LOCATIEBEPALING VAN EEN ROBOT

MET BEHULP VAN LANDMARKS IN GRIJSBEELDEN

Naam :	Sander Beekmans
Studierichting :	Kunstmatige Intelligentie Autonome Systemen
Faculteit :	FNWI
Afstudeerbegeleider :	Ben Kröse
Locatie afstudeerproject :	WCW
Datum :	oktober 2001
Kernwoorden :	robot lokalisatie, landmark detectie

UNIVERSITEIT VAN AMSTERDAM

Samenvatting

In deze scriptie zal ik ingaan op het detecteren van landmarks in de omgeving van de robot om ze vervolgens te kunnen gebruiken voor het bepalen van zijn locatie in het wereldmodel. Een landmark wordt hier gedefinieerd als een herkenningspunt voor de robot. Deze herkenningspunten heeft de robot nodig om zijn omgeving te kunnen herkennen en daarmee zijn locatie in deze omgeving te kunnen bepalen. Om dit te kunnen doen heeft de robot een representatie nodig van het wereldmodel. Deze representatie zal bestaan uit een database met daarin beelden van alle locaties in het wereldmodel, gerepresenteerd als een set van landmarks, die hij in de initialisatie fase heeft gevonden. Het nieuwe beeld, met daarin de landmarks die de robot waarneemt wanneer deze zijn positie moet bepalen, wordt vergeleken met die in de database, waarna er een locatie zal worden vastgesteld.
De methode die hier beschreven zal worden is van toepassing op een ruimte binnen in een gebouw, zoals een kantoor of woonruimte. De robot moet de ruimte van te voren eerst eenmaal hebben bezocht zodat deze een representatie hiervan kan maken.

Inhoudsopgave

Hoofdstuk 1

Inleiding

Voor een robot is het van belang, bij het uitvoeren van zijn taak, over een goed navigatiesysteem te beschikken. Het is namelijk van belang dat een robot, zonder hulp van buitenaf, zich in een omgeving kan verplaatsten en weet waar hij zich in deze omgeving bevindt. Hiervoor maakt een robot gebruik van een Perception-Action architectuur [1]. Dit houdt in dat de robot eerst zijn omgeving waarneemt, en aan de hand daarvan een beslissing neemt over zijn volgende actie. Het perception gedeelte bestaat uit het herkennen van situaties en deze vervolgens koppelen aan de juiste actie. Het herkennen van een situatie bestaat vaak uit het vergelijken van de situatie met de vooraf geleerde situaties. Deze scriptie houdt zich bezig met dat deel van het perception gedeelte, de robot moet in staat zijn om zijn omgeving te kunnen herkennen om daarmee zijn locatie te kunnen vaststellen.

Er zijn verschillende methodes ontwikkeld voor het bepalen van de locatie van een robot. Een daarvan is odometrie. Hier maakt de robot gebruik van het aantal omwentelingen van de wielen. Als dit aantal bekend is, en er is bekend hoeveel de robot aflegt wanneer deze één omwenteling maakt, kan de afgelegde afstand berekend worden. Nadeel hiervan is, dat deze methode nogal onnauwkeurig is, vooral wanneer er een lange afstand is afgelegd. De wielen kunnen b.v slippen, waardoor de afgelegde afstand niet meer overeenkomt met die, die de robot meent te hebben afgelegd. Daarom is er een methode nodig die, door gebruik te maken van de omgeving, zijn locatie opnieuw kan bepalen.
Toch kan de odometrie wel nuttig zijn als gebruik gemaakt wordt van een soort Markov localization, waarbij zowel een motion model wordt gebruikt als een observation model [11]. Het motion model beschrijft dan wat het effect is van een bewegingsopdracht van de robot op zijn locatie. Als dit nu gebruikt wordt in combinatie met het observation model, waarbij de omgeving van de robot gebruikt wordt voor de locatiebepaling, kan de locatie beter worden bepaald. (m.b.v het observation model zou een locatie kunnen worden gevonden die niet mogelijk is gezien het motion model)

Bij het gebruik maken van de omgeving voor het bepalen van de locatie van de robot in het wereldmodel zijn er verschillende manieren waarbij gebruik gemaakt wordt van een sensor die de wereld waarneemt. Twee soorten sensoren worden veel gebruikt, namelijk sonar en beeldsensoren.

Bij de eerste wordt er gebruik gemaakt van ultrasone en infrarode sensoren [10].
Door gebruik te maken van sonar kunnen heel goed afstanden worden bepaald, maar de richting is moeilijker vast te stellen. Hierdoor is het moeilijk te bepalen welk deel van de omgeving wordt geobserveerd.
Aangezien de robot (een Nomad Scout robot) is uitgerust met een standaard camera en hyperbolische spiegel, zal de methode die in dit verslag wordt besproken gebruik maken van beeldinformatie en in dit geval met behulp van herkenningspunten (landmarks) in het beeld.

Bij het visueel detecteren van landmarks zijn er verschillende manieren voorhanden. De landmarks kunnen worden gedetecteerd dmv hun kleur [7] of vorm [1, 2]. Wanneer gebruik gemaakt wordt van kleur, wordt er gekeken naar de kleurhistogrammen van de landmarks en wordt de geometrische informatie niet meegenomen. In het geval dat er gebruik gemaakt wordt van vorm, zal gekeken worden hoe het landmark in kwestie er uit ziet en wordt dus juist wel gelet op geometrische informatie.
Voor het bepalen van de positie van de robot wordt gebruik gemaakt van de landmarks die de robot in beeld heeft wanneer deze zijn locatie wil weten. Er wordt dan gekeken naar eenzelfde soort combinatie van landmarks die de robot eerder is tegengekomen, toen deze zijn omgeving geleerd had.

De representatie van het wereldmodel van de robot is in dit onderzoek van te voren vastgelegd. Dmv de beeldinformatie die de robot krijgt wordt de omgeving waarin de robot zich bevindt vergeleken met de representaties in het wereldmodel en op deze manier de locatie gevonden die het meest lijkt op de huidige omgeving.

1.1 Doelstelling

Het uiteindelijke doel is een robot, die zich op eigen kracht kan voortbewegen in de ruimte, zonder tussenkomst van de mens. De robot moet zijn locatie kunnen bepalen aan de hand van de landmarks die hij ziet. Deze landmarks staan ook in de database, die in beginsel gemaakt is bij het opzetten van een model van de omgeving.
In het beeld dat de robot ziet moeten de landmarks gevonden worden als zijnde de duidelijkste kenmerken in het beeld, m.a.w delen die opvallen in het beeld en dus tekenend zijn voor dit beeld.
Op deze manier moet de robot zich dus door de ruimte kunnen bewegen zonder hierbij te botsen of dat hij zijn locatie niet kan bepalen en de mens moet inspringen.
Het hier beschreven experiment is van toepassing als de robot zijn locatie in de omgeving niet meer weet en deze moet kunnen bepalen. Er wordt dus geen enkele informatie gebruikt van een eventuele locatie (een zekere voorkennis), deze moet geheel bepaald kunnen worden aan de hand van het beeld van de omgeving.

1.2 Korte beschrijving van de eigen aanpak

Het gedeelte van het navigatiesysteem van de robot, dat ik in dit verslag zal onderzoeken, is het gedeelte voor het bepalen van de locatie van de robot. Als de robot zijn locatie weet, kan het deze gebruiken voor het maken van een volgende actie, zoals een beweging in de richting van een doellocatie. Het bepalen van zijn locatie bestaat uit twee delen:

1. *Het detecteren van landmarks in de omgeving van de robot.*
   Deze stap bestaat uit het maken van een beeld van de omgeving waar de robot zich bevindt, en uit dit beeld de landmarks halen die tekenend zijn voor het beeld.

2. *Deze landmarks koppelen aan een locatie.*
   Hier worden de gevonden landmarks vergeleken met die uit de database, die de robot heeft gevonden tijdens het initialisatie proces, en wordt aan de hand hiervan de locatie van de robot bepaald.

1.3 Overzicht van het verslag

In dit verslag komen de volgende onderwerpen aan bod:

In hoofdstuk 2 zal ik enige methoden uit de literatuur beschrijven voor het detecteren van landmarks in een beeld. De beelden die hier dan gebruikt worden zullen grijsbeelden zijn, omdat dit ook de beelden zijn die gebruikt zijn in mijn onderzoek. Er wordt dus niet ingegaan op het gebruik van kleurhistogrammen voor het vinden van de landmarks. Verder zal in hoofdstuk 2 beschreven worden hoe deze landmarks gebruikt kunnen worden voor het bepalen van de locatie van de robot.

Hoofdstuk 3 zal een beschrijving zijn van de methode die ik heb toegepast voor het detecteren van de landmarks en het bepalen van de locatie aan de hand van deze landmarks.

In hoofdstuk 4 zullen de resultaten beschreven worden die ik heb behaald. Deze zullen worden beschreven aan de hand van de locatie die de robot vindt en de locatie waar hij zich oorspronkelijk bevindt. Er kan dus gekeken worden hoe het systeem werkt aan de hand van de fout in afstand tussen de oorspronkelijke locatie en de gevonden locatie.

Als laatste zal in hoofdstuk 5 de conclusie komen alsmede de verbeteringen die aangebracht kunnen worden in het systeem.

Hoofdstuk 2

Achtergrond

Bij het vergelijken van beelden zijn er verschillende mogelijkheden voorhanden. Een simpele manier is, om een globale correlatie maat tussen de geleerde beelden en het beeld van de huidige situatie te nemen. Er wordt dan gekeken hoe goed de beelden op elkaar lijken. Een voordeel van deze methode is dat hij simpel en relatief robuust is. Grote nadelen zijn echter dat deze methode niet robuust is bij het samentrekken of wijder worden van een beeld als gevolg van het voor of achteruit bewegen van de robot en dat de methode niet robuust is voor object occlusie of verplaatsing. Om deze nadelen uit de weg te gaan, kan het beeld worden gerepresenteerd aan de hand van enkele gebieden, die tekenend zijn voor dit beeld, de landmarks.

Wanneer een robot bij zijn locatiebepaling gebruik maakt van landmarks, kan er gekozen worden uit verschillende representaties voor deze landmarks. Zij kunnen kunstmatig aangebracht worden in de omgeving [9], of er kan gebruik gemaakt worden van de objecten in de ruimte zelf, en deze te gebruiken als landmarks.
In dit tweede geval zijn er meerdere mogelijkheden, waar op gelet kan worden bij het zoeken naar de landmarks. Deze kunnen herkend worden aan hun kleur of vorm. Wanneer, zoals de in dit verslag beschreven methode, gebruik gemaakt wordt van grijsbeelden, valt de methode om een landmark aan zijn kleur te detecteren weg.
Wat nu van belang is, is om naar de vorm te kijken van een landmark. Om de landmarks dan te detecteren wordt gebruik gemaakt van een edge detector.

In dit hoofdstuk zullen twee verschillende methodes aan bod komen die gebruik maken van natuurlijke landmarks voor het bepalen van de locatie van de robot. In beide gevallen zal gebruik gemaakt worden van grijsbeelden. De eerste methode zal gebruik maken van een edge detector voor het zoeken naar de landmarks in het beeld, de tweede kijkt alleen in horizontale richtingen naar grote overgangen in intensiteit (wat is een zeker opzicht ook een soort edge detection is) om op deze manieren randen op te sporen.
Om deze twee methoden te beschrijven zal ik in het eerste geval gebruik maken van het werk van Sim (Sim, 1998) en in het tweede geval dat van Gaussier e.a. (Gaussier, 2000)

Ik zal hier een versimpelde versie beschrijven van de eigenlijke methodes die gebruikt worden, zodat duidelijk wordt hoe de manieren werken voor de locatiebepaling. Voor een uitgebreide beschrijving verwijs ik naar de beide papers [1] en [2].
Paragraaf 2.1 zal de methode van Sim behandelen, paragraaf 2.2 die van Gaussier e.a..

2.1 De methode van Sim

2.1.1 Edge detection gebruiken voor het detecteren van landmarks

Een edge in een beeld bestaat uit een sterke intensiteits verandering. Omdat deze edges in een beeld meestal object randen voorstellen, is de edge detection een goede manier om een beeld op te delen in verschillende delen die verschillende objecten voorstellen.

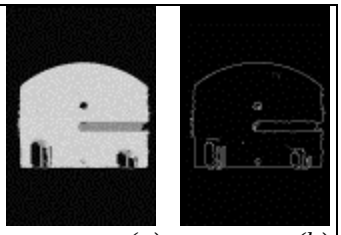
In deze methode wordt een landmark voorgesteld als het lokale maximum van de edge element distribution in een beeld, dus daar waar er veel edges voorkomen in een kleine omgeving. Het blijkt namelijk dat de edge map van een beeld veel informatie bevat, en een bijkomend positief punt is, dat deze niet gevoelig is voor lichtveranderingen. In dit geval zal gebruik gemaakt worden van de Canny edge detector (Canny, 1986).

Canny edge detector

De Canny edge detector heeft als input een grijsbeeld, en als output een beeld waarop de posities te zien zijn van de intensiteits veranderingen.

De werking van deze edge detector kan worden opgedeeld in verschillende fases.
Eerste stap is het toepassen van een Gaussische filter. Hierdoor wordt ruis uit het beeld gefilterd. Belangrijk is wel, dat de breedte van de Gaussische kernel niet te groot wordt gekozen, want dan zullen de fijnere details in het beeld verdwijnen.
Vervolgens wordt voor elke pixel de gradiënt grootte (gradiënt grootte is gelijk aan $\|\nabla f\|,\ \nabla f(x,y) = \begin{cases} f_x(x,y) \\ f_y(x,y) \end{cases}$) en gradiënt richting berekend. De gradiënt grootte is namelijk een maat voor de randsterkte, omdat het de locale verandering berekent in grijswaarde in de richting van een hogere grijswaarde. De gradiënt richting is gelijk aan de richting van de rand, omdat het in de richting wijst van de grootste toename in grijswaarde.
Als nu vervolgens een gradiënt grootte van een pixel groter is, dan dat van zijn twee buren in de gradiënt richting, dan wordt het gezien als een rand, anders als de achtergrond. Op deze manier wordt een dikke lijn die de rand voorstelt teruggebracht tot een lijn van één pixel breed.

Laatste stap is om de zwakke randen te verduidelijken dmv hysteresis thresholding. Dit weerhoudt een rand ervan om opgebroken te worden in stukjes. Er worden dan twee thresholds gebruikt, T1 en T2, met T1 > T2. Elk punt boven T1 wordt nu bewaard, evenals elk segment dat daaraan vastzit en dat boven T2 ligt.



(a) *(b)*

figuur 2.1
beeld a is het beeld voor de Canny edge detection, beeld b nadat de Canny edge detection is toegepast

Om nu de landmarks in een beeld te vinden, wordt de volgende formule toegepast:

$$D(x) = \frac{1}{\|\Omega\|} \int_{x' \in \Omega} E(x')dx'$$

Hier is $E : \Re^2 \rightarrow \Re^2$ de output van de Canny edge detector toegepast op een beeld.
D is hier de density (dichtheid van de edges) van E in de omgeving Ω. ($x \in \Re^2$)
Vrij vertaald zegt deze formule: *De density is de som van de output van E van elk punt in de omgeving* Ω*, genormaliseerd door de gehele omgeving* Ω.

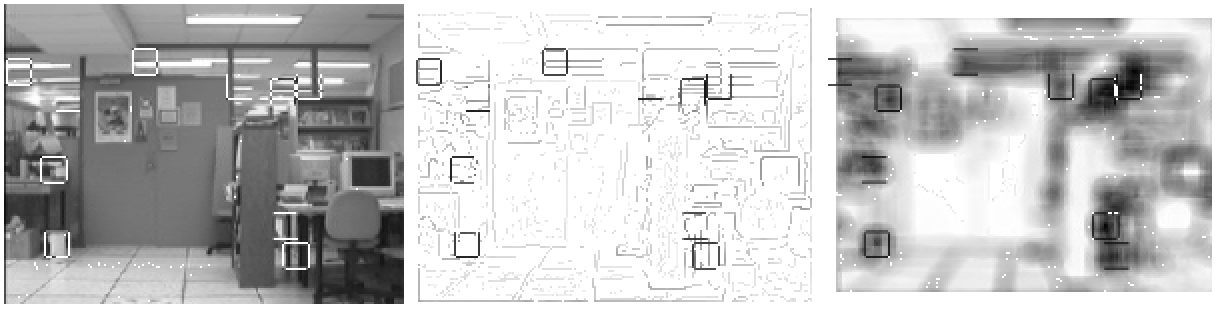
Om nu vervolgens de set van landmarks te krijgen, wordt de volgende formule toegepast:

$$C = \left\{ l \mid \|D(l)\| > D_m + tD_s \wedge \|D(l)\| \geq \|D(l')\| \forall (l') \in \Omega \right\}$$

Elk landmark $l \in \Re^2$ stelt een positie in het beeld voor, D_m en D_s zijn gemiddelde en standaard deviatie van D over het gehele beeld en t is een threshold die door de gebruiker van te voren is bepaald.

Dus: C bestaat uit een set van lokale maxima van D die hoger liggen dan een bepaalde threshold.

Voorbeeld:

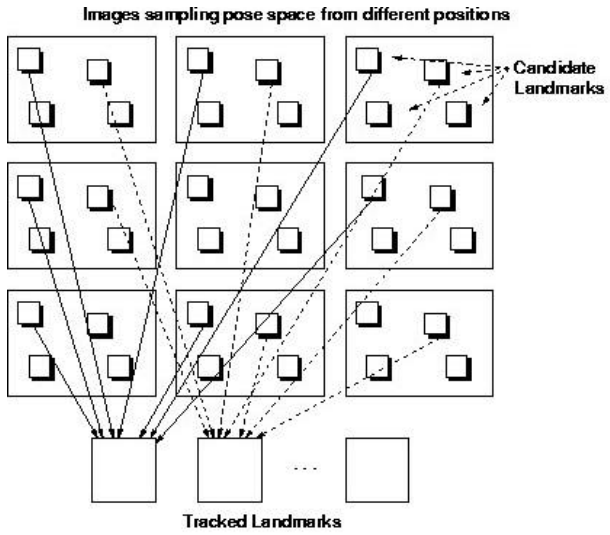


figuur 2.2

Links is het originele beeld, in het midden hetzelfde beeld, maar dan met een Canny edge detectie erop toegepast. Het rechter plaatje is de uitkomst van de density functie toegepast op het middelste beeld. Donkere gebieden representeren een hogere density dan lichtere gebieden. In elk beeld zijn de potentiële landmarks aangegeven met een vierkant.

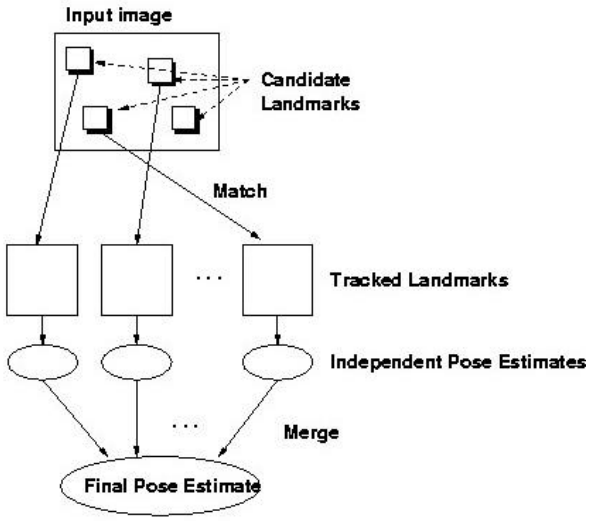
2.1.2 Locatiebepaling m.b.v de gevonden landmarks

De beelden die als leerset gaan dienen worden verzameld. Deze beelden bevatten een aantal verschillende poses in de omgeving. Uit deze beelden worden de landmark candidates gehaald zoals is beschreven in paragraaf 2.1.1. Hierna worden de tracked landmarks gemaakt. Deze bestaan uit een set van candidate landmarks. (zelfde landmark gevonden in beelden met verschillende poses van de robot) De tracked landmarks worden gerepresenteerd door een karakteristiek prototype, dat verkregen wordt door het coderen van een set van candidate landmarks m.b.v. de principale componenten analyse. Vervolgens wordt deze set van tracked landmarks opgeslagen.



figuur 2.3
candidate landmarks die samen de tracked landmarks vormen

Op deze manier is er dus een representatie gemaakt van het wereldmodel, van de gevonden landmarks is de locatie bekend.
Wanneer nu de locatie van de robot bepaald moet worden, worden uit het beeld dat de robot van de locatie maakt, de candidate landmarks gehaald. Deze worden gematched met de geleerde sets van tracked landmarks. Dit matchen gebeurt door het candidate landmark te matchen met de prototypes van de gevonden tracked landmarks in de eerste fase. Voor elke candidate landmark die gematched is met een tracked landmark, wordt de positie bepaald. Er wordt een reconstructie van de candidate berekend die gebaseerd is op de tracked candidates in de tracked landmark (Er zitten in een tracked landmarks verschillende candidates. Van elk van deze candidates in de positie bekend) Het eindresultaat is een lineaire combinatie van de verschillende posities van tracked landmarks.
Omdat er nu voor elke candidate landmark uit het nieuwe beeld een positie is bepaald, wordt van al deze een gemiddelde genomen als de eindpositie.



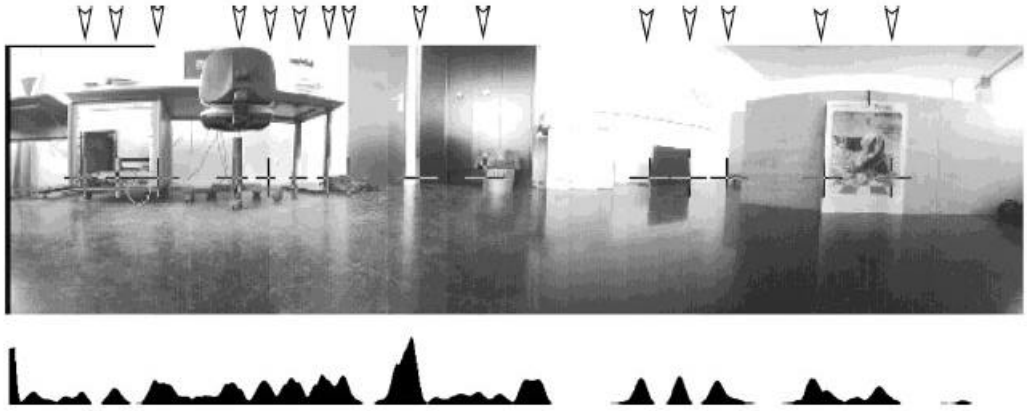
figuur 2.4
het locatiebepalings proces

2.2 De methode van Gaussier e.a.

2.2.1 Intensiteits overgangen gebruiken voor het detecteren van landmarks

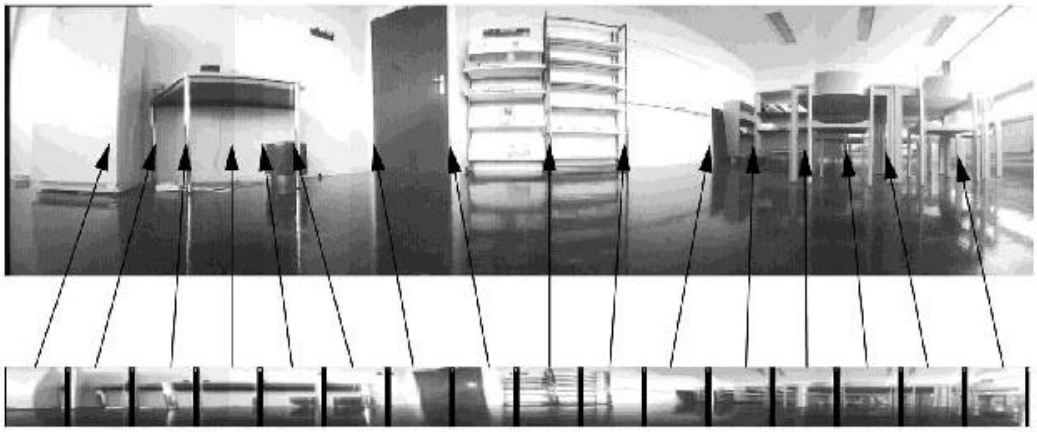
De methode die in deze paragraaf wordt beschreven voor het detecteren van landmarks in een panoramabeeld is de basis voor de methode die ik gebruikt heb in mijn onderzoek. Gebruikte formules en dergelijke zullen dan ook in het volgende hoofdstuk worden beschreven.

Het beeld dat hier gebruikt wordt is een panoramabeeld. Van dit panoramabeeld wordt van elke beeld kolom het gewogen gemiddelde genomen. Punten in het midden van de kolom krijgen grotere gewichten mee, dan die aan de randen. Er ontstaat dan een ééndimensionaal signaal, dat vervolgens gedifferentieerd wordt. Wat je nu krijgt, is een signaal met maxima en minima. Deze worden gebruikt voor de bepaling van de plaats van de landmarks, die hier worden voorgesteld als lokale beeldjes in het panoramabeeld. Deze lokale beeldjes zullen dan op de plek zitten waar zo'n maximum voorkomt. In een beeld krijg je verscheidene van deze maxima (waarbij ook inbegrepen de absolute waarde van de minima) en je kan b.v. de 15 grootste waarden hiervan nemen als de middelpunten van de lokale beeldjes.



figuur 2.5
panoramabeeld met de 15 gekozen maxima

Vervolgens worden deze lokale beeldjes, met de 15 maxima als middelpunt, geconstrueerd. Van het panoramische beeld wordt om een maximum een band genomen van 148 x 288 pixels (maximum is middelpunt). Deze wordt vervolgens geschaald naar een beeldje van 32 x 32 pixels.



figuur 2.6
locale 32 x 32 beeldjes die geleerd worden als landmarks

Het panoramabeeld, dat oorspronkelijk een grootte had van 1246 x 288 pixels wordt op deze manier dus gerepresenteerd als 15 beeldjes van 32 x 32 pixels. Gebieden die geen belangrijke informatie bezitten voor de herkenning van het beeld worden op deze manier niet bewaard.

2.2.2 Locatiebepaling m.b.v de gevonden landmarks

Elke landmark dat nu gevonden wordt in het nieuwe beeld, wordt vergeleken met de al geleerde lokale beeldjes die zijn gevonden in de initialisatiefase. (de vergelijking bestaat uit de norm van het verschil tussen de pixels van beiden) Diegene die het best corresponderen, worden gebruikt als de landmarks. Hun positie in het beeld worden vergeleken met de posities in de geleerde panorama's. De som van deze absolute hoeken geeft de gelijkheid tussen de panorama's. Als uiteindelijke locatie zal nu de locatie worden genomen van het panoramabeeld uit de database, dat het meest overeenkomt met het nieuwe beeld, m.a.w, waarbij de som van de absolute hoeken het kleinst is.

2.3 Conclusie

Vanwege het feit, dat er een basismethode nodig is voor het detecteren van de landmarks in het beeld, is het van belang om van beide manieren, beschreven in paragrafen 2.1 en 2.2, de voor en nadelen te bekijken, om zo een keuze te kunnen maken welke manier het beste geschikt is om te gebruiken. Wat dan opvalt is, dat de manier zoals gebruikt door Sim, het beste effect lijkt te hebben als gebruik gemaakt wordt van beelden met een hoge resolutie, waarbij er veel attributen in het beeld voorkomen. De beelden die ik gebruik voor dit onderzoek, beelden die over het algemeen in gangen zijn genomen, bevatten veel intensiteits overgangen in horizontale richting, zoals voorkomt bij o.a deuren. Ook zijn er weinig kleine opvallende kenmerken in de beelden, zoals o.a. terminals, stoelen etc, die van belang zijn als de landmarks voorwerpen in de omgeving voorstellen. Er zullen dus bijna geen gebieden in het beeld voorkomen waar de dichtheid van de randen erg hoog is. Vandaar dat de basismethode die Gaussier toepast, zoals beschreven in paragraaf 2.2, beter van toepassing is op de beelden die ik zal gebruiken dan dat van Sim. De methode van Gaussier is namelijk vooral gericht op dat soort kenmerken die in onze gebruikte panoramabeelden veel voorkomen.

Hoofdstuk 3

Beschrijving van de eigen methode

In dit hoofdstuk zal mijn methode worden beschreven voor de locatiebepaling van de robot. De robot die hier gebruikt wordt is een Nomad Scout robot, die gebruik maakt van een omnidirectioneel beeld.

figuur 3.1
de Nomad Scout robot

figuur 3.2
beeld verkregen door Nomad Scout Robot

Het proces bestaat uit twee delen. Het eerste deel is het construeren van een database. In deze database is de representatie opgeslagen van de omgeving waar de robot zich kan bevinden. Deze wordt gebruikt als vergelijking voor het tweede deel, waarin de robot zijn huidige locatie moet kunnen opvragen doormiddel van het vergelijken van het beeld dat hij nu ziet met de representaties in de database. Dit vergelijken van beelden wordt gedaan aan de hand van landmarks in het beeld. De landmarks die ik zal gebruiken worden beschreven als een duidelijk kenmerk in het beeld. Het gaat hier niet om kunstmatige landmarks die in de omgeving zijn geplaatst (denk aan rode stippen etc) en de landmarks zullen ook niet bepaalde voorwerpen als stoelen of terminals voorstellen. De landmarks die hier worden gebruikt

bevatten een deel van het beeld dat duidelijk kenmerkend is voor dat beeld, waar wordt gelet op grote overgangen van licht/donker intensiteit. Voorbeeld van een belangrijk kenmerk kan een donkere deurpost op een lichte achtergrond zijn.

3.1 Construeren van de database

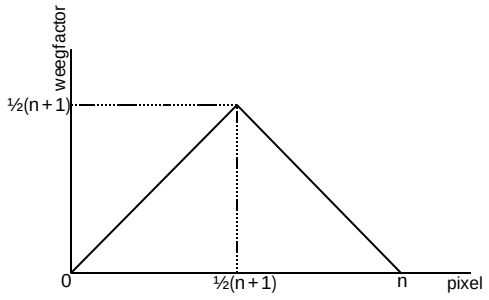
3.1.1 Beelden van het wereldmodel verkrijgen

Om een model van de wereld te maken moet de robot eerst op alle locaties in zijn wereld geweest zijn. De robot wordt rondgeleid door de omgeving en maakt hierbij steeds beelden, die opgeslagen worden samen met de locatie van de robot, waar deze zich bevond bij het maken van het beeld. Op deze manier ontstaat er een database met beelden, gekoppeld aan een locatie, van de omgeving waar de robot zich kan bevinden.

3.1.2 Locaties van de landmarks zoeken

Het omnidirectionele beeld uit figuur 3.2 wordt omgezet tot een panoramabeeld. Eerste stap is om in het panoramabeeld op zoek te gaan naar de kenmerkende eigenschappen in het beeld. Ik heb er voor gekozen om te zoeken naar grote overgangen in de intensiteit van de grijswaarden. Hierbij wordt gekeken waar de overgangen in horizontale richting het grootst zijn.
Om de locatie van deze overgangen in het beeld te vinden, wordt in het panoramabeeld van elke pixelkolom een gewogen gemiddelde genomen, waarbij de waarden in het midden van de pixelkolom een groter gewicht meekrijgen dan die aan de buitenkanten op de volgende manier:

$$\overline{p}_j = \frac{\sum_{i=1}^{\frac{1}{2}(n+1)} i * p_j^i + \sum_{i=\frac{1}{2}(n+1)+1}^{n} (n+1-i) * p_j^i}{n}$$



figuur 3.3

Hierbij is n het aantal pixels in een kolom, $\overline{p}_j$ het gemiddelde van de pixels in kolom j en p_j^i de waarde van pixel i in kolom j.

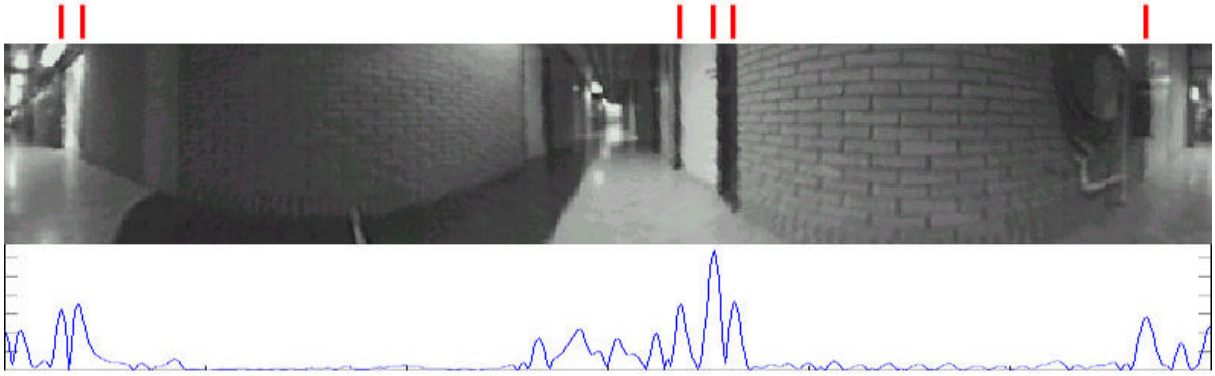
De binnenste pixels krijgen een hogere wegingsfactor, omdat dit het belangrijkste deel van het beeld is. Onder en bovenzijde van het beeld bevatten vaak vloer en plafond, die voor het herkenningsproces niet van belang zijn.

Het panoramabeeld dat gebruikt wordt, is 600 x 99 pixels, dus er ontstaat een eendimensionaal signaal met hierin de 600 gemiddeldes.

Als nu vervolgens dit signaal gedifferentieerd wordt, zal er een signaal ontstaan waar de waarden hoog zijn bij een grote overgang tussen de gemiddeldes van de pixelkolommen. Er wordt gedifferentieerd met behulp van Fuzzy afgeleiden [3]. De formule hiervoor is:

$$F'(x) = f'(x) * G(x)$$

waarbij $F'(x)$ de afgeleide is, die verkregen is na een convolutie van de gewone afgeleide $f'(x)$ van het eendimensionale signaal met de gaussian $G(x)$. De gaussian die gebruikt wordt heeft een standaard deviatie van 3 pixels. De reden voor het gebruik van een fuzzy afgeleide is, dat ruis op deze manier weg gefilterd wordt. In figuur 3.4 is een panoramabeeld weergegeven met bijhorende afgeleide.



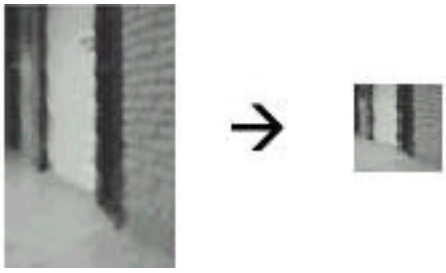
figuur 3.4
panoramabeeld met bijhorende absolute waarden van de afgeleide

Bij een verhoging van de standaard deviatie van de gaussian, zullen toppen die nu dicht bij elkaar liggen samensmelten tot één top, een verlaging zal juist meerdere smallere toppen geven.

Nu de waarden van de afgeleiden bekend zijn, wordt gekeken waar de zes hoogste waarden voorkomen. De hoogte van de waarde van de afgeleide geeft namelijk aan hoe duidelijk het kenmerk in het beeld is. (het aantal van zes is zo gekozen, omdat gebleken is dat dit aantal over het algemeen de belangrijke gebieden in een beeld vertegenwoordigt en de lage toppen in een beeld op deze manier niet meegenomen worden) De bijbehorende locaties van deze waarden zullen vervolgens gebruikt worden als locaties van de landmarks. Het beeld is 600 pixels breed, dus de locatie wordt opgeslagen als een getal $1 \leq l_i \leq 600$ met i het nummer van het landmark l. In figuur 3.4 is met rode markeringen aangegeven waar deze locaties in het beeld zich bevinden.

3.1.3 De landmarks construeren

Als de locaties in het panoramabeeld bekend zijn, worden de landmarks uit het beeld geconstrueerd. Uit het panoramabeeld wordt een band genomen van 64 pixels breed en 99 pixels hoog. Het middelpunt van deze band is de locatie die eerder gevonden is. Deze band van 64 bij 99 pixels wordt vervolgens geschaald naar een 32 x 32 groot landmark zoals te zien is in figuur 3.5.



figuur 3.5
schaling van het beelddeel naar een landmark

De waarden voor de pixels in het landmark worden als volgt berekend:

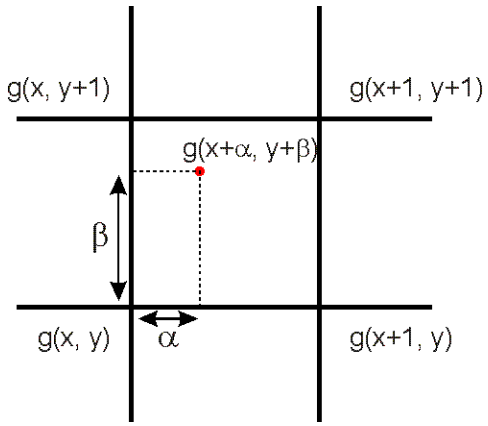
$$f(x, y) = g(2x, \frac{99}{32} y)$$

waarbij $f(x, y)$ de waarde van het pixel in het landmark is en $g(2x, \frac{99}{32} y)$ de waarde van het pixel in de band uit het panoramabeeld. Deze waarden 2 en 99/32 zijn zo gekozen, omdat de oorspronkelijke breedte van 64 pixels moet worden geschaald naar 32 pixels. De hoogte van 99 pixels wordt ook teruggebracht naar 32 pixels. Omdat de waarden in g niet allemaal op een rasterpunt liggen, worden deze berekend dmv bilineaire interpolatie [3].

bilineaire interpolatie

Wanneer de grijswaarde van een punt berekend moet worden dat geen rasterpunt is, waarbij de x en y waarde geen integers zijn, kan gebruik worden gemaakt van bilineaire interpolatie. Het gaat hier dan om een punt $g(x+a, y+b)$, waarbij x en y hele getallen zijn en a en b tussen 0 en 1 liggen. Deze waarde kan dan berekend worden met behulp van de volgende formule:

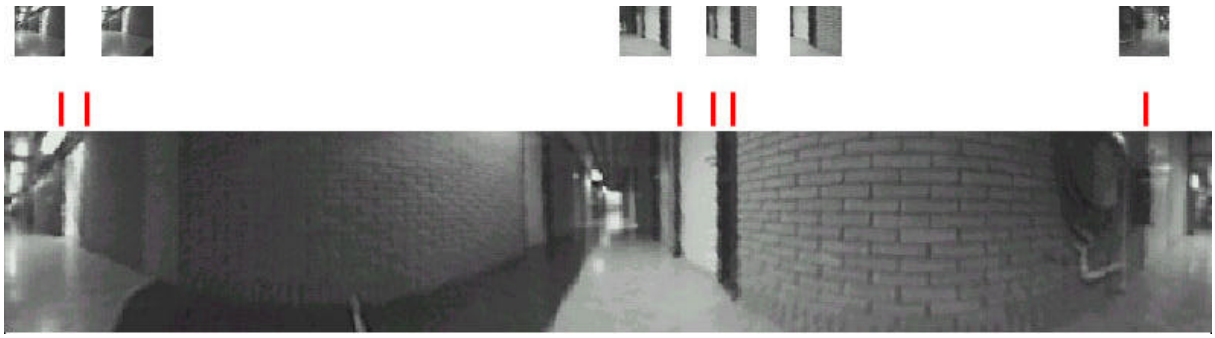
$$\begin{aligned} g(x+a,y+ß) = {} & (1-a)(1-ß)\, g(x,y) + \\ & a(1-ß)\, g(x+1,y) + \\ & (1-a)ßg(x,y+1) + \\ & aßg(x+1,y+1) \end{aligned}$$



figuur 3.6
bilineaire interpolatie

Het landmark wordt vervolgens opgeslagen.

In figuur 3.7 is bij elk van de gevonden locaties het landmark weergegeven.



figuur 3.7
gevonden landmarks in het panoramabeeld

Als dit vervolgens voor alle panoramabeelden in de dataset wordt gedaan, ontstaat een database met daarin de gevonden landmarks voor alle beelden. Deze set bestaat uit het aantal beelden maal 6 landmarks per beeld. Bij een klein gebied, waar dus weinig panoramabeelden nodig zijn om de omgeving te beschrijven, is de grootte van de set landmarks nog aanvaardbaar, maar bij een verdubbeling van het oppervlakte zal ook het aantal landmarks worden verdubbeld. Vandaar dat er dus een manier moet worden gezocht om de groei van het aantal landmarks niet toe te laten nemen naarmate het oppervlakte wordt vergroot. Een manier om dit te bereiken is om de landmarks te gaan clusteren, zodat er een groep prototypes ontstaat. Als er nu nieuwe landmarks worden gevonden, zullen deze kunnen worden vervangen door een prototype.
Voor het clusteren heb ik het k-means algoritme gebruikt [5].

k-means algoritme

het algoritme voor k-means clustering werkt als volgt:

1. Kies een k aantal clusters. Neem k willekeurige samples als clustermiddens.
2. Bereken voor elk sample de afstand tot de clustermiddens. Kies als cluster het clustermidden dat het dichtst bij ligt.
3. Verschuif de clustermiddens zo, zodat ze het middenpunt worden van de samples die tot dat cluster behoren.
4. Herhaal stap 2 en 3, totdat er geen clustermidden meer verschuift.

Vervolgens moet een manier worden gevonden om de dimensionaliteit van de landmarks te verlagen, want de dimensionaliteit van 1024 (32 x 32 pixels) is te hoog om goed te kunnen clusteren. Een manier om de dimensionaliteit te verlagen, is door gebruik te maken van principale componenten analyse [6].

Principale componenten analyse

De principale componenten analyse wordt toegepast wanneer het van belang is om de dimensionaliteit waarmee gewerkt wordt te verlagen.
Het doel is dan, het projecteren van de vectoren $\boldsymbol{z}^n$ in een d-dimensionale ruimte $(z_1,...,z_d)$ naar een $\boldsymbol{x}^n$ in een M-dimensionale ruimte $(x_1,...,x_M)$, waarbij $M < d$.
Een vector $\boldsymbol{z}$ kan gerepresenteerd worden als

$$\boldsymbol{z} = \sum_{i=1}^{d} x_i \boldsymbol{u}_i$$

Hier is $\boldsymbol{u}_i$ een orthonormale vector.
Voor x_i vinden we nu de volgende formule:

$$x_i = \boldsymbol{u}_i^T \boldsymbol{z}$$

M.a.w een rotatie van het coördinaten assenstelsel.
Vervolgens willen we enkel een subset $M < d$ van de basisvectoren $\boldsymbol{u}_i$, zodat er alleen gebruik gemaakt wordt van M coëfficiënten x_i. De overige coëfficiënten worden dan vervangen door een constante b_i. Elke vector $\boldsymbol{z}$ wordt dan benaderd door:

$$\tilde{\boldsymbol{z}} = \sum_{i=1}^{M} x_i \boldsymbol{u}_i + \sum_{i=M+1}^{d} b_i \boldsymbol{u}_i$$

De originele vector $\boldsymbol{z}$ had eerst een dimensionaliteit van d, maar deze is nu verlaagd naar een dimensionaliteit van $M < d$.
Als we nu te maken hebben met een set N vectoren $\boldsymbol{z}^n$ $(n = 1,...,N)$, willen we de basisvectoren $\boldsymbol{u}_i$ en de coëfficiënten b_i zo kiezen, dat de originele vectoren $\boldsymbol{z}$ het beste benaderd worden. Hierbij kunnen we kijken naar de fout die gemaakt wordt voor elke $\boldsymbol{z}^n$ als we deze dimensiereductie toepassen. Deze wordt gegeven als:

$$\boldsymbol{z}^n - \tilde{\boldsymbol{z}}^n = \sum_{i=M+1}^{d} (x_i^n - b_i)\boldsymbol{u}_i$$

Om deze nu zo klein mogelijk te krijgen, wordt gekeken naar de som van de kwadratische fout. Deze moet zo klein mogelijk zijn. De kwadratische fout is gegeven als

$$E_M = \frac{1}{2}\sum_{n=1}^{N} \left\| \boldsymbol{z}^n - \tilde{\boldsymbol{z}}^n \right\|^2 = \frac{1}{2}\sum_{n=1}^{N} \sum_{i=M+1}^{d} (x_i^n - b_i)^2$$

De som van de kwadratische fout is nu gedefinieerd als

$$E_M = \frac{1}{2}\sum_{i=M+1}^{d}\sum_{n=1}^{N}\{\boldsymbol{u}_i^T(\boldsymbol{z}^n - \bar{\boldsymbol{z}})\}^2 = \frac{1}{2}\sum_{i=M+1}^{d}\boldsymbol{u}_i^T\Sigma\boldsymbol{u}_i$$

Hier is gebruik gemaakt van het feit, dat als we de afgeleide van E_M met betrekking tot b_i naar 0 brengen, we voor b_i vinden

$$b_i = \frac{1}{N}\sum_{n=1}^{N}x_i^n = \boldsymbol{u}_i^T\bar{\boldsymbol{z}}$$

Het gemiddelde van de vectoren $\boldsymbol{z}$, $\bar{\boldsymbol{z}}$, is gelijk aan

$$\bar{\boldsymbol{z}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{z}^n$$

Bij de som van de kwadratische fout wordt gebruik gemaakt van de covariantie matrix Σ van de set vectoren $\{\boldsymbol{z}^n\}$. Deze is gegeven als

$$\Sigma = \sum_n(\boldsymbol{z}^n - \bar{\boldsymbol{z}})(\boldsymbol{z}^n - \bar{\boldsymbol{z}})^T$$

Het is nu dus de bedoeling om deze kwadratische fout te minimaliseren. Dit minimum vindt plaats als geldt

$$\Sigma\boldsymbol{u}_i = l_i\boldsymbol{u}_i$$

Hier zijn de basisvectoren dus de eigenvectoren van de covariantie matrix.
Het minimum van E_M vinden we nu dus als

$$E_M = \frac{1}{2}\sum_{i=M+1}^{d}l_i$$

M.a.w het minimum wordt gevonden door de $d-M$ kleinste eigenwaarden l_i te nemen en deze samen met de corresponderende eigenvectoren niet te gebruiken.

Het algoritme voor de principale componenten analyse is als volgt:

- Bereken gemiddelde van de vectoren $\boldsymbol{z}^n$ en trek dit gemiddelde vervolgens af van elk van deze vectoren.
- Bereken de covariantie matrix. Hierdoor worden de eigenvectoren en eigenwaarden gevonden.
- Bewaar alleen de eigenvectoren die corresponderen met de M grootste eigenwaarden.
- Projecteer de input vectoren $\boldsymbol{z}^n$ op de eigenvectoren, zodat de componenten gevonden worden van de getransformeerde vectoren $\boldsymbol{x}^n$ in de M-dimensionale ruimte.

Wanneer deze beiden gebruikt worden is het van belang dat een goed aantal principale componenten en clusters wordt gekozen.

3.1.4 Het aantal principale componenten en clusters bepalen

Om de dimensie van de set landmarks te verkleinen (deze is nu 1024), als doel om het clusteren van de landmarks te verbeteren, heb ik op een set van 894 landmarks uit de database de principale componenten analyse toegepast. Om nu het beste aantal principale componenten vast te stellen, worden twee methodes vaak toegepast: het Kaiser criteria en de scree test. Allereerst staat in tabel 3.1 uitgezet per principale component wat de eigenwaarde is, en voor hoeverre deze meetelt aan het percentage van de totale variantie.

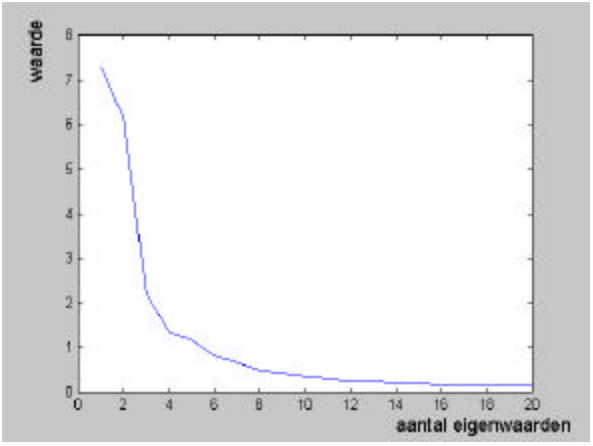
	Eigenwaarde	% totale variantie	Cumulatieve %
1	7.2758	27.25	27.25
2	6.1528	23.05	50.30
3	2.1953	8.22	58.52
4	1.3430	5.03	63.55
5	1.1759	4.40	67.95
6	0.8126	3.04	70.99
7	0.6651	2.49	73.48
8	0.4850	1.82	75.30
9	0.4168	1.56	76.86
10	0.3546	1.33	78.19
11	0.3130	1.17	79.36
12	0.2431	0.91	80.27
13	0.2367	0.89	81.16
14	0.2077	0.78	81.94
15	0.1916	0.72	82.66
16	0.1641	0.61	83.27
17	0.1517	0.57	83.84
18	0.1409	0.53	84.37
19	0.1370	0.51	84.88
20	0.1315	0.49	85.37

tabel 3.1

Hieronder staat voor beide manieren het aantal principale componenten dat gevonden wordt als beste aantal principale componenten om te behouden, de rest wordt niet bewaard.

- **Het Kaiser criterium:**
  Alleen diegene worden gebruikt, waarvan de eigenwaarde groter is dan 1. Anders gezegd houdt dit in, dat een component alleen bewaard wordt, “als deze minstens evenveel afleidt als het equivalent van één originele variabele” (Kaiser, 1960).

- **De scree test:**
  “scree”is een geologische term die staat voor het puin dat zich onder aan een steile berg verzamelt. Hiermee wordt verwezen naar het punt waar de helling van het plot overgaat van sterk dalend naar flauw dalend. Deze methode werd voor het eerst beschreven door Cattell (Cattell, 1966).
  Om deze methode toe te passen, wordt er een plot gemaakt van de eigenwaarden:



figuur 3.8

Om nu het aantal eigenvectoren te kiezen, wordt gekeken waar er een knik is in het figuur en de steilheid van het plot overgaat van steil naar vlak.

Bij het Kaiser criterium worden soms te veel principale componenten bewaard, in tegenstelling tot de scree test, hier zijn het er soms juist te weinig. Maar over het algemeen werken beide methodes goed onder normale omstandigheden, waarbij er relatief weinig componenten zijn en veel datareeksen.
In het geval waar gebruik gemaakt wordt van het Kaiser criterium zullen er 5 eigenvectoren bewaard worden. Gebruik makend van de scree test worden er 4 eigenvectoren gekozen.

De principale componenten die gevonden worden (bij een keuze van 5 componenten) zien er als volgt uit:

figuur 3.9
principale vectoren

Hierbij zijn de waarden zo bewerkt, zodat ze af te beelden zijn. Sommige waarden van de componenten zijn namelijk negatief en dus niet als grijswaarde af te beelden. De waarden zijn dus eerst verschoven totdat ze allemaal positief zijn, en vervolgens is er contrast stretching gebruikt zodat ze het gehele grijswaarden gebied bestrijken.

Vervolgens wordt het aantal clusters bepaald.
De landmarks worden ingedeeld in K clusters $\{C_1, C_2, ..., C_K\}$, waarbij cluster C_k bestaat uit n_k landmarks.
Het clustermidden van een cluster wordt gedefinieerd als

$$\boldsymbol{m}^{(k)} = (1/n_k)\sum_{i=1}^{n_k} \boldsymbol{x}_i^{(k)}$$

Hierbij is $\boldsymbol{x}_i^{(k)}$ het landmark *i* in cluster C_k
Om nu te bepalen wat de fout is voor elk landmark, wordt gekeken naar de afstand van het landmark tot zijn clustermidden. Een manier om dit te doen is gebruik te maken van de

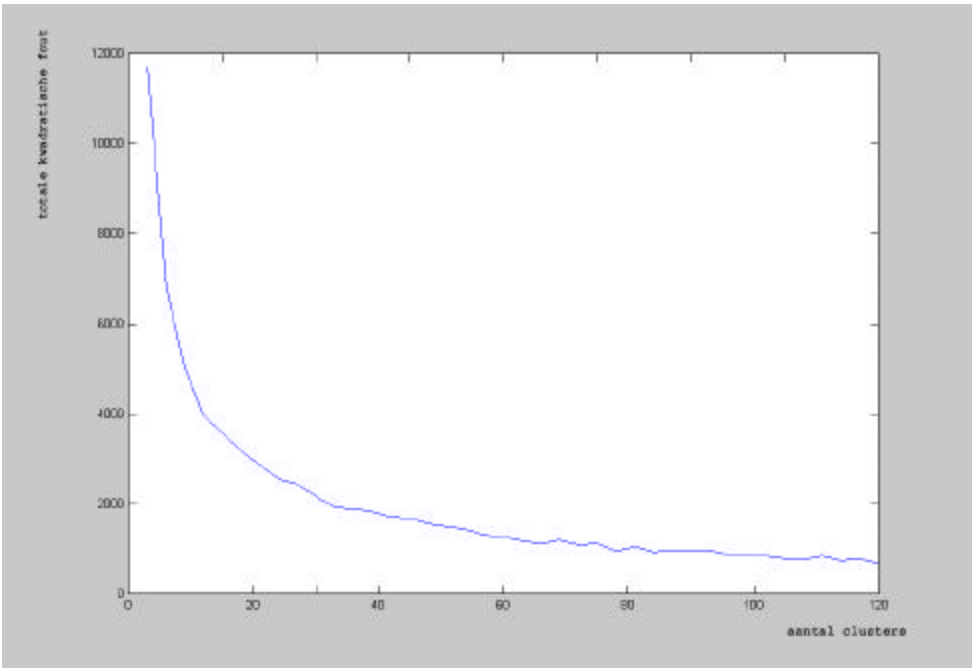
kwadratische fout. De kwadratische fout is de som van de kwadratische Euclidische afstand van elk landmark tot zijn clustermidden $\boldsymbol{m}^{(k)}$:

$$e_k^2 = \sum_{i=1}^{n_k} (\boldsymbol{x}_i^k - \boldsymbol{m}^k)^T (\boldsymbol{x}_i^k - \boldsymbol{m}^k)$$

Vervolgens worden de kwadratische fouten bij elkaar opgeteld zodat de totale kwadratische fout ontstaat:

$$e^2 = \sum_{i=1}^{K} e_i^2$$

Om nu te kijken wat het optimale aantal clusters is, wordt het aantal clusters uitgezet tegen e^2

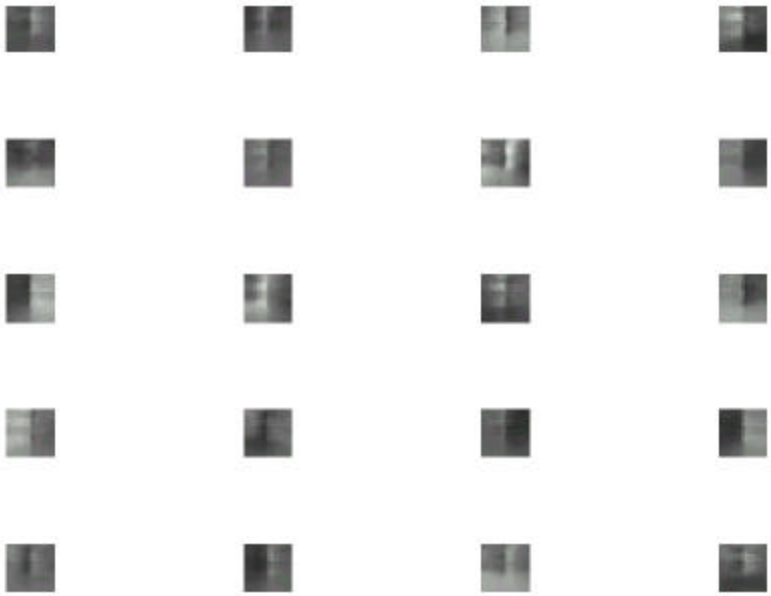


figuur 3.10
clusterplot gebruik makend van 5 principale componenten

Er zijn nu verschillende manieren om het aantal clusters te bepalen dat een goed resultaat geeft. Het is hier de bedoeling dat het aantal clusters niet te groot wordt, want dan bestaat er een grote kans dat landmarks ten onrechte in verschillende clusters worden ingedeeld.
Bij een te laag aantal clusters daarentegen zullen veel landmarks bij hetzelfde cluster worden ingedeeld, waarbij er ook veel foutief tot hetzelfde cluster zullen worden gerekend.
Een methode is om na het clusteren een cluster te splitsen, als het te veel clusters bevat waarbij de variantie onwaarschijnlijk hoog is. Twee clusters worden daarentegen bij elkaar gevoegd, als de clustermiddens dicht bij elkaar liggen. De waarde die gekoppeld wordt aan ‘dichtbij’ wordt door de gebruiker van te voren bepaald (ISODATA, Ball & Hall, 1964).
Vaak wordt ook gekeken naar zgn. outliers. Dit zijn patterns die ver verwijderd zijn van de rest van de data, of clusters met een aantal patterns lager dan een van te voren ingestelde threshold. Deze worden dan verwijderd.

Als figuur 3.10 gebruikt wordt, kan gekeken worden waar een sterke daling overgaat in een gematigde daling. Deze ligt rond de 20 clusters.

Deze clustermiddens zijn weergegeven in figuur 3.11.



figuur 3.11
gevonden clustermiddens

3.1.5 Representatie van de database

Nu er voor elk landmark een prototype is gevonden (de clustermiddens), kan elk panoramabeeld dus voorgesteld worden als een reeks van 6 categorieën die de landmarks voorstellen en een bijbehorende reeks die de locaties van die landmarks in het panoramabeeld weergeeft. Eveneens is aan elk van deze representaties een locatie gekoppeld die aangeeft waar dit panoramabeeld in de omgeving van de robot zich bevindt. Elk panoramabeeld wordt nu dus gerepresenteerd als zijnde:

$$landmarknummer(lm_1,...,lm_6)$$
$$landmarklocatie(loc_1,...,loc_6)$$
$$panoramabeeldlocatie(locx, locy)$$

Hier is lm_i het nummer van landmark *i*, waarbij deze een getal kan aannemen van 1 t/m het aantal gekozen clusters.
loc_i is de locatie van lm_i in het panoramabeeld, met $1 \le loc_i \le 600$.
$locx$ en $locy$ zijn de x en y waarden van de locatie van het panoramabeeld in de omgeving van de robot.

3.2 Het bepalen van de locatie van een nieuw beeld

3.2.1 Een nieuw beeld omzetten

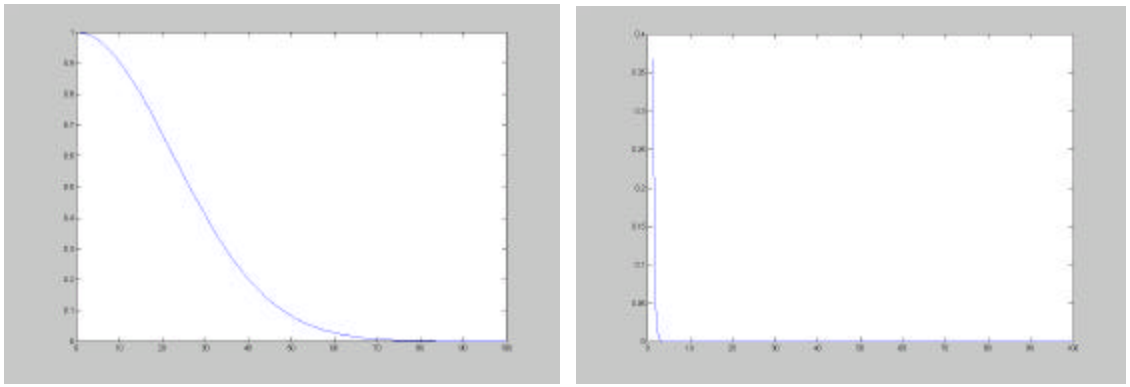
De methode die beschreven is in hoofdstuk 3.1 voor het representeren van een panoramabeeld wordt toegepast op alle panoramabeelden die deel gaan uitmaken van de database van de robot. Over het algemeen is het de bedoeling dat in deze database panoramabeelden zitten van de locaties waar de robot zich kan bevinden. Er ontstaat dan nu een database met daarin alle panoramabeelden gerepresenteerd als reeksen van getallen. Wanneer de robot nu zijn locatie moet kunnen bepalen, maakt deze een beeld van zijn omgeving, en wordt dit beeld op dezelfde manier gerepresenteerd als de panoramabeelden in de dataset. De locaties van de landmarks worden eerst gezocht, waarna de landmarks gemaakt worden. Deze landmarks worden vervolgens op de principale componenten geprojecteerd. Vervolgens wordt voor elk van deze landmarks de afstand berekend tot elk clustermidden. Deze afstand tot elk clustermidden zegt iets over de kans dat een bepaald landmark tot dat cluster C behoort. De kans wordt berekend op de volgende manier:

$$P(\boldsymbol{x}^i \in C_j) = \frac{1}{\sum_{j=1}^{20} e^{-d_{ij}^2 * a}} e^{-d_{ij}^2 * a}$$

Hierbij is d_{ij} de afstand tussen landmark i en clustermidden j en is gegeven als

$$d_{ij} = \left| \boldsymbol{m}^j - \boldsymbol{x}^i \right|$$

en a is een van te voren ingestelde constante. Deze constante is te gebruiken om de *e*-macht te vormen, bij een lage respectievelijk hoge a zal deze er als volgt uitzien:



figuur 3.12 a) en b)
a) lage a (0,001)
b) hoge a (1,0)

Er is hier gebruik gemaakt van een *e*-macht, omdat wordt aangenomen dat de clusters gaussisch verdeeld zijn. De a heb ik hier zo gekozen, zodat deze het beste resultaat lijkt te geven, hij zou ook bepaald kunnen worden aan de hand van de verdeling van de data. Hier zou dan voor elk cluster de Mahalanobis afstand berekend moeten worden.
Om nu vervolgens te bepalen welke labels een landmark krijgt, wordt gekeken welke clustermiddens staan voor een gelijkenis van 70 %. Op deze manier kan een landmark dus

meerdere labels krijgen en zal het nieuwe panoramabeeld worden voorgesteld door meerdere reeksen. Door gebruik te maken van een grens van 70 % zal over het algemeen een niet al te groot aantal reeksen ontstaan.

3.2.2 Verschillende reeksen vergelijken

Wanneer voor een nieuw beeld de reeksen zijn gevonden, moeten deze worden vergeleken met de reeksen in de database, om zo te bepalen met welke panoramabeelden het nieuwe beeld de beste gelijkenissen vertoont. Hier is het noodzaak om een goed algoritme te vinden om de reeksen, die voorgesteld worden als strings, te vergelijken. De volgende string matching algoritmes zijn bekend [4]:

- exact string matching
- approximate string matching

Bij een exact string matching algoritme gaat het erom dat een bepaalde string b exact voorkomt in een (mogelijk langere) tekst a. Deze methode is niet direct van toepassing, omdat het mogelijk moet zijn dat de strings van elkaar kunnen verschillen. De gedachte hierachter is dat twee panoramabeelden niet exact dezelfde landmarks hoeven te bezitten, er kan verschil zijn tussen de panoramabeelden doordat b.v in het ene beeld een landmark niet zichtbaar was doordat er iets voor was geplaatst.
Hier moet dus een approximate string matching algoritme gebruikt worden. De meest bekende algoritmes maken gebruik van de Hamming afstand (Hamming, 1950) of de Levenshtein afstand (Levensthein, 1975).
De Hamming afstand tussen twee strings van dezelfde lengte telt het aantal posities waar de karakters in de beide strings verschillen. De Levenshtein afstand telt het aantal operaties op een string b, zodat deze past in tekst a. Een operatie kan zijn:

1. Een vervanging van een karakter in b door een ander karakter. Het karakter correspondeerde met een ander karakter in a.
2. Een toevoeging van een karakter in b. Een karakter in tekst a correspondeerde met een lege plaats in string b.
3. En verwijdering van een karakter uit b. Een karakter in string b correspondeerde met een lege plaats in tekst a.

Nadeel van de methode die gebruik maakt van de Hamming afstand is, dat als er in een string een landmark mist (en dus niet is vervangen door een ander landmark) de rest van de posities ook verschillende landmarks bevat.

Voorbeeld 1:

string a : 1 2 3 4 5 6
string b : 1 3 4 5 6 8

Dit probleem is op te lossen door gebruik te maken van de Levenshtein afstand. Hier kan een landmark worden toegevoegd, zodat de opvolgende landmarks ook weer dezelfde positie hebben. (in het voorbeeld wordt bij string b na landmark 1 een landmark 2 toegevoegd en vervolgens wordt landmark 8 verwijderd, de strings zijn nu gelijk en de Levenshtein afstand is in dit geval dus 2)
In dit geval werkt het algoritme goed, maar in sommige gevallen niet:

Voorbeeld 2:

string a : 4 3 6 2 5 1
string b : 4 3 8 6 7 1

Om deze strings gelijk te maken moeten er in string b drie landmarks vervangen worden door een ander. Maar als gekeken wordt naar het aantal gezamenlijke landmarks zijn dat er 4, dus twee landmarks zijn verschillend. Worden de landmarks die verschillend zijn uit beide strings verwijderd, dan krijgen we de volgende twee strings:

string a : 4 3 6 1
string b : 4 3 6 1

Het algoritme dat ik gebruik gaat dan ook als volgt te werk:

- Bepaal alle landmarks *lm* die zowel in string a als in string b zitten.

$$\{\forall lm_i \mid lm_i \in (a \cap b)\}$$

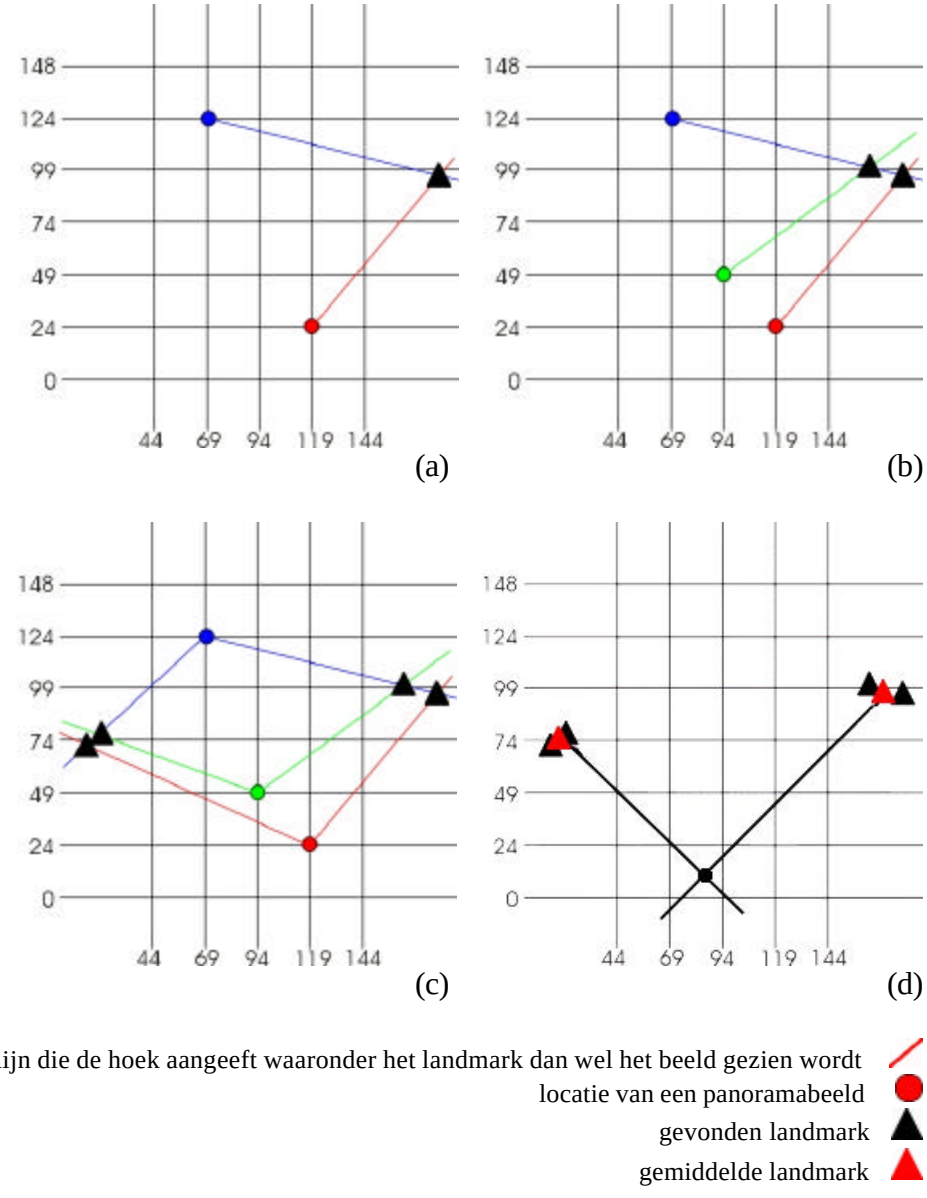
- Verwijder uit beide strings de landmarks die niet in beiden voorkomen.
- Dupliceer string a, om rekening te houden met draaiing van het beeld.
- Pas een exacte string matching toe, om na te gaan of de volgorde van de landmarks klopt.

Het dupliceren van string a is nodig, vanwege het feit dat beide oorspronkelijke panoramabeelden onder een andere hoek genomen zouden kunnen zijn. Het laatste landmark in een beeld zou dus evengoed het eerste kunnen zijn geweest.
Het aantal landmarks dat uit beide strings verwijderd mag worden heb ik ingesteld op maximaal twee. Dit wil zeggen dat de twee panoramabeelden minimaal vier gezamenlijke landmarks moeten bevatten en dat deze landmarks in dezelfde volgorde in de beelden moeten voorkomen. Is aan deze beide criteria voldaan, dan wordt de string uit de database als gelijkend beschouwd met het nieuwe beeld.

3.2.3 De locatie van het nieuwe beeld berekenen

Wanneer bekend is welke beelden uit de database gevonden zijn, kan m.b.v deze beelden de locatie berekend worden van het nieuwe beeld. Hierbij wordt gebruik gemaakt van de hoeken waaronder een bepaald landmark voorkomt in het panoramabeeld. De locatie van een landmark in het beeld geeft namelijk aan onder welke hoek de robot dit landmark heeft waargenomen. Het panoramabeeld is 600 pixels breed, dit is gelijk aan 360°. Eén pixel staat nu voor 0,6°.
Er kan nu tussen twee gevonden panoramabeelden uit de database een gezamenlijk landmark worden genomen. Vanuit de locaties van het panoramabeelden wordt een lijn getrokken in de richting van het landmark, waar deze twee lijnen elkaar snijden bevindt het landmark zich. Dit is weergegeven in figuur 3.13 (a). Vervolgens kan één van beide panoramabeelden vervangen worden door een die hetzelfde landmark bevat en opnieuw een snijpunt worden gezocht. Op deze manier krijg je voor hetzelfde landmark verschillende locaties, die vanwege onnauwkeurigheden iets van elkaar kunnen verschillen. (figuur 3.13 (b)) Wordt dit nu vervolgens ook gedaan voor de andere landmarks (figuur 3.13 (c)), dan kan op deze manier een soort representatie van de wereld gemaakt worden met de locaties van de landmarks

daarin. Van de locaties voor hetzelfde landmark wordt een gemiddelde genomen en vanuit dit landmark kan een lijn getrokken worden onder de hoek waar het landmark wordt gezien vanuit het nieuwe beeld. Er zijn twee van deze lijnen nodig om een snijpunt te bepalen, die de locatie van het nieuwe panoramabeeld, en dus de locatie van de robot, aangeeft. In figuur 3.13 (d) is dit weergegeven.



figuur 3.13
methode voor het berekenen van de locatie van het nieuwe beeld

Het bepalen van de locatie voor de robot wordt ook voor meerdere combinaties gedaan, op eenzelfde manier als bij de landmarks en ook hier zal een gemiddelde worden genomen als eindlocatie van de robot.

3.3 Conclusie

In dit hoofdstuk heb ik beschreven hoe het systeem werkt voor het bepalen van de locatie van de robot in het wereldmodel. Om nu te bekijken in hoeverre het systeem goed werkt kan gekeken worden naar de behaalde resultaten. Deze resultaten zullen in hoofdstuk 4 worden beschreven. Vervolgens zullen in hoofdstuk 5 mogelijke aanpassingen aan het systeem worden besproken.

Hoofdstuk 4

Resultaten

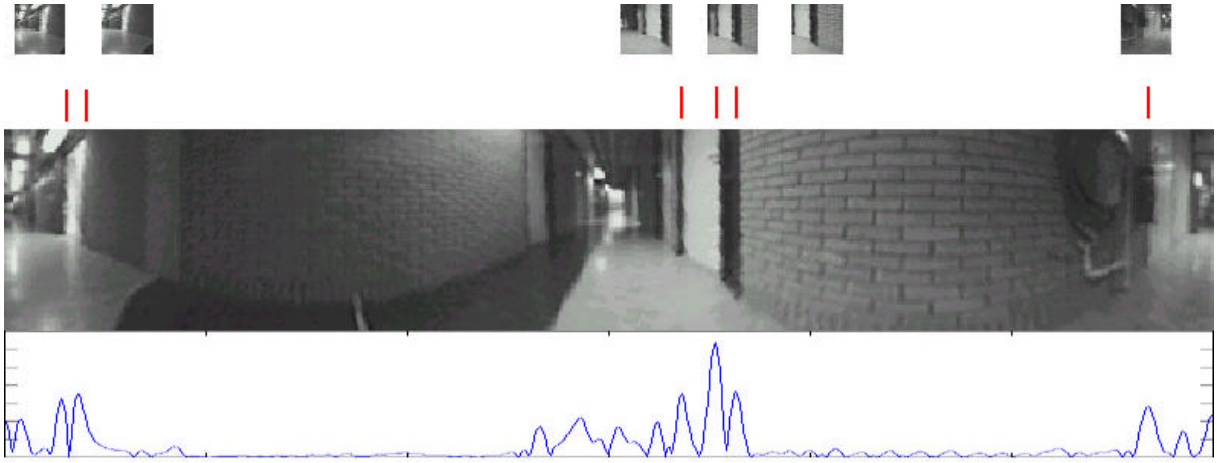
In dit hoofdstuk zal ik de resultaten van het onderzoek eerst beschrijven aan de hand van een specifiek voorbeeld, en vervolgens in het algemeen.

4.1 De resultaten voor een specifiek voorbeeld

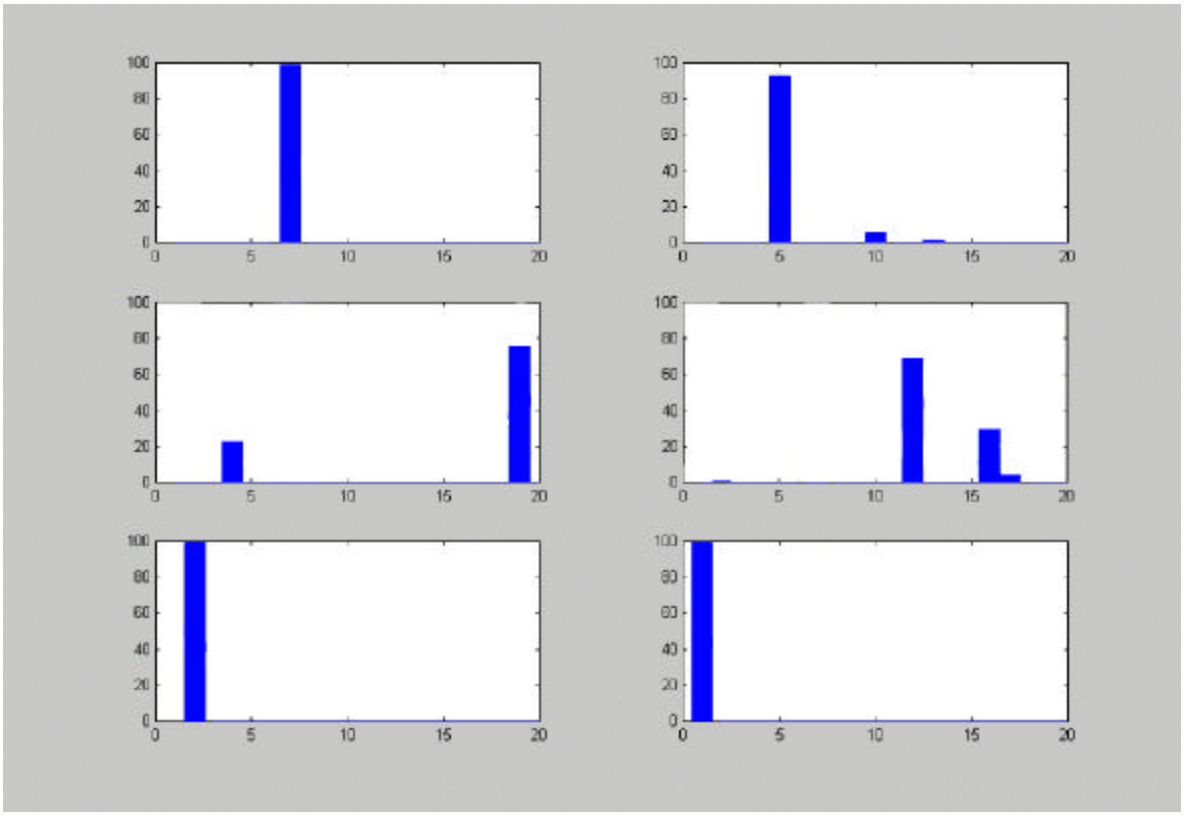
Als wereldmodel voor de robot is gebruik gemaakt van een gang van ongeveer 8 bij 2 meter, waarin 165 beelden zijn genomen. Afbeelding 4.1 geeft aan wat de locaties van deze panoramabeelden in de gang zijn. Van deze 165 beelden zijn 16 willekeurige beelden gebruikt als testset (is ongeveer 10 %) en de rest van de beelden is gebruikt voor de database (leerset). Van de beelden in de database zijn eerst de representaties gemaakt zoals beschreven in hoofdstuk 3.1.
Om nu een specifiek voorbeeld te nemen heb ik als testbeeld het beeld genomen dat in figuur 4.1 staat aangegeven met een rode stip.
De locatie hiervan is (69, 0). Deze waarden zijn aangegeven in centimeters. Het is nu de bedoeling om de locatie van dit panoramabeeld te bepalen en vervolgens te kijken hoever deze is verwijderd van zijn oorspronkelijke locatie van (69, 0).

figuur 4.1
locaties panoramabeelden

Nadat de landmarks zijn gevonden in het testbeeld (zie figuur 4.2) en deze zijn geprojecteerd op de principale vectoren, worden de afstanden berekend van elk landmark tot alle clustermiddens. Deze afstanden worden gebruikt om de kans te berekenen dat een landmark tot een clustermidden behoord. Hierbij ontstaat het plot in figuur 4.3 en worden de gelijkende clustermiddens voor elk landmark gevonden zoals te zien is in tabel 4.1.



figuur 4.2
testbeeld op locatie (69, 0) met bijbehorende afgeleide en landmarks

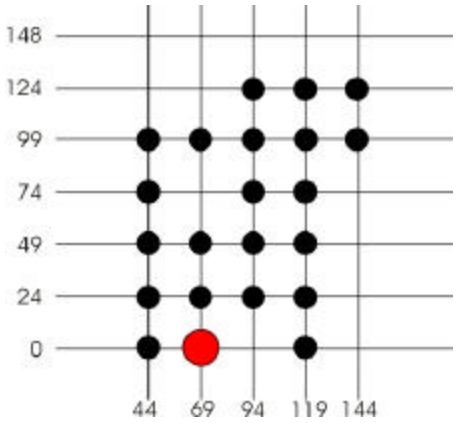


figuur 4.3 1t/m 6
de kans dat een landmark tot een clustermidden behoort
op de horizontale as staan de clustermiddens, op de verticale as de kans in percentage

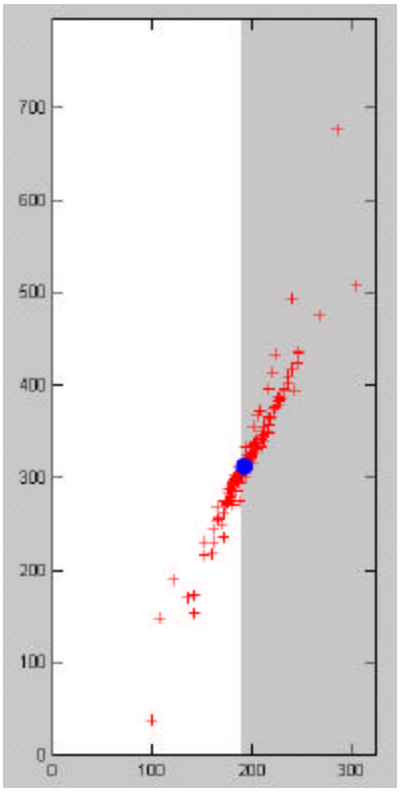
landmark	clustermidden
1	7
2	5
3	19
4	12 & 16
5	2
6	1

tabel 4.1
landmark 1 t/m 6 met de gelijkende clustermiddens

Nu worden voor het testbeeld alle mogelijke reeksen gemaakt, in dit geval zijn dat er 2. Deze worden nu vergeleken met de reeksen in de database waarbij de gelijkende beelden worden gevonden. In figuur 4.4 is te zien welke leerbeelden zijn gevonden, m.a.w de beelden die minstens 4 landmarks gemeenschappelijk hebben, waarbij deze ook nog eens in dezelfde volgorde terugkomen. Nu alle beelden bekend zijn, worden de locaties van de landmarks berekend. Dit berekenen van de landmarks gebeurt voor alle mogelijke combinaties, dus voor elk gevonden leerbeeld in combinatie met de andere gevonden leerbeelden. Hierdoor worden voor elk landmark meerdere locaties gevonden, zoals te zien is in figuur 4.5, waar met + alle locaties zijn aangegeven voor landmark 5. (sommige gevonden locaties vallen buiten het gebied van de gang, in het figuur grijs weergegeven)



figuur 4.4
teruggevonden leerbeelden
de rode stip stelt het testbeeld voor,
de zwarte stippen de gevonden leerbeelden



figuur 4.5
locaties voor landmark 5 aangegeven met + en gemiddelde locatie aangegeven met ·

Van al deze locaties wordt een gemiddelde genomen, die bewaard wordt als zijnde de locatie van het landmark.

Nu er voor alle landmarks een locatie is gevonden, wordt de locatie van het zoekbeeld berekend. Dit gebeurt ook weer voor alle mogelijke combinaties, waarna er een gemiddelde genomen wordt. De gevonden locatie voor het testbeeld op locatie (69, 0) is (72, 6). De afstand van de eigenlijke locatie naar de gevonden locatie is 7 cm.

4.2 De resultaten voor alle beelden in de testset

In tabel 4.2 zijn de resultaten voor de 16 panoramabeelden in de testset weergegeven.

Beeld		Resultaat	Oorspronkelijke locatie beeld	Verschil in cm
1		(41, 105)	(44, 124)	19
2		(72, 6)	(69, 0)	7
3		(71, 79)	(69, 74)	5
4		(88, -3)	(94, 0)	7
5		(93, 204)	(94, 198)	6
6		(130, 167)	(119, 148)	22
7		(114, 474)	(119, 322)	152
8		(45, 447)	(44, 449)	2
9		(70, 597)	(44, 598)	26
10		(77, 539)	(69, 548)	12
11		(85, 478)	(94, 499)	23
12		(107, 541)	(94, 548)	15
13		(172, 497)	(119, 499)	53
14		(90, 489)	(144, 499)	55
15		(166, 533)	(144, 548)	27
16		(147, 766)	(144, 772)	7

tabel 4.2
resultaten

Het gemiddelde verschil tussen de oorspronkelijke locaties van een beeld en het gevonden resultaat is 27 cm.

Zoals te zien is, is het resultaat niet voor alle beelden even goed. Om te bepalen waar dit aan ligt, worden de resultaten van testbeeld 7 in de volgende paragraaf uitvoerig bekeken.

4.3 Onderzoek naar het slechte resultaat van testbeeld 7

Allereerst moet worden gekeken naar de gevonden reeksen. Hier is al te zien, dat er een probleem optreedt bij landmark 1 en 5:

landmark	clustermidden
1	11
2	14
3	13
4	20
5	11
6	3

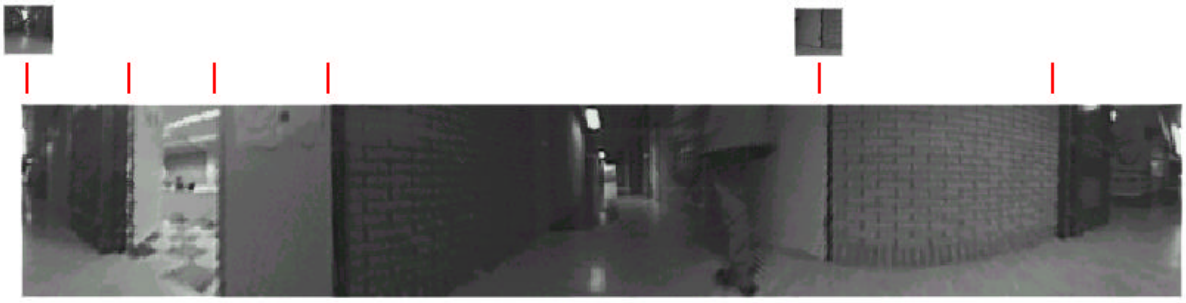
tabel 4.3
landmark 1 t/m 6 met de gelijkende clustermiddens

Hieronder zijn de bijbehorende beelden te zien:

landmark 1	landmark 5	clustermidden 11
		

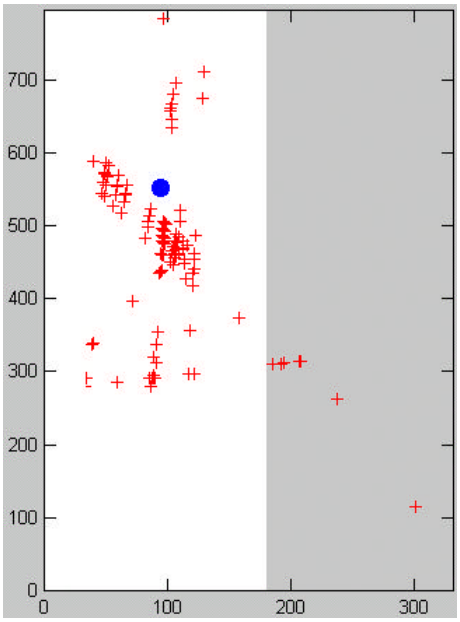
tabel 4.4
landmark 1 en 5 en bijbehorend clustermidden 11

Zowel landmark 1 als landmark 5 worden tot hetzelfde cluster gerekend. Hierdoor zullen er verschillende locaties in de omgeving met elkaar worden vergeleken. Dit is in figuur 4.6 te zien. De twee landmarks die tot hetzelfde cluster worden gerekend zijn ook weergegeven. Er wordt nu geprobeerd om een locatie te berekenen voor het landmark met clustermidden 11, en wanneer er twee leerbeelden met elkaar vergeleken worden die elk clustermidden 11 bezitten als landmark, kan het zijn dat ze beiden een verschillend landmark gebruiken.



figuur 4.6
beeld 7 met landmarks 1 en 5

De locaties die nu gevonden worden zijn weergegeven in figuur 4.7



figuur 4.7
gevonden locaties voor clustermidden 11 (landmarks 1 en 5)

Zoals is te zien, is er totaal geen duidelijke samenhang tussen alle gevonden locaties (weergegeven met +) en wordt voor zowel landmark 1 als 5 een locatie gevonden (weergegeven met •) die niet overeenkomt met de oorspronkelijke locaties van de twee landmarks.

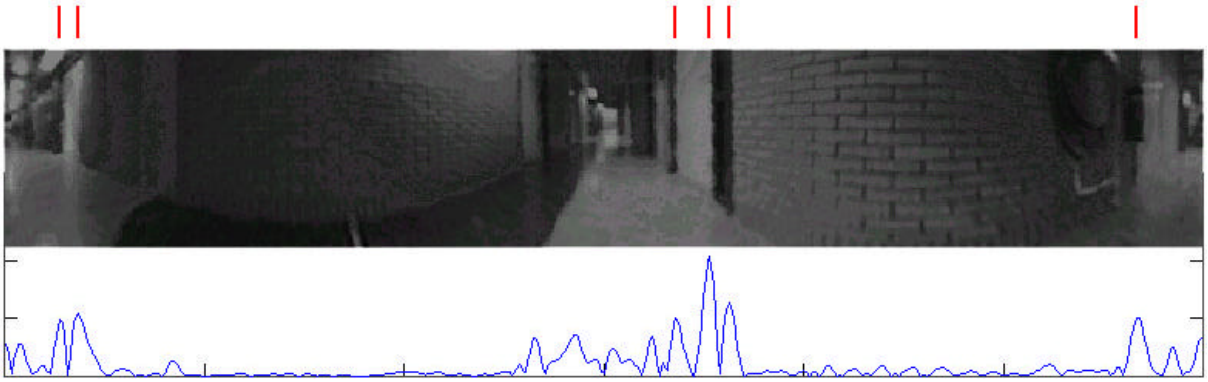
In dit voorbeeld gaat het fout, omdat verschillende landmarks in een beeld tot hetzelfde clustermidden worden gerekend. Hiervoor zal nog een oplossing moeten worden gevonden.

4.4 De werking van het systeem bij een andere lichtintensiteit

Bij het onderzoek heb ik de beschikking gehad over een set beelden, die allemaal met eenzelfde lichtintensiteit zijn genomen. Om nu te kijken of het systeem ook goed omgaat met beelden waarbij deze lichtintensiteit anders is, is beeld 2 uit figuur 4.8 a) donker gemaakt, waardoor het effect ontstaat alsof het beeld is genomen met een lagere lichtintensiteit. Het is nu van belang, dat bij het detecteren van de landmarks in het beeld dezelfde locaties worden gevonden en wanneer dit gebeurd, dat de gevonden landmarks dan tot hetzelfde cluster worden gerekend. Wanneer aan deze beide criteria wordt voldaan, zal het systeem hoogst waarschijnlijk werken onder verschillende lichtintensiteiten. (tot op zekere hoogte natuurlijk, wanneer het beeld te donker zal zijn, zullen er geen duidelijke intensiteits overgangen meer worden gevonden)

figuur 4.8
bij figuur b) is de licht intensiteit lager

De gevonden locaties van de landmarks in het beeld zijn weergegeven in figuur 4.9. Vergelijking met figuur 4.2 laat zien, dat de locaties van de landmarks hetzelfde zijn.



figuur 4.9
gevonden locaties van de landmarks

Echter, wanneer vervolgens zal worden gekeken tot welk cluster elk landmark wordt gerekend, worden er hier andere resultaten gevonden.

landmark	oude clustermidden	clustermidden
1	7	18
2	5	18
3	19	18
4	12 & 16	11
5	2	9
6	1	18

tabel 4.5
landmark 1 t/m 6 met de gelijkende clustermiddens voor oorspronkelijke en nieuwe situatie

De landmarks die gevonden worden in het nieuwe beeld zijn allemaal een stuk donkerder, dus wanneer de afstand van de landmarks tot elk clustermidden wordt berekend is te zien dat ze allemaal dichter bij de donkere clustermiddens komen te liggen. Nu het op dit punt fout gaat, zal de rest van het systeem ook niet meer goed functioneren.

Hieruit blijkt nu, dat het systeem niet goed kan omgaan met verschillen in lichtintensiteit, vooral niet bij heel grote verschillen, zoals in dit voorbeeld. Dit probleem is misschien met kleine aanpassingen op te lossen. Hier denk ik dan aan intensiteits onafhankelijke representaties van de landmarks, of de intensiteit van de landmarks aanpassen nadat ze in het testbeeld zijn gevonden. Hier zal nog onderzoek naar moeten worden gedaan.

Hoofdstuk 5

Conclusie

Van te voren wil ik vaststellen, dat er enige kanttekeningen moeten worden geplaatst bij de behaalde resultaten in het onderzoek. Bij het testen van het systeem, heb ik alleen beschikking gehad over een set van 165 panoramabeelden, gemaakt in een gang van acht bij twee meter. Helaas heb ik het systeem dus niet kunnen testen met een set beelden, genomen in een grotere omgeving. In de set beelden miste ook die van een omgeving in een kamer, waar de soorten landmarks er anders uit zien dan die in een gang. Hier zijn namelijk veel meer kleinere objecten te vinden.

Zoals eerder beschreven is de gemiddelde afstand van de berekende locatie van de robot en de eigenlijke locatie 27 centimeter. Dit gemiddelde wordt vooral door enkele uitschieters bepaald, namelijk twee gevallen van ruim 50 centimeter en één van 152 centimeter. Om het gemiddelde dus te verlagen, is gekeken waar het bij deze uitschieters fout ging. Het probleem, dat twee landmarks in hetzelfde beeld tot eenzelfde cluster worden gerekend, is één van de grootste oorzaken voor een verkeerde locatiebepaling. Hier zal dus nog een oplossing voor moeten worden gevonden.
Andere punten die kunnen bijdragen aan een betere locatiebepaling, zijn het instellen van een aantal vrije parameters. Het systeem bevat namelijk een aantal van deze parameters, die mogelijk nog niet optimaal zijn gekozen. Denk hierbij o.a aan de volgenden:

- *De grootte van de Gaussiche kernel die is gebruikt voor het bepalen van de afgeleidde.* Uit figuur 3.4 en 3.7 is namelijk te zien, dat verscheidene maxima dicht bij elkaar liggen, waardoor de gevormde landmarks sterk op elkaar lijken. Bij het clusteren van deze landmarks kunnen de twee dus tot een zelfde cluster worden ingedeeld, hoewel ze verschillende locaties hebben in het beeld. Zal de standaard deviatie van de gebruikt gaussian worden vergroot, dan zullen de twee gevonden maxima samensmelten tot een, waardoor er dus ook maar één landmark uit zal ontstaan. Door veel beelden te bekijken, zal dus een optimale standaard deviatie moeten worden gevonden.
- *Het aantal landmarks per panoramabeeld*. Wanneer er meer landmarks per beeld worden gebruikt, zal het systeem beter bestand zijn tegen object occlusie en verplaatsing. Door middel van veel testen zal ook dit aantal beter kunnen worden bepaald.

- *De parameters die zijn gebruikt bij het bepalen van de reeksen en gelijkenis tussen beelden*. De a , die gebruikt is in paragraaf 3.2.1 kan, zoals daar vermeld, beter worden bepaald. Ook kan worden onderzocht wat het optimale aantal verschillende landmarks tussen twee beelden is, om deze toch als gelijkend te beschouwen. Dit hangt natuurlijk weer nauw samen met het eerder genoemde aantal landmarks per beeld.

Een tweede belangrijk punt waar nog verdere aanpassingen nodig zijn, is dat er in het systeem nog geen rekening is gehouden met de lichtintensiteit, zoals gebleken is uit paragraaf 4.4.

Wanneer er nog onderzoek zal worden gedaan naar de verschillende hierboven genoemde punten, zal het systeem naar mijns inzien nog betere resultaten opleveren.

Bibliografie

[1] P. Gaussier, C. Joulain, J.P. Banquet, S.Leprêtre, A.Revel, The visual homing problem : An example of robotics/biology cross vertilization, Robotics and Autonomous Systems 30, 2000

[2] R. Sim, Mobile robot localization using learned landmarks, TR-CIM-98-3, 12 november 1998

[3] A.W.M Smeulders, R. van den Boomgaard, An introduction to image processing and computer vision (syllabus), Universiteit van Amsterdam, augustus 1996

[4] Handbook of computer science and engineering, Hoofdstuk 6: Pattern matching and text compression algorithms

[5] A, K. Jain, R. C. Dubes, Algorithms for clustering data, Prentice Hall, 1988

[6] B.D. Ripley, Pattern recognition and neural networks, Cambridge University Press, 1996

[7] L.A. Cornelissen, Automatisch vinden en volgen van stabiele landmarks in kleurenbeeldreeksen van een robotomgeving (scriptie), Universiteit van Amsterdam, juni 1998

[8] Lineaire algebra voor Bis en AI (syllabus), Universiteit van Amsterdam

[9] B. Zitová, J. Flusser, Landmark recognition using invariant features, Pattern recognition letters 20, 541-547, 1999

[10] B. Triggs, Model-based sonar localisation for mobile robots, Proceedings of the international workshop on intelligent robotic systems, 1993

[11] B.J.A Kröse, N. Vlassis, R. Bunschoten, Y. Motomura, A probabilistic model for appearance-based robot localization, november 2000