Het automatisch volgen van objecten met een Pan-Tilt-Zoom camera

Plaats, datum: Zoetermeer, 11 januari 2016

Opgesteld door: **Gert den Neijsel** | 93005101 | G.C.H.denNeijsel@student.hhs.nl

In opdracht van: **Julien Vijverberg, Siqura / TKH Group**

Afstudeerverslag van de opleiding Technische Informatica, Haagse Hogeschool

Documentrevisies

Datum	**Versie**	**Aanpassing**
23-09-2015	0.1	Initiële opzet
14-10-2015	0.45	45% voortgangsrapport
27-10-2015	0.6	Eerste versie naar begeleidend examinator
02-12-2015	0.8	Versie t.b.v. tussentijds assessment
05-01-2016	0.96	Versie ter review naar opdrachtgever
11-01-2016	1.0	Definitieve versie t.b.v. assessment

Distributielijst

Versie	**Bedrijfsmentor J. Vijverberg**	**Begeleider P. Burghouwt**	**2e examinator T. Andrioli**	**Gecommitteerde en archief**
0.1	X			
0.45	X	X		
0.6	X	X		
0.8		X	X	
0.96	X			
1.0	X	X	X	X

Referaat

Gert den Neijsel, “Het automatisch volgen van objecten met een Pan-Tilt-Zoom camera’’. Afstudeerverslag opleiding Technische Informatica, Haagse Hogeschool, 2015.

Descriptoren
camera surveillance
computer vision
image processing
visual object tracking
algorithms
appearance model
feature points
OpenCV
PTZ control

Samenvatting

Deze opdracht komt voort uit de wens om boorplatformen te bewaken en naderende vaartuigen te detecteren. Dat gebeurt in eerste instantie met vaste camera’s.

Als deze beweging waarnemen, dan moet een Pan-Tilt-Zoom (PTZ) camera gericht en ingezoomd worden op de positie waar een object wordt gedetecteerd waarna het object gevolgd moet worden.

Dit project is een onderzoeksproject. De twee hoofddoelen zijn:

- Onderzoek naar het meest geschikte tracking algoritme voor deze toepassing.
    - Het geselecteerde tracking algoritme is DSST .
- Onderzoek naar de meest geschikt regelaar voor het aansturen van een PTZ camera.
    - De regelaar die gekozen is, is een ‘rule based’ regelaar.

Het project had niet als doel om software te ontwikkelen, de vraag was om een demo te tonen. Het onderzoek is succesvol afgesloten met een live demo.

Voorwoord

Na een drietal mislukte pogingen om binnen mijn werk aan een afstudeeropdracht te komen (en een paar buiten mijn werk), liep ik tegen deze afstudeeropdracht bij Siqura aan. De combinatie bleek ideaal: Siqura bood mij de mogelijkheid om in deeltijd af te studeren, het is vlak bij mijn huidige werk en het kennisgebied van deze opdracht is mijn absolute favoriet.

Ik ben al jaren geïnteresseerd in- en bezig met beeldverwerking en beeldbewerking.
Een logische volgende stap was het halen van informatie uit beelden. Dit vakgebied bleek computer vision te heten. Ik wil hier in de toekomst meer van leren.

Ik heb een minor Embedded Computer Vision design gedaan bij de HAN (Hogeschool van Arnhem en Nijmegen). Sinds die tijd heb ik een aantal studieprojecten met software ontwikkeling gedaan op het gebied van Computer Vision en embedded systems. Voorbeelden hiervan zijn: Analoge kWh meter aflezen met behulp van zelfgeschreven algoritmes, fish-eye lens correctie met zelfgeschreven algoritme, personen tellen met Vision (onderzoek naar Background Subtraction algoritmes).

De kennis die ik opgedaan heb bij de Haagse Hogeschool is basiskennis van ANSI C, C# en beperkte wiskundekennis. Tijdens de minor van de HAN heb ik basiskennis van Computer Vision opgedaan en kennis gemaakt met C++ en met OpenCV. Tijdens dit project heb ik mijn kennis op het gebied van Linux, C++ en OpenCV en Computer Vision aanzienlijk uitgebreid.

Ik wil mijn dank uitspreken aan Kees Veraart, mijn manager bij KPN voor het beschikbaar stellen van de benodigde tijd en gelegenheid om te kunnen afstuderen bij een extern bedrijf.

Tevens wil ik Julien Vijverberg, mijn afstudeerbegeleider bij Siqura bedanken voor zijn hulp en geduld en voor het beantwoorden van alle vragen.

Gert den Neijsel, Zoetermeer, 11 januari 2016.

Inhoud

1 Achtergronden ... 1

1.1 Het bedrijf ... 1
1.2 Aanleiding ... 1

2 Het projectresultaat ... 2

2.1 Probleemstelling ... 3
2.2 Doelstelling ... 3
2.3 Projectaanpak en fasering ... 4

3 Theoretisch kader ... 6

3.1 Definitie van ‘visual tracking’ ... 6
3.2 Opbouw van ‘visual trackers’ ... 7
3.3 Problemen bij tracking ... 8
3.4 Eisen aan een tracker voor gebruik op zee ... 9

4 Onderzoek Vision componenten ... 10

4.1 Testen van trackers ... 10
4.2 Selectie van de beste tracker ... 12
4.3 Signatuur opbouwen ... 15
4.4 Detecteren van objecten ... 17

5 Onderzoek naar PTZ aansturing ... 22

5.1 Methoden voor PTZ motor aansturing ... 22
5.2 Regelaar voor de PTZ motor aansturing ... 24
5.3 Normaliseren van PTZ snelheidswaarden ... 27
5.4 Werking van de PTZ regelaar ... 30

6 Instellen van camerastreams ... 33

7 Conclusies en aanbevelingen ... 37

8 Evaluatie ... 39

8.1 Productevaluatie ... 39
8.2 Procesevaluatie ... 39
8.3 Evaluatie beroepstaken ... 44

Literatuurlijst ... 45

Verklarende woorden en afkortingen ... 47

Bijlage A – Featurepoint detectors in OpenCV 3.0.0 ... 49

Bijlage B – Toelichting trackers VOT2014 ... 50

Bijlage C – Voorbeeld van ONVIF commando ... 51

Bijlage D – PTZ CGI besturing en cameraspecificaties ... 52

Bijlage E - UML ontwerp. ... 55

Bijlage F – Flowchart van de Lucas Kanade detector ... 57

Bijlage F – Metingen aan video streams ... 58

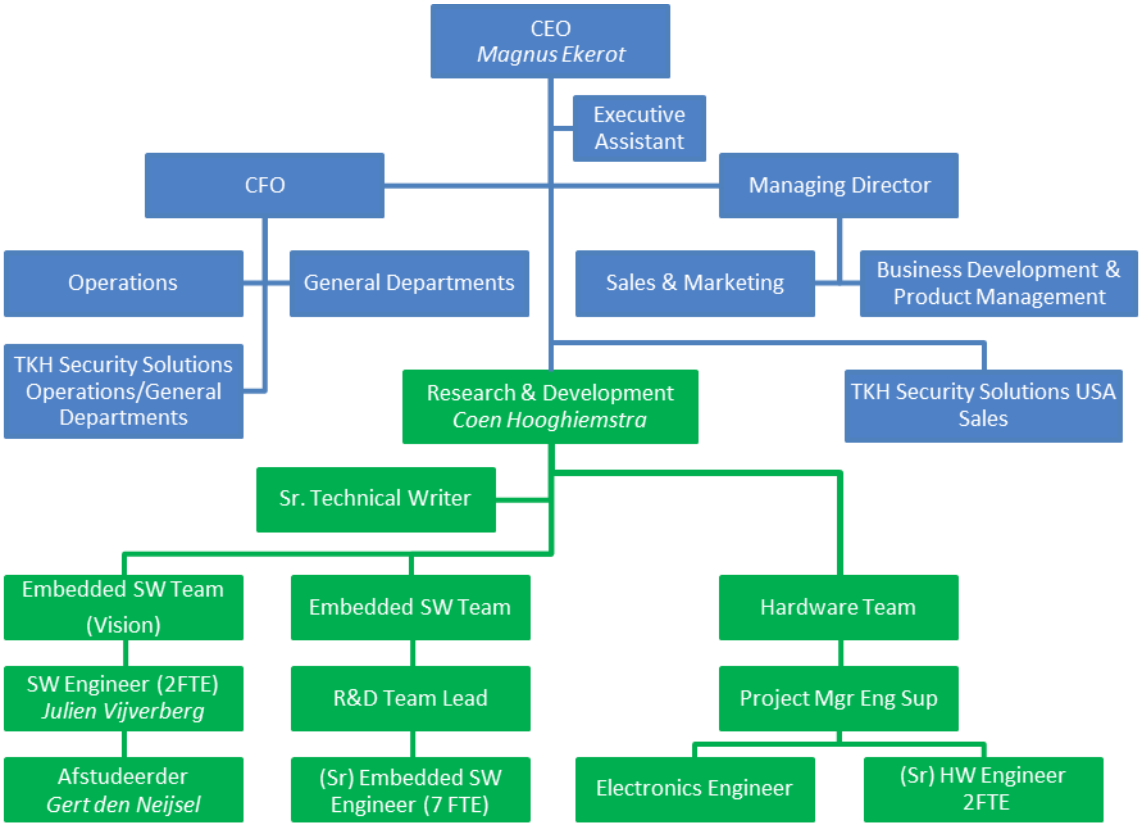
1 Achtergronden

Dit hoofdstuk geeft een overzicht van de omgeving waarin het project uitgevoerd wordt en de reden waarom.

1.1 Het bedrijf

Siqura BV is gevestigd in Gouda en biedt geavanceerde video surveillance oplossingen. Deze oplossingen omvatten IP-camera's, video-encoders, netwerk videorecorders, fiber apparatuur, video management en video content analyse (VCA) software. Siqura levert en ontwikkelt video-analyse algoritmen voor surveillance-doeleinden. Hierbij is te denken aan algoritmes voor inbraakdetectie, detectie van achtergebleven bagage en voor het tellen van mensen.

Siqura is deel van TKH Security Solutions (zie blauwe deel in Figuur 1), onderdeel van de aan de AEX genoteerde TKH Group. TKH komt van ‘Twentse Kabel Holding’, voorheen ‘Twentse Kabel Fabriek’ (TKF). Bij de gehele holding werken wereldwijd circa 3700 mensen. Bij Siqura in Gouda werken circa 55 mensen. Deze opdracht vindt plaats bij de R&D afdeling te Gouda (zie groene/onderste deel in Figuur 1).



Figuur 1 Organogram Siqura / TKH Security Solutions

1.2 Aanleiding

Siqura BV werkt aan projecten voor de automatische detectie van objecten op grote afstand en detectie van schepen bij statische offshore constructies (boorplatformen en dergelijke). Siqura draagt bij aan een internationaal project “Advancing Plug & Play Smart Surveillance” (APPS, 2015). Dit project maakt daar deel van uit.

Met dit onderwerp hebben zich binnen Siqura al meerdere stagiaires en afstudeerders beziggehouden binnen verschillende vakgebieden zoals elektronica of software ontwikkeling. Voorbeelden hiervan zijn een project waarbij Background Subtraction gedaan is met een DSP,

een ander voorbeeld is gebruik van PTZ camera’s in combinatie met thermische camera’s (PTZ staat voor ‘Pan-Tilt-Zoom’, oftewel richten door draaien, kantelen en zoomen).

Dit project is het vervolg op een project van Jeffrey van Pernis (Haagse Hogeschool vestiging Delft) welke tot doel had om een PTZ camera te kunnen aansturen vanuit locatie- en beeldinformatie verkregen uit een vast opgestelde camera.

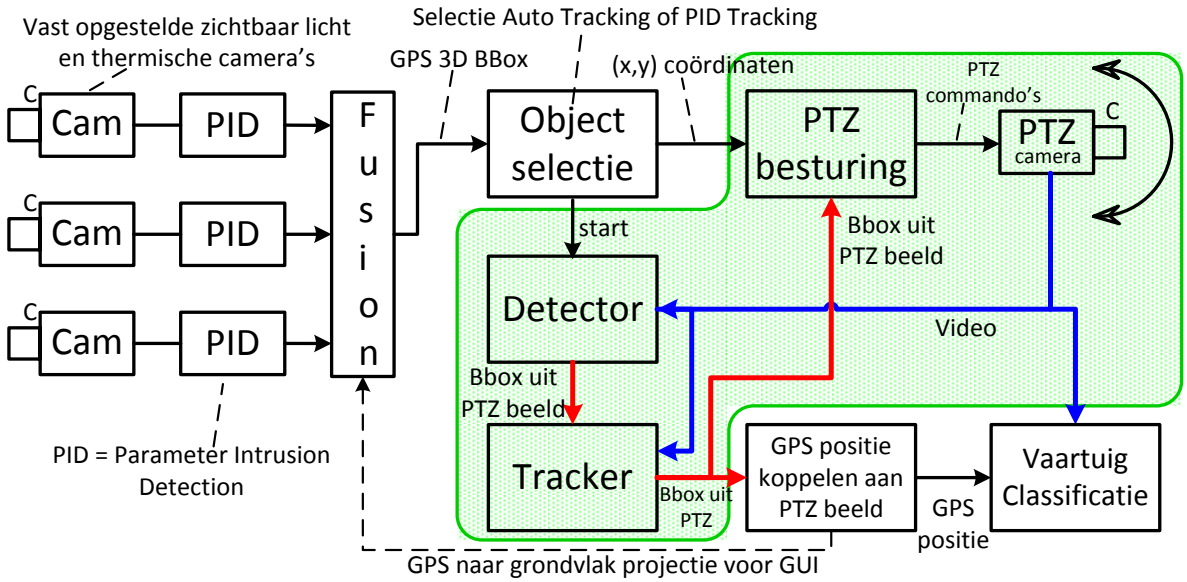
Figuur 2 toont een overzicht van de componenten welke deel uitmaken van het APPS project. De blokjes met groene achtergrond zijn de componenten uit dit project. Het startpunt voor ‘PTZ Auto Tracking’ (het groene kader) is een (x, y) coördinaat afkomstig uit een vaste camera en daarbij een startsignaal vanuit ‘Object selectie’. De PTZ camera wordt op dit coördinaat gericht en de detector wordt gestart. Zodra de detector een object heeft gedetecteerd wordt de tracker gestart. De tracker volgt het object met de PTZ camera.

In het blokje ‘Object selectie’ wordt een handmatige keuze door een operator gedaan.

- De keuze ‘PID tracking’ zorgt ervoor dat de detectie en PTZ besturing rechtstreeks door ‘PID’ gedaan wordt.
- De keuze ‘Auto Tracking’ zorgt ervoor dat ‘PTZ Auto Tracking’ (het groene kader) de detectie en PTZ besturing doet.

De vaste camera’s zijn van verschillende typen. Het zijn zowel camera’s voor zichtbaar licht als warmtebeeld Long-wave Infrared (LWIR) camera’s. Warmtebeeldcamera’s geven een duidelijk contrast van schepen ten opzichte van de omgeving, ongeacht of het dag of nacht is. Daardoor zijn ze het meest geschikt om “Parameter Intrusion Detection” (gebiedsbewaking) te starten.

Meer uitleg over het groene deel uit het blokschema is te vinden in hoofdstuk 5.4. Een eenvoudig ontwerp in UML met de omschrijving van de werking is te vinden in “Bijlage E”.



Figuur 2 Blokschema van het APPS project. De componenten van dit project staan in het groene kader

2 Het projectresultaat

Dit hoofdstuk legt uit waarom dit project gedaan wordt en wat aan het eind van het project opgeleverd gaat worden.

2.1 Probleemstelling

In de aanleiding (hoofdstuk 1.2) is aangegeven dat dit project betrekking heeft op het bewaken van boorplatformen op zee. Het doel is dit zo te automatiseren dat er voor monitoring geen personeel op een platform nodig is. Bewaking wordt op afstand vanuit een meldkamer gedaan waarbij het camerasysteem zo accuraat mogelijk weergeeft en met zo min mogelijk valse meldingen aangeeft of er vaartuigen naderen.

Om het gebied rond een platform te bewaken zijn meerdere camera’s nodig. Hiervoor wordt gebruik gemaakt van een aantal vast opgestelde camera’s met vast blikveld. Deze hebben het nadeel dat een object niet meer gevolgd kan worden als het object het gezichtsveld van de camera verlaat. Daarom zijn er naast vaste camera’s ook PTZ camera’s in gebruik waarmee objecten gevolgd kunnen worden.

Om naderende objecten te kunnen volgen, moet een Pan-Tilt-Zoom (PTZ) camera gericht en ingezoomd worden op de positie waar een object wordt gedetecteerd door een vaste camera.

2.2 Doelstelling

Het belangrijkste deel van deze opdracht is het selecteren van het meest geschikte ‘Object Tracking’ algoritme voor deze toepassing; het onderscheiden (detecteren) en volgen van objecten in een maritieme omgeving, oftewel op zee. Het praktische doel van dit project is dat er een werkende demo getoond wordt waarmee vaartuigen automatisch met een PTZ camera gevolgd worden.

De te bouwen demosoftware bestaat uit meerdere onderdelen. Om deze te realiseren is onderzoek uitgevoerd naar de diverse componenten. Hiertoe zijn een hoofdvraag en een aantal deelvragen opgesteld.

De hoofdvraag voor dit onderzoek is:

Wat is de beste configuratie voor PTZ auto-tracking voor het volgen van objecten op zee?

De hoofdvraag is te beantwoorden door de onderstaande deelvragen uit te werken.

Deelvragen:

1. Welke tracker is het meest geschikt voor het detecteren en volgen van vaartuigen op zee?
    a. Hoe kunnen trackers getest worden?
    b. Welke tracker stelt de minste eisen aan hardware qua rekenkracht, geheugen, opslag?
2. Hoe kun je een signatuur (‘Appearance Model’) van een object opbouwen en bijhouden?
    a. Hoe kan een object initieel gedetecteerd worden?
    b. Hoe ga je om met de afwijking tussen de locatie van een object uit de vaste camera en de locatie van dat object in het PTZ beeld?
3. Wat is de meest geschikte regelaar om de PTZ camera in de pas te laten lopen met de tracker?
    a. Is een lage orde PID regelaar voldoende of is een hogere orde regelaar, of zelfs een niet lineaire regelaar nodig? (PID staat voor “Proportioneel, Integrerend, Differentiërend” en komt uit de regeltechniek)
    b. Hoe kunnen we het optimale zoomniveau bepalen zodat de tracker het object niet kwijtraakt?
    c. Hoe kunnen we na kwijtraken van het object dit weer terugvinden?

Optionele deelvragen (afhankelijk van de beschikbare tijd):

4. Hoe om te gaan met occlusies (overlappingen in het beeld)?

5. (Hoe) Kunnen beelden van vaste camera en PTZ camera samen gebruikt worden om tracking te verbeteren (Fusion)?

2.3 Projectaanpak en fasering

Dit project bevatte veel onbekende factoren zodat vooraf niet te overzien was hoe een ontwerp er uit moest gaan zien. Eerst moest de werking van de diverse componenten onderzocht worden. Om die reden is een onderzoeks-aanpak gehanteerd en niet een ontwerpmethodiek zoals RUP of ASI. Een ontwerpmethodiek ligt meer voor de hand als de werking en samenhang van de componenten vooraf voor een groot deel inzichtelijk is.

Vanuit het onderzoek is software ontwikkeld met als doel om een demo te tonen bestaande uit software op een PC met werkende tracking en PTZ aansturing. De bij Siqura gebruikelijke werkwijze is aangehouden. Er is een iteratieve aanpak gehanteerd en een fasering waarbij het mogelijk was om fasen apart aan te pakken en zo nodig een stap terug te gaan. Zie voor de oorspronkelijke planning het “Plan van Aanpak” (externe bijlage).

Bij Siqura wordt de Kanban methode gebruikt, waarbij activiteiten afgerond worden voordat er aan iets nieuws begonnen wordt en waarbij niet getracht wordt vooraf alles te plannen en voor te bereiden. De methode heeft grote overeenkomsten met Scrum met onder andere het ordenen van activiteiten met Post-It’s op een whiteboard en wekelijkse stand-ups.

Het onderzoek is oorspronkelijk opgebouwd rond de hoofd en deelvragen. Naarmate het onderzoek vorderde bleek dat de fasering uit 4 logische onderdelen bestaat. De deelvragen komen hier niet helemaal mee overeen, er is geen deelvraag bij de fase ‘PTZ aansturing’.

De complete doorlooptijd voor het project was 14 weken. Hieronder staat een overzicht van de fasering en planning. Zie voor een detailleerde planning het “Plan van Aanpak”.

Fasering:

- ***Onderzoek naar (feature point) tracking algoritmes (5 weken).***
  Activiteiten: literatuuronderzoek en experimenten, onder andere het testen van trackers en testen en maken van demo-code.
- ***Onderzoek naar detectiemechanismen (3 weken).***
  Activiteiten: literatuuronderzoek, experimenten en ontwikkelen van demo-code.
- ***Onderzoek naar methoden van aansturing van PTZ camera’s (2 weken).***
  Activiteiten: literatuuronderzoek, experimenten en ontwikkelen van demo-code.
- ***Onderzoek naar ‘regelaars’ voor de aansturing van PTZ camera’s (2 weken).***
  Activiteiten: literatuuronderzoek, experimenten en ontwikkelen van demo-code.
- ***Optionele uitbreidingen / reservetijd (2 weken).***

De resultaten van elke fase zijn ‘vastgeplakt’ aan de voorgaande fase om zo tot een werkend geheel te komen.

Risico’s:

Risico	Impact	Kans	Kans verlagende maatregel	Impact verlagende maatregel
Verlies van projectdocumentatie	Hoog	Klein	Documenten plaatsen op server van Siqura. Lokale versies van documenten op laptop synchroniseren met Dropbox.	Meerdere versies bijhouden en opslaan.
Verlies van programmacode	Hoog	Klein	Regelmatig code uploaden naar versiebeheersysteem	Meerdere versies bijhouden en opslaan.
Automatische initialisatie blijkt te lastig	Middel	Middel	In vroeg stadium gaan testen met opnames van werkelijke situatie.	Werken met betrouwbare vooraf opgenomen beelden.
Geometrische kalibratie van PTZ of statisch camera is te onnauwkeurig	Laag	Middel	Zo vroeg als mogelijk is de samenwerking tussen vaste- en PTZ camera testen (met vaste opstelling bij Siqura)	Werken met cameraopstelling in gecontroleerde omgeving

Omgeving

Bij Siqura wordt software met Qt in C++ geschreven voor gebruik onder Linux (GCC).
Als basis hiervoor wordt gebruik gemaakt van PC’s met een 32-bits Linux versie (standaard: Ubuntu 14.04 LTS). Daarnaast wordt gebruik gemaakt van de eigen software libraries van Siqura waar OpenCV 3.0.0 deel van uitmaakt.

Bij dit onderzoek zijn de volgende PC’s gebruikt:

- Dell Latitude D830, dual core 2,2GHz Core2Duo CPU, 4GB geheugen, 120GB SSD
- Dell Latitude E6530, quad core 2,67GHz i5 CPU, 4GB geheugen, 120GB SSD
- Merkloze PC met quad core 2,67GHz i5 CPU, 4GB geheugen, 300GB harddisk en ATIRV370 videokaart

LET OP: Een applicatie die niet specifiek ontwikkeld is voor het gebruik van meerdere CPU cores maakt slechts gebruik van één enkele CPU core. Dit geldt ook voor de demo.
Een CPU met één core met een hogere kloksnelheid is hier het meest geschikt.

De volgende PTZ camera’s zijn gebruikt (zie “Bijlage D” voor de specificaties):

- Pelco Spectra III (draagbare demoset) in combinatie met een Siqura EVE-1
- Siqura HSD820H2 (camera op Siqura parkeerterrrein)

Het initiële onderzoek is gedaan met de Pelco en daarna herhaald met HSD820H2 camera.

3 Theoretisch kader

In dit hoofdstuk wordt aan de hand van literatuuronderzoek aangegeven hoe tracking algoritmes werken.

Voordat dit onderzoek gestart is, heeft de opdrachtgever tijdens de interviews een aantal suggesties gedaan: “kijk eens naar de VOT challenge” en “kijk eens naar Feature Point trackers”. Het onderzoek is begonnen door naar de documenten van de VOT Challenge te bekijken (zie hoofdstuk 4.1 voor meer informatie).

Van daaruit zijn de ‘papers’ van de geteste trackers bekeken. Omdat deze niet te lezen zijn voor iemand die niet deskundig is in dit vakgebied is het onderzoek breder getrokken.
Er is gezocht naar algemeen beschrijvende en vergelijkende onderzoeken.

De in dit hoofdstuk vermelde documenten zijn gepubliceerde onderzoeken gevonden via de ‘IEEE Xplore’ zoekmachine (IEEE, 2015).

3.1 Definitie van ‘visual tracking’

Tracking is de analyse van een reeks videobeelden met het doel om een object in het beeld te volgen en de locatie van het object in het beeld (de scene) te bepalen. Tracking gebeurt met een tracking algoritme, hierna ‘tracker’ genoemd.

Tracking wordt onder andere gebruikt voor automatische surveillance, monitoren van verkeer, automatisch besturen van voertuigen, volgen van groepen dieren of bij sportactiviteiten.

Er zijn verschillende typen trackers. Er zijn trackers die een enkel object volgen, maar ook die meerdere objecten volgen. Over het algemeen wordt het type tracker geclassificeerd aan de hand van de componenten waaruit de tracker bestaat, zie hoofdstuk 0.

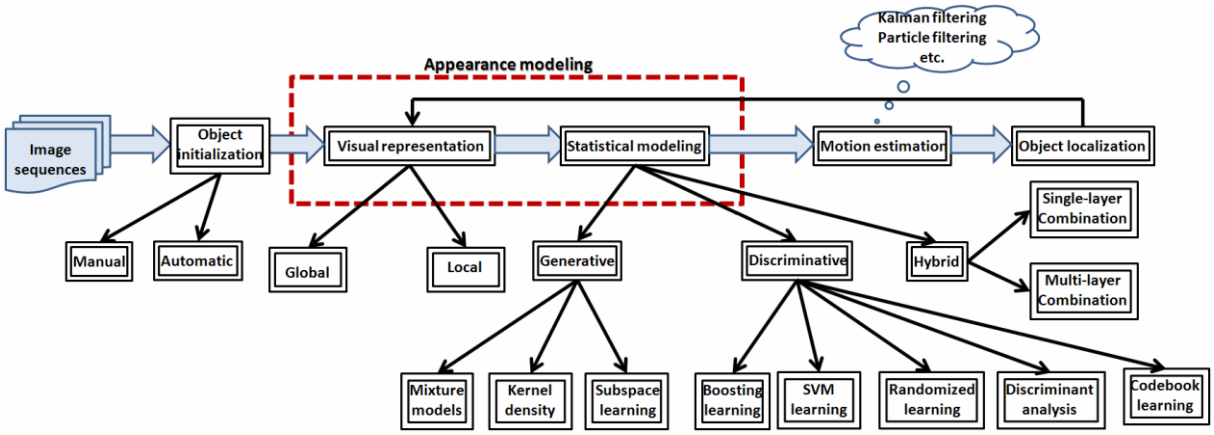
Dit onderzoek beperkt zich tot ‘short-term single-target model-free visual trackers’ (Čehovin, et al., 2015, p. 1). Het ‘single-target’ betekent dat ze een enkel object volgen. Het ‘short-term’ betekent dat het gevolgde object nooit geheel buiten beeld verdwijnt, er is dus geen her-detectie nodig. Het ‘model-free’ houdt in dat er vooraf geen model “geleerd” wordt. Het enige dat opgegeven wordt is een beginlocatie van het te volgen object.

De keuze voor de ‘single-target’ tracker is bij de start van het project door de opdrachtgever gegeven.

Het object wordt in het eerste beeld van de reeks gemarkeerd met een kader (bounding box). Daarna dient de tracker de locatie van het object zo correct mogelijk aan te geven. Als de tracker het object kwijtraakt dan is er niet voorzien in een mechanisme dat het object opnieuw detecteert.

Dit type trackers is niet zelf in staat de initiële locatie van een object te bepalen. Daarvoor is object detectie of zelfs object classificatie nodig. De objectdetectie wordt besproken in …
Object classificatie is een aparte tak binnen computer vision en valt buiten dit onderzoek.

3.2 Opbouw van ‘visual trackers’



Figuur 3 Model van trackers volgens (Li, et al., 2013)

Over het algemeen bestaat een tracker uit vier componenten, zie Figuur 3:

1. Object initialisatie
2. Model of signatuur opbouwen (‘appearance modelling’)
3. Bewegingsrichting inschatten (‘motion estimation’)
4. Locatiebepaling (‘object localisation’).

1. Object initialisatie

Het doel hiervan is aangeven waar een object zich bevindt zodat het gevolgd kan worden. Dit kan handmatig of automatisch. Bij handmatige initialisatie wordt met de muis een kader (bounding box) geselecteerd. Voor automatische initialisatie is objectdetectie nodig. Eenvoudige vormen van detectie zijn het detecteren van een bepaalde kleur of helderheid (afwijkend van de achtergrond) of het waarnemen van beweging door het verschil tussen twee plaatjes te nemen.

Een verdergaande mogelijkheid voor objectdetectie is object classificatie welke objecten aan de hand van kenmerken werkelijk herkent en classificeert.

2. Signatuur opbouwen (‘appearance model’)

Dit bestaat vaak uit twee onderdelen: een ***visuele representatie*** en een ***statistisch model*** opbouwen. Visuele representatie richt zich op het bouwen van een model aan de hand van robuuste descriptoren gebaseerd op verschillende visuele eigenschappen of bewegingsvectoren.

Statistisch modelleren heeft als doel een effectief wiskundig model te bouwen. Doordat het model telkens bijgewerkt wordt is te voorspellen hoe het er in het volgende plaatje uitziet.

3. Bewegingsrichting inschatten (‘motion estimation’)

Dit dient om te voorspellen in welke richting een object in het beeld zich begeeft. Dit gebeurt door richtingsvectoren weer te geven.

4. Object locatiebepaling

Dit gebeurt door patroonvergelijking (matching) in combinatie met inschatting op basis van bewegingsvectoren.

Detaillering binnen ‘appearance modelling’

Figuur 3 geeft een globaal overzicht, maar er ontbreekt veel detail. Het gaat te ver om hier in dit rapport uitgebreid op in te gaan. Er zijn tientallen zo niet honderden verschillende trackers die op verschillende manieren werken en er komen er jaarlijks tientallen bij. De evolutie binnen visual tracking is nog steeds in volle gang.

Voor meer uitleg over visual tracking wordt verwezen naar de volgende vergelijkings onderzoeken: (Yilmaz, et al., 2006), (Yang, et al., 2011), (Li, et al., 2013) en (Smeulders, et al., 2014).

Voor iets meer detaillering worden hieronder lijsten van de verschillende opties weergegeven:

‘Visual representation’ in Figuur 3 is te splitsen in twee groepen.

Onder **‘*Global* visual representation’** vallen:
- Raw pixel representatie (eenvoudige onbewerkte pixel weergave).
- Optical flow representatie (bewegingsvectoren).
- Histogram representatie (geeft verdeling van helderheidswaarden weer).
- Covariance representatie (uitdrukken in schaalgrootte).
- Wavelet filtering-based representatie (weergave van richting en schaal).
- Active contour representatie (omtrek van een object).

Onder **‘*Local feature-based* visual representation’** vallen:
- Local template based (opdelen in stukjes).
- Segmentation-based (superpixel, regio’s markeren).
- Key point based: SIFT, MSER, SURF, corner feature-based.
- Feature pool-based (onder andere ‘histogram of oriented gradients’; HoG).
- Saliency detection-based (gebaseerd op menselijk waarnemen).

‘Statistical modelling’ in Figuur 3 bestaat uit twee hoofdgroepen:
‘Generative’ en ‘Discriminative.

Bij ‘Generative’ wordt een signatuur opgebouwd aan de hand van de visuele kenmerken van een object. Vervolgens wordt gekeken of deze kenmerken zich ook rond de opgegeven objectlocatie bevinden. Deze methode raakt afgeleid als in de achtergrond dezelfde visuele kenmerken gevonden worden als in het object.

Bij ‘Discriminative’ is het doel om classificatie (herkenning) toe te passen en zo een duidelijk onderscheid tussen het object (voorgrond) en achtergrond te krijgen. Het leert aan de hand van ‘training samples’en is dus afhankelijk van de kwaliteit van deze samples. Dit is effectief en efficiënt, maar kost veel rekenkracht.

De ‘Generative’ en de ‘Discriminative’ methoden zijn in zekere mate complementair, ze vullen elkaar aan. Vandaar dat er ook hybride mengvormen zijn waarbij methodes samengevoegd worden om zo de sterke punten uit beide methodes te benutten.

Tot zover deze compacte uitleg van ‘visual tracking’.
Het vereiste kennisniveau bevindt zich ver boven HBO niveau. Het meeste van wat hier genoemd wordt is materiaal om op te promoveren. Voor verdere uitleg verwijs ik naar de eerder genoemde vergelijkings- onderzoeken. (Li, et al., 2013) is goed leesbaar.

3.3 Problemen bij tracking

De opsomming in dit hoofdstuk is ontstaan door in de hiervoor genoemde documentatie te kijken welke problemen algemeen voorkomen bij trackers.

Er zijn vele zaken die het volgen van objecten lastig maken en waarmee rekening gehouden dient te worden:
- Camera sensors van slechte kwaliteit:
  - lage resolutie
  - lage bit-diepte
  - lage lichtgevoeligheid en daardoor traagheids-onscherpte (‘‘blur’')
  - kleurvervorming.

    - ruis
    - onscherpe beelden
- Moeilijke omstandigheden:
    - veranderend omgevingslicht
    - laag contrast
    - snelle camera- of object bewegingen
    - complexe bewegingen, niet in een vloeiende lijn
    - geheel- of gedeeltelijke overlappingen (occlusies)
    - kleine objecten
    - volgen van objecten die een veranderende vorm hebben:
        - veranderen van houding (‘pose change’)
        - rotatie, uit beeld draaien van objecten
        - vervorming (perspectief)
- Tijd gerelateerde vereisten:
    - ‘real-time’ verwerking: verwerking binnen een voorspelbare, liefst korte tijd
    - hoge ‘frame-rate’: embedded systemen zijn meestal beperkt in beschikbaar geheugen en in rekenkracht. Daardoor werken ze vaker met video met minder beelden per seconde (schokkerige beelden) waardoor tracking lastiger wordt.
- Bij het volgen van objecten met meerdere camera’s (‘fusion’) waarbij er geen overlap is in het beeldveld van de camera’s.

Al deze problemen kunnen er voor zorgen dat het tracken lastig of zelfs onmogelijk wordt om objecten te volgen. Er zijn veel verschillende type trackers ontwikkeld, elk met hun eigen specialisatie en elk met het vermogen om met een aantal van deze problemen beter om te gaan dan andere trackers.

3.4 Eisen aan een tracker voor gebruik op zee.

In hoofdstuk 2.1 2.2 is aangegeven dat de implementatie van een tracker moet gebeuren in de taal C++. Er wordt ontwikkeld op een Linux platform (32-bits / Intel CPU), met de mogelijkheid om de software op een embedded system te compileren.

Binnen dit project worden de volgende eisen gesteld aan een tracker (geordend op prioriteit). Deze eisen zijn gesteld door de opdrachtgever:

1. Robuustheid: het consequent blijven volgen van een enkel vaartuig
2. Snel- en lage processorbelasting in verband met het installeren op embedded camera hardware
3. Gebruik maken van Open Source software

Robuustheid is te definiëren als de mate waarin een tracker in staat is om objecten te blijven volgen. Zie ook de definitie van Robuustheid in hoofdstuk 4.1 zoals gesteld in de VOT challenge.

Hoe beter een tracker om kan gaan met de ‘moeilijke omstandigheden’ uit hoofdstuk 3.3 hoe robuuster deze is.

De kenmerken van robuustheid voor gebruik op zee zijn verder te detailleren:

- Om kunnen gaan met complexe bewegingen van een schip (draaien, stampen, rollen) ten opzichte van de op stabiele ondergrond staande camera.
- Om kunnen gaan met golven (oorzaak van ruis, onscherpe beelden, vormveranderingen, occlusies)
- Om kunnen gaan met drifting (voor de uitleg zie onderaan hoofdstuk 4.1)
- Om kunnen gaan met laag contrast

Deze kenmerken komen voort uit eigen observaties van beelden van schepen op zee en zijn benoemd in gesprekken met de opdrachtgever.

4 Onderzoek Vision componenten

In dit hoofdstuk is het onderzoek naar tracking en detectie algoritmes uitgewerkt.

4.1 Testen van trackers

Dit hoofdstuk geeft antwoord op de volgende deelvragen:

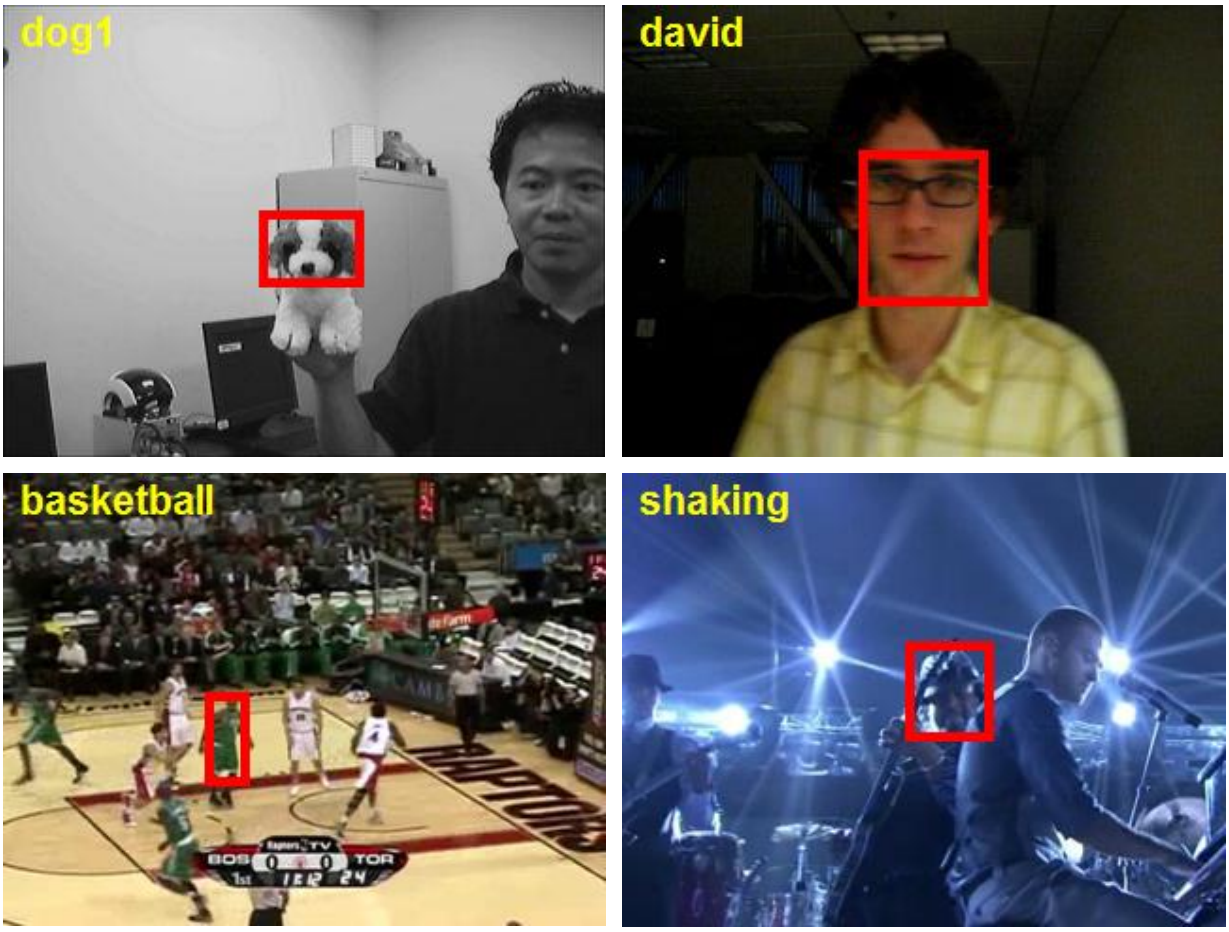
“Hoe kunnen trackers getest worden?”

“Welke tracker stelt de minste eisen aan hardware qua rekenkracht, geheugen, opslag?”

“Welke tracker is het meest geschikt voor het detecteren en volgen van vaartuigen op zee?

Alle schrijvers van wetenschappelijke artikelen beweren dat hun product het beter doet dan die van anderen. Uit eerdere onderzoeksprojecten is gebleken dat een wetenschappelijke ‘paper’ er veelbelovend uit kan zien, maar dat de resultaten in de praktijk voor de gewenste toepassing andere resultaten geven dan verwacht.

Bij het lezen van diverse papers waarin tracking algoritmes getest worden, komen regelmatig dezelfde plaatjes voorbij. De reden hiervoor is dat er, om te kunnen vergelijken, vaak een aantal standaard datasets (reeksen plaatjes) gebruikt worden. Als voorbeeld een aantal namen: ‘Dog’, ‘David’, ‘Basketball’ en ‘Shaking’, zie Figuur 4. Deze reeksen plaatjes bevatten specifieke problemen waar trackers mee om moeten kunnen gaan, zoals: variatie in grootte, richting, houding, veranderingen in belichting, onscherpte, occlusie etc. (zie hoofdstuk 3.3).



Figuur 4 Voorbeelden van veelgebruikte datasets (de dataset namen staan linksboven in de plaatjes)

Bij elke dataset hoort een ground truth definitie. Deze is met de hand gemaakt door in elk beeld met een bounding box aan te geven waar een object zich bevindt. Elke onderzoeker test hierop zijn tracker door de resultaten van zijn tracker te vergelijken met de ground truth.

Voorbeelden van vergelijkende onderzoeken zijn (Smeulders, et al., 2014) en (Ma & Liu, 2015). Beide onderzoeken beoordelen trackers op verschillende criteria en op verschillende manieren.

Door het gebruik van een beperkt aantal datasets en het niet gebruiken van een gestandaardiseerde methode bestaat het risico is dat testresultaten ‘gekleurd’ zijn. Daarom is het nodig om te beschikken over een testmethode die onafhankelijk, betrouwbaar en reproduceerbaar is.

Een mogelijke testmethode was tijdens de interviews met opdrachtgever al genoemd: de VOT Challenge, oftewel de ‘Visual Object Tracking’ challenge (Kristan, et al., 2015). Een ‘wedstrijd’ om te kijken hoe trackers onder diverse moeilijke omstandigheden presteren en uiteindelijk een ranglijst samen te stellen van trackers aan de hand van hun prestaties.

Vanuit de VOT challenge is de VOT toolkit beschikbaar (Čehovin, et al., 2014). De VOT toolkit is een geautomatiseerde testomgeving waarin Matlab en C++ gebaseerde trackers geïntegreerd kunnen worden en getest en vergeleken worden op een standaard manier.

De VOT toolkit is gebruikt tijdens de ICCV2013 (International Conference in Computer Vision), de ECCV2014 (European Conference on Computer Vision) en de ICCV2015 workshops om performance van trackers te vergelijken. Tijdens deze jaarlijks terugkerende workshops worden veel nieuwe trackers getoond. Dat de VOT challenge toolkit bij deze workshops als referentie geldt, was voor mij een goede reden om deze ook te gaan gebruiken.

De VOT toolkit bestaat uit een kant-en-klaar framework waarmee trackers getest kunnen worden en waarmee rapportages over de prestaties gemaakt kunnen worden. Een tracker wordt hierin als aparte module geïntegreerd. De prestaties van de tracker zijn onafhankelijk van de toolkit. De toolkit maakt gebruik van een selectie van 25 datasets uit een totaal aantal van 193 zorgvuldig geselecteerde datasets. Hieronder zijn er enkele waarin zee en golven voorkomen, zoals “surfing”.

Er zijn momenteel geen vergelijkbare toolkits waarmee trackers vergeleken kunnen worden.

Bij de VOT toolkit wordt de uitvoer van de tracker vergeleken met de ground truth en wordt er gemeten op de volgende eigenschappen:

- Bepalen van het middelpunt (eigenlijk zwaartepunt; Centroid) van een regio.
- Overlap van gedetecteerde regio en ground truth regio.
- Het aantal frames- of de tijdsduur dat een tracker een object kan volgen (gecombineerd met een drempelwaarde van de overlap waarbij deze nog goedgekeurd wordt).
- Het aantal keren dat de tracker het object kwijtraakt (en waarbij de tracker opnieuw opgestart wordt aan de hand van ground truth beelden).

Uiteindelijk worden bovenstaande punten uitgedrukt in twee meetwaarden aan de hand waarvan scores toegekend worden:

Robuustheid wordt in de VOT challenge gemeten door het aantal keren te tellen dat een tracker het object volledig kwijtraakt en opnieuw gestart dient te worden.

Nauwkeurigheid (‘Accuracy’) wordt in de VOT challenge gemeten door te meten in welke mate de bounding box die door de tracker wordt aangegeven overlapt met de ground truth bounding box.

Bij tracking treden de volgende standaardfouten op:

- Afwijking: de tracker wijkt af van de ground truth.
- Valse positieven: de tracker volgt een object welke niet het doel is.
- Valse negatieven: de tracker herkent het object niet en volgt deze dus niet.

Ondanks dat trackers allemaal voor een specifiek doel ontwikkeld zijn en hun sterke en zwakke plekken hebben, worden ze allemaal op dezelfde manier getest.

Het gebruik van een bounding box zoals bij deze methode gedaan wordt is de oudste methode om objecten te markeren. Het is niet ideaal om een object te markeren. Het zou beter zijn om een nauwsluitend kader om het object te trekken zodat er geen achtergrond in het trackingproces meegenomen wordt.

Met een bounding box wordt mogelijk wel achtergrond meegegeven wat mogelijk door de tracker als deel van het object gezien en gevolgd wordt. Dit effect heet ‘drifting’. Het gebruik van een bounding box ligt het meest voor de hand omdat dit het testmechanisme universeel houdt en omdat alle beschikbare datasets met bounding boxes werken (‘ground truth’ data).

4.2 Selectie van de beste tracker

Tijdens het onderzoek was het de bedoeling om de VOT toolkit te gebruiken om feature point trackers te testen. De opdrachtgever had immers aangegeven dat hij primair onderzoek wilde naar feature point trackers (zie hoofdstuk 2.3). Mijn aanname was dat ik een selectie kon maken uit één van de feature point trackers uit OpenCV.

Dit was een misvatting en leverde geen bruikbaar resultaat op. De reden hiervoor was dat ik niet volledig begreep hoe trackers werken. In feite was ik alleen bezig met het testen van feature point *detectors* welke slechts één component van een tracker zijn. Om te begrijpen hoe trackers functioneren heb ik mij alsnog verdiept in de werking en opbouw van trackers. Het resultaat van dit onderzoek is te lezen in hoofdstuk 0.

Om aan de wens van de opdrachtgever tegemoet te komen (onderzoek naar de bruikbaarheid van feature point trackers) is gekeken welke feature point trackers in de VOT challenge getest zijn. Onder de 38 trackers die tijdens de VOT challenge getest zijn waren er twee feature point trackers; Matrioska en Matflow (zie ‘Bijlage B’ voor de beschrijvingen). Qua prestaties (zie Figuur 5) zaten deze trackers onder het gemiddelde.

Hierop is in samenspraak met de opdrachtgever besloten om het onderzoek breder te trekken en naar de beste tracker te zoeken, ongeacht hoe deze werkt.

Bij dit selectieproces is gekeken naar verschillende onderzoeken naar gelijksoortige trackers, zoals (Smeulders, et al., 2014) en (Ma & Liu, 2015). De evolutie in trackingalgoritmes gaat zeer snel. Jaarlijks verschijnen er tientallen nieuwe trackers met steeds betere prestaties. Het vergelijkingsmateriaal veroudert snel en wordt niet meer als vergelijksmateriaal gebruikt.

Daarop is besloten om te kijken naar het meest recente onderzoek en daar de resultaten van te evalueren. Het meest recente onderzoek is de VOT Challenge 2015. Deze loopt echter nog en is in december 2015 afgerond. Hierdoor is het niet bruikbaar voor gebruik in dit onderzoek. Het breedste en meest recent afgeronde onderzoek is de VOT Challenge 2014. Deze levert een ranglijst op van trackers (de ‘long list’) gesorteerd op prestaties van diverse kenmerken.

De top 5 score van trackers is te zien in Tabel 1, de uitleg van de gebruikte symbolen staat onder de tabel.

Tracker		*baseline*			*region_noise*					
		R_A	R_R	R	R_A	R_R	R	R_Σ	Snelheid	Open Source code beschikbaar
1	DSST*	**5.41**	11.93	**8.67**	**5.40**	12.33	**8.86**	**8.77**	7.66	Ja
2	SAMF*	**5.30**	13.55	**9.43**	**5.24**	12.30	**8.77**	**9.10**	1.69	Nee
3	KCF*	**5.05**	14.60	9.82	**5.17**	12.49	**8.83**	**9.33**	24.23	Ja
4	DGT	10.76	9.13	9.95	8.31	9.73	9.02	9.48	0.23	Nee
5	PLT_14*	13.88	**6.19**	10.03	13.12	**4.85**	8.99	9.51	**62.68**	Nee

Tabel 1 Top 5 score van trackers uit de VOT Challenge 2014

In Tabel 1 zijn de beste scores **groen** gemarkeerd, de tweede plaats is **rood** gemarkeerd en de derde plaats **blauw**. De kolomnamen R_A, R_R en R staan voor ‘Ranking by Accuracy ‘, ‘Ranking by Robustness’ en ‘Total Ranking’. De kolom R_Σ geeft de eindscore aan, het gemiddelde van ‘baseline’ en ‘region_noise’. De uitleg van ‘baseline’ en ‘region_noise’ volgt verderop.

Een lager getal bij ‘ranking’ is een betere score. Bij de snelheid is een hogere waarde een hogere snelheid. Een * achter de naam van de tracker geeft aan dat de prestaties geverifieerd zijn door het VOT comité.

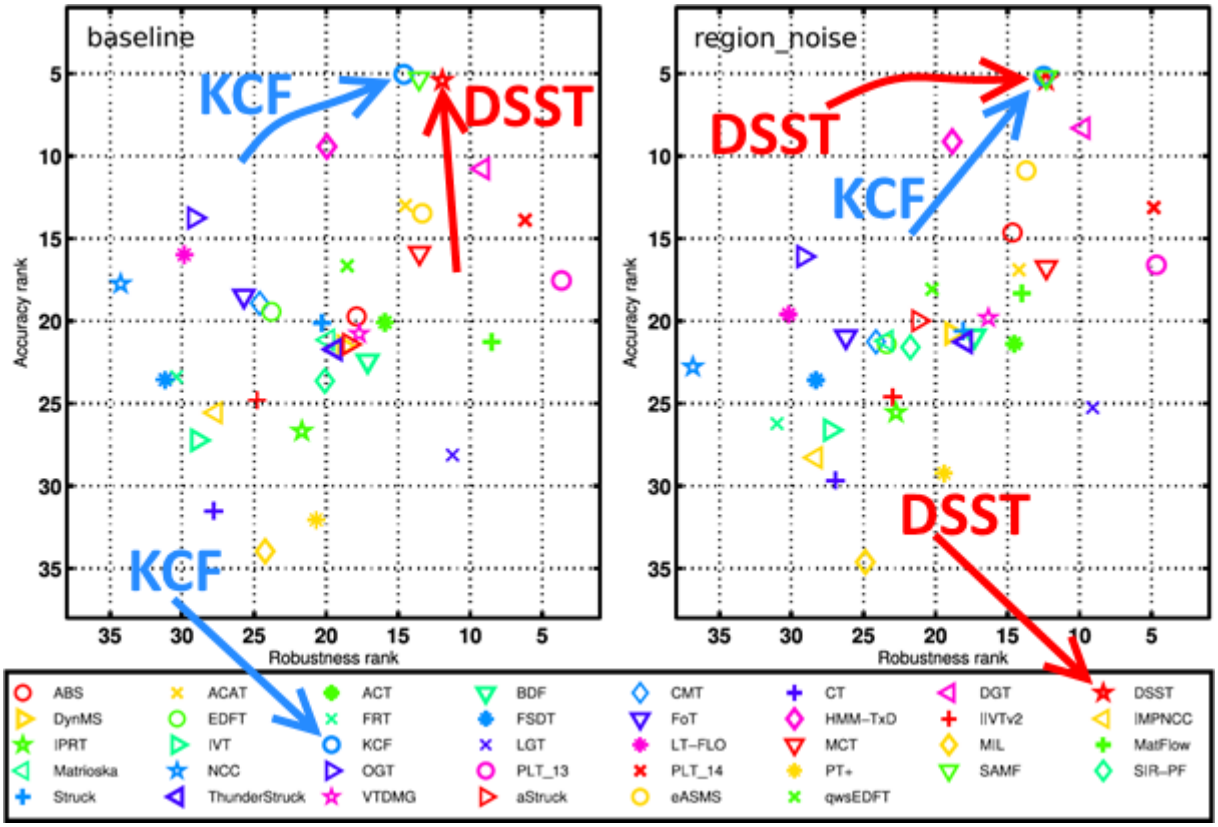
Nu de selectie op robuustheid en snelheid gedaan is, volgt de selectie op de derde eis:
Het beschikbaar zijn van ‘Open Source’ broncode.

Bij het zoeken naar broncode bleek dat deze van de meeste trackers niet openbaar gemaakt wordt. Alleen van twee best geteste trackers (DSST en KCF) was de broncode beschikbaar. Daarmee bleven deze in de selectie over. In ‘Bijlage B’ staat een beschrijving van de werking van beide trackers.

In Figuur 5 zijn de resultaten te zien van de VOT2014 challenge (het beste te bekijken in kleur). De score van een tracker wordt hierin weergegeven in een AR-plot; Accuratesse (nauwkeurigheid) versus Robuustheid. Linksonder in de grafiek is slecht, rechtsboven is goed.

De linker grafiek (‘baseline’) geeft het optimale resultaat weer in vergelijking met de initiële ‘ground truth‘ bounding box, de rechter grafiek (‘region_noise’) is het gevolg van starten met een bounding box die maximaal 10% in grootte en plaats afwijkt van de ground truth.

Hieruit kan afgeleid worden dat volgens de VOT2014 challenge DSST het meest robuust is. In beide grafieken staat DSST op dezelfde plaats. Dat betekent dat zelfs als de bounding box niet exact het object omsluit, het object toch goed gevolgd wordt en weinig last van ‘drifting’ ondervindt.



Figuur 5 VOT2014 accuracy-robustness ranking plots (Kristan, et al., 2014)

De resultaten van VOT2014 geven aan dat de DSST tracker als beste ‘allround tracker’ getest is. De KCF tracker staat op de derde plaats, maar geeft de meest accurate bounding box en volgens de test sneller dan DSST. DSST is de meest robuuste van de twee.

In (Kristan, et al., 2014, p. 14) is te zien dat er een omgekeerd evenredige relatie is tussen snelheid en accuratesse en robuustheid. Voor het volgen van vaartuigen op zee is de robuustheid het belangrijkste.

Bevindingen met beelden uit de haven

DSST en KCF zijn getest en gedemonstreerd op beelden uit de haven van Rotterdam (de datasets staan op de server in Gouda). Er zijn geen beelden beschikbaar van werkelijke omstandigheden op zee. De beelden uit de haven komen voldoende overeen met te verwachten werkelijke omstandigheden op zee: Er komen grote en kleine, snelle en langzame schepen in voor. Van sleepboot en binnenvaartschip tot speedboot. Het effect van de golven is voldoende realistisch, hoewel bij harde wind op zee occlusies kunnen ontstaan door overslaande golven.

De demo-code van beide trackers berekent en toont hoeveel beelden per seconde de tracker kan halen. De beelden uit de haven van Rotterdam hebben een resolutie van 1024x768 bij 25 beelden per seconde. Beide trackers gaven met de gebruikte PC (DualCore 2,2GHz) aan dat ze ver boven die vereiste 25 beelden per seconde uitkwamen (DSST: 80fps, KCF: 60fps).

De processorbelasting bij de demoprogramma’s van beide trackers is gemiddeld 53%. Hierin moet vermeld worden dat na elk beeld dat door het demoprogramma weergegeven wordt 10ms vertraging is ingebouwd.

Het geheugengebruik van de demoprogramma’s van beide trackers is vergelijkbaar. Met het programma ‘top’ is gekeken wat het geheugengebruik is.
Dit zijn de maximaal waargenomen waarden als videoweergave is uitgeschakeld

(videoweergave is alleen nodig voor demo en debuggen): ‘Data + Stack’; 52kB, ‘Shared Memory’; 6kB, ‘Resident Size’; 40kB, ‘Virtual Image’; 96kB

DSST is bij de havenbeelden in alle gevallen sneller dan KCF. Waarschijnlijk komt dat doordat DSST een efficiënter zoekpatroon heeft dan KCF en dat valt pas op bij een hogere resolutie beeld. De VOT challenge maakt gebruik van lage resolutie beelden van 320x200 tot maximaal 640x480, terwijl de havenbeelden 1024x768 zijn. Dat strookt met de observatie dat KCF bij de havenbeelden sneller wordt als de bounding box en dus het object groot is.

Dit kan getest worden door de havenbeelden om te zetten naar een dataset voor de VOT challenge en daar de trackers nogmaals mee te testen. Binnen dit project is daar echter geen tijd meer voor.

DSST is bij de havenbeelden op het oog het meest robuust. Dat komt overeen met de bevindingen uit de VOT challenge.

Hiermee kan antwoord gegeven worden op de deelvraag:
“Welke tracker is het meest geschikt voor het detecteren en volgen van vaartuigen op zee?”
DSST is op dit moment de meest geschikte tracker voor het volgen van vaartuigen op zee.

De deelvraag *“Welke tracker stelt de minste eisen aan hardware qua rekenkracht, geheugen, opslag?”* is als volgt te beantwoorden:
DSST is de snelste en stelt daardoor de minste eisen qua rekenkracht. De eisen qua geheugen en opslag zijn niet bekend en de VOT challenge bevat geen tests met betrekking tot geheugen en opslag. Zoals hiervoor is aangegeven hebben beide trackers dezelfde lage behoeften qua geheugen. Schijfopslag voor data is voor beide trackers niet nodig. De hoeveelheid benodigd geheugen is geen belemmering om de trackers op een embedded system te laten werken.

Het antwoord op de deelvraag *“Hoe kunnen trackers getest worden?”* is niet in één zin te geven. In dit hoofdstuk is aangegeven hoe trackers op een generieke wijze met elkaar vergeleken kunnen worden. De prestaties van een tracker qua nauwkeurigheid worden gemeten door de overeenkomsten tussen ground truth en bounding box van de tracker te vergelijken. Op dit moment is de VOT toolkit het meest geschikt om dit mee te doen.

Aanbevelingen voor vervolgonderzoek: Inmiddels is de VOT2015 challenge afgerond. Hierin zijn 62 trackers getest waarvan vele nieuwe. In de VOT2015 is te zien dat DSST beneden gemiddeld scoort. Een aanbeveling is om te kijken naar een nog beter presterende tracker.

4.3 Signatuur opbouwen

Dit hoofdstuk geeft antwoord op de volgende deelvraag:

“Hoe kun je een signatuur (‘Appearance Model’) van een object opbouwen en bijhouden?”

De signatuur of ‘Appearance Model’ is de visuele representatie van een object, oftewel: hoe kun je dat object omschrijven. Daar zijn vele methodes voor, zie de ‘lijstjes’ in hoofdstuk 0. Het opbouwen van een signatuur wordt binnen dit project op verschillende punten gedaan. De tracker maakt intern gebruik van een signatuur, maar ook de detector maakt gebruik van een signatuur. Dit hoofdstuk loopt vooruit op hoofdstuk 4.4 door alvast te kijken welke methode van signatuur opbouwen geschikt is om een detector mee te bouwen.

Voordat dit project begon is door de opdrachtgever de aandacht gevestigd op ‘Feature Point’ of ‘Key Point’ detectie algoritmes. Genoemd is onder andere FREAK ‘Fast Retina Keypoint’ (Alahi, et al., 2012). Feature Point detectors kijken naar specifieke delen in het beeld, met het doel deze in een ander beeld te herkennen. Een eenvoudige vorm van Feature Point detector is een Corner Detector die hoeken in groepen pixels herkent.

Elke Feature Point detector is ontwikkeld met een eigen specifiek doel. In een eerder deel van het onderzoek (zie hoofdstuk 4.2) is alleen gekeken welke detector de meeste feature points

opleverde. Dat was de ORB detector. Er is niet naar andere eigenschappen gekeken zoals snelheid, gevoeligheid voor schaalveranderingen of rotatie. In Bijlage A staat een overzicht van Feature Point detectors uit OpenCV en de eigenschappen.

Bij de keuze voor een methode voor het opbouwen van een signatuur wordt gekeken op basis van welke visuele kenmerken een vaartuig van zijn omgeving (de zee) onderscheiden kan worden. Daarbij vallen een aantal methodes af. Eenvoudige veelgebruikte varianten zijn onderscheid maken op basis van kleur of helderheid. Dit valt af omdat de kleur van een schip niet te voorspellen is, net als de helderheid. Andere methoden zijn ‘edge’ of ‘line’ detectors. Die zijn niet bruikbaar omdat hiermee niet te bepalen is wat deel van het schip is of niet.

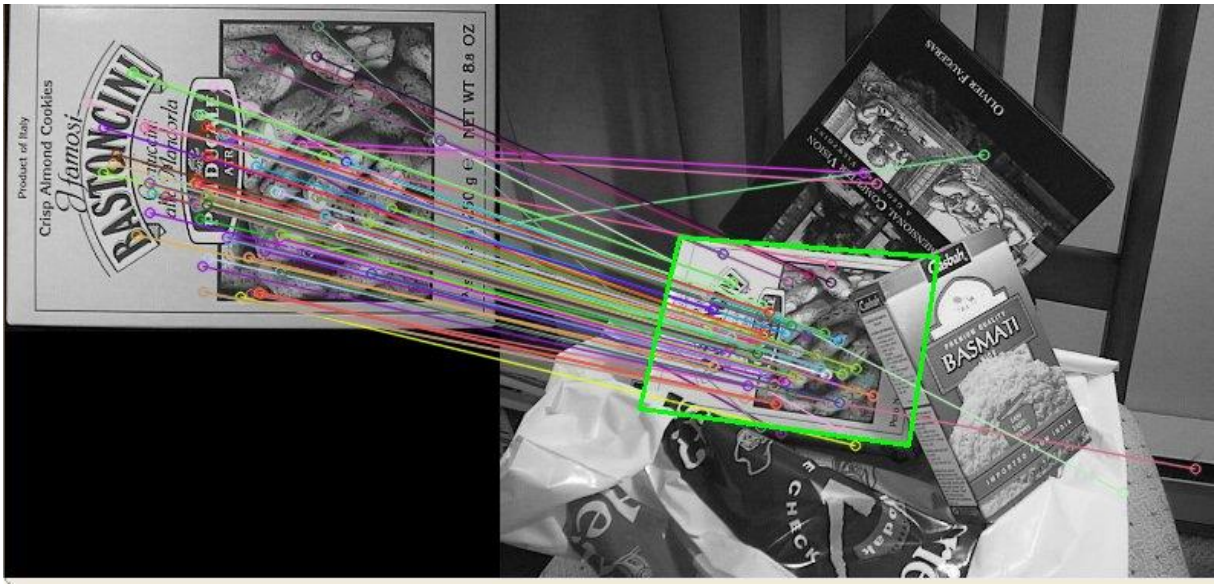
Er zijn meer geavanceerde methodes beschikbaar, deze vergen echter meer onderzoektijd dan binnen de afstudeeropdracht beschikbaar is.

In Figuur 6 is een voorbeeld te zien van het gebruik van Feature Points. In het linker deel van het beeld is een koekjesdoos te zien (een object). In het rechter deel van het plaatje is een scene te zien waarin een zelfde koekjesdoos voorkomt (met occlusie door de basmati doos).

De feature point detectie bestaat eruit dat markante punten die in het linker plaatje gedetecteerd zijn (de ‘best matches’), teruggevonden worden in het rechter plaatje. Om de punten uit het object te kunnen terugvinden, worden de gedetecteerde punten in een lijst opgenomen.

Het originele object is gemarkeerd met een bounding box, en in de scene waarin het object herkend wordt, wordt deze bounding box gereconstrueerd. Aan de hand van de layout van de keypoints in het object wordt uitgerekend in de scene hoe de bounding box daar komt te vallen. Die reconstructie heet een homography. Om een homography te kunnen uitrekenen zijn minstens 4 overeenkomstige punten (good matches) tussen object en scene nodig. Hoe meer punten, hoe beter de afbeelding.

Uitgaande van de keuze voor feature point detectie is het antwoord op de deelvraag: “*Door gedetecteerde punten te verzamelen ontstaat een representatie.*“
In het volgende hoofdstuk wordt uitgelegd hoe dit (een corner detector) gebruikt wordt om een vaartuig op zee te detecteren.



Figuur 6 Voorbeeld van feature point detectie en een Homography (OpenCV, 2015)

4.4 Detecteren van objecten

Dit hoofdstuk geeft antwoord op de volgende deelvraag:

“Hoe kan een object initieel gedetecteerd worden?”

Zoals in hoofdstuk 3.1 is aangegeven, is de tracker niet in staat zelf de beginlocatie voor tracking te bepalen, deze moet aan de tracker in de vorm van een bounding box opgegeven worden. Een object (in dit geval een schip) moet dus gedetecteerd worden.

Binnen dit project zijn er twee verschillende startmomenten (zie Figuur 2 op pagina 2). Tracking wordt automatisch geïnitialiseerd vanuit “Parameter Intrusion Detection” (PID) of na handmatig starten. Vanuit de PID wordt er een bounding box of coördinaten meegegeven, de objectdetectie daarna dient ter verificatie. Bij het handmatig starten dient de PTZ camera gericht te worden waarna de objectdetectie gestart wordt.

In bewegingsdetectie zit een gelaagdheid:

- “Motion Estimation” – is er beweging?
- “Extraction of Relevant Motion” – wat is het relevante deel?
- “Motion Prediction” – voorspellen waar een object naar toe gaat.

Hier gaat het om het detecteren van relevante beweging. Daar zijn twee methodes voor:

- Background Subtraction
- Optical Flow / motion vectors

Background Subtraction

In een eerder onderzoek (Neijsel, 2015) zijn diverse background subtraction algoritmes onderzocht. Het meest effectief is een vorm van background subtraction waarbij een model wordt opgebouwd van de achtergrond. Daarbij worden meerdere opeenvolgende beelden samengevoegd tot een stabiele achtergrond waarbij een bewegend object duidelijk afsteekt tegen de achtergrond.

Dat is prima bruikbaar bij een stabiele achtergrond waar weinig gebeurt. Met golven op zee werkt dat niet goed doordat er veel beweging is van de golven. Een klein vaartuig zal door veel beweging niet goed waarneembaar zijn. Het opbouwen van een model met background subtraction kost tijd. Afhankelijk van de rekenkracht en beeldgrootte kan dit oplopen tot meerdere seconden. Het is niet geschikt om binnen enkele frames een detectie uit te voeren.

Optical Flow / motion vectors

Er zijn meerdere Optical Flow algoritmes, maar er zijn twee hoofdrichtingen: “Sparse Optical Flow” en “Dense Optical Flow”. Bij de “Sparse” variant worden goed herkenbare punten gemarkeerd en gevolgd, bij de “Dense” variant wordt elke pixel die van zijn buren onderscheiden kan worden gevolgd. De dichtheid (“density”) is veel hoger.

Er is getest met gemodificeerde demo-code van Optical Flow algoritmes uit OpenCV.

De eerste methode is "Sparse Optical Flow" met gebruik van de OpenCV functies GoodFeaturesToTrack() en CalcOpticalFlowPyrLK(). GoodFeaturesToTrack() is "Shi and Tomasi Feature Tracking" wat werkt met feature points (corner detection). CalcOpticalFlowPyrLK() is "Pyramidal Lucas Kanade Optical Flow".

De andere methode is “Farnebäck Dense Optical Flow”, met gebruik van de OpenCV functie calcOpticalFlowFarneback() (Farnebäck, 2003).

Lucas Kanade Sparse Optical Flow

De "Pyramidal Lucas Kanade Optical Flow" methode volgt een aantal feature points op het bewegende object. Zoals in 4.3 aangegeven is, werken feature point trackers uitstekend en betrouwbaar als er voldoende ‘good matches’ zijn. Op de beelden uit de haven werkte dit accuraat als het object groot genoeg was. Bij kleinere objecten (verder weg), onscherpte of

laag contrast wordt de bruikbaarheid minder. Het grootste nadeel van het gebruik van feature points is dat het niet geschikt is om de omtrek van een object te markeren.

Bij snel bewegende objecten ontstaat ‘motion blur’: bewegingsonscherpte. Hierop kunnen geen feature points herkend worden. Het gevolg is dat een snel object niet gedetecteerd wordt. Hier is iets aan te doen door voor een wijder blikveld te zorgen; uitzoomen, groothoeklens gebruiken, of een gevoeliger camera te gebruiken met kortere sluitertijden/belichtingstijden.

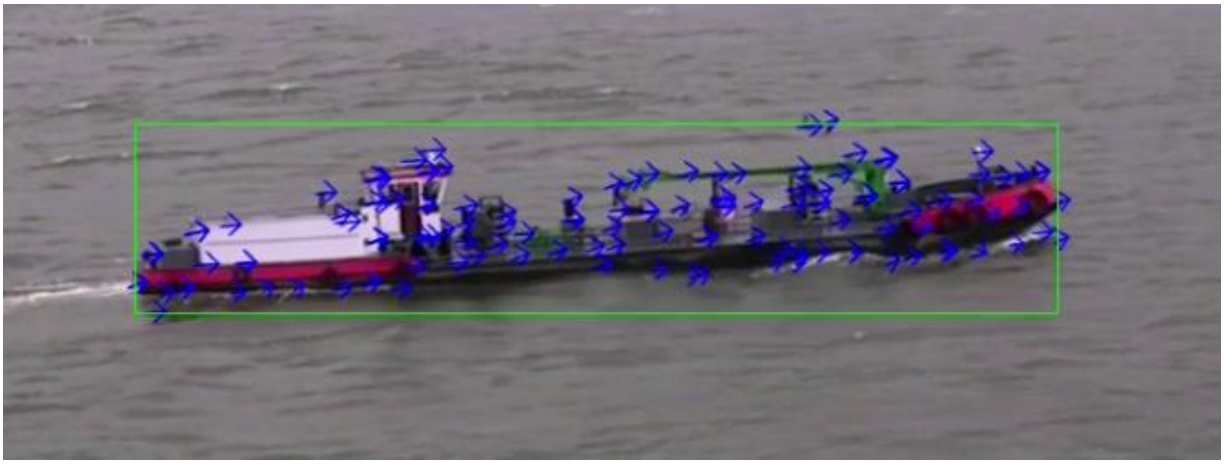
De werking van “Lucas Kanade Optical Flow” is zichtbaar gemaakt door het bewegende object te markeren met vectoren die de bewegingsrichting aangeven (3x de normale lengte), zie Figuur 7. Alle 'ruis' is weggewerkt door kleine vectoren (lengte<1,5pixels) en grote vectoren (lengte>16pixels) weg te gooien.

Door te filteren op de richting en grootte van de vectoren blijft het bewegende object over. Het object wordt gemarkeerd door om de eindpunten van de vectoren een bounding box te trekken. Door een minimum grootte voor een bounding box in te stellen (in de demo: 19x19 pixels) worden misdetecties voorkomen.

Deze methode werkt goed. In de demo is ervoor gekozen om zo snel mogelijk objecten te detecteren uit opeenvolgende frames. Het is zelfs mogelijk om langzame bewegingen te registreren door ‘oude’ punten gedurende meerdere frames te onthouden. Dit is in de demo niet toegevoegd.

Zie “Bijlage F” voor een flowchart met een ontwerp van de werking van de detector.

Aanbeveling voor vervolgonderzoek: Op dit moment ‘anticipeert’ de detector niet. Het is mogelijk op basis van de laatste bewegingsvector van het object de volgende locatie voor de bounding box te voorspellen. Dit heeft alleen nut bij snelle objecten. Het kan ook gebruikt worden om traagheid in het systeem te compenseren (overdracht naar tracker, traagheid in PTZ aansturing). Het werkt nadelig bij snelle richtingsveranderingen.



Figuur 7 Lucas Kanade Optical Flow (weergegeven vectoren zijn 3x werkelijke lengte)

Farnebäck Dense Optical Flow

De andere methode die is onderzocht is gebaseerd op de OpenCV functie calcOpticalFlowFarneback(). In de OpenCV demo wordt gebruik gemaakt van de methode drawOptFlowMap() om het effect zichtbaar te maken. Deze projecteert een raster van vectoren over het originele beeld (zie groene vectoren in Figuur 8). Zodra er beweging wordt waargenomen tussen twee opeenvolgende beelden, geeft het raster van vectoren de richting en afstand van de beweging weer.

Op basis hiervan is het goed mogelijk om de omtrek van een object weer te geven.

De OpenCV functie calcOpticalFlowFarneback() genereert een matrix met dezelfde beeldresolutie als het bronbestand. Elke 'pixel' is echter geen grijswaarde maar een vector met de richting en afstand welke het verschil tussen beide plaatjes geeft.

Om hier iets mee te kunnen, dient de matrix eerst geconverteerd te worden. Ik heb hiervoor de functie flowToGray() geschreven. Deze berekent van elke 'pixel' de lengte, en schrijft deze naar een nieuw grayscale plaatje. Als er geen beweging is, is de waarde van de pixel nul (zwart), als de verschuiving tussen de twee plaatjes 4 pixels is, heeft de pixel de grijswaarde 4 (255 is wit). Zie Figuur 9.

Om dit visueel te verduidelijken is in Figuur 9 de waarde van elke pixel met de waarde 8 vermenigvuldigd. Hiermee wordt zichtbaar dat hoe hoger de snelheid is, hoe lichter het object is. Ook is te zien hoe groot een object is, want alle pixels in dat object gaan met dezelfde snelheid in dezelfde richting.

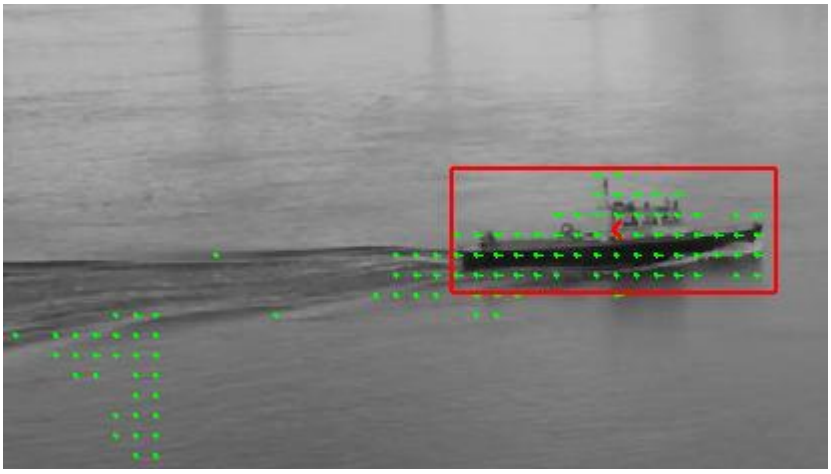
Door geschikte drempelwaardes in te stellen, zoals bij de Lucas Kanade methode, blijft alleen het object over. Bij beide methodes is het zinvol om tussen bepaalde drempelwaarden te filteren. Lage waarden (zoals 1) staan voor ruis, bijvoorbeeld veroorzaakt door golfslag. Hoge waarden (groter dan 15) zijn willekeurige uitschieters, en dus ruis. Met havenbeelden is proefondervindelijk bepaald dat waarden tussen 4 en 15 bruikbaar zijn. Zie Figuur 10.

De filterwaarden zullen in een definitieve opstelling ook proefondervindelijk bepaald moeten worden, want de drempelwaarden zijn afhankelijk van de frame-rate en de snelheid van naderende objecten. Een lagere frame-rate of hogere object snelheid levert grotere verschillen op. Een poging tot het automatisch bepalen van drempelwaarden op basis van een histogram was niet succesvol.

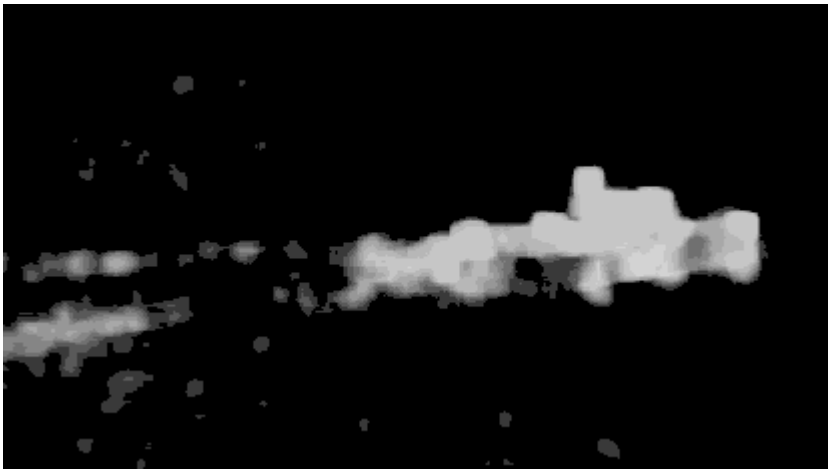
Een histogram is als een staafdiagram waarbij de lengte van elke staaf staat voor het aantal pixels van een bepaalde grijswaarde. Een histogram loopt van zwart (waarde 0) tot wit (waarde 255). Een automatische keuze van een drempelwaarde zou kunnen zijn door die grijstint (of grijstinten) te selecteren waarvan er de meeste pixels zijn. Dit is de manier waarop een automatische threshold functie werkt.

Als er veel golfslag is, levert dit veel pixels op met de waarde 1 of 2. Automatisch bepalen van de drempelwaarde werkt dan niet. Het proefondervindelijk instellen van drempelwaarden en het gebruiken van meerdere opeenvolgende pixelwaarden levert de beste resultaten op.

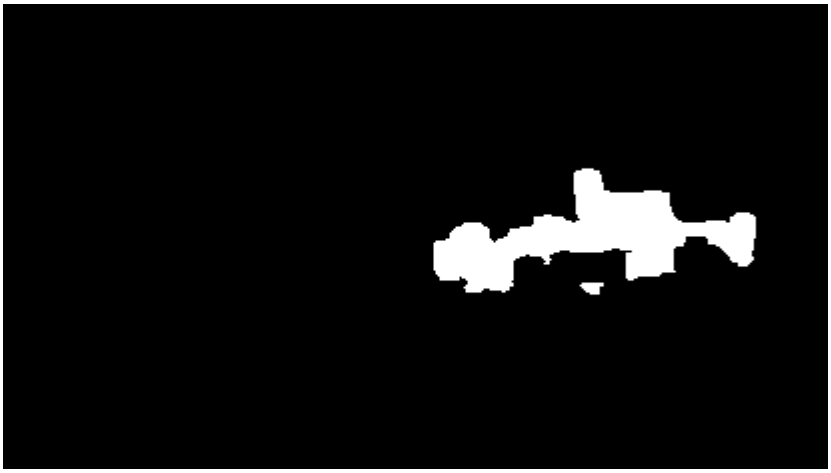
Na het toepassen van drempelwaarden ontstaat er een binair plaatje (zwart en wit) waarin duidelijk een object gemarkeerd is. Hierop is het eenvoudig om een bounding box te bepalen. Voor het bepalen van de bounding box wordt het grootste bewegende object genomen.



Figuur 8 Videobeeld van schip gemarkeerd met bounding box



Figuur 9 Optical Flow vertaald naar grijswaarden



Figuur 10 Resultaat na toepassen van drempelwaarden op Figuur 9

Selectie van Optical Flow methode

Beide detectiemethodes zijn getest met realistische beelden uit de haven van Rotterdam met onder andere golfslag, recht op de camera af varende trage schepen en snelle vlak voor de camera langsvarende schepen.

Detectie met Optical Flow heeft zijn beperkingen. Bewegingen in het platte vlak (naar boven, onder, links, rechts) zijn eenvoudiger te detecteren dan een beweging richting de camera.

Uit de evaluatie van zowel de Lucas Kanade Sparse Optical Flow als de Farnebäck Dense Optical Flow komen de volgende bevindingen:

Bij gebruik van camerabeelden met een resolutie van 1024 x 768 kost het verwerken van één frame met Farnebäck bijna 1 seconde (op Core2Duo 2,2GHz processor). Dat is erg traag en niet geschikt om op een embedded device te implementeren.

Farnebäck is bruikbaar met videobeelden uit een filmpje (waarmee getest is), maar is niet bruikbaar met live beelden van een camera. Het levert problemen op bij snel bewegende vaartuigen, maar ook bij hele langzame vaartuigen.

Lucas Kanade Sparse Optical Flow heeft een hoge verwerkingssnelheid. Op genoemde PC was er nauwelijks vertraging waar te nemen. Afhankelijk van de implementatie is Lucas Kanade geschikt voor zowel snelle als trage vaartuigen. Zoals gezegd is het grootste nadeel het niet exact aan kunnen geven van de vorm van een object. Voor het markeren van een object is het niet relevant of de omtrek exact aangegeven wordt. Dit nadeel heeft alleen effect als er automatisch beeldvullend op het object ingezoomd wordt.

De snelheid is de belangrijkste vereiste voor het detectiemechanisme. Om die reden is gekozen voor Lucas Kanade. Voor de werking van de Lucas Kanade detector zoals deze hiervoor beschreven is, zie “Bijlage F”. Tabel 2 geeft een schematisch overzicht van de verschillen tussen Dense en Sparse optical flow.

	Lucas Kanade Sparse Optical Flow	Farnebäck Dense Optical Flow
Nauwkeurigheid bij omtrek markeren	Laag	Hoog
Gevoeligheid voor (snelheids) onscherpte	Hoog	Laag
Geschiktheid voor detecteren trage objecten	Goed	Redelijk
Geschiktheid voor detecteren kleine objecten	Redelijk	Redelijk
Geschiktheid voor detecteren grote objecten	Goed (bij gebruik feature point detection en homography)	Redelijk
Snelheid	Hoog (>10 frames/sec)	Laag (<1 frame/sec)

Tabel 2 Vergelijking tussen Dense en Sparse optical flow

Het antwoord op de deelvraag *“Hoe kan een object initieel gedetecteerd worden?”* is:
Door gebruik van Lucas Kanade Sparse Optical Flow in combinatie met het gebruik van Feature Points.

Beelden vanaf een boorplatform op zee waren niet beschikbaar. Het heeft de voorkeur om de detector verder te testen en in te regelen met beelden van open zee vanaf een boorplatform.

Een interessant vervolgonderzoek zou zijn om de “Shi and Tomasi” feature point detector te vervangen door een andere feature point detector uit OpenCV (zie hoofdstuk 4.3 en “Bijlage A – Featurepoint detectors in OpenCV 3.0.0”). Wellicht zijn hiermee betere resultaten te bereiken.

5 Onderzoek naar PTZ aansturing

Dit hoofdstuk geeft antwoord op de deelvraag: *“Wat is de meest geschikte regelaar om de PTZ camera in de pas te laten lopen met de tracker?”* en de bijbehorende sub-deelvragen:

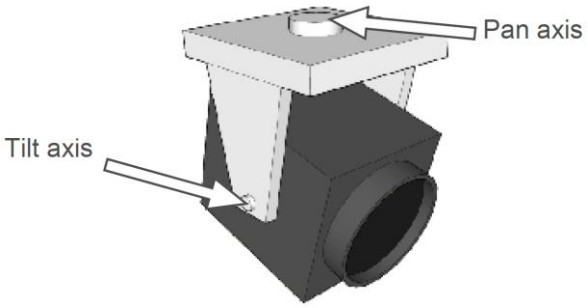
a. *Is een lage orde PID regelaar voldoende of is een hogere orde regelaar, of zelfs een niet lineaire regelaar nodig?*
b. *Hoe kunnen we het optimale zoomniveau bepalen zodat de tracker het object niet kwijtraakt?*
c. *Hoe kunnen we na kwijtraken van het object deze weer terugvinden?*

Het antwoord op deze vragen wordt gegeven aan de hand van experimenten en analyse, ondersteund door literatuuronderzoek.

5.1 Methoden voor PTZ motor aansturing

Eerst dient uitgezocht te worden hoe PTZ camera’s aangestuurd kunnen worden.
Daarom geeft dit hoofdstuk geeft antwoord op de vraag :
“Wat is de meest geschikte methode om een PTZ camera aan te sturen?”.

De fysieke bewegingen van een PTZ camera zijn te ontleden in de componenten Pan (draaien), Tilt (kantelen) en Zoom (dichterbij halen of afstand nemen), zie Figuur 11. Deze bewegingen kunnen elk apart of in combinatie aangestuurd worden.



Figuur 11 Pan en Tilt assen van een PTZ camera (ONVIF, 2008)

Om de fysieke bewegingen Pan, Tilt en Zoom in gang te zetten dienen er softwarematig commando’s naar de camera verstuurd te worden.
Er zijn drie methoden om deze commando’s te sturen:

- Absolute Move
- Continuous Move
- Relative Move

Bij de ‘**Absolute Move**’ wordt per bewegingsrichting (Pan, Tilt, Zoom) een absolute waarde voor de bestemming opgegeven. Dit werkt als het opgeven van coördinaten of richtingsvector. Bij de standaard versie van ‘Absolute Move’ worden de motoren met de hoogste snelheid naar de coördinaten gestuurd. Optioneel kunnen voor Pan, Tilt en Zoom de bewegingssnelheden apart opgegeven worden. ‘Absolute Move’ is snel en accuraat qua positioneren, maar bewegen tussen verschillende posities verloopt schokkerig.

Bij de ‘**Continuous Move’** wordt voor Pan, Tilt en Zoom alleen een snelheid opgegeven. Dit levert een richtingsvector op. De ingezette beweging blijft doorgaan totdat er een ‘Continuous Move’ commando komt met andere waardes (of nullen om te stoppen) of tenzij er een timeout waarde voor de tijdsduur bereikt is. Met ‘Continuous Move’ kunnen vloeiende bewegingen gerealiseerd worden.

De ‘**Relative Move**’ is een tussenvorm tussen ‘Absolute Move’ en ‘Continuous Move’. Per bewegingsrichting (vector) wordt een tijdsduur en dus nieuwe locatie voor die beweging ingesteld. Ook met ‘Relative Move’ kunnen vloeiende bewegingen gerealiseerd worden.

Het softwarematig aansturen van een PTZ camera kan op verschillende manieren:

- ONVIF standaard
- CGI van de camera webpagina
- API van de camera leverancier

Aansturen met behulp van de ONVIF standaard

De basis voor de aansturing bestaat meestal uit een webserver die zich bij de camera bevindt. De gangbare benaming hiervoor is een NVT (‘Network Video Transmitter’); een netwerk video server bestaande uit een IP camera of analoge camera in combinatie met een video encoder die data over een IP netwerk verstuurt naar een NVC (‘Network Video Client’).

De NVC is bijvoorbeeld een PC met een webbrowser.
De communicatie tussen NVT en NVC verloopt op één van de drie genoemde manieren; via ONVIF, een camera API of via de camera webpagina / CGI.

ONVIF is een non-profit organisatie die als doelstelling heeft om standaarden te ontwikkelen voor het aansturen van IP gebaseerde beveiligingsapparatuur (ONVIF, 2015). Voor het besturen van beveiligingscamera’s is de ONVIF Profile S standaard (ONVIF, 2011) gedefinieerd. Hierin is onder andere een generieke methode beschreven waarmee PTZ camera’s aangestuurd worden. De standaard omvat vele functies, zoals device discovery, configureren en uitvragen van camera’s, authenticatie enzovoorts.

Als communicatiemedium voor ONVIF wordt SOAP gebruikt. SOAP staat voor ‘Simple Object Access Protocol’. Het is een XML achtige structuur welke over HTTP (web-services) getransporteerd wordt. Een voorbeeld van een ONVIF (SOAP/XML) commando is te zien in “Bijlage C – Voorbeeld van ONVIF commando”. Hierin staat een voorbeeld van een ‘Absolute Move’ commando.

Het besturen van PTZ camera’s met ONVIF commando’s gebeurt op een bijzonder robuuste en generieke wijze. Het grote nadeel van het gebruik van ONVIF commando’s is de traagheid. Gemiddeld is elk ONVIF commando 1000 bytes groot. Bij gebruik van een video encoder wordt het commando vanaf de encoder via een seriële verbinding van maximaal 9600 bits per seconde naar de camera verstuurd. Dat betekent dat elke aansturing er minimaal één seconde over doet om bij de camera aan te komen. Daarna moet het commando nog door de camera verwerkt worden. Uit observaties (met gebruik van WireShark captures) blijkt dat er ongeveer 1,5 seconde zit tussen het geven van het commando en een reactie in het beeld.

Aansturen via CGI van de camera web pagina

Een aanzienlijk snellere manier is door de PTZ camera aan te sturen via de URI (URL) naar de ingebouwde webserver. De enige data die verstuurd wordt is een HTTP GET commando met de parameters voor de aansturing. Een voorbeeld URI van de aanvraag is:

http://192.168.178.47/api/ptz.cgi?PTZNumber=1&PanSpeed=14&TiltSpeed=56&ZoomSpeed=-16

Het versturen van deze aanvraag kost minder dan 10ms. Het type commando dat zo verstuurd wordt is een ‘Continuous Move’. Een ander bruikbaar commando dat zo verstuurd kan worden is een “Go to Preset”. Uiteraard moet deze preset dan wel ingesteld zijn. Een preset kan gebruikt worden in plaats van een ‘Absolute Move’ om een beginpositie in te stellen.

Deze methode van aansturen via CGI is bruikbaar voor alle Siqura camera’s. De bij dit project gebruikte Siqura Eve One encoder maakt gebruik van een vergelijkbare aansturing, maar via een andere URI.

Voor details zie “Bijlage D – PTZ CGI besturing”, voor de bron hiervan zie “PTZ Camera and Codec Programming Interface v1.5” van (TKHSecurity, 2015).

Aansturen via de API van de camera leverancier

De meest directe aansturing kan via de API (‘Application Programming Interface’) welke door de leverancier van de camera geleverd wordt. Deze communiceert via de meest directe route met de camera hardware. De API bestaat uit een software bibliotheek geschreven voor een bepaalde programmeertaal en een bepaald operating system.

Bij Siqura is de CGI interface de API. Zie “Bijlage D” .

Keuze van de methode van aansturing

Het antwoord op de vraag *“Wat is de meest geschikte methode om een PTZ camera aan te sturen?” is:*

Vanwege de snelheid, soepelheid in beweging en eenvoud van aansturing is ervoor gekozen om de PTZ camera aan te sturen met ‘Continuous Move’ commando’s via de CGI interface.

5.2 Regelaar voor de PTZ motor aansturing

Dit hoofdstuk geeft antwoord op de deelvraag: *“Wat is de meest geschikte regelaar om de PTZ camera in de pas te laten lopen met de tracker?”* en de bijbehorende sub-deelvraag: *Is een lage orde PID regelaar voldoende of is een hogere orde regelaar, of zelfs een niet lineaire regelaar nodig”*

Naar onderstaande twee regelsystemen is literatuuronderzoek gedaan:

- PID
- Fuzzy Logic

De opdrachtgever ging uit van PID. Van Fuzzy Logic was mij bekend dat het zeer geschikt is voor digitale systemen. Na raadplegen van een collega (ervaringsdeskundige) gaf deze aan dat Fuzzy Logic het eenvoudigste te implementeren is, maar “Dynamische PID” het snelst en meest nauwkeurig is.

Meet- en regeltechniek is geen onderdeel van de opleiding Technische Informatica, daardoor is de technische diepgang van dit deel van het onderzoek beperkt. De uiteindelijk gekozen oplossing is niet PID of Fuzzy Logic, maar is ‘rule-based’ en heeft grote overeenkomsten met Fuzzy Logic. De uitleg van PID en Fuzzy Logic wordt gegeven om de uiteindelijke keuze te motiveren.

Voor de uitleg van beide systemen is het boek “Mechatronics – electronic control systems in mechanical and electrical engineering” (Bolton, 2008) geraadpleegd. De uitleg van PID is terug te vinden in hoofdstuk 15, Fuzzy Logic in hoofdstuk 16.3.2. Daarnaast is in het boek gekeken naar andere mogelijke meet- en regelsystemen.

PID regelsysteem

Een PID regelsysteem bestaat uit een combinatie van de componenten Proportioneel, Integrerend en Differentiërend. De meest gebruikte combinaties zijn P, PI, PD en PID.

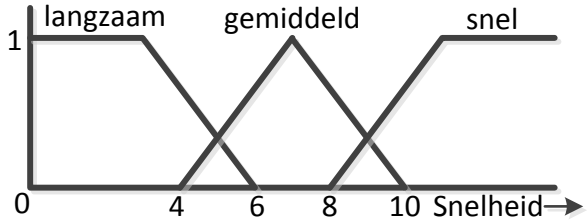
Het proportionele deel zorgt dat een uitgangssignaal evenredig verloopt met het ingangssignaal. Het proportionele deel heeft het gedrag van een veer of slinger; het uitgangssignaal slingert rond de ingestelde waarde totdat het deze waarde bereikt heeft, afhankelijk van hoe hard er aangestuurd wordt.

Het integrerende deel kijkt naar het verleden en geeft de som van de recente ingangssignalen als uitgangssignaal; het heeft een dempende werking. Het differentiërende deel ‘anticipeert’ en stuurt bij snelle verandering van het ingangssignaal het uitgangssignaal harder of verder bij dan nodig is om de eindwaarde te bereiken.

Een PID regelsysteem moet ingeregeld worden om te bepalen hoe een proces moet worden bijgestuurd. Voor elk van de componenten P, I, D moet de instelling bepaald worden. Het systeem wordt ingeregeld aan de hand van onder andere responstijden die specifiek voor een bepaald systeem zijn. Een slecht ingeregeld systeem kan instabiel worden en gaan oscilleren.

Fuzzy Logic regelsysteem

Een Fuzzy Logic of Fuzzy Control regelsysteem is een ‘rule-based’ systeem en werkt op basis van meerdere logische niveaus oftewel Fuzzy Sets (zie Figuur 12Figuur 12 Voorbeeld van een Fuzzy Logic ingangsbegrippen bestaande uit meerdere niveaus met overlap).



Figuur 12 Voorbeeld van een Fuzzy Logic ingangsbegrippen bestaande uit meerdere niveaus met overlap

De karakteristiek van een fuzzyset heeft de vorm van een driehoek of een trapezium. Tussen de fuzzy sets zit overlap zodat er overgangsgebieden zijn waarbij de uitgangswaarde de som is van in welke mate de ingangswaarde zich in elke fuzzy set bevindt. Per fuzzy set is gedefinieerd wat de vereiste actie (‘rule’) is.

Bevindingen uit experimenten

Onderstaande tekst loopt vooruit op hoofdstuk 5.4 (Werking van de PTZ regelaar) door aan te geven voor welk systeem is gekozen. In hoofdstuk 5.4 wordt uitgelegd met welke experimenten het gedrag is vastgesteld.

In Hoofdstuk 5.3 wordt uitgelegd hoe karakteristieken van PTZ camera’s bepaald worden en hoe het systeem ingeregeld wordt.

In eerste instantie klonk PID als de logische keuze voor een regelaar. PID is snel en accuraat mits goed ingeregeld. Om PID in te kunnen regelen is een model of formule nodig en deze ontbrak. Om een beter beeld van het gewenste gedrag te krijgen is eerst met de Pelco PTZ cameraopstelling geëxperimenteerd.

Het lukte om de camera proportioneel aan te sturen. In een stabiele situatie was de aansturing traag in de zin dat het centreren van een object in beeld erg traag maar gecontroleerd verliep. Als het object gecentreerd was, werden de PTZ motoren permanent aangestuurd waardoor trilling in het beeld ontstond. Bij proportioneel sterker aansturen ontstond al snel instabiliteit en oscillaties waarbij het beeld rond het centrum heen en weer sprong. Om het object snel en gecontroleerd te centreren zou een PID regeling nodig zijn.

Een volgend experiment was om de aansturing van de motoren te stoppen zodra het object in een 10% regio midden in het beeld gecentreerd was. Dat leverde het gewenste resultaat op in de vorm van een rustig beeld. De PTZ camera kon met deze ‘dode zone’ toevoeging harder proportioneel aangestuurd worden zonder oscillaties, maar het bleef traag. Met het in gebruik nemen van de ‘dode zone’ is echter de eerste stap gezet naar een ‘rule-based’ systeem.

Een andere bevinding was dat er maximale PTZ snelheden waren waarbij de tracker het object nog kon volgen. Hier was alleen mee om te gaan door hier beperkingen op te zetten, oftewel meer ‘rules’ zodat tracking niet afgebroken zou worden.

In een latere fase is met de Siqura camera geëxperimenteerd waarop de uiteindelijke demo gegeven moet worden. De aansturing van deze camera verloopt niet geleidelijk zoals bij de

Pelco, maar stapsgewijs waarbij de snelheidsverschillen tussen de stappen behoorlijk groot zijn. Daar is beter mee om te gaan als er gewerkt wordt met een ‘rule-based’ systeem.

Een andere factor die er voor pleit een ‘rule-based’ systeem te kiezen is de vertraging die door de videostream vanaf de camera veroorzaakt wordt. Deze kan behoorlijk variëren en is van diverse factoren afhankelijk. Zie hoofdstuk 6 voor onderzoek, bevindingen en oplossingen.

Keuze voor een regelsysteem

Tabel 3 toont een vergelijking tussen Fuzzy Logic en PID regelaars. Tabel 3 komt uit (Baeten, 2003). In (Reus, 1994, p. 26) en (STOWA, 1997) zijn dezelfde argumenten terug te vinden.

Het uiteindelijke regelsysteem is een ‘rule-based’ systeem, maar is strict genomen geen Fuzzy Logic omdat er geen overgangssituaties gedefinieerd zijn. Uit experimenten bleek dat er geen overgangssituaties nodig zijn. Dit ‘rule-based’ systeem heeft (in ieder geval voor Tabel 3) dezelfde eigenschappen als Fuzzy Logic.

Na de experimenten is het gedrag van het systeem bekend. Aan de hand van onderstaande bevindingen kan de keuze in het voordeel voor het ‘rule-based’ systeem bevestigd worden:

- Het betreft geen lineair systeem
- Er is niet op eenvoudige wijze een wiskundig model af te leiden (vanwege de verschillende typen PTZ camera’s).
- Het gedrag van het systeem kan in eenvoudige regels omschreven worden.
- Door het gebruik van de ‘rules’ is het eenvoudig aan te passen voor andere PTZ camera’s.
- “*Een fuzzy regeling is meer inzichtelijk voor niet-regeltechnici dan veel andere regelingen omdat de regeling beschreven is door middel van linguïstische kennisregels. Het veranderen van de regeltechnische instellingen is daarom voor niet regeltechnici een minder abstracte aangelegenheid dan de verandering van P-, I- en D-constanten”* (STOWA, 1997, p. 18)

De redenen waarom PID minder geschikt is voor het regelen van de PTZ aansturing zijn:

- PID kan niet goed overweg met niet-lineaire systemen
- Het gedrag van het systeem is niet op eenvoudig wijze te vatten in een enkele formule
- PID moet bij het gebruik van andere componenten opnieuw ingeregeld worden. Hier is kennis en ervaring van regeltechniek voor nodig.

	Fuzzy Logic (‘rule-based’)	**PID**
Mathematisch model van te regelen systeem	Niet noodzakelijk	Wenselijk
Eigenschappen v.h. systeem	Lineair, Niet-lineair	Hoofdzakelijk lineair, Moeilijker bij niet-lineair
Aantal in- en uitgangen	1/1 tot multi/multi	Essentieel 1/1 (single-input single-output systeem)
Typische toepassingsgebieden	Systemen waarbij ervaring, kennis en intuïtie belangrijke elementen zijn	Lineair single-input single-output systeem
Regelstructuur	Afzonderlijke regels	Eén enkele formule die het volledig bereik omvat
Eenvoud van de regels	Eenvoudig (natuurlijke taal)	Ervaring en kennis nodig
Aanpassing van de regels	Op basis van ervaringen met het te regelen systeem	Op basis van systeemkennis

Ontwerpprocedure	Kwalitatief op basis van natuurlijk taalgebruik	Kwantitatief op basis van logische formules
Theoretische analyse (stabiliteit, betrouwbaarheid, robuustheid)	Niet berekenbaar, wel simuleerbaar	Berekenbaar.
Robuustheid (tegen veranderingen van het systeem)	Zeer goed	Matig

Tabel 3 Vergelijking tussen Fuzzy Logic en PID (Baeten, 2003)

Het antwoord op de deelvraag: *“Wat is de meest geschikte regelaar om de PTZ camera in de pas te laten lopen met de tracker?”* en de bijbehorende sub-deelvraag: “*Is een lage orde PID regelaar voldoende of is een hogere orde regelaar, of zelfs een niet lineaire regelaar nodig”* is:
De meest geschikte regelaar voor de PTZ tracker is een niet-lineaire, ‘rule-based’ regelaar. Deze is gekozen omdat deze eenvoudiger aan te passen is voor niet-regeltechnici.

Aanbeveling voor vervolgonderzoek

Een aanbeveling voor vervolgonderzoek is om alsnog te onderzoeken in hoeverre PID betere prestaties levert door een deskundige op het gebied van meet- en regeltechniek.

De ‘rule-based’ regelaar reageert op de bewegingsrichting en ijlt daardoor na zodat het object tijdens beweging niet gecentreerd is. Met een PID regelaar kan mogelijk bereikt worden dat het object meer in het beeld gecentreerd wordt doordat deze anticipeert. Dat betekent tevens dat tracking gelijkmatiger verloopt.

5.3 Normaliseren van PTZ snelheidswaarden

Het doel van dit deel van het onderzoek is om een methode te beschrijven waarmee een ContinuousMove commando op verschillende PTZ camera’s hetzelfde resultaat geeft.

De waarden van Pan speed, Tilt speed en Zoom speed die bij de onderzochte camera’s opgegeven kunnen worden liggen tussen de -100 en 100. Deze getallen hebben geen eenheid en hebben met verschillende camera’s verschillende resultaten.

De hier beschreven methode drukt Pan en Tilt snelheden uit in graden per seconde.

Uitgevoerde experimenten
De experimenten zijn uitgevoerd met de Core2Duo PC. Hiermee zijn de karakteristieken van de Pelco III en van de Siqura HSD820H2 camera vastgesteld. Beide camera’s worden op vergelijkbare wijze via CGI (HTTP GET commando) aangestuurd (zie “Bijlage D”).
De resultaten van alle metingen staan in “Bijlage D”.

Metingen voor ‘Pan’ uitvoeren:
De vermelde PTZ camera’s kunnen doorlopend 360 graden ronddraaien.
De werkwijze is de camera via de webpagina op een referentiepunt richten. Voer URI in met de gewenste Pan snelheid en druk ‘Enter’. Meet met een stopwatch de begin en eindtijd van de beweging totdat de camera weer op het referentiepunt gericht is. Herhaal metingen en bepaal het gemiddelde om meetfouten te verkleinen. Als de meettijd te klein is voor 1 omwenteling dan meerdere omwentelingen om meetfout te verkleinen.

Een andere, meer nauwkeurige meetmethode is een videobestand maken van de camerabeweging, bijvoorbeeld door video op te nemen via de webpagina van de camera of met een screen recorder.
De begin en eindtijden van de beweging kunnen bepaald worden met gebruik van het programma ‘melt’ (Meltytech, LLC., 2015).

Het programma ‘melt’ geeft het framenummer weer. Door het eindframe van het beginframe af te trekken kan de tijd berekend worden. Bij 25fps is de tijd per frame: 1000 / 25 = 40ms.

Metingen voor ‘Tilt’ uitvoeren:
Er kan een vergelijkbare werkwijze gehanteerd worden als bij ‘Pan’. De ‘tilt’ beweging heeft een kortere tijdsduur aangezien er een maximaal aantal graden beweging mogelijk is. Bepaal eerst hoe de ‘tilt’ hoek van de camera gemeten kan worden: Bij de Pelco wordt de hoek in het beeld weergegeven. Bij de gebruikte Siqura HSD820H2 is deze op te vragen met de URI:

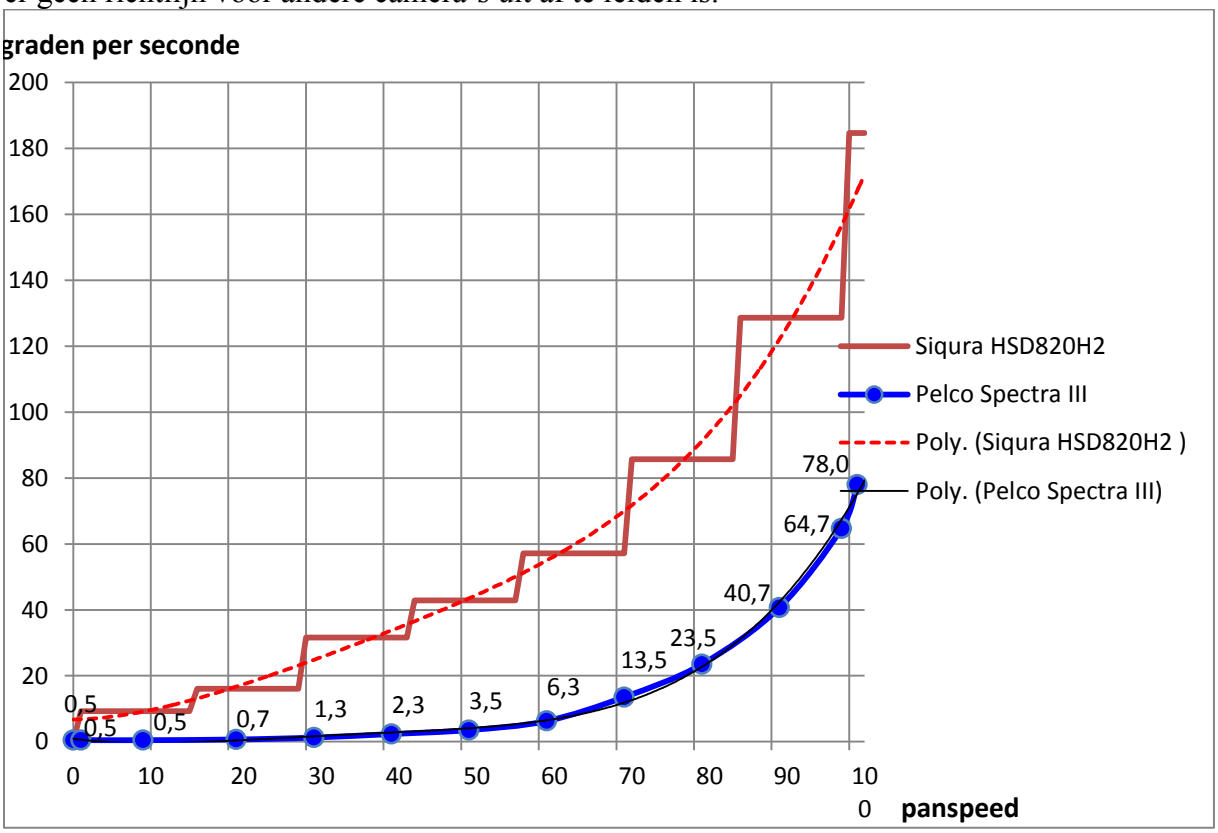
```
http://<servername>/cgi-bin/com/ptz.cgi?query=position
```

Pan en Tilt snelheden

Omdat er geen gebruik gemaakt wordt van een gestandaardiseerde methode, is het belangrijk om de karakteristiek van de PTZ besturing te kennen. Deze karakteristiek moet per cameratype éénmalig vastgesteld worden.

Dit is gebeurd voor zowel de Pelco als de Siqura camera (zie Figuur 13). In beide gevallen is het niet-lineaire verloop duidelijk te zien. Bij de Siqura camera wordt de aansturing wordt zelfs stapsgewijs geregeld. De rode stippellijn (Poly. Siqura) is een benadering van de karakteristiek als vloeiende lijn.

Dezelfde metingen zijn ook verricht voor Tilt speed. Zie “Bijlage D” voor de grafieken en tabellen van Pan en Tilt speed. Er zijn geen karakteristieken gemaakt van Zoom speed omdat er geen richtlijn voor andere camera’s uit af te leiden is.



Figuur 13 Besturingskarakteristieken van gebruikte camera's. Zie ‘Bijlage D’ voor een tabel met waarden.

Experimenten met Zoom snelheden

Bij de Siqura HSD820H2 camera is er geen onderscheid in snelheid bij de ‘Continuous Move’ zoom. Welke waarde ook gekozen wordt, de snelheid is altijd hetzelfde. Het snelheidsverloop is niet lineair en tijdens de zoom actie gaat de camera uit focus en wordt het beeld wazig. De ‘Relative Move’ variant (rzoom) levert goede resultaten op bij waarden tussen +/-4 en 32. Dit is de voorkeursmethode voor aansturing van Zoom.

De aansturing van de zoomlens bij de Pelco camera is erg onnauwkeurig. Van een snelheids-bereik van 0 tot 100 voor inzoomen en 0 tot -100 voor uitzoomen zijn alleen de waardes -1 (uitzoomen), 0 (stilstand) en 1 (inzoomen) bruikbaar. Elke hogere waarde is te snel.

Stabiliteit in het zoommechanisme is aangebracht door hysterese te gebruiken.
Er wordt ingezoomd tot een object 35% van de hoogte of breedte van het scherm vult. Door de traagheid in de aansturing (doorschieten) vult het object dan circa 50% van de hoogte of breedte van het scherm. Er wordt uitgezoomd als een object meer dan 70% van de hoogte of breedte van het scherm beslaat en stopt bij circa 50% hoogte of breedte.

De genoemde percentages zijn experimenteel bepaald en gelden alleen voor de Pelco camera welke alleen voor demo-doeleinden gebruikt wordt. Bij elk ander type camera dienen opnieuw proefondervindelijk goede zoomniveau’s bepaald worden.

Bevindingen met betrekking tot Zoom aansturing

Op 17 december was er in de haven van Rotterdam een demodag van het APPS project. Hier vond een demonstratie plaats van de classificatie van schepen. De classificatie vindt plaats op de beelden die afkomstig zijn uit de PTZ camera. In die situatie levert de PTZ camera beelden met een resolutie van 2048x1536 beeldpunten.

Tijdens de demonstratie zijn beelden getoond waarin schepen 50% tot 70% van de breedte van het beeld besloegen. De afmetingen in het beeld zijn dus niet kritisch. Het schip moet echter wel in zijn geheel in het beeld passen, te ver ingezoomd is voor de classificatie niet bruikbaar.

Een andere reden voor deze 50% tot 70% zoomfactor is dat de afmetingen van de bounding box vaak niet accuraat zijn, dit wordt veroorzaakt door de detector. Dit is in hoofdstuk 4.4 uitgelegd. De bounding box is dan kleiner dan het object dat het eigenlijk dient te omsluiten.

Te ver inzoomen heeft een ander nadeel: het bemoeilijkt tracking. De snelheid van aansturen (Pan en Tilt) is al snel te hoog waardoor het lastig is om een schip in het beeld te centreren. Belangrijker nog: Als de bounding box rond een schip dat door de tracker gevolgd wordt de buitenrand van het beeld raakt, dan stopt tracking. Een schip moet dus volledig in beeld blijven.

Het antwoord op de deelvraag “*Hoe kunnen we het optimale zoomniveau bepalen zodat de tracker het object niet kwijtraakt?”* is:

Het zoomniveau is niet kritisch, een zoomniveau van 50 tot 70% wordt aanbevolen. Het te volgen object dient altijd volledig in beeld te blijven en geen beeldranden te raken daar anders de tracking stopt.

Focus snelheid

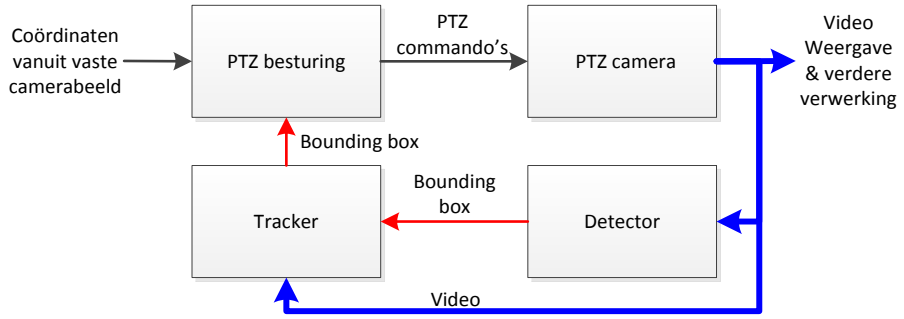
Een algemeen probleem is dat de focusafstand van de lens verandert als er snel ingezoomd wordt (in mindere mate bij uitzoomen). De focusafstand wordt automatisch door de camera zelf geregeld. Zoom moet dus langzaam aangestuurd worden. Als een getracked object uit focus valt, dan stopt de tracking en PTZ aansturing.

Aanbevelingen voor vervolgonderzoek

- Onderzoek hoe automatische kalibratie van PTZ snelheden gedaan kan worden zodat eenvoudig andere camera’s gebruikt kunnen worden zonder dat een gebruiker of installateur het systeem moet inregelen.
- Onderzoek hoe automatisch de vertraging van de PTZ aansturing gemeten kan worden. Hiermee kan de regelaar automatisch ingeregeld worden. Dit verbetert de snelheid en nauwkeurigheid van de PTZ aansturing.

5.4 Werking van de PTZ regelaar

Voor het inregelen van een regelsysteem, ongeacht of het PID of Fuzzy Logic is (zie hoofdstuk 5.2) dient het gedrag van het systeem bepaald te worden. In Figuur 14 is de regellus weergegeven waarvan het gedrag bepaald is door te experimenteren.



Figuur 14 Blokschema PTZ auto tracker

De PTZ besturing krijgt een locatie op en richt daar de PTZ camera op. De detector wordt gestart, detecteert een object en start de tracker door een bounding box van het object aan de tracker door te geven. Daarna komt de detector er niet meer aan te pas.

De tracker volgt het gemarkeerde object en geeft als resultaat een bounding box van de nieuwe locatie van het object door aan de PTZ besturing. De PTZ besturing (de regelaar) vertaalt de bounding box naar (x, y) coördinaten en de juiste commando's voor de PTZ camera. De PTZ camera geeft het nieuwe camerabeeld door aan de tracker.

Tot zover de te voorspellen werking van de regelaar. Het gewenste gedrag is bepaald door experimenteren. De gedachte daarbij is dat een gedetecteerd object altijd in het beeld gecentreerd dient te worden.

<u>Uitgevoerde experimenten</u>

De experimenten zijn uitgevoerd met de Core2Duo PC en de Pelco camera.
Alle experimenten beginnen op dezelfde manier. Benodigd is een duidelijk te detecteren en te volgen object met voldoende detail en contrast. Hier is een Rabo Reader gebruikt. Om goed gevolgd te worden dienen de afmetingen van het object in het beeld niet te groot of te klein te zijn. Een richtlijn om te beginnen is dat het object 20% van de hoogte van het beeld beslaat.

Het object wordt met de hand in het blikveld van de camera gebracht. Vervolgens wordt de detector gestart en het object in beweging gebracht. Onderzocht is het gedrag bij de volgende bewegingen:

- Detectie starten met beweging richting het centrum van het beeld. Na detectie het object stil houden. De PTZ camera dient het object te centreren.
- Detectie starten met beweging richting een beeldrand en de beweging voortzetten. De PTZ camera dient het object te volgen met als doel om het in beeld te centreren.

Een volgend niveau is omgaan met snelheidsverschillen:

- Detectie starten dichtbij een beeldrand met beweging richting het centrum van het beeld. Na detectie object stil houden. De PTZ camera dient het object ***zo snel mogelijk*** te centreren (zonder oscillatie rond het centrum van het beeld).
- Terwijl de tracking actief is het object sneller richting een beeldrand bewegen en de beweging voortzetten. De PTZ camera dient zijn snelheid te verhogen naarmate het object de beeldrand nadert en het object te volgen met als doel om het ***zo snel mogelijk*** in beeld te centreren.

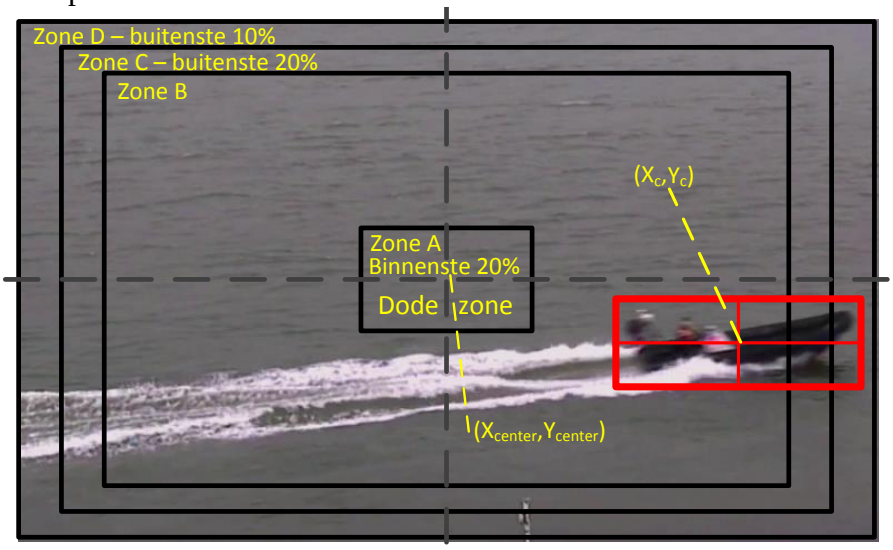
Dit heeft uiteindelijk geleid tot de rule-set hieronder waarin verschillende zones zijn gedefinieerd..

Uitleg van de gedefinieerde rule-set

De doelstelling van de regelaar is om te zorgen dat een object dat met de tracker gevolgd wordt, midden in beeld gehouden wordt. Er wordt bijgestuurd door een snelheid voor Pan en Tilt op te geven.
Het Zoom niveau wordt afgeleid van de grootte van de bounding box. Er wordt alleen ingezoomd als het object gecentreerd is. Over de Zoom later meer, de wijze van aansturing verschilt per camera – hier is momenteel geen generieke uitspraak over te doen.

Zodra tracking niet meer mogelijk is, zoomt de camera uit. Hierdoor is het niet meer mogelijk om de tracking te hervatten omdat de signatuur die de tracker van het object heeft niet meer overeenkomt met het beeld. Er moet dan opnieuw detectie plaatsvinden. Bij de demo is her-detectie niet vereist. Het kan automatisch, maar dat is deel van een toekomstig ontwerp.

Er is gekozen voor PTZ aansturing met constante snelheden omdat dit het systeem eenvoudig, voorspelbaar en daardoor robuust houdt.



Figuur 15 Zone-indeling van het beeld. In dit voorbeeld bevindt de bounding box rond het object zich in de buitenste zone. Daardoor wordt het object met de hoogste snelheid gevolgd.

Het beeld is ingedeeld in een viertal zones:

- **Zone A**. De “dode zone”. Het is altijd de bedoeling dat een object hierin wordt gecentreerd. Zodra het centrum van de bounding box hierin ligt stopt het aansturen van de Pan en Tilt motoren (snelheid 0). Zone A is de enige zone waarin wordt ingezoomd omdat het beeld hier het meest stabiel is.
  Merk op: Een object wordt niet volledig gecentreerd! Het aansturen van Pan en Tilt stopt als de zone bereikt is.
- **Zone B**. Als het centrum van de bounding box zich hierin bevindt, worden de Pan en Tilt motoren met constante snelheid zo aangestuurd dat het middelpunt van het object / de bounding box gecentreerd wordt in zone A.
- **Zone C**. is voor objecten die zich richting de rand van het beeld verplaatsen. Als de buitenrand van de bounding box binnen deze zone valt, dan wordt de snelheid van de Pan en Tilt motoren met een hogere constante snelheid aangestuurd dan in zone B zodat de bounding box weer in zone B komt te liggen.
- **Zone D**. Als zone C, maar met de hoogste constante snelheid.

Hoe dichter een object bij de rand van het beeld komt hoe hoger de snelheid van Pan en Tilt. Als de bounding box de buitenrand van het beeld raakt, dan stopt de tracking – ook al is het object nog zichtbaar. Vandaar het versnellen in de zones C en D.

Fysieke aansturing en fysiek gedrag

Bij de demo is gekozen om de PTZ aansturing met vaste intervallen te doen. De tracker werkt de locatie van de bounding box van het te volgen object elk frame bij, de PTZ aansturing gebeurt 5x per seconde; om de 5 frames.

De keuze hiervoor is omdat dit een vloeiende beweging in de aansturing geeft en minder berekeningen vraagt dan wanneer dit elk frame gebeurt. Het effect is dat objecten snel en met een vloeiende beweging gevolgd worden. De vloeiende beweging ontstaat doordat het mechanische deel van de camera tijd nodig heeft om zich aan een nieuwe snelheid aan te passen.

Bepalen van Pan en Tilt snelheden

De waarden van Pan en Tilt zijn proefondervindelijk bepaald:
Zone B: 16 m/s, Zone C: 32 m/s, Zone D: 43 m/s.

De maximale waarde voor Pan is gebaseerd op de maximale snelheid van rotatie die de camera en de tracker kunnen verwerken. De beperkende factor is niet de snelheid van de motoren, maar de ‘motion-blur’ die de camera ervaart tijdens snelheidsovergangen waardoor objecten onscherp worden. De mate van ‘motion-blur’ is afhankelijk van een aantal factoren; Het gebruikte type camera, de hoeveelheid beschikbaar licht en de sluitertijd of belichtingstijd van de camera.

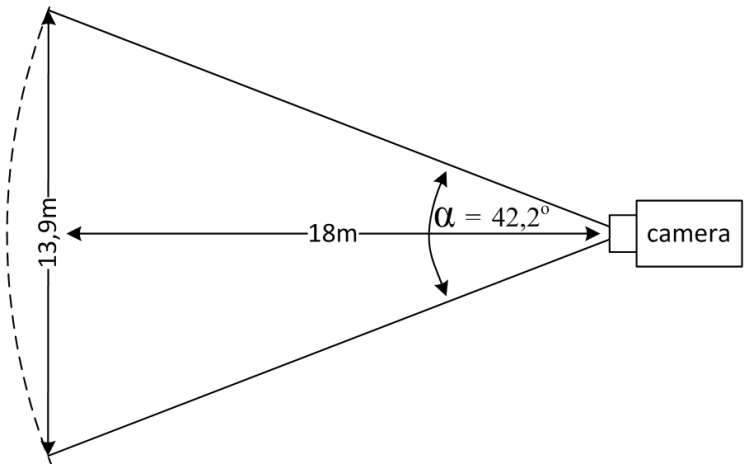
Als vooraf bekend is wat de maximale snelheid en de afstand van objecten ten opzichte van de camera is (zoals in Zone D), dan kan het aantal graden per seconde berekend worden. Voorbeeld: De PTZ camera staat in het pand van Siqura op de 1^{e} etage voor het raam en heeft daarmee een afstand van 18m tot de rijstrook voor verkeer dat van links komt. Met deze gegevens kan eenvoudig uitgerekend worden wat de maximale Pan speed is, zie Figuur 16.

We berekenen hoeveel graden er afgelegd worden door een voertuig in 1 seconde:

De maximum snelheid op de weg is 50km/h. In meters per seconde is dit 50 / 3,6 = 13,9 m/s. We werken met rechte hoeken, dus met de helft van de driehoek uit Figuur 16:

Overstaande: 13,9 / 2 = 6,95 m
Aanliggende: 18m
Tan(α) = overstaande / aanliggende => Tan(α) = 6,95 / 18 = 0,386
Inverse tangens: $\text{Tan}^{-1}(\alpha) = \text{Tan}^{-1}(0{,}386) = 21{,}1$ graden
De volledige hoek is 21,1 * 2 = 42,2 graden.

De maximale rotatiesnelheid van de camera bij een auto bij 50km/h op 18m afstand wordt dan 42,2 ^{o}/s. Let op: dit is geen constante snelheid. De hoogste snelheid is recht voor de camera. Deze snelheid is een indicatie. De bruikbaarheid dient experimenteel bevestigd te worden.



Figuur 16 Berekenen van Pan speed

Nog wat voorbeelden ter indicatie (zonder berekening):

- Een auto op de andere rijstrook (21m) met 13,9m/s heeft een snelheid van 40 ^{o}/s.
- Een auto op 18m afstand met 60km/h (16,7m/s) heeft een snelheid van 46,8 ^{o}/s.

Herdetectie

Er resteert nog één deelvraag: “*Hoe kunnen we na kwijtraken van het object deze weer terugvinden?* ”. Aangezien dit project niet verder gaat dan een demo is dit niet volledig uitgewerkt. Er kunnen echter wel aanbevelingen gedaan worden.

Het antwoord op de deelvraag is:

Tracking stopt als het object niet meer zichtbaar is of als het de beeldrand raakt. Mogelijk is een object tijdelijk niet zichtbaar door occlusie, bijvoorbeeld een paal in beeld. Als de tracker het object tijdelijk kwijtraakt en daarna weer herkent gaat tracking verder.

- Maak een keuze na hoeveel frames tracking afgebroken wordt (in de demo: 50 frames).
- Bepaal het gewenste gedrag als tracking stopt (in de demo: uitzoomen en afsluiten). Andere keuzes, waarbij ook gekozen moet worden gedurende hoe lang geprobeerd moet worden, zijn:
    - Niet uitzoomen en de beweging voortzetten aan de hand van de laatste bewegingsvector.
    - Herdetectie starten, met hierbij de keuze: gebeurt dit vanuit tracking of vanuit “Parameter Intrusion Detection”? *(zie “Object selectie” in Figuur 2, pagina 2).*

6 Instellen van camerastreams

Dit hoofdstuk geeft antwoord op de vraag “*Hoe kan een videostream van een camera optimaal ingesteld worden voor het gebruik met tracking en PTZ aansturing?”.*

Deze vraag wordt beantwoord door te experimenteren (metingen te verrichten) en aan de hand van literatuuronderzoek

Er zijn twee soorten problemen te onderkennen:

- Latency (vertraging) van de video data
- Verminking van video data

Bij de PTZ camera op het parkeerterrein bleek een lastig waar te nemen probleem te bestaan. Deze camera had een dermate grote latency dat het onmogelijk was om vanuit tracking PTZ aansturing te doen.

Tijdens het onderzoek op het kantoor van Siqura traden in de loop van het project verschillende problemen op met videostreams. Met name H.264 streams raakten bij gebruik van de Core2Duo PC bij elke resolutie hopeloos verminkt.

Op 17 december was er in de haven van Rotterdam een demo dag van het APPS project. De hoge resolutie H.264 stream (2048*1536*30fps) werkte feilloos op de Core2Duo PC met de ptztracker demo applicatie. Dit heeft geleid tot een nieuw stuk onderzoek over video streams. De problemen met H.264 streams hebben een dermate grote impact dat het de moeite waard is om dit hoofdstuk er aan te wijden.

Voor dit deel van het onderzoek zijn een aantal experimenten uitgevoerd:

- Meten van de latency (vertraging) van de videostream.
- Bepalen wanneer er datacorruptie optreedt en bepalen van de optimale instellingen van de videostream met beste beeldkwaliteit (balans tussen resolutie en kwaliteit).

Alle experimenten zijn eerst uitgevoerd met de Core2Duo PC en daarna met de i5 PC.

Meten van Latency

Bij een vast opgestelde camera is dit lastig waar te nemen. Bij de PTZ camera op het parkeerterrein gaat er 4 a 5seconden versneld afspelen van de video stream aan vooraf, daarna ziet de weergave er normaal uit.

Dit probleem is gedetecteerd doordat de tijd in het camerabeeld 4,2 seconden verder was dan de tijd op de PC terwijl beiden met dezelfde time-server gesynchroniseerd werden.

Door de 4,2s vertraging is het onmogelijk om PTZ besturing vanuit tracking te doen. Tegen de tijd dat de besturing reageert is het object al lang verdwenen. Om die reden wordt er geen demo gedaan met de camera op het parkeerterrein maar met de Pelco camera.

In een H.264 videostream zitten relatieve timestamps. Deze geven het tijdstip weer sinds de stream gestart is en dienen ter synchronisatie. Er is geen bruikbare informatie over latency uit af te leiden. Latency is als volgt te meten:

- Visueel:
  - Door op de PC een timer of stopwatch te laten draaien en de camera te richten op het beeldscherm. Vergelijk de PC tijd met de videotijd.
  - Door op de camera tijdweergave aan te zetten, de camera en de PC te synchroniseren met dezelfde tijdserver (beter is: de camera synchroniseren met de tijd van de PC) en de tijd tussen camerabeeld en PC te vergelijken.
- Met behulp van het commando: `mplayer -benchmark <videostream>`
  eventueel met optie `-nocache` (dit werkt niet met Eve-One encoder)

Bij de visuele methode zijn de volgende handelingen uitgevoerd:

- Weergave van seconden aanzetten bij de PC tijd.
- Camerabeeld weergeven met een programma dat geen buffering en vertraging kent: Het bovengenoemde `‘mplayer’` en/of een eenvoudig OpenCV programma alleen bestaande uit een ‘capture’ en een ‘imshow’.
- Terwijl beide tijden in beeld zijn een aantal seconden beeld opnemen met een screenrecorder of een extern opname apparaat.
- Bekijk het filmpje met het programma ‘**`melt`**’ (Meltytech, LLC., 2015).

Het programma ‘**`melt`**’ geeft het framenummer weer. Door het eindframe van het beginframe af te trekken kan de tijd berekend worden. Bij 25fps is de tijd per frame: 1000 / 25 = 40ms.

De latency van de HSD820H2 PTZ camera op het parkeerterrein is met deze methode bepaald op 4,2s. In “Bijlage G” zijn de metingen te zien voor de Pelco camera via de Eve One encoder. Deze latency is 230ms bij een MJPEG stream. Met deze latency is tracking nog zeer snel.

Bepalen wanneer er datacorruptie optreedt

Onderstaande problemen zijn waargenomen zowel met de Core2Duo PC als de i5 PC.

Symptomen bij problemen met videostreams:

- Datacorruptie waardoor het beeld vanaf de onderkant verminkt raakt.
- Datacorruptie onzichtbaar in de video maar met foutmeldingen van het programma, tracking of detectie werkt niet- of niet goed.

Zodra tijdens het lezen van een videostream de CPU belasting de 100% raakt, gaat het decoderen van de videostream mis. Er blijft wel beeld zichtbaar, maar en kunnen er geen bewerkingen meer op de videodata gedaan worden.

Bij zelfgeschreven C++ code maakt het niet uit of voor de videoafhandeling de Siqura code gebruikt wordt of de OpenCV 3.0.0 / FFmpeg code.

Bij Siqura code treedt geen verminking op, is er niets zichtbaar maar ‘bevriest’ de stream na enkele seconden. De ‘Simulation Source Manager’ maakt gebruik van ***avconv*** voor het converteren van de stream en schrijft logging naar een tijdelijk bestand in de map waar de executable staat. Dit is te gebruiken voor foutzoeken.

OpenCV 3.0.0 blijft video weergeven, maar met datacorruptie en uiteindelijk een onbruikbare stream. Foutmeldingen worden als rode tekst weergegeven (‘decode errors’) in het terminal venster waarin de applicatie draait.

De problemen zijn het ergste bij het gebruik van een relatief trage PC zoals de Core2Duo PC en aanzienlijk minder bij een i5 of snellere PC. Bij weergave van de streams in VLC player wordt in de meeste gevallen de stream correct weergegeven.

Oorzaken en oplossingen van latency en datacorruptie

De resterende tekst van dit hoofdstuk is verzameld uit (Axis, 2015) en (FFmpeg, 2015).
Er zit een grote overlap in beide bronnen, maar samen geven ze een zeer compleet overzicht.

Latency kan op 3 punten ontstaan:

- Bij de camera (tijdens beeldverwerking / encoding)
- In het netwerk (routers, switches, firewalls, kabellengtes)
- Bij de ontvanger (client buffer, decoder vertraging, weergave vertraging)

Het verhelpen van latency heeft een keerzijde; de videokwaliteit wordt lager tenzij geïnvesteerd wordt in snellere hardware (CPU, netwerk) en/of software.

‘Startup Latency’ wordt veroorzaakt doordat de client eerst moet synchroniseren met een I-frame. Standaard wordt om de 250 beelden een I-frame ingelast. Een I-frame is een ‘volledig’ beeld, de andere beelden (P- en B-frames) zijn updates en geen volledig beeld. Door het gebruik van P- en B-frames hoeft er minder data verstuurd te worden wat minder tijd kost.

Een andere notatie van bovenstaande is ‘GOV’ (Group of Video). Dat is het aantal P- en B-frames na een I-frame. In het voorbeeld hierboven is de GOV waarde 250.

Latency en videocorruptie kunnen verminderd worden door:

Aan camerazijde:

- Zo veel mogelijk I- frames te gebruiken (dit vraagt een grotere bandbreedte, meer CPU kracht aan client zijde en levert een lagere kwaliteit video op).
- ‘init delay’ of ‘vbv buf size’ op de laagst mogelijk waarde instellen waarbij video nog een acceptabele kwaliteit heeft.
- Het aantal camerastreams dat in gebruik is beperken, zeker als de instellingen anders zijn (resolutie, encoding en dergelijke; dit kost CPU kracht in de camera).
- Verlagen van de videoresolutie. Het effect is kwadratisch; halveren van de resolutie zorgt dat de hoeveelheid data tot een kwart afneemt.
  Voorbeeld: 800x600 is 480.000 pixels, 400x300 is 120.000 pixels)
- Verlagen van het aantal beelden per seconde (frame-rate / refresh-rate).
- Verlagen van de bit-diepte (van 24 bits naar 16 bits, of naar 8 bits grayscale).
- Verlagen van kwaliteitsniveau/compressie en dus bit-rate.
- VBR aanzetten in plaats van CBR.
- Field-encoding aanzetten in plaats van frame-encoding.
- Buffering/ Cacheing aan client zijde uitzetten (bij OpenCV 3.0.0 is niets in te stellen).

Met betrekking tot het netwerk:

- Bij voorkeur een zo eenvoudig mogelijk netwerk gebruiken (alleen een Gbit switch).
- UDP aanzetten in plaats van TCP (UDP is sneller doordat het minder overhead heeft, maar kan ‘packet loss’ veroorzaken. Als dat gebeurt, dan heeft TCP de voorkeur).

Aan client zijde:

- Snelle CPU en snelle videokaart gebruiken.
- Bij hoge beschikbare bandbreedte van het netwerk, maar tragere CPU:
  MJPEG geeft minder latency (geen buffering) maar gebruikt ongeveer 10x de bitrate van H.264.
- Bij lage bandbreedte van het netwerk maar bij gebruik van een snelle CPU:
  H.264 geeft de meeste compressie en gebruikt de meest CPU capaciteit, in afnemende mate gevolgd door H.264, MPEG4, MJPEG, JPEG.

De vraag “*Hoe kan een videostream van een camera optimaal ingesteld worden voor het gebruik met tracking en PTZ aansturing?”* kan als volgt beantwoord worden:

Voor PTZ tracking doeleinden is onderstaande aanbevolen:

- Om een foutloze videoverwerking en weergave te krijgen is een snelle CPU nodig (Intel i5 2,67GHz CPU draait soepel; experimenteel bepaald).
- Gebruik bij voorkeur een MJPEG videostream, deze heeft geen latency, in tegenstelling tot H.264 streams. (klopt niet voor OpenCV 3.0.0; experimenteel waargenomen)
- Onderzoek welke resolutie videobeelden voldoende is voor het doel. Lager is beter en zorgt voor minder CPU belasting en daardoor voor minder decodeerfouten.

Helaas is het ook met deze nieuwe kennis niet gelukt alle problemen op te lossen.
De PTZ camera op het parkeerterrein blijft last houden van 4,2s latency en is daardoor niet geschikt voor een demo.

Aanbeveling voor vervolgonderzoek

Een aanbeveling voor vervolgonderzoek is:

Onderzoek hoe automatisch de vertraging van het videosignaal (latency) bepaald kan worden.

Hiermee kan de PTZ regelaar verbeterd worden met betrekking tot reactiesnelheid en nauwkeurigheid.

7 Conclusies en aanbevelingen

De hoofdvraag voor dit onderzoek is:

Wat is de beste configuratie voor PTZ auto-tracking voor het volgen van objecten op zee?

De hoofdvraag is beantwoord door de onderstaande deelvragen uit te werken.

Deelvragen:

1. *Welke tracker is het meest geschikt voor het detecteren en volgen van vaartuigen op zee?*
   a. *Hoe kunnen trackers getest worden?*
   b. *Welke tracker stelt de minste eisen aan hardware qua rekenkracht, geheugen, opslag?*

Trackers kunnen getest worden door de nauwkeurigheid te meten waarmee een object in een reeks beelden gevolgd wordt. Het object wordt gemarkeerd met een bounding box. De positie en grootte van deze bounding box wordt vergeleken met de bekende positie in beeld, de 'ground truth'.

In dit onderzoek is de meest geschikte tracker geselecteerd aan de hand van de uitkomsten van de 2014 Visual Object Challenge (VOT2014). De tracker DSST is de winnaar van de VOT2014 als meest robuuste en beste allround tracker. Bij de VOT2014 is de VOT toolkit als test tool gebruikt. Deze was op het moment van schrijven de meest geschikte tool om trackers te testen en met elkaar te vergelijken.

De DSST tracker is verder getest met beelden uit een maritieme omgeving; de haven van Rotterdam. Deze zijn voldoende representatief om de uiteindelijke gebruikssituatie te testen. Ook met deze tests kwam DSST als beste en snelste tracker uit de test. De tracker heeft geen opslag nodig. De tracker stelt zeer lage eisen qua geheugen.

2. *Hoe kun je een signatuur (‘Appearance Model’) van een object opbouwen en bijhouden?*
   a. *Hoe kan een object initieel gedetecteerd worden?*
   b. *Hoe ga je om met de afwijking tussen de locatie van een object uit de vaste camera en de locatie van dat object in het PTZ beeld?*

Uitgelegd is dat er vele manieren zijn om een signatuur van een object op te bouwen. Binnen dit project is ervoor gekozen om voor het opbouwen van een signatuur bij het detectiemechanisme feature points te gebruiken. De DSST tracker maakt ook gebruik van een signatuur, dit is echter deel van de ‘receptuur’ van de tracker en is geen keuzeoptie.

3. *Wat is de meest geschikte regelaar om de PTZ camera in de pas te laten lopen met de tracker?*
   a. *Is een lage orde PID regelaar voldoende of is een hogere orde regelaar, of zelfs een niet lineaire regelaar nodig?*
   b. *Hoe kunnen we het optimale zoomniveau bepalen zodat de tracker het object niet kwijtraakt?*
   c. *Hoe kunnen we na kwijtraken van het object deze weer terugvinden?*

a) Er is gekozen om een niet-lineaire (rule-based) regelaar te implementeren omdat er in de aansturing van PTZ meerdere elementen te benoemen zijn die niet op een generieke wijze aan te sturen zijn. De toegepaste regelaar is in delen in te regelen en daarmee eenvoudig in te zetten voor andere modellen camera’s.

b) Het zoomniveau is niet kritisch, een zoomniveau van 50% tot 70% wordt aanbevolen. Het te volgen object dient altijd volledig in beeld te blijven en geen beeldranden te raken daar anders de tracking stopt.

c) Aangezien dit project niet verder gaat dan een demo is dit slechts deels uitgewerkt. Bij de demo zoekt de tracker gedurende 50 frames naar het object. Als het object niet binnen die tijd terug gevonden is, wordt tracking afgebroken en is herdetectie nodig. Bij een uiteindelijke implementatie is nog steeds herdetectie nodig. De vraag die dan beantwoord moet worden is niet op welke plaats in het ontwerp, maar onder welke voorwaarden herdetectie gestart moet worden.

Optionele deelvragen (afhankelijk van de beschikbare tijd):

4. *Hoe om te gaan met occlusies (overlappingen in het beeld)?*
(Hoe) Kunnen beelden van vaste camera en PTZ camera samen gebruikt worden om tracking te verbeteren (Fusion)?

Deze vragen zijn niet uitgewerkt.

<u>Aanbevelingen voor vervolgonderzoek:</u>

1. De tracker voldoet aan alle eisen, maar verbetering is altijd mogelijk. Aan het einde van dit project is de VOT2015 challenge afgerond en is er een nieuwe ranglijst van ‘beste trackers’ waarbij DSST onder het gemiddeld scoort. Een aanbeveling is om te onderzoeken of uit de VOT2015 een beter presterende tracker te halen is.

2. Aanbeveling voor vervolgonderzoek: Op dit moment ‘anticipeert’ de detector niet. Het is mogelijk op basis van de laatste bewegingsvector van het object de volgende locatie voor de bounding box te voorspellen. Dit heeft alleen nut bij snelle objecten. Het kan ook gebruikt worden om traagheid in het systeem te compenseren (overdracht naar tracker, traagheid in PTZ aansturing). Het werkt nadelig bij snelle richtingsveranderingen.

3. De gekozen ‘rule based’ regelaar voldoet prima. Deze heeft echter als nadeel dat hij na-ijlt en het object niet gecentreerd in het beeld houdt. Een aanbeveling voor vervolgonderzoek is om alsnog te onderzoeken in hoeverre PID betere prestaties levert door iemand die deskundig is op het gebied van meet- en regeltechniek.

4. De detector werkt momenteel op basis van de OpenCV functie GoodFeaturesToTrack(). Dit is "Shi and Tomasi Feature Tracking" wat werkt met feature points (corner detection). Een aanbeveling is om te onderzoeken of een andere feature point detector (uit OpenCV) zoals ORB betere resultaten oplevert.

5. De detector is getest op beelden uit de haven van Rotterdam. Beelden vanaf een boorplatform op zee waren niet beschikbaar. Het heeft de voorkeur om de detector verder te testen en in te regelen met beelden van open zee vanaf een boorplatform..

6. Onderzoek hoe automatisch de vertraging van het videosignaal (latency) bepaald kan worden.Hiermee kan de PTZ regelaar verbeterd worden met betrekking tot reactiesnelheid en nauwkeurigheid.

7. Onderzoek hoe automatisch de vertraging van de PTZ aansturing gemeten kan worden. Hiermee kan de regelaar automatisch ingeregeld worden. Dit verbetert de snelheid en nauwkeurigheid van de PTZ aansturing.

8. Onderzoek hoe automatische kalibratie van PTZ snelheden gedaan kan worden zodat eenvoudig andere camera’s gebruikt kunnen worden zonder dat een gebruiker of installateur het systeem moet inregelen.

8 Evaluatie

Dit hoofdstuk bevat de beoordeling die de afstudeerder zelf geeft over de aanpak (proces, werkzaamheden, activiteiten) en de resultaten (producten) van de aanpak.

8.1 Productevaluatie

Het resultaat van dit project is een goed werkende demo bestaande uit een detector, tracker en PTZ besturing. Ik ben tevreden met het product dat ik gemaakt heb. Helaas heb ik het niet werkend kunnen maken in de haven van Rotterdam. Dat was geen deel van de opdracht, maar het was voor mij de kers op het toetje geweest.

Het praktisch einddoel van dit project was een demo met permanent opgestelde camera’s bij Siqura; een vaste camera en een PTZ camera. Dat doel is niet gehaald omdat de PTZ camera op het parkeerterrein niet bruikbaar bleek. De demo is in twee delen gedaan:

1) Met opnamen waarbij de Pelco voorbijgaand verkeer volgt.
2) Een live demo met de Pelco waarbij toeschouwers zelf de werking en prestaties van het systeem konden testen.

De demo is zeer positief ontvangen. De filmpjes van de demo zijn beschikbaar.

Het project had twee hoofddoelen: het selecteren van de meest geschikte tracker en het bepalen welke PTZ regelaar het meest geschikt is. Ik denk dat ik er in geslaagd ben de beste tracker te selecteren die op dit moment beschikbaar is.

De door mij gekozen ‘rule-based’ PTZ besturing is eenvoudig, snel, robuust, prima bruikbaar en universeel inzetbaar. De productmanager van de encoder wil de functionaliteit zelfs in de encoder inbouwen. Achteraf gezien denk ik dat er een betere oplossing mogelijk is op basis van PID. Dit valt buiten mijn kennisgebied.

Ik denk dat ik meest geschikte methode van detectie heb gekozen (sparse optical flow) en daarbij de juiste filtering heb toegepast voor een snelle en effectieve detectie.

Ik denk dat ik een degelijk stukje onderzoek heb gedaan. Ik ben er tevreden mee.

8.2 Procesevaluatie

Dit hoofdstuk geeft aan hoe ik zelf oordeel over de totstandkoming van producten; hoe fases en activiteiten zijn afgerond en hoe ik omgegaan ben met problemen en keuzes.

Vanuit mijn werkend bestaan ben ik gewend om onder tijdsdruk zeer zelfstandig te moeten werken en problemen zelf op te lossen. Dat was in dit geval ook zo.

Ik heb het afstuderen in deeltijd gedaan. Dat wil zeggen 2 dagen per week bij Siqura en de rest in eigen tijd. De richtlijn aan tijdsbesteding is 40 uur per week, dat heb ik zeker gehaald.

Bij de start van het project kon ik echt niet inschatten hoe lang ik voor activiteiten nodig zou hebben. Ik heb samen met mijn opdrachtgever een inschatting van de verschillende fasen gemaakt. Ook hij kon tijdsduur van onderdelen slecht inschatten.

Het lastige bij dit onderzoek was dat ik geen enkele referentie had, er waren teveel variabelen. Achteraf gezien was de inschatting van de tijd behoorlijk goed, al moet ik zeggen dat ik mijn uiterste best gedaan heb om elke geplande stap binnen de tijd te halen. Ik overlegde wekelijks met mijn opdrachtgever om te bepalen of ik nog op het goede pad zat en kreeg steeds een bevestiging. Toch bleef ik er onzeker bij.

Planning en fasering

De oorspronkelijk volgorde van de planning was:

1) Onderzoek naar tracking algoritmes
2) Implementatie van tracking algoritme op PC
3) PTZ regeling
4) Automatische initialisatie van tracking
5) Implementatie op embedded system
6) Optionele uitbreidingen.

Naarmate het onderzoek vorderde, veranderde de volgorde. De uiteindelijke volgorde was:

1) Onderzoek naar tracking algoritmes
2) Implementatie van tracking algoritme op PC
3) (*was 4*) Automatische initialisatie van tracking
4) (*was 3*) PTZ regeling
5) (*nieuw*) Integratie van mijn ‘ptztracker’ project in Jeffrey’s ‘ptzcontrol’ project

In deze volgorde sloot het onderzoek naar de Vision componenten 1+2 en 3 goed op elkaar aan. Het hele Vision deel is gedaan in de geplande tijd. De PTZ aansturing was echt een apart stuk onderzoek, dat kostte alles bij elkaar meer tijd dan verwacht. Het initiële deel van het onderzoek naar de PTZ regeling was volgens planning klaar. Ik was van plan aan de ‘implementatie op embedded systeem’ beginnen, maar dit kwam te vervallen.

Tijdens een overleg met stakeholders kwam naar voren dat er meer behoefte was aan een uitgebreide PC versie van de demo code dan aan een embedded versie (zie punt 5; integratie van projecten). Achteraf gezien heeft deze keuze de kwaliteit van het onderzoek verbeterd want ik kwam dingen tegen die ik eerder niet had voorzien. Bovendien had het embedded deel me veel tijd gekost met het risico dat ik het niet af kreeg.

Aan optionele uitbreidingen ben ik niet meer toegekomen. Ik heb gekozen voor kwaliteit van wat ik al had in plaats van kwantiteit in het aantal onderwerpen.

Fases 1 en 2: ’Onderzoek naar tracking algoritmes’ en ‘implementatie op PC’

Ik ben eerst begonnen met de toolkit. Mijn inschatting was dat als je een tracker wilt evalueren, dat je eerst een betrouwbaar testmechanisme moest hebben. Dat is dus waar ik de eerste 2 weken hardnekkig mee bezig geweest ben totdat het functioneerde. Daarbij kwam ik veel (bij)zaken tegen waarop ik niet had geanticipeerd en die niks te maken hadden met Vision, Linux of C++, dat voelde als nutteloos. Om grip op het geheel te houden heb ik nauwkeurig een logboek van handelingen bijgehouden waar ik op terug kon grijpen zodat ik dingen kon reproduceren. Het aan de praat krijgen van de toolkit voelde aan als een mijlpaal. Toen had ik het gevoel dat ik werkelijk kon beginnen.

De volgende stap was het integreren en testen van trackers in de toolkit. Daarbij heb ik een hoop geleerd van zowel Linux (hoe het zit met de locaties van libraries en library-versies, include directories, linken en dergelijke), QtCreator (de ontwikkelomgeving), CMake/Make en het compileren van C++ applicaties en OpenCV. Door onder andere het gestoei met het compileren van OpenCV en C++ applicaties kan ik nu zeggen dat ik in staat ben dit makkelijk te reproduceren en troubleshooten.

Waar ik duidelijk kennis in tekort kom, is diepgang in de theorie achter trackers. Ik miste hier kennis op diverse gebieden; Vision, wiskunde (beide zijn op universitair niveau) maar toch ook de C++ ervaring om dingen op te kunnen zetten en aan te kunnen passen. Daardoor was het soms lastig en tijdrovend om keuzes te maken. Ik wilde in ieder geval bereiken dat ik overzicht had van wat tracker componenten doen. Dat is behoorlijk gelukt.

Uiteindelijk ben ik hierdoor tot de ontdekking gekomen dat datgene waarvan ik dacht dat het volledige trackers waren in feite niet meer waren dan alleen de feature detectors uit de trackers. Zonder verdere studie ben ik niet in staat om hiermee zelf trackers te implementeren. Hoe dan ook heb ik er veel van geleerd en de drang blijft bestaan om er meer van te begrijpen.

Fase 3: ‘Automatische initialisatie van tracking’ (Feature point detectors / optical flow)

Ik ben tevreden met mijn onderzoek naar feature point detectors, homography projectie en Optical Flow detectiemechanismen. Daar heb ik veel van geleerd.

Ik denk dat ik nu goed op de hoogte ben van de modulaire opbouw van OpenCV en het gebruik ervan. Daarbij blijft het probleem dat het soms een black box is omdat je functies aanroept waarvan je de werking niet doorgrondt (ja, je kunt de source codes uitpluizen, maar dat kost veel tijd en dat had ik niet). Dat betekent veel lezen in OpenCV documentatie en papers van gebruikte algoritmes.

Ik was al bezig met volgende fase in project (uitzoeken hoe PTZ camera aan te sturen en uit te lezen) toen ik erachter kwam dat mijn selectie (Dense Optical Flow) niet de juiste was omdat deze te traag was. Toen moest ik een stap terug en kijken hoe ik de andere methode (Sparse Optical Flow) bruikbaar kon krijgen, oftewel een iteratie.

Fase 4: ‘PTZ regeling’

Ik ben intens bezig geweest met het uitzoeken van PTZ aansturing. Daarbij de nodige zijpaden bewandeld die voor mij noodzakelijk waren om de materie te doorgronden:

- Het versturen van http commando’s vanuit C++ (http get, http post en de communicatie daarbij. In eerste instantie werkend gemaakt met Qt libraries omdat deze horen bij de ontwikkelomgeving, maar later omgebouwd voor gebruik met de Boost libraries welke deel uitmaken van de Siqura libraries.
- Het aan de praat krijgen van gSoap; een SOAP/XML client in C++ voor het versturen van ONVIF commando’s. Dit was nodig omdat ik de beschikbare code uit het project van Jeffrey onvoldoende kon doorgronden en bovendien werkte de Jeffrey code niet. Toen ik alles werkend had heb ik de code weer omgebouwd voor gebruik met de Boost libraries zodat er geen overbodige toevoegingen aan de Siqura code waren.

Ik heb gedurende het project regelmatig gebruik gemaakt van WireShark om te zien welke commando’s en terugmeldingen er over het netwerk gaan. Hier was ik al behendig in en dat heeft me voordeel en tijdswinst opgeleverd.

Fase 5: Combineren/integreren van projecten

De ‘implementatie op embedded systeem’ is komen te vervallen. In plaats daarvan was er het verzoek tot combineren van mijn project en die van mijn voorganger. Het combineren van mijn code en de code van Jeffrey heeft veel tijd gekost. Ik kreeg delen van de code van Jeffrey niet aan de praat. Julien (opdrachtgever) heeft mij geholpen met troubleshooten omdat dit mij veel tijd kostte. Het heeft mijn troubleshooting vaardigheden verbeterd.

Ik heb veel geleerd over het debuggen met gebruik van de compiler. Waar ik eerder niet in staat was de vinger op de zere plek te leggen, weet ik nu precies hoe ik de debugger kan gebruiken zodat ik fouten kan lokaliseren. Dit is nuttig bij bijvoorbeeld problemen met pointers (verwijzingen naar geheugen) en andere problemen die niet tijdens het compileren aan het licht komen.

Het was de bedoeling om de eind-demo te geven met dit combiproject. Uiteindelijk heb ik de projecten wel gecombineerd, maar niet met de verwachte functionaliteit. Daarvoor moest ik de code van Jeffrey teveel voor verbouwen en daar kon- en wilde ik geen tijd meer aan besteden. Daarnaast had ik teveel problemen met de PTZ camera op het parkeerterrein.

Demodag en PTZ camera op parkeerterrein (onverwachte diepgang)

Op 17 december was er in de haven van Rotterdam een demodag van het APPS project. Helaas werkte mijn code daar niet, achteraf gezien door een tikfout in mijn code. Doordat ik WireShark captures had kon ik achteraf controleren en corrigeren wat mis ging.

Tot mijn verbazing werkte de hoge resolutie H.264 stream (2048 x 1536 x 30fps) feilloos op de Core2Duo PC met de ptztracker demo applicatie terwijl bij Siqura elke stream, waaronder die van de PTZ camera op het parkeerterrein voor problemen zorgde. Dit heeft tot een nieuw stuk onderzoek geleid over video streams. De problemen met H.264 streams hebben een dermate grote impact dat het de moeite waard was om er een hoofdstuk aan te wijden.

Op de demodag heb ik de samenhang van de componenten van de deelnemers aan het APPS project gezien en hoe mijn deel op dit project aansluit. Hier heb ik gezien hoe uit thermische beelden een schip wordt gedetecteerd en hoe uit het beeld van de PTZ camera object classificatie (het herkennen van het type schip) plaatsvindt. Hieruit kon ik afleiden wat (ongeveer) het gewenste zoomniveau is: zolang een schip er maar in zijn geheel op staat, zeg de helft van de breedte of hoogte van het beeld. Er wordt gewerkt met hoge resoluties, dus het zoomniveau is niet belangrijk.

Achteraf gezien heeft deze dag er voor gezorgd dat er meer diepgang in het onderzoek kwam omdat hierbij meer voor PTZ aansturing en streaming zaken onderzocht moesten worden. Er kwam meer naar boven dan ik daarvoor in had kunnen schatten. Dit heeft onder andere gezorgd voor een betere onderbouwing van:

- de keuze voor de ‘rule-based’ regelaar
- de vertaling van PTZ snelheden naar graden per seconde (‘universeel’ maken)
- een logische verklaring van problemen met- en troubleshooting mogelijkheden van videostreams

UML / ontwerptekeningen

In een vroeg stadium in het project ben ik begonnen met het beschrijven van een ontwerp in UML (use cases, sequence diagrams). Het doel was voor mijzelf verduidelijken hoe het gedrag van PTZ autotracking moest zijn. Ik kreeg er onvoldoende grip op en strandde in mijn pogingen het in UML te vatten. Het was voor mij eerst nodig om het gedrag van de diverse componenten te leren kennen door te experimenteren.

Ik heb een UML ontwerp gemaakt maar ben van mening dat UML binnen dit project geen meerwaarde heeft. Dit project kent één use case en er vindt geen gebruikersinteractie plaats

Bij de minor Computer Vision en bij Siqura is men gewend om flow-chart achtige tekeningen te maken (geen UML). Voor uitleg van de globale werking heb ik naast een eenvoudig UML ontwerp een blokschema gemaakt. Voor het begrip van de detector heb ik een flow-chart gemaakt. Blokschema’s en flowcharts zijn voor iedereen te lezen.

De tracker is een black box waarvan alleen de in- en uitgangsvoorwaarden bekend zijn. Het gedrag van de PTZ aansturing was naar mijn mening het best te verklaren door een plaatje te maken van regio’s in het beeld. Van de tracker en de PTZ aansturing heb ik het gedrag omschreven in tekst.

Keuze voor snelheid

Bij het gehele onderzoek is ernaar gestreefd dat alle componenten met snelle bewegingen om kunnen gaan. Een keuze voor het om kunnen gaan met snelle bewegingen maakt het systeem robuust. Een systeem dat om kan gaan met snelle veranderingen heeft als voordeel dat het ook kan omgaan met langzame veranderingen.

Leerpunten en ervaringen

Ik besef me tot mijn grote ergernis dat ik op het gebied van C++ nog maar weinig weet en er zeker niet behendig mee ben. Het gaat dan over onderwerpen als smart pointers, interface classes, static casting, op een correcte manier object georiënteerd programmeren. Het kost me erg veel tijd om dingen uit te zoeken. Aan de andere kant geeft dit aan dat ik toch een hoop geleerd heb van C++ maar ik ben nog lang niet op een punt waarover ik tevreden ben.

Ik heb de nodige ervaring opgedaan met het versiebeheersysteem Subversion. Ook dat heb ik nu in de vingers. Tot ongeveer halverwege het project maakt ik lokale kopieën die met Dropbox gesynchroniseerd worden, maar geeft niet dezelfde zekerheid en controle.

Ik heb dit project / onderzoek met veel plezier uitgevoerd, ik kan wel zeggen dat dit mijn leukste klus ooit was. Ik heb veel geleerd. In mijn huidige (en eerdere) werk was ik altijd de deskundige en deed ik alles met twee vingers in de neus. Hier had ik soms het gevoel alsof ik een peuter was die bij alles hulp nodig had. Niet dat ik snel om hulp vraag . Ik probeer er eerst zelf uit te komen, daar leer ik het meeste van, maar omwille van de tijd moet je soms om hulp vragen. Ik leer weer eens wat en heb daar veel plezier in.

Wat ook een hele leuke ervaring is, is de totaal andere sfeer en bedrijfscultuur. Op de R&D afdeling bij Siqura zitten mensen die serieus met techniek bezig zijn, onderzoeken, ontwikkelen en diepgang zoeken.

Dat brengt mij op het laatste; ik vind het onderzoeken en het experimenteren het leukste. Ik zie er een uitdaging in om zaken helder te formuleren, goed te documenteren en te beargumenteren (ik hoop dat ik er wat van bak; zo niet dan wil ik er beter in worden).

8.3 Evaluatie beroepstaken

Om aan te geven in welke mate een beroepstaak is behaald wordt verwezen naar de hoofstukken en paragrafen waarin de betreffende werkzaamheden zijn uitgevoerd. Hierbij wordt aangegeven of een beroepstaak geheel of gedeeltelijk en op welke niveau(2, 3 of 4) is uitgevoerd (zie Tabel 4).

		Taakrol		
		Geleid	zelfstandig	Sturend
Context	simpel	1	2	3
	lastig	2	3	4
	complex	3	4	5

Tabel 4 Niveauwaarden bij beroepstaken

A1 : Analyseren van het probleemdomein

Dit heb ik aangetoond gedurende het gehele onderzoek, door het doen van onderzoek naar tracking algoritmes, feature point detectors, optical flow, ptz aansturing, streams.
Geheel uitgevoerd op niveau 4(complexe materie, zelfstandig uitgevoerd).
Zie hiervoor hoofdstuk 0 tot en met 6.

G1 : Praktische aspecten hanteren in (internationale) projecten

Dit heb ik aangetoond bij onder andere het opstellen van de opdrachtomschrijving en met het projectplan/plan van aanpak.
Geheel uitgevoerd op niveau 3.

C8 : Ontwerpen van een technisch informatie systeem

Deze beroepstaak is aan het begin van het project gekozen omdat ik verwachtte dat er een grotere ontwerpcomponent en ontwikkelcomponent in het project zou zitten. Vanuit het project is alleen een demo vereist.

Bij Siqura wordt niet gewerkt met UML maar met flowchart achtige tekeningen. Vanuit de minor Embedded Vision Design ben ik gewend aan Input-Proces-Output diagrammen en beschrijvingen. Dit zijn eenvoudige blokdiagrammen.

In hoofdstuk 4.4 is de werking van de detector omschreven. De flowchart van de werking van de detector staat in “Bijlage F”. In hoofdstuk 5.4 is een ontwerp omschreven van de PTZ regelaar. Ter ondersteuning is een tweetal blokschema’s gemaakt (Figuur 2 op pagina 2, Figuur 14 op pagina 30).
In “Bijlage E” staat een UML ontwerp
Geheel uitgevoerd op niveau 3.

E24 : Kwantitatieve analyse maken van de prestaties van systemen

Dit is op meerdere plaatsen in het project voorbij gekomen:
- Bij het inregelen van de gevoeligheid van de detector (hoofdstuk 4.4), de grenzen waarbinnen een betrouwbare object detectie gedaan wordt.
- Bij het meten van de snelheid bij het versturen van ONVIF commando’s (h. 5.1).
- Bij het meten van latency van videostreams en met het beschrijven van de methode hoe dit te doen (hoofdstuk 6).

Bij onderstaande punten is dit heel sterk; er is een afhankelijkheid. De PTZ regelaar is ontworpen om te werken met snelheden in graden per seconde en is onafhankelijk gemaakt van een type camera.
- Bij de analyse van de gewenste werking van de PTZ regelaar, de zones in het beeld en de bij de zones horende snelheden (hoofdstuk 5.2).
- Bij het bepalen van snelheden van verschillende PTZ camera’s en dit omvormen naar getallen in graden per seconde, inclusief een methode om dit voor andere camera’s te doen (hoofdstuk 5.3). In hoofdstuk 5.4 is beschreven hoe de maximale snelheid voor de PTZ regelaar berekend kan worden.

Geheel uitgevoerd op niveau 3.

Literatuurlijst

Alahi, A., Ortiz, R. & Vandergheynst, P., 2012. FREAK: Fast Retina Keypoint. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 510-517.
APPS, 2015. *Advancing Plug & Play Smart Surveillance.* [Online]
Available at: http://www.apps-project.eu/
[Geopend 30 12 2015].
Baeten, J., 2003. *Beknopte inleiding tot de regeltechniek, Fuzzy Logic H3.* [Online]
Available at: http://sts.bwk.tue.nl/7y500/readers/FuzzyLogic_H3.pdf
[Geopend 23 12 2015].
Bolton, W., 2008. *Mechatronics.* 4th edition red. Harlow: Pearson Education Limited.
Čehovin, L., Leonardis, A. & Kristan, M., 2015. Visual object tracking performance measures revisited. *arXiv:1502.05803 [cs.CV]*.
Čehovin, L., Nebehay, G. & Vojíř, T., 2014. *Visual Object Tracking (VOT) challenge evaluation toolkit.* [Online]
Available at: https://github.com/votchallenge/vot-toolkit
[Geopend 1 9 2015].
Čehovin, L., Nebehay, G. & Vojíř, T., sd *Visual Object Tracking (VOT) challenge evaluation toolkit.* [Online]
Available at: https://github.com/votchallenge/vot-toolkit
[Geopend 1 9 2015].
Danelljan, M., Häger, G., Khan, F. S. & Felsberg, M., 2014. *Accurate Scale Estimation for Robust Visual Tracking.* sl, IEEE Conference on Computer Vision and Pattern Recognition.
Danelljan, M., Khan, F., Felsberg, M. & Van de Weijer, J., 2014. Adaptive color attributes for real-time visual tracking. *Conference on Computer Vision and Pattern Recognition CVPR,* Volume 2014, pp. 1090-1097.
Farnebäck, G., 2003. Two-frame motion estimation based on polynomial expansion.. *Image Analysis,* 2003(Springer), pp. 363-370.
FFmpeg, 2015. *FFmpeg - Latency.* [Online]
Available at: https://trac.ffmpeg.org/wiki/StreamingGuide#Latency
[Geopend 02 01 201].
Grit, R., 2011. *Project Management, zesde druk.* Groningen/Houten: Noordhoff.
IEEE, 2015. *IEEE Xplore Digital Library.* [Online]
Available at: http://ieeexplore.ieee.org.ezproxy.hhs.nl:2048/Xplore/home.jsp
[Accessed 28 02 2015].
Itseez, 2015. *Open Source Computer Vision Library.* [Online]
Available at: http://opencv.org
[Accessed 28 02 2015].
Kristan, M., Matas, J., Čehovin, L. & Pflugfelder, R., 2015. *VOT Challenge.* [Online]
Available at: http://www.votchallenge.net/
[Geopend 11 10 2015].
Kristan, M. et al., 2015. A Novel Performance Evaluation Methodology. *arXiv:1503.01313 [cs.CV]*.
Kristan, M. et al., 2014. The Visual Object Tracking VOT2014 challenge results. *Computer Vision - ECCV 2014 Workshops, Proceedings, Part II. Lecture Notes in Computer Science 8926,* 12 9, pp. 191-217.
Li, X. et al., 2013. A survey of appearance models in visual object tracking. *arXiv:1303.4803v1 [cs.CV]*.
Ma, C. & Liu, C., 2015. Two dimensional hashing for visual tracking. *Computer Vision and Image Understanding,* 2015(135), pp. 83-94.

Maresca, M., 2014. Matrioska: Real-Time Tracking by Detection Using Keypoints. *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on,* pp. 720-725.
Meltytech, LLC., 2015. *MLT Multimedia Framework.* [Online]
Available at: http://www.mltframework.org
[Geopend 10 1 2016].
Neijsel, G. d., 2015. *Personen tellen in een klaslokaal,* Zoetermeer: sn
ONVIF, 2008. *ONVIF PTZ coordinate spaces.* [Online]
Available at:
http://www.onvif.org/Portals/0/documents/whitepapers/ONVIF_PTZ_coordinate_spaces.pdf
[Geopend 23 11 2015].
ONVIF, 2011. *ONVIF Profile S Specification, Version 1.0.* [Online]
Available at:
http://www.onvif.org/Portals/0/documents/op/20984_11763r01ONVIF_Members-Profile_S_Specification.pdf
[Geopend 23 11 2015].
ONVIF, 2015. *ONVIF - The IP based Security Standard.* [Online]
Available at: http://www.onvif.org
[Geopend 23 11 2015].
OpenCV, 2015. *Features2D + Homography to find a known object.* [Online]
Available at: http://docs.opencv.org/3.0.0/d7/dff/tutorial_feature_homography.html
[Geopend 12 10 2015].
Pernis, J. v., 2015. *Automatische besturing PTZ camera - stageverslag,* Gouda: Siqura.
Reus, N. d., 1994. *Assessment of Benefits and Drawbacks of Using Fuzzy Logic Especially in Fire Control Systems.* [Online]
Available at: https://www.researchgate.net/publication/235053036
[Geopend 23 12 2015].
Siqura, 2015. *Siqura.* [Online]
Available at: http://www.siqura.com
[Geopend 24 11 2015].
Smeulders, A. W. M. et al., 2014. Visual Tracking: An Experimental Survey. *IEEE Transactions on pattern analysis and machine intelligence,* Issue VOL.36, pp. 1442-1468.
STOWA, 1997. *De toepasbaarheid van Fuzzy Control bij het zuiveren van stedelijk afvalwater.* [Online]
Available at:
http://www.stowa.nl/Upload/publicaties2/mID_4924_cID_3914_08980464_1997-32_toepasbaarheid-fuzzy-control.pdf
[Geopend 23 12 2015].
The VOT2014 committee, sd *Visual Object Tracking (VOT) Challenge.* [Online]
Available at: http://www.votchallenge.net/
[Geopend 1 9 2015].
TKHSecurity, 2015. *User manuals Video over IP - Archive.* [Online]
Available at: http://www.tkhsecurity-usa.com/optelecom_C01/Modules/PaginaA/PaginaA_Module.asp?comid=6&modid=413
[Geopend 24 11 2015].
Yang, H. et al., 2011. Recent advances and trends in visual tracking: A review. *Neurocomputing 74,* pp. 3823-3831.
Yilmaz, A., Javed, O. & Shah, M., 2006. Object tracking: A survey. *Acm computing surveys (CSUR) 38,4, Article 13.*

Verklarende woorden en afkortingen

API
‘Application Programming Interface’; een verzameling definities voor communicatie tussen computerprogramma’s onderling of voor communicatie met apparatuur. De API bestaat meestal uit bibliotheken geschreven voor een bepaalde programmeertaal en operating system.

Appearance Model
‘Appearance’: verschijning, uiterlijk of aanzien.
‘Appearance Model’: representatie of “signatuur”.

BB / bounding box
Kader om regio in het beeld (zie ROI) waar bewerkingen op gedaan worden.

Bit-diepte / bit depth
Het aantal bits dat gebruikt wordt om de waarde van één pixel in uit te drukken. Grijswaarde beelden zijn vaak 8-bits (256 waarden), kleurenbeelden zijn vaak 24-bits per pixel (circa 16 miljoen kleuren).

Centroid
Zwaartepunt (eenvoudig gesteld: het middelpunt).

CGI
‘Common Gateway Interface’; een methode waarmee een webserver met externe programma’s kan communiceren; CGI programma’s of CGI scripts. Het is de eenvoudigste manier om dynamisch de inhoud van een web site te beïnvloeden.

Drifting
Een object wordt in het beeld gemarkeerd met een bounding box. De bounding box omsluit nooit exact het object, dus om het object heen zijn stukjes achtergrond te zien. Het probleem is dat deze stukjes achtergrond als deel van de voorgrond gezien worden en abusievelijk gevolgd worden. Het probleem is groter als er occlusie optreedt.

Frame
Een enkel beeld (uit een reeks beelden).

fps
Frames per second / aantal beelden per seconde.

Ground Truth
Een weergave van het ideale resultaat, meestal met de hand samengesteld.

Homography
Projectie/transformatie van een vlak in perspectief. Stel: de bron is een rechthoek (2D), dan is de projectie een rechthoek in perspectief (2D) alsof deze vanuit een andere hoek (in 3D) bekeken wordt.

HTML
‘HyperText Markup Language‘; qua structuur met XML vergelijkbare opmaaktaal waarmee gestructureerde gegevens worden weergeven in de vorm van platte tekst.

HTTP
‘Hypertext Transfer Protocol’; transportmechanisme voor verzenden van data (in principe tekst) van en naar webservers.

LWIR
Long-wave infrared camera’s zijn warmtebeeld camera’s voor golflengtes van 7μm – 14 μm.

Matching
Vergelijken; ‘pattern matching’ is patroonvergelijking.

NVC
Network Video Client; bijvoorbeeld een PC met webbrowser om video afkomstig van een IP camera te bekijken.

NVT
Network Video Transmitter; een IP camera of een analoge camera met video encoder.

Occlusie / occlusion
Een object verdwijnt (gedeeltelijk) achter een ander object, overlapping.

ONVIF
‘Open Network Video Interface Forum’; non-profit organisatie die als doelstelling heeft om standaarden te ontwikkelen voor het van netwerk video producten.

Pose
De houding, stand of positie van een object.

Real-time
Real-time verwerking: Real-time slaat niet op een zo snel mogelijke afhandeling, maar op het verwerken binnen een controleerbare, voorspelbare en reproduceerbare tijd.

Resolutie
Het oplossend vermogen van een beeld of beeldscherm. Een term waarmee de kwaliteit en scherpte van een beeld wordt uitgedrukt. Resolutie slaat op de dichtheid van het aantal beeldpunten (pixels). Hoe hoger de resolutie hoe meer beeldpunten er zijn en hoe scherper de afbeelding is. Wordt uitgedrukte in horizontale * verticale resolutie of bijvoorbeeld in ‘pixels per inch’

ROI; Region of Interest
Uitsnede uit het beeld / sectie in het beeld welke geselecteerd is om verdere bewerkingen op te doen, vaak gemarkeerd met een bounding box.

SOAP
‘Simple Object Access Protocol’; XML achtige structuur welke over HTTP (web-services) getransporteerd wordt. **gSOAP** is een open source C/C++ software development kit voor SOAP/XML web services.

Tracking-by-detection
Dit werkt door het opbouwen van een model door het object in de voorgrond te onderscheiden van de rest. Het model “leert”, dat wil zeggen, het wordt bijgewerkt met informatie uit nieuwe beeldinformatie.

URI
‘Uniform Resource Identifier’; kan op dezelfde manier gebruikt worden als de URL, maar kan bestaan uit een relatief pad zoals het deel achter de hostnaam: `/ptz.cgi?pan=43&tilt=5`

URL
‘Uniform Resource Locator’; Een URL is een formele beschrijving om een bron te kunnen benaderen, zoals: `protocol://[gebruiker:wachtwoord@]host(naam)[:poort]/path`

XML
‘Extensible Markup Language’; qua structuur met HTML vergelijkbare opmaaktaal waarmee gegevens gestructureerd worden weergeven in de vorm van platte tekst.

Bijlage A – Featurepoint detectors in OpenCV 3.0.0

Binnen OpenCV bestaat het feature point detectie mechanisme uit 3 componenten:

1. Feature detector of keypoint detector
2. Descriptor extractor (binary descriptors)
3. Descriptor matcher

OpenCV is zodanig opgebouwd dat combineren of uitwisselen van deze componenten tussen de verschillende algoritmes heel eenvoudig is. Als voorbeeld kan eenvoudig de ‘feature detector’ van SURF gebruikt worden in combinatie met de ‘descriptor extractor’ van ORB.

Van de hieronder genoemde algoritmes zijn FAST en STAR alleen feature detectors. De anderen zijn zowel Feature detector als Descriptor extractor.
De Descriptor matcher is een aparte component (voorbeeld: Brute Force matcher of FLANN)

Dit is de lijst van Featurepoint detectors die OpenCV 3.0.0 bevat in volgorde van leeftijd:

Naam	**Bijzonderheden**	**Referentie**
SIFT*	“Scale Invariant Feature Transform”	[Lowe04]
SURF*	invariant to rotations; “Speeded Up Robust Features”	[Bay06]
FAST		[Rosten06]
STAR		[Agrawal08]
MSER	“Maximally stable extremal region extractor” (2002-2011).	
ORB	invariant to rotations, but not scale, “Oriented-Brief , An efficient alternative to SIFT or SURF”	[RRKB11]
BRISK	invariant to rotations and scale, “Binary Robust Invariant Scalable Keypoints”	[LCS11]
FREAK	invariant to rotations and scale, “Fast Retina Keypoint”	[AOV12]
KAZE		[ABD12]
AKAZE		[ANB13]
BRIEF	not invariant to rotations and scale, “Binary Robust Independent Elementary Features”	[Calonder10]

*gepatenteerd

Alle feature point detectors hierboven zijn bedoeld voor het gebruik met grayscale images.

Voor bovenstaande referenties verwijs ik naar de bibliografie van OpenCV 3.0.0:
http://docs.opencv.org/3.0.0/d0/de3/citelist.html

Voor MSER zie https://en.wikipedia.org/wiki/Maximally_stable_extremal_regions

Bijlage B – Toelichting trackers VOT2014

Tekst afkomstig uit (Kristan, et al., 2015).

Matrioska (Maresca, 2014) ontleedt tracking in twee aparte modules: detecteren en leren. De detectiemodule bepaalt welke keypoint gebaseerde methode (ORB, FREAK, BRISK, SURF, etc.) op dat moment het meest geschikt is om het object nauwkeurig te detecteren. De lerende module houdt een model bij van het object en volgt hierbij een ‘groeien en snoeien’ tactiek.

Matrioska Best Displacement Flow (MatFlow) (Maresca, 2014)
Matflow is een doorontwikkeling op Matrioska. Het gebruikt BDF (Best Displacement Flow) om het optimale traject te bepalen als Matrioska aangeeft dat het een “low confidence score” heeft. Deze “confidence score” is gebaseerd op het aantal keypoints dat in het object gevonden is. BDF maakt gebruik van clustering om de bounding box te updaten en BDF voert de “Consensus-Based Reinitialization” procedure uit om keypoints die als outliers aangemerkt waren te hergebruiken.

Discriminative Scale Space Tracker (DSST) (Danelljan, et al., 2014)
DSST is een doorontwikkeling op de Minimum Output Sum of Squared Errors (MOSSE) tracker. Het verschil is dat er rekening gehouden wordt met het groter of kleiner worden van objecten, oftewel het schaalniveau wordt aangepast. Dit gebeurt door een “discriminative correlation filter” te trainen op regio’s bestaande uit helderheidswaarden in grijswaarde beelden. Het correlatiefilter wordt vervolgens toegepast om de grootte van het object in het volgende frame te schatten. Tevens worden de intensiteitseigenschappen uit de MOSSE tracker gecombineerd met een pixel-dichte weergave van HOG (Histogram of Gaussians) eigenschappen.

Kernelized Correlation Filter tracker (KCF) (Danelljan, et al., 2014)
Deze is gebaseerd op eenvoudige Histogram of Gaussians (HOG) kenmerken. KCF werkt hetzelfde als een “Kernel Ridge Regression” getraind met duizenden beeldfragmenten rond het object welke verschillende translaties hebben. De verbeteringen ten opzichte van de vorige versie zijn schaal-onafhankelijkheid, sub-cel inschatting van maxima en het vervangen van het update model door een lineaire interpolatie met een meer robuust update schema.

Bijlage C – Voorbeeld van ONVIF commando

‘Absolute Move’ commando (met authenticatie)
Onderstaande is afkomstig uit een Wireshark netwerk capture.

Bovenaan staat de HTTP header. Daaronder de SOAP/XML structuur, deze wordt verzonden als leesbare tekst zonder opmaak. In het `<s:Header>` gedeelte worden de authenticatie gegevens meegegeven. Deze worden standaard bij elk bericht meegestuurd. Door de authenticatie op de camera uit te zetten kan per bericht circa 700 bytes bespaard worden. Onderstaand bericht is in zijn geheel 1106 bytes groot.

Als response op dit commando vanaf de NVC komt een SOAP/XML bericht van de NVT terug. De response is zelfs groter dan de request was, maar in de praktijk kan de response genegeerd worden. De waardes die voor Pan, Tilt en Zoom coördinaten en snelheden gebruikt worden zijn van het type float en liggen tussen -0,99f en 1,00f.

```
POST /onvif/ptz_service HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8;
action="http://www.onvif.org/ver20/ptz/wsdl/AbsoluteMove"
Host: 192.168.178.47
Content-Length: 1106
Accept-Encoding: gzip, deflate
Connection: Close

<s:Envelope
 xmlns:s="http://www.w3.org/2003/05/soap-envelope">

<s:Header>
 <Security s:mustUnderstand="1"
  xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
  <UsernameToken>
   <Username>Admin</Username>
   <Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordDigest">KV/dpSIFgA0CU/gHcorf7sWks64=
    </Password>
   <Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-1.0#Base64Binary">
qu3ftBn59UipSG6hO6ZpOAMBAAAAAA==</Nonce>
   <Created xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2015-11-19T15:21:18.826Z</Created>
  </UsernameToken>
 </Security>
</s:Header>

<s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <AbsoluteMove xmlns="http://www.onvif.org/ver20/ptz/wsdl">
   <ProfileToken>
    ProfileToken_1_1
    </ProfileToken>
   <Position>
    <PanTilt
     x="-0.177215189"
     y="-0.4556962"
     xmlns="http://www.onvif.org/ver10/schema"/>
    <Zoom
     x="0.5"
     xmlns="http://www.onvif.org/ver10/schema"/>
   </Position>
</s:Body>
</s:Envelope>
```

Bijlage D – PTZ CGI besturing en cameraspecificaties

Aansturing van de Pelco Spectra III via Siqura Eve One encoder:

Onderstaande tabel “Eve One PTZ control commands” is afkomstig uit “PTZ Camera and Codec Programming Interface v1.5” van (TKHSecurity, 2015),
Hoofdstuk 2.8 (pagina 25 van de pdf): *"ptz.cgi - Parameter Configuration"*

Voorbeeld camera URI waarbij <servername> een geldige hostnaam of IP adres is:

```
http://<servername>/api/ptz.cgi?PTZNumber=1&PanSpeed=14&TiltSpeed=56&ZoomSp
eed=-16
```

De standaard timeout van een Eve One encoder voor een ‘Continuous Move’ commando is 15 seconden. Daarna stopt alle beweging.

Eve One PTZ control commands:

Parameter	Type	Bereik	Omschrijving
PanSpeed	Integer	+/-100	Change PTZ horizontal move speed. Positive = right, 0 = stop. Note: since only speed is managed by the UI, it is the camera's responsibility to reduce Pan and Tilt speed when zoomed in and vice-versa.
TiltSpeed	integer	+/-100	Change PTZ vertical move speed. Positive = up, 0 = stop.
ZoomSpeed	Integer	+/-100	Change PTZ zoom speed. Positive = zoom in, 0 = stop.
GotoPreset	Integer	1 .. 100	Move PTZ to specified preset number.
SetPreset	Integer	1 .. 100	Set specified preset to the current PTZ position.

Aansturing van Siqura HSD820H2 compatible camera’s:

Onderstaande tabel is afkomstig uit “Siqura API - Interface Specification” (Siqura, 2015).

Voorbeeld camera URI waarbij <servername> een geldige hostnaam of IP adres is:

```
http://<servername>/cgi-bin/com/ptz.cgi?continuouspantiltmove=14,5&
zoomsteps=-16
```

De Siqura HSD820 familie van camera’s kunnen via de CGI verschillende typen aansturing ondersteunen; ‘Absolute Move’, ‘Relative Move’, ‘Continuous Move’.
Gebruik deze URI voor een overzicht van de commando’s welke de camera ondersteunt:

```
http://<servername>/cgi-bin/com/ptz.cgi?info=1
```

Siqura HSD820 PTZ control commands:

Parameter	Type	Bereik	Omschrijving
continuouspantiltmove	Integer	+/-100	Positive values mean right (pan) and up (tilt), negative values mean left (pan) and down (tilt). "0, 0" means stop.
continuouszoommove	integer	+/-100	Positive values mean zoom in and negative values mean zoom out. Higher value gives higher speed. "0" means stop.
gotoserverpresetno	Integer	1 .. 256	Move to the position associated with the specified preset position number.
query=position	-	-	Returns the current parameter values (position).
info=1	integer	1	Returns a description of available PTZ commands.

Specificaties Pelco Spectra III PAL (DD53CBW18-X)

Vertical Tilt:	+2° to -92°
Manual Pan/Tilt Speeds:	Pan: 0.1°~80°/sec manual operation, 150°/sec Turbo
	Tilt: 0.1°~40°/sec manual operation,
Maximum motor speed:	Preset Speeds: Pan 250°/sec, Tilt 200°/sec
Zoom Speed (optical range):	2.9/4.2/5.8 seconds
Optical zoom:	18
Horizontal Angle of view:	51° at 3.8 mm wide zoom, 3° at 68.4 mm telephoto zoom
Max. encoding/resolution:	native: analog 724x582, via Eve One encoder: 960x576
Image sensor:	1/4” CCD

Specificaties Siqura HSD820H2

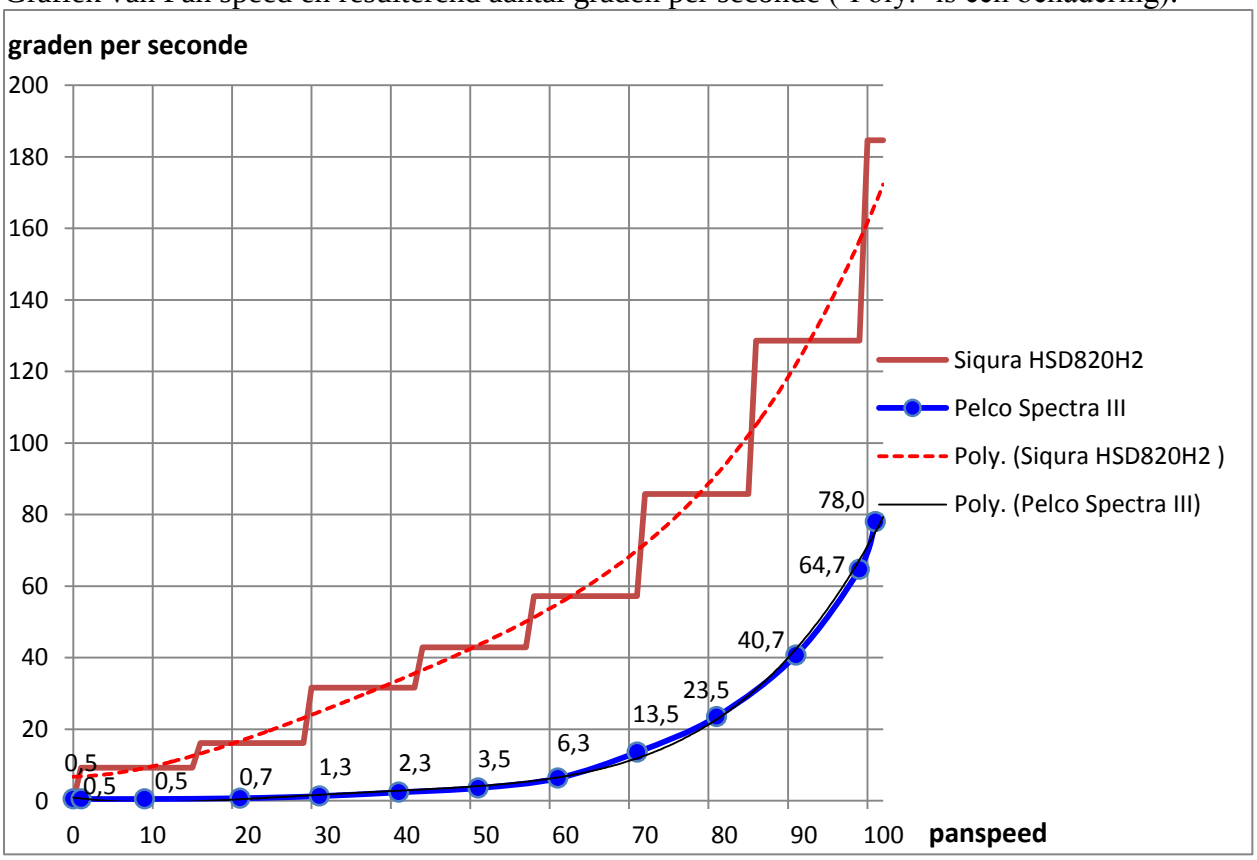
Vertical Tilt:	+10° to -100°
Manual Pan/Tilt Speeds:	Pan 0.5°~90°/s
	Tilt: not specified
Maximum motor speed:	Pan: 400°/s, Tilt: not specified
Zoom Speed (optical range):	not specified
Optical zoom:	20
Horizontal Angle of view:	55.8° at 4.7mm wide zoom, 4° at 94mm telephoto zoom
Max. encoding/resolution:	H.264 1080p30
Image sensor:	1/2.8" progressive CMOS

Voor beide camera’s geldt:

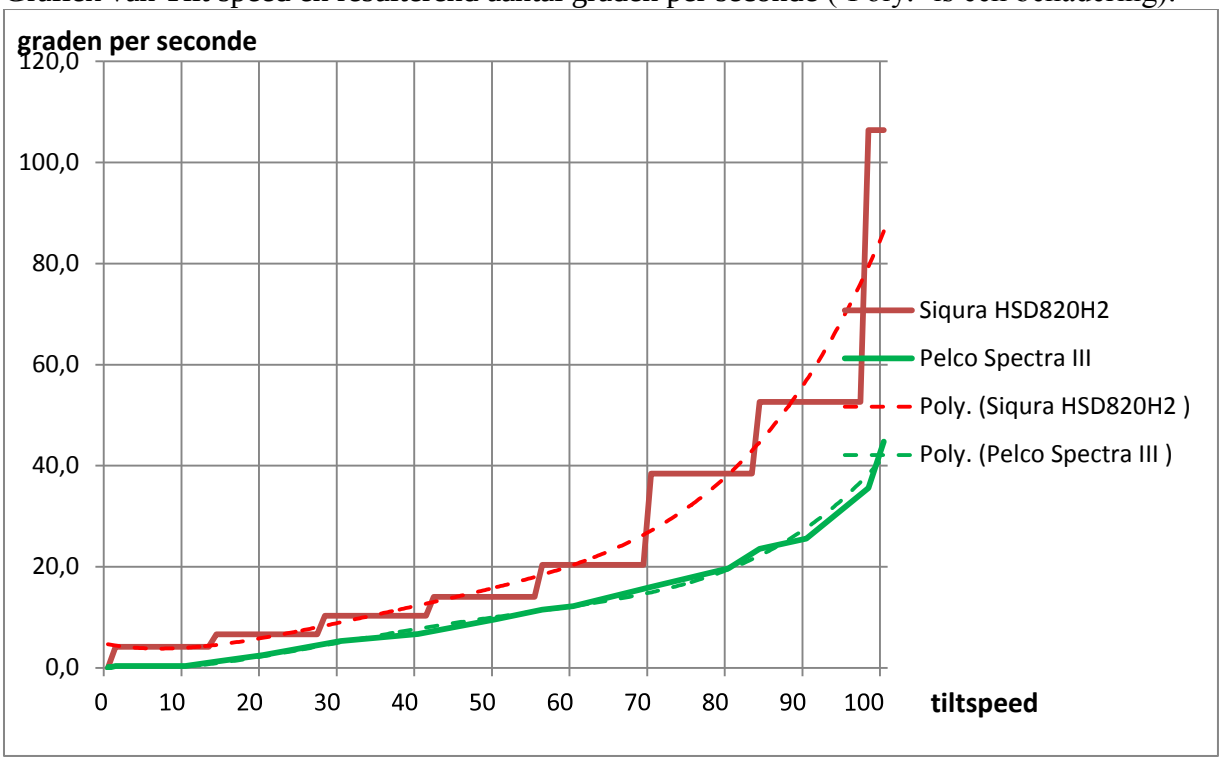
Camera type	High-speed PTZ dome camera
Pan Movement 360°	Continuous pan rotation

Pan and tilt speed proportional to zoom ratio (ON/OFF)

Grafiek van Pan speed en resulterend aantal graden per seconde (‘Poly.’ is een benadering):



Grafiek van Tilt speed en resulterend aantal graden per seconde (‘Poly.’ is een benadering):

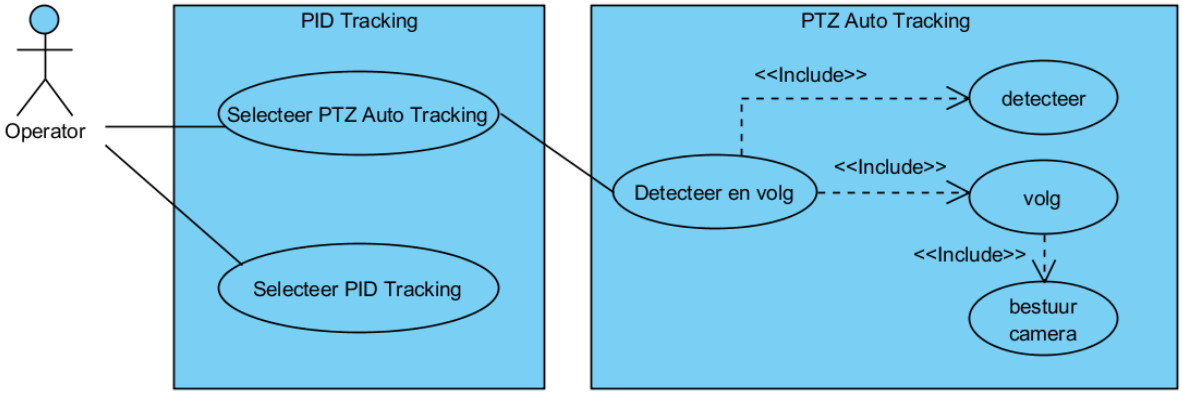


PanSpeed	Graden per seconde	
	HSD820	Pelco
0	0,00	0,00
1	9,23	0,49
10	9,23	0,49
14	16,07	0,49
20	16,07	0,68
28	31,58	1,07
30	31,58	1,27
40	31,58	2,33
42	42,86	2,53
50	42,86	3,53
56	57,14	5,87
60	57,14	6,27
70	85,71	13,53
80	85,71	23,47
84	128,57	28,27
90	128,57	40,67
98	184,62	64,67
100	184,62	78,00

	Graden per seconde	
TiltSpeed	HSD820	Pelco
0	0,0	0,0
1	4,2	0,4
10	4,2	0,4
13	4,2	1,3
14	6,7	1,3
20	6,7	2,5
27	6,7	5,3
28	10,3	5,3
30	10,3	5,3
40	10,3	6,7
41	10,3	6,7
50	14,1	9,6
55	14,1	10,3
56	20,4	11,5
60	20,4	12,2
69	20,4	15,2
70	38,5	16,0
80	38,5	19,7
83	38,5	23,6
84	52,6	23,6
90	52,6	25,6
97	52,6	35,6
98	106,4	35,6
100	106,4	44,8

Bijlage E - UML ontwerp.

Het UML ontwerp is zeer eenvoudig.
In feite bestaat het slecht uit één Use Case: “Detecteer en volg”. Ter verduidelijking is deze uitgesplitst in drie deelactiviteiten (zie Figuur 17). Dit project gaat over “PTZ Auto Tracking”. De aansturing komt vanuit “PID Tracking”. “PID Tracking” valt buiten dit project maar deze is getekend omdat daar een keuze door een operator gedaan wordt.



Figuur 17 Use Case diagram van PTZ Auto Tracking (rechts)

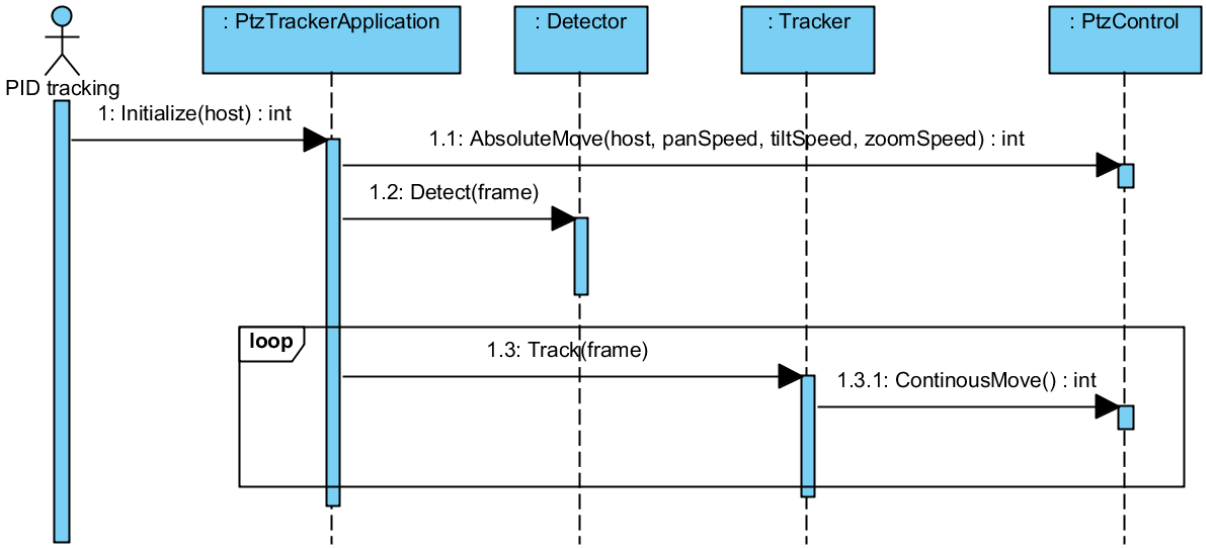
Bij deze Use Case horen de volgende beschrijvingen:

Naam	***Detecteer en volg***
Samenvatting	Een vaartuig in het beeld van de PTZ camera wordt gedetecteerd en gevolgd.
Actoren	Geen
Aannamen	Auto tracking is geactiveerd. De PTZ camera is gericht op een regio waar een vaartuig gedetecteerd dient te worden.
Beschrijving	Een vaartuig in het beeld wordt gedetecteerd. De plaats in beeld wordt bepaald. Het vaartuig wordt in beeld gehouden door het te volgen met de camera.
Uitzonderingen	[einde tracking] Het vaartuig is niet meer zichtbaar. Het volgen wordt gestopt.
Resultaat	De locatie van het vaartuig in het camerabeeld is bepaald. Het vaartuig wordt gevolgd.

Naam	***Detecteer***
Samenvatting	Een vaartuig in het beeld van de PTZ camera wordt gedetecteerd.
Actoren	Geen
Aannamen	Auto tracking is geactiveerd. De PTZ camera is gericht op een regio waar een vaartuig gedetecteerd dient te worden.
Beschrijving	Een vaartuig in het beeld wordt gedetecteerd. De plaats in beeld wordt bepaald.
Uitzonderingen	[timeout] Er is geen vaartuig gedetecteerd.
Resultaat	De locatie van het vaartuig in het camerabeeld is bepaald.

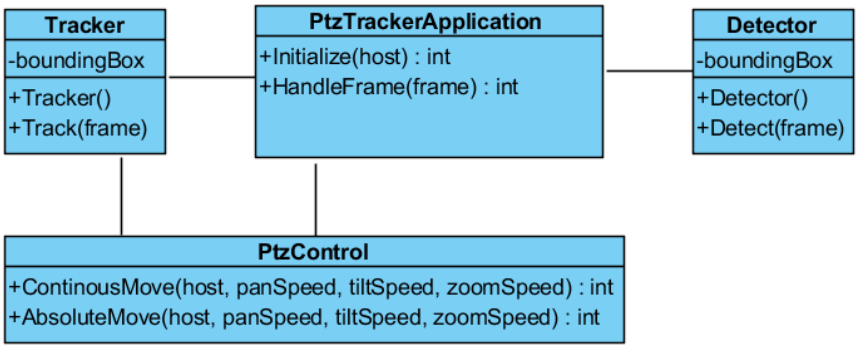
Naam	***Volg***
Samenvatting	Een vaartuig in het beeld van de PTZ camera wordt gevolgd.
Actoren	Geen
Aannamen	Een vaartuig is gedetecteerd en de locatie in beeld is bekend.
Beschrijving	Een vaartuig dat in het beeld is gemarkeerd wordt gevolgd.
Uitzonderingen	[einde tracking] Het vaartuig is niet meer zichtbaar. Het volgen wordt gestopt.
Resultaat	Het vaartuig wordt gevolgd.

In Figuur 18 is de volgorde van handelingen schematisch weergegeven. Er is geen interactie met gebruikers. De applicatie wordt gestart, vanuit PtzTrackerApplication wordt initieel de camera gericht en worden de Detector en Tracker geïnitialiseerd. De detector geeft als resultaat een boundingBox welke aan de Tracker wordt doorgegeven. De Tracker blijft het object in beeld volgen zolang deze zichtbaar is en stuurt hierbij de motoren van de PTZ camera aan.



Figuur 18 Sequence Diagram van PTZ AutoTracking

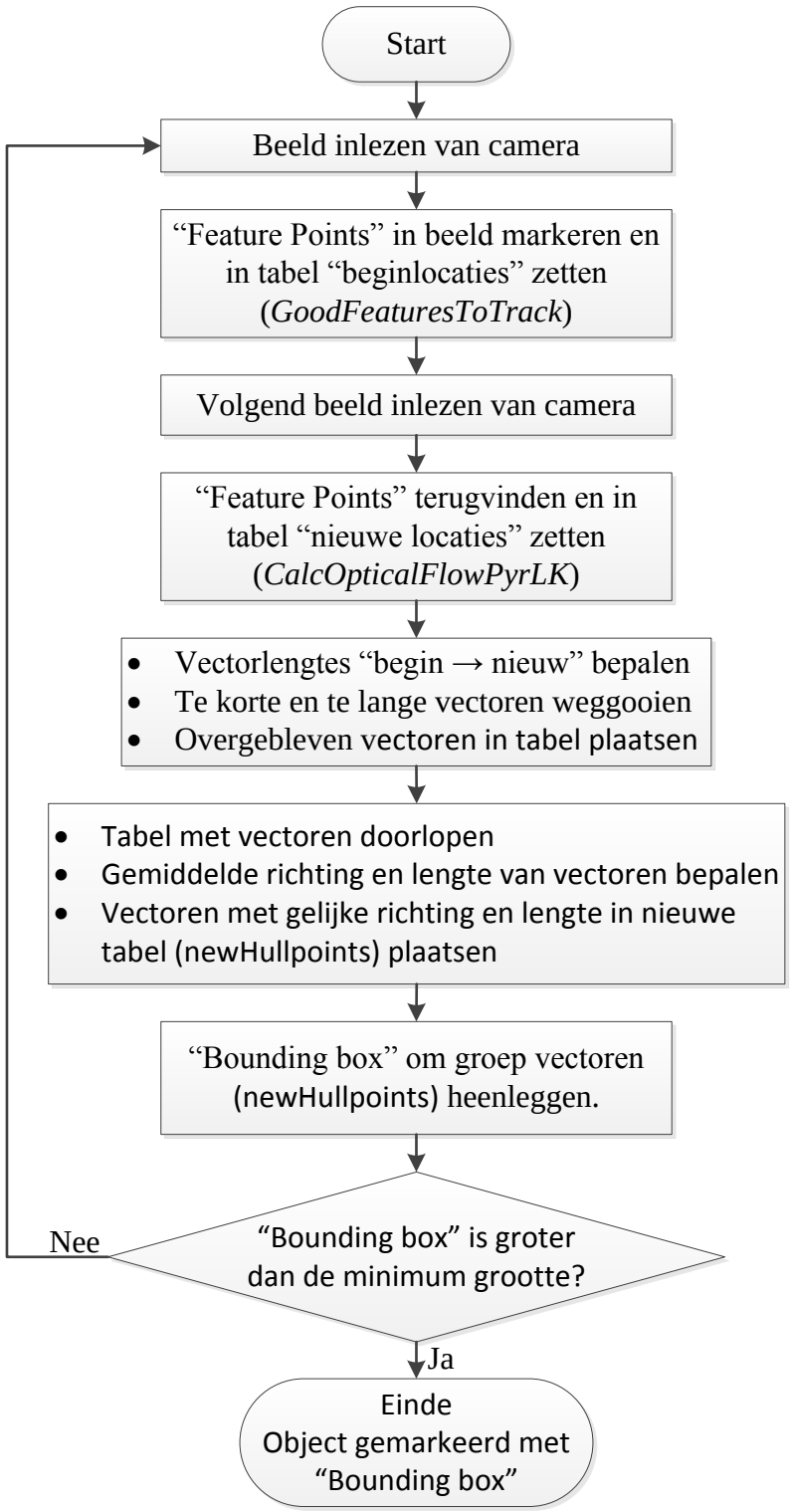
Figuur 19 geeft een overzicht van de benodigde Klassen. Ook dit is zeer eenvoudig gehouden.



Figuur 19 Klasse Diagram van PTZ AutoTracking

Bijlage F – Flowchart van de Lucas Kanade detector

Dit is een bijlage bij hoofdstuk 4.4, pagina 18.



Bijlage F – Metingen aan video streams

Bevindingen m.b.t. processorsnelheid
Op het netwerk bij Siqura zijn de volgende snelheden bereikt:

Met een MJPEG stream van de Pelco camera kan de Core2Duo de resolutie 960x576 bij 25 beelden per seconde aan bij een bitrate van 6Mbit/s (kwaliteit is dan 45%). Een hogere bitrate levert bij deze PC datacorruptie op doordat de processorbelasting de 100% nadert.

De i5 PC’s hebben met deze camera geen beperkingen laten zien qua resolutie en frame-rate.

Metingen gedaan aan Pelco / Eve One videostreams:
Voor onderstaande metingen geldt dat het netwerk bestaat uit een PC, een camera en een Gigabit ethernet switch; dit zijn de optimale omstandigheden.
Tijden van de trackers met de D830 Core2Duo laptop en Pelco stream 1
(MJPEG, 950x576, 25fps. Bitrate 6Mbit/s, Quality 65%)
DSST: 265ms, 273ms, 267ms, 267ms, 267ms => gemiddeld 267ms vertraging
KCF: 379ms, 315ms, 315ms, 330ms, 282ms, 301ms, 348ms => gemiddeld 325ms vertraging

De stream Pelco 2 raakt te veel verminkt om er iets mee te kunnen
(H.264HP, 950x576, 25fps. GOP size = 1 frame, maximum bit rate 6Mbit/s, quality 70%)
Ondanks de datacorruptie toch maar eeen aantal tijden voor een indruk:
DSST: 559ms, 550ms, 532ms, 566ms
KCF: 429ms, 513ms, 528ms, 533ms, 482ms, 415ms

N.B.: Quality is afhankelijk van Bitrate. Een hogere Bitrate staat een hogere Quality toe.

OpenCV 3.0.0. imshow met 1ms WaitKey en verder niets:
201ms, 184ms, 183ms, 231ms, 230ms, 183ms, 179ms, 186ms, 199ms, 186ms, 184ms
Hier zit een duidelijk spreiding in: een groep cijfers rond de 180ms en een groep cijfers rond de 230ms. Dit komt door de 25fps; elke 40ms een nieuw frame
Er is dus 4 a 5 frames vertraging en 24ms verwerkingstijd.