Afstudeerverslag

Het herkennen en locatie voorspellen van knikkers op basis van kleur met een camera en PLC bij Alten Nederland.

Hanno van Megchelen
15077918
HBO-ICT Network & Systems Engineering
De Haagse Hogeschool
Versie 1.0
31-05-2019

ALTEN

Voorwoord

Dit verslag beschrijft het afstudeertraject dat is uitgevoerd door Hanno van Megchelen bij ALTEN Nederland. Tijdens het traject is er een opdracht uitgevoerd voor op het gebied van computer-vision op industriële computers.

Het afstudeerverslag is gericht op de examinatoren van de Haagse Hogeschool. Er wordt uitgelecht wat de afstudeerder tijdens het afstudeertraject heeft uitgevoerd. Hierbij worden keuzes die gemaakt zijn verantwoord.

De afstudeerder betuigt dank aan de begeleiden van ALTEN René van Schie voor de feedback en begeleiding die ik heb gekregen tijden het project.

Capelle a/d IJssel, 31-05-2019
Hanno van Megchelen

Referaat

In verband met het aantonen van kennis op beurzen heeft ALTEN Nederland een opdracht opgesteld om computer-vision uit te voeren op een PLC. Hierbij moet er een knikker worden op worden opgepakt door een robotarm. Om de knikker op te kunnen pakken moet de positie van de knikker worden voorspeld door middel van computer-vision. De afstudeerder heeft in dit project de knikkervoorspelling gerealiseerd op de PLC. Voor het uitvoeren van de opdracht is er eerst een analyse over het onderwerp uitgevoerd. Waarna het er in drie incrementen het vision-algoritme is geïmplementeerd op de PLC.

Na het realiseren van het algoritme werkte het vision-algoritme naar behoren. Er kon door middel van een reeks van afbeeldingen een correcte voorspelling gemaakt worden 900ms voordat de knikker zich op de voorspelde positie zou bevinden. Dit is genoeg tijd voor een robotarm om er naartoe te bewegen. Echter is er bij het uitvoeren van het algoritme op de PLC naar voren gekomen dat de beschikbaar gestelde PLC niet geschikt was voor computer-vision. Deze had niet genoeg rekenkracht om de gemaakte afbeeldingen op tijd te bewerken. Hierdoor zou de voorspelling te laat gemaakt worden.

Aanbevolen wordt om een nieuwere PLC te gebruiken voor het uitvoeren van het algoritme. Hierdoor kan de huidige algoritme zonder aanpassingen correct worden gebruikt.

Inhoudsopgave

Lijst met figuren en tabellen

1. Inleiding

ALTEN Nederland B.V. probeert altijd te groeien. Dit gebeurt in de vorm van aannemen afstudeerders en werknemer en het werven van nieuwe klanten. ALTEN staat hiervoor op beurzen en organiseren kennisdagen. Het is echter niet altijd makkelijk om aan te tonen dat zij een leuk en aantrekkelijk bedrijf is met voldoende kennis is huis. Om dit aan te tonen is een opstelling nodig dat op een aantrekkelijke manier kan laten zien wat ATLEN kan. Hiervoor is er een afstudeeropdracht opgesteld dat kennis laat zien op het gebied van computer-vision.

Voor de opdracht moest er een vision-algoritme op een Bachmann PLC ontwikkeld worden dat een knikker en de kleur ervan op een lineaire knikkerbaan kan detecteren en zijn positie kan voorspellen om in de toekomst de knikker op te pakken met een robotarm. Deze opstelling kan dan op een beurs getoond worden om interesse te wekken bij studenten.

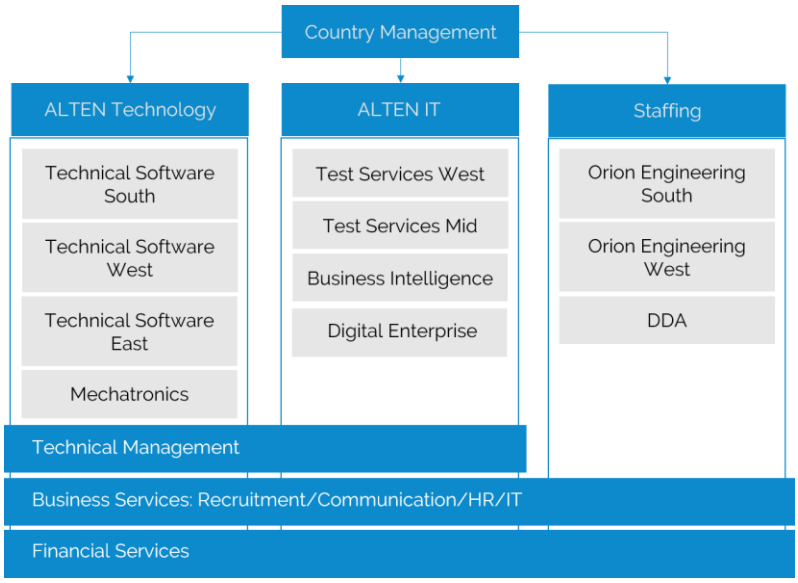
De afstudeeropdracht is in 3 fases uitgevoerd. In de eerste fase is er de opdracht geanalyseerd, waarbij er ontbrekende kennis over het probleemdomein is onderzocht. Ook wordt er in deze fase besproken hoe de eisen van het project zijn opgesteld. Na de analyse is er een fase voor het ontwerpen, realiseren en testen van het product. Dit is uitgevoerd in drie incrementen. Hierbij wordt er besproken welke keuzes zijn gemaakt bij de ontwikkeling van het product. De laatste fase is de afrondingsfase. Hierin wordt er een conclusie gegeven met de behaalde resultaten en een evaluatie of het afstudeerproces goed is verlopen.

2. Organisatie

ALTEN is een technisch consultancy- en engineeringsbureau dat gevestigd is in 25 landen. Het bedrijf is in Frankrijk in 1988 opgericht door de CEO Simen Azoulay. Huidig heeft ALTEN wereldwijd 30.000 werknemers in dienst en heeft een jaarlijkse omzet van 1,975 miljard euro. In Nederland is ALTEN gevestigd op vier locaties:

- Capelle a/d IJssel
- Eindhoven
- Apeldoorn
- Amstelveen

Op deze locaties werken in totaal 1000 werknemers verdeeld over verschillende afdelingen. Deze zijn te zien in het organogram van Figuur 1.



FIGUUR 1 - ORGANOGRAM ALTEN [1]

Binnen ALTEN Technology en ALTEN IT worden opdrachten uitgevoerd binnen verschillende industrieën, zoals Automotive, Hightech Industrie, Defensie/security, Energie, Lucht- en ruimtevaart, Multimedia, Verkeer en vervoer, Telecom. Hierbij geven ze advies en werken ze mee binnen project. Consultants werken steeds bij een andere opdrachtgever. Hierdoor wordt er weinig binnenshuis projecten uitgevoerd.

De afstudeeropdracht wordt uitgevoerd binnen de afdeling 'Technical Software West'. Dit is een afdeling gelokaliseerd in Capelle a/d IJssel. Binnen deze afdeling werken veel consultants die gespecialiseerd zijn in embedded software en automatisering. Hier is een kleinere afdeling speciaal voor afstudeeropdrachten. Ook wordt deze afstudeeropdracht hier uitgevoerd. De uitvoer van de afstudeeropdracht heeft een impact op het aantonen van kennis binnen deze afdeling. Hierdoor kunnen er nieuwe klanten en werknemers geworven worden.

3. Opdracht

3.1 Aanleiding

ALTEN Nederland B.V. probeert altijd te groeien. Dit gebeurt in de vorm van aannemen afstudeerders en werknemer en het werven van nieuwe klanten. ALTEN staat hiervoor op beurzen en organiseren kennisdagen. Het is echter niet altijd makkelijk om aan te tonen dat zij een leuk en aantrekkelijk bedrijf is met voldoende kennis is huis. Om dit aan te tonen is een opstelling nodig dat op een aantrekkelijke manier kan laten zien wat ATLEN kan. Hiervoor is deze afstudeeropdracht opgesteld.

3.2 Probleemstelling

Momenteel worden berekeningen van beeldbewerking niet uitgevoerd op PLC's, maar op een externe computer dat communiceert met de PLC. Dit is echter minder efficiënt, omdat er communicatie moet worden omgezet tussen de externe computer en de PLC. Er is daardoor de behoefte om een vision-algoritme te creëren dat op een Bachmann PLC binnen zijn programmacyclus uitgevoerd kan worden.

3.3 Doelstelling

Het hoofddoel van de opdracht is om in 17 weken een vision-algoritme op een Bachmann PLC te ontwikkelen dat een knikker en de kleur ervan op een lineaire knikkerbaan kan detecteren en zijn positie kan voorspellen om in de toekomst de knikker op te pakken met een robotarm.

3.4 Resultaat

Er is een vision-algoritme op een Bachmann PLC dat de kleur van een knikkers kan herkennen en de positie hiervan kan voorspellen op een knikkerbaan.

3.5 Op te leveren producten

- Plan van Aanpak
- Analyserapport met requirements
- Ontwerprapport
- Testrapport
- Knikker herkenningsalgoritme

3.6 Aanpak

Voordat de afstudeeropdracht van start kon gaan, moest er eerst een ontwikkelmethode gekozen worden om ervoor te zorgen dat het ontwikkelproces gecontroleerd verloopt. In dit hoofdstuk worden de afwegingen en keuzes opgesteld die gemaakt zijn om deze methode te kiezen.

3.6.1 Strategie

Voor het kiezen van een strategie is er de keuze tussen drie mogelijkheden:

- Een lineaire strategie
- Een iteratieve strategie
- Een incrementele strategie

Elk van deze strategieën heeft zijn voor en nadelen. Om een keuze te maken moet de afstudeeropdracht vergeleken worden met de voor- en nadelen van de strategieën.

Bij de afstudeeropdracht is het door gebrek aan kennis van belang dat er een uitgebreide analyse gemaakt wordt van het huidige systeem en de componenten die nodig bij de uitvoeren van de opdracht. Hierbij worden ook de eisen opgesteld. Dit is mogelijk, omdat de opdracht een duidelijk doel heeft en geen grote omvang. Bij het realiseren van de opdracht wordt er telkens gebouwd op vorige onderdelen van het project. Dit betekend dat er pas verder gewerkt kan worden als alle onderdelen van het huidige systeem werken. Dit geld ook als er functies worden toegevoegd. Als laatste wordt het resultaat aan het einde niet geïmplementeerd in het veld.

Met de bovengenoemde eigenschappen van het project kan er gekeken worden naar de strategie die hierbij past. De analyse van dit project gebeurd in één keer volledig. Dit zou een indicatie kunnen geven dat er ook lineair gewerkt kan worden. Echter past een lineaire strategie niet bij de andere eigenschappen van het project. Elk onderdeel dat wordt toegevoegd aan het project moet werken voordat er aan een volgend onderdeel kan worden gewerkt. Hierdoor moet je elk onderdeel los van elkaar ontwerpen, realiseren en testen. Dit is kenmerkend voor een incrementele en iteratieve strategie. Bij iteratief werken is echter het doel van tevoren vaak niet duidelijk. Dit is echter wel zo bij dit project. Daarom is er gekozen voor de incrementele strategie met een uitgebreide eerste analyse. Ook kan deze strategie toegepast worden om een werkend systeem snel te ontwikkelen.

Bron: [2]

3.6.2 Methode

Na de keuze van de incrementele strategie, zijn er een aantal methodes die hierbij passen. Deze zijn als volgt:

- Rational Unified Process (RUP)
- Scrum
- Rapid Application Development (RAD)

Elke methode heeft voor- en nadelen. Aan de hand van deze voor- en nadelen en het soort opdracht is een methode gekozen. Bij de gekozen methode wordt ook het gebruik van de methode binnen het project beschreven.

Rational Unified Process

RUP is een incrementele methode die bedoeld is voor grote projecten waarbij de eisen binnen het project nog kunnen veranderen. Doordat er per increment binnen een groot project veel veranderd is het in het begin nog onzeker wat het eindproduct precies gaat zijn. Hiervoor moet er per increment gekeken worden naar de eisen van het project en probleemdomein. Echter is RUP niet geschikt voor kleine projecten, omdat er extra fases zijn voor het aanpassen van eisen die onnodig zijn voor een klein project. Dit zorgt voor een langere ontwikkeltijd.

Scrum

Scrum is een methode die wordt gebruikt om in een multidisciplinair team in sprints een product te ontwikkelen. Net als bij RUP staan de eisen aan het begin van het project nog niet vast en kunnen daarom na elke sprint aangepast worden. Bij Scrum is het belangrijk dat iedereen in het team weet waar teamgenoten mee bezig zijn. Hierdoor wordt er veel gecommuniceerd tussen collega's. Dit voorkomt onduidelijkheden binnen het productieproces. Na elke sprint volgt een evaluatie van het opgeleverde product. Een nadeel van scrum is dat het er tijd verloren gaat aan de verplichtingen die bij scrum horen. Hierdoor is het eindproduct wat de klant wil, maar kan het langer duren voordat producten ontwikkeld zijn.

Rapid Application Development

RAD is een methode die wordt gebruikt om snel te ontwikkelen. Om snel te kunnen ontwikkelen moeten de eisen van tevoren goed in kaart worden gebracht. Dit gaat moeilijk bij grote projecten, dus deze methode wordt vooral gebruikt bij projecten met een beperkte omvang. Doordat er snel geïtereerd wordt is er veel betrokkenheid van de gebruikers en opdrachtgever. Er kan snel besproken worden of het gemaakte product naar wens is. Een nadeel van deze methode is dat de eerste oplevering vaak laat is, omdat er een lange definitie fase moet worden gehouden om de eisen en het probleemdomein vast te stellen.

Keuze

Voor dit project is er gekozen voor RAD. Deze methode past het best bij dit project aangezien het een relatief klein project is met goed in kaart te brengen eisen. Deze kunnen is het begin van het project vast gesteld worden, zodat er daarna snel geïtereerd kan worden. Dit is nuttig aangezien er een beperkte tijd is op dit project uit te voeren. Bij het ontwikkelen wordt er telkens gebouwd op de resultaten van het vorige increment. Dit is bij dit project van toepassing aangezien er één product gemaakt wordt dat elk increment wordt uitgebreid.

Bron: [2]

Uitvoering RAD

Binnen dit project wordt er een aangepaste versie van RAD gebruikt. Hierbij wordt er een uitgebreide analyse uitgevoerd op missende kennis op de doen en om de eisen vast te stellen. Daarna wordt er incrementeel ontworpen, gerealiseerd en getest. Bij traditioneel RAD wordt er per increment gekeken of de gebruikers van het project tevreden zijn met het product, maar los van de opdrachtgever zijn deze er niet. Hierdoor wordt er alleen een product evaluatie gehouden met de opdrachtgever tijdens het testen. Na het incrementeel werken wordt het product en alle tussenproducten overdragen aan de opdrachtgever. In Figuur 2 wordt er een visueel overzicht gegeven van het gebruik van RAD binnen het afstudeerproject.

FIGUUR 2 - UITVOERING RAD BINNEN AFSTUDEEROPDRACHT [2]

Het bovenstaande figuur wordt in de rest van het verslag gebruikt om aan te geven in welke fase van RAD er gewerkt wordt. Hierbij wordt de overdracht in dit verslag niet behandeld. Hierbij is namelijk alleen de code via GitHub opgeleverd met de bijbehorende documentatie.

3.7 Planning

In dit hoofdstuk wordt de fasering en de planning gegeven, zoals deze is opgesteld in het plan van aanpak (bijlage A).

3.7.1 Fasering

Het afstudeerproject van verdeelt in drie fases. Deze zijn in Tabel 1 weergegeven.

Fasering	Activiteit	Tijdsbestek
Oriëntatiefase	*Maken plan van aanpak* *Maken van (requirement)analyse* *Mogelijk literatuuronderzoek*	*4 weken*
Incrementele werken (Sprints)	*Ontwerpen van software*	*11 weken in iteraties van 2 - 3 weken*
	Implementeren van het ontwerp	
	Testen implementatie van de software	
Afronding	*Afronden project en documentatie*	*1 week afronden*

TABEL 1 - FASERING VAN AFSTUDEERPROJECT

3.7.2 Planning blokschema

Na het opstellen van de fasering binnen het afsturen is er een globale planning gemaakt waarin de faseringen worden verdeeld in de beschikbare tijd binnen het project. De globale planning is in Tabel 2 opgesteld.

Weken	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
Fasen																	
Oriëntatiefase	■	■	■	■	■												
Increment 1						■	■	■									
Increment 2									■	■	■						
Increment 3												■	■	■			
Afronding															■	■	■

TABEL 2 - GLOBALE PLANNING

In deze planning worden geen details gegeven wat er in elke fase uitgevoerd. Ter toevoeging van de globale planning is er na het opstellen van de eisen een increment planning opgesteld waarin de eisen per increment zijn ingedeeld. Deze is te vinden in hoofdstuk 4.6.

4. Analyse

In dit hoofdstuk wordt de analysefase van het afstudeerproject behandeld.

Tijdens de analyse is het probleemdomein en zijn de eisen opgesteld. Daarnaast is er ontbrekende kennis opgedaan die nodig was om keuzes te maken en de opdracht uit te voeren. Als eerste zijn de systeemeisen opgesteld aan de hand van de randvoorwaarden en gesprekken met de opdrachtgever. Aan het einde van de analyse en het opstellen van de functionele eisen is er een incrementplanning opgesteld, waarbij de eisen per increment zijn ingedeeld.

4.1 Systeemeisen

Om achter de systeemeisen te komen is er met de opdrachtgever overlegd wat er belangrijk was bij dit project. De uitkomsten van het gesprek en de grenzen die zijn opgesteld in het plan van aanpak (bijlage A), zijn meegenomen in de systeemeisen. In Tabel 3 zijn de systeemeisen te opgesteld.

Uit de bespreking met de opdrachtgever volgde dat er geen externe systemen gebruikt mogen worden bij het ophalen en bewerken van een afbeelding. Hierdoor moest de camera direct aangesloten kunnen worden aangesloten op de PLC. Daarom moest de camera ondersteund worden door de PLC. Ook was er besloten dat vorm van de knikkerbaan volledig ontwikkeld zou worden door de afstudeerder. Hierdoor zijn er systeemeisen opgesteld door de afstudeerder die eisen stellen aan de knikkerbaan, zodat het herkennen van de knikker mogelijk wordt. Deze zijn ook al besproken in grenzen van het plan van aanpak. Als laatste is er gediscussieerd over de precisie van het vision-algoritme. Hier kwam naar voren dat het gewenst is dat de voorspelling een afwijking mag hebben van één knikker grootte. Hierbij is er uitgegaan van een knikker van 16mm groot. Deze wordt geleverd door de opdrachtgever. Ook is er naar voren gekomen dat de er geen specifieke snelheid is waarop het algoritme uitgevoerd moet worden. Het is belangrijker dat de voorspelling goed wordt uitgevoerd.

ID	Eis	Herkomst eis
SE01	De camera moet kunnen communiceren met de PLC.	Zonder communicatie kan de afbeelding niet verkregen worden, dus kan er geen knikker herkend worden.
SE02	De camera moet ondersteund worden door de PLC.	Volgt uit bijlage B hoofdstuk 3. De PLC heeft beperkte ondersteuning voor camera's.
SE03	De camera kan een kleurenafbeelding maken.	Volgt uit grenzen in plan van aanpak.
SE04	De knikkerbaan mag niet dezelfde kleur zijn als de knikkers.	Volgt uit bijlage B hoofdstuk 4. Als er weinig kleurverschil is, dan kan de knikker niet herkend worden.
SE05	Knikkerbaan moet lineair zijn.	Volgt uit grenzen in plan van aanpak.
SE06	De knikkers mogen niet transparant zijn.	Volgt uit grenzen in plan van aanpak.
SE07	De voorspelling mag er maximaal één knikkergrootte naast zitten.	Gesprekken met de opdrachtgever.

TABEL 3 - SYSTEEMEISEN

Na het opstellen van de systeemeisen is er een analyse uitgevoerd om ontbrekende kennis in te vullen. De kennis die mist is opgedeeld in een aantal te beantwoorden vragen. Deze zijn als volgt:

- Wat zijn de specificaties en mogelijkheden van een Bachmann PLC?
- Welke camera's zijn geschikt voor het uitvoeren van de afstudeeropdracht?
- Welke beeldbewerkingsmethodes zijn er om de knikker te kunnen herkennen?
- Hoe kan de positie van de knikker voorspeld worden?

4.2 Bachmann PLC

De PLC die gebruikt moest worden stond vast, omdat deze al aanwezig was bij de opdrachtgever. Dit is een Bachmann MC205 module met een voeding module. Door te weinig ervaring met deze PLC moest er gekeken worden naar de werking en mogelijkheden hiervan. Hierna kon deze kennis ook gebruikt worden om in een later stadium van het afstuderen keuzes te maken. De specificaties van de MC205 zijn in Tabel 4 weergegeven.

Onderdeel	Specificatie
CPU	600 MHz ATOM E620 (single core)
RAM	1 GB DRAM DDR2
nvRAM-0	512 kB
Intern geheugen	64 MB, 40 MB vrij voor applicatieprogramma
Extern geheugen	CFast flash kaart tot 8 GB
Interfaces	USB 2.0, 2x Seriële poort (COM 1 - 2) en 2x ethernet
Extra functies	
Watchdog	
Synchroniserende puls, ook voor I/O bus, veldbussen SERCOS & CAN, ethernet	
Real-time klok met batterij	
VxWorks besturingssysteem met Bachmann systeem uitbereiding	
3 status LED's	

TABEL 4 - SPECIFICATIES MC205 [3]

Bij dit project was het snelheid van de processor belangrijk, aangezien deze de berekeningen uitvoert van het algoritme. Echter is de snelheid van de processor die in de Bachmann PLC beperkt. Hierdoor zal er gekeken tijdens de testen van het vision-algoritme ook gekeken worden of deze PLC geschikt voor het uitvoeren hiervan.

Bachmann PLC's draaien op Bachmann's eigen besturingssysteem dat gebaseerd is op VxWorks. Dit is een real-time operatie systeem met mogelijkheden voor multi-threading, dat veel gebruikt wordt voor embedded applicaties. Doordat het Bachmann besturingssysteem hierop gebaseerd is, kon er gekeken worden of multi-threading toegepast ging worden bij de uitvoering van het project.

Voor het programmeren op de PLC wordt het programma Solution Center gebruikt. Dit is een op Eclipse gebaseerde IDE. De PLC is met verschillende talen te programmeren. De talen die hieronder vallen zijn:

- Sequentie functie diagram (SFC)
- Functieblokdiagram (FBD)
- Ladder diagram (LD)
- C/C++

Tijdens het ontwerpen is er een keuze gemaakt welke programmeertaal gaat worden gebruikt. Een uitgebreidere analyse van de Bachmann PLC is de vinden in bijlage B.

4.3 Camera's voor PLC

Voordat er aan de realisatiefase gewerkt kon worden, moest er eerst een camera gekozen worden om aan te sluiten op de PLC. Aanvankelijk was er geen rekening gehouden met de problemen die hierbij kwamen kijken. Na onderzoek bleek dat niet alle camera's ondersteund werden door de PLC. Daardoor moest er onverwachts gekeken worden naar verschillende soorten camera's die ondersteund worden. Hierbij is er gekeken naar drie verschillende camera's:

- USB-camera
- IP-camera
- Seriële poort camera

Voor de bovenstaande camera's komen voort uit de aansluitingen die aanwezig zijn op de PLC. Zoals opgesteld is in Tabel 4. Voor het kiezen van een camera is er gekeken naar de snelheid van de dataoverdracht, de compatibiliteit met de PLC en extra apparatuur die nodig is voor het aansluiten.

4.3.1 USB-camera

Een USB-camera wordt aangesloten op een apparaat door middel van het USB-protocol. Dit is een serieel protocol dat de opvolger is van de seriële poort. Veel apparaten ondersteunen USB aangezien het een industriële standaard is. Echter zijn er vaak drivers nodig om randapparatuur te kunnen gebruiken via USB. In Tabel 5 in een overzicht opgesteld met de eigenschappen van een USB-camera die belangrijk zijn voor het kiezen van een camera.

Snelheid dataoverdracht	60MB/s
VxWorks drivers	Weinig beschikbaar
Extra hardware	Geen
Kosten	Laag

TABEL 5 - EIGENSCHAPPEN USB-CAMERA

Het grootste voordeel van een USB-camera is de snelheid. Bij USB 2.0, welke de Bachmann ondersteund, is de snelheid van de dataoverdracht 60MB/s. Met deze snelheid kan een niet gecomprimeerde afbeelding met een resolutie van 640*480 in 15 milliseconde verstuurd worden. Daarnaast zijn USB-camera's vaak goedkoop en vereisen geen extra apparatuur.

Het grootste nadeel van een USB-camera is dat de meeste drivers niet worden ondersteund door VxWorks. Hierdoor zouden er handmatig driver gemaakt moeten worden om een afbeelding op te vragen van een camera. [4] De ontwikkeltijd is hierdoor lang.

4.3.2 IP-Camera

Een andere optie voor het aansluiten van een camera op de PLC is een IP-camera. Deze camera's worden aangesloten door middel van een ethernet verbinding. IP-camera's worden vaak gebruikt als beveiligingscamera's, omdat ze een videostream over het internet kunnen versturen. Hierdoor kan deze stream op een andere locatie ontvangen worden en opgeslagen. Ook kunnen veel camera's een enkele afbeelding versturen door middel van het HTTP-protocol. In Tabel 6 in een overzicht opgesteld met de eigenschappen van een IP-camera die belangrijk zijn voor het kiezen van een camera.

Snelheid dataoverdracht	12,5 MB/s
VxWorks drivers	Geen drivers nodig
Extra hardware	Mogelijk Power over Ethernet switch
Kosten	Hoog

TABEL 6 - EIGENSCHAPPEN IP-CAMERA

De IP-camera is verbonden met de PLC met een 100Mbps (12,5MB/s) verbinding. Het verzenden van dezelfde afbeelding als bij de USB-camera duurt 74ms. Dit is langzamer dan een USB-camera. Ook zijn IP-camera's vaak duurder, aangezien de productie vaak duurder is. Ook kunnen er extra kosten bij komen als de stroomkabel niet lang genoeg is. Dan moet er een power over ethernet switch gekocht worden.

Ondanks de bovenstaande nadelen is het grootste voordeel dat er geen driver nodig zijn voor het opvragen van een afbeelding. Veel IP-camera's kunnen namelijk via HTTP een afbeelding versturen. Elk apparaat dat een TCP/IP-verbinding kan maken en een HTTP-verzoek kan genereren, kan afbeeldingen ontvangen. Dit maakt een IP-camera makkelijk om mee te werken. De afbeelding wordt vaak verstuurd als jpg-afbeelding.

4.3.3 Seriële poort camera

Communicatie via een seriële poort [5] is de voorganger van USB. Hierdoor wordt deze interface bijna niet meer gebruikt op commerciële computers. Waar de seriële poort nog wel te vinden is, is in de industrie waar de technologie vaak ouder is. Ook wordt deze interface gebruikt voor routers en switches voor de configuratie hiervan. De camera's [6] die met deze interface communiceren, zijn heel simpel en worden vooral gebruikt in kleine embedded systemen. In Tabel 7 zijn de eigenschappen van een Seriële poort camera gegeven.

Snelheid dataoverdracht	115,2 Kbit/s -> 14,4 KB/s
VxWorks drivers	Geen drivers nodig
Extra hardware	Convertor voor goede aansluiting
Kosten	Laag

TABEL 7 - EIGENSCHAPPEN SERIËLE POORT CAMERA

Voordelen van deze camera zijn dat er driver nodig is om de camera aan te sturen en dat de prijs laag is. Echter is de snelheid van de dataoverdracht erg laag. Met 14,4 KB/s duurt het maar liefst 64 seconde om de afbeelding te verzenden. In die tijd is de knikker al klaar met rollen van de knikkerbaan. Ook kan de camera niet direct aangesloten worden op de PLC en moet er een convertor gekocht worden.

4.3.4 Keuze Camera

Van de bovenstaande besproken camera's moest een keuze gemaakt worden, welke gebruikt gaat worden bij de afstudeeropdracht. Hiervoor zijn alle voor- en nadelen tegen elkaar opgezet. Elke eerder besproken eigenschap van de camera's heeft een waarde **--**, **-**, **+** of **++** gekregen, deze is relatief. Zo wordt er een **+** gegeven aan een eigenschap die voldoende is en een **++** aan eigenschap die voldoende is, maar beter dan een **+**. Deze waardering is in Tabel 8 te vinden.

Camera	Snelheid dataoverdracht	VxWorks drivers	Extra hardware	Kosten	Totaal
USB	++	--	++	+	+++
IP	+	++	+	-	+++
Seriële poort	--	++	-	+	**0**

TABEL 8 - BEOORDELING CAMERA'S

De beoordeling zijn gebaseerd op de eigenschappen van de camera's die eerder in dit hoofdstuk beschreven zijn. Bij het maken van een keuze is er in de eerste instantie naar het totaal gekeken. Dit is berekend door de **+** tegen een **–** weg te strepen. Hieruit volgde dat de seriële camera afviel en dat de USB- en IP-camera dezelfde totale beoordeling hebben. Doordat deze camera's dezelfde totaalwaarde hadden, moest er onderscheid gemaakt worden tussen het belang van de verschillende eigenschappen.

Bij de uiteindelijke keuze tussen een USB-camera en IP-camera is er gekeken naar het belang van verschillende eigenschappen. Hierbij is er voor gelet op de snelheid dataoverdracht en de compatibiliteit met VxWorks. Deze twee eigenschappen zijn belangrijk voor het behalen van de afstudeeropdracht. De extra hardware en kosten zijn vooral belangrijk voor ALTEN. Bij de vergelijken van de compatibiliteit en dataoverdracht is er gekeken of één van de eigenschappen onvoldoende was. In Tabel 8 is te zien dat de USB-camera absoluut niet compatibel is met VxWorks. Aangezien de snelheid dataoverdracht bij de IP-camera wel voldoende is, is er uiteindelijk gekozen voor de IP-camera. Deze is compatibel met VxWorks en er wordt verwacht dat de snelheid dataoverdracht voldoende is.

4.4 Beeldbewerkingsmethodes

Na het kiezen van een camera is er gekeken naar mogelijk beeldbewerkingsmethodes die gebruikt konden worden voor het vinden van een knikker. Tijdens de analyse was er nog geen keuze gemaakt welke uiteindelijk worden gebruikt, maar er is naar gekeken ter oriëntatie en voor invulling van missende kennis. In dit deelhoofdstuk worden de besproken methodes kort herhaald. In het analyserapport (bijlage B) zijn deze uitgebreider behandeld. Doordat deze missende kennis in de analyse al is opgehaald, kon er later in het project meer gefocust worden op het ontwerpen en implementeren.

4.4.1 Grayscale-conversie

De eerste beeldbewerkingsmethode die is behandeld is grayscale-conversie. Dit is een methode die vaak als eerste wordt toegepast bij beeldbewerking. Bij deze methode wordt namelijk een kleurenafbeelding omgezet naar een zwartwitafbeelding. Zwartwitafbeeldingen zijn vaak makkelijker te bewerken dan kleurenafbeeldingen, omdat elke pixel in een zwartwitafbeelding bestaat uit een enkele waarde van 0 tot 255 en een pixel in een kleurenafbeelding bestaat uit drie waardes van 0 tot 255, namelijk rood, groen en blauw.

Grayscale-conversie bestaat vaak uit de volgende drie stappen:

1. *Het verwijderen van de gammacompressie*
2. *Het berekenen van de grijswaarde van een pixel*
3. *(Optioneel) Het uitvoeren van gammacompressie op de grijswaarde*

Gammacompressie is de waarde die ervoor zorgt dat een afbeelding op een beeldscherm er hetzelfde uitziet als in het echt. Voor een goede zwartwitrepresentatie moet dit er eerst afgehaald worden, waarna de grijswaarde van elke pixel berekend kan worden. Deze formule is te vinden in bijlage B.

4.4.2 JPG-compressie en decompressie

Een IP-camera stuurt een afbeelding als JPG-afbeelding. Hierdoor is deze een stuk kleiner, maar moet deze wel eerst gedecodeerd worden naar een afbeelding die te bewerken is. Hiervoor moest er gekeken worden naar de opbouw van een JPG-afbeelding en de manier van compressie en decompressie. JPG-compressie is een vrij ingewikkelde vorm van compressie als het vergeleken wordt met bijvoorbeeld PNG. De volgende stappen moeten worden uitgevoerd:

1. *Kleurenformaat van RGB veranderen naar YCbCr (kleurenformaat gebaseerd op lichtintensiteit i.p.v. kleur)*
2. *Sub-sampling van Cb en Cr delen*
3. *Omzetten naar frequentiedomein door middel van discrete cosinus transformatie*
4. *Verwijderen van hoge frequenties*
5. *Herorganiseren van waardes voor beter compressie*
6. *Huffmancodering uitvoeren*

Om van een JPG-afbeelding een bewerkbare RGB-afbeelding te maken, moet de stappen achterstevoren en tegenovergesteld uitgevoerd worden.

4.4.3 Smoothing

Nadat greyscale-conversie is uitgevoerd op een afbeelding worden er daarna vaak filtermethodes uitgevoerd om bepaalde elementen uit een afbeelding te onderscheiden van de rest. Filtermethodes [7] werken door met een filter over alle pixels in de afbeelding te gaan en elke pixel te bewerken op basis van het filter en de omliggende pixels. De eerste filtermethode waarnaar is gekeken, is smoothing.

Smoothing is een techniek om oneffenheden uit een afbeelding te halen. Hierbij word een pixel bewerkt door het gemiddelde te berekenen met de omliggende pixels en deze waarde te gebruiken als nieuwe

waarde van de pixel. Hierdoor wordt de afbeelding minder “scherp”, maar worden waardes die onverwachts uitschieten verwijderd uit de afbeelding. Een variant van smoothing door gemiddeldes is smoothing door de mediaan te gebruiken. In Figuur 3 is het resultaat gegeven van smoothing.

FIGUUR 3 - VOOR EN NA SMOOTHING [8]

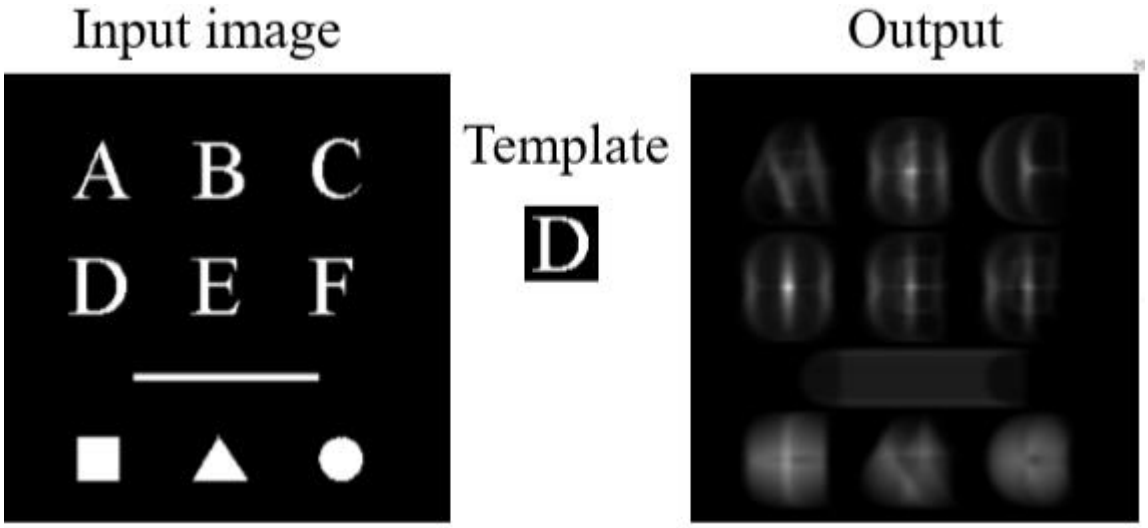
4.4.4 Edge-detectie

Edge-detectie is een methode om het verschil tussen pixels te meten. Op plekken waar het verschil groot is wordt de waarde van de te bewerken pixel ook hoog en als het verschil laag is wordt nieuwe waarde ook laag. Hierdoor wordt een afbeelding gecreëerd dat lijnen heeft op plekken waar het contrast hoog is. Dit is vaak om objecten heen. In Figuur 4 is het resultaat van deze methode gegeven.

FIGUUR 4 - VOOR EN NA EDGE-DETECTIE [9]

4.4.5 Template matching

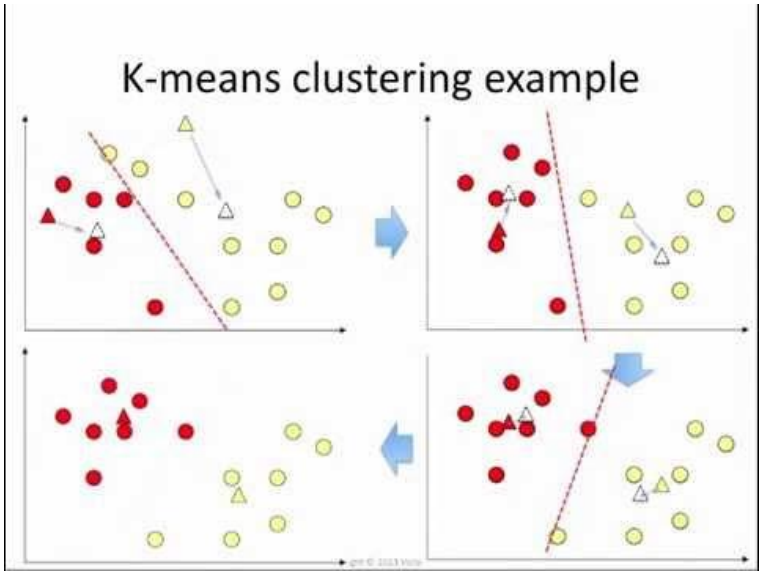
Template matching is een filtermethode om specifieke voorwerpen te vinden in een afbeelding. Hierbij worden delen van de afbeelding vergeleken met een zogenoemde template, het object dat gedetecteerd moet worden. De template is tevens het filter voor de methode. Het filter gaat over de hele afbeelding heen en kijkt elke stap hoeveel pixels in de afbeeldingen overeenkomen met de pixels van het filter. Hoe meer pixels er overeenkomen hoe hoger de waarde in de bewerkte afbeelding. Op de plek met de hoogste waarde in de bewerkte afbeelding is de kans het grootst dat daar het gewenste object is. In Figuur 5 is het resultaat van template matching gegeven.



FIGUUR 5 - VOOR EN NA TEMPLATE MATCHING [9]

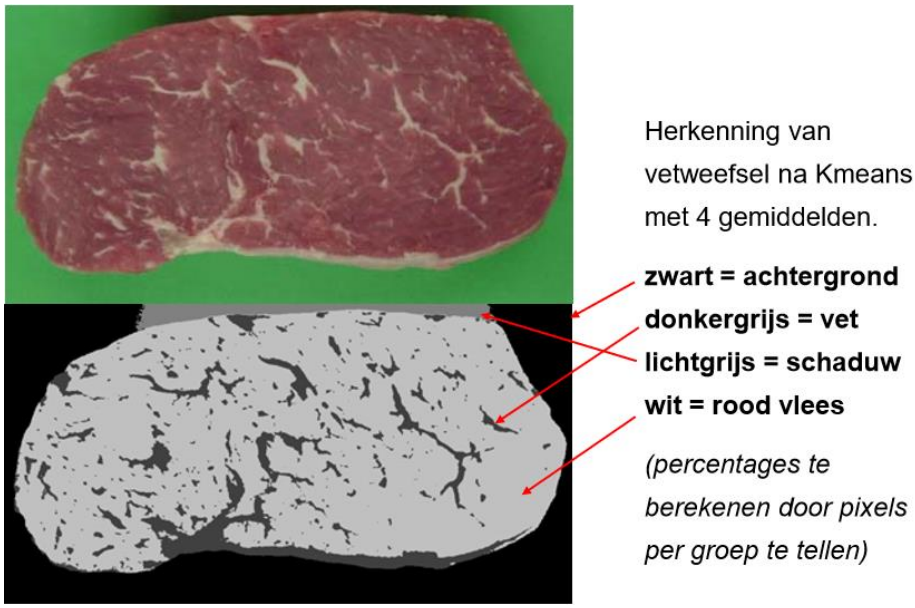
4.4.6 K-means

K-means is een methode om data te groeperen. Waardes van pixels worden dynamisch ingedeeld in een X aantal groepen bepaald door K. Elke groep bestaat uit waardes die het dichts bij elkaar in de buurt zitten. Na het indelen van de groepen krijgen alle pixels die ingedeeld zijn in een groep dezelfde waarde. De werking van K-means is te vinden in Figuur 6.



FIGUUR 6 - WEKRING K-MEANS

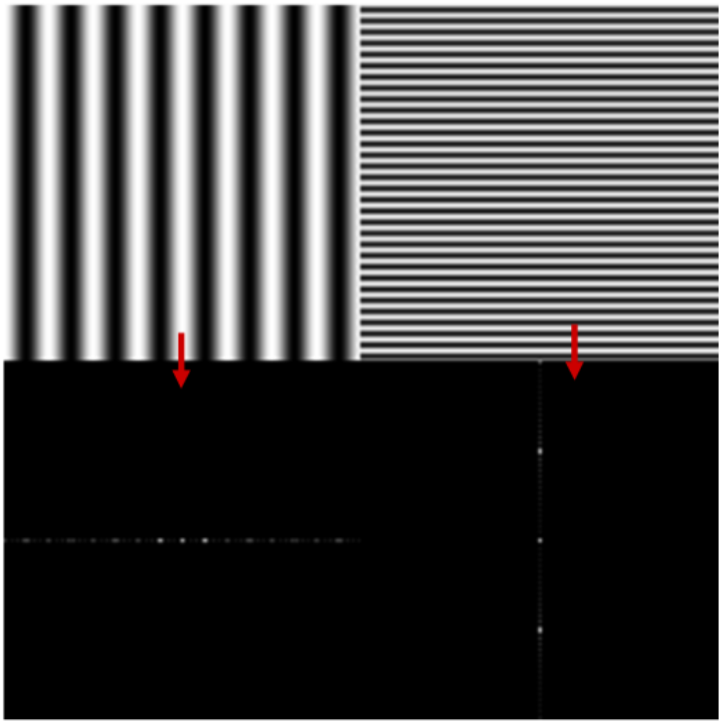
In het voorbeeld moet de data opgedeeld worden in twee groepen. Voor elke groep wordt er een variabele aangemaakt. Aan deze variabele wordt data toegekend dat is de buurt zit van dit punt. Van deze groep wordt het gemiddelde berekend. Dit gemiddelde wordt de nieuwe waarde van de aangemaakte variabele. Dit proces herhaalt zich tot de variabele niet meer verandert. Een uitgebreidere uitleg is gegeven in bijlage B. Het resultaat van K-means op een afbeelding is te vinden in Figuur 7.



FIGUUR 7 - RESULTAAT K-MEANS MET K = 4 [9]

4.4.7 Fouriertransformatie

Elke afbeelding bestaat uit een optelling van horizontale en verticale golven. Deze golven kunnen gesplitst worden in door middel van een fouriertransformatie. Het resultaat van de fouriertransformatie is een afbeelding met een representatie van de golven die aanwezig zijn en in welke mate een frequentie voorkomt. Een voorbeeld van een fouriertransformatie is de discrete fouriertransformatie. Deze maakt een afbeelding in het frequentiedomein. Het resultaat van de DFT is te vinden in Figuur 8.



FIGUUR 8 - RESULTAAT DFT [9]

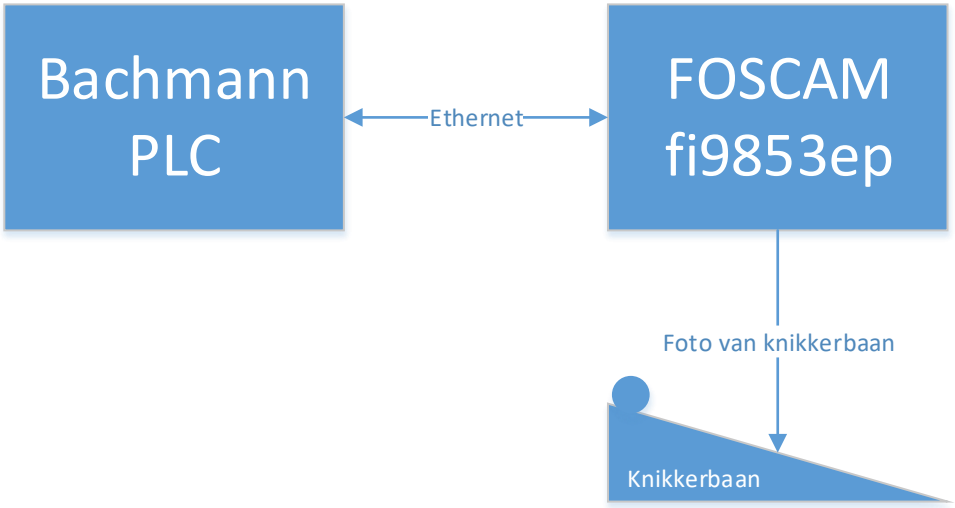
De frequenties worden gerepresenteerd door pixelswaardes. Hoe dichter bij het midden van de afbeelding hoe lager de frequentie. Een afbeelding kan ook bewerkt worden in het frequentiedomein. Bijvoorbeeld kan smoothing door de hoge frequenties uit het frequentiedomein te halen en kan edge-detectie uitgevoerd worden door lage frequenties weg te halen.

4.5 Projecteisen

Doordat er aan het begin van de opdracht te weinig kennis was over de te gebruiken apparatuur, zijn de eisen opgesteld na de analyse. Hierdoor konden de eisen opgesteld worden met behulp van de analyse en in overleg met de opdrachtgever. Eerst zijn de systeemeisen opgesteld. Aan de hand hiervan is de probleemruimte opgesteld. Waarna de functionele eisen zijn opgesteld.

4.5.1 Probleemdomein

Na het opstellen van de systeemeisen en het uitvoeren van de analyse is het probleemdomein opgesteld waarbij er met de keuze van de camera is gekeken welk merk camera zou worden gebruikt. De camera die is gekozen is de FOSCAM fi9853ep. Dit is een IP-camera die verbonden kan worden met de Bachmann PLC. De keuze voor deze camera en de instellingen hiervan worden later in dit deelhoofdstuk behandeld. Het probleemdomein is opgesteld in Figuur 9.



FIGUUR 9 - PROBLEEMDOMEIN

Het probleemdomein bestaat uit drie onderdelen. De PLC, de camera en de knikkerbaan. Hierbij moest de PLC gebruikt worden, omdat deze beschikbaar was gesteld door de opdrachtgever en goed gebruikt kan worden voor computer-vision doordat er C++ op geprogrammeerd kan worden. De IP-camera is gekozen voor de compatibiliteit met de PLC en de knikkerbaan moest nog gemaakt worden.

FOSCAM fi9853ep

Bij het opstellen van het probleemdomein moest er gekeken worden naar een IP-camera voor het uitvoeren van de opdracht. Hierbij moest er gelet worden op de volgende eigenschappen:

- De camera ondersteund HTTP voor het verzenden van afbeeldingen.
- De camera heeft een 100Mbit/s ethernet aansluiting.

Met deze twee eigenschappen kan de camera worden aangesloten op de PLC en kunnen er afbeeldingen worden opgehaald. Om een geschikte camera uit te zoeken is er gekeken naar populaire IP-camera's met goede reviews. Daarna is er gekeken of deze camera's HTTP ondersteunen. Op deze manier kwam het merk FOSCAM naar voren. Bij dit merk is er duidelijk aangegeven hoe er een afbeelding opgehaald kan worden via HTTP. Hierdoor is het makkelijk om ermee te werken. Daarnaast zijn er bij deze camera's verschillende manier om via HTTP afbeeldingen te verkrijgen. Er kan namelijk een losse afbeeldingen worden binnengehaald door middel van een snapshot, maar er kan ook een reeks van afbeeldingen worden binnengehaald door middel van een stream. Om bij dit merk een camera te kiezen is er gekeken naar gebruikers gemak en prijs. Uiteindelijk is er voor de FOSCAM fi9853ep [10] gekozen.

Deze camera kan boven de knikkerbaan worden gehangen en heeft alle HTTP mogelijkheden die nodig zijn. Daarnaast is de prijs van deze camera relatief laag vergeleken met andere IP-camera's. Zoals eerder is genoemd worden afbeeldingen bij IP-camera's vaak verzonden als jpg-afbeelding. Dit is ook het geval bij deze camera. Er kon geen ander formaat gekozen worden. Echter kon er wel gekozen worden met welke resolutie de afbeelding wordt gemaakt.

Doordat de PLC beperkte rekenkracht heeft, is er voor een resolutie van 640x480 (VGA) gekozen. Dit is het de minimale resolutie is de camera ondersteund. Deze resolutie is te gebruiken aangezien er weinig detail nodig is voor het herkennen van een knikker. Er is weinig detail nodig doordat het vereist is dat er een groot contrast is tussen de knikkerbaan en knikker.

Knikkerbaan

Naast de FOSCAM fi9853ep moest ook de knikkerbaan opgenomen worden in het probleemdomein. Hiervoor moest er besloten worden hoe de knikkerbaan eruit zou komen te zien. Er was al bekend dat de knikkerbaan lineair is. Er moest echter nog wel gekeken worden naar:

- Materiaal
- Kleur
- Hellingshoek
- Lengte

Eerst is er gekeken naar het materiaal en de kleur van de knikkerbaan. Hiervoor is er bij een bouwmarkt gekeken naar geschikte onderdelen waar een knikker overheen kan rollen. Hierbij moest er worden uitgegaan van een witte knikker van 16mm. Deze was geleverd door de opdrachtgever. Na zoeken is er gekozen om van een hoeklat [11] te gebruiken. Deze is een recht stuk hout en kan makkelijk geverfd worden om hem de goede kleur te geven. Aangezien de knikker wit was, is ervoor gekozen om de knikkerbaan zwart te maken. Hierdoor is er een groot contrast tussen de twee. Ook is er met de geleverde knikker voldaan aan de eis SE06.

Nadat het materiaal en kleur was gekozen, moest de hellingshoek en lengte bepaald worden. Er is geen precieze hellingshoek gekozen, maar er is voor gezorgd dat de knikker er een aantal seconde over zou doen om over de gehele knikkerbeen te rollen. Uiteindelijk is er besloten dat de knikkerbaan was 50cm lang zou worden met een hellingshoek ongeveer 5 graden. Dit resulteerde uiteindelijk in een afdaling van 2,5 seconde.

4.5.2 Functionele eisen

De functionele eisen zijn opgesteld om de te behalen functionaliteit vast te stellen. Deze zijn grotendeels opgesteld in overleg met de opdrachtgever. Voor de opdrachtgever waren de volgende wensen van belang:

- Een herkenningsalgoritme voor het herkennen van een knikker.
    - Mogelijk ook het herkennen van kleur
- Een voorspellingsalgoritme voor het voorspellen van de knikkerpositie.
- Dat het algoritme op de Bachmann PLC draait en werkt.

Deze wensen zijn omgeschreven tot eisen. Hierbij zijn er ook eisen opgesteld die nodig zijn om de wensen te kunnen realiseren. Daarnaast zijn er eisen opgesteld die voortkomen uit het probleemdomein. De eisen zijn geordend op prioriteit. Hiervoor is de MoSCoW methode gebruikt. Eisen die nodig om de kern van het vision-algoritme te realiseren hebben een hogere prioriteit gekregen. De geordende eisen zijn te vinden in Tabel 9.

ID	Eis	Prioriteit	Herkomst
FE01	Het vision-algoritme kan een non-transparante knikker herkennen.	M	Wensen van de opdrachtgever
FE02	Het vision-algoritme kan de positie van de knikker bepalen.	M	Nodig voor het voorspelling van knikkerpositie
FE03	Het vision-algoritme kan de plek van een knikker op een knikkerbaan kan voorspellen.	M	Wensen van de opdrachtgever
FE04	Het vision-algoritme kan op de Bachmann PLC worden uitgevoerd in de programmacyclus	M	Wensen van de opdrachtgever
FE05	Het vision-algoritme kan de afbeelding van JPG naar een bewerkbaar beeld omzetten.	M	IP-camera verzendt afbeelding in JPG-formaat
FE06	De PLC moet een afbeelding kunnen krijgen van de IP-camera.	M	Een afbeelding is nodig voor de uitvoering van het algoritme
FE07	De PLC moet een output genereren waarbij het resultaat van het vision-algoritme getoond wordt.	M	Nodig om werking vision-algoritme aan te tonen
FE08	Het vision-algoritme kan de kleur van de knikker kan herkennen.	S	Wensen van de opdrachtgeer
FE09	Het vision-algoritme voorspelt alleen de positie van knikker met de verwachte kleur.	S	Wensen van de opdrachtgever
FE10	De PLC moet een cartesisch coördinaat genereren op basis van de voorspelde knikkerpositie in de afbeelding.	C	Voor het interacteren met knikker door extern systeem.
FE11	De knikkervoorspelling moet gecontroleerd kunnen worden door middel van een interface.	C	Voor start/stop en aangeven van systeemvariabelen.

TABEL 9 - FUNCTIONELE EISEN

Deze eisen zijn besproken met en goedgekeurd door de opdrachtgever.

4.6 Incrementplanning

Nadat de functionele en systeemeisen waren opgesteld, moesten deze verdeeld worden in de uit te voeren incrementen. Hierbij zijn alleen de eisen ingedeeld niet nog uitgevoerd moesten worden. Aangezien de knikkerbaan nog gemaakt moest worden, zijn deze systeemeisen ook ingedeeld in deze planning. In Tabel 10 is de incrementplanning te vinden.

Increment	Eisen
1: Ophalen Afbeelding	• FE05 • FE06 • SE04 • SE05
2: Positie bepalen knikker	• FE01 • FE02 • FE07
3: Positie voorspellen knikker	• FE03 • FE04 • FE08 • FE09
Extra	• FE10 • FE11

TABEL 10 - INCREMENTPLANNING

De naamgeving van elk increment komt overeen met de uitgevoerde eisen binnen het increment. Hierbij is de indeling van de eisen voortgekomen uit de prioriteit en afhankelijkheid van de eisen. Namelijk voordat de positie voorspeld kan worden, moet eerst de positie bepaald worden. Echter moet er überhaupt een afbeelding zijn om te kunnen bewerken.

5. Increment 1: Ophalen afbeelding

Increment 1 was de start van het incrementele werken. Hierin werden de eisen gerealiseerd. Bij de uitvoering is er per increment ontworpen, gerealiseerd en getest. In het eerste increment lag de focus op het verkrijgen van een bewerkbare afbeelding. De eisen voor dit increment zijn in Tabel 11 opgesteld.

Increment	Eisen
1: Ophalen Afbeelding	• FE05 • FE06 • SE04 • SE05

TABEL 11 - EISEN INCREMENT 1

5.1 Ontwerp

In dit hoofdstuk wordt de ontwerpfase van het eerste increment beschreven.

Tijdens het project is er in C++ geprogrammeerd. Dit is een object georiënteerde programmeertaal. Een veel gebruikte ontwerpmethode voor het ontwerpen van object georiënteerde programma's is UML. UML is een ISO standaard voor object georiënteerd modeleren. Het is heel visueel en er kan snel en overzichtelijk een ontwerp gemaakt worden dat goed begrepen wordt door anderen die UML kennen. Er wordt functionaliteit beschreven en er worden verbanden tussen onderdelen in de code gevisualiseerd. Door dat UML een standaard is, wordt in de praktijk veel gebruikt. Hierdoor is het ontwerp door andere mensen beter te begrijpen en te onderhouden. Daarom is de keuze gemaakt om UML te gebruiken.

Bij UML moet er rekening worden gehouden dat alleen de structuur van de code wordt aangegeven. De werkelijke invulling van de code wordt hiermee niet beschreven. Hierdoor duurt kan de realisatie langer duren, omdat er dan nog niet bekend is hoe functies geprogrammeerd kunnen worden.

Voor het ontwerp van het systeem zijn er per increment drie diagrammen opgesteld. Een use-casediagram, een klassenmodel en een sequentiediagram. Voor dit increment worden deze in dit hoofdstuk behandeld en zijn ook te vinden in het ontwerprapport (Bijlage C).

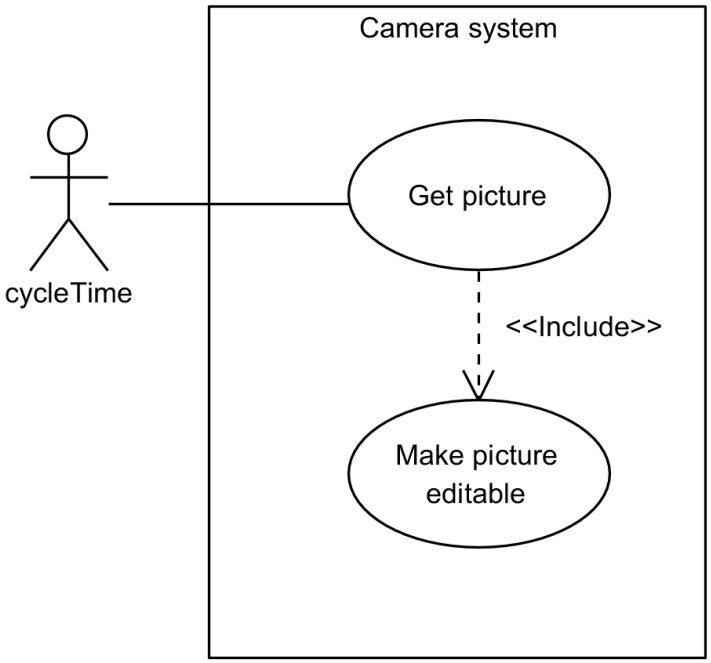
5.1.1 Use-casediagram

Voor het ontwerpen van dit increment zijn eerst de eisen opgeschreven naar use-cases. Deze zijn opgesteld in een use-case diagram. Daarnaast is er per use-case een beschrijving gemaakt dat de stappen van de software beschrijft die uitgevoerd moeten worden.

Bij het opstellen van de use-cases is er gelet op de functionele eisen die bij dit increment horen.

- Het ophalen van een afbeelding bij de IP-camera
- Het decoderen van een JPG-afbeelding naar een RGB-afbeelding

Het use-casediagram is opgesteld in Figuur 10.



FIGUUR 10 - USE-CASEDIAGRAM INCREMENT 1

In het use-casediagram is te zien dat de cyclustijd van de PLC de het programma start. Dit betekend ook dat elke cyclus van de PLC een afbeelding wordt opgevraagd. Naast het starten van het programma, is er geen externe interactie met het systeem. Er is gekozen om 'Make picture editable' te includeren in 'Get picture', omdat deze wordt uitgevoerd direct na het ophalen van de afbeelding en niet wordt aangeroepen door de cyclustijd.

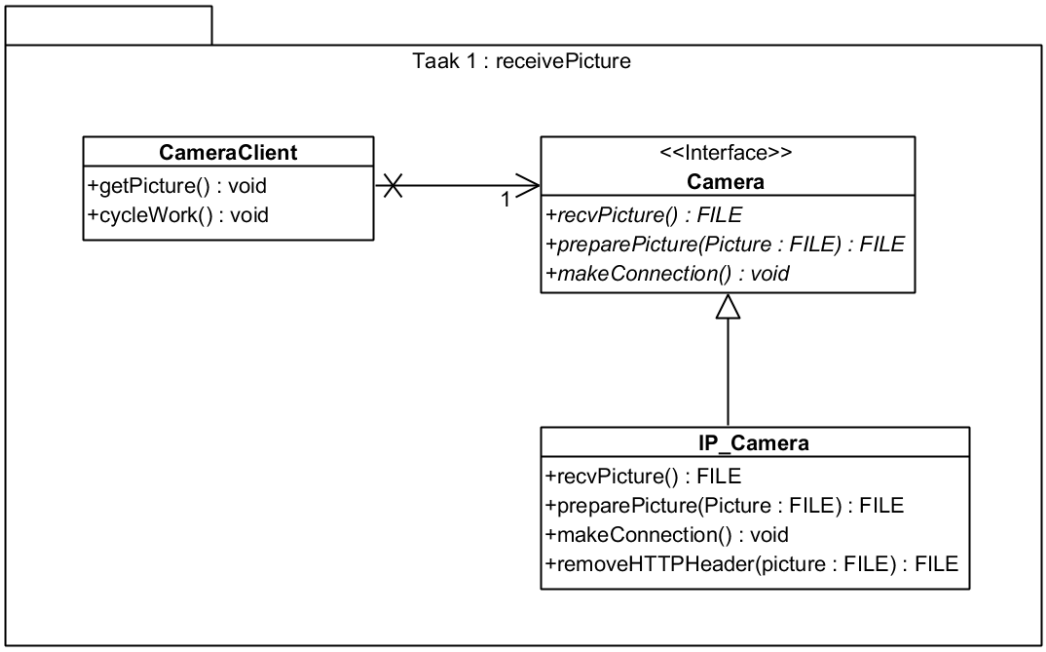
Voor elke use-case van het use-casediagram is een beschrijving opgesteld in Tabel 12 is een voorbeeld gegeven van een beschrijving. De rest van de beschrijvingen zijn in bijlage C gegeven.

ID:	UC1
Use case:	Get Picture
Brief description:	Opvragen van een afbeelding aan het begin van de cyclustijd.
Primary actors:	Cycle Time
Secondary actors:	Geen
Preconditions:	1. Een nieuwe PLC-cyclus is begonnen
Main flow:	1. De PLC maakt verbinding met de camera. 2. De camera stuurt connectie bevestiging. 3. De PLC vraagt een afbeelding op bij de camera. 4. De camera stuurt de afbeelding. 5. Include (Make picture editable). 6. PLC slaat afbeelding op.
Postconditions:	1. De afbeelding is opgeslagen
Alternative flows:	None

TABEL 12 - BESCHRIJVING 'GET PICTURE'

5.1.2 Klassendiagram

Het klassendiagram voor het ophalen van een afbeelding bestaat uit twee onderdelen. Het analyse ontwerp en het uiteindelijke ontwerp. Het analyse ontwerp bestaat uit klassen waarbij er alleen invulling is gegeven aan de methodes van de klassen en is er aangegeven wat de relatie is tussen de klassen. In het uiteindelijke klassendiagram zijn ook alle attributen gemodelleerd en zijn de methodes zo nodig aangepast. In Figuur 11 is het analyse klassendiagram opgesteld. Deze is niet te vinden in het ontwerprapport. Hierin staat alleen het uiteindelijke ontwerp.



FIGUUR 11 - ANALYSE KLASSENDIAGRAM INCREMENT 1

In het analyse klassendiagram is te zien dat er twee onderdelen ontworpen zijn. De camera en de CameraClient. De methodes zijn opgesteld aan de hand van de use-casebeschrijvingen.

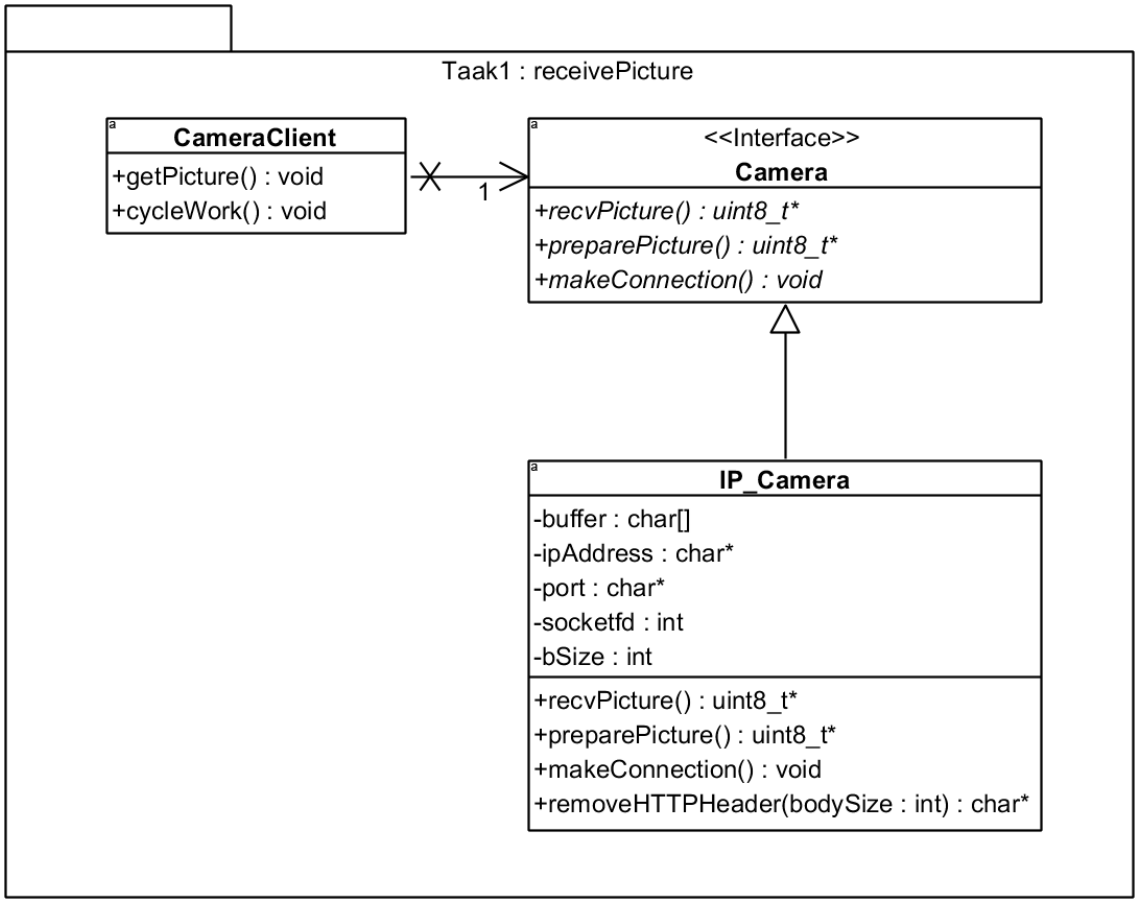
CameraClient

CameraClient is een klasse die nodig is door de werking van een Bachmann PLC. Bij het programmeren van C++ op de PLC moet een klasse worden aangemaakt waarbij een methode cyclisch uitgevoerd wordt. In het geval van dit ontwerp is dit de klassen CameraClient. Binnen deze klasse wordt de methode '*cycleWork()*' cyclisch aangeroepen.

Camera en IP_Camera

IP_Camera is een overerving van de klasse Camera. Camera is een abstracte klasse. Er is voor deze constructie gekozen zodat er makkelijk een ander soort camera op aangesloten kan worden zonder dat de rest van de code veranderd. Het kan namelijk mogelijk zijn dat USB-camera's in de toekomst wel ondersteund worden door Bachmann PLC's.

Naast het analyse klassendiagram is ook het uiteindelijke klassendiagram opgesteld. Hierin zijn de attributen van de klassen ook ingevuld en zijn er wat methodes veranderd. Dit klassendiagram is te vinden in Figuur 12.



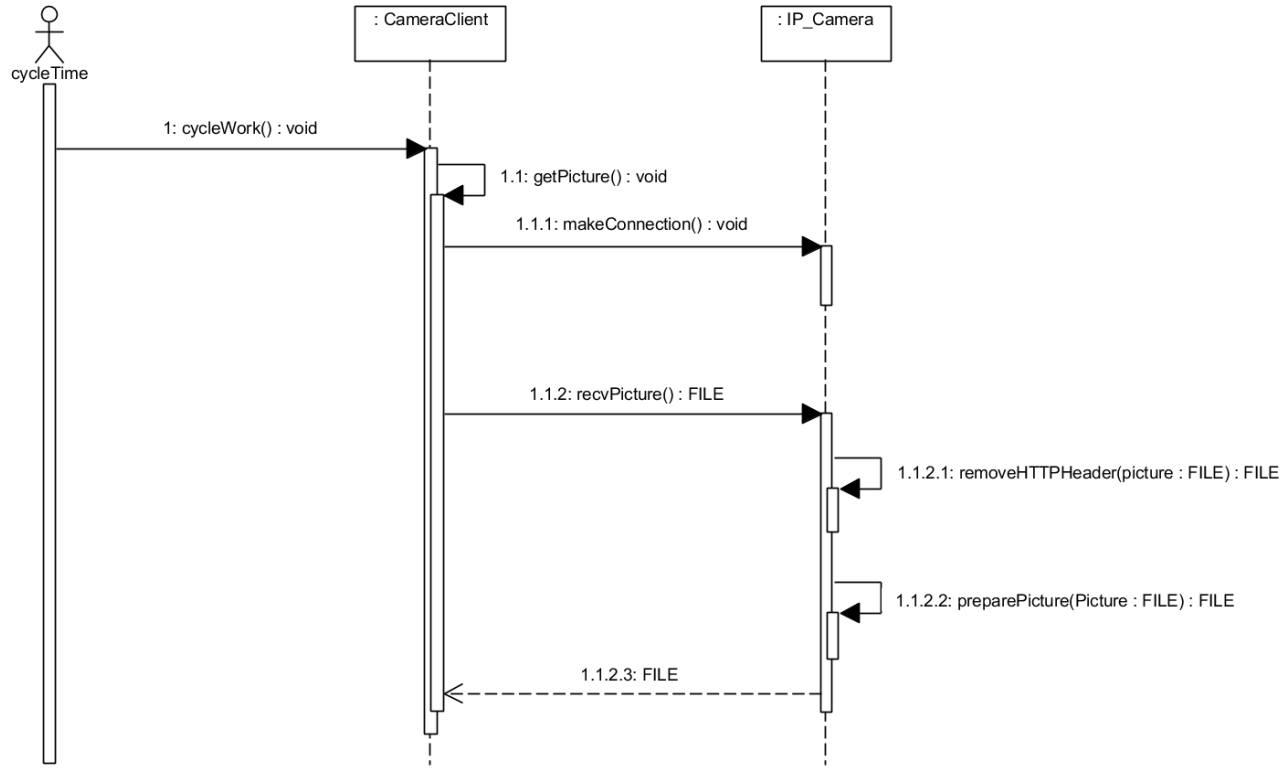
FIGUUR 12 - UITEINDELIJKE KLASSENDIAGRAM INCREMENT 1

De veranderingen die zijn gemaakt volgen uit de werking van het omgaan met afbeeldingen binnen C++. Dit kwam naar voren bij het realiseren van het ontwerp. Deze veranderingen zijn al meegenomen in het ontwerp. Dit veranderingen worden in het hoofdstuk 5.2 realisatie verder besproken.

Zoals in figuur 10 te zien is zijn de attributen van IP_Camera ingevuld. Deze attributen zijn nodig om een connectie te maken met de camera en bij het opvragen van een afbeelding.

5.1.3 Sequentiediagram

Na het maken van een klassendiagram is er een sequentiediagram opgesteld. Hierin is de volgorde aangegeven waarin de methodes uit het klassendiagram moeten worden uitgevoerd. Deze volgorde is gebaseerd op de use-casebeschrijvingen. Het sequentiediagram is te vinden in Figuur 13.



FIGUUR 13 - SEQUENTIEDIAGRAM INCREMENT 1

Zoals te zien is zijn alle methodes die opgesteld zijn in het klassendiagram gebruikt in het sequentiediagram. Daarnaast is het sequentiediagram opgesteld na het maken van het analyse klassendiagram. Dit komt, omdat deze is opgesteld voordat er attributen waren toegevoegd aan het klassendiagram. Attributen worden namelijk niet meegenomen in het sequentiediagram. Doordat deze is gemaakt voor het invullen van de attributen, zijn de veranderingen van het klassendiagram niet meegenomen in het sequentiediagram.

5.1.4 Keuzes

Tijdens het eerste increment moesten er keuzes gemaakt worden bij het maken van het ontwerp. De belangrijkste keuzes zijn het gebruik maken van threads (taken) en het gebruik van de abstracte klasse Camera.

Threads

Bij het ontwerpen van het systeem moest er gekeken worden naar het gebruik van multi-threading bij het realiseren van het vision-algoritme. Bij multi-threading worden twee stukken programma door elkaar heen uitgevoerd. Hierbij wordt er steeds afgewisseld tussen de twee programma's. Dit is gewenst als twee processen tegelijkertijd moeten worden uitgevoerd en niet afhankelijk zijn van elkaar.

Er is met de opdrachtgever en expertcollega's besproken of dit ook toegepast zou kunnen worden bij de uitvoering van de afstudeeropdracht. Uit deze gesprekken kwam naar voren dat het merendeel huidig nut niet inzag om multi-threading toe te passen bij de beeldbewerking. De rede hiervoor was dat de beeldbewerking en het ophalen van de foto's te afhankelijk van elkaar zijn. Het kan voor komen dat sommige threads te veel tijd krijgen en andere te weinig, waardoor de snelheid van het algoritme inconsistent kan worden. Echter kan het wel nuttig zijn als er mogelijk een robotarm toegevoegd gaat worden in de toekomst. Dan kan de robotarmbesturing losgetrokken worden van de bewerking. Hierdoor wordt er bij de ontwerpen rekening gehouden met de mogelijkheid dat in de toekomst multi-threading kan worden gebruikt.

Camera <<interface>>

Bij het ontwerpen van het klassendiagram moest er een keuze gemaakt worden of er een abstracte klasse zou worden gebruikt. Door een abstracte klasse te gebruiken kunnen er meerdere verschillende camera's gebuikt worden zonder de klasse CameraClient aan te passen. De benodigde methodes worden gedefinieerd in de abstracte klasse en deze worden geïmplementeerd in de afgeleide klassen. Hierdoor kan dezelfde methode hergebruikt worden met een andere implementatie. Dit maakt de software uitbreidbaar en onderhoudbaar.

Als er geen abstracte klasse wordt gebruikt. Zou er een directe link zijn tussen de IP_Camera klasse en CameraClient. De implementatie hiervan zou sneller zijn en de werking hiervan blijft ook hetzelfde. Echter zou CameraClient aangepast moeten worden, als er een nieuw soort camera moet worden geïmplementeerd. Hierdoor wordt deze constructie vaak gebruikt voor kleine systemen waarbij alles binnen het systeem vast staat.

Met behulp van een tabel waarin alle eigenschappen van een abstracte klasse en directe connectie een beoordeling hebben gekregen, is er een keuze gemaakt. De tabel is opgesteld in Tabel 13

	Onderhoudbaar	**Uitbreidbaar**	**Complexiteit ontwerp**	**Snelheid van implementatie**	**Totaal**
Abstracte klasse	**++**	**+**	**-**	**-**	**+**
Directe connectie	**-**	**--**	**+**	**+**	**-**

TABEL 13 - KEUZE EIGENSCHAPPEN ABSTRACTE KLASSE

Zoals eerder in figuur 10 te zien is, is er gekozen om een abstracte klasse te gebruiken. Huidig wordt er geen USB-camera ondersteund, maar dat kan in de toekomst veranderen. Aangezien een USB-camera een snellere dataoverdracht heeft, bestaat de kans dat deze in de toekomst gebruikt gaat worden. Door deze constructie kan het systeem beter onderhouden en uitgebreid worden. Zo nodig zouden er ook meerdere camera's tegelijkertijd gebruikt kunnen worden.

Programmeertaal

Zoals eerder is besproken kan er op de Bachmann PLC geprogrammeerd worden in verschillende programmeertalen. Deze worden hieronder herhaald:

- SCL
- LAD
- FBD
- C/C++

De eerste drie talen worden vooral gebruikt voor het aansturen van digitale en analoge output op basis van digitale en analoge input. Hierbij wordt er veel geprogrammeerd met logische operaties. Deze operaties zijn voorgeprogrammeerd en worden in sequentie gebruikt om een functionaliteit te bereiken. Daarnaast worden deze operaties vaak gerepresenteerd en gevisualiseerd met blokken. Deze worden met lijnen aan elkaar verbonden. Deze talen zijn echter beperkt bij geavanceerde algoritmes als computer-vision. Zoals bij deze opdracht het geval is.

C en C++ zijn programmeertalen waarbij zelf operaties geprogrammeerd moeten worden voordat deze gebruikt kunnen worden. De programmeertaal is gebaseerd op tekst in plaats van visuele blokken. Doordat er veel zelf geprogrammeerd moet worden is het mogelijk om complexe algoritmes te maken. Hierdoor is het ideaal voor het maken van een vision-algoritme. Aangezien er bij deze opdracht afbeeldingen bewerkt moeten worden, is ervoor gekozen om C++ te gebruiken voor het uitvoeren van de opdracht. Deze keuze is ook uitgewerkt in Tabel 14. Hierin is aan elke eigenschap van een programmeertaal een beoordeling gegeven. Aan de hand van deze beoordeling is er een keuze gemaakt.

	I/O bewerkingen	Voorgeprogrammeerd	Te programmeren complexiteit	Totaal
SFC/FBD/LD	**++**	**++**	**--**	**++**
C/C++	**+**	**+**	**++**	**++++**

TABEL 14 - KEUZE PROGRAMMEERTAAL

Er wordt niet direct op de PLC geprogrammeerd, maar op een computer die aangesloten is op de PLC via een ethernet kabel. In solution center wordt de code gecompileerd en naar de PLC gestuurd. Om code te compileren voor een extern systeem is een toolchain nodig. Om het programma te compileren voor de PLC wordt een GCC gebaseerde toolchain gebruikt. In deze toolchain staan de libraries die de PLC kent en gebruikt kunnen worden voor het programmeren. C++ bestaat uit verschillende versies. Huidig wordt C++14 (uit 2014) veel gebruikt. Echter is de toolchain voor de Bachmann PLC gebaseerd op C++98 (uit 1998). Dit is een oude versie van C++. Hierdoor zijn niet alle functionaliteiten van C++ ondersteund. Hiermee moet rekening gehouden worden bij het realiseren van de opdracht.

5.2 Realisatie ophalen afbeelding

In dit hoofdstuk wordt de realisatiefase van increment 1 behandeld.

Tijdens het realiseren van het ontwerp moesten er twee functies geïmplementeerd worden. De connectie met de IP-Camera en het omzetten van een jpg-afbeelding naar een rgb-afbeelding. Om een verbinding te maken met een IP-camera en een HTTP-request hierover te sturen, moest er gekeken worden naar de stappen die uitgevoerd moeten worden om deze connectie tot stand te brengen. De stappen voor het convergeren van een jpg afbeelding zijn besproken in bijlage B.

5.2.1 Uitvoeren computer

Bij het opstellen van de eisen is ervoor gekozen om de uitvoering van het vision-algoritme op de Bachmann PLC als een aparte eis op te stellen. Er is namelijk een kans dat het niet mogelijk is om het algoritme uit te voeren op de PLC. Door het eerst op de computer uit te voeren kan in ieder geval de correcte werking aangetoond worden. In het derde increment wordt het algoritme geïmplementeerd op de PLC. Echter is er bij het ontwerpen van de software ervan uit gegaan dat het wordt gerealiseerd op de PLC. Hierdoor hoeven er later geen aanpassingen aan het ontwerp te worden gemaakt.

5.2.2 Connectie camera

Voor het opvragen van een afbeelding bij de camera, moest er een verbinding gemaakt worden waarover HTTP-requests worden verzonden. Hiervoor wordt het TCP-protocol gebruikt. Dit is een protocol dat data verzendt over een netwerk. Hierbij wordt er gegarandeerd dat de data goed verstuurd en ontvangen wordt. Voor het maken van een TCP-verbinding zijn de volgende benodigdheden:

- Een IP-adres van de camera en computer
- Een poortnummer van de camera en computer
- Een uniek socket-ID.

Voor het maken van een connectie zijn standaard libraries beschikbaar. Deze worden standaard gebruikt voor het maken van TCP-connecties. De benaming voor de libraries verschilt per besturingssysteem, maar de werking is hetzelfde.

5.2.3 Afbeelding opvragen

Voor het opvragen van een afbeelding moest een HTTP-request opgesteld worden en naar de camera gestuurd worden. Het model dat hiervoor gebruikt wordt is Foscam FI9853EP. Deze camera kan volledig bestuurd worden door middel van het HTTP-protocol. Er kunnen HTTP-get en -put berichten gestuurd om dit te eigenschappen te veranderen of afbeeldingen op te vragen. [12] Tijdens de implementatie is alleen het commando om een afbeelding op te vragen gebruikt. Het commando dat is gebruikt wordt hieronder gegeven.

"**GET /cgi-bin/CGIProxy.fcgi?cmd=snapPicture2&usr=*******&pwd=******* HTTP/1.1\r\nhost: 192.168.134.97:88\r\nConnection: keep-alive\r\n\r\n**"

In de code is de gebruikersnaam en wachtwoord anders. Bij dit commando geeft de camera als reactie een jpg-afbeelding terug met een resolutie van 640*480. Welke omgezet moet worden naar een bewerkbare afbeelding. Een niet gecomprimeerde afbeelding van die grote is 921,6 KB groot (bijlage B). Echter is de afbeelding die door de camera wordt geleverd 10 tot 20 KB. Dit laat zien hoeveel een jpg een afbeelding kan comprimeren.

5.2.4 JPG naar RGB

Om de afbeelding die is verkregen van de camera te kunnen bewerken moet deze gedecodeerd worden van jpg naar RGB. De stappen die hiervoor uitgevoerd moeten worden zijn in hoofdstuk 4.3.2 besproken. Echter kost het veel tijd om elke stap te implementeren. Daarvoor is er gekozen om een library te gebruiken waarbij de stappen al zijn geïmplementeerd. De library die wordt gebruikt heet STD image [13]. Dit is een public domain library gemaakt in een enkele headerfile om afbeeldingen te coderen en decoderen. Doordat het een enkele headerfile is, kan deze makkelijk gebruikt worden in de software aangezien deze alleen geïnclude hoeft te worden. Ook is het platform onafhankelijk zolang de code gecompileerd kan worden.

Uit deze library is de methode '*stbi_load_from_file'* gebruikt. Deze opent een file, decodeert de afbeelding en geeft een pointer naar een uint8_t array terug met de RGB-waardes van de afbeelding. Elke drie bytes representeert een pixel. Deze array wordt in een queue gezet, zodat de afbeelding gebruikt kan worden door een andere thread. Een queue is een stuk geheugen dat gedeeld wordt tussen threads.

5.2.5 Resultaat

Het resultaat van dit increment is een opgeslagen jpg-afbeelding en een array met RGB-waardes. Deze resultaten zijn in het testrapport (bijlage E) getoond. De code van het product is te vinden bijlage D.

In dit increment is ook de knikkerbaan gemaakt op basis van de systeemeisen en het probleemdomein. De resultaten van de knikkerbaan zijn in Figuur 14 te vinden. Hiermee is voldaan een de eisen SE04 en SE05.

FIGUUR 14 - KNIKKERBAAN

5.3 Testen ophalen afbeelding

In dit hoofdstuk wordt de testfase van eerste increment behandeld.

Voor het testen van de eisen zijn testen opgesteld tijdens het ontwerpen. Deze zijn uitgevoerd na het implementeren van de software. De testen zijn uitgevoerd om te controleren of er aan de eisen voldaan is. De testen van dit hoofdstuk zijn gebaseerd op de use-cases en eisen van increment 1. Per use-case is er een test opgesteld. Hierbij zijn de handelingen aangegeven die de tester moet uitvoeren om een eis te testen. Hierbij is ook het verwachte resultaat opgesteld en of dit resultaat is behaald. De testen zijn in Tabel 15 en Tabel 16 gegeven. De testen zijn ook te vinden in het testrapport (bijlage E).

Test	1. FE05
Use-case	UC01: Get picture
Omschrijving	Opvragen van een afbeelding aan het begin van de cyclustijd.
Pre-conditie	Er staat geen afbeelding in de map waar marbleVoorspeller.exe is opgeslagen
Testproces	1. Ga naar map waar marbelVoorspeller.exe is opgeslagen 2. Open marbleVoorspeller.exe
Verwachte uitkomst	In de map van marbleVoorspeller.exe is een picture.jpg gemaakt met het huidige beeld van de camera.
Uitkomst	Uitkomst is zoals verwacht.

TABEL 15 - TEST UC01

Test	2. FE06
Use-case	UC02: Make picture editable
Omschrijving	Het decoderen van een afbeelding, zodat elke pixel uit een RGB-waarde bestaat.
Pre-conditie	Er is een afbeelding opgevraagd bij de camera en in de map van marbleVoorspeller.exe staat deze afbeelding als picture.jpg
Testproces	1. Ga naar map waar marbelVoorspeller.exe is opgeslagen 2. Open marbleVoorspeller.exe
Verwachte uitkomst	Er verschijnt een scherm met de RGB-waardes van elke pixel
Uitkomst	Uitkomst is zoals verwacht.

TABEL 16 - TEST UC02

De testen zijn besproken met de opdrachtgever om de voortgang te bespreken. Na dit increment was het deelproduct naar wens. Door het uitvoeren van de eis FE05 is er ook voldaan aan de systeemeisen SE01, SE02 en SE03.

6. Increment 2: positie bepalen knikker

Increment 2 was gefocust op het bewerken van de foto die bij increment 1 was opgehaald. Met als doel het bepalen van de positie van de knikker. Hiervoor is net als bij het eerste increment ontworpen, gerealiseerd en getest. De eisen die binnen dit increment zijn uitgevoerd zijn opgesteld in Tabel 17.

Increment	Eisen
2: Positie bepalen knikker	• FE01 • FE02 • FE07

TABEL 17 - EISEN POSITIE BEPALEN KNIKKER

6.1 Ontwerp: positie bepalen knikker

In dit hoofdstuk wordt de ontwerpfase van het tweede increment beschreven.

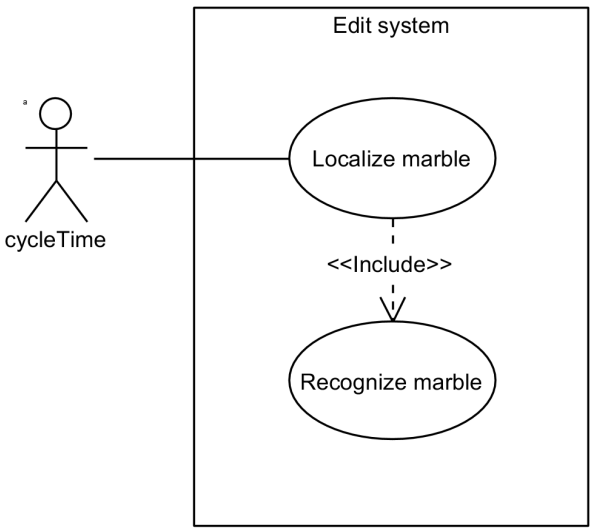
Zoals in het eerste increment is er voor dit increment ook een use-casediagram, klassendiagram en sequentie diagram omgesteld. Deze staan grotendeels los van de diagrammen van increment 1. Dit komt doordat de code van dit increment wordt uitgevoerd in een nieuwe thread. Aan het einde van dit hoofdstuk worden de keuzes besproken die zijn gemaakt voor het ontwerpen van increment 2.

6.1.1 Use-casediagram

Tijdens het ontwerpen van increment 2 zijn de eisen FE01 en FE02 omgezet tot use-cases. Eis FE07 is niet omgezet tot use-case, omdat deze alleen gebruikt wordt voor het controleren van de eisen. De use-case beschrijven, dus de volgende functies:

- Het lokaliseren van de knikker op de afbeelding
- Het herkennen van de knikker op de afbeelding

In Figuur 15 is het use-casediagram opgesteld.



FIGUUR 15 - USE-CASEDIAGRAM POSITIE BEPALEN KNIKKER

Om de locatie van de knikker te kunnen bepalen moet eerst de knikker onderscheiden worden van de rest van de afbeelding. Daardoor wordt ‘Recognize marble’ geïncludeerd in ‘Localize marble’.

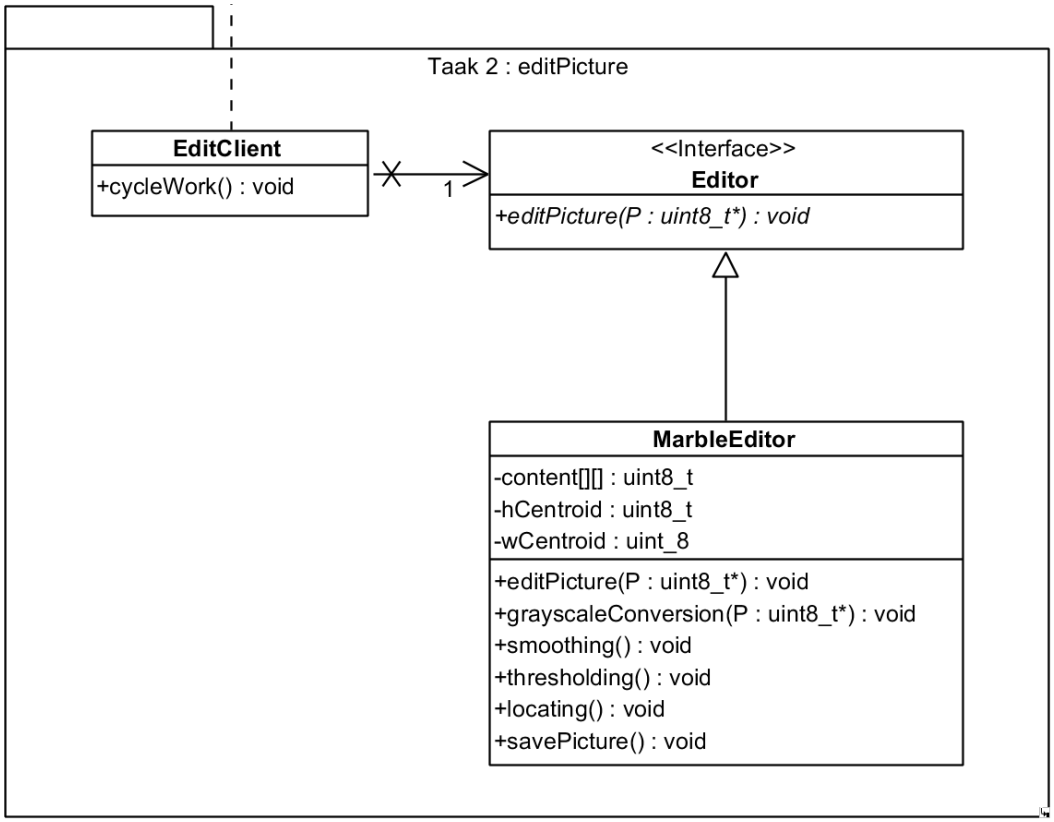
In Tabel 18 is de beschrijvingen van use-case ‘Localize marble’ uitgewerkt. De tweede use-case is uitgewerkt in het bijlage C.

ID:	UC3
Use case:	Localize marble
Brief description:	Lokaliseren van knikker op een afbeelding
Primary actors:	Cycle Time
Secondary actors:	Geen
Preconditions:	1. Een nieuwe PLC-cyclus is begonnen 2. Er is een bewerkbare afbeelding beschikbaar
Main flow:	1. De PLC vraagt een bewerkbare afbeelding op. 2. De PLC ontvangt de bewerkbare afbeelding. 3. Include(Recognize marble) 4. De PLC ontvangt een bewerkte afbeelding met knikker 5. De PLC zoekt het middelpunt van de knikker 6. De PLC ontvangt het middelpunt 7. De PLC zet een marker op het middelpunt 8. De PLC slaat afbeelding op.
Postconditions:	1. De afbeelding met een marker op het middelpunt van de knikker is opgeslagen
Alternative flows:	None

TABEL 18 - BESCHRIJVING ‘LOCALIZE MARBLE’

6.1.2 Klassendiagram: positie bepalen knikker

Tijdens dit increment is er meteen een klassendiagram gemaakt met de attributen al ingevuld. Er is hiervoor gekozen omdat de attributen volgen uit de methodes voor beeldbewerking. Deze worden verder in dit deelhoofdstuk behandeld. In Figuur 16 is het klassendiagram gegeven.



FIGUUR 16 - KLASSENDIAGRAM POSISTIE BEPALEN KNIKKER

EditClient

EditClient is de klasse met de functie '*cycleWork()*'. Deze wordt net als bij CameraClient cyclisch uitgevoerd. EditClient heeft zijn eigen taak (thread) waarin de code wordt uitgevoerd. EditClient is echter wel verbonden met CameraClient om de afbeelding te krijgen die is binnengehaald bij de camera. Als er afbeeldingen zijn opgehaald wordt de afbeelding verkregen en bewerkt en als er geen afbeelding beschikbaar is wordt er verder geen code uitgevoerd in de cyclus. Binnen '*cycleWork()*' wordt '*editPicture()*' aangeroepen van de <<interface>> Editor. Dit is een brug naar de implementatie van '*editPicture()*' van de klasse MarbleEditor.

MarbleEditor

MarbleEditor is de overgeërfde klassen van de <<interface>> Editor. Hierin wordt de afbeelding bewerkt, zodat de knikker onderscheiden kan worden van de rest en de positie kan worden bepaald. Om de afbeeldingen te onderscheiden van de andere elementen in de afbeelding is ervoor gekozen om de volgende bewerkingsmethodes te gebruikten:

- grayscaleConversion()
- smoothing()
- thresholding()

De werking van deze methodes zijn behandeld in bijlage B. De afbeelding komt binnen van CameraClient als een 1-dimensionale array. Binnen '*grayscaleConversion()*' wordt deze omgezet naar een zwartwit-afbeelding in 2-dimentionale array en opgeslagen in het attribuut '*content*'. Dit is nodig doordat de

filtermethodes kijken naar de omliggende pixels. Dit is makkelijker te realiseren als er gewerkt wordt in een 2-dimentionale array waarbij de positie van elke pixel in de afbeelding overeenkomt met de positie in de array.

'*Smooting()*' wordt gebruikt om de details uit de afbeelding te halen. Hierdoor worden piekwaardes verwijderd uit de afbeelding die bij latere bewerkingen voor moeilijkheden kunnen zorgen.

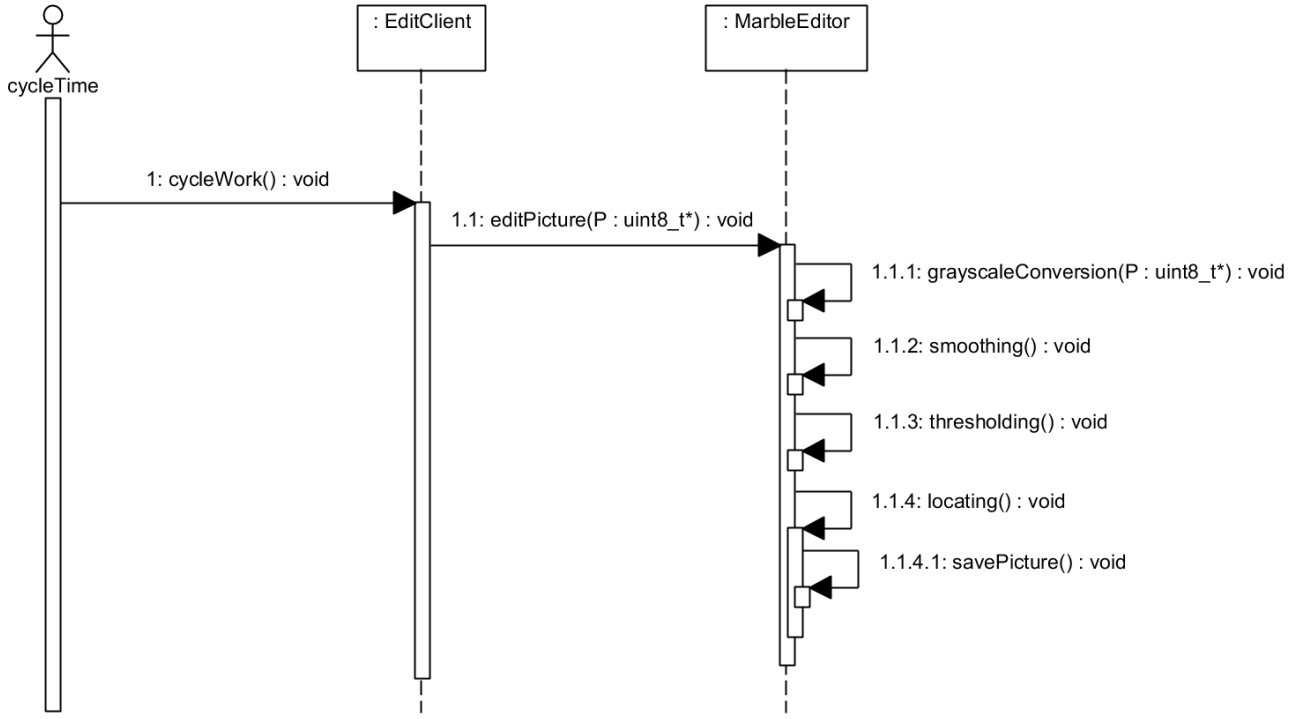
'*Thresholding()*' wordt gebruikt om de knikker te onderscheiden van de rest van de afbeelding. Aangezien de knikkerbaan zwart is en de knikker een hoog contrast heeft met de knikker, kan deze methode gebruikt worden. Bij goede thresholding wordt de knikker wit in de afbeelding en de rest zwart.

Met de bovenstaande methodes kan de knikker onderscheiden worden van de rest. Naast deze bewerkingen is er ook de methode '*locating()*' opgesteld die de positie van de knikker kan bepalen. De positie wordt opgeslagen in de attributen '*hCentroid*' en '*wCentroid*'. Deze worden in het volgende increment ook gebruikt voor het voorspellen van de knikkerpositie.

De laatste methode die is toegevoegd in het ontwerp is '*savePicture()*'. Deze methode wordt gebruikt om de beeldbewerkingsmethodes te controleren door de data in '*content[][]*' op te slaan in een afbeelding.

6.1.3 Sequentiediagram: positie bepalen knikker

Na het maken van een klassendiagram is er een sequentiediagram opgesteld. Hierin is de volgorde aangegeven waarin de methodes uit het klassendiagram moeten worden uitgevoerd. Deze volgorde is gebaseerd op de use-casebeschrijvingen. Het sequentiediagram is te vinden in Figuur 17.



FIGUUR 17 - SEQUENTIEDIAGRAM POSITIE BEPALEN KNIKKER

Bij het maken van dit sequentie diagram zijn geen keuzes gemaakt, omdat de indeling van de methodes volledig volgt uit het klassendiagram en de use-casebeschrijvingen.

6.1.4 Keuzes: positie bepalen knikker

Tijdens het ontwerpen in increment 2 zijn keuzes gemaakt die bijdragen aan het opgestelde ontwerp. De belangrijkste keuzes die gemaakt zijn, is het bepalen welke beeldbewerkingsmethodes gingen worden gebruikt en de of de abstracte klasse Editor nodig was. Deze keuzes worden in dit deelhoofdstuk besproken.

Beeldbewerkingsmethodes

Voor het kiezen van beeldbewerkingsmethodes die nodig waren om de positie te bepalen van de knikker, is er eerst gekeken naar de beeldbewerkingsmethodes die zijn behandeld in Bijlage B. Zelfs met de analyse van deze methodes was het lastig om een keuze te maken. Het is namelijk moeilijk in te schatten wat het resultaat van de methodes zijn zonder ervaring te hebben met de camera die gebruikt wordt. Echter zijn er wel methodes die vrijwel altijd gebruikt worden.

GrayscaleConversion

Methodes als '*grayscaleConversion()*' en een vorm van '*smooting()*' worden bijna altijd gebruikt in het begin van de beeldbewerkingen. Deze methodes zijn om de afbeelding voor te bereiden om ingewikkeldere methodes uit te voeren. '*grayscaleConversion()*' wordt alleen gebruikt als de originele afbeelding een kleurenafbeelding is. Echter als dit het geval is, is deze methode van groot belang. Het is namelijk makkelijker en sneller om een zwartwitafbeelding te bewerken. Een pixel bestaat dan namelijk uit één byte in plaats van drie bytes.

Smoothing

'*smooting()*' is belangrijk voor het verwijderen van detail en ruis in een afbeelding. De oppervlakte van objecten in een afbeelding zijn vaak niet effe en hebben veel imperfecties. Dit zorgt ervoor dat vervolg bewerkingen, bijvoorbeeld edge-detectie niet goed werken. Aangezien er naar verschil in waarde wordt gekeken. Om deze rede wordt een vorm '*smooting()*' bijna altijd gebruikt.

Thresholding vs K-means

De echte keuze voor beeldbewerking is de methode die gebruikt moet na '*smooting()*'. Hierbij moet de knikker onderscheiden worden van de rest. Een makkelijke manier die gebruikt kan wordt als er veel contrast is, is thresholding. Hierbij wordt er een grenswaarde bepaald en worden alle pixelwaarde beneden deze waarde naar 0 gezet en de pixelwaardes boven deze waarde naar 255. Hierdoor wordt ideaal gezien alleen de knikker wit en rest zwart. Deze methode is makkelijk te implementeren en is snel, maar is handig als er veel in de afbeelding veranderd. Als eigenschappen van bijvoorbeeld van de knikkerbaan veranderen, dan kan dat ervoor zorgen dat de ingestelde grenswaarde niet meer klopt.

K-means is een methode dat dynamisch pixels indeelt in groepen. Hierbij worden de pixel met waardes die dicht bij elkaar zitten, ingedeeld in dezelfde groep. Na het indelen van de pixels krijgt elke pixel binnen een groep dezelfde pixelwaarde. Er kan zelf gekozen worden hoeveel groepen er gemaakt moeten worden. Het voordeel van deze methode is dat de methode dynamisch is, dus als er veranderingen zijn in de opstelling de methode niet aangepast hoeft te worden. Echter is er weinig invloed op het resultaat van de methode. De methode kan pixels ongewenst indelen in een groep waardoor het resultaat niet klopt. Om dit op te lossen moet er veel getest worden. Dit kost tijd. Ook is de implementatie van de methode ingewikkelder en kost het meer rekenkracht.

Om een keuze te maken tussen thresholding en K-means is er een keuzetabel opgesteld die eerder ook gebruikt is of andere keuzes te maken. De keuzetabel is te vinden in Tabel 19.

	Gemak implementatie	**Dynamisch**	**Voorspelbaarheid uitkomst**	**Totaal**
Thresholding	**+**	**-**	**+**	**+**
K-means	**-**	**++**	**-**	**0**

TABEL 19 - EIGENSCHAPPEN THRESHOLDING EN K-MEANS

Er is gekozen voor thresholding. Deze methode is niet dynamisch, maar is makkelijk te implementeren en de uitkomst is heel voorspelbaar. Daarnaast is het niet veel werk om een de grenswaarde te veranderen. K-means kost te veel tijd om te implementeren en geeft geen zekerheid dat het werkt.

De keuzes van de beeldbewerkingsmethodes kunnen in latere incrementen veranderen, doordat er dat ervaring in de praktijk wordt opgedaan met de camera. Bij het realiseren van het ontwerp wordt er ook naar het resultaat van de bewerkingen gekeken. Hieruit blijkt ook of de gemaakte keuze correct was.

Editor <<interface>>

Ook bij het ontwerp van het tweede increment moest er een keuze gemaakt worden of er een abstracte klasse zou worden gebruikt. De keuze hiervoor had grotendeels dezelfde redenatie als bij de keuze van voor de klasse Camera <<interface>>. Er kan in de toekomst gekozen worden om andere bewerkingen uit te voeren. Als dit het geval is kan er door de abstracte klasse Editor makkelijk een andere bewerkingsklasse toegevoegd worden. Er is hier weer gekozen voor uitbreidbaarheid en onderhuidbaarheid.

6.2 Realisatie: positie bepalen knikker

In dit hoofdstuk wordt de realisatiefase van increment 2 behandeld.

Tijdens de realisatie van increment 2, moest er goed gelet worden op de uitkomsten van de beeldbewerkingsmethodes. Bij het ontwerp was er een inschatting gemaakt of de gekozen methodes genoeg waren voor het herkennen van de knikker, maar dit moest nog blijken in het resultaat. Hiervoor moest er een afbeelding opgeslagen worden na elke beeldbewerkingsmethode. Zo kon er stap voor stap gekeken worden of het resultaat gewenst is. Na het afstellen van de beeldbewerkingsmethodes moest ook de positie van de knikker bepaald worden.

6.2.1 Implementatie beeldbewerkingsmethodes

Tijdens de realisatie van increment 2 moesten er drie beeldbewerkingsmethodes geïmplementeerd worden:

- Grayscale-conversie
- Smoothing
- Thresholding

Daarnaast moest het lokaliseren van de knikker geïmplementeerd worden.

Grayscale-conversie

Voor de implementatie van grayscale-conversie is de uitgevoerde analyse gebruikt. Hierbij was er gekeken naar de manier waarop een zwartwitpixel wordt samengesteld uit een RGB-pixel. De conversie bestaat uit drie stappen. Waarbij de eerste en de laatste stap het weghalen en toevoegen van gammacorrectie is. Tijdens de implementatie is er echter gemerkt dat dit weinig invloed heeft op de te bewerken afbeelding. Dit is getest door de gemiddelde waarde te bereken van de som van alle pixels. Met en zonder gammacorrectie was het gemiddelde hetzelfde. Doordat er vrijwel geen verschil was is er gekozen om geen gammacorrectie te gebruiken.

Voor het berekenen van de grijswaardes van de pixels is de volgende formule [14] gebruikt:

$$Y = 0.2126R + 0.7152G + 0.0722B$$

Deze is ook gegeven in bijlage B. Bij het uitvoeren van deze formule wordt de array met pixels 3x zo klein. Hierbij is het resultaat opgeslagen in een 2-dimensionale array. De grootte van de array staat gelijk aan de hoogte en breedte van de afbeeldingen. In dit geval 480x640. Hierdoor kunnen filtermethodes makkelijker geïmplementeerd worden.

Smoothing

Bij de implementatie van smoothing was het belangrijk dat ruis zo veel mogelijk werd verwijderd. Het komt namelijk voor dat er licht reflecteert op de knikkerbaan. Hierdoor ontstaan er licht vlekken op de afbeelding. Bij het implementeren is er een kernel van 3x3 gebruikt om het gemiddelde te berekenen van de middelste en omliggende pixels. Hierbij hebben de omliggende pixel een hogere weging gekregen dan de middelste knikker. Hierdoor wordt de afbeelding waziger. Daarnaast kan smoothing vaker uitgevoerd worden wanneer nodig. De grootte en invulling van de kernel is bepaald door het telkens uitproberen van het vision-algoritme. Door het resultaat van thresholding te bekijken kan ook gekeken worden of

smoothing goed is gelukt. Als er namelijk meer is gedetecteerd dan alleen de knikker, moet smoothing aangepast worden.

Thresholding

Bij de implementatie van thresholding is er vooral gekeken naar de grenswaarde waarop de bewerking moet controleren. Als de grenswaarde te laag is kan er teveel herkend worden. Echter als de grenswaarde te hoog is wordt de knikker niet goed herkend. Bij thresholding wordt namelijk de waarde van de pixel op 0 of 255 gezet op basis van deze grenswaarde. Voor thresholding is er eerst gekeken naar de pixelwaardes van de knikker. Dit is handmatig uitgevoerd met behulp van de website [15] RawPixel waarbij je per pixel de waarde van een afbeelding kan bekijken. Ook kan er op deze website een dump van pixels omgezet woorden tot een zichtbare afbeelding. Hierdoor hoever er geen extensie zoals .jpg toegevoegd te worden aan de afbeelding. Door het resultaat van smoothing te analyseren op RawPixel is er een grenswaarde van 200 gekozen.

Lokaliseren knikker

Nadat de knikker geïsoleerd was van de rest van de afbeelding, moest de positie van de knikker bepaald worden. Hiervoor moest het middelpunt van de knikker worden berekend. Dit middelpunt wordt gerepresenteerd met een (w,h) coördinaat. Waarbij w de breedte is en h de hoogte. Om het middelpunt te berekenen is de onderstaande formule [16] gebruikt.

$$(w_c, h_c) = \left(\frac{1}{n}\sum_{i=1}^{n} w_i, \frac{1}{n}\sum_{i=1}^{n} h_i\right)$$

Deze formule representeert de som van de hoogte van alle witte pixels en de som van de breedte van alle witte pixels. Deze sommaties worden gedeeld door het aantal witte pixels. Hiermee wordt het gemiddelde van hoogte en breedte berekend, ofwel het middelpunt van alle witte pixels. Deze methode werkt alleen als de knikker goed geïsoleerd is. Aangezien afwijkende waardes ervoor zorgen dat het berekende middelpunt afwijkt van het daadwerkelijke middelpunt. Hierdoor is het belangrijk dat smoothing en thresholding goed worden afgesteld.

Om het resultaat te kunnen verifiëren is er punt in de bewerkte afbeelding gezet, waarvan de plek overeenkomt met het berekende middelpunt. Dit punt heeft een waarde van 150, zodat deze zowel te zien is op een witte als zwarte achtergrond. De code van deze methode is te vinden in Figuur 18

```
//Finding all white pixels and averaging the position of each the white pixels. Giving the centroid.
for (int h = 0; h < HEIGHT; h++) {
    for (int w = 0; w < WIDTH; w++) {
        if (content[h][w] == 255) {
            temphCentroid += h;
            tempwCentroid += w;
            count++;
        }

    }
}
if(count!=0)
location = make_pair((tempwCentroid / count), (temphCentroid / count));
else location = make_pair(0, 0);


cout<<location.first<<","<<location.second<<endl;
//putting a dot on the centroid of the marble
content[location.first][location.second] = 150;

//saving picture with dot on centroid marble
savePicture("centroid.pgm");
```

FIGUUR 18 - CODE LOKALISEREN KNIKKER

Opslaan afbeelding

Het resultaat van alle bewerkingen moest worden opgeslagen om te verifiëren of de methodes naar behoren werken. Hiervoor is het .pgm [17] formaat gebruikt. Dit is een simpel formaat zonder compressie. Deze wordt veel gebruikt in beeldbewerking, omdat er geen codering en decodering nodig is. De waardes van de pixel zijn een op een met de waardes van de afbeelding in de software. Om te zorgen dat de computer het bestand herkent als .pgm en de dimensies van de afbeeldingen kloppen, moet er een header worden toegevoegd. Deze bestaat uit een:

- Magic nummer voor het aangeven van bestandsformaat
- Breedte
- Hoogte
- Maximale grijswaarde

Bij deze afbeelding zijn deze waardes: P5, 640, 480 en 255.

6.3 Testen: positie bepalen knikker

In dit deelhoofdstuk wordt de testfase van increment 2 behandeld.

Binnen dit increment is er getest of de knikker naar behoren is herkend en of de positie goed is voorspeld. Om dit aan te tonen worden het resultaat van de beeldbewerkingsmethodes opgeslagen in een afbeelding. Door creëren van een resultaat wordt er ook voldaan aan eis FE07. In Tabel 20 is een testcase opgesteld die is gebruikt bij het testen van eis FE01 en FE07. De test voor het lokaliseren van de knikker is hier niet opgesteld, maar is te vinden in bijlage E.

Test	3. FE01 FE07
Use-case	UC03: Edit picture
Omschrijving	Bewerken van een afbeelding om knikker te onderscheiden.
Pre-conditie	Er in een afbeelding ‘picture.jpg’ beschikbaar, waarin een knikker afgebeeld is. Deze .jpg staat in de map van marbleVoorspeller.exe
Testproces	1. Ga naar map waar marbelVoorspeller.exe is opgeslagen 2. Open marbleVoorspeller.exe
Verwachte uitkomst	In de map van marbleVoorspeller.exe is de afbeelding threshold.pgm gemaakt met het resultaat van de bewerkingen. In deze afbeeldingen is de knikker wit en de rest zwart.
Uitkomst	Uitkomst is zoals verwacht.

TABEL 20 - TEST UC03

In dit increment is aan alle eisen voldaan. De positie van de knikker werd opgeslagen, zodat deze in het volgende increment zou kunnen worden gebruikt. De resultaten van de testen zijn besproken met de opdrachtgever. Dit was tevens het evaluatiemoment van dit increment. Hier kwam naar voren dat de opdrachtgever tevreden was met het resultaat.

7. Increment 3: Positie voorspellen knikker

In increment 3 lag de focus op het voorspellen van de positie van de knikker. Hierbij wordt er gebruikt gemaakt van de knikkerposities die zijn bepaald in increment 2. Daarnaast moest binnen dit increment het vision-algoritme geïmplementeerd worden op de Bachmann PLC. Waarbij er ook gekeken wordt of de PLC geschikt is voor het uitvoeren van computer-vision. De eisen die binnen dit increment zijn uitgevoerd zijn opgesteld in Tabel 21.

Increment	Eisen
3: Positie voorspellen knikker	• FE03 • FE04 • FE08 • FE09

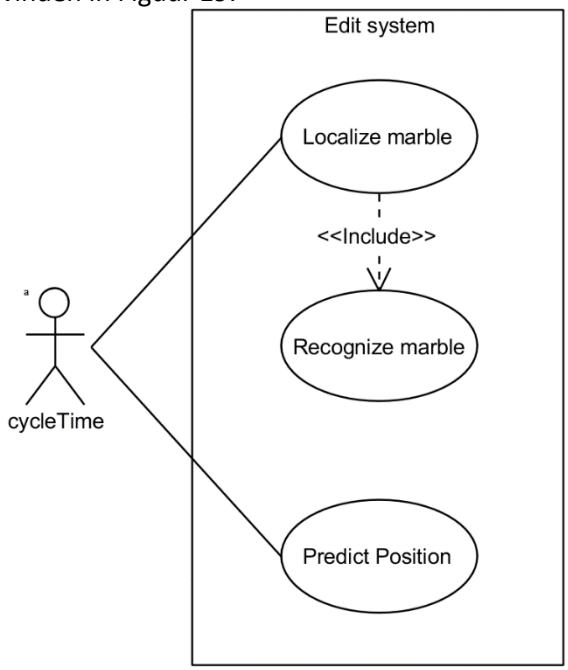
TABEL 21 - EISEN INCREMENT: POSITIE VOORSPELLEN KNIKKER

7.1 Ontwerpen: Positie voorspellen knikker

Tijdens dit increment is de taak van increment 2 aangepast. Hierdoor is er weinig nieuws qua ontwerp. Daarom zijn is dit hoofdstuk het volledige ontwerp gegeven. Hierbij is bij het klassendiagram van incrementen 1 en 2 gecombineerd. Alle ontwerpen zijn ook in bijlage C opgesteld.

7.1.1 Use-casediagram: Positie voorspellen knikker

Tijdens dit increment waren de twee belangrijkste eisen de positievoorspelling van de knikker en de implementatie van het algoritme op de PLC. Daarnaast was er gepland dat de kleurherkenning van de knikker zou worden geïmplementeerd. Dit had echter een minder hoge prioriteit. Hierdoor is er eerst gefocust op de positievoorspelling en de implementatie op de PLC. Voor dit increment is het use-casediagram van increment 2 aangepast. Hierbij is de positievoorspelling toegevoegd. Het diagram is te vinden in Figuur 19.



FIGUUR 19 - USE-CASEDIAGRAM: POSITIE VOORSPELLING KNIKKER

De bijbehorende beschrijving is gegeven in Tabel 22.

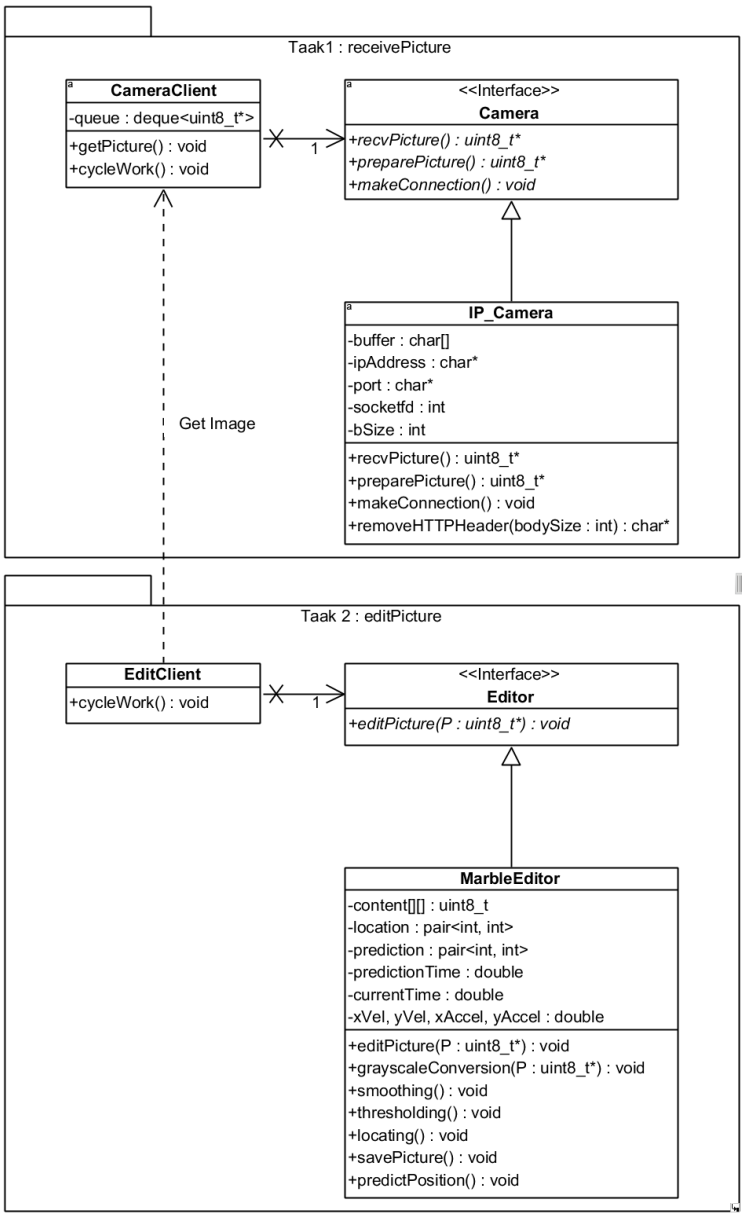
ID:	UC5
Use case:	Predict Position
Brief description:	Het voorspellen van de positie van een knikker
Primary actors:	Cycle Time
Secondary actors:	Geen
Preconditions:	1. De vorige en huidige positie van de knikker is beschikbaar.
Main flow:	1. De PLC haalt de huidige positie. 2. De PLC berekend voorspelling. 3. De PLC staat de voorspelling op.
Postconditions:	1. Er is een positie beschikbaar voor verder gebruik.
Alternative flows:	None

TABEL 22 - BESCHRIJVING 'PREDICT POSITION'

7.1.2 Klassendiagram: Positie voorspellen knikker

Tijdens dit increment is er weinig veranderd aan het klassendiagram. Er is in MarbleEditor alleen de methode '*predictPosition()*' toegevoegd. Daarnaast zijn er attributen toegevoerd die nodig zijn om de positie te kunnen voorspelling en zijn er bestaande attributen aangepast om de code overzichtelijker te maken. Aangezien er weinig aanpassingen zijn gemaakt, is in Figuur 20 het totale klassendiagram gegeven van alle incrementen samen. Deze is ook te vinden in het bijlage C.

In dit increment moest er geen nieuw sequentiediagram gemaakt worden, maar is het sequentiediagram van increment 2 aangepast. Hierbij is de nieuwe methode toegevoegd en is er een loop om *´smoothing()´* gezet, om aan te geven dat deze functie 3x wordt uitgevoerd. Aangezien er weinig aanpassingen gemaakt zijn is het sequentiediagram alleen opgesteld in bijlage C.



FIGUUR 20 - VOLLEDIG KLASSENDIAGRAM

7.2 Realisatie: Positie voorspellen knikker

In dit hoofdstuk wordt de realisatiefase van het derde increment behandeld.

Tijdens de realisatie van increment 3 moest er gekeken worden hoe de voorspellen zou worden uitgevoerd. Daarnaast werd er tijdens dit increment getest of de Bachmann PLC geschikt is voor beeldbewerking. Hiervoor is de executie tijd van het algoritme gemeten op zowel de Bachmann PLC als de PC.

7.2.1 Analyse positievoorspelling

Voor het maken van de positie voorspelling waren er de volgende vragen die beantwoord moesten worden.

- Hoe kan de positie voorspeld worden?
- Welke gegevens zijn er nodig voor het voorspellen van de positie?
- Hoe kunnen de gegevens die bekend zijn gebruikt worden voor het voorspellen van de positie?

Het voorspellen van een positie kan op verschillende manieren afhankelijk van de situatie. Zo is het voorspellen van een positie moeilijker als er bochten gemaakt worden door het te voorspellen object. Bij dit project is er echter spaken van een lineaire bewegingen over de knikkerbaan. Bij een lineaire beweging kan de positie voorspelt worden door middel van acceleratie. Met acceleratie kan de snelheid en plaatst worden berekend, als de tijd bekend is. Dit kan worden gemeten in de code.

Zwaartekracht en wrijvingskracht

In het geval van een naar beneden rollende knikker, resulteert de acceleratie uit zwaartekracht. Naast zwaartekracht wordt wrijvingskracht uitgeoefend op de knikker, waardoor de acceleratie lager wordt. Doordat knikker niet recht naar beneden valt, maar op een helling naar beneden gaat, telt de zwaartekracht niet volledig voor de acceleratie, maar is er een resulterende kracht afhankelijk van de hoek van de knikkerbaan. Als deze resulterende kracht wordt opgeheven met de wrijvingskracht, kan de acceleratie van de knikker worden berekend. Hiermee kan ook de positie voorspeld kunnen worden, mist de acceleratie constant blijft. Het nadeel van deze methode is echter dat de berekening afhankelijk is van de hellingshoek van de knikkerbaan. Hierdoor moet het algoritme aangepast worden als de knikkerbaan aangepast wordt. Hierdoor is er gekeken naar een voorspelling die onafhankelijk is van zwaartekracht en wrijving.

Positie, snelheid en acceleratie

Een andere manier van voorspellen is het gebruikt maken van de positiebepaling van increment 2. Door na elk frame de positie te vergelijken met de vorige bekende positie kan de snelheid bepaald worden. Als dit wordt herhaald kan het verschil in snelheid gemeten worden en zo ook de acceleratie. Met deze methode is het voorspellen onafhankelijk van de knikkerbaan, mits deze lineair is. Hierdoor hoeft het algoritme niet aangepast te worden dat de hellingshoek verandert. Daarnaast worden veranderingen in acceleratie ook meegenomen in de voorspelling, omdat deze met elke nieuwe positie wordt geüpdatet. Door een berekening uit te voeren met de huidige positie, snelheid en acceleratie kan de positie voorspeld worden. De formule voor de berekening is hieronder getoond. [18]

$$x_1 = x_0 + v * \Delta t + \frac{1}{2} * a * \Delta t^2$$

x_1 = te voorspellen positie, x_0 = huidige positie, v = huidige snelheid, a = huidige acceleratie en Δt = verschil in tijd tussen nu en tijd van de te voorspellen positie.

7.2.2 Implementatie Bachmann PLC

Tijdens dit increment moest het vision-algoritme dat is ontwikkeld op de PC geïmplementeerd worden op de PLC. Hierbij hoefde er aan het ontwerp niks veranderd te worden. Echter waren er libraries voor het aanmaken van sockets die hernoemd moesten worden. Echter was de functionaliteit het zelfde. Het algoritme is geïmplementeerd op de PLC voor de implementatie van de positievoorspelling, zodat mogelijke fouten in het huidige algoritme opgelost konden worden.

Probleem met camera

Tijdens het implementeren van het algoritme op de PLC was het de eerste keer dat er meerdere keren achter elkaar een afbeelding werd opgevraagd. Hiervoor was er telkens één afbeelding gebruikt om het algoritme te testen aangezien het resultaat niet afhankelijk was van meerdere afbeeldingen. Echter werkt de PLC cyclisch waardoor het algoritme automatisch meerdere keren wordt uitgevoerd. Toen er voor de tweede keer een afbeelding werd opgehaald, gaf de PLC een errormelding. Hierdoor moest de PLC opnieuw opgestart worden. Na dit probleem te hebben getest, bleek het dat de camera de http-verbinding afsloot, nadat er een afbeelding was opgevraagd. Hierdoor moest er per cyclus opnieuw connectie worden gemaakt met de camera. Dit zorgde ervoor dat de tijd tussen twee foto's tussen de 500ms en 1000ms was. Dit was niet geschikt voor het voorspellen van de positie aangezien de knikker er 2500ms over doet om op de knikkerbaan van boven naar beneden te rollen. Hierbij zou je 2 tot 5 foto's kunnen vergelijken als de te voorspellen positie op 2500 ms is. Als er na 1000ms moet worden voorspeld kan het voorkomen dat er één foto beschikbaar is.

Oplossing voor camera

Een oplossing hiervoor zou zijn om de camera een constante stream aan afbeeldingen te laten sturen en deze te ontvangen. Hierdoor is er geen einde aan het http-response, dus wordt de connectie niet gesloten. Echter deed deze functie het tegen verwachting in niet op de FOSCAM fi9853ep. In de handleiding stond dat dit wel zou moeten kunnen, maar na verder onderzoek kwam er naar voren dat de hardware het streamen ondersteunt, maar de firmware echter niet. Het bleek dus dat deze camera niet geschikt was voor de realisatie voor het project.

Na dit probleem met de opdrachtgever te hebben besproken is ervoor gekozen om niet te zoeken naar een nieuwe camera die mogelijk wel werkt, maar de focus te leggen op de werking van het algoritme en de uitvoering hiervan op de PLC. Hiervoor is er een filmpje gemaakt van een knikker die van de knikkerbaan rolt. Uit dit filmpje zijn met behulp van VLC viewer afbeeldingen gehaald, die worden gebruikt om de het algoritme te testen. Deze afbeeldingen zijn opgeslagen als .jpg om het ophalen van de afbeeldingen bij de camera te simuleren. Door de problemen met de camera duurde deze eis langer dan gepland. Hierdoor was er geen tijd meer in increment 3 om de eisen FE08 en FE09 te realiseren.

7.2.3 Implementatie positievoorspelling

Voor de implementatie van de positievoorspelling is de formule gebruikt die is opgesteld in hoofdstuk 7.2.3. Daarnaast zijn de foto's gebruikt die zijn gehaald uit het gemaakte filmpje tijdens de implementatie op de PLC. In eerste instantie zijn dit foto's die om de 100ms gemaakt zijn dit staat gelijk aan 10 fps (frames per seconde). Later wordt er getest met hogere en lagere fps.

Snelheid knikker

Voordat de formule gebruikt kon worden voor de voorspelling, moest eerst de snelheid en acceleratie van de knikker berekend worden. Om de snelheid te berekenen moet de afstand tussen twee foto's gemeten worden en moet deze afstand worden gedeeld door de tijd tussen de twee foto's. Het resultaat is de snelheid met als eenheid pixels per milliseconde (pix/ms). De tijd die wordt gebruikt om de snelheid te berekenen is afhankelijk van de fps waarmee de afbeeldingen worden opgehaald. Bij de beschikbaar gestelde foto's is dit 100ms.

Acceleratie knikker

Nadat de snelheid van de knikker is berekend kan de acceleratie worden berekend. Hiervoor moet met een derde foto opnieuw de snelheid van de knikker bepaald worden. Deze nieuwe snelheid moet vergeleken worden met de vorige snelheid, zodat de verandering in snelheid kan worden bepaald. Deze verandering moet gedeeld worden door de tijd tussen frames (100ms). Het resultaat hiervan is de huidige acceleratie van de knikker in pix/ms². De snelheid en acceleratie worden elke frame geüpdatet, waardoor de voorspelling aangepast wordt op basis van verandering in snelheid en acceleratie.

Positievoorspelling

Met de implementatie van de snelheid en acceleratie bepaling kon de formule voor de positievoorspelling geïmplementeerd worden. De enige variabele in de formule die nog niet berekend was, was de tijd tot de te voorspellen positie. Dit kon echter simpel berekend worden door de huidige tijd af te trekken van de tijd waarop de knikkerpositie voorspelt moet worden. Hiermee zijn alle gegevens bekend en kunnen deze ingevoerd worden in de formule. Deze is zowel ingevoerd voor de breedte van de foto, als voor de hoogte. Echter was voor de huidige voorspelling alleen de voorspelling in de breedte relevant, aangezien de knikkerbaan in de hoogte constant was, dus de knikker positie in de hoogte van de foto veranderde niet.

De bovengenoemde uitleg resulteerde in de code die te zien is in Figuur 21.

```
//Setting old position
oldXVel = xVel;
oldYVel = yVel;

//calculating velocity and acceleration of the marble
if(location.first != 0 && location.second !=0 && oldLocation.first != 0 && oldLocation.second != 0){
    xVel = (double)(location.first - oldLocation.first)/timePerFrame;
    yVel = (double)(location.second - oldLocation.second)/timePerFrame;

    xAccel = (xVel - oldXVel)/timePerFrame;
    yAccel = (yVel - oldYVel)/timePerFrame;
}
//Predicting the position of the marble based on current location, velocity, acceleration and remaining time.
if(currentTime<=msTime && location.first != 0 && location.second !=0 && oldLocation.first != 0 && oldLocation.second != 0){
    predX = location.first + (xVel*(msTime - currentTime)) + 0.5*(xAccel*pow((msTime - currentTime),2));
    predY = location.second + (yVel*(msTime - currentTime)) + 0.5*(yAccel*pow((msTime - currentTime),2));

}
```

FIGUUR 21 - IMPLEMENTATIE POSITIEVOORSPELLING

7.2.4 Controle positievoorspelling

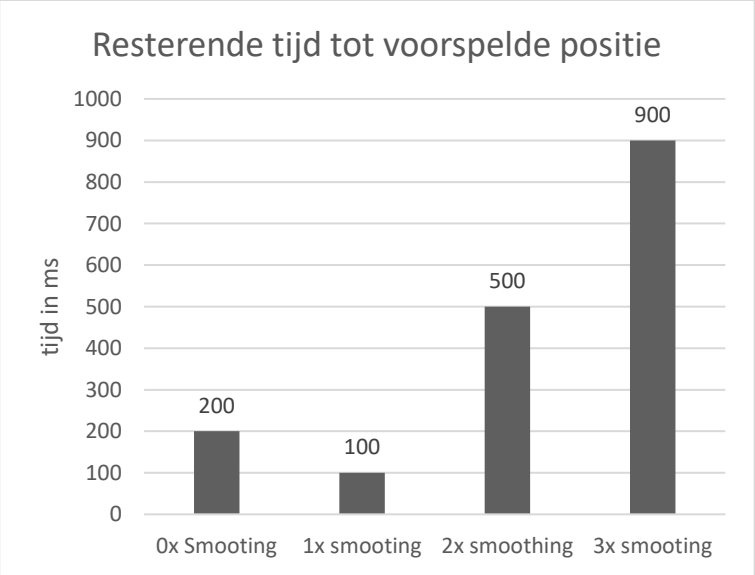
Na het implementeren van de positievoorspelling moest er gecontroleerd worden of de werking correct was. Hiervoor is het algoritme uitgevoerd onder verschillende omstandigheden en met verschillende parameters. Zo is de positievoorspelling onder de volgende omstandigheden uitgevoerd:

- Kwaliteit van voorspellen zonder smoothing en herhaalde smoothing.
- Looptijd van het algoritme op PC en Bachmann PLC met en zonder smoothing.
- Kwaliteit van voorspelling met verschillende fps.

Bij de controles is er telkens één parameter aangepast. Als uitgangsituatie is de fps 10, wordt smoothing 3x herhaald en is de te voorspelling positie op het tijdstip 1500ms. Dit waren de instellingen bij de controles tenzij anders wordt aangegeven. Bij de controles wordt ook bepaald of de Bachmann PLC geschikt is voor het uitvoeren van het vision-algoritme. Hierbij wordt de looptijd van het algoritme vergeleken met de duur van het gemaakte filmpje. De duur van het filmpje kwam overeen met de duur dat de knikker erover doet om de knikkerbaan af te leggen, 2500ms. Met 10fps zijn dit 25 foto's die verwerkt moeten worden door het algoritme.

Geen smoothing tot herhaalde smooting

Voor de eerste controle werd er gekeken naar de invloed van smoothing op de kwaliteit van de positievoorspelling. Dit was tevens een controle hoe vaak smoothing nodig is om de knikker goed te herkennen. In Figuur 22 is het resultaat van de controle gegeven. Hierbij is het aantal keer dat smoothing is uitgevoerd opgezet tegen de resterende tijd waarna alle voorspellingen binnen de marge van één knikker vallen.

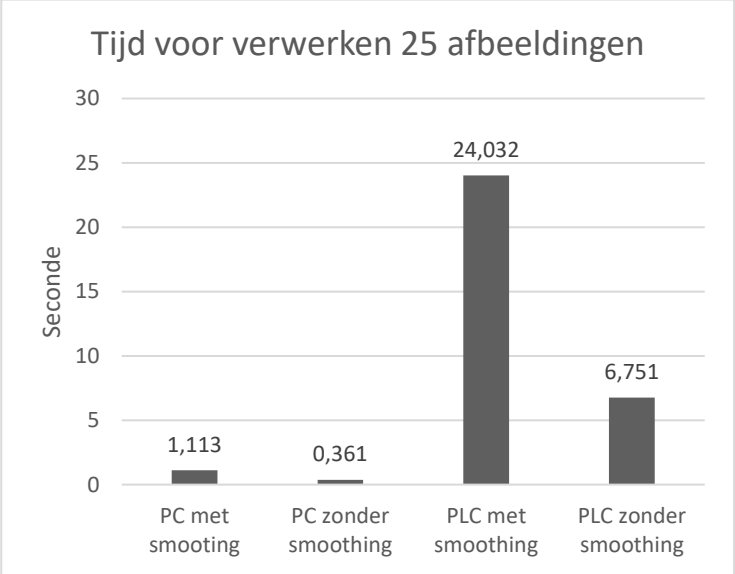


FIGUUR 22 - CONTROLE INVLOED SMOOTHING OP VOORSPELLING

In het bovenstaande figuur is duidelijk de invloed van smoothing te zien op de kwaliteit van de voorspelling. Dit komt doordat bij minder smoothing ruis slechter wordt verwijderd, dus de positiebepaling minder precies wordt. Hierdoor wordt de voorspelling gemaakt met posities die niet overeenkomen met de daadwerkelijke posities van de knikker. Door de controle bleek dat 3x smoothing het beste resultaat gaf. In de volgende controle werd de invloed van smoothing op de executietijd van het algoritme gecontroleerd. Dit is namelijk de zwaarste methode in het algoritme.

Executietijd met en zonder smoothing

Voor deze controle is er gekeken naar de executietijd van het algoritme op de PC en de Bachmann PLC. Hierbij werd er gekeken de geschiktheid van de PLC voor het uitvoeren van een vision-algoritme en naar de invloed van smoothing op de executietijd. Aangezien het filmpje 2500ms duurt, is het gewenst dat de executietijd 2500ms is of lager. Als het langer duurt kunnen de bewerkingen de knikker niet bijhouden. Hierdoor zal de voorspelling te laat gemaakt worden of worden er met het gebruik van een camera te weinig foto's gemaakt voor een correcte voorspelling. Voor deze controle is er 2x smoothing gebruikt. De resultaten zijn in Figuur 23 getoond.



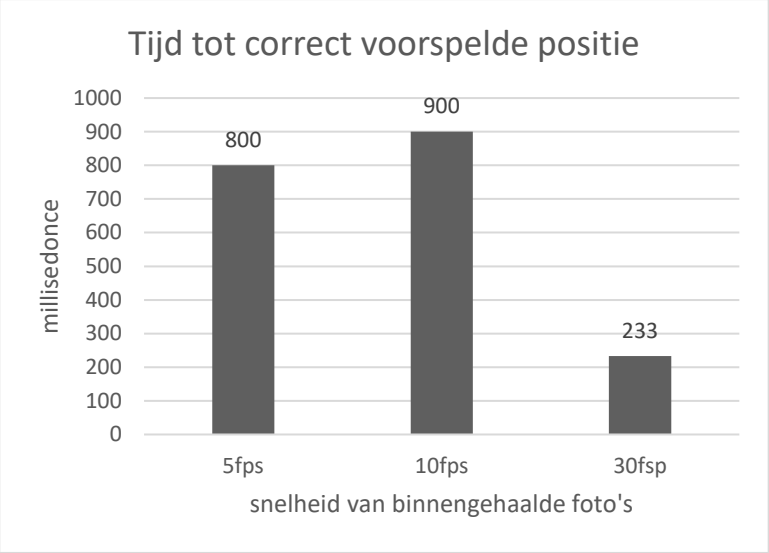
FIGUUR 23 - EXECUTIETIJD MET EN ZONDER SMOOTHING

In het bovenstaande figuur vallen er twee dingen op. De belangrijkste is de duur van de executietijd op de Bachmann PLC. De executietijd van het algoritme op de PLC was 24 seconde. Dit houdt in dat er de PLC er bijna een seconde over doet om één frame te verwerken. Dit is een factor 10 langzamer dan gewenst. Hieruit blijkt dat deze Bachmann PLC niet geschikt is voor het uitvoeren van het vision-algoritme. Dit komt doordat de processor van de PLC niet krachtig genoeg is om de operaties, die nodig zijn voor het bewerken van een afbeelding, uit te voeren. Ondanks dat de PLC niet krachtig genoeg is, werkt het vision-algoritme wel naar behoren. Als er een krachtigere Bachmann PLC zou worden gebruikt kan het algoritme waarschijnlijk wel binnen de gewenste tijd uitgevoerd worden.

Naast het verschil tussen de executietijd op de PC en PLC, is er ook een groot verschil in het gebruik van smoothing. Met 2x smoothing is het verschil bijna 18 seconde op de PLC. Dit is bijna een verdriedubbeling van de executietijd. Dit houdt in dat smoothing bijna 9 seconde duurt. Echter is smoothing een belangrijk algoritme voor de knikkerherkenning en de positievoorspelling, zoals in figuur 19 is aangetoond.

Positievoorspelling met verschillende fps

In de laatste controle wordt er gekeken naar de positievoorspelling waarbij de tijd tussen frames anders is. Zo wordt er gekeken naar de voorspelling met de tijd tussen foto's van 33ms, 100ms en 200ms. Hierbij wordt er vergeleken wat de resterende tijd is nadat de knikker positie correct voorspeld is. Het resultaat is opgesteld in Figuur 24.

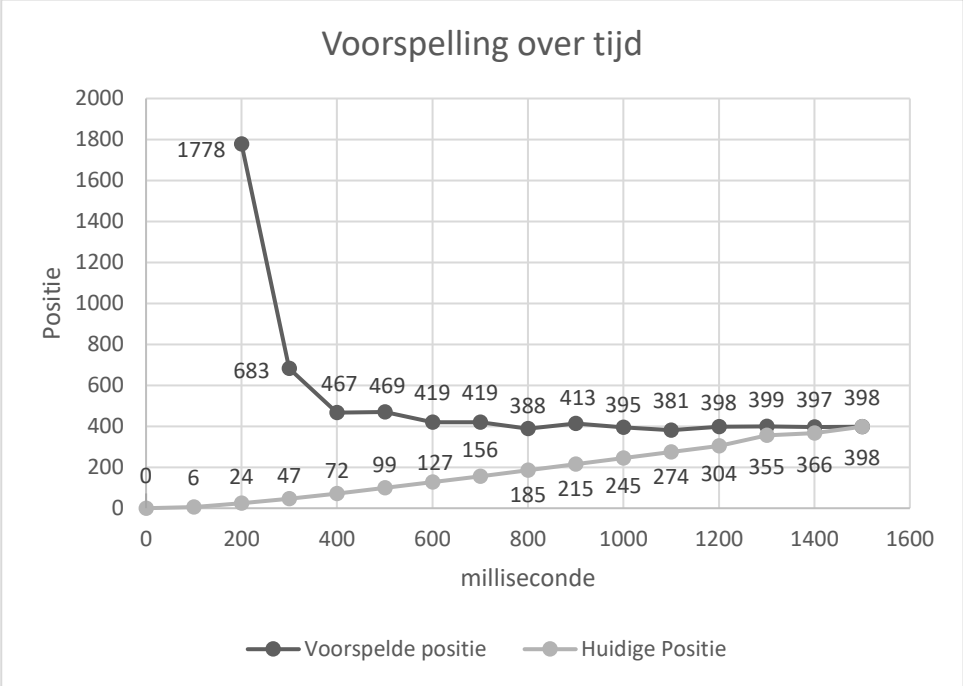


FIGUUR 24 - POSITIEVOORSPELLING MET VERSCHILLENDE FPS

De resultaten van de controle laten zien dat bij verschillende fps er een duidelijk verschil is in de tijd die over is na een correcte voorspelling. Hierbij is het belangrijk dat er niet alleen naar de tijd gekeken wordt, maar ook naar het aantal foto's die zijn opgehaald nadat de correcte voorspelling is gemaakt. Namelijk bij 30 fps is de voorspelling gemaakt, 7 foto's voordat de knikker bij voorspelde positie is. Bij 5 fps is dit 4 foto's. Hierdoor heeft elke situatie zijn eigen voor en nadelen. Zo is de 30fps geschikt bij knikkerbanen met een hoge hellingshoek. Hierbij beweegt de knikker sneller, dus moet de reactietijd van het algoritme sneller zijn. Echter is de 5 fps geschikt voor een knikkerbaan met een lage hellingshoek en bij systemen met lage rekenkracht. 10 fps zit hier tussenin en heeft de beste voorspelling en heeft de voorspelling goed 9 foto's voordat de knikker bij de voorspelde positie is.

Resultaten

De resultaten zijn met de opdrachtgever besproken. Hierbij er gekeken of het algoritme op tijd de correcte voorspelling gaf. Door te kijken naar de snelheid van een mogelijk toe te voegen robotarm om de knikker op de voorspelde positie op de pakken, is er besloten dat de voorspelling snel genoeg gemaakt wordt. Het is waarschijnlijk dat de robotarm snel genoeg is om in 900ms naar de gewenste positie te bewegen. Dit is echter alleen besproken en niet getest, aangezien er geen robotarm aanwezig was. Het resultaat van de positievoorspelling met 10 fps is in Figuur 25 getoond. Hierin is de huidige positie en voorspelde positie weergegeven op elk tijdstip waar een foto is opgehaald.



FIGUUR 25 - RESULTAAT POSITIEVOORSPELLING

Er is te zien dat de eerste voorspelling er ver naast zit en dat daarna de voorspelling snel beter wordt. De voorspelling zit al snel binnen de dimensies van de afbeelding. Als op dat punt de robotarm al zou worden aangestuurd, hoeft de arm als er wel een goede voorspellen wordt gemaakt minder ver te bewegen.

Na het implementeren van de positievoorspelling en het implementeren van het algoritme op de Bachmann PLC, was er geen tijd meer om de eisen FE08 en FE09 te implementeren. Dit volgde uit de problemen met de onverwachte problemen met de camera. Hierdoor duurde het implementeren van het algoritme op de PLC langer dan gepland.

7.3 Testen: positie bepalen knikker

In dit deelhoofdstuk wordt de testfase van increment 3 beschreven.

Binnen dit increment is er getest of de positie van de knikker op tijd wordt voorspeld. Om dit aan te tonen is er een lijst gegenereerd dat bij elke positiebepaling ook een voorspelling geeft. Hierdoor kan er gekeken worden vanaf welk punt de positie goed is voorspeld. In Tabel 23 is een testcase opgesteld die is gebruikt bij het testen van eis FE03 en FE04. Als de test geslaagd is, is er ook voldaan aan de eis SE07.

Test	1. FE03 FE04
Use-case	UC05: Predict Position
Omschrijving	Met de Bachmann PLC de voorspelling van de positie op 1500ms uitvoeren.
Pre-conditie	De Bachmann PLC staat aan en het programma is geüpload. Device-view is geopend in Solutioncenter.
Testproces	1. Ga bij device naar de geïnstalleerde applicaties. 2. Start MarblePredictor.
Verwachte uitkomst	Er wordt na elke cyclus een huidige positie getoond met een positievoorspelling waar de knikker op het tijdstip 1500ms gaat zijn. 900ms voordat de knikker op het te voorspellen punt is, wordt er een voorspelling gegeven die maximaal 30 pixels afwijking heeft.
Uitkomst	Uitkomst is zoals verwacht.

TABEL 23 - TEST UC05

In dit increment is alleen voldaan aan de eisen FE03 en FE04. De positie van de knikker werd correct voorspeld. Hiermee is ook voldaan aan de systeemeis SE07. De resultaten van de testen zijn besproken met de opdrachtgever. Dit was tevens het evaluatiemoment van dit increment. Hier kwam naar voren dat de opdrachtgever tevreden was met het resultaat.

7.3.1 Geïmplementeerde eisen

Na de testen van increment drie was de incrementele fase afgerond. Tijdens incrementen zijn alle systeemeisen gerealiseerd, maar niet alle functionele eisen. In Tabel 24 is een overzicht van de functionele eisen die zijn gerealiseerd.

ID	Eis	Prioriteit	Resultaat
FE01	Het vision-algoritme kan een non-transparante knikker herkennen.	M	Voldaan
FE02	Het vision-algoritme kan de positie van de knikker bepalen.	M	Voldaan
FE03	Het vision-algoritme kan de plek van een knikker op een knikkerbaan kan voorspellen.	M	Voldaan
FE04	Het vision-algoritme kan op de Bachmann PLC worden uitgevoerd in de programmacyclus	M	Voldaan
FE05	Het vision-algoritme kan de afbeelding van JPG naar een bewerkbaar beeld omzetten.	M	Voldaan
FE06	De PLC moet een afbeelding kunnen krijgen van de IP-camera.	M	Voldaan
FE07	De PLC moet een output genereren waarbij het resultaat van het vision-algoritme getoond wordt.	M	Voldaan
FE08	Het vision-algoritme kan de kleur van de knikker kan herkennen.	S	Niet voldaan
FE09	Het vision-algoritme voorspelt alleen de positie van knikker met de verwachte kleur.	S	Niet voldaan
FE10	De PLC moet een cartesisch coördinaat genereren op basis van de voorspelde knikkerpositie in de afbeelding.	C	Niet voldaan
FE11	De knikkervoorspelling moet gecontroleerd kunnen worden door middel van een interface.	C	Niet voldaan

TABEL 24 - RESULTATEN FUNCTIONELE EISEN

8. Conclusie

Het doel van de afstudeeropdracht was het implementeren van een vision-algoritme op een Bachmann PLC om knikkers te herkennen en de positie hiervan de voorspellen op basis van kleur. Tijdens de opdracht is het vision-algoritme in drie incrementen geïmplementeerd. Hierbij werd er eerst een afbeeldingen opgehaald bij de camera. Waarna deze afbeelding werd bewerkt zodat de knikker kon worden herkend en de positie hiervan kon worden bepaald. Als laatste in het algoritme geïmplementeerd op de Bachmann PLC en is de positievoorspelling toegevoegd.

Bij het uitvoeren van het laatste increment kwam naar voren dat de Bachmann PLC niet snel genoeg was om het vision-algoritme uit te voeren. Hierbij was de werking van het algoritme correct, maar was de executietijd van het algoritme op de PLC te lang. Om dit op te lossen zou er een krachtigere Bachmann PLC moeten worden aangeschaft. Zo kan de executietijd omlaag gaan en kan de werking van het algoritme behouden worden.

Naast de problemen met de PLC waren er in het laatste increment problemen met de IP-camera. Deze bleek na elke afbeelding de connectie te verbreken. Hierdoor duurde het ophalen van een afbeelding te lang. Het was niet mogelijk om een stream te openen doordat de firmware hiervoor niet beschikbaar was. Als oplossing is er een filmpje gemaakt waaruit afbeeldingen zijn gehaald. Deze afbeeldingen zijn gebruikt voor het testen van het algoritme en voor de uitvoering van het algoritme op de PLC. Hierdoor bleek dat het algoritme wel werkte. Bij het vervolg van deze opdracht moet er gekeken worden naar een IP-camera die wel een stream van afbeeldingen kan opzetten, of moet er een USB-driver geschreven worden voor het gebruik van een USB-camera.

Het vision-algoritme is in staat 900ms voordat de knikker bij de te voorspellen positie is een correcte voorspelling te geven met een nauwkeurigheid van één knikker afwijking. De opdrachtgever is tevreden met die resultaat. Het algoritme kan daarom in de toekomst gebruikt worden om de knikker op de voorspelde positie te laten oppakken door een robotarm. Hiervoor moet de positie die is voorspeld omgezet worden tot een commando dat naar de robotarm kan woorden gestuurd.

9. Evaluatie

9.1 Proces

In dit hoofdstuk wordt besproken of de gemaakte keuzes voldeden aan de verwachtingen en wordt er gekeken naar de correctheid en fouten in de werkwijze. Hierbij worden ook de verbeterpunten benoemd en hoe het in de toekomst anders zou worden gedaan.

Bij het uitvoeren het afstudeerproject was de projectmethode RAD gebruikt. Bij de keuze van de een mogelijk te gebruiken methode was er gekeken naar verschillende voor- en nadelen van meerdere strategieën en methoden. Bij het zoeken van alternatieve methodes waren er te snel methodes opgesteld die bekend waren. Hierdoor is er enigszins met tunnelvisie gekeken naar de verschillende methodes. Dit wil niet zeggen dat de gekozen methode incorrect is, maar er waren misschien andere methodes geweest die ook correct waren. In de toekomst kunnen er meer methodes vergeleken kunnen worden en een dieper onderzoek gedaan worden waarvoor deze methodes gebruikt kunnen worden. Verder is RAD toegepast hoe het is aangeleerd op de opleiding. Dit werkte goed binnen de uitvoering van het project.

Bij de analyse van het project is er systematisch te werk gegaan om gebrekkige kennis op te doen en keuzes te voor onderdelen die werden gebruikt. Hierbij moest er een keuze gemaakt worden welke camera zou worden gebruikt. Hierbij is er alleen gekeken naar de camera's die konden worden aangesloten op de PLC. Deze zijn met elkaar vergeleken op basis van opgestelde criteria. Bij het vergelijken van deze criteria was er te veel gelet op de specificaties op papier en niet of dit ook vertaald zou worden naar de realiteit. Hierdoor is er te snel een keuze gemaakt voor een IP-camera. Later in het project bleek dat deze camera niet geschikt was voor het uitvoeren van de opdracht. Hierdoor moest er een oplossing bedacht worden. Dit koste extra tijd in het increment 3, waardoor er twee eisen niet konden worden uitgevoerd. Ondanks dat de IP-camera niet ideaal was, zou het gebruik van een USB-camera ook lastig zijn. Hiervoor zou een driver zou moeten worden geschreven. Als deze camera was gekozen, was er misschien geen tijd meer geweest voor het realiseren van het algoritme.

Het ontwerpen van de software ging goed. Hierbij is er structureel gewerkt. Eerst is er in elk increment de eisen omgezet naar use-cases, waarna het klassendiagram is opgesteld. Daarna is aan de hand van de use-cases en het klassendiagram een sequentiediagram opgesteld. Door deze structuur aan te houden zijn de ontwerpen beter te begrijpen en is er af te leiden waar de ontwerpen vandaan komen. Daarnaast is er zo ontworpen dat de software die erop gebaseerd is, onderhoudbaar en uitbreidbaar is. Doordat er veel tijd is gestoken in het ontwerpen ging het realiseren goed en was er weinig opvallend gebeurd tijdens de realisatie van het ontwerp.

Het opstellen en uitvoeren van de testen ging goed. Hierbij was er per eis gekeken waaraan deze moest voldoen en is er een test opgesteld waarbij het resultaat aan zou tonen of er aan de eis is voldaan. Na elke test is er daarnaast gekeken of de opdrachtgever tevreden was of dat er aanpassingen aan het product gemaakt moesten worden. Dit verliep volledig naar wens. Alleen in het laatste increment hadden de testen van de niet uitgevoerde eisen wel opgesteld moeten worden. In de toekomst moeten eerst alle testen gemaakt worden voor alle eisen die zijn ingepland.

In zijn geheel is het project goed verlopen. In de toekomst zou een soortgelijk project waarschijnlijk ook op de zelfde manier uitgevoerd worden met wat aanpassingen, zodat keuzes beter worden gemaakt.

9.2 Beroepstaken

In dit deelhoofdstuk wordt genoemd waarom er is voldaan aan de beroepstaken die zijn opgesteld in het afstudeerplan.

A1: Analyseren probleemdomein & opstellen probleemstelling
Bij het analyseren van het probleemdomein is er onderzocht welke onderdelen er nodig zijn om de opdracht te voltooien. Hierbij is er gekeken naar de PLC, de camera en de knikkerbaan. Deze onderdelen volgde uit de probleemstelling en doel van de opdracht. In hoofdstuk 3.2 is het probleemdomein opgesteld en in hoofdstuk 4.5.1 het probleemdomein.

A3: Vergaren en analyseren van requirements
Voor het vergaren en analyseren van de eisen is er een analyse uitgevoerd om kennis op te doen binnen het vakgebied. Deze zijn met de klantwensen en het probleemdomein meegenomen om de eisen op te stellen. De eisen zijn gegeven in hoofdstuk 4.5.2.

C6: Ontwerpen software
Tijdens het afstudeerproject is er object georiënteerd geprogrammeerd. Hierbij is er rekening gehouden hoe onderhoudbaar en uitbreidbaar het ontwerp is. Hiervoor zijn in het klassendiagram abstracte klassen gebuikt. Daarnaast zijn alle ontwerpen van elkaar af te leiden. In hoofdstuk 7.1.2 is het volledige klassendiagram getoond dat gemaakt is.

D14: Realiseren van software
Tijdens het afstudeerproject is de ontworpen software gerealiseerd. Hierbij is er gebruikt gemaakt van bestaande libraries, welke zijn gecombineerd met eigen methodes om de gewenste functionaliteit te behalen. In hoofdstuk 7.2.3 is er een stuk code getoond.

D15: Testen
Voor het testen is er een testplan opgesteld. Hierin is voor elk uitgevoerde eis een test geschreven en uitgevoerd. Dit testplan in omgezet naar een testrapport toen de testen zijn uitgevoerd. Een voorbeeld van een test is uitgewerkt in Hoofdstuk 5.3.

Gc: Kritisch, onderzoeken en methodisch werken
Tijdens het afstuderen is er gebruik gemaakt van de methode RAD. Hierbij zijn er de fases die bij RAD horen gevolgd. In het begin van elke fase is er een terugkoppeling gemaakt met de structuur van RAD. Deze staat in het begin van elke nieuwe fase.

Gf: Leren Leren: voorbereiden op volgende studiefase en beroep
Bij de afstudeeropdracht is er geleerd om een volledig project alleen te doorlopen en om feedback te vragen wanneer nodig. Hierbij is er ingezien dat veel communicatie met de opdrachtgever belangrijk is. Hiervoor is er vaak met de opdrachtgever gepraat om de wensen te achterhalen en of de focus van het project goed was. Dit ging niet altijd even vloeiend. De evaluatie van het proces is in hoofdstuk 9.2 omgesteld.

10. Bronvermelding

[1] ALTEN, *Orgranogram ALTEN.*

[2] H. v. d. Bosch, *SO Strategieën en Methoden,* Deflt: De Haagse Hogeschool, 2017.

[3] B. e. GmbH, „Processor Modules," Mei 2018. [Online]. Available: https://www.bachmann.info/fileadmin/media/Produkte/Steuerungssystem/Produktblaetter/MC200-series_en.pdf.

[4] rolfk, „Desktop RT target with USB camera support," 19 november 2015. [Online]. Available: https://forums.ni.com/t5/LabVIEW/Desktop-RT-target-with-USB-camera-support/td-p/3218384.

[5] Wikipedia, „Serial Port," Wikipedia, 29 November 2018. [Online]. Available: https://en.wikipedia.org/wiki/Serial_port.

[6] Adafruit, „TTL Serial JPEG Camera with NTSC Video," Adafruit, [Online]. Available: https://www.adafruit.com/product/397.

[7] Mathlab, „What is image filtering in the spatial domain," Mathlab, [Online]. Available: https://www.mathworks.com/help/images/what-is-image-filtering-in-the-spatial-domain.html.

[8] K. Hong, „MATLAB TUTORIAL : DIGITAL IMAGE PROCESSING 6 - SMOOTHING : LOW PASS FILTER," bogotobogo, 2016. [Online]. Available: https://www.bogotobogo.com/Matlab/Matlab_Tutorial_Digital_Image_Processing_6_Filter_Smoothing_Low_Pass_fspecial_filter2.php.

[9] P. Burghouwt, „Course Documents Image processing & Computer Vision," de Haagse Hogeschool, 2016. [Online]. Available: https://blackboard.hhs.nl/webapps/blackboard/content/listContent.jsp?course_id=_61290_1&content_id=_1945220_1&mode=reset.

[10] FOSCAM, „FOSCAM fi9853ep," [Online]. Available: https://www.foscam.nl/index.php/fi9853ep.html.

[11] Gamma, „Hoeklat grenen wit gegrond 20x20 mm 270 cm," [Online]. Available: https://www.gamma.nl/assortiment/hoeklat-grenen-wit-gegrond-20x20-mm-270-cm/p/B444693.

[12] Foscam, „Foscam IPCamera CGI User Guide," 6 November 2015. [Online]. Available: https://www.foscam.es/descarga/Foscam-IPCamera-CGI-User-Guide-AllPlatforms-2015.11.06.pdf.

[13] S. Barrett, „Github.com," 5 maart 2019. [Online]. Available: https://github.com/nothings/stb/blob/master/stb_image.h.

[14] Wikipedia, „SRGB," Wikipedia, 6 februari 2019. [Online]. Available: https://en.wikipedia.org/wiki/SRGB.

[15] R. Mazur, „RAW pixel," 2014. [Online]. Available: http://rawpixels.net/.

[16] P. Burghouwt, *Vak Computer Vision & ImageProcessing week 5,* Delft, 2018.

[17] J. Poskanzer, „pgm - Netpbm grayscale image format," 2016. [Online]. Available: http://netpbm.sourceforge.net/doc/pgm.html.

[18] B. Waszak, „gamasutra.com," 11 april 2018. [Online]. Available: https://www.gamasutra.com/blogs/BartlomiejWaszak/20181104/329703/Movement_Prediction.php.

Lijst met afkortingen

Afkorting	Betekenis
PLC	Programmable logic controler
RTOS	Real time operating system
USB	Universal serial bus
MB/s	Megabyte per seconde
Gb/s	Gigabit per seconde
ms	Milliseconde
RGB	Rood, groen, blauw
YCrCb	Kleuren formaat gerepresenteerd in een driedimensionale ruimte. Y is lichtintensiteit
HD-filter	Hoogdoorlaat-filter
DFT	Discrete fouriertransformatie
HTTP	Hypertext Transfer Protocol
FTP	File Transfer Protocol
Cartesisch coördinaat	Coördinaat met xyz-waarde
LAN	Local Area Network
Fps	Frames per seconde

Documentbeheer

Versiebeheer

Versie	Datum	Status	Auteur	Opmerking
0.1	06-02-2019	Concept	Hanno van Megchelen	Initiële versie
0.2	11-04-2019	Concept	Hanno van Megchelen	Voor conceptbespreking
0.3	20-04-2019	Concept	Hanno van Megchelen	Voor tussentijds assessment
1.0	31-05-2019	Definitief	Hanno van Megchelen	Afgerond afstudeerverslag

Distributie

Naam	Functie
René van Schie	Technical consultant
Wim Mooijekind	Begeleidend examinator
Edwin Döderlein	Expert Examinator

Bijlage

Bijlage	Titel	Blz.
A	Plan van aanpak	61
B	Analyserapport	73
C	Ontwerprapport	95
D	Code	Digitaal
E	Testrapport	113

Bijlage A:
Plan van Aanpak

Het herkennen en locatie voorspellen van knikkers op basis van kleur met een camera en PLC bij Alten Nederland.

Versie: 1.0
Status: Goedgekeurd
Datum: 31 mei 2019
Auteur: Hanno van Megchelen

Inhoudsopgave

1. Inleiding

ALTEN Nederland B.V. probeert altijd te groeien. Dit gebeurt in de vorm van het aannemen afstudeerders en werknemer en het werven van nieuwe klanten. ALTEN staat hiervoor op beurzen en organiseren kennisdagen. Het is echter niet altijd makkelijk om aan te tonen dat zij een leuk en aantrekkelijk bedrijf is met voldoende kennis is huis. Om dit aan te tonen wil ALTEN een systeem ontwikkelen dat knikkers kan oppakken met een robotarm en deze weer bovenaan neerlegt.

De afstudeeropdracht is een deel dit systeem. Hierbij moet de kleur van de knikker herkend worden en worden voorspeld waar de knikker opgepakt moet worden. Hiervoor is een camera en Bachmann PLC beschikbaar. Het plan van aanpak voor de ontwikkeling van dit vision-algoritme is in dit document opgesteld.

2. Project

In dit hoofdstuk wordt het project in kaart gebracht. Daarbij wordt de probleem-, doelstelling en betrokken personen en instanties van het project opgesteld.

2.1. Probleemstelling

Momenteel worden berekeningen van beeldbewerking niet uitgevoerd op PLC's, maar op een externe computer dat communiceert met de PLC. Dit is echter minder efficiënt, omdat er communicatie moet worden omgezet tussen de externe computer en de PLC. Er is daardoor de behoefte om een vision-algoritme te creëren dat op een Bachmann PLC binnen zijn programmacyclus uitgevoerd kan worden.

2.2. Doelstelling

Het hoofddoel van de opdracht is om in 16 weken een vision-algoritme op een Bachmann PLC te ontwikkelen dat een knikker en de kleur ervan op een lineaire knikkerbaan kan detecteren en zijn positie voorspellen.

2.3. Instanties en Personen

In dit hoofdstuk worden de instanties en personen besproken die te maken hebben met de afstudeeropdracht.

2.3.1. Betrokken Instanties

- ALTEN Nederland B.V.
  Fascinatio Boulevard 582
  2909 LE Capelle a/d IJssel
  Tel: 010-4637700

- De Haagse Hogeschool
  Faculteit IT & Design
  Rotterdamseweg 137
  2628 AL Delft
  Tel: 015-2606200

2.3.2. Betrokken Personen

- Opdrachtgever & bedrijfsmentor:
  René van Schie
  Fascinatio Boulevard 582
  2909 VA Capelle a/d IJssel
  Tel: 010 4637700
  E-mail: rene.van.schie@alten.nl

- Eerste examinator:
  Edwin Doderlein
  Rotterdamseweg 137
  2628 AL Delft
  E-mail: e_doderlein_1989@hotmail.com

- Tweede examinator:
  Wim Mooijekind
  Rotterdamseweg 137
  2628 AL Delft
  E-mail: wmictconsultancy@gmail.com

- Afstudeerder:
  Hanno van Megchelen
  Bosboom-Toussaintplein 115
  2624 DJ Delft
  Tel: 06-52458513
  E-mail: hannomeggel@gmail.com

3. Fasering

In dit hoofdstuk wordt er besproken in welke fases het project is opgedeeld. Bij dit project wordt er gebruik gemaakt van de Rapid Application Development (RAD) methode. Dit is een incrementele strategie waarbij het project na elke increment wordt uitgebreid. In elk increment wordt er ontworpen, geïmplementeerd en getest. Voorafgaande van het incrementele werken wordt een uitgebreide analyse uitgevoerd. In Tabel 25 is een overzicht gegeven welke activiteiten er bij elke fase hoort.

Fasering	Activiteit	Tijdsbestek
Oriëntatiefase	*Maken plan van aanpak* *Maken van (requirement)analyse* *Mogelijk literatuuronderzoek*	*4 weken*
Incrementele werken (Sprints)	*Ontwerpen van software*	*11 weken in iteraties van 2 - 3 weken*
	Implementeren van het ontwerp	
	Testen van de software	
Afronding	*Afronden project en documentatie*	*1 week afronden*

TABEL 25 - GLOBALE FASERING

4. Producten

In hoofdstuk 3 Fasering is genoemd dat RAD gebruikt gaat worden als ontwikkelmethode. Hierbij volgt uit elke fase een product dat opgeleverd moet worden. Echter zijn er ook dynamische producten die aangepast worden na elk increment. Deze producten behoren tot '*Incrementeel werken'.*

- **Oriëntatiefase**
    - Analyserapport met requirements
- **Incrementeel werken**
    - Ontwerprapport
    - Testrapport
    - Knikker herkenningsalgoritme
- **Afronding**
    - Afstudeerverslag

5. Globale Planning

In Tabel 26 is een globale planning opgesteld. Hierbij is ook de lengte van elk increment gepland. Het begin van is op 04-02-2019 en het einde van de afronding is op 31-05-2019.

Weken	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*22*
Fasen																
Oriëntatiefase																
Increment 1																
Increment 2																
Increment 3																
Afronding																

TABEL 26 – GLOBALE PLANNING

6. Kwaliteit

In dit hoofdstuk worden de grenzen, randvoorwaarden en kwaliteitszorg besproken. Deze zorgen ervoor dat de afstudeeropdracht soepel verloopt en de kwaliteit van het eindproduct wordt gewaarborgd.

6.1. Grenzen

- Het project wordt uitgevoerd in werkweken van 40 uur.
- Het project duurt 16 weken.
- Het project wordt gerealiseerd op een Bachmann PLC
- De knikkers zijn niet transparant.
- De knikkers hebben een hoog contrast met de kleur van de knikkerbaan.
- De knikkerbaan is lineair.
- De camera kan in kleur fotograferen of filmen.

6.2. Randvoorwaarden

- Alle benodigde hardware werkt zonder defecten.
- Er worden wekelijks afspraken gemaakt met de bedrijfsmentor om de voortgang en kwaliteit te bespreken.

6.3. Kwaliteitszorg

Om de kwaliteit van het project te waarborgen, wordt er gebruik gemaakt van:

- ALTEN's Redmine omgeving voor versiebeheer.
- Object georiënteerde programmeerparadigma's.
- Object georiënteerd ontwerpen.

7. Risicoanalyse

Nummer	1.
Risico	Vertraging van afstudeeropdracht, door ziekte.
Omschrijving	De afstudeerder loopt vertraging op, doordat hij voor een langere tijd niet op kantoor kan werken door ziekte.
Kans	Klein
Impact	**Groot:** De afstudeeropdracht kan niet op tijd afgerond worden, waardoor deze niet wordt gehaald.
Actie (preventief)	Alle code en documentatie regelmatig online back-uppen, zodat er mogelijk thuis gewerkt kan worden. Waardoor vertraging beperkt wordt.
Actie (correctief)	Overleggen met bedrijfsmentor en 2e examinator over mogelijkheden om achterstand in te halen en eventueel kijken naar verlenging van de afstudeeropdracht.

Nummer	2.
Risico	Te weinig begeleiding door uitval bedrijfsbegeleider.
Omschrijving	Doordat de begeleider uitvalt krijgt de afstudeerder afdoende begeleiding tijdens het afstuderen.
Kans	Middel
Impact	**Klein:** De afstudeerder krijgt minder begeleiding, waardoor het afstudeerproces moeizamer verloopt. Dit hoeft niet te leiden tot het falen van de afstudeeropdracht.
Actie (preventief)	Wekelijkse afspraken maken met begeleider, waardoor er altijd een moment is voor wekelijkse begeleiding.
Actie (correctief)	In overleg met ALTEN Nederland B.V. zoeken naar een nieuwe bedrijfsmentor. Dit ook doorgeven aan de Haagse Hogeschool.

Nummer	3.
Risico	Knikkers zijn niet te herkennen.
Omschrijving	Door reflectie van licht op de knikker en de transparantie van knikkers, zijn deze niet te detecteren door middel van computer vision.
Kans	Klein
Impact	**Groot:** De knikker kan niet herkend worden waardoor het doel van het project niet gehaald kan worden.
Actie (preventief)	Het uitsluiten van knikkers die transparant zijn of licht reflecteren.
Actie (correctief)	De knikkers afschermen van directe lichtbronnen en uitsluitend indirect licht gebruiken voor de belichting.

Nummer	4.
Risico	Bachmann PLC niet krachtig genoeg voor het uitvoeren van het vision-algoritme.
Omschrijving	Het kan dat de Bachmann PLC niet voldoende rekenkracht en opslag heeft om het vision-algoritme uit te voeren.
Kans	Groot
Impact	**Groot:** Doordat het vision-algoritme niet op de PLC uitgevoerd kan worden, kan het originele doel van de opstelling niet behaald worden.
Actie (preventief)	Onderzoek doen naar de specificaties van de Bachmann PLC en hier rekening mee houden tijdens het ontwerpen en maken van het vision-algoritme. Bijvoorbeeld door het zo min mogelijk gebruiken van externe libraries.
Actie (correctief)	Het vision-algoritme uitvoeren op een computer met meer rekenkracht en geheugen. Hierdoor kan de werking van het algoritme aangetoond worden.

Nummer	5.
Risico	Camera of Bachmann PLC gaan kapot
Omschrijving	Tijdens het project gaat de hardware kapot, waardoor er niet verder gewerkt kan worden.
Kans	Klein
Impact	**Groot:** Als de hardware kapot gaat, dan kan er vertraging opgelopen worden, omdat er niet kan worden gewerkt aan de afstudeeropdracht.
Actie (preventief)	Opnemen in de randvoorwaarde dat alle hardware het naar behoren werkt.
Actie (correctief)	De hardware vervangen.

Nummer	6.
Risico	De opdracht wordt buiten verwachte planning uitgevoerd door foute inschatting niveau opdracht.
Omschrijving	Het kan blijken dat de opdracht te makkelijk of te moeilijk is, waardoor de opdracht niet volgens te planning wordt uitgevoerd.
Kans	Klein
Impact	**Middel:** Als de opdracht niet volgens de tijdsplanning wordt uitgevoerd, dan kan het ervoor zorgen dat de procesvoering van het project niet meer klopt.
Actie (preventief)	Tijdens het project reflecteren op de voortgang en de planning hierop bijwerken.
Actie (correctief)	De opdracht binnen de scope aanpassen zodat deze wel binnen de tijdsplanning af komt.

Nummer	7.
Risico	Het is niet mogelijk om een camera aan te sluiten om een Bachmann PLC.
Omschrijving	Door gebrek aan driver kan het mogelijk zijn dat de Bachmann PLC geen beelden kan krijgen van een camera.
Kans	Middel
Impact	**Groot:** Als er geen camera direct op de PLC aangesloten kan worden, dan is er geen beeld om mee te werken en kan de afstudeeropdracht niet uitgevoerd worden.
Actie (preventief)	Onderzoek doen naar camera's die direct bestuurd kunnen worden door C++ of waarvan een C++ library beschikbaar is dat gebruikt kan worden.
Actie (correctief)	Een extern apparaat toevoegen dat de beelden opneemt en stuurt naar de PLC.

Bijlage B: Analyserapport

Het herkennen en locatie voorspellen van knikkers op basis van kleur met een camera en PLC bij Alten Nederland.

Versie: 1.0
Status: Definitief
Datum: 31 mei 2019
Auteur: Hanno van Megchelen

Inhoudsopgave

1. Inleiding

In opdracht van ALTEN Nederland wordt een afstudeerproject uitgevoerd, waarbij er de positie van een knikker moest worden voorspeld met een behulp van een camera en een Bachmann PLC. Voor de uitvoering van het project was het nodig om ontbrekende kennis op te doen door middel van een analyse. Ook moesten de eisen van het eindproduct worden opgesteld. Dit document is ter ondersteuning van het afstudeerdossier.

In dit rapport is er een analyse gemaakt over de Bachmann PLC, de camera en beeldbewerkingsmethodes. Daarna is het probleemdomein opgesteld op basis van de geanalyseerde onderwerpen. Met het probleemdomein en de klantwensen is zijn de eisen opgesteld.

2. Specificaties Bachmann PLC

Voor het uitvoeren van de afstudeeropdracht wordt er gebruik gemaakt van een Bachmann MC205 Processor Module met een NT255 voeding. In Figuur 26 zijn deze te zien.

FIGUUR 26 - NT255 VOEDING EN MC205 PROCESSOR MODULE

Deze componenten zijn ontwikkeld door Bachmann electronic GmbH en worden veel gebruikt in de energiesector.

2.1 MC205 processor module

De MC025 [1] module is het besturende element in de opstelling. Hierop wordt de gemaakte software uitgevoerd. Op deze module draait een VxWorks besturingssysteem met Bachmann uitbereidingen. Dit houdt in dat Bachmann electronic GmbH zijn eigen besturingselementen heeft toegevoegd aan het VxWorks besturingssysteem, meer over VxWorks is te vinden in *hoofdstuk 2.3*. Doordat VxWorks als basis wordt gebruikt, kan de PLC in C/C++ geprogrammeerd worden. Hierdoor kunnen ingewikkelde algoritmes, zoals een vision-algoritme, makkelijker geïmplementeerd worden, dan bijvoorbeeld op een Siemens PLC. De enige limitatie is de rekencapaciteit van de PLC zelf. In Tabel 27 zijn de specificaties van de MC205 te vinden.

Onderdeel	**Specificatie**
CPU	600 MHz ATOM E620
RAM	1 GB DRAM DDR2
nvRAM-0	512 kB
Intern geheugen	64 MB, 40 MB vrij voor applicatieprogramma
Extern geheugen	CFast flash kaart tot 8 GB
Interfaces	USB 2.0, 2x Seriële poort (COM 1 - 2) en 2x ethernet
Extra functies	
Watchdog	
Synchroniserende puls, ook voor I/O bus, veldbussen SERCOS & CAN, ethernet	
Real-time klok met batterij	
VxWorks besturingssysteem met Bachmann systeem uitbereiding	
3 status LED’s	

TABEL 27 - MC205 SPECIFICATIES

2.2 NT255 voeding

De NT255 [2] is de voeding dat stroom levert aan de andere stroom modules. Hierin zitten beschermingen voor te hoge stroom en omgekeerde polariteit. De voeding werkt op 18 tot 34 Volt DC en heeft een vermogen nodig van max. 68 watt. Daarnaast wordt er gecontroleerd of alle stromen correct zijn. Mocht dit niet zo zijn wordt er een error melding gegenereerd. In Tabel 28 is een lijst met specificaties opgesteld.

Invoer	
Voltage gebied	18 tot 34 Volt
Ingangsvoltage, piekwaarde	+40V met *t < 1 s/min*
Vermogen	Max. 68 Watt
Omgekeerde polariteit bescherming	Elektronisch
Ingangsvoltage controle	Ja, voor stroom falen berichten
Uitvoer	
Uitgangsvermogen	45 Watt
Uitgangsspanning, -stroom	+5 V/ 6 A +15 V/ 0.5 A -15V/ 0.5 A
Stroomstoringomzeiling	18 ms, stroomstoringbericht na 3 ms

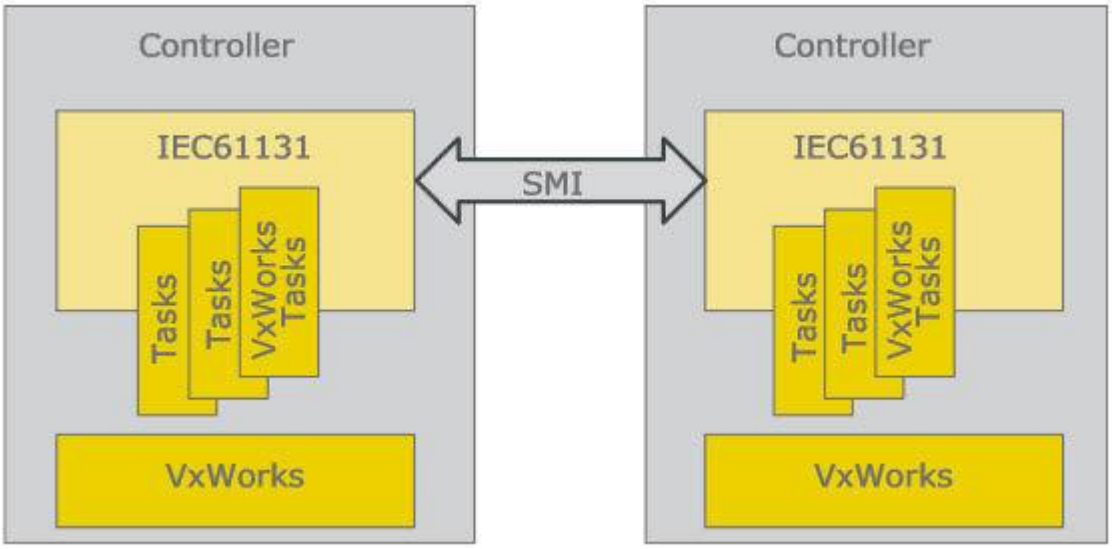
TABEL 28 - NT255 SPECIFICATIES

2.3 VxWorks

VxWorks [3] is een real-time besturingssysteem (RTOS) dat ontwikkeld is door Wind River Systems. Dit besturingssysteem is gemaakt voor embedded systemen die real-time functionaliteit nodig hebben. Vaak gaat dit gepaard met multi-threading en -processing. Hierdoor kunnen meerdere applicaties naast elkaar draaien. Deze kunnen worden gestart bij het opstarten van het besturingssysteem, in een andere applicatie of handmatig. Dit kan zo nodig gebruikt worden om meerdere applicaties te starten op de PLC.

VxWorks heeft zijn eigen scheduler voor het bepalen welke thread (taak) actief is. Voor het regelen van real-time afhankelijke taken wordt gebruik gemaakt van prioriteit. Dit wil zeggen dat de taak met de hoogste prioriteit eerst wordt afgehandeld. Ook wanneer er een taak met een lage prioriteit bezig en er een nieuwe taak bij komt met een hogere prioriteit. De taak met de lage prioriteit wordt dan afgekapt.
Als er taken zijn met een gelijke prioriteit, dan wordt er gebruikt gemaakt van round-robin. Dit houdt in dat er wordt afgewisseld tussen de taken na een vaste tijdseenheid. Dit wordt ook gebruikt bij het regelen van niet real-time afhankelijke taken. Hierbij wordt er echter wel een versie gebruikt dat dynamisch een prioriteit toekend aan taken. Deze taken krijgen meer tijd om uitgevoerd te worden, maar er wordt altijd geschakeld tussen taken.

Bij de PLC wordt er geschakeld tussen taken van VxWorks zelf en taken die worden gestart door de PLC. In Figuur 27 is een schematische afbeelding te zien hoe VxWorks is geïntegreerd op de Bachmann PLC. De IEC61131 is de programmeerbare controller op de PLC. Op het laagste level draait VxWorks. De IEC61131 maakt taken aan en stuurt deze naar VxWorks om uitgevoerd te worden. Welke ook de volgorde van uitvoeren bepaald.



FIGUUR 27 - SCHEMATISCHE AFBEELDING VXWORKS OP EEN BACHMANN PLC [4]

3. Camera compatibel met Bachmann PLC

Het vision-algoritme moet uitgevoerd worden op de PLC. Hiervoor moeten de beelden van de camera beschikbaar zijn in de PLC. Dat kan door rechtstreeks een camera aan te sluiten op de PLC en de afbeeldingen uit te lezen. Ook kan het door de beelden te verzenden vanuit een extern systeem. Om te bepalen welke camera nuttig is voor de opstelling, moet er gekeken worden wat de aansluitingsmogelijkheden er zijn. Binnen de aansluitingsmogelijkheden wordt er gekeken naar een aantal eigenschappen om kijken welke het beste past bij de afstudeeropdracht. De eigenschappen die belangrijk zijn, zijn als volgt:

- Snelheid dataoverdracht
- Cameradrivers voor VxWorks
- Extra benodigde hardware
- Kosten

Bij het onderzoeken naar aansluitingsmogelijkheden wordt er alleen gekeken naar interfaces die de PLC ondersteund. Uit tabel 1 volgen de volgende interface technieken:

- USB 2.0
- Ethernet
- Seriële poort

3.1 USB 2.0 camera

De universal serial bus (USB) is een industriële standaard voor het aansluiten van randapparatuur. In deze standaard zijn kabels, communicatie en stroomvoorziening gespecificeerd. USB 1.0 is de opvolger van de seriële poort. USB 2.0 is de tweede generatie van de standaard. Het is sneller en heeft meer connectie mogelijkheden. USB wordt bijna door alle apparaten ondersteund. Daardoor is het een populaire methode om camera's aan te sluiten. Echter zijn er vaak driver nodig om de camera's aan te kunnen sturen. In de onderstaande Tabel 29 zijn de eigenschappen te vinden van een USB-camera.

Snelheid dataoverdracht	60MB/s
VxWorks drivers	Weinig beschikbaar
Extra hardware	Geen
Kosten	Laag

TABEL 29 - EIGENSCHAPPEN USB-CAMERA

Als er gekeken wordt naar de snelheid dataoverdracht, extra hardware en kosten, dan zou de USB-camera een goede optie zijn. 60MB/s is meer dan genoeg voor bijvoorbeeld een 8 bit 640x480 JPEG-beeld te verzenden in korte tijd.

Bij de bovengenoemde JPEG bestaat elke pixel uit 3 bytes. Een afbeelding heeft 640*480*3 = 921.600 bytes. Met een dataoverdracht van 60MB/s is de tijd dat het kost om een afbeelding te versturen, 901.600 /60.000.000 = 0.015 seconde -> 15 milliseconde. Met de aanname dat er geen compressie plaatsvindt.

Naast de snelheid zijn USB-camera's aantrekkelijk, omdat ze vaak relatief goedkoop zijn. Dit komt doordat USB veel gebruikt wordt en de technologie niet duur is. Hierdoor is het minder werk om USB te integreren op camera's en worden deze goedkoper. Daarnaast hoeft er geen extra hardware aangeschaft te worden.

Het grootste probleem bij USB-camera's is de drivercompatibiliteit met VxWorks voor versie 7. Elke cameraproducent heeft zijn eigen drivers om via USB de beelden op te vragen. Deze worden vaak alleen ondersteunt op veel gebruikte besturingssystemen zoals Windows, Linux en macOS. Aangezien VxWorks

vaak wordt gebruikt voor embedded systemen met specifieke functionaliteit, worden er hier minder drivers voor gemaakt. Om een USB-camera rechtstreeks aan te sluiten de PLC, moet er een driver geschreven worden. [5]

Sinds VxWorks 7 [6] ondersteund het, de USB video driver. Dit is een gestandaardiseerde driver om videocamera's aan te sluiten door middel van USB [7]. Ondanks dat deze VxWorks versie beschikbaar op de PLC, kan de USB-camera moeilijk gebruikt worden, omdat de Bachmann uitbereiding van VxWorks deze driver niet herkent.

3.2 Ethernet camera

Camera's kunnen ook aangesloten worden met ethernet. De meest gebruikte vorm hiervan zijn IP-camera's. Daarnaast zijn er camera's die communiceren via het GIGe video-protocol dit wordt veel gebruikt in de industrie. Naar GIGe video wordt niet verder gekeken, omdat dit protocol vooral bedoeld is voor het verzenden over een 1 Gb/s verbinding. Deze is niet beschikbaar op de PLC. Ook zijn er drivers nodig om via dit protocol een afbeelding te ontvangen. Welke niet aanwezig zijn om de Bachmann PLC.

IP-camera's worden gebruikt om over het internet een videostream te verzenden. Er is hiervoor geen intern opname apparaat nodig. Om een afbeelding op te vragen kan HTTP gebruikt worden. Hiervoor zijn er geen camera specifieke drivers nodig. De camera moet echter altijd in een LAN (local area network) verbonden zijn met de PLC. Meer eigenschappen over de verbinding tussen een IP-camera en een PLC worden in Tabel 30 gegeven.

Snelheid dataoverdracht	12,5 MB/s
VxWorks drivers	Geen drivers nodig
Extra hardware	Mogelijk Power over Ethernet switch
Kosten	Hoog

TABEL 30 - EIGENSCHAPPEN IP-CAMERA

Als de IP-camera wordt vergeleken met de USB-camera's dan is al snel te zien dat de dataoverdracht snelheid een stuk kleiner is. Met dezelfde berekening die is uitgevoerd bij de USB-camera duurt het 74ms om dezelfde 640x480 JPEG-afbeelding verzenden. Dit is aanzienlijk langzamer. Ook zijn deze camera's vaak duurder, omdat de productie van deze camera's duurder zijn. Ook kunnen er extra kosten bij komen als power over ethernet gebruikt wordt. Dit kan gewenst zijn als de stroomkabel niet lang genoeg is of dat men maar één kabel wil gebruiken voor data en stroom.

Ondanks deze nadelen is het grootste voordeel heel belangrijk bij de afstudeeropdracht. Er zijn namelijk geen drivers nodig om de camera aan te sturen. Elk apparaat dat een HTTP-verzoek kan genereren en de response kan verwerken, kan ook afbeeldingen ontvangen van een IP-camera. Dit gewenst bij deze PLC, omdat er weinig drivers beschikbaar zijn.

3.3 Seriële poort camera

Communicatie via een seriële poort [8] is de voorganger van USB. Hierdoor wordt deze interface bijna niet meer gebruikt op commerciële computers. Waar de seriële poort nog wel te vinden is, is in de industrie waar de technologie vaak ouder is. Ook wordt deze interface gebruikt voor routers en switches voor de configuratie hiervan. De camera's [9] die met deze interface communiceren, zijn heel simpel en worden vooral gebruikt in kleine embedded systemen. In Tabel 31 zijn de eigenschappen van een Seriële poort camera gegeven.

Snelheid dataoverdracht	115,2 Kbit/s -> 14,4 KB/s
VxWorks drivers	Geen drivers nodig
Extra hardware	Convertor voor goede aansluiting
Kosten	Laag

TABEL 31 - EIGENSCHAPPEN SERIËLE POORT CAMERA

Het eerste dat opvalt bij dit soort camera's is dat de snelheid van de dataoverdracht heel laag is. Met de berekening van de vorige camera's duurt het 64 seconde om dezelfde afbeelding te verzenden. In de tijd dat een foto is binnengehaald is de situatie al zodanig veranderd dat deze niet meer relevant is. Ook kunnen deze camera's niet direct aangesloten worden op de seriële poort. Er moet hiervoor een convertor gebruikt worden. De enige voordelen van deze camera's zijn dat er geen drivers nodig zijn en de prijs laag is.

4. Beeldbewerkingsmethodes

Als er door middel van een camera een afbeelding gemaakt wordt en verzonden naar de PLC, dan moet er daarna gekeken worden of er een knikker te vinden is en waar deze zich bevindt. Hiervoor moeten beeldwerkingsmethodes uitgevoerd worden. In dit hoofdstuk worden beeldbewerkingsmethodes besproken die mogelijk gebruikt kunnen worden bij het vinden van de knikker. Hierbij gaat het over het gebruik en werking van de methodes en niet over de technische implementatie. Daarnaast is de aanname gemaakt dat de te bewerken afbeeldingen een 640*480 resolutie hebben met een 8-bit waarde representatie. Deze aanname is gemaakt, omdat het uitleggen van beeldbewerkingsmethodes hiermee duidelijker is.

4.1 Grayscale-conversie

Grayscale-conversie is het omzetten van een kleurenafbeelding naar een zwartwitafbeelding. Dit wordt in beeldbewerking vaak gebruikt, omdat het bewerken van een zwartwitafbeelding makkelijker en sneller is dan het bewerken van een kleurenafbeelding. Dit komt doordat een pixel in een zwartwitafbeelding bestaat uit een enkele waarde en een kleurenpixel vaak uit drie waardes, namelijk rood, groen en blauw (RGB). Bij een 640*480 afbeelding bestaat de zwartwitafbeeldingen uit 307.200 bytes tegen 921.600 bytes van een kleurenafbeelding. Hierdoor is het minimaal drie keer sneller om een zwartwitbeeld te bewerken. Ook het bewerken van een zwartwitafbeelding gaat beter door de opbouw van de pixels. Bij een kleurenafbeelding moet er per pixel gekeken worden naar drie bytes. Elke heeft een drie waarde waarvan het nut moet worden bepaald of welke vergeleken moet worden met de overeenkomende waarde van een andere pixel. Hierdoor worden beeldbewerkingsmethodes complexer dan nodig.

Elke waarde in een kleurenafbeelding is bewerkt met een zogenoemde gammawaarde, zodat de afbeelding realistisch wordt weergegeven op het beeldscherm. Dit heet gammacompressie. Om een kleurenafbeelding te converteren zijn de volgende stappen nodig:

4. *Het verwijderen van de gammacompressie*
5. *Het berekenen van de grijswaarde van een pixel*
6. *(Optioneel) Het uitvoeren van gammacompressie op de grijswaarde*

Voor het berekenen van de grijswaarde van een kleurenpixel, wordt elke kleur van de pixel bij elkaar opgeteld met een vooraf bepaalde weging per kleur. Bijvoorbeeld als er gekeken wordt naar sRGB [10] een standaard RGB-formaat (IEC 61966-2-1:1999). Dan is de conversie:

$$Y = 0.2126R + 0.7152G + 0.0722B$$

In de bovenstaande formule is Y is de grijswaarde. De wegingen zijn gedefinieerd in de het sRGB formaat. Nadat de grijswaarde is berekend kan er optioneel nog gammacompressie uitgevoerd worden op deze waarde. Hierdoor wordt het beeld realistisch weergegeven op het beeldscherm. Dit is niet altijd nodig bij beeldbewerking zolang de belangrijke voorwerpen zichtbaar zijn in de afbeelding.

4.1.1 Thresholding

In het verlengde van deze methode bestaat thresholding. Dit is een techniek dat op basis van een grenswaarde pixels zwart (0) of wit (255) maakt. Hierdoor krijg je een binaire afbeelding.

4.2 JPG-compressie en decompressie

7. Kleurenformaat van RGB veranderen naar YCbCr
8. Sub-sampling van Cb en Cr delen
9. Omzetten naar frequentiedomein door middel van discrete cosinus transformatie
10. Verwijderen van hoge frequenties
11. Reordering for better compression
12. Removal of zeros series
13. Lossless entropy coding to remove some more extra data
14. Final packing

4.3 Filtering

Het filteren [11] van een beeld is het bewerken of verbeteren van een beeld. Voorbeelden van bewerkingsmethodes zijn “smoothing”, “edge-detection” en “template matching”. Filteren wordt gedaan met een filter of kernel. Dit is meestal een 2-dimentionale matrix. Door middel van deze matrix wordt elke pixel bewerkt aan de hand van zijn omliggende pixels. Dit zijn neihgborhood calculations. De bewerkte pixels worden niet in de originele afbeelding overschreven, maar in een nieuwe afbeelding gezet.

4.3.1 Smoothing

Smoothing is een basis filter methode, welke vaak als eerste op een zwartwitafbeelding wordt toegepast. Deze methode kijkt naar de pixels om de te bewerken pixel heen en berekend de gemiddelde waarde van deze pixels en de te bewerken pixel. Dit gemiddelde wordt de nieuwe pixelwaarde. Door deze methode wordt de afbeelding waziger en verliest deze detail. Hierdoor kan bijvoorbeeld ruis verwijderd worden. In Figuur 28 is het resultaat van zo’n filter gegeven.

FIGUUR 28 - VOOR EN NA SMOOTHING [12]

Zoals hierboven te zien is, is het resultaat na smoothing een stuk waziger dan ervoor. Hierbij zijn afwijkende waardes uit de afbeelding gefilterd. Waardoor vervolg bewerkingsmethodes hier niet door worden beïnvloed. Zo kan thresholding worden toegepast. Dit is een techniek dat op basis van een grenswaarde pixels zwart of wit maakt. Hierdoor krijg je een binaire afbeelding.

4.3.2 Edge-detection

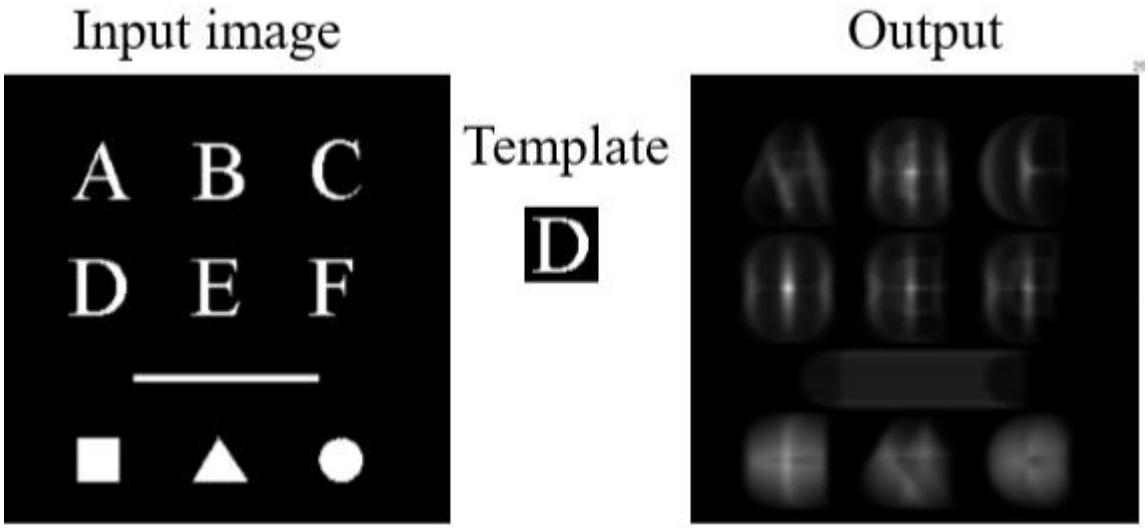
Edge-detection is een methode dat lijnen zet op plekken waar aanliggende pixels veel van waarde verschillen. Als de waarde niet verschilt wordt die plek zwart. Hierdoor kunnen lijnen om objecten getekend worden. Het filter dat hiervoor gebruikt wordt heet een hoogdoorlaat-filter (HD-filter). In Figuur 29 is het resultaat van edge-detection gegeven.

FIGUUR 29 - VOOR EN NA EDGE DETECTION [13]

Zoals hierboven te zien is, is het verschil op de grond ook groot genoeg ook als rand gemeten te worden. Als dit niet gewenst is, kan er eerst smoothing toegepast worden om daarna edge-detection uit te voeren. Het filter dat is toegepast bij het bovenstaande filter heet een HD-sobelfilter. Dit filter bestaat uit een filter dat twee keer over de afbeelding heen gaat. Één keer normaal en de tweede keer een kwartslag gedraaid. De eerste keer wordt er naar het verschil in waarde in de verticale richting gekeken en daarna in het verschil in de horizontale richting. Hierdoor wordt het verschil in alle richtingen meegenomen bij het bepalen van de randen.

4.3.3 Template matching

Template matching is de meest specifieke methode die tot nu toe behandeld is. Met deze methode is je filter een kleine afbeelding en vergelijk je deze met delen van je originele afbeelding. Als de pixels in niet overeenkomen wordt er die plek zwart. Echter hoe meer pixels wel overeenkomen hoe witter de plek wordt. Hierdoor kunnen hele specifieke vormen herkend worden in een afbeelding. In Figuur 30 is het resultaat van template matching gegeven.

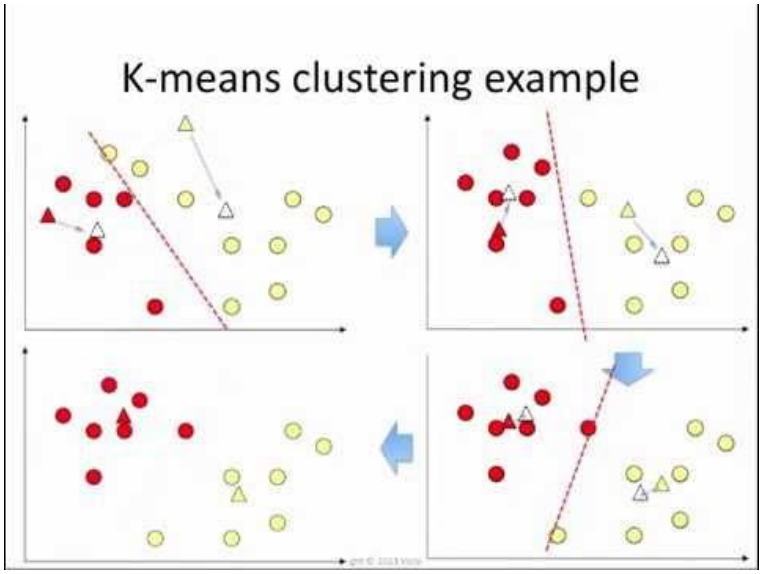


FIGUUR 30 - VOOR EN NA TEMPLATE MATCHING [13]

Zoals in de bovenstaande afbeelding te zien is, zijn er op meerdere plekken overeenkomsten gevonden. De twee plekken waar de afbeelding en de template het meest overeenkwamen waren bij de B en D. Toch is het duidelijk in de output waar de D stond in de input. Door dit te combineren met andere methodes, kan bijvoorbeeld de D geïsoleerd worden van de rest en daarna donkerder worden gemaakt.

4.4 K-means

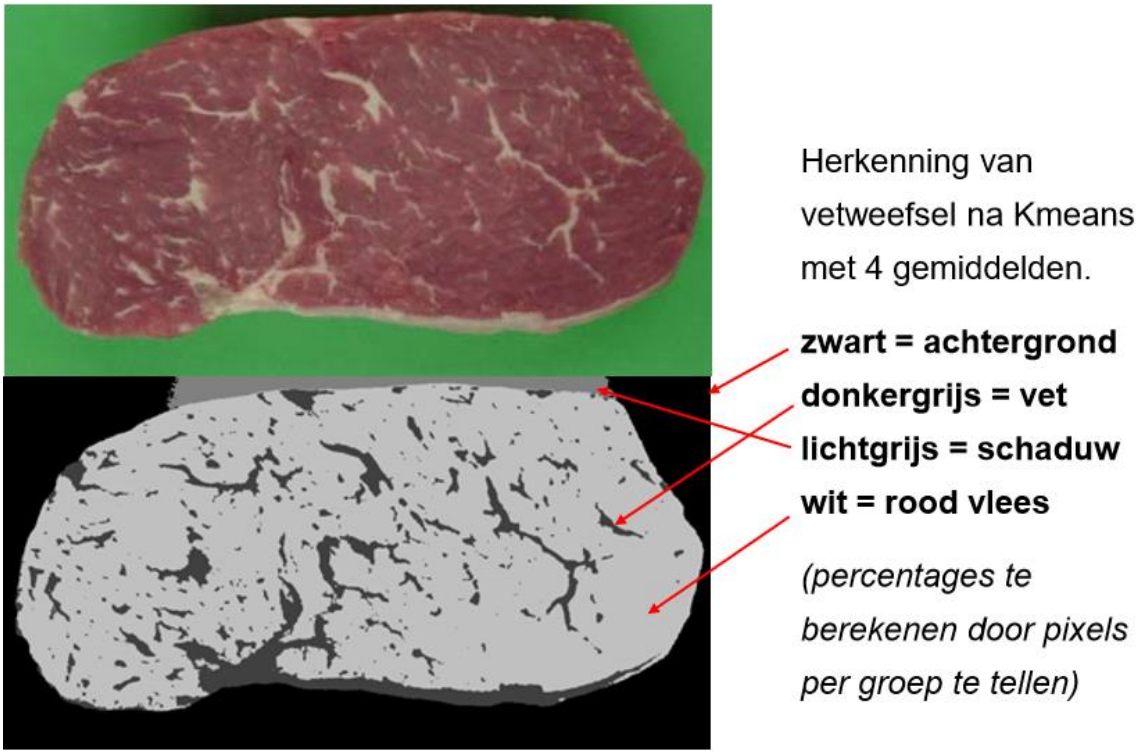
K-means is een manier om data te groeperen in clusters. De waarde K in de naam staat gelijk aan de variabele K die wordt gebruikt in het algoritme. K in hierbij het aantal groepen dat die er gemaakt moeten worden. Voor elke te vinden groep wordt er een variabele aangemaakt, bijvoorbeeld ***p***, met een waarde binnen de uiterste waardes van de data. Aan elke ***p*** wordt data toegekend die in de buurt van deze waarde zit. Van deze data wordt het gemiddelde berekend en de uitkomst wordt toegekend ***p***. Hierna wordt opnieuw data toegekend die in de buurt van ***p*** zit, opnieuw het gemiddelde berekend en het gemiddelde aan toegekend aan ***p***. Dit herhaalt zich tot ***p*** niet meer verandert. Bij een afbeelding wordt daarna per groep de bijbehorende waarde ***p*** aan de data toegekend. In Figuur 31 is een afbeelding gegeven met de werking van het algoritme.



FIGUUR 31 - WERKING K-MEANS

In het bovenstaande voorbeeld zijn de driehoeken de waarde ***p*** en is de waarde van K, twee. Dit algoritme wordt niet alleen gebruikt voor beeldbewerking, maar ook in data-analyse van big-data. In het geval van beeldbewerking gaat het om het maken van kleurgroepen.

Het resultaat van K-means om een afbeelding en in Figuur 32 te vinden. Waarbij de kleuren worden opgedeeld in 4 groepen.



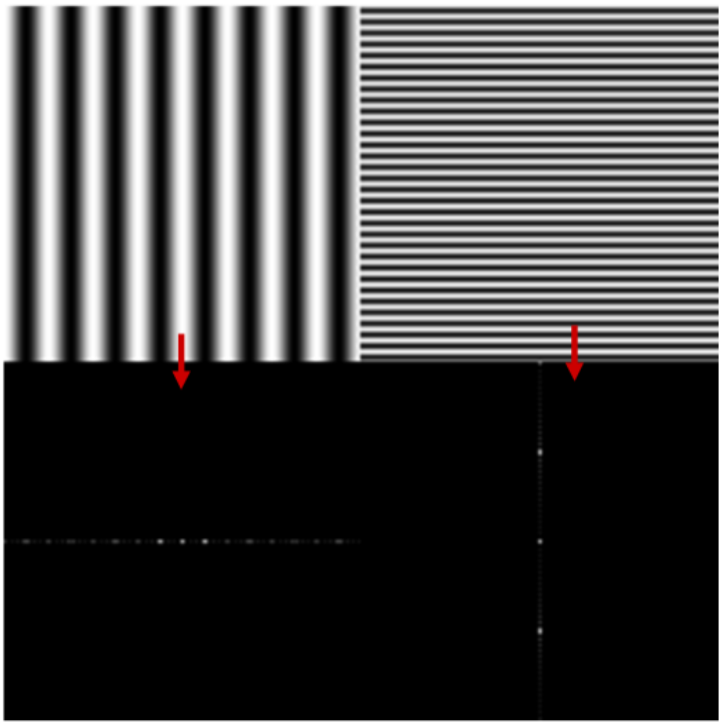
FIGUUR 32 - RESULTAAT K-MEANS MET K = 4 [13]

Zoals hierboven te zien is, is er een duidelijke scheiding gemaakt tussen de verschillende kleuren. Ook is het verschil in kleur van bijvoorbeeld het vlees opgeheven en is elke groep een eigen unanime kleur. Dit maakt vervolg bewerkingen makkelijker.

Dit algoritme kan gebruikt worden als er een duidelijk verschil in kleur is tussen het te meten object en de rest. Als dit verschil te klein is, dan kan K-means ervoor zorgen dat het object en de achtergrond in dezelfde groep wordt ingedeeld, waardoor het object verdwijnt.

4.5 Fouriertransformatie

Een afbeelding is een optelling van golven. Deze golven kunnen gesplitst worden door een fouriertransformatie uit te voeren over de afbeelding. Een afbeelding met gesplitste golven laat het frequentiedomein zien. Er zijn veel verschillende fouriertransformaties, maar bij afbeeldingen wordt vaak de discrete fouriertransformatie (DFT) gebruikt. Het resultaat van deze DFT is een afbeelding waarin elke pixel een frequentie representeert. De x-as representeert een horizontale frequentie en de y-as een verticale frequentie. De intensiteit van een pixel geeft aan hoe veel een bepaalde frequentie voorkomt in de originele afbeelding. In afbeeldingen zijn patronen vaak duidelijk te zien in het frequentiedomein. Een voorbeeld hiervan is in Figuur 33 gegeven.



FIGUUR 33 - VOORBEELDEN DFT [13]

Hierboven is duidelijk te zien dat een horizontaal patroon, dus de lijnen lopen verticaal, op de x-as strippen zet op bij de voorkomende frequenties. Hetzelfde is te zien voor een verticaal patroon en de y-as.

Naast het herkennen van patronen kunnen ook beeldbewerking methodes als ‘smoothing’ en ‘edge-detection’ gerealiseerd worden door het frequentiedomein te bewerken. Door de hoge frequenties weg te halen uit het frequentiedomein wordt het detail uit de afbeelding gehaald (smoothing). Door de lage frequenties uit de afbeelding te halen wordt de invulling weggehaald en blijven alleen de randen over (edge-detection). Deze manier van filteren is sneller dan de in hoofdstuk 5.2 behandelde filter methode. Echter is deze moeilijker te implementeren, doordat het algoritme ingewikkelder is.

5. Eisen van het project

In dit hoofdstuk worden de eisen van het project en het probleemdomein opgesteld aan de hand van de analyse

5.1 Systeemeisen

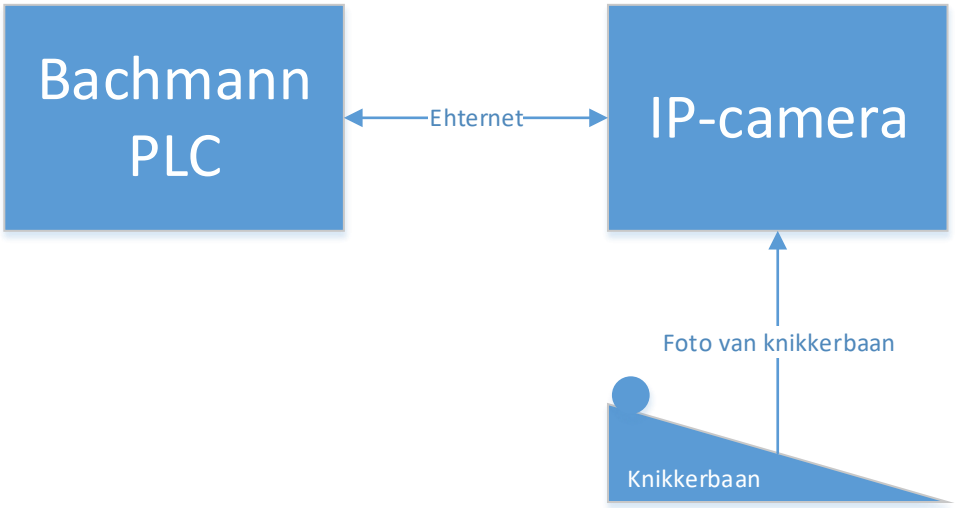
De systeemeisen zijn opgesteld aan de hand van de analyse en van de grenzen (bijlage 1) uit het Plan van Aanplak. Deze zijn in tabel 6 weergegeven.

ID	Eis	Herkomst eis
SE01	De camera moet kunnen communiceren met de PLC.	Zonder communicatie kan de afbeelding niet verkregen worden, dus kan er geen knikker herkend worden.
SE02	De camera moet ondersteund worden door de PLC.	Volgt uit bijlage B hoofdstuk 3. De PLC heeft beperkte ondersteuning voor camera's.
SE03	De camera kan een kleurenafbeelding maken.	Volgt uit grenzen in plan van aanpak.
SE04	De knikkerbaan mag niet dezelfde kleur zijn als de knikkers.	Volgt uit bijlage B hoofdstuk 4. Als er weinig kleurverschil is, dan kan de knikker niet herkend worden.
SE05	Knikkerbaan moet lineair zijn.	Volgt uit grenzen in plan van aanpak.
SE06	De knikkers mogen niet transparant zijn.	Volgt uit grenzen in plan van aanpak.
SE07	De voorspelling mag er maximaal één knikkergrootte naast zitten.	Gesprekken met de opdrachtgever.

TABEL 32 - SYSTEEMEISEN

5.2 Probleemruimte

Uit de systeemeisen volgt de probleemruimte. Hierbij zijn alle onderdelen waarmee het systeem gemaakt moet worden opgesteld. De probleemruimte bestaat uit een Bachmann PLC, een IP-camera en een knikkerbaan. De PLC is door middel van een ethernetkabel verbonden met de camera. Welke een foto maakt van de knikkerbaan en deze naar de PLC stuurt. Een overzicht hiervan is opgesteld in Figuur 34.



FIGUUR 34 - PROBLEEMRUIMTE

Specificaties van de PLC en IP-camera worden respectievelijk in hoofdstuk 2 en 7.3 besproken.

5.2.1 Specificaties IP-camera

Bij het opstellen van de probleemruimte is er gekozen voor een IP-camera. Deze past het best bij de systeemeisen van het project. Er is namelijk geen driver nodig om de camera op de PLC aan te sluiten. De IP-camera communiceert met de PLC via HTTP. Hierbij is de camera de server en de PLC de client. De PLC stuurt een HTTP-get verzoek en de camera stuurt als reactie de afbeelding. Deze afbeelding wordt verzonden als JPG-formaat. Hierdoor zal de verzendtijd minder zijn dan in hoofdstuk 3.2 besproken is. Ook kan deze camera kleurenafbeeldingen verzenden.

5.3 Functionele eisen

De eisen zijn opgesteld aan de hand van IEEE … standaard en geprioriteerd op basis van de MoSCoW methode. Uit het probleemdomein en analyse zijn de eisen naar voren gekomen. De functionele eisen zijn in Tabel 33 gegeven.

ID	Eis	Prioriteit	Commentaar
FE01	Het vision-algoritme kan een non-transparante knikker herkennen.	M	Transparante knikkers zijn uitgesloten.
FE02	Het vision-algoritme kan de positie van de knikker bepalen.	M	
FE03	Het vision-algoritme kan de plek van een knikker op een knikkerbaan kan voorspellen.	M	
FE04	Het vision-algoritme kan op de Bachmann PLC worden uitgevoerd in de programmacyclus	M	
FE05	Het vision-algoritme kan de afbeelding van JPG naar een bewerkbaar beeld omzetten.	M	IP-camera verzendt afbeelding in JPG-formaat
FE06	De PLC moet een afbeelding kunnen krijgen van de IP-camera.	M	
FE07	De PLC moet een output genereren waarbij het resultaat van het vision-algoritme getoond wordt.	M	
FE08	Het vision-algoritme kan de kleur van de knikker kan herkennen.	S	
FE09	Het vision-algoritme voorspelt alleen de positie van knikker met de verwachte kleur.	S	
FE10	De PLC moet een cartesisch coördinaat genereren op basis van de voorspelde knikkerpositie in de afbeelding.	C	Voor het interacteren met knikker door extern systeem.
FE11	De knikkervoorspelling moet gecontroleerd kunnen worden door middel van een interface.	C	Voor start/stop en aangeven van systeemvariabelen.

TABEL 33 - FUNCTIONELE EISEN

6. Incrementplanning

In dit hoofdstuk wordt er opgesteld welke eisen uitgevoerd worden per increment. Deze planning kan aangepast worden aan de hand van het resultaat van een increment. De incrementplanning is in Tabel 34 gegeven, waarin de incrementen ook een benaming hebben gekregen.

Increment	Eisen
1: Ophalen Afbeelding	• FE05 • FE06 • SE04 • SE05
2: Positie bepalen knikker	• FE01 • FE02 • FE07
3: Positie voorspellen knikker	• FE03 • FE04 • FE08 • FE09
Extra	• FE10 • FE11

TABEL 34 - INCREMENTPLANNING

In het eerste increment worden ook twee systeemeisen gerealiseerd aangezien de knikkerbaan nog gemaakt moet worden. De eisen die in het extra increment staan zijn eisen die worden uitgevoerd als alle andere eisen voortijdig gerealiseerd zijn. Deze worden buiten beschouwing gelaten als de realisatie verloopt volgens de planning of als de realisatie uitloopt.

7. Bronvermelding

[1] B. e. GmbH, „Processor Modules," Mei 2018. [Online]. Available: https://www.bachmann.info/fileadmin/media/Produkte/Steuerungssystem/Produktblaetter/MC200-series_en.pdf.

[2] B. e. GmbH, „System Modules," Bachmann, mei 2018. [Online]. Available: https://www.bachmann.info/fileadmin/media/Produkte/Steuerungssystem/Produktblaetter/NT255_en.pdf.

[3] W. River, „Application programmer's guide 6.6," Wind River, Alameda, 2007.

[4] B. e. GmbH, „System overview," november 2018. [Online]. Available: https://www.bachmann.info/uploads/tx_sbdownloader/systemoverview_11_2018_en.pdf.

[5] rolfk, „Desktop RT target with USB camera support," 19 november 2015. [Online]. Available: https://forums.ni.com/t5/LabVIEW/Desktop-RT-target-with-USB-camera-support/td-p/3218384.

[6] W. River, „VxWorks 7 Release Notes," Wind River, Alameda, 2018.

[7] Wikipedia, „USB video device class," Wikipedia, 21 september 2018. [Online]. Available: https://en.wikipedia.org/wiki/USB_video_device_class.

[8] Wikipedia, „Serial Port," Wikipedia, 29 November 2018. [Online]. Available: https://en.wikipedia.org/wiki/Serial_port.

[9] Adafruit, „TTL Serial JPEG Camera with NTSC Video," Adafruit, [Online]. Available: https://www.adafruit.com/product/397.

[10] Wikipedia, „SRGB," Wikipedia, 6 februari 2019. [Online]. Available: https://en.wikipedia.org/wiki/SRGB.

[11] Mathlab, „What is image filtering in the spatial domain," Mathlab, [Online]. Available: https://www.mathworks.com/help/images/what-is-image-filtering-in-the-spatial-domain.html.

[12] K. Hong, „MATLAB TUTORIAL : DIGITAL IMAGE PROCESSING 6 - SMOOTHING : LOW PASS FILTER," bogotobogo, 2016. [Online]. Available: https://www.bogotobogo.com/Matlab/Matlab_Tutorial_Digital_Image_Processing_6_Filter_Smoothing_Low_Pass_fspecial_filter2.php.

[13] P. Burghouwt, „Course Documents Image processing & Computer Vision," de Haagse Hogeschool, 2016. [Online]. Available: https://blackboard.hhs.nl/webapps/blackboard/content/listContent.jsp?course_id=_61290_1&content_id=_1945220_1&mode=reset.

Bijlage C: Ontwerprapport

Het herkennen en locatie voorspellen van knikkers op basis van kleur met een camera en PLC bij Alten Nederland.

Versie: 1.0
Status: Definitief
Datum: 31 mei 2019
Auteur: Hanno van Megchelen

Inhoudsopgave

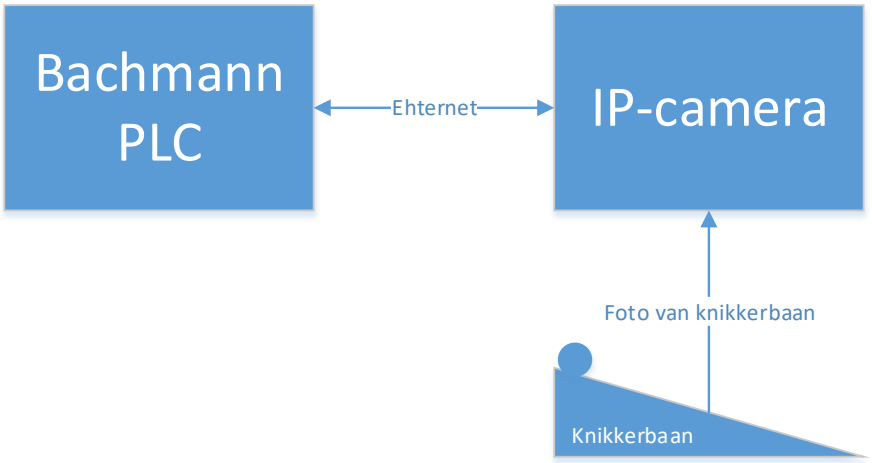
1. Inleiding

In opdracht van ALTEN Nederland wordt een afstudeerproject uitgevoerd, waarbij er de positie van een knikker moest worden voorspeld met een behulp van een camera en een Bachmann PLC. Voor de uitvoering van het project was het nodig om de software te ontwerpen, zodat de software die gemaakt wordt overzichtelijk is en makkelijk te bergrijpen in de toekomst.

In dit rapport zijn er per increment objectgeoriënteerde ontwerpen gemaakt. Hierbij zijn er use-cases met beschrijving opgesteld op basis van de uit te voeren eisen. Daarna is er een klassendiagram gemaakt op basis van de uit de voeren use-cases. Na het opstellen van het klassendiagram is er een sequentiediagram opgesteld waarin de methodes uit het klassendiagram zijn geordend op de aangegeven volgorde van het use-casediagram.

2. Opdracht en eisen

Er moet een vision-algoritme gemaakt worden dat knikkers op een knikkerbaan herkent en zijn positie voorspelt, zodat een robotarm deze op de voorspelde positie kan oppakken. Bij dit project is de scope het maken van het vision-algoritme en niet het oppakken van de knikker. Het vision-algoritme moet op een Bachmann PLC geïmplementeerd worden, zodat het algoritme cyclisch uitgevoerd wordt. De PLC heeft een connectie met een IP-camera om een foto te maken van de knikkerbaan. Een schematische weergave van de opstelling staat om Figuur 35.



FIGUUR 35 - SCHEMATISCHE WEERGAVE OPSTELLING

In het analyserapport (bijlage …) zijn de eisen van het project opgesteld. Deze zijn verdeeld in systeemeisen en functionele eisen. Deze eisen zijn herhaald in Tabel 35 en Tabel 36.

ID	Eis	Herkomst eis
SE01	De camera moet kunnen communiceren met de PLC.	Zonder communicatie kan de afbeelding niet verkregen worden, dus kan er geen knikker herkend worden.
SE02	De camera ondersteund worden door de PLC.	Volgt uit hoofdstuk 3. De PLC heeft beperkte ondersteuning voor camera's.
SE03	De camera kan een kleurenafbeelding maken.	Volgt uit grenzen in plan van aanpak.
SE04	De knikkerbaan mag niet dezelfde kleur zijn als de knikkers.	Volgt uit hoofdstuk 4. Als er weinig kleurverschil is, dan kan de knikker niet herkend worden.
SE05	Knikkerbaan moet lineair zijn.	Volgt uit grenzen in plan van aanpak.
SE06	De knikkers mogen niet transparant zijn.	Volgt uit grenzen in plan van aanpak.

TABEL 35 - SYSTEEMEISEN

ID	Eis	Prioriteit	Commentaar
FE01	Het vision-algoritme kan een non-transparante knikker kan herkennen.	M	Transparante knikkers zijn uitgesloten.
FE02	Het vision-algoritme kan de positie van de knikker bepalen.	M	
FE03	Het vision-algoritme kan de plek van een knikker op een knikkerbaan kan voorspellen.	M	
FE04	Het vision-algoritme kan op de Bachmann PLC worden uitgevoerd in de programmacyclus	M	
FE05	Het vision-algoritme moet de afbeelding van JPG naar een bewerkbaar beeld omzetten.	M	IP-camera verzendt afbeelding in JPG-formaat
FE06	De PLC moet een afbeelding kunnen krijgen van de IP-camera.	M	
FE07	De PLC moet een output genereren waarbij het resultaat van het vision-algoritme getoond wordt.	M	Ter controle werking vision-algoritme.
FE08	Het vision-algoritme kan de kleur van de knikker kan herkennen.	S	
FE09	Het vision-algoritme voorspelt alleen de positie van knikker met de verwachte kleur.	S	
FE10	De PLC moet een cartesisch coördinaat genereren op basis van de voorspelde knikkerpositie in de afbeelding.	C	Voor het interacteren met knikker door extern systeem.
FE11	De knikkervoorspelling moet gecontroleerd kunnen worden door middel van een interface.	C	Voor start/stop en aangeven van systeemvariabelen.

TABEL 36 - FUNCTIONELE EISEN

De verwachte uitvoering van de bovengenoemde eisen is opgedeeld in incrementen. Deze planning is in Tabel 37 te vinden.

Increment	Eisen
1: Ophalen Afbeelding	• FE05 • FE06 • SE04 • SE05
2: Positie bepalen knikker	• FE01 • FE02 • FE07
3: Positie voorspellen knikker	• FE03 • FE04 • FE08 • FE09
Extra	• FE10 • FE11

TABEL 37 - INCREMENTPLANNING

3. Increment 1: Ophalen afbeelding

In het eerste increment worden de onderstaande eisen (Tabel 38) uitgevoerd. Hierbij wordt ervoor gezorgd dat de opstelling van het project af is en er een afbeelding gemaakt kan worden die te bewerken is.

Increment	Eisen
1: Ophalen Afbeelding	• FE05 • FE06 • SE04 • SE05

TABEL 38 - EISEN VOOR INCREMENT 1

Van de functionele eisen is een object georiënteerd model gemaakt. Hiervoor is een use-casediagram gemaakt. Waarna de use-casebeschrijvingen zijn opgesteld. Op basis van deze use-cases is een klassendiagram en sequentiediagram opgesteld.

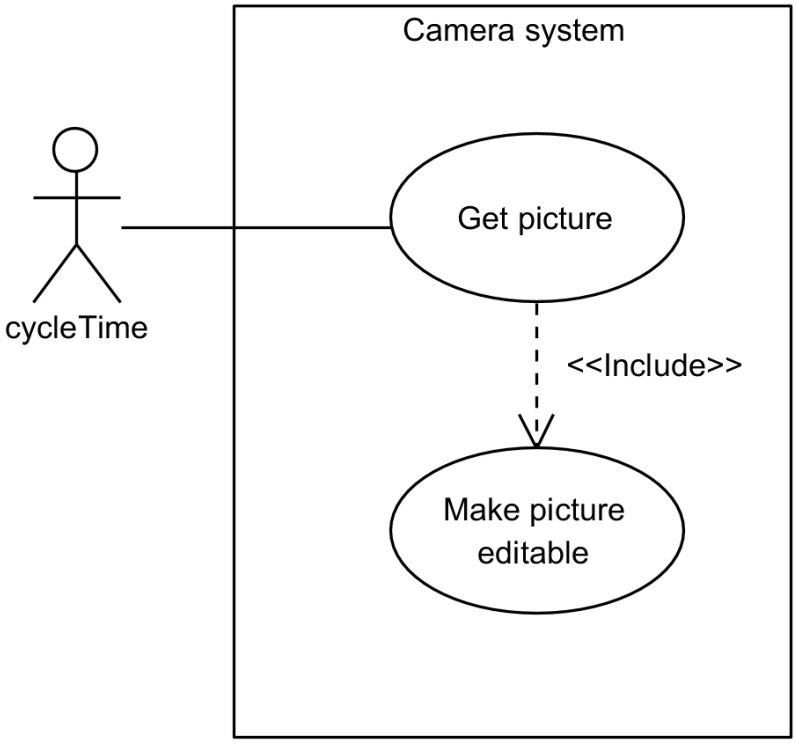
Bij het vision-algoritme worden er verschillende taken aangemaakt die elk een aparte functie hebben. In dit increment wordt een taak aangemaakt voor het ophalen van afbeeldingen. In de volgende incrementen wordt er een taak aangemaakt die deze afbeeldingen bewerkt. Dit is het producer/consumer principe van multi-threading. De taak dat de afbeeldingen opvraagt is de producer en de taak dat de afbeeldingen bewerkt de consumer. Door dit toe te passen kunnen er continu afbeeldingen worden opgehaald zonder dat er gewacht hoeft te worden op de bewerking van de afbeelding. Dit is niet het geval als alle code in één taak uitgevoerd zou worden. Dan kan er pas een nieuwe afbeelding worden opgehaald als alle code die erna komt voltooid is.

3.1 Use-casediagram

In dit deelhoofdstuk is het use-casediagram opgesteld, waarin de functionele eisen van increment 1 verwerkt zijn. Voor increment 1 moeten er twee functies gerealiseerd zijn.

- Het ophalen van een afbeelding bij de camera.
- Het decoderen van de afbeelding, zodat deze bewerkbaar is.

Het opvragen van de afbeelding wordt elke cyclus van de PLC uitgevoerd. Na het ontvangen van de afbeelding wordt deze direct gedecodeerd. In het geval van een IP-camera is dit het decoderen van een JPG-afbeelding naar een RGB-afbeelding. Aan de hand van de bovengenoemde uit te voeren handelingen is in Figuur 36 het use-casediagram opgesteld.



FIGUUR 36 - USE-CASEDIAGRAM CAMERA SYSTEEM

In het bovenstaande use-casediagram wordt de interactie met de camera beschreven. Hierdoor is 'camera system' de naam voor dit diagram. Aangezien er elke cyclus een afbeelding opgevraagd wordt is de actor, die opvragen van de afbeelding start, de cyclustijd. Binnen het opvragen van de afbeelding wordt ook het decoderen van de afbeelding uitgevoerd. Hierdoor valt dit onder het opvragen van de afbeelding en heeft het geen eigen actor.

Bij het use-casediagram zijn de use-casebeschrijvingen opgesteld in Tabel 39 en Tabel 40.

ID:	UC1
Use case:	Get Picture
Brief description:	Opvragen van een afbeelding aan het begin van de cyclustijd.
Primary actors:	Cycle Time
Secondary actors:	Geen
Preconditions:	2. Een nieuwe PLC-cyclus is begonnen
Main flow:	7. De PLC maakt verbinding met de camera. 8. De camera stuurt connectie bevestiging. 9. De PLC vraagt een afbeelding op bij de camera. 10. De camera stuurt de afbeelding. 11. Include (Make picture editable). 12. PLC slaat afbeelding op.
Postconditions:	2. De afbeelding is opgeslagen
Alternative flows:	None

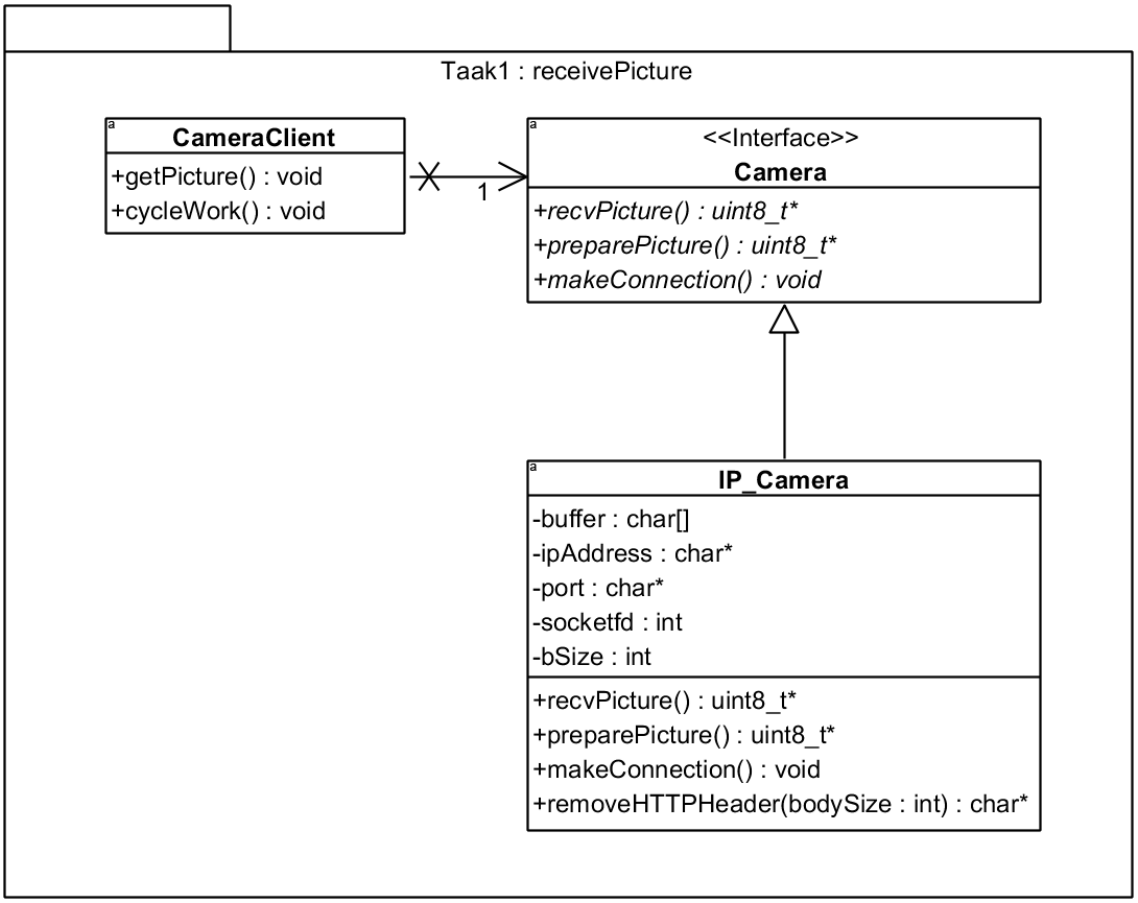
TABEL 39 - BESCHRIJVING 'GET PICTURE'

ID:	UC2
Use case:	Make picture editable
Brief description:	Het decoderen van de afbeelding zodat deze bewerkt kan worden.
Primary actors:	Cycle Time
Secondary actors:	Geen
Preconditions:	1. Een nieuwe PLC-cyclus is begonnen
Main flow:	1. De PLC ontvangt de afbeelding van de camera. 2. De PLC decodeert de afbeelding naar een RGB-afbeelding.
Postconditions:	1. De afbeelding is een bewerkbare RGB-afbeelding
Alternative flows:	None

TABEL 40 - BESCHRIJVING 'MAKE PICTURE EDITABLE'

3.2 Klassendiagram

In Figuur 37 is het klassendiagram van increment 1 gegeven. De klassen zijn gebaseerd op de werking van de PLC en het use-casediagram. In het klassendiagram is een <<interface>> camera opgesteld, zodat het makkelijker is om zo nodig andere soorten camera's te implementeren.



FIGUUR 37 - KLASSENDIAGRAM OPVRAGEN AFBEELDING

3.2.1 CameraClient

CameraClient is een klasse die voortkomt uit de werking van een Bachmann PLC. In een Bachmann PLC kunnen zogenoemde taken gestart worden. Binnen deze taak wordt een methode cyclisch aangeroepen. In het geval van CameraClient is dit de methode '*cycleWork()*'. Bij het aanmaken van een taak kan de cyclustijd en de prioriteit worden aangegeven. De cyclustijd geeft aan hoe vaak de taak moet worden uitgevoerd en de prioriteit geeft aan hoe belangrijk de taak is. Als er namelijk meerdere taken zijn, wordt eerst de taak met de hoogste prioriteit uitgevoerd. Elke taak wordt uitgevoerd in zijn eigen thread.

3.2.2 IP_camera

Tussen de PLC en de camera wordt een TCP-connectie opgezet. Hierbij is de PLC de client en de camera de server. De PLC vraagt afbeeldingen op bij de camera door middel van een HTTP-verzoek. Doordat de afbeelding op deze manier verzonden wordt, moet er rekening gehouden worden met HTTP-headers. Deze moeten namelijk verwijderd worden voordat de afbeelding kan worden opgeslagen. Om een afbeelding op te kunnen vragen moet een socket geopend worden en deze verbinden met de camera. Hiervoor moeten de volgende attributen bekend zijn:

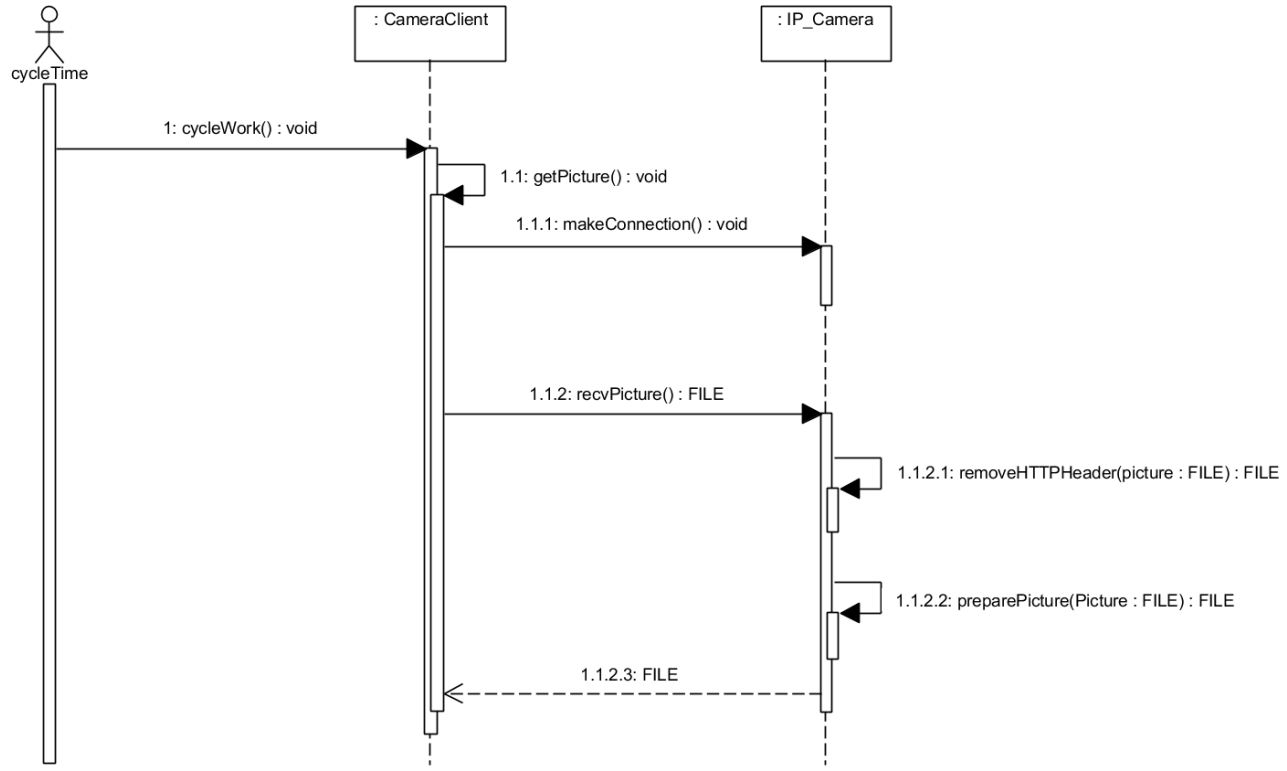
- IP-adres
- Poortnummer van het communicatieprotocol

Met de attributen wordt een sockets geopend en een TCP-verbinding gemaakt met met een uniek ID. Over deze verbinding wordt gebruikt om met de fysieke camera te communiceren.

Na het opvragen van een afbeelding stuurt IP-camera de afbeelding als JPG-formaat. Voordat deze bewerkt kan worden, moet de afbeelding eerst worden gedecodeerd. De methode '*preparePicture()*' is hiervoor verantwoordelijk.

3.3 Sequentiediagram

Op basis van het klassendiagram en de use-casebeschrijvingen is een sequentiediagram opgesteld. Deze is in Figuur 38 te zien. Hierin wordt getoond in welke volgorde de methodes uitgevoerd worden.



FIGUUR 38 - SEQUENTIEDIAGRAM OPVRAGEN AFBEELDING

4. Increment 2: Positie bepalen knikker

In het eerste increment test lag de focus op het ophalen van de afbeelding bij de IP-camera. In het tweede increment moet deze afbeelding bewerkt worden, zodat de knikker kan worden geïsoleerd van de rest en de positie hiervan kan worden bepaald. De eisen die bij dit increment horen zijn in Tabel 41 herhaald.

Increment	Eisen
2: Positie bepalen knikker	• FE01 • FE02 • FE07

TABEL 41 - EISEN POSITIE BEPALEN KNIKKER

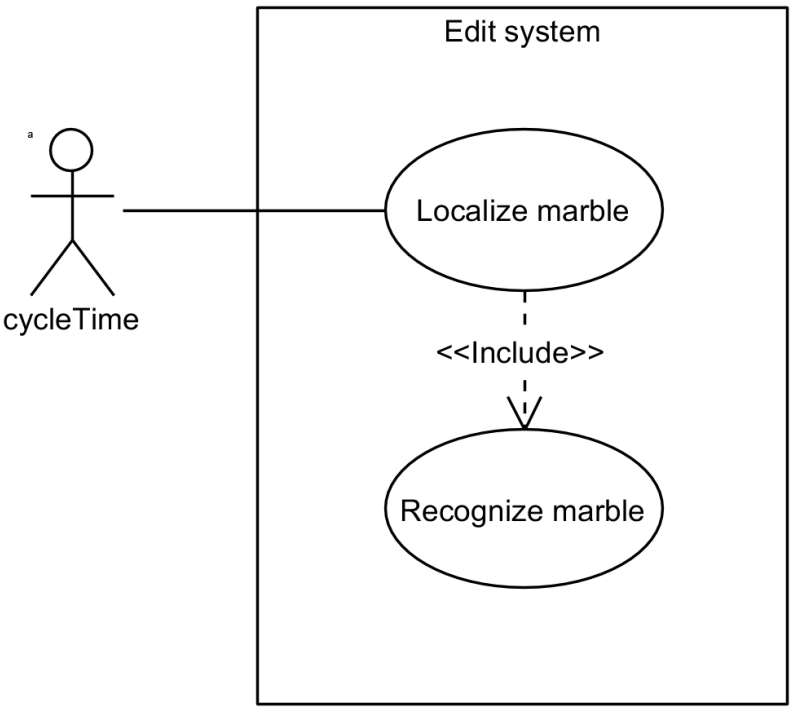
Het ontwerp van deze eisen bestaat uit dezelfde onderdelen als in increment 1.

4.1 Use-casediagram

In dit deelhoofdstuk is het use-casediagram opgesteld, waarin de functionele eisen van increment 2 verwerkt zijn. Voor increment 2 moeten er drie functies gerealiseerd zijn.

- Herkennen van de knikker op een afbeelding
- Het lokaliseren van de knikker op de afbeelding
- Opslaan bewerkte beeld om de werking van algoritme aan te tonen.

Het opslaan van het bewerkte beeld is niet opgenomen in het use-casediagram, omdat dit een losse functie is en niks toevoegt aan het doel van de positievoorspelling, maar gebruikt wordt ter controle of het vision-algoritme naar behoren werkt. In het klassendiagram hoofdstuk 4.2 wordt dit wel meegenomen in het ontwerp. In Figuur 39 is het use-casediagram opgesteld.



FIGUUR 39 - USE-CASEDIAGRAM POSITIE BEPALEN KNIKKER

In Tabel 42 en Tabel 43 zijn de beschrijvingen van de use-cases uitgewerkt.

ID:	UC3
Use case:	Localize marble
Brief description:	Lokaliseren van knikker op een afbeelding
Primary actors:	Cycle Time
Secondary actors:	Geen
Preconditions:	3. Een nieuwe PLC-cyclus is begonnen 4. Er is een bewerkbare afbeelding beschikbaar
Main flow:	9. De PLC vraagt een bewerkbare afbeelding op. 10. De PLC ontvangt de bewerkbare afbeelding. 11. Include (Recognize marble) 12. De PLC ontvangt een bewerkte afbeelding met knikker 13. De PLC zoekt het middelpunt van de knikker 14. De PLC ontvangt het middelpunt 15. De PLC zet een marker op het middelpunt 16. De PLC slaat afbeelding op.
Postconditions:	2. De afbeelding met een marker op het middelpunt van de knikker is opgeslagen
Alternative flows:	None

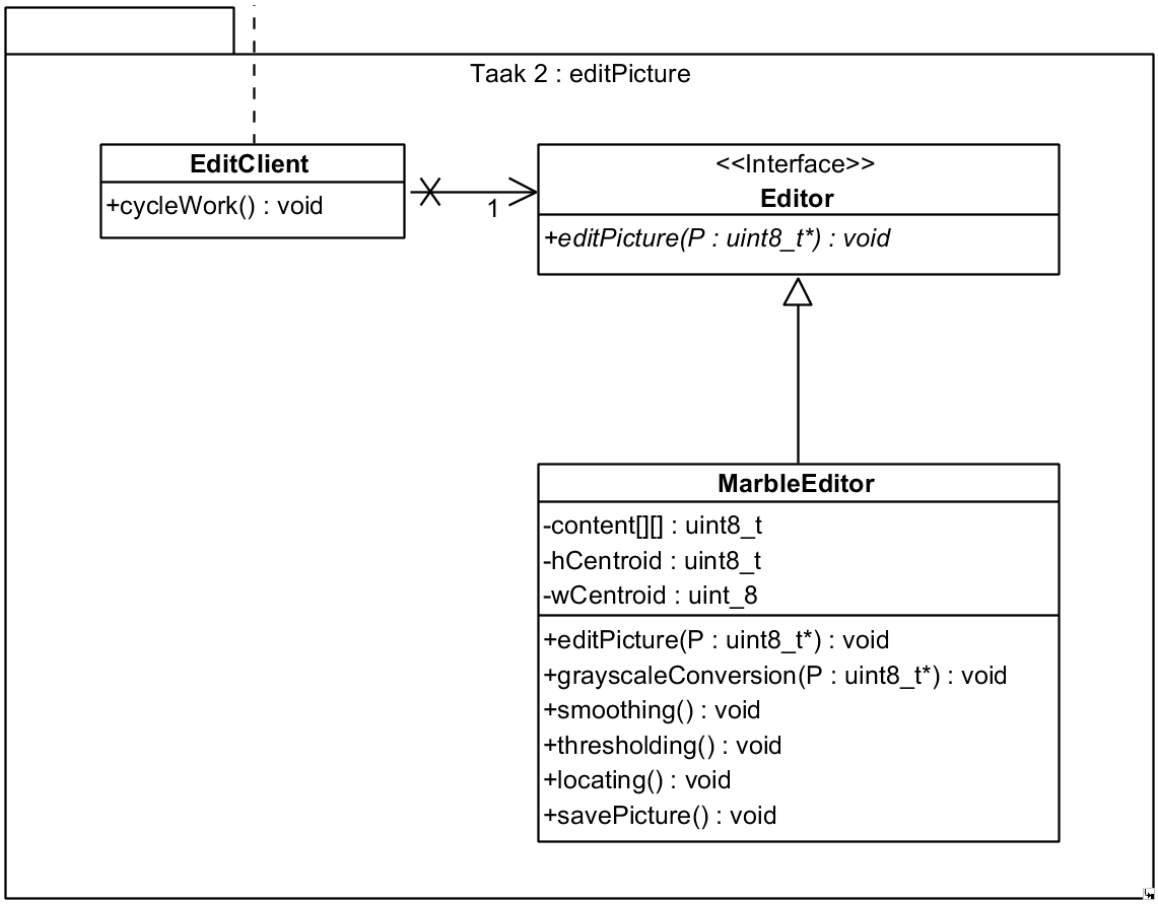
TABEL 42 - BESCHRIJVING 'LOCALIZE MARBLE'

ID:	UC4
Use case:	Recognize marble
Brief description:	Herkennen van een knikker op een afbeelding
Primary actors:	Cycle Time
Secondary actors:	Geen
Preconditions:	1. Een nieuwe PLC-cyclus is begonnen 2. Er is een bewerkbare afbeelding beschikbaar
Main flow:	1. De PLC voert greyscale-conversie uit op de afbeelding 2. De PLC ontvangt een zwartwitafbeelding 3. De PLC voert smoothing uit op de afbeelding 4. De PLC ontvangt een vervaagde afbeelding 5. De PLC voert thresholding uit op de afbeelding 6. De PLC ontvangt een binaire afbeelding
Postconditions:	1. De afbeelding is bewerkt zodat de knikker geïsoleerd is van de rest van de afbeelding
Alternative flows:	None

TABEL 43 - BESCHRIJVING 'RECOGNIZE MARBLE'

4.2 Klassendiagram

In dit hoofdstuk is het klassendiagram voor increment 2 opgesteld. Hierbij is er gelet op onderhoudbaarheid en uitbreidbaarheid. Voor deze doeleinde is er een <<interface>> toegevoegd. In Figuur 40 is het opgestelde klassendiagram te vinden.



FIGUUR 40 - KLASSENDIAGRAM POSITIE BEPALEN KNIKKER

4.2.1 EditClient

EditClient is net als CameraClient (hoofdstuk 3.3) een klasse die de Bachmann PLC gebruikt voor het cyclisch uitvoeren van de code. Deze klasse wordt in een nieuwe taak uitgevoerd om het bewerken van een afbeelding te scheiden van het ophalen van een afbeelding. EditClient is verbonden aan CameraClient voor het verkrijgen van de afbeelding die CameraClient heeft gekregen van de IP-camera

4.2.2 MarbleEditor

MarbleEditor is de overerfde klasse van Editor <<interface>>. Binnen deze klasse worden de methodes geïmplementeerd voor het bewerken van de afbeelding en het lokaliseren van de knikker. Om vanuit EditClient de bewerkingsmethodes aan te roepen, is er een methode 'editPicture()' opgesteld die als brug werkt tussen EditClient en MarbleEditor. Deze methode moet ook als virtuele methode gedefinieerd worden in de klasse Camera.

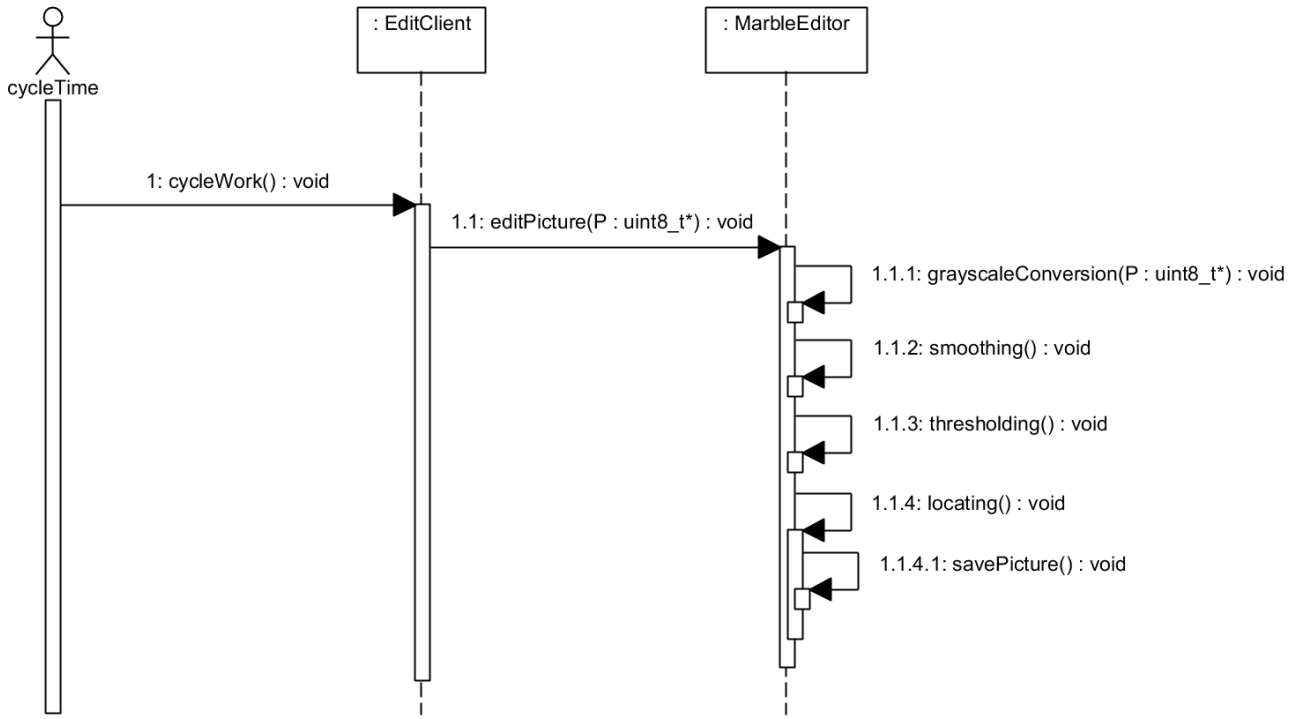
Voor het bewerken van de afbeelding en het isoleren van de knikker moeten er een paar achtereenvolgende bewerkingsmethodes uitgevoerd worden.

- greyscaleConversion()
- smoothing()
- thresholding()

Dit zijn methodes die zijn behandeld in het analyserapport (Bijlage B). Naast de bewerkingen is er een methode opgesteld voor het lokaliseren van de knikker nadat deze is geïsoleerd en één voor het opslaan van de bewerkte afbeelding. Het opslaan van de afbeelding is nodig om de functionaliteit van de methodes te controleren.

4.3 Sequentiediagram

Op basis van het klassendiagram en de use-casebeschrijvingen is een sequentiediagram opgesteld. Deze is in Figuur 41 te zien. Hierin wordt getoond in welke volgorde de methodes uitgevoerd worden.



FIGUUR 41 - SEQUENTIEDIAGRAM POSITIE BEPALEN KNIKKER

5. Increment 3: Positie voorspellen knikker

In het derde increment wordt er de positie voorspelling van de knikker ontworpen. Hierbij wordt de positie van de knikker uit increment 2 gebruikt. De eisen die bij dit increment horen zijn in Tabel 44 herhaald.

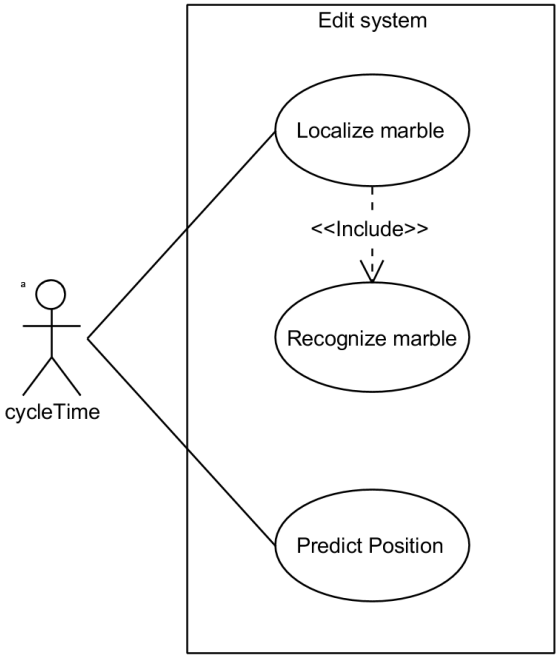
Increment	Eisen
3: Positie voorspellen knikker	• FE03 • FE04 • FE08 • FE09

TABEL 44 - EISEN POSITIE VOORSPELLEN KNIKKER

Het ontwerp van deze eisen bestaat uit dezelfde onderdelen als in increment 1.

5.1 Use-casediagram

Tijdens dit increment is de functionaliteit voor FE03 ontworpen. Hiervoor is een extra use-case toegevoegd in het use-casediagram “Edit System”. Het aangepaste use-casediagram is te vinden in Figuur 42 en de bijbehorende beschrijving in Tabel 45.



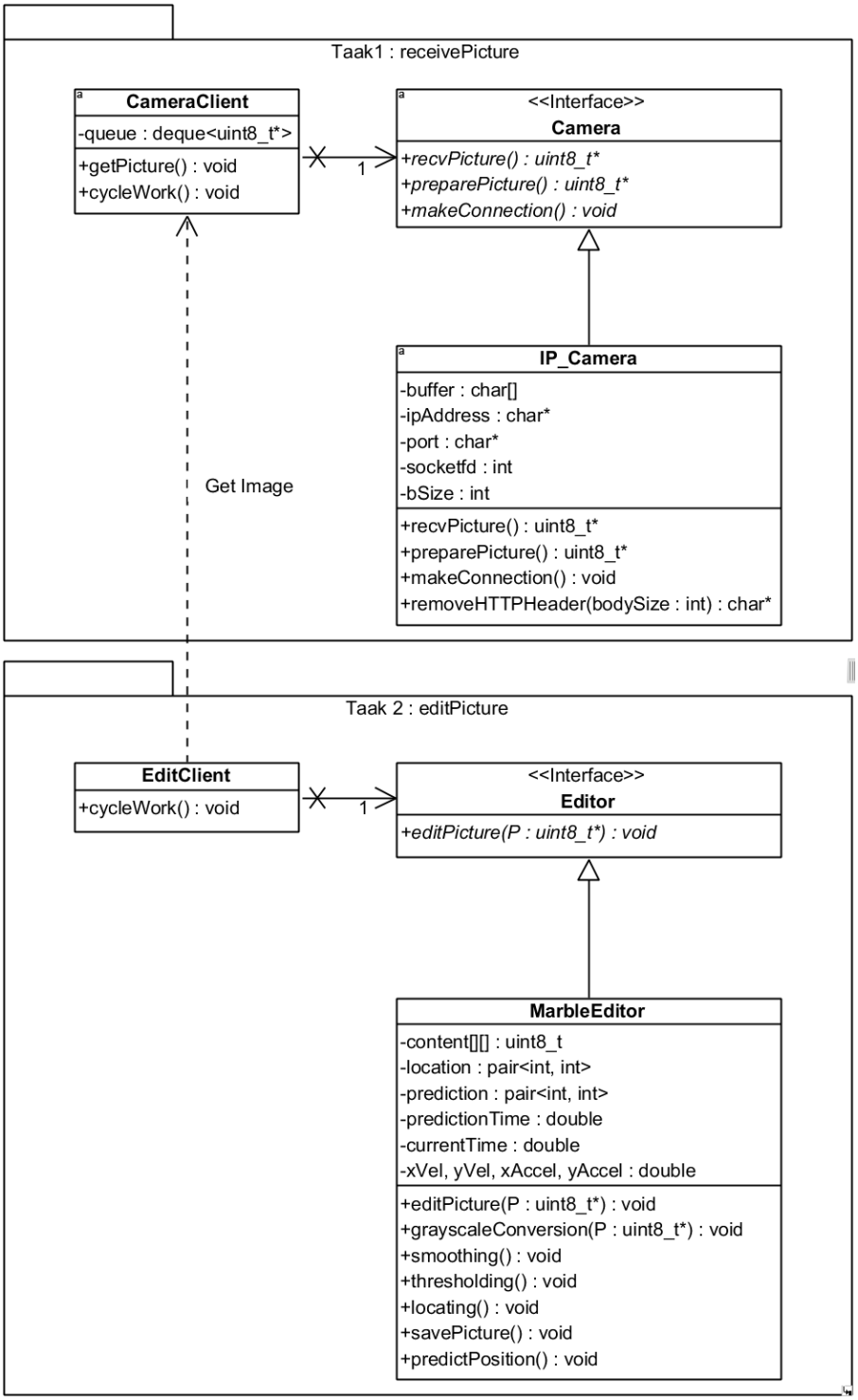
FIGUUR 42 - USE-CASEDIAGRAM POSITIE VOORSPELLEN KNIKKER

ID:	UC5
Use case:	Predict Position
Brief description:	Het voorspellen van de positie van een knikker
Primary actors:	Cycle Time
Secondary actors:	Geen
Preconditions:	2. De vorige en huidige positie van de knikker is beschikbaar.
Main flow:	4. De PLC ontvangt de huidige positie. 5. De PLC berekend de versnelling. 6. De PLC berekend acceleratie. 7. De PLC berekend voorspelling.
Postconditions:	2. Er is een positie voorspeld van de knikker
Alternative flows:	None

TABEL 45 - USE-CASEBESCHRIJVING 'PREDICT POSITION'

5.2 Klassendiagram

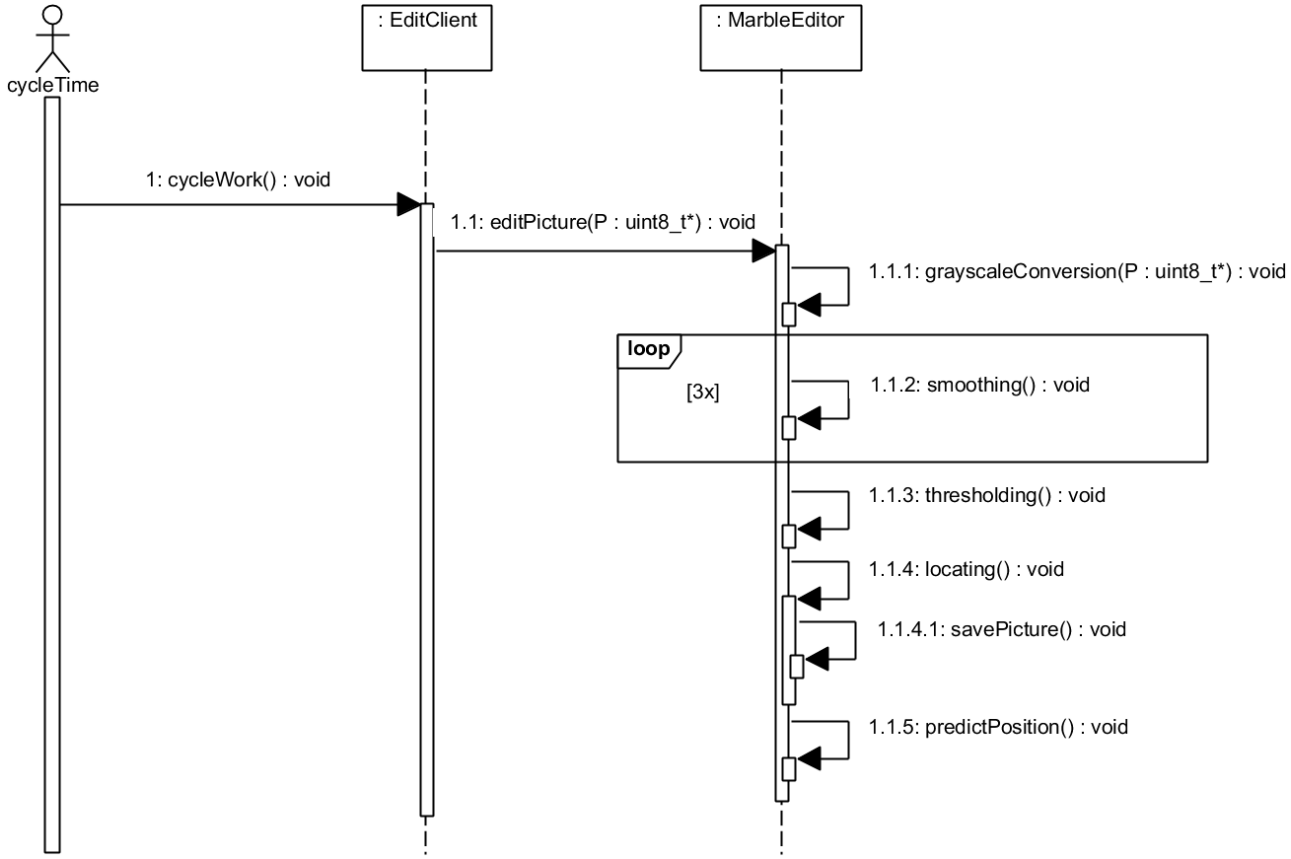
Tijdens dit increment is alleen de functie predictPosition() en de benodigde attributen toegevoegd. Aangezien er weinig verandering zijn is in Figuur 43 het volledige klassendiagram getoond.



FIGUUR 43 - KLASSENDIAGRAM POSITIE VOORSPELLEN KNIKKER

5.3 Sequentiediagram

Tijdens increment 3 zijn er aanpassingen gemaakt in het sequentiediagram van increment 2. Zo moest smoothing 3x uitgevoerd worden en is predictPosition() toegevoegd. Dit is het uiteindelijke sequentiediagram van taak 2. In Figuur 44 is dit diagram opgesteld.



FIGUUR 44 - SEQUENTIE POSITIE VOORSPELLEN KNIKKER

Bijlage E: Testrapport

Het herkennen en locatie voorspellen van knikkers op basis van kleur met een camera en PLC bij Alten Nederland.

Versie: 1.0
Status: Definitief
Datum: 31 mei 2019
Auteur: Hanno van Megchelen

Content

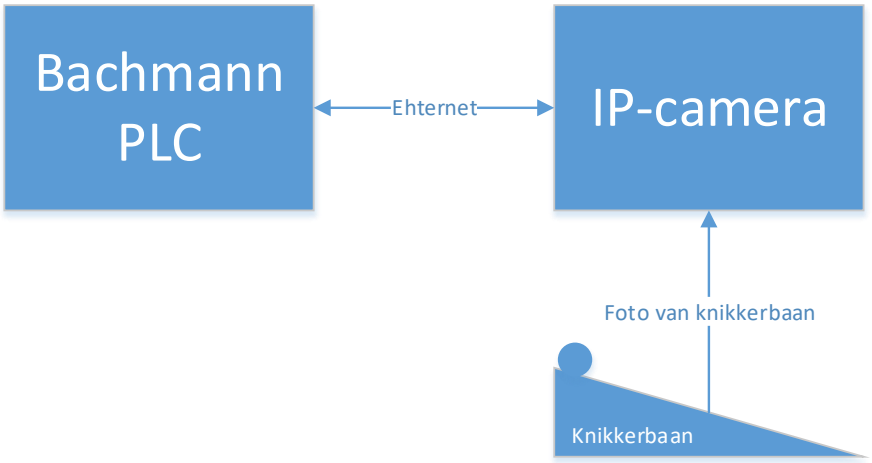
1. Inleiding

In opdracht van ALTEN Nederland wordt een afstudeerproject uitgevoerd, waarbij er de positie van een knikker moest worden voorspeld met een behulp van een camera en een Bachmann PLC. Voor de uitvoering van het project was het nodig om de software te testen, zodat er gecontroleerd wordt aan welk is voldaan.

In dit rapport zijn er per increment testen opgesteld. Hierbij is er per use-case een test gemaakt. Hierin wordt er een sequentie aan handelingen genoemd die de tester uit moet voeren. Na deze handelingen is er een verwacht resultaat. Als het resultaat overeenkomt met het verwachte resultaat is er aan de eis voldaan.

2. Opdracht en eisen

Er moet een vision-algoritme gemaakt worden dat knikkers op een knikkerbaan herkent en zijn positie voorspelt, zodat een robotarm deze op de voorspelde positie kan oppakken. Bij dit project is de scope het maken van het vision-algoritme en niet het oppakken van de knikker. Het vision-algoritme moet op een Bachmann PLC geïmplementeerd worden, zodat het algoritme cyclisch uitgevoerd wordt. De PLC heeft een connectie met een IP-camera om een foto te maken van de knikkerbaan. Een schematische weergave van de opstelling staat om Figuur 45.



FIGUUR 45 - SCHEMATISCHE WEERGAVE OPSTELLING

In het analyserapport (bijlage B) zijn de eisen van het project opgesteld. Deze zijn verdeeld in systeemeisen en functionele eisen. Deze eisen zijn herhaald in Tabel 46 en Tabel 47.

ID	Eis	Herkomst eis
SE01	De camera moet kunnen communiceren met de PLC.	Zonder communicatie kan de afbeelding niet verkregen worden, dus kan er geen knikker herkend worden.
SE02	De camera ondersteund worden door de PLC.	Volgt uit hoofdstuk 3. De PLC heeft beperkte ondersteuning voor camera's.
SE03	De camera kan een kleurenafbeelding maken.	Volgt uit grenzen in plan van aanpak.
SE04	De knikkerbaan mag niet dezelfde kleur zijn als de knikkers.	Volgt uit hoofdstuk 4. Als er weinig kleurverschil is, dan kan de knikker niet herkend worden.
SE05	Knikkerbaan moet lineair zijn.	Volgt uit grenzen in plan van aanpak.
SE06	De knikkers mogen niet transparant zijn.	Volgt uit grenzen in plan van aanpak.

TABEL 46 - SYSTEEMEISEN

ID	Eis	Prioriteit	Commentaar
FE01	Het vision-algoritme kan een non-transparante knikker kan herkennen.	M	Transparante knikkers zijn uitgesloten.
FE02	Het vision-algoritme kan de positie van de knikker bepalen.	M	
FE03	Het vision-algoritme kan de plek van een knikker op een knikkerbaan kan voorspellen.	M	
FE04	Het vision-algoritme kan op de Bachmann PLC worden uitgevoerd in de programmacyclus	M	
FE05	Het vision-algoritme moet de afbeelding van JPG naar een bewerkbaar beeld omzetten.	M	IP-camera verzendt afbeelding in JPG-formaat
FE06	De PLC moet een afbeelding kunnen krijgen van de IP-camera.	M	
FE07	De PLC moet een output genereren waarbij het resultaat van het vision-algoritme getoond wordt.	M	
FE08	Het vision-algoritme kan de kleur van de knikker kan herkennen.	S	
FE09	Het vision-algoritme voorspelt alleen de positie van knikker met de verwachte kleur.	S	
FE10	De PLC moet een cartesisch coördinaat genereren op basis van de voorspelde knikkerpositie in de afbeelding.	C	Voor het interacteren met knikker door extern systeem.
FE11	De knikkervoorspelling moet gecontroleerd kunnen worden door middel van een interface.	C	Voor start/stop en aangeven van systeemvariabelen.

TABEL 47 - FUNCTIONELE EISEN

De verwachte uitvoering van de bovengenoemde eisen is opgedeeld in incrementen. Deze planning is in Tabel 48 te vinden.

Increment	Eisen
1: Ophalen Afbeelding	• FE05 • FE06 • SE04 • SE05
2: Positie bepalen knikker	• FE01 • FE02 • FE07
3: Positie voorspellen knikker	• FE03 • FE08 • FE09
Extra	• FE10 • FE11

TABEL 48 - INCREMENTPLANNING

Meer informatie over tabel 1, 2 en 3 is te vinden in het analyserapport (bijlage …)

3. Increment 1: Ophalen afbeelding

In dit hoofdstuk worden de testen beschreven die zijn uitgevoerd om te controleren of de software voldoet aan de functionele eisen van increment 1. In Tabel 49 zijn deze functionele eisen gegeven. In bijlage C zijn de use-cases opgesteld die volgen uit de functionele eisen. De testen zijn opgesteld aan de hand van deze use-cases.

Increment	Eisen
1: Ophalen Afbeelding	• FE05 • FE06 • SE04 • SE05

TABEL 49 - EISEN VOOR INCREMENT 1

3.1 Testen

In Tabel 50 en Tabel 51 zijn de testen gegeven die zijn uitgevoerd op 25-03-2019 om 11:34 door Hanno van Megchelen.

Test	1. FE05
Use-case	UC01: Get picture
Omschrijving	Opvragen van een afbeelding aan het begin van de cyclustijd.
Pre-conditie	Er staat geen afbeelding in de map waar marbleVoorspeller.exe is opgeslagen
Testproces	1. Ga naar map waar marbelVoorspeller.exe is opgeslagen 2. Open marbleVoorspeller.exe
Verwachte uitkomst	In de map van marbleVoorspeller.exe is een picture.jpg gemaakt met het huidige beeld van de camera.
Uitkomst	Uitkomst is zoals verwacht.

TABEL 50 - TEST UC01

Test	1. FE06
Use-case	UC02: Make picture editable
Omschrijving	Het decoderen van een afbeelding, zodat elke pixel uit een RGB-waarde bestaat.
Pre-conditie	Er is een afbeelding opgevraagd bij de camera en in de map van marbleVoorspeller.exe staat deze afbeelding als picture.jpg
Testproces	1. Ga naar map waar marbelVoorspeller.exe is opgeslagen 2. Open marbleVoorspeller.exe
Verwachte uitkomst	Er verschijnt een scherm met de RGB-waardes van elke pixel
Uitkomst	Uitkomst is zoals verwacht.

TABEL 51 - TEST UC02

4. Increment 2: Positie bepalen knikker

In dit hoofdstuk worden de testen beschreven die zijn uitgevoerd om te controleren of de software voldoet aan de functionele eisen van increment 2. In Tabel 52 zijn de functionele eisen gegeven die zijn gerealiseerd in increment 1. In bijlage C zijn de use-cases opgesteld die volgen uit de functionele eisen. De testen zijn opgesteld aan de hand van deze use-cases.

Increment	Eisen
2: Positie bepalen knikker	• FE01 • FE02 • FE07

TABEL 52 - EISEN VOOR INCREMENT 2

4.1 Testen

In Tabel 53 en Tabel 54 zijn de testen opgesteld die zijn uitgevoerd op 15-04-2019 om 10:05 door Hanno van Megchelen.

Test	1. FE01 FE07
Use-case	UC03: Edit picture
Omschrijving	Bewerken van een afbeelding om knikker te onderscheiden.
Pre-conditie	Er in een afbeelding 'picture.jpg' beschikbaar, waarin een knikker afgebeeld is. Deze .jpg staat in de map van marbleVoorspeller.exe
Testproces	1. Ga naar map waar marbelVoorspeller.exe is opgeslagen 2. Open marbleVoorspeller.exe
Verwachte uitkomst	In de map van marbleVoorspeller.exe is de afbeelding threshold.pgm gemaakt met het resultaat van de bewerkingen. In deze afbeeldingen is de knikker wit en de rest zwart.
Uitkomst	Uitkomst is zoals verwacht.

TABEL 53 - TEST UC03

Test	1. FE02
Use-case	UC04: Locate marble
Omschrijving	Het decoderen van een afbeelding, zodat elke pixel uit een RGB-waarde bestaat.
Pre-conditie	Er in een afbeelding 'picture.jpg' beschikbaar, waarin een knikker afgebeeld is. Deze .jpg staat in de map van marbleVoorspeller.exe
Testproces	1. Ga naar map waar marbelVoorspeller.exe is opgeslagen 2. Open marbleVoorspeller.exe
Verwachte uitkomst	Er wordt een pixelcoördinaat geprint dat overeenkomt met een stip op de gemaakte afbeelding centroid.pgm. De stip zit in het middel van de knikker.
Uitkomst	Uitkomst is zoals verwacht.

TABEL 54 - TEST UC04

5. Increment 3: Positie voorspellen knikker

In dit hoofdstuk worden de testen beschreven die zijn uitgevoerd om te controleren of de software voldoet aan de functionele eisen van increment 3. In Tabel 55 zijn de functionele eisen gegeven die zijn gerealiseerd in increment 3. In bijlage C zijn de use-cases opgesteld die volgen uit de functionele eisen. De testen zijn opgesteld aan de hand van deze use-cases.

Increment	Eisen
2: Positie voorspellen knikker	• FE03 • FE04 • FE08 • FE09

TABEL 55 - EISEN VOOR INCREMENT 3

5.1 Testen

In Tabel 56 is de test opgesteld die zijn uitgevoerd op 10-05-2019 om 15:00 door Hanno van Megchelen.

Test	1. FE03 FE04
Use-case	UC05: Predict Position
Omschrijving	Met de Bachmann PLC de voorspelling van de positie op 1500ms uitvoeren.
Pre-conditie	De Bachmann PLC staat aan en het programma is geüpload. Device-view is geopend in Solutioncenter.
Testproces	1. Ga bij device naar de geïnstalleerde applicaties. 2. Start MarblePredictor.
Verwachte uitkomst	Er wordt na elke cyclus een huidige positie getoond met een positievoorspelling waar de knikker op het tijdstip 1500ms gaat zijn. 900ms voordat de knikker op het te voorspellen punt is, wordt er een voorspelling gegeven die maximaal 30 pixels afwijking heeft.
Uitkomst	Uitkomst is zoals verwacht.

TABEL 56 - TEST UC05