikt.

De kD-binary tree methode is efficiënter in rekentijd dan de brute-force methode; volgens Cox (1993) heeft de zoektocht door de kD-binary tree de orde $O(mn^{1-1/d})$, met $d$ de dimensie waarin wordt gezocht naar overlap.

We illustreren de Uhlmann heuristiek met de binary tree uit § 5 als input. Stap 1 bestaat uit het initialiseren van doelen. We beschouwen een voorbeeld met één doel $i$. De validatie-regio van het doel geeft overlap met meting 7 uit figuur 8 linksboven. Het resultaat van de clustering moet dan ook zijn dat meting 7 aan dit doel wordt toegewezen, zodat zij samen een cluster vormen. Volgens stap 3 moet een rechthoek worden gevormd welke de validatieregio omsluit; deze loopt evenwijdig aan de assen (figuur 10 linksboven bij meting 7).

Nu moet de binary tree doorlopen worden: eerst wordt de rechthoek vergeleken met het eerste element uit de binary tree: meting 10. Aangezien $x_{10} < x_{min,i}$ wordt volgens stap 7 rechts van de binary tree verder gegaan (meting 8). Alle metingen links van meting 10 in de binary tree worden hierdoor nooit meer beschouwd (figuur 10 rechtsboven). Meting 8 is nu de huidige knoop geworden en heeft het label $y$-mediaan. Volgens stap 5 zien we $y_8 < y_{min,i}$, zodat stap 7 wordt uitgevoerd en meting 5 de huidige knoop ($x$-mediaan) wordt.

Daar $x_{min,i} \leq x_5 \leq x_{max,i}$ wordt volgens stap 6 voor deze meting gekeken of deze binnen de validatie-regio van het doel valt. Verder moeten stappen 7 (rechts / links) uitgevoerd worden. Stap 7 links zegt dat links van meting 5 een nieuwe huidige knoop wordt. Maar omdat hier geen knoop meer aanwezig is (zie figuur 8), eindigt dit pad in stap 8. Er wordt hier gewacht, omdat stap 7 rechts nog doorlopen moet worden: meting 7 wordt de huidige knoop. Ook voor deze meting wordt de Mahalanobische afstand $d_{i7}$ bepaald.

Volgens figuur 10 linksonder zijn uiteindelijk alleen $d_{i5}$ en $d_{i7}$ berekend. Meting 7 is de enige meting binnen de validatie-regio ($d_{i7} < \gamma$) en wordt daardoor aan het doel toegewezen. Hierna moeten nog eventuele andere doelen worden vergeleken met de binary tree. Hierdoor kan een meting aan meer doelen zijn toegewezen. In zo'n geval moet een *supercluster* worden gevormd; zie § 9.

# 7 De bucketingmethode

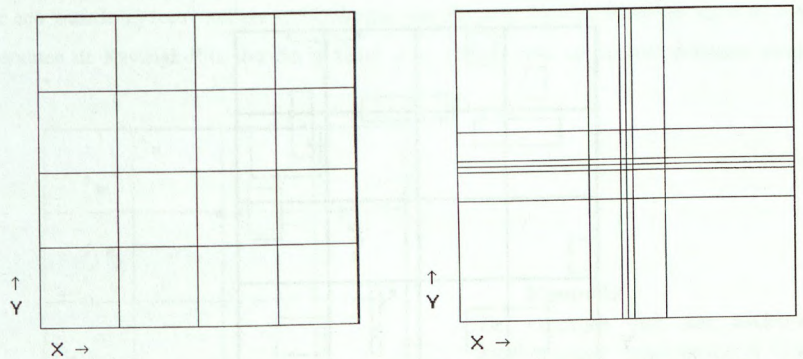
De bucketingmethode, zie bijvoorbeeld Edahiro *et al.* (1984) en Zhang *et al.* (1992), is in 3 delen te splitsen, die in de volgende subparagrafen worden behandeld, namelijk:

- het maken van een *bucket-coördinatenstelsel* (§ 7.1);
- het toewijzen van doelen aan *buckets* (§ 7.2);
- het zoeken naar *associaties* van metingen en doelen (§ 7.3).

## 7.1 Bucket-coördinatenstelsel

De idee bij de methode is om de 2D-ruimte op te splitsen in zogenaamde *buckets* of hokjes, zodat een bucket-coördinatenstelsel ontstaat (zie figuur 11 links). Horizontaal en verticaal moeten evenveel buckets komen; het aantal hangt af van het aantal geïnitialiseerde doelen ($n$). Zowel horizontaal als verticaal moeten $\sqrt{n}$ buckets komen (naar boven afgerond). Dit komt er op neer dat ongeveer evenveel buckets zijn geïnitialiseerd als doelen, zodat bij goed gespreide doelen in elke bucket gemiddeld één doel ligt. Bij het zoeken naar toewijzingen van doelen en metingen, is dan de verwachting dat maar één toewijzingsmogelijkheid moet worden onderzocht, wat in § 7.2 verduidelijkt wordt.

Volgens Edahiro *et al.* (1984) en Zhang *et al.* (1992) moeten de buckets van gelijke grootte worden gemaakt (zoals in figuur 11 links). Maar in de toepassing van MTT geldt, dat dicht bij de radar (het centrum van het bucket-coördinatenstelsel) aanzienlijk meer metingen worden waargenomen. Rekening houdend met de kansdichtheidsverdeling van de radarplots




**Figuur 11.** *Links een bucket-coördinatenstelsel, de ruimte is verdeeld in 2-dimensionale buckets, in dit geval 16 buckets, zodat er 10 tot 16 doelen geïnitialiseerd moeten zijn. Rechts een logaritmisch bucket-coördinatenstelsel (de sensor bevindt zich in het centrum).*

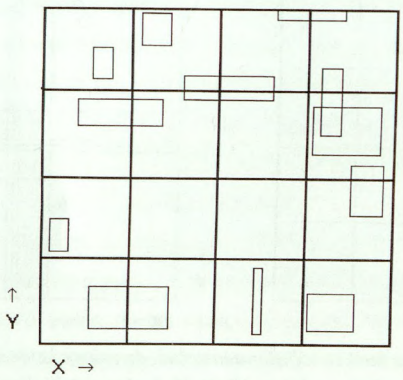
en de eis, dat er gemiddeld gesproken één doel in elke bucket ligt, wordt het gebruik van een logaritmische schaal voor de bucket-afmetingen zinvoller geacht, met de kleinste buckets dicht bij de sensor (figuur 11 rechts).

Aangezien de radar in poolcoördinaten meet zou het fysisch gezien correcter zijn om logaritmisch verdeelde schillen aan te brengen. Het MTT-algoritme, en dus de toewijzing, vindt in $(x, y)$-coördinaten plaats; met name het gebruik van $\Sigma_i$ maakt een implementatie in poolcoördinaten zeer problematisch. Verwacht wordt, dat de Cartesische aanpak niet tot fouten in de uiteindelijke clustering zal leiden.

De beschrijvingsmethode is gebaseerd op identieke rechthoekige buckets (figuur 11 links); de implementatie in dit onderzoek is gedaan met logaritmisch verdeelde buckets (figuur 11 rechts).

### 7.2 Toewijzing van doelen aan buckets

Nu het coördinatenstelsel is opgesteld, moeten doelen worden toegewezen aan de buckets. Hiertoe wordt per doel een rechthoek geplaatst rondom de validatie-regio van het doel. Het bepalen van de grootte van een rechthoek vindt op dezelfde wijze plaats als in de Uhlmann-heuristiek. Een doel wordt toegewezen aan een bucket indien een gedeelte van de rechthoek rondom het doel overlap heeft met de bucket. Een doel kan zo aan meerdere buckets worden toegewezen, zoals getoond in figuur 12.




**Figuur 12.** *Doelen (dunne rechthoeken) hebben overlap met de buckets (dikke rechthoeken). Waar overlap is wordt het doel toegewezen aan de bucket.*

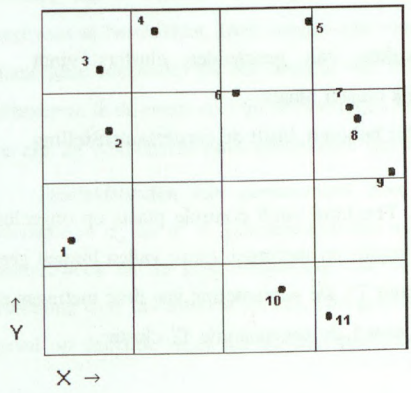
### 7.3 Associaties van metingen en doelen

In de associatiestap behoeft de bucketingmethode nu niet alle mogelijke combinaties van metingen en doelen af te gaan. Per meting wordt gekeken binnen welke bucket deze valt. Vervolgens worden de Mahalanobische afstanden $d_{ij}$ bepaald voor de meting met alle doelen die aan de bucket zijn toegewezen. In figuur 13 zijn de metingen in het bucket-coördinatenstelsel geprojecteerd. Eenvoudig is te zien in welke bucket ze vallen. Indien figuur 12 er naast wordt gehouden kan worden bepaald voor welke combinaties $d_{ij}$ wordt berekend. Is een doel toegewezen aan een meting op basis van de onderlinge statistische afstand, dan moeten deze worden geclusterd, zoals in § 3 beschreven.

## 8 De lower-bound methode

De lower-bound methode van Orr *et al.* (1992) lijkt sterk op de brute-force methode en moet eigenlijk worden gezien als een verbetering ervan. Ook Orr *et al.* (1992) beschouwen alle mogelijke combinaties zodat de rekentijd O(*mn*) is. Echter, deze ligt beneden de waarde voor de brute force methode, omdat niet voor alle combinaties $d_{ij}$ berekend wordt. Er wordt namelijk eerst een minder rekenintensieve lower bound test gedaan. Orr *et al.* (1992) stellen vast, dat het inverteren van $\Sigma_i$ de bottleneck vormt bij de berekening van $d_{ij}$ volgens (2), met name wanneer de radar meer doelsparameters meet (zoals de snelheid); dan neemt de dimensie van $\Sigma_i$ toe (die is $2 \times 2$ in ons voorbeeld).

Orr *et al.* (1992) merken op dat $d_{ij}$ niet voor elke combinatie bepaald hoeft te worden, als er een waarde $g_{ij}$ (eenvoudiger te berekenen dan $d_{ij}$) is te vinden, waarvoor $d_{ij} > g_{ij} > \gamma$. Zij benutten de Rayleigh-Ritz theorie in Horn *et al.* (1985) voor de positief definiete variantie-




**Figuur 13**
*De metingen (op het bucket-coördinatenstelsel geprojecteerd) kunnen alleen overlap hebben met de validatie-regio's van een doel dat aan dezelfde bucket is toegewezen.*

covariantiematrix $\Sigma_i$. De uitdrukking $y = x^T A\, x$ kan daarom van boven worden begrensd door: $y_1 = \lambda_{max}\, x^T x \geq y$, waarbij $\lambda_{max}$ de grootste eigenwaarde van matrix A is. Daar deze positief definiet is, geldt: $y_2 = \text{trace (A)}\, x^T x \geq y_1$ zodat

$$x^T A\, x \leq \lambda_{max} x^T x \leq \text{trace (A)}\, x^T x \tag{5}$$

De matrix $\Sigma_i$ heeft dezelfde eigenschappen als matrix A, behalve dat deze geïnverteerd moet worden voor de Mahalanobische afstand. Doordat $A = \Sigma_i^{-1}$ verandert het ongelijkheidsteken; voor de afstand $d_{ij}$ leidt dit tot de begrenzing:

$$d_{ij} \geq g_{ij} = (s_i - p_j)^T (s_i - p_j) / \text{trace}\,(\Sigma_i) \tag{6}$$

Dus als $g_{ij}$ groter is dan de threshold, is het berekenen van de Mahalanobische afstand overbodig. Dit kan verbeteringen opleveren voor de beschreven clusteringmethoden.

## 9 Vorming van superclusters

In §3 is vermeld dat indien een meting aan meerdere doelen is toegewezen, al deze doelen en de meting in één *supercluster* moeten komen, evenals alle metingen die aan de geclusterde doelen zijn toegewezen. Omdat bij de besproken toewijzingsmethoden op dezelfde manier superclusters gevormd moeten worden, behandelen we dit apart.

Nadat de associaties tussen doelen en metingen zijn vastgesteld, vindt per associatie clustervorming plaats, waarbij vijf situaties kunnen optreden:

1. als zowel doel als meting niet eerder geclusterd zijn wordt een nieuwe cluster geïnitialiseerd;
2. als het doel reeds tot een cluster behoort, maar de meting niet, wordt deze toegevoegd aan de doelcluster;
3. als de meting reeds tot een cluster behoort, maar het doel niet, wordt dit toegevoegd aan de metingcluster;
4. als meting en doel reeds deel uitmaken van gescheiden clusters vindt samensmelting tot een gemeenschappelijk cluster plaats;
5. als meting en doel reeds tot hetzelfde cluster behoren, blijft de clustersamenstelling ongewijzigd.

Op deze wijze worden superclusters gevormd. Tenslotte vindt controle plaats op ongeclusterde doelen (geen meting binnen de validatie-regio) en metingen (deze vallen binnen geen enkele validatie-regio, zie clusters 4 en 5 uit figuur 7). De verzameling van deze metingen en doelen wordt als een onafhankelijk cluster beschouwd, de zogenaamde '0'-cluster.

## 10 Testen van de clusteringmethoden

De behandelde methoden zijn getest op hun efficiëntie met als maat de benodigde tijd voor clustervorming. Er zijn vier methoden geïmplementeerd, namelijk:

de *brute-force* methode, de *binary tree* methode en de *bucketingmethode*

en ieder al of niet met *lower-bound* methode, dus resulterend in zes combinaties.

Het testen (op een SUN Sparc ultra workstation) is gedaan voor grote datasets, omdat clusteren volgens Cox (1993) dan een bottleneck kan gaan vormen. Het initialiseren van de metingen en de doelen valt buiten de tijdmeting, omdat binnen MTT alle inputvariabelen bekend zijn. De methoden zijn in standaard (ANSI) C geprogrammeerd en onder verschillende operating systems (UNIX, Windows, DOS) te compileren.

De benodigde input-gegevens die uit data-files gelezen worden, zijn:

1. de doelspositie in poolcoördinaten (afstand en azimut);
2. de onzekerheid met betrekking tot de doelspositie, uitgedrukt in de rotatiehoek van de foutenellips, $\sigma_x^2$ en $\sigma_y^2$; deze drie grootheden bepalen $\Sigma_i$ (zie § 3);
3. de plotpositie (de meting) in poolcoördinaten;

terwijl in de source code zijn opgenomen:

4. de threshold $\gamma$, die de afmeting van de validatie-regio bepaalt (zie § 3);
5. het bereik van de radar te gebruiken voor de opdeling van de ruimte in buckets (zie § 7.1).

Voor de vergelijking van de methodes zijn geen echte meetgegevens gebruikt: radar-data zijn veelal geclassificeerd. Wel is de kansdichtheidsverdeling van de plots (als functie van de afstand) gebruikt, zoals gemeten met een werkelijke rondzoekradar met een bereik van 74 km tijdens slecht weer (daar komen false alarms in voor, maar deze zijn door hun aard niet als zodanig aan te wijzen). In het kader van dit onderzoek was het niet mogelijk over meer gegevens te beschikken. Deze empirische verdeling is gebruikt om in *Matlab* (software voor numerieke wiskunde) tot een afstand van 80 km 2 data-sets doelsgegevens en metingen te genereren. In de eerste set zijn de doelen en metingen *onafhankelijk* gegenereerd, in de tweede set zijn de synthetische plots *afhankelijk* van de gesimuleerde doelsposities.

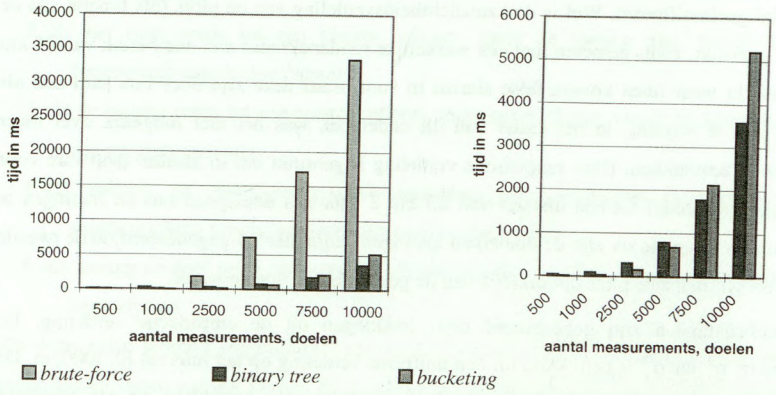
Doelsafstanden zijn gegenereerd door trekkingen uit de empirische verdeling. De variantie in $\sigma_x^2$ en $\sigma_y^2$ is getrokken uit een uniforme verdeling op het interval $[0, 4000]$ m. De azimut-hoek en de hoekverdraaiing van de foutenellips zijn getrokken uit een uniforme verdeling over het interval $[0, 2\pi]$. Het genereren van plot-posities gaat in het onafhankelijke geval op dezelfde wijze als de doelsposities. In het afhankelijke geval is een doel uniform

getrokken uit de lijst met doelen, vervolgens is daar een rechthoek omheen geplaatst volgens (3) en (4) en vindt plotgeneratie plaats door middel van uniforme trekking binnen die box.

In het *onafhankelijke* geval zijn 500, 1000, 2500, 5000, 7500 en 10.000 als aantallen doelen en metingen gebruikt. Elke tijdswaarneming (in milliseconden) behorende bij één simulatie bestaat uit een gemiddelde van 20 onafhankelijke simulatieruns, wat voldoende blijkt voor een betrouwbaar gemiddelde; de resultaten staan in figuur 14. Verwacht wordt dat in het *afhankelijke* geval meer doelen en metingen geclusterd moeten worden; daarom zijn in dit geval batches van 250, 500, 750, 1000, 1250, 1500 en 2000 doelen en metingen gebruikt. Ook hier bestaat één simulatie uit een gemiddelde van 20 onafhankelijke waarnemingen.

De tijd nodig voor het clusteren bleek niet significant meetbaar bij minder dan 250 doelen en 250 metingen, wat onderstreept dat clustering alleen bij veel doelen een bottleneck kan vormen.

De *lower-bound methode* is als toevoeging op de andere drie methoden getest, maar geeft in de 2D-geometrie geen significante tijdwinst. De brute force methode is met lower-bound altijd trager dan zonder. Voor de andere twee methoden is hier weinig over te zeggen. In het afhankelijke geval bij de binary tree methode met of zonder lower-bound zijn de verschillen miniem. Bij de bucketingmethode geeft de lower-boundmethode over het algemeen een zeer kleine versnelling. Bij de bespreking van de resultaten blijft de lower-bound methode dan ook buiten beschouwing.




**Figuur 14.** *De rekentijden voor de clustermethoden bij gelijk aantal doelen en metingen; rechts hetzelfde diagram onder weglating van de resultaten van de brute-force methode.*

De *brute-force methode* is in het afhankelijke geval de traagste methode behalve bij 250 doelen, de binary tree methode is dan trager. In het onafhankelijke geval kost de brute-force methode ook de meeste tijd, behalve als het aantal metingen veel kleiner of veel groter is dan het aantal doelen (een ordegrootte verschil). De genoemde orde $O(mn)$ van de methode volgt ook uit de testresultaten. De rekentijd van *de binary tree methode* groeit sneller dan de vermelde orde $O(mn^{1-1/d})$ door de gebruikte quicksort-methode, die namelijk $O(n \log n)$ kost, maar in de worst-case $O(n^2)$, als de elementen al gesorteerd zijn. Het traagst (het snelst) is de binary tree methode indien het aantal metingen veel groter (kleiner) is dan het aantal doelen. Gebleken is dat het opstellen van de boom bij veel metingen de meeste tijd kost.

De *bucketingmethode* is in het afhankelijke geval de snelste methode, behalve als het aantal metingen veel kleiner is dan het aantal doelen; dan is de binary tree methode het snelst. In het onafhankelijke geval blijkt de bucketingmethode het snelst indien weinig doelen zijn geïnitialiseerd en veel metingen. Dit kan worden verklaard als de meeste tijd wordt besteed aan het maken van een bucket-coördinatenstelsel en het plaatsen van doelen in de buckets. De bucketingmethode is in het onafhankelijke geval twee keer de langzaamste, namelijk bij 500 metingen en 7500 of 10.000 doelen.

Het dichtst bij de werkelijkheid liggen de gevallen met een gelijk aantal metingen en doelen. Het is een reële veronderstelling dat de radar ongeveer net zoveel metingen detecteert als geïnitialiseerde doelen. In deze zogenaamde evenwichtssituatie blijkt het verschil tussen het onafhankelijke en het afhankelijke geval miniem te zijn. In het laatste geval wordt er meer geclusterd; blijkbaar kost het daadwerkelijk clusteren nauwelijks tijd. In de evenwichtssituatie blijkt ook dat de bucketingmethode veel sneller is dan de binary tree methode, als het aantal metingen / doelen hoogstens gelijk is aan 5000, terwijl het omgekeerde zich voordoet als het aantal metingen / doelen minstens gelijk is aan 7500 (zie figuur 14).

## 11 Conclusies

De gegeven rekenresultaten zijn gebaseerd op data voor een 2-dimensionale ruimte. De beschreven methoden zijn aan te passen voor clustering van data met hogere dimensies; wat dan een efficiënt clusteringalgoritme is, verdient nader onderzoek. Bij het trekken van conclusies kan meespelen dat synthetische data in plaats van meetgegevens zijn gebruikt en ook de beperkte omvang van de beschikbare gegevens.

Zowel het onafhankelijke als het afhankelijke geval zijn onderzocht, waartussen het verschil miniem is gebleken. Voor de clusteringtijd maakt het weinig uit of doelen en

metingen daadwerkelijk geclusterd moeten worden. Het zoeken naar de juiste associaties blijkt de meeste tijd te kosten. Bij weinig doelen blijkt de clustertijd insignificant te zijn, zodat de keuze voor een te implementeren clusteringmethode in de 2-dimensionale geometrie gemaakt moet worden op basis van een situatie met veel doelen.

De lower-bound methode, die geen tijdwinst blijkt op te leveren voor de beschouwde clusteringmethoden, kan wellicht in hogere dimensies wel voordelen opleveren.

De brute-force methode blijkt volgens verwachting niet efficiënt te zijn. Vooral bij grotere aantallen doelen en metingen groeit de tijd voor het clusteren vergeleken met de andere methoden explosief, zodat bij implementatie van deze methode in het MHT-algoritme in het onafhankelijke geval de doelen niet meer real-time gevolgd kunnen worden.

Tabel 1 geeft aan welke methode het best dan wel het slechtst is in een bepaalde situatie. Daarbij is de combinatie veel metingen en weinig doelen of andersom, weinig aannemelijk. De bucketingmethode heeft bij weinig geïnitialiseerde doelen de voorkeur, de binary tree methode wordt pas interessant bij grofweg meer dan 5000 geïnitialiseerde doelen.

Of de bucketingmethode voor weinig doelen en de binary tree methode voor veel doelen in de 2-dimensionale geometrie het efficiëntst zijn, kan slechts blijken na verder onderzoek naar mogelijke andere algoritmen. De lower-bound methode levert voor de 2-dimensionale datasets geen tijdwinst op. Nader onderzoek kan uitwijzen of dit ook geldt bij grotere dimensies van de dataset. Voor de binary tree en bucketingmethode zonder lower-bound verbeteringsstap is er onderscheid te maken naar het aantal doelen: bij weinig doelen (minder dan 250) vormt clusteren geen bottleneck. De clusteringtijd is voor beide methoden niet significant meetbaar. Bij veel doelen blijkt de bucketingmethode sneller te zijn tot 5000 doelen en evenzoveel metingen, daarna wordt de binary tree methode efficiënter.

**Tabel 1.** *De beste en de slechtste methoden voor clustering.*

| | *beste* methoden | |
|---|---|---|
| | *weinig* (≈ 1000) doelen | *veel* (> 5000) *doelen* |
| *weinig* of *veel* metingen | bucketing | binary tree |
| | *slechtste* methoden | |
| *weinig* (≈ 1000) metingen | Brute-force | bucketing |
| *veel* (> 5000) metingen | binary tree | brute-force |

## 12 Referenties


Aho, A.V., J.E. Hopcroft en J.D. Ullman (1983) '*Data structures and algorithms*', Addison-Wesley, Reading, Mass.

Bar-Shalom, Y. en T.E. Fortmann (1988) '*Tracking and data association*', Academic Press, Mathematics in Science and Engineering **179** , San Diego, CA.

Bar-Shalom, Y. en Xiao-Rong Li (1995) '*Multitarget-multisensor Tracking: principles and techniques*', YBS Publishing, Storrs.

Collins, J.B. en J.K. Uhlmann (1992) '*Efficient gating in data association with multivariate Gaussian distributed states*', IEEE Transactions on Aerospace and Electronic systems **28** (3), 909–916.

Cox, I.J. (1993) '*A review of statistical data association techniques for motion correspondence*', International Journal of Computer Vision **10**, 53–66.

Edahiro, M., I. Kokubo en T. Asano (1984) '*A new point-location algorithm and its practical efficiency - Comparison with existing algorithms*', ACM Transactions on Graphics **3**, 86–109.

Heijn, J.S. (1997) '*The association problem in Multiple Target Tracking*', Afstudeerscriptie, Faculteit der Economische Wetenschappen en Econometrie, Universiteit van Amsterdam.

Horn, R.A. en C.R. Johnson (1985) '*Matrix analysis*', Cambridge University Press.

Orr, M.J.L., J. Hallam en R.B. Fisher (1992) '*Fusion through interpretation*', Research Paper 572, Dept. of Artificial Intelligence, Edinburgh University, 1992. Abstract: '*Fusion through interpretation*', in G. Sandini (ed.), Second European Conference on Computer Vision, Springer-Verlag, New York, 801–805.

Preparata, F.P. en M.I. Shamos (1997) '*Computational Geometry: an introduction*', Springer-Verlag, New York.

Reid, D.B. (1979) '*An algorithm for tracking Multiple Targets*', IEEE Transactions on Automatic Control, Vol. AC-24, 843–854.

Theil, A. en A.G. Huizing (1997) '*The Multitarget Tracking Testbed: functionality of the tracking module*', version 1.0, TNO-FEL report: FEL-97-A232, Den Haag.

Uhlmann, J.K. (1992) '*Algorithms for Multiple-Target Tracking*', American Scientist **80**, 128–141.

Zhang, Z.Y. en O.D. Faugeras (1992) '*Three-dimensional motion computation and object segmentation in a long sequence of stereo frames*', International Journal of Computer Vision **7**, 211–241.



Ontvangen: februari 2001
Geaccepteerd: 24 april 2001