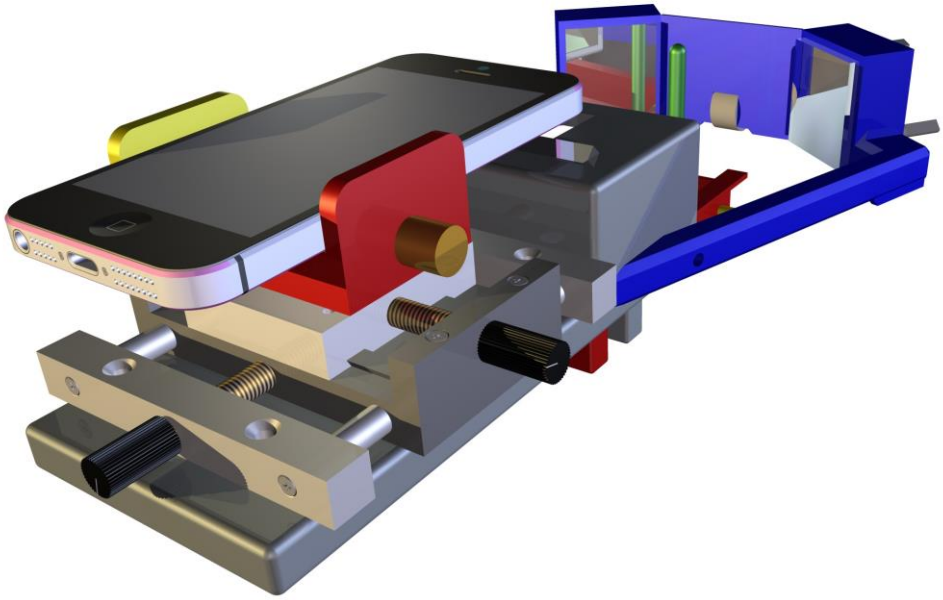
Software voor de contactloze kopdiktemeter



Afdeling	IT & Design
Studie	Technische Informatica, Delft
Schooljaar	2014 – 2015
Afstudeerblok	September 2014 – juni 2015
Startdatum	September 2014
Einddatum	Juni 2015
Begeleider/examinator	J.D. Schagen
Tweede begeleider/examinator	M.G. Maris
Versie	0.5
Datum	4 juni 2015
Student	Kaj Toet
Studentnummer	11102470

Versiebeheer

Versie	Datum	Auteur	Aanpassingen
0.1	11-12-'14	Kaj Toet	Eerste draft
0.2	12-1-'14	Kaj Toet	Tweede draft met nieuwe hoofdstukindeling en notities
0.3	2-3-'15	Kaj Toet	Tussentijds Assessment
0.4	25-3-'15	Kaj Toet	Tussentijds Assessment
0.5	4-6-'15	Kaj Toet	Inlevering

Distributielijst

Naam	Organisatie	Functie	Reden
J.H.R. Lambers	De Haagse Hogeschool	Opdrachtgever	Proces begeleider
J.D. Schagen	De Haagse Hogeschool	Docent	1ste Examinator
M.G. Maris	De Haagse Hogeschool	Docent	2de Examinator

Referaat

Toet, K., Afstudeerverslag “Software voor de contactloze kopdiktemeter”, Delft, De Haagse Hogeschool.

Dit verslag beschrijft het proces en de activiteiten die plaats hebben gevonden in de periode van september 2014 tot maart 2015 in het kader van bovengenoemde afstudeeropdracht. De opdracht is uitgevoerd door K. Toet, student Technische Informatica aan De Haagse Hogeschool Delft.

De opdracht betreft het uitwerken van een idee van het bedrijf Green Eyes voor het bepalen van de kopdikte van een plantensteel. Hiervoor dient een applicatie voor een Android smartphone ontwikkeld te worden, die de nodige informatie verzamelt en naar een computer verstuurt voor beeldanalyse. Vervolgens stuurt de computer de uitkomst van de analyse terug naar de gebruiker van de smartphone. Het verzamelen van de gegevens moet contactloos geschieden, opdat de steel geen afwijkend groeiproces krijgt.

Aanvullend is onderzocht of de beeldanalyse op de Android telefoon zelf kan plaatsvinden, omdat dit de noodzaak om beeldanalyse op een externe computer uit te voeren, overbodig maakt

Descriptoren:

- Kopdiktemeter
- Android
- Software ontwikkelen
- Mobiel platform
- Netwerkcommunicatie

Adresgegevens

Afstudeerder

Naam	Kaj Toet
Telefoonnummer	06-43595507
E-mailadres	kvtoet@gmail.com
Onderwijsinstelling	De Haagse Hogeschool
Afstudeerrichting	Technische Informatica, Software Ontwikkeling
Studentnummer	11102470

Afstudeerbegeleider

Naam	J.D. Schagen
E-mailadres	j.d.schagen@hhs.nl
Telefoonnummer	+31 (0)70 445 8410

Tweede afstudeerbeleider

Naam	M.G. Maris
E-mailadres	m.g.maris@hhs.nl
Telefoonnummer	070-4458441

Bedrijfsgegevens

Naam	De Haagse Hogeschool
Straat / Nummer	Rotterdamseweg 137
Postcode/ Plaats	2628AL, Delft

Opdrachtgever

Naam	Jan Lambers
Telefoonnummer	015-2606335
E-mailadres	j.h.r.lambers@hhs.nl

Samenvatting

In het kader van een samenwerkingsverband van de HHS, vertegenwoordigd door de heer Lambers en de organisatie GreenEyes, heeft Kaj Toet, student Technische Informatica aan de HHS, zich gedurende 17 weken bezig gehouden met het ontwikkelen van een applicatie voor Android smartphones.

De applicatie moet contactloos de kopdikte van plantenstelen meten. Daarna moet de applicatie zo snel mogelijk de foto van de steel naar een computer sturen, waar de gegevens geanalyseerd worden. Vervolgens moeten de resultaten van de analyse (o.a. het aantal vruchten van de plant) terug worden gekoppeld naar de gebruiker van de Android smartphone. De gebruiker heeft de informatie nodig om de te verwachten vruchtproductie van het gewas in te schatten en indien nodig in te grijpen om de vruchtproductie te verhogen.

Green Eyes is een samenwerking tussen 41 telers (MKB), een veredelaar in voedingstuinbouw, een teelt technisch toeleveringsbedrijf van gewasmanagement automatisering en klimaatbeheersing, en de kennisinstellingen Hogeschool INHolland, HAS Den Bosch, Wageningen UR, TNO en Lentiz Onderwijsgroep. Het doel van de samenwerking van deze partijen is de huidige teeltkennis te delen en op een hoger plan te brengen.

Het project is uitgevoerd aan de hand van de methode Prototyping. De doelen en eisen zijn tussentijds steeds aangepast als gevolg van de uitkomsten van testen van prototypes. Het was een iteratief proces. In deze afstudeerscriptie wordt verslag gedaan van het ontwikkelproces van de applicatie op de smartphone.

Uit de applicatie die uiteindelijk ontwikkeld is met de afstudeerscriptie, is gebleken dat het meten van de kopdikte van een plantensteel d.m.v. een Android telefoon mogelijk is, al treedt er wel vertraging op door o.a. compressie, decompressie en netwerkcommunicatie. Uit onderzoek is verder gebleken dat de beeldanalyse ook lokaal op de Android telefoon zelf uitgevoerd kan worden, in plaats van op een externe computer. Dit zorgt er voor dat er met alleen de Android telefoon gewerkt kan worden zonder dat er netwerkcommunicatie plaatsvindt.

Inhoud

1 Inleiding 7
2 Definitie afstudeeropdracht 8
3 Aanpak / werkwijze 12
4 Literatuuronderzoek 17
5 Eisen aan de applicatie 20
6 Onderzoek, prototyping en uitwerking 22
7 Systeemontwerp 39
8 Testen 50
9 Conclusie 56
10 Aanbevelingen 57
11 Reflectie 60
12 Bibliografie 62
13 Bijlagen 64

1 Inleiding

1.1 Aanleiding

Door Kaj Toet, afstudeerder van De Haagse Hogeschool Delft, is 17 weken gewerkt aan het ontwikkelen van een applicatie voor Android smartphones om de dikte van plantenstelen te meten. Het project is een uitwerking van een onderzoek van GreenEyes.

1.2 Doel

Het doel van de opdracht is het uitwerken van een onderzoek om de dikte van plantensteel te meten zonder deze aan te raken, zodat er geen celverdikking optreedt waardoor de groei af zal wijken. Deze applicatie wordt ontwikkeld voor een Android telefoon. Door de dikte van een plantensteel te meten kan worden voorspeld hoeveel vruchtengroei een plant zal krijgen. Aan de hand van deze informatie kunnen de omgevingsfactoren van de plant worden aangepast om de productie bij te sturen en de vruchtengroei te maximaliseren.

In dit afstudeerverslag is uitgebreid beschreven hoe de ontwikkeling van de applicatie tot stand is gekomen en hoe de applicatie functioneert en moet worden geconfigureerd. Aan het einde van het verslag worden de toegepaste competenties toegelicht als onderdeel van het afstudeerverslag.

1.3 Organisatie en omgeving

1.3.1 Opdrachtgever

De afstudeeropdracht is in opdracht gegeven door dhr. Lambers van de Haagse Hogeschool.

De positie van Jan Lambers is projectleider “Kopdiktemeter”.

1.3.2 GreenEyes

Via de Haagse Hogeschool en INHolland wordt de opdracht verstrekt om een applicatie te ontwikkelen die tellingen, of schattingen kan maken van de vruchtproductie van planten. INHolland maakt daarbij deel uit van een samenwerkingsverband met Green Eyes en INHolland heeft een samenwerkingsverband met de Haagse Hogeschool. Green Eyes wordt gezien als de eindklant van het systeem. Jan Lambers vertegenwoordigd de Producent, het expertise centrum van de Haagse Hogeschool TIS Delft.

Het ontwikkelen van de Contactloze Kopdiktemeter is een project dat opgezet is door Green Eyes.

> *Green Eyes is een samenwerking tussen 41 telers (MKB), een veredelaar in voedingstuinbouw, een teelt technisch toeleveringsbedrijf van gewasmanagement automatisering en klimaatbeheersing, en de kennisinstellingen Hogeschool Inholland, HAS Den Bosch, Wageningen UR, TNO en Lentiz Onderwijsgroep. Het zal de huidige (wereldwijd meest geavanceerde) teeltkennis wetenschappelijk expliciet en overdraagbaar maken en het effect van dynamisch ingrijpen in de plantbalans begrijpelijk maken. Kennis die onontbeerlijk is voor de toekomstige generatie telers in de schoolbanken en voor de ondernemers van nu die innovaties binnen hun bedrijf ook in het huidige economische klimaat willen blijven realiseren. Daarmee voorziet dit project in een gat in de markt, door middel van eenvoudige, betrouwbare en betaalbare fenotypering, in combinatie met inzichtelijke modellen die de interpretatie van de data verhelderen en het MKB in de glastuinbouw in staat stelt een internationale koppositie te behouden en verder te verstevigen.* [1]

2 Definitie afstudeeropdracht

2.1 Achtergrond

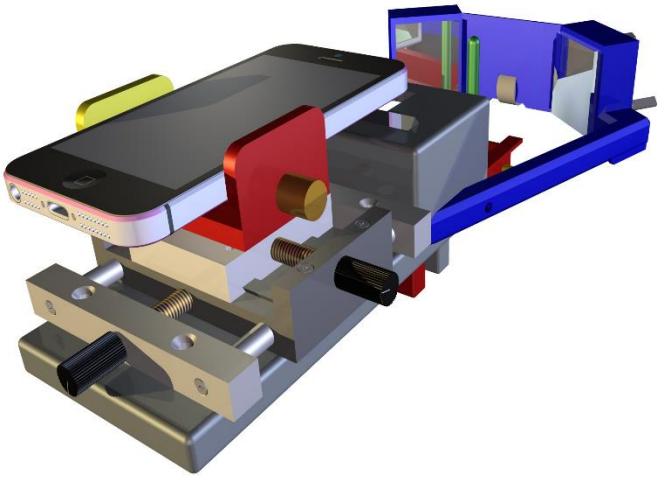
In het samenwerkingsverband Green Eyes wordt gezocht naar manieren om gewasontwikkeling te optimaliseren. Het is gebleken dat er een gebrek aan kennis beschikbaar is over hoe gewassen reageren op nieuwe teeltomstandigheden. Er wordt gezocht naar goedkope oplossingen om gegevens over de ontwikkeling van het gewas te monitoren. Zo kunnen de telers inzicht krijgen in de effecten van bijvoorbeeld duurzamere teeltmethoden.

Eén van deze oplossingen is de zogenaamde contactloze kopdiktemeter. De kopdikte van een gewas is een begrip uit de tuinbouwwereld dat niet exact is gedefinieerd. Meestal wordt er met de dikte van de steel van het gewas, de dikte direct onder de bloemkop bedoeld. Onder de bloemkop wordt de vrucht gevormd. Door deze dikte over een tijdsbestek bij te houden kan er iets gezegd worden over de reactie van het gewas op nieuwe teeltmethodes.

Wanneer men op dit moment nauwkeurige metingen wil verrichten, wordt er bij de meting nog een schuifmaat gebruikt. Hierbij wordt contact gemaakt met de steel van de plant en dit leidt tot onregelmatigheden in het groeipatroon. De plant verzet zich namelijk tegen krachten van buitenaf door extra energie te stoppen in het verstevigen van zijn steel. Als de plant van buitenaf beïnvloed is, kan daardoor niet meer met zekerheid gezegd worden dat zijn groeipatroon ook geldt voor het grootste gedeelte van het gewas, waarop geen metingen verricht zijn.

Een oplossing hiervoor is het contactloos meten van de kopdikte, waardoor de groei niet meer wordt beïnvloed. Green Eyes heeft een project opgestart om dit probleem op te lossen.

Jan Lambers van de Haagse Hogeschool heeft de leiding over dit project. Binnen het project is een opstelling ontwikkeld waaraan een smartphone bevestigd kan worden. Met deze opstelling, die in de hand vastgehouden kan worden, kan de kopdikte met de juiste applicatie op een smartphone worden bepaald.



FIGUUR 1 DE CONTACTLOZE KOPDIKTEMETER MET EEN TELEFOON.

2.2 Doelstelling opdracht

De doelstelling van deze opdracht is het ontwikkelen van een mobiel te hanteren systeem dat contactloos de dikte van een cilindervormig object, zoals een steel, kan vaststellen.

2.3 Hoofdvraag

De hoofdvraag die in het onderzoek dient te worden beantwoord is:

Wat zijn de meest geschikte technieken om camerabeelden van een Android telefoon te faciliteren voor verdere beeldanalyse?

2.4 Deelvragen

De hoofdvraag is in vijf deelvragen opgesplitst. Iedere deelvraag legt de nadruk op een ander deelsysteem van de software. Om de deelvragen te beantwoorden moeten de volgende onderdelen worden onderzocht en ontwikkeld: beeldacquisitie, netwerkcommunicatie, beeldverwerking en visualisatie van de resultaten. Om tot een werkend systeem te komen, moeten deze onderdelen succesvol samengevoegd worden in één geheel.

Dit leidt tot de volgende deelvragen:

1. Welke techniek dient er gebruikt te worden om de camerabeelden te maken?
2. Hoe kan zo nauwkeurig en snel mogelijk met de gekozen cameratechniek informatie uitgewisseld worden tussen de Android smartphone en de computer?
3. Is het mogelijk om de beeldverwerking op de telefoon te laten plaatsvinden?
4. Hoe kan de gebruiker zo duidelijk mogelijk met een User Interface worden voorzien van de resultaten van de analyse?
5. Welke functionaliteiten voor configuratie en debugging hebben toegevoegde waarde?

2.5 Opdrachtbeschrijving

Als de opdracht is uitgevoerd, is het mogelijk de camera-afbeeldingen van een Android telefoon te verzenden naar een externe computer voor analyse. Dit gegeven kan in de toekomst gebruikt worden om de kopdikte van een plantensteel te meten met behulp van de beeldanalyse van de opdrachtgever. Met de beeldanalyse kan worden berekend hoeveel vruchtengroei de desbetreffende plant ongeveer zal hebben.

Met deze oplossing zullen uiteindelijk de leden van Green Eyes betere schattingen kunnen doen van de productie van vruchten en zal het bedrijf gerichter de productie kunnen ondersteunen.

2.6 SMART doelen

Het SMART-principe wordt gebruikt om eenvoudige doelstellingen op te zetten.

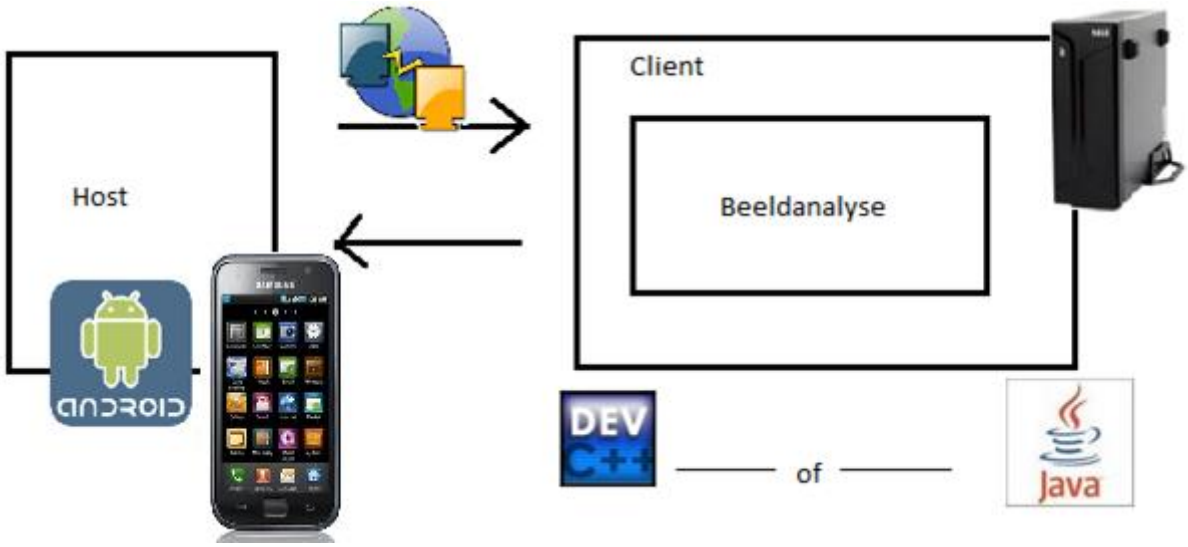
De letters voor SMART staan voor:

- Specifiek
  - Is de doelstelling eenduidig?
- Meetbaar
  - Wanneer is het doel bereikt?
- Acceptabel
  - Is het doel acceptabel voor de doelgroep?
- Realistisch
  - Is het doel haalbaar?
- Tijdgebonden
  - Wanneer is het doel bereikt?

2.6.1 Specifiek

Om de hoofdvraag te beantwoorden is een applicatie op de Android telefoon (host) voor de contactloze kopdiktemeter ontwikkeld. Het betreft een applicatie die hoge resolutie foto's maakt, daarvan een of

meerdere lijnen naar een externe computer stuurt voor analyse. Voor deze externe computer zijn client-applicaties ontwikkeld waar de host software mee kan communiceren.



FIGUUR 2 EEN OVERZICHT VAN DE SOFTWARE VOOR DE CONTACTLOZE KOPDIKTEMETER.

De applicatie op de Android telefoon:

- maakt hoge resolutie afbeeldingen met de camera
- verstuurt hiervan een of meerdere lijnen van pixels over het netwerk
- ontvangt de resultaten en toont deze aan de gebruiker

De applicatie op de Android telefoon besteedt de beeldanalyse uit. Deze wordt gedaan door de client, waarmee wordt gecommuniceerd over het netwerk. Van deze client-applicatie wordt enkel de schil om de beeldanalyse geschreven. De beeldanalyse wordt door de opdrachtgever ontwikkeld en toegevoegd aan de client-applicatie.

2.6.2 Meetbaar

Als de opdracht is uitgevoerd, kan er een verbinding tussen de Android telefoon en een computer met de client-applicatie worden opgezet, waarbij de Android telefoon de lijnen van de afbeeldingen van de camera opstuurt naar de computer.

2.6.3 Acceptabel & Realistisch

Uit een initiële voorstudie is gebleken dat een applicatie die foto's analyseert haalbaar is met de beschikbare technische kennis en frameworks. Er zijn meerdere technieken en applicatie-frameworks beschikbaar om de benodigde componenten van de applicatie te ontwikkelen. De complexiteit van de te gebruiken technieken maakt dat niet iedere techniek even geschikt is voor implementatie. Om tot de beste werking van de applicatie te komen, zal er geëxperimenteerd en getest moeten worden welke techniek het meest geschikt is om in te passen in de applicatie. De keuze voor iedere gebruikte techniek is gemaakt in overleg met de opdrachtgever en aan de hand van berekeningen en waarnemingen.

Aangezien geen informatie over de beeldanalyse van de opdrachtgever beschikbaar is, is er geen eis om de beeldanalyse verder uit te ontwikkelen.

2.6.4 Tijdsgebonden

De ontwikkeling van de applicatie is uitgevoerd in 17 weken. Tijdens en na deze 17 weken is er een verslag tot stand gekomen over de ontwikkeling van dit project.

2.7 Wensen voor de applicatie

Er zijn voor de applicatie wensen van de opdrachtgever opgesteld. Deze wensen zijn tot stand gekomen op basis van gesprekken met de opdrachtgever:

Allereerst dient er ontwikkeld te worden voor Android. Er moet ontwikkeld worden voor een Android telefoon van het merk Wolfgang, een goedkope telefoon oorspronkelijk te koop bij de Aldi.

De resolutie van de afbeelding gemaakt door de camera van de Android telefoon moet hoog genoeg zijn voor succesvolle beeldanalyse. Het eindresultaat, de kopdikte, moet minimaal een nauwkeurigheid hebben van 0,1mm/pixel voor de diameter van de steel.

Er dient via het netwerk data gestuurd te kunnen worden, van de Android telefoon naar een computer. De data die gestuurd wordt, moet in ieder geval de lijnen van de afbeelding te bevatten die nodig zijn voor een succesvolle beeldanalyse. Dat zijn minimaal 9 lijnen. Er moet ingesteld kunnen worden welke van het totaal aantal aanwezige lijnen dat zijn.

Voor de netwerkcommunicatie dient een applicatie geschreven te worden in C voor de externe computer. Deze applicatie dient in C geschreven te worden gebruikmakend van de Dev-C++ IDE (specifiek versie 4.9.9.2) [2]. Het gaat hier om de client-applicatie.

Het is naast de in C geschreven client-applicatie eenvoudig om een client-applicatie voor de externe computer ook in Java te schrijven, aangezien de host-applicatie voor Android al in Java geschreven wordt. Dit kan onder andere handig zijn om testen uit te voeren.

Vervolgens is het nodig om een aantal lijnen van de afbeelding uit te lezen. Er is geen wens om de gehele afbeelding te analyseren. Indien mogelijk is het handig om wel een optie te hebben om de hele afbeelding van de camera op te vragen.

De snelheid van het maken van afbeeldingen hoeft niet hoog te zijn, aangezien het gaat om een prototype applicatie. De opdrachtgever stelt dat 1 afbeelding per seconde minimaal gewenst is voor deze applicatie.

Er dient gekeken te worden of de beeldanalyse ook lokaal op de Android telefoon zelf kan draaien. Daarvoor dient een *loadtest* uitgevoerd te worden. De loadtest omvat een beeldanalyse, waarbij het niet gaat om de functionaliteit, maar om het evenaren van de belasting van het proces.

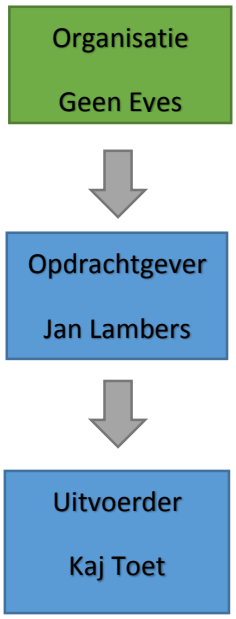
De beeldanalyse berekeningen worden niet verder uitontwikkeld naast de eerder genoemde loadtest. De ontwikkeling van de beeldanalyse valt buiten het bereik van dit onderzoek.

3 Aanpak / werkwijze

Dit hoofdstuk beschrijft de beschikbare methoden en technieken met de bijbehorende voor- en nadelen. Daarna wordt toegelicht welke werkwijze voor de ontwikkeling van de applicatie is toegepast.

3.1 Uitgangspunten

Om de applicatie te ontwikkelen is gewerkt in een klein team. Er is één ontwikkelaar en één opdrachtgever. De opdrachtgever is tevens de hoofdgebruiker van de applicatie.



Het is een nieuw te ontwikkelen applicatie met complexe technieken, zoals het in hoog tempo maken van hoge resolutie foto's om gebruikt te worden voor beeldverwerking. Er zal daarom geëxperimenteerd moeten worden met verschillende technieken om tot de beste oplossing te komen.

De gebruiker zal gedurende de ontwikkeling feedback geven bij de opgeleverde tussenproducten, waarbij tijdens het ontwikkelproces nieuwe eisen kunnen worden geformuleerd. Er is vooraf geen contract met vereisten opgesteld.

De applicatie is niet een onderdeel van een kritiek systeem. De veiligheid van de applicatie heeft daarmee geen prioriteit.

Om de complexiteit van de applicatie te beperken, kan het project onderverdeeld worden in meerdere kleine problemen, die iteratief aangepakt zullen worden.

3.2 Methode en technieken

Aan de hand van bovenstaande gegevens zijn een set van project technieken en methodes voorgeselecteerd. Deze worden in de volgende tabel vergeleken met een set met eisen op basis van de gestelde uitgangspunten. Op basis van een eenvoudige score zijn de eisen getoetst met de methoden. Met deze tabel worden deze technieken en hun toepasbaarheid kort geëvalueerd.

TABEL 1: VERGELIJKING METHODIEKEN, LEGENDA: + JA, +/- OM HET EVEN, - NEE

	Eis		Waterval	Rapid Application Development (RAD)	Spiral	Incremental	Prototyping	RUP (Rational Unified Proces)	Agile	SCRUM
Geschikt voor een een/tweepersoons team?	+		+	+	+	-	+	+	+	-
Geschikt voor kort durend project?	+		+	+	-	-	+	+	+	+
Is het een kritieke applicatie?	-		+/-	-	+	+/-	+/-	+/-	+/-	+/-
Staan de eisen vooraf al vast?	-		+	+	+	+/-	-	+	-	+/-
Levert het project een stabiel en voorspelbaar systeem op?	-		+	+	+	+	-	+	+/-	+/-
Is er een contract tussen ontwikkelaar en klant als basis voor communicatie?	-		+/-	+/-	+	+/-	-	+/-	+/-	+/-
Kan het project in sequentiële stappen worden uitgevoerd?	-		+	+/-	+/-	+	-	+	-	+/-
Kan er geëxperimenteerd worden met technieken?	+		-	+	-	+	+	+	+	+/-
Is Risk assessment belangrijk?	-		+/-	-	+	-	-	+/-	+/-	+/-
Is de gebruiker actief betrokken bij het ontwikkelen?	+		-	+	+/-	+/-	+	+/-	+/-	+/-
Planning mogelijk voor meerdere kleine producten?	+		-	+/-	+	+	+	-	+	+/-
11 eisen			3,5/ 11	7,5/ 11	3/ 11	4,5/ 11	10,5/ 11	5/ 11	9/ 11	5,5/ 11

Er is een iteratief proces nodig, waarbij tussentijds onderdelen opgeleverd worden. De evaluatie leidt tot de gewenste aanpassingen.

Aangezien iteratief gewerkt wordt, kan de Watervalmethode niet gebruikt worden. De eisen zijn vooraf grotendeels onbekend en worden gaandeweg in het project duidelijker.

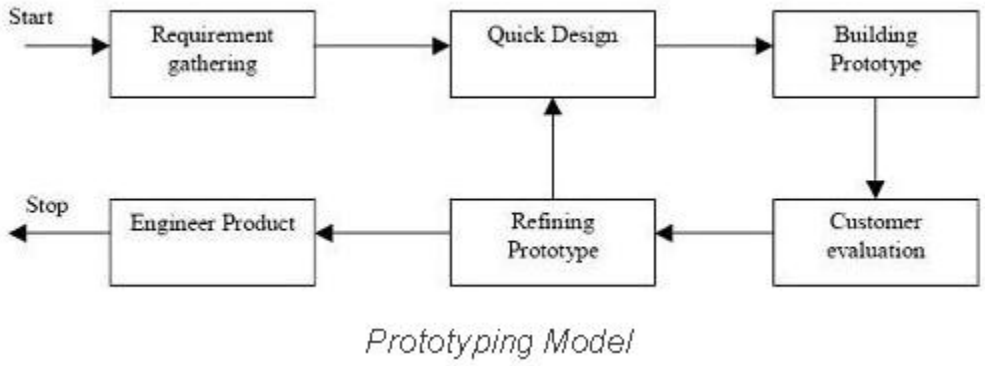
Rapid Application Development wordt vooral toepast op projecten die een vooraf voorspelbaar systeem opleveren. Tevens is het net als de Watervalmethode bij Rapid Application Development belangrijk dat de eisen vooraf gespecificeerd kunnen worden. Rapid Application Development is toepasbaar als de technische eisen gemakkelijk haalbaar zijn. In dit project zijn de technische eisen niet vooraf makkelijk te specificeren.

De Incremental methode wordt gebruikt bij grotere projecten met een langere duur. Informatie systemen op het web gebruiken deze methode vooral.

Risk Assessment, risico inschatting, kan handig zijn, maar is voornamelijk bedoeld voor projecten met strenge eisen, zoals bij Realtime, tijdskritieke, systemen en systemen met een prioriteit voor veiligheid. In het geval van dit project zal een te hoge focus op Risk Assessment in de weg staan van het behalen van het doel. Risk Assessment is vaak ongewenst als er zuinig moet worden omgegaan met de beschikbare bronnen (processorkracht, tijd, etc.). De Spiral methode is vooral georiënteerd op Risk Assessment.

SCRUM wordt niet gebruikt, omdat SCRUM niet bedoeld is voor kleinere teams. Er wordt wel net als bij SCRUM gewerkt met kleinere, beheersbare taken waarbij iedere taak een prioriteit toegewezen. Deze opdrachten worden gaandeweg vastgesteld.

Prototyping kan worden gebruikt als de doelen en eisen niet compleet helder zijn en tussentijds kunnen veranderen. De gebruiker werkt samen met de ontwikkelaar. Bij het bouwen van de prototypes, wordt aangenomen dat het prototype misschien waardeloos is en niet gebruikt dient te worden. De prototypes kunnen daarna eventueel verder uitgewerkt en geoptimaliseerd worden, maar dat is niet een vereiste. Het project kan worden opgedeeld in meerdere onderdelen, die individuveel ontwikkeld dienen te worden.



FIGUUR 3 DE STAPPEN IN DE PROTOTYPING METHODE

Agile software ontwikkeling omvat het tussentijds evalueren van producten, waarbij verandering in de eisen en wensen kan ontstaan. Er wordt veel met de gebruiker overlegd.

Concluderend aan de tabel kan gesteld worden dat de Prototyping methode en Agile software ontwikkeling geschikt zijn voor dit project. Aangezien in dit project meerdere prototypes gemaakt moeten worden en de eisen niet vooraf compleet helder zijn, is er gekozen om de Prototyping methode te gebruiken. Daarbij wordt daarnaast Agile te werk gegaan, waarbij tussentijds prototypes beoordeeld worden en resultaten worden teruggespeeld aan de opdrachtgever.

3.3 Planning

Allereerst zijn er enkele bekende factoren in de te ontwikkelen applicatie. Zo dient er voor Android ontwikkeld te worden en wordt er gebruik gemaakt van een camera, OpenGL en netwerkcommunicatie.

Er is tevens al code voor o.a. de beeldanalyse geschreven en aanwezig, welke bestudeerd dient te worden. Deze code kan misschien van belang zijn bij het ontwikkelen van de Android applicatie.

Tijdens de ontwikkeling worden eerst de kritieke delen bepaald. De applicatie wordt opgedeeld in meerdere, losse componenten. De losse componenten worden uitgewerkt en getest.

Vervolgens worden de componenten samengevoegd in een werkend geheel. Hierbij kunnen nieuwe problemen ontstaan, welke opgelost dienen te worden.

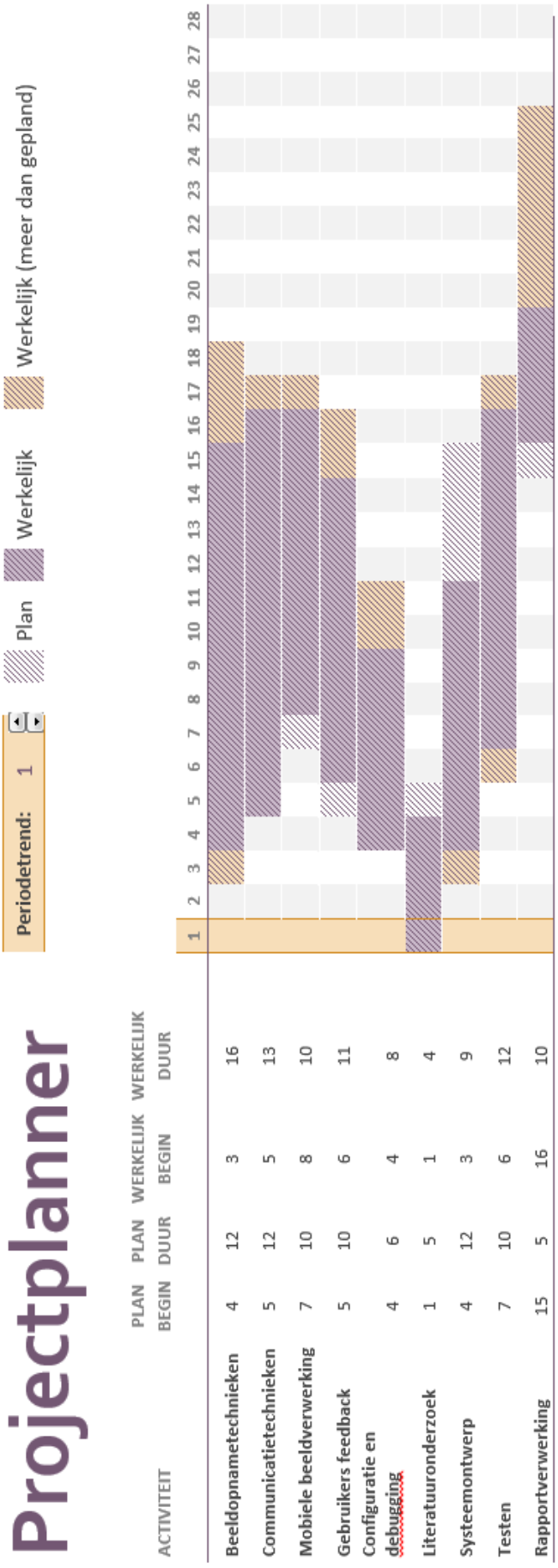
Het eindproduct is de Android applicatie. Er is documentatie gemaakt ter ondersteuning van de Android applicatie. Deze documentatie bestaat uit een systeem specificatie en een interface specificatie voor de netwerk communicatie.

De ontwikkeling van de applicatie duurt 17 weken van 40 werkuren per week. Na de ontwikkeling van de applicatie is het verslag van het project opgesteld.

De ontwikkeling van de applicatie is als volgt opgedeeld:

- Na 4 weken (25% van de tijd) zal een bezoek bij de lector dhr. Lambers met de begeleidend examinator plaatsvinden.
- Na 7 á 8 weken (45% van de tijd) zal een voortgangsverslag aan de begeleidend examinator worden gestuurd.
- Na ongeveer 10 weken (60% van de tijd) zal het concept afstudeer dossier met de begeleidend examinator worden besproken.
- Na ongeveer 13 weken wordt een assessment afgelegd met de begeleidend examinator en de tweede examinator.

Het opstellen van het verslag heeft plaatsgevonden in de weken na de ontwikkeling: week 15 t/m 25. In figuur 4 is de projectplanner is weergegeven.



FIGUUR 4 DE GANTT-CHART VOOR DIT PROJECT.

4 Literatuuronderzoek

In het hoofdstuk literatuuronderzoek worden onderzochte en gebruikte technieken toegelicht. Omdat er bij de ontwikkeling van de applicatie gebruik is gemaakt van prototyping, is de informatie voor dit hoofdstuk deels verzameld tijdens het ontwikkelen van de software.

- host/camerazijde
  - Android besturingssysteem
  - Java programmeertaal
- grafische weergave
  - OpenGL d.m.v. het LibGDX framework
- beeldanalyse
  - gebruikte voorbeeldcode in Python programmeertaal
  - berekeningen inzage
- client-applicaties
  - één geschreven in de Java programmeertaal gebruikmakend van LibGDX
  - één geschreven in C++ gebruikmakend van Dev-C++

4.1 Analyseren code en wiskundige formules

Oorspronkelijk was het de bedoeling om het algoritme van de opdrachtgever dat gebruikt wordt voor de beeldanalyse te vertalen naar de Java programmeertaal. Gedurende de ontwikkeling werd echter duidelijk dat het algoritme van de beeldanalyse nog niet voldoende ontwikkeld was.

Uiteindelijk werd een versie van de beeldanalyse van het bedrijfsproject “Contactloze diktebepaling van tomatenstengels door segmentatie en lijndetectie” [3] verkregen en gebruikt als basis om een beeldanalyse algoritme te schrijven ten behoeve van een *loadtest.*

4.2 Java Programmeertaal

Om applicaties te ontwikkelen voor Android, is het handig om te programmeren in Java, aangezien Android een Java Virtual Machine draait.

De Java Virtual Machine, oftewel JVM, is een abstracte computer die gecompileerde Java programma’s virtueel draait. De JVM wordt virtueel uitgevoerd binnen het hardware platform en besturingssysteem. Alle Java programma’s worden gecompileerd voor de JVM door middel van bytecode. Daarom moet de JVM geïmplementeerd worden op een bepaald platform voordat de gecompileerde Java programma’s kunnen draaien op dat systeem. [4]

Java kan tevens gebruikt worden op andere platformen dan Android, zoals bijvoorbeeld iOS (d.m.v. RoboVM[1]), Windows, Linux of Mac OSX. Java is een object georiënteerde programmeertaal. Java ondersteunt garbage collection[2], maar onder Android (d.m.v. de Dalvik VM) wordt het niet aangeraden om hier veel gebruik van te maken, omdat dit voor vertraging van het systeem kan zorgen.

4.3 Android

De software voor de contactloze kopdiktemer wordt ontwikkeld voor Android, aangezien de Wolfgang telefoon enkel het Android besturingssysteem kan draaien. Bovendien is Android erg ontwikkelaarsvriendelijk en is er geen licentie nodig om te ontwikkelen voor het besturingssysteem.

[1] RoboVM vertaald Java bytecode naar native ARM of x86 code voor gebruik in iOS apps. [26]
[2] Garbage collection is een vorm van automatisch geheugenbeheer (memory management).

Android is een mobiel besturingssysteem gebaseerd op de Linux kernel en momenteel ontwikkeld door Google. Met een User Interface gebaseerd op directe manipulatie, is Android voornamelijk gericht op gebruik door middel van een aanraakgevoelig scherm. [5]

In de te ontwikkelen applicatie dient gebruik te worden gemaakt van bepaalde systeemafhankelijke functies van Android. Uiteindelijk is de cameratechniek uitgewerkt door middel van Android platformafhankelijke code en de Wifi hotspot functionaliteit. Daarnaast is ook gekeken naar platformafhankelijke JPEG-decompressie, maar dat bleek uiteindelijk niet gunstig.

4.4 OpenGL

Om te laten zien hoe de plantensteel georiënteerd en gepositioneerd staat, wordt er gebruik gemaakt van OpenGL visualisatie. OpenGL wordt ondersteund door het Android platform in de vorm van OpenGL ES, een mobiele variant van OpenGL. In dit geval is gebruik gemaakt van OpenGL ES 2.0; een ruim ondersteunde versie met veel mogelijkheden (voor b.v. optimalisatie).

Op de desktop wordt OpenGL ES in theorie geëmuleerd. OpenGL ES is voornamelijk een subset van het gewone OpenGL, die aanwezig is op desktop platformen. Omdat een OpenGL ES een subset van de op de desktop beschikbare OpenGL, kan er indirect ontwikkeld worden voor OpenGL ES.

Als een software interface voor grafische hardware is het hoofddoel van OpenGL om twee en drie dimensionele objecten te tekenen in een framebuffer. Deze objecten zijn opgebouwd uit vertices (die geometrische objecten opmaken) of pixels (die afbeeldingen opmaken). OpenGL voert verscheidene rekentaken uit om deze te converteren naar pixels om het uiteindelijke beeld te vormen. [6]

4.5 LibGDX Framework

Om het makkelijker te maken om de software voor de contactloze kopdiktemeter te ontwikkelen, wordt er gebruik gemaakt van het LibGDX framework.

LibGDX is een lichtgewicht, bekend en veelgebruikt framework om 3D applicaties te maken in Java. LibGDX maakt het mogelijk om een Java-applicatie snel en gemakkelijk te porten naar o.a. iOS, Android, desktop en HTML/WebGL.

Het framework is een kleine schil met de meest essentiële code om een op OpenGL gebaseerd programma te maken d.m.v. de Java programmeertaal. Het merendeel van de gebruikte functies is abstract opgezet, waardoor er onafhankelijk van platformen (PC, Android, WebGL) ontwikkeld kan worden.

LibGDX staat het toe om low-level te programmeren en direct toegang te geven tot file systems, invoerapparatuur, geluidsapparaten en OpenGL via OpenGL ES 2.0 of 3.0.

In het LibGDX framework zijn een set van API's ingebouwd voor het tekenen van sprites en tekst, het bouwen van User Interfaces, het afspelen van geluidseffecten en muziek, lineaire algebra en trigonometrie berekeningen, het lezen van JSON en XML en meer. [7]

4.6 Dev-C++

In het project is er de eis (eis M.2) om ondersteuning te bieden voor connectiviteit met het Dev-C++ (specifiek versie 4.9.9.2) [2] IDE. Er is onderzoek gedaan naar deze IDE en vervolgens is de functionaliteit van de client-applicatie vertaald van Java naar C++ gebruikmakend van Dev-C++.

Bloodshed Dev-C++ is een IDE (Integrated Development Environment) voor de C en C++ programmeertaal. Het gebruikt een MinGW port van GCC (GNU Compiler Collection). Dev-C++ kan ook gebruikt worden in combinatie met Cygwin en andere op GCC gebaseerde compilers.

4.7 Componenten

De verschillende componenten van de Android applicatie zijn:

- de camera,
- de invoer aan de hand van de User Interface,
- de beeldanalyse en
- de uitvoer d.m.v. o.a. OpenGL op het scherm.

De camera kan zowel hoge resolutie foto's maken in een laag tempo, als video in een lagere resolutie en een hoger tempo, maar niet beide tegelijkertijd. In dit project wordt gebruik gemaakt van hoge resolutie foto’s.

De GUI geeft de mogelijkheid voor de gebruiker om bv. instellingen te wijzigen. De GUI werkt systeemonafhankelijk. De GUI laat tevens informatie van de berekende uitkomsten zien.

De uitvoer op het beeld bevat een 3D representatie van de plantensteel d.m.v. OpenGL en een directe weergave van de gemaakte camerabeelden. Deze twee beelden kunnen naast elkaar of over elkaar worden weergegeven.

De beeldanalyse omvat het hart van de applicatie. De beeldanalyse zorgt namelijk voor de berekening van de positie, dikte en oriëntatie van de steel. De beeldanalyse kan lokaal op de Android telefoon zelf gebeuren of extern op een ander apparaat, zoals bijvoorbeeld op een laptop. In het laatste geval moeten instructies en data over het netwerk worden verzonden.

Bij netwerkcommunicatie moet er worden gekeken naar de type van transmissie (UDP, TCP), de benodigde snelheid en de onoverkomelijke vertraging die optreedt.

5 Eisen aan de applicatie

Er zijn eisen gesteld aan de software voor de contactloze kopdiktemeter. Deze eisen zijn verzameld conform de MoSCoW methodologie. Hieronder wordt de MoScoW methodologie uitgewerkt en staan de eisen uitgewerkt volgens de MoSCoW indeling.

De MoSCoW-methode is een wijze van prioriteiten stellen in onder meer de software engineering. De eisen aan het resultaat van een project worden ermee ingedeeld. Het is een afkorting, waarvan de letters staan voor:

- M - must haves: deze eisen (requirements) moeten in het eindresultaat terugkomen, zonder deze eisen is het product niet bruikbaar;
- S - should haves: deze eisen zijn zeer gewenst, maar zonder is het product wel bruikbaar;
- C - could haves: deze eisen zullen alleen aan bod komen als er tijd genoeg is;
- W - won't haves (ook wel would haves genoemd): deze eisen zullen in dit project niet aan bod komen maar kunnen in de toekomst, bij een vervolgproject, interessant zijn.

De kleine letters 'o' in de afkorting hebben geen betekenis, maar maken de afkorting makkelijker te onthouden. [8]

De eisen zijn verzameld voor en tijdens het ontwikkelen van de software voor de contactloze kopdiktemer.

In het hoofdstuk Testen op pagina 50 wordt er gekeken of er aan de eisen is voldaan.

Must have	
M.1	Ondersteuning van de Android telefoon van het merk Wolfgang.
M.2	Ondersteuning voor de Dev-C++ ontwikkelomgeving.
M.3	Verzending van (een variabel in te stellen aantal) horizontale lijnen van de gemaakte afbeelding over het netwerk; minimaal 9 lijnen.
M.4	Beschrijving van de netwerkcommunicatie.
M.5	De Android-applicatie gebruik laten maken van threading.
M.6	De netwerkcommunicatie asynchroon laten verlopen o.a. d.m.v. meerdere communicatiekanalen.
M.7	Opstellen en uitvoeren van berekeningen om de uiteindelijke nauwkeurigheid in te schatten van de beeldanalyse.
M.8	Een nauwkeurigheid van 0,1mm/pixel voor de afbeelding van de camera.
M.9	Een snelheid van minstens 1 afbeelding per seconde.

Should have	
S.1	De hele afbeelding van de camera uitlezen en verzenden over het netwerk.
S.2	Een manier om het uitlezen van de lijnen te draaien voor verzending (horizontaal/verticaal)
S.3	Het mogelijk maken om netwerkadressen en porten in te stellen voor de netwerkcommunicatie.
S.4	Toegang tot de accelerometer data van de telefoon.
S.5	Visualisatie van de afbeelding gemaakt door de camera op het scherm van de Android telefoon.
S.6	Visualisatie van de positie, oriëntatie en dikte van de steel in 3D met OpenGL op het scherm van de Android telefoon.
S.7	Tekstuele informatie getoond op het scherm van de Android telefoon, zoals netwerkadressen.
S.8	Debug-informatie die wordt gegeneerd tijdens de werking van de applicatie.
S.9	Een testroutine om de mogelijke nauwkeurigheid van de beeldanalyse te berekenen.
S.10	Het mogelijk maken om optioneel de flitser van de camera te gebruiken.

Could have	
C.1	Een loadtest om te kijken of de beeldanalyse eventueel op de Android telefoon zelf kan draaien. Deze loadtest omvat een basis vorm van de beeldanalyse.
C.2	Een methode om automatisch een Wifi hotspot op te zetten op de Android telefoon.
C.3	Ondersteuning voor een verbinding met USB i.p.v. Wifi.

Won't have	
W.1	Een volledige, complete beeldanalyse voor gebruik met de contactloze kopdiktemeter module.

6 Onderzoek, prototyping en uitwerking

Er worden meerdere prototypes gemaakt, omdat vooraf onzeker is welke technieken kunnen voldoen aan de eisen.

De prototypes omvatten de werking van:

- de beeldopnametechnieken
- de communicatietechnieken
- de mobiele beeldverwerking
- de gebruikers feedback
- configuratie en debugging

Voor de prototypes worden eerst ontwerpen gecreëerd d.m.v. UML en eventueel schetsen.

Als de prototypes goed blijken te werken worden ze verder uitgewerkt. Indien de prototypes ongunstig blijken, worden er nieuwe prototypes ontwikkeld. Dit proces gaat in samenwerking met de klant.

Nadat de prototypes per onderdeel zijn ontwikkeld worden ze samengevoegd in een werkend geheel.

Vervolgens wordt documentatie ontwikkeld met uitleg over de werking van de Android applicatie in samenwerking met de client-applicatie.

De Android applicatie wordt getest in samenwerking met de client-applicatie om de snelheid en vertraging van de oplevering en analyse van de camerabeelden te bepalen. De hieruit volgende testresultaten zijn in de bijlage opgenomen.

Naast de vereiste onderzoeken wordt er een loadtest op de Android telefoon zelf uitgevoerd. Deze loadtest wijst uit of de beeldanalyse zelfstandig op de Android telefoon kan draaien.

6.1 Beeldopnametechnieken

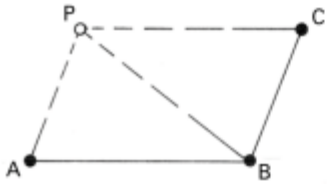
Een belangrijk onderdeel van deze afstudeeropdracht is om te bepalen hoe de beste kwaliteit behaald kan worden voor het streamen van informatie aan de hand van de wensen van de opdrachtgever.

6.1.1 Achterwaartse insnijdingsmethode

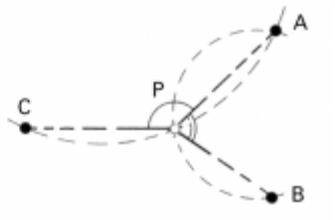
Voor het berekenen van de positie van de steel wordt de achterwaartse insnijdingsmethode gebruikt in de beeldanalyse van de opdrachtgever. Om dat te realiseren wordt de Android telefoon op een module met spiegels vastgeklemd.

Voorwaartse en achterwaartse insnijding zijn vormen van triangulatie die, gebruikmakend van bekende punten, een enkel nieuw punt bepalen.

Bij voorwaartse insnijding wordt hoekmeting verricht in minimaal twee bekende punten. In onderstaande abeelding bij figuur 2.5 zijn dat punt A, B en C. Bij achterwaartse insnijding wordt slechts hoekmeting verricht in het te bepalen punt d.m.v. drie bekende punten.



Figuur 2.5. Voorwaartse insnijding.



Figuur 2.6. Achterwaartse insnijding.

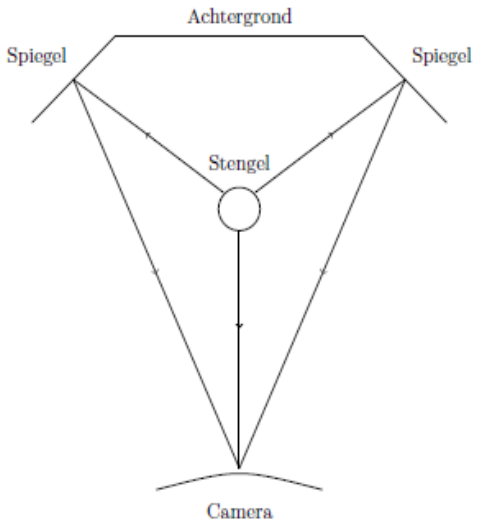
FIGUUR 5 VOORWAARTSE EN ACHTERWAARTSE INSNIJDING

Als in figuur 2.6 de hoek APB gemeten, dan bepaalt deze hoek een cirkel door A, B en P en geeft hiermee een meetkundige plaats voor P. Meting van de hoek CPA bepaalt een tweede meetkundige plaats. De twee meetkundige plaatsen kunnen gebruikt worden om de ligging van punt P te berekenen. Een eis is wel dat P niet op één cirkel met A, B en C ligt. [9]

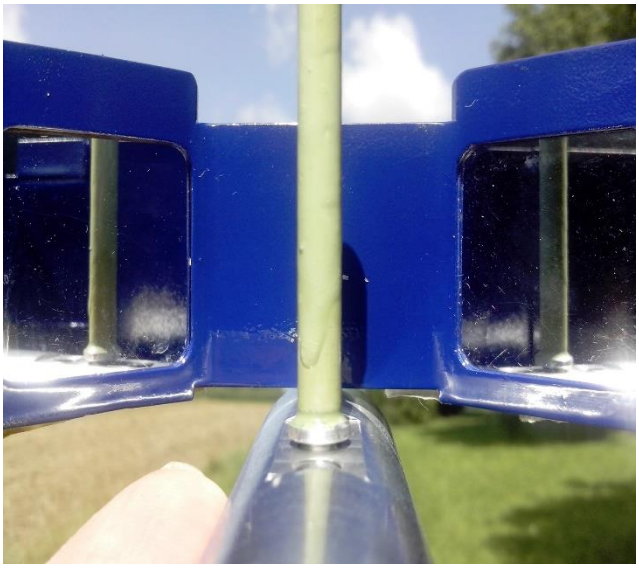
Achterwaartse insnijding is een methode die veel toegepast wordt in landmeetkunde, zoals b.v. bij het meten van boringen. In het rapport “Drie dimensionele metingen van horizontale boringen” [10] wordt de achterwaartse insnijdingmethode toegepast op horizontale boringen. Bij dit onderzoek werd ook duidelijk dat de oriëntatie van de boring de meting kan veranderen. Met de oriëntatie moet dus rekening gehouden worden bij het bepalen van de positie.

Met achterwaartse insnijding zou in theorie de kopdikte, de positie en de oriëntatie van de steel uit de foto te berekenen moeten kunnen zijn.

De nauwkeurigheid van de meting is afhankelijk van de nauwkeurigheid van de referentiepunten.



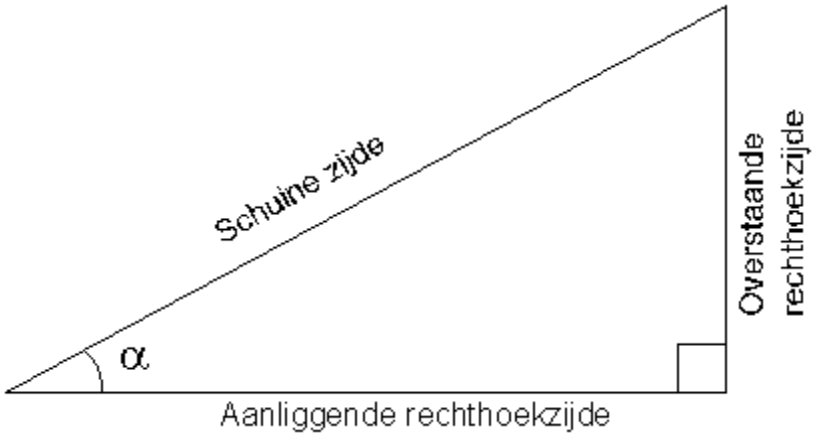
FIGUUR 6 EEN BOVENAANZICHT VAN DE KOPDIKTEMETER



FIGUUR 7 EEN AFBEELDING GEMAAKT DOOR DE CAMERA VAN DE TELEFOON

6.1.2 Berekening resolutievereiste

De onzekerheid van de meting van de diameter van de steel mag maximaal 0,1 mm/pixel bedragen (eis M.8). De gemiddelde openingshoek van de camera van de Wolfgang telefoon (eis M.1) is 57,6 graden. Het gemiddelde meetbereik is 3 tot 15 mm. Het optimale voorwerppunt ligt op 80mm. Met de afstand van 8cm en de helft van de openingshoek van de camera van 65 graden kan d.m.v. de tangens de breedte van het beeld gemeten worden.



FIGUUR 8 DE AANLIGGENDE ZIJDE (AFSTAND TOT STEEL) EN DE (OPENINGS)HOEK ZIJN BEKEND. DE OVERSTAANDE ZIJDE WORDT BEREKEND.

De functie die gebruikt wordt om de nauwkeurigheid te berekenen is (eis M.7):

> tan(openingshoek camera / 2) * afstand tot steel (cm) * 2 / resolutiebreedte (pixels) * 10 = nauwkeurigheid (mm)

Voor het gebruik van de module van de contactloze kopdiktemeter is de afstand tot de steel ongeveer 8 tot 13,5 cm.

Voor een camera openingshoek van 65 graden en een afstand tot de steel van 8cm, is de volgende berekening van toepassing.

> De breedte van de afbeelding op 8 cm afstand is: tan(57,6 / 2 graden) * 80mm * 2 = 87,96 mm

Gegeven dat de maximale resolutie van een foto 3264x2448 bedraagt, is de gemeten nauwkeurigheid 87,96 mm / 3264 pixels = 0.027 mm/pixel. De afstand kan 3 keer zo groot worden tot het plafond van 0,1 mm/pixel nauwkeurigheid bereikt is bij een afbeelding van 3264 pixels breed en een openingsboek van 65 graden.

Voor een foto op een maximale resolutie van 3264x2448 is de nauwkeurigheid op een afstand van 8cm, 0,027 mm/pixel. Daarmee wordt nog steeds voldaan aan de eis van maximaal 0,1mm/pixel. Op een afstand van 13,5 cm is dat 0,045 mm/pixel. Op een afstand van 2 x 13,5 cm kan een foto dus nog steeds voldoen aan de eis van een nauwkeurigheid van 0,1mm/pixel. Dit is een meting onder optimale condities. In de praktijk staat de focus van de camera op oneindig, wat leidt tot een beeld wat altijd een beetje uit focus is, wat het moeilijker maakt om de meting te doen.

> In het hoofdstuk Testen is een test gedaan om de nauwkeurigheid te bewijzen aan de hand van een foto d.m.v. de contactloze kopdiktemeter. Zie hiervoor pagina 51.

Voor video op een maximale resolutie van 1280x720 is de nauwkeurigheid op een afstand van 8cm, 0.068 mm/pixel. Video voldoet op een afstand van 13,5cm niet meer aan de eis met 0.115 mm/pixel.

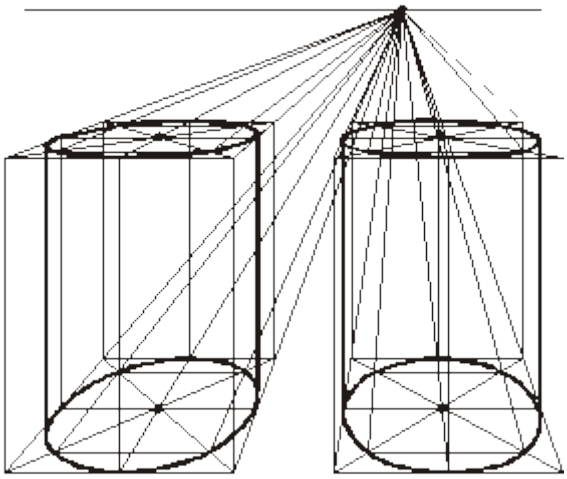
Voor de Wolfgang telefoon is geen Full HD resolutie van 1920x1080 voor video opname beschikbaar. In het geval dat 1920x1080 als resolutie wordt gebruikt wordt wel voldaan aan de eis van 0,1mm/pixel met 0,046mm/pixel voor 8cm afstand en 0,077mm/pixel voor 13,5 cm afstand.

Het is dus niet mogelijk om voor dit doel video-opname te gebruiken. In de toekomst is wellicht een nog hogere nauwkeurigheid gewenst.

6.1.3 Overige invloeden op resolutievereiste

Er worden drie metingen gedaan: 2 keer wordt de steel gemeten via de spiegels en 1 keer zonder een spiegel. Deze drie beelden dienen om door middel van de achterwaartse insnijdingsmethode de positie van de cilindervorm t.o.v. het focuspunt te bepalen. In theorie kan het gemiddelde van deze drie metingen ook gebruikt worden om een nauwkeuriger meting van de diameter te krijgen.

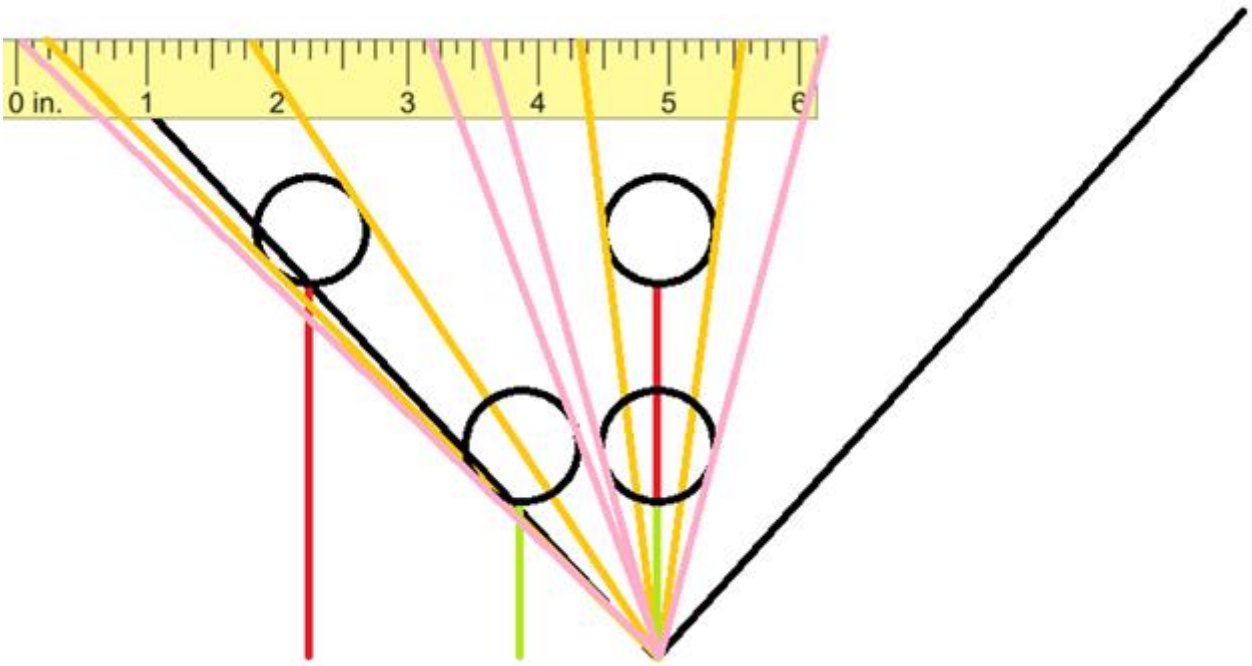
Er treedt perspectivische verkorting op. Deze perspectivische verkorting zorgt er voor dat objecten in de hoek van de camera “uitgerekt” zijn. Dit kan er voor zorgen dat een object groter lijkt in de beeldanalyse als er geen rekening wordt gehouden met de verkorting. In theorie wordt de nauwkeurigheid groter naarmate er meer pixels beschikbaar zijn voor de analyse.



FIGUUR 9 VOORBEELD VAN PERSPECTIVISCHE VERKORTING. DE LINKER CILINDER NEEMT MEER PIXELS OP HET SCHERM IN BESLAG IN DE BREEDTE, AANGEZIEN DEZE VERDER IN DE HOEK VAN DE CAMERA STAAT.

Op onderstaande afbeelding is de perspectivische verkorting te zien.

Op de cirkel rechts onderaan is ook occlusie te zien: de cirkel lijkt kleiner dan deze werkelijk is, omdat het beeld vanuit het camerapunt enigszins vertekend is. De occlusie wordt kleiner als het voorwerp verder van de camera is.



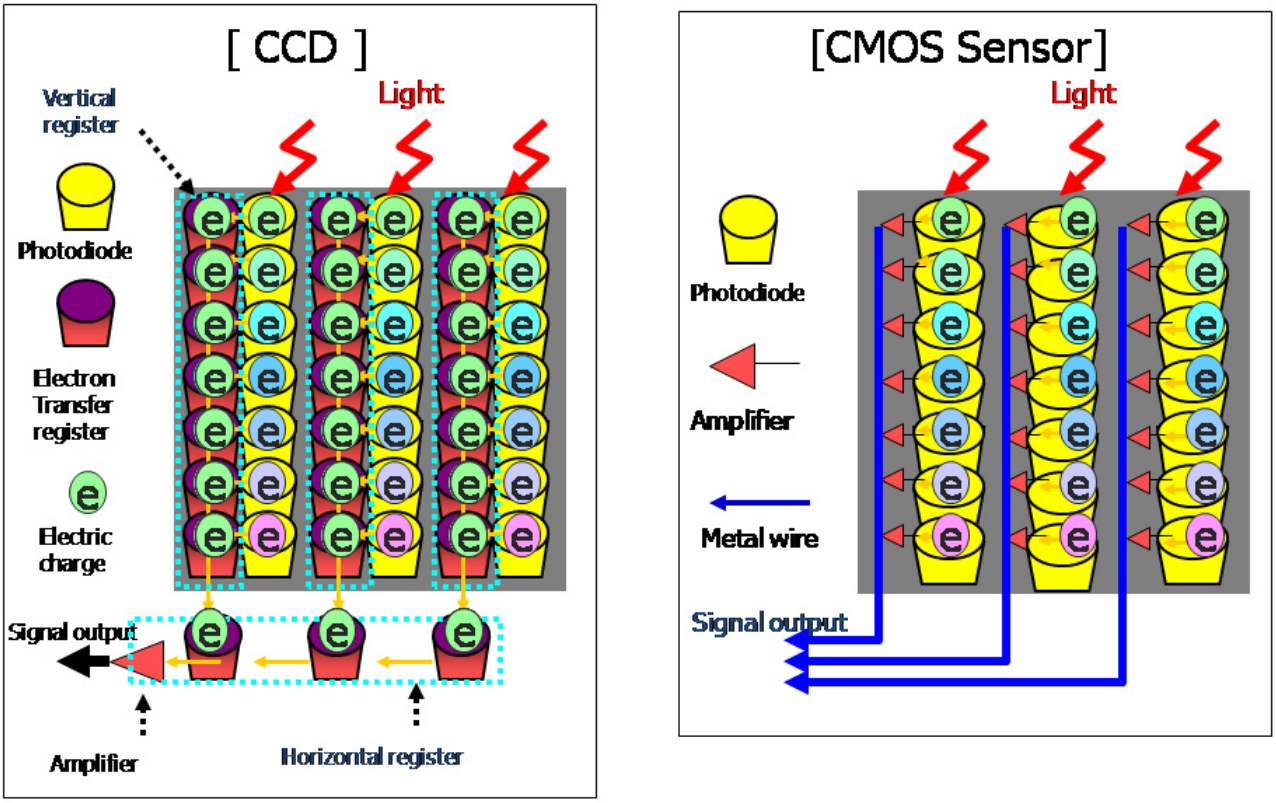
FIGUUR 10 DE LINIAAL STELT HET BEELDSCHERM VOOR MET IEDERE MILLIMETER EEN PIXEL. DE CIRKELS AAN DE ZIJKANT NEMEN MEER RUIMTE OP DE LINIAAL IN BESLAG.

6.1.4 Uitlezen van enkele lijnen

Er zijn maar enkele lijnen van de 3264x2448 pixels van de afbeelding echt nodig voor de beeldanalyse (eis M.3). Het heeft de voorkeur om lijn per lijn pixels uit te lezen in plaats van de complete foto met het maximaal aantal beschikbare lijnen.

Het uitlezen van een deel van een foto (bv. een enkele lijn van pixels) is niet mogelijk op de Wolfgang telefoon en ook niet op de meeste andere Android telefoons. In theorie zou het wel mogelijk moeten zijn aangezien het gros van de (Android) telefoons een zogenaamde CMOS-chip gebruikt voor het maken van foto's. Maar Android geeft hier op het moment geen ondersteuning voor.

CMOS chips gebruiken transistoren bij iedere pixel om de lading te vervoeren over traditionele bedrading. Hierdoor wordt iedere pixel apart behandeld. Traditionele fabricage processen zijn gewend om CMOS te produceren. Het is hetzelfde als het produceren van microchips. Omdat ze eenvoudiger zijn om te produceren, zijn CMOS sensoren goedkoper te maken dan CCD sensoren. [11]



FIGUUR 11 IN THEORIE KUNNEN DE ELEKTRONEN INDIVIDUEEL UITGELEZEN WORDEN BIJ EEN CMOS SENSOR. BIJ EEN CCD SENSOR WORDEN ALLE ELEKTRONEN NAAR DE ZIJKANT GELOKT ALS DEZE UITGELEZEN WORDEN.

6.1.5 Fotostream

De enige mogelijkheid die overblijft, is om een zogenaamde constante serie van complete, hoge resolutie foto's te maken, oftewel een fotostream, waarbij zo snel mogelijk achter elkaar foto's worden gemaakt.

6.1.6 JPEG compressie

Dit heeft echter nadelen. Zo kan maar ongeveer één keer per seconde een foto gemaakt worden in hoge resolutie. Op de meeste (Android) telefoons wordt de foto opgeleverd in het JPEG formaat. Een nadeel is dat JPEG een lossy compressiemethode gebruikt, oftewel de gedecomprimeerde data is niet zo nauwkeurig als de data voor compressie. Er treedt informatieverlies op, zoals b.v. artefacten of kleurverlies.

Wie bekend is met JPEG afbeeldingen, kent de onregelmatigheden bekend als compressie artifacten die te zien zijn in JPEG afbeeldingen. Deze artifacten kunnen gezien worden in de vorm van ruis om randen met veel contrast of blokkerige afbeeldingen. Deze worden veroorzaakt door de quantisatie stap van het JPEG algoritme. Deze artifacten kunnen verminderd worden door geen of een lager niveau van compressie toe te passen. [12]

Een tweede, belangrijker, nadeel van het JPEG formaat is dat het niet mogelijk is een enkele lijn van pixels van de afbeelding uit te lezen. De foto moet eerst gedecomprimeerd worden of in zijn geheel opgestuurd worden over het netwerk en gedecomprimeerd worden op de externe computer.

Als de foto in zijn geheel, als JPEG afbeelding gecomprimeerd, verstuurd wordt over het netwerk zorgt dit nog steeds voor een grote vertraging. Het versturen van foto's gaat dan veel trager en er worden minder foto's per minuut verzonden.

Een betere methode is daarom om de JPEG afbeelding eerst te decomprimeren op de Android telefoon. Vervolgens worden enkel de benodigde lijnen van de afbeelding verstuurd. Dit scheelt aanmerkelijk in bandbreedte. De decompressie neemt echter tijd in beslag, wat resulteert in vertraging. Tevens is het batterijverbruik hoger en kunnen het aantal gemaakte foto's per minuut gelimiteerd worden door de JPEG decompressie.

6.1.7 RAW formaat

Nieuwere telefoons (met Android 5.0) kunnen soms foto's maken in het RAW formaat (eis W.1). Het RAW formaat past geen compressie toe. Hiermee zou geen decompressie nodig zijn op de telefoon en dit zou een goede uitkomst zijn voor deze opdracht. De iPhone en iPad hebben het equivalente TIFF formaat.

6.1.8 Beeldanalyse opsplitsen

Een andere methode is om het beeldanalyse te splitsen in twee delen. Een deel van de beeldanalyse gebruikt een videostream met een lagere resolutie, bijvoorbeeld om de positie van de steel te bepalen. Vervolgens wordt, bijvoorbeeld pas als de positie correct is, een hoge resolutie foto gemaakt en verstuurd, waarmee de uiteindelijke plantensteeldikte wordt berekend.

In de bovenstaande methode dient opgemerkt te worden dat het nodig is dat de CMOS-chip genoeg tijd heeft om beelden op te vangen en te verwerken. In één van de testen startte een videostream in de tussenpoos tussen foto's van een fotostream in. De CMOS-chip had hierbij te weinig tijd om de beelden op te vangen en te verwerken, waardoor de beelden donker en onbruikbaar werden. Later is gebleken dat dit wel goed mogelijk is, maar met een andere aanpak.

6.1.9 Videostream met gehele beelden

Een andere mogelijkheid is het gebruik van een videostream. In dat geval worden bijvoorbeeld Android specifieke functies gebruikt om het hele beeld te comprimeren en te verzenden. Het comprimeren van de beelden kost veel tijd, waardoor vertraging optreedt (eis M.9). Tevens kunnen artifacten optreden bij hevige compressie, wat een goede beeldanalyse snel in de weg kan zitten.

Feature	Apple	Adobe	MS IIS	MPEG-DASH
On-Demand & Live	Yes	Yes	Yes	Yes
Adaptive bitrates	Yes	Yes	Yes	Yes
Delivery Protocol	HTTP	HTTP	HTTP	HTTP
Origin Server	Web Server	Adobe Media Server	MS IIS	Web Server
Media Container	MP2 TS	MP4-part 14, FLV	MP4-part 14	MP2 TS, Fragmented MP4
Video Codecs	H.264 Baseline Level	H.264	Agnostic	H.264, SVC, Multiview Coding,

				MPEG 4 AAC
Default Segment Duration	10 seconds	2 seconds	4 seconds	Flexible
End-to-End Latency	30 seconds	6 seconds	>1.5 seconds	Flexible
File Type on Server	Fragmented	Contiguous	Contiguous	Fragmented
Client Dependence	No	Flash Player	Silverlight	No

FIGUUR 12 EEN TABEL MET OVERZICHTEN VAN VERTRAGINGEN BIJ HET GEBRUIK VAN VIDEOSTREAMS.

In de paper “Implementation of HTTP Live Streaming for an IP Camera using an Open Source Multimedia Converter” [13] staan de resultaten van enkele metingen gemaakt met een commerciële IP camera met een resolutie van 704x480 en 30 beelden per seconde. Daarbij is duidelijk op te merken (bij “End-to-End Latency”) dat de vertraging zelfs bij het gebruik van geavanceerde apparatuur bij een lage resolutie meerdere seconden in beslag neemt. Daarmee is het opzetten van een videostream vanaf een Android telefoon, die gehele beelden verstuurd, uitgesloten als optie.

6.2 Communicatietechnieken

6.2.1 Huffman encoding

Het maken van foto's levert inherente vertraging op. Foto's worden op Android meestal opgeleverd in het JPEG formaat, en worden dus gecomprimeerd voordat de data uitgelezen kan worden. Echter is de ongecomprimeerde data voor analyse vereist, omdat de lijnen van de foto niet uitgelezen kunnen worden als de foto gecomprimeerd is. Dit komt doordat er voor de compressie van de foto gebruik wordt gemaakt van Huffman encoding.

Bij Huffman codering worden symbolen (bijvoorbeeld bytes, DCT coëfficiënten, etc.) op basis van statistische voorkomendheden gecodeerd in verschillende lengtes. Een symbool dat vaker voorkomt zal worden gecodeerd met een code welke maar enkele bits in beslag neemt, terwijl symbolen die minder vaak voorkomen worden gecodeerd met meer bits. [14]

Als de lijnen van de foto individueel gecomprimeerd waren, was de compressie geen probleem geweest. Dan zouden de benodigde lijnen zonder gedecomprimeerd te worden, verstuurd kunnen worden en de niet benodigde lijnen weggegooid kunnen worden. Aangezien de hele foto met Huffman encoding gecomprimeerd is, kunnen de lijnen niet individueel geselecteerd worden en moet de foto als gecomprimeerd geheel verzonden worden.

De foto moet daarbij na de verplichte compressie ook weer gedecomprimeerd worden, wat leidt tot vertraging.

6.2.2 Threading

Om de applicatie zo snel mogelijk te laten verlopen, wordt er gebruik gemaakt van threading (eis M.5). De threads zorgen er voor dat sommige processen parallel kunnen worden uitgevoerd.

Als dat niet gebeurt, dan wordt alles sequentieel uitgevoerd, waarbij er lange periodes zijn waar bijvoorbeeld gewacht moet worden door de processor tot de foto gemaakt is, of totdat het verzenden van de foto klaar is.

Zie voor een overzicht van de verschillende threads pagina 40 .

Er zijn een aantal hoofdtaken aanwezig. Allereerst moet de foto gemaakt worden, vervolgens moet de foto gedecomprimeerd worden, waarna de foto getoond moet worden en moet worden verzonden.

Er moet dus een thread zijn om de foto te maken en een thread om de foto te decomprimeren. De verkleining van de foto gebeurt op de zelfde thread als de decompressie. Het tonen van de foto gebeurt echter weer op apart op de OpenGL thread. Het verzenden van de foto gebeurt uiteindelijk ook op een aparte netwerkthread. Daarnaast is er nog een tweede netwerkthread voor de data aanvragen.

Het grotendeel van de applicatie, draait in verschillende threads. De fotodata wordt hierom opgeleverd in een “buffer”, een soort tussenstop. Elke keer als een foto gemaakt is, wordt deze toegevoegd aan de buffer, waarna iets later deze uit de buffer wordt gehaald om verzonden te worden.

6.2.3 Asynchrone netwerkcommunicatie

Als een opdracht wordt gestuurd vanaf de externe computer, is de kans groot dat deze pas een halve seconde later wordt uitgevoerd op de Android smartphone en dat het antwoord dus niet direct beschikbaar is. Er moet rekening worden gehouden dat er niet direct een juiste reactie volgt na een opdracht. Daarom loopt de netwerkcommunicatie asynchroon (eis M.6). Het alternatief is dat de netwerkcommunicatie synchroon loopt, en dus het maken van foto's gestopt moet worden tijdens en rond het sturen, ontvangen en verwerken van opdrachten.

De lengte van de vertraging die kan optreden bij het wisselen van opdrachten hangt af van de lengte van proces om de foto te verkrijgen. Dit houdt in het maken van de foto en het decomprimeren van de foto. Zie hiervoor het hoofdstuk Testen op pagina 50.

6.2.4 Compressie vermijden

Om de vertraging tussen het wisselen van opdrachten te verkorten, evenals de vertraging tussen foto's in, dient de (de)compressietijd verkort te worden en/of de tijd van het maken van foto's. De (de)compressietijd kan geëlimineerd worden door het gebruik om een ongecomprimeerd formaat bij het maken van de foto's. Dat wordt echter nog niet vaak ondersteund op Android telefoons. Het gaat hier voornamelijk om het RAW foto-formaat (eis W.1). Op Android 5.0 is dit op een selectief aantal apparaten beschikbaar, zoals op het moment van schrijven de Nexus 5 en Nexus 6. Echter staat de ondersteuning wel in de kinderschoenen. Er kan ook gekeken worden naar YUV[3] ondersteuning. De iPhone van Apple heeft TIFF ondersteuning, ook een RAW formaat.

6.2.5 Belasting netwerk

Naast de vertraging door bezetting door processen, maakt ook de grootte van het over het netwerk te verzenden pakket uit. Een overbezetting van het netwerkverkeer leidt tot extra vertraging. Daarom is het aantal te verzenden lijnen instelbaar. Het is ook niet aan te raden om de gehele foto te versturen in de praktijk, enkel voor testdoeleinden.

6.2.6 Geen netwerkvertraging

Als de foto geanalyseerd wordt op de Android telefoon zelf, valt de vertraging door de netwerkcommunicatie weg. Het proces kan wel alsnog meer vertraging opleveren, omdat de Android telefoon een tragere processor heeft.

6.3 Mobiele beeldverwerking

Om de kopdiktemeter geschikt te maken voor gebruik in de kassen, is er gekeken naar de mogelijkheid om de beeldverwerking op een mobiele telefoon uit te kunnen voeren (eis C.1). Omdat beeldverwerking de processor intensief benut, is er onderzocht welke eisen dit stelt aan de mobiele telefoon en onder welke omstandigheden het mogelijk is om de kopdiktemeter mobiel te gebruiken.

Aangezien er een hoge vereiste is aan de resolutie, is de shuttertijd van de camera behoorlijk groot. Terwijl de camera bezig is om de foto te maken, is er veel ongebruikte processorkracht over. De tijd tijdens het maken van de foto kan makkelijk ingevuld worden door de beeldanalyse van de vorige foto uit te voeren op de telefoon in plaats van op een externe computer.

Om te onderzoeken of de beeldanalyse ook op de Android telefoon zelf uitgevoerd kan worden, is er een beeldanalyse-programma geschreven.

> Deze beeldanalyse wordt verder besproken in het hoofdstuk Systeemontwerp onder het kop Lokale berekening. Zie hiervoor pagina 40.

[3] YUV is een kleurruimte die ook in PAL-televisiecodeersysteem wordt gebruikt. De Y staat voor het helderheidssignaal en U en V zijn kleurcomponenten. Uit YUV kan een RGB (Rood Groen Blauw) signaal berekend worden.

Na onderzoek is bevonden dat het beeldanalyse-algoritme prima lokaal op de Android telefoon kan draaien in de huidige opstelling.

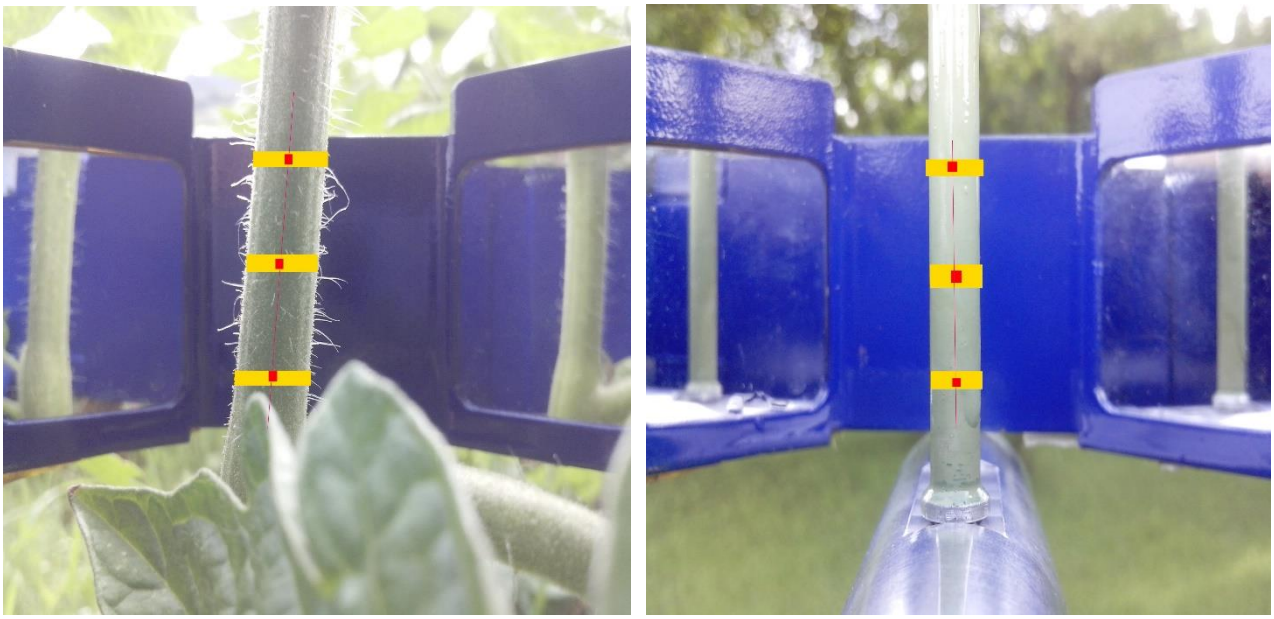
Voor resultaten van dit onderzoek zie het hoofdstuk Testen op pagina 50.

De code van de beeldanalyse uit het verslag van het bedrijfsproject “Contactloze diktebepaling van tomatenstengels door segmentatie en lijndetectie” [3] is gebruikt als basis. De beeldanalyse uit dit verslag voldoet niet aan alle eisen voor een succesvolle beeldanalyse, maar biedt een fundering om mee te werken en een schatting van de *workload*. Deze code is aangepast, geoptimaliseerd en draaiend gekregen op een Android telefoon. De resulterende beeldanalyse heeft een ander algoritme om het analyseprobleem op te lossen. Het belangrijkste verschil is dat de rotatie van de steel anders wordt berekend. Deze aanpassing is gemaakt op basis van een idee van de opdrachtgever.

De code draait sneller dan de Android telefoon foto's kan maken en zorgt daarmee dus niet voor extra vertraging.

Er worden in de geschreven beeldanalyse 3 lijnen met pixels uit de afbeelding uitgelezen en gebruikt voor de beeldanalyse. Het aantal uitgelezen en te verwerken lijnen bepaalt de snelheid van het algoritme. Het is daarom belangrijk om een zo min mogelijk aantal lijnen te verwerken. Het vinden van de steel in de 3 lijnen met pixels, leidt tot 3 verschillende punten (op de afbeelding gemaakt door de camera). Deze punten kunnen verticaal zijn opgesteld, dan staat de camera in relatie tot de steel recht. De drie punten kunnen ook meer diagonaal staan, in welk geval dat gedetecteerd wordt en uit de hoek tussen deze punten de hoek van de rotatie berekend kan worden. In principe kan dit proces ook met alleen twee lijnen worden uitgevoerd.

Onderstaand zijn afbeeldingen te zien die gemaakt zijn door de contactloze kopdiktemer. Links staat de steel scheef, rechts staat de steel recht. De gele vlakken stellen de delen van de lijnen voor waar de groene steel wordt onderscheiden van de blauwe achtergrond. De rode punten geven de middelpunten van de steel aan voor de drie lijnen.



FIGUUR 13 AFBEELDINGEN GEMAAKT DOOR DE CONTACTLOZE KOPDIKTEMER.

Het kan handig zijn om een groter aantal lijnen te gebruiken in de beeldanalyse. Een reden hiervoor kan zijn dat het gebruiken van meerdere lijnen een hogere nauwkeurigheid oplevert of het gebruiken van meerdere lijnen bepaalde artefacten tegen kan gaan.

Indien er 32 lijnen verwerkt worden met hetzelfde algoritme in plaats van 3, draait het proces nog steeds zo snel als mogelijk, omdat de beeldanalyse langer duurt dan de andere parallel draaiende processen.

Zie voor een overzicht van de gemeten tijden het hoofdstuk Testen op pagina 50.

Daaruit kan afgeleid worden dat de beeldanalyse nog uitgebreid kan worden, met meer complexe methodes. Dit kan handig zijn voor het onderscheiden van kleuren, bijvoorbeeld de kleuren van de achtergrond en de steel.

Het is belangrijk om niet de gehele afbeelding te verwerken bij elke gemaakte foto. Dit kan snel een te grote taak zijn voor de processor van de telefoon en leiden tot vertraging. Indien toch de hele afbeelding geanalyseerd dient te worden, wordt aangeraden om zoveel mogelijk gebruik te maken van de GPU, aangezien deze beelden veel sneller kan verwerken.

De in de bijlage beschreven uitwerking is voornamelijk bedoeld als “loadtest”. Door de eindgebruiker zal een betere versie gebruikt worden, namelijk de beeldanalyse van de opdrachtgever. Deze beeldanalyse zal betere resultaten opleveren.

Wellicht een bijkomend voordeel van het draaien van de beeldanalyse op de Android telefoon zelf, is dat er minder vertraging optreedt, aangezien er geen gegevens worden verstuurd over het netwerk.

Er kan wel verwacht worden dat het batterijverbruik toeneemt door de hogere belasting van de CPU.

6.4 Gebruikers feedback

Bij het gebruik van de applicatie krijgt de gebruiker feedback. Dit is te onderscheiden in feedback tijdens het gebruik en feedback van de analyse van de applicatie. Te onderscheiden is:

- Feedback over de positionering van de camera
- Feedback over het proces van de beeldanalyse
- Feedback over de uitvoer van de analyse

6.4.1 Feedback over de positionering van de camera

De positie van de camera ten opzichte van de steel is van belang voor de accuraatheid van de meting. Om een goede meting te kunnen doen zal de gebruiker de camera rechtop en recht voor de steel moeten houden. Wanneer de gebruiker de camera schuin houdt, of de plant niet goed in beeld is, geeft de applicatie een foutmelding.

Allereerst werd er gedacht aan een weergave d.m.v. OpenGL voor feedback aan de gebruiker (eis S.6). Een 3D representatie van de steel zou op het scherm de positie van de steel laten zien. De gebruiker zou dan de telefoon kunnen bijsturen om een gunstige positie te krijgen.

Om een 3D weergave via OpenGL te verkrijgen is eerst informatie nodig over positie en rotatie. Het verkrijgen van deze informatie kan relatief lang duren en is niet altijd beschikbaar. Dit proces geeft een grote vertraging.

Zie voor het vergelijken van de metingen van de vertragingen het hoofdstuk Testen op pagina 50.

Omdat het proces, zoals hierboven beschreven is, te veel vertraging gaf, is gekeken naar feedback d.m.v. directe beelden van de camera. Een directe feed van de camera kan wel wat lastiger te interpreteren zijn, omdat het beeld door spiegels niet een normaal aanzicht oplevert (eis S.5). De vertraging is wel kleiner dan bij het wachten op resultaat van positie en rotatie van de steel, maar kan nog steeds behoorlijk zijn vanwege de benodigde decompressie van de afbeelding.

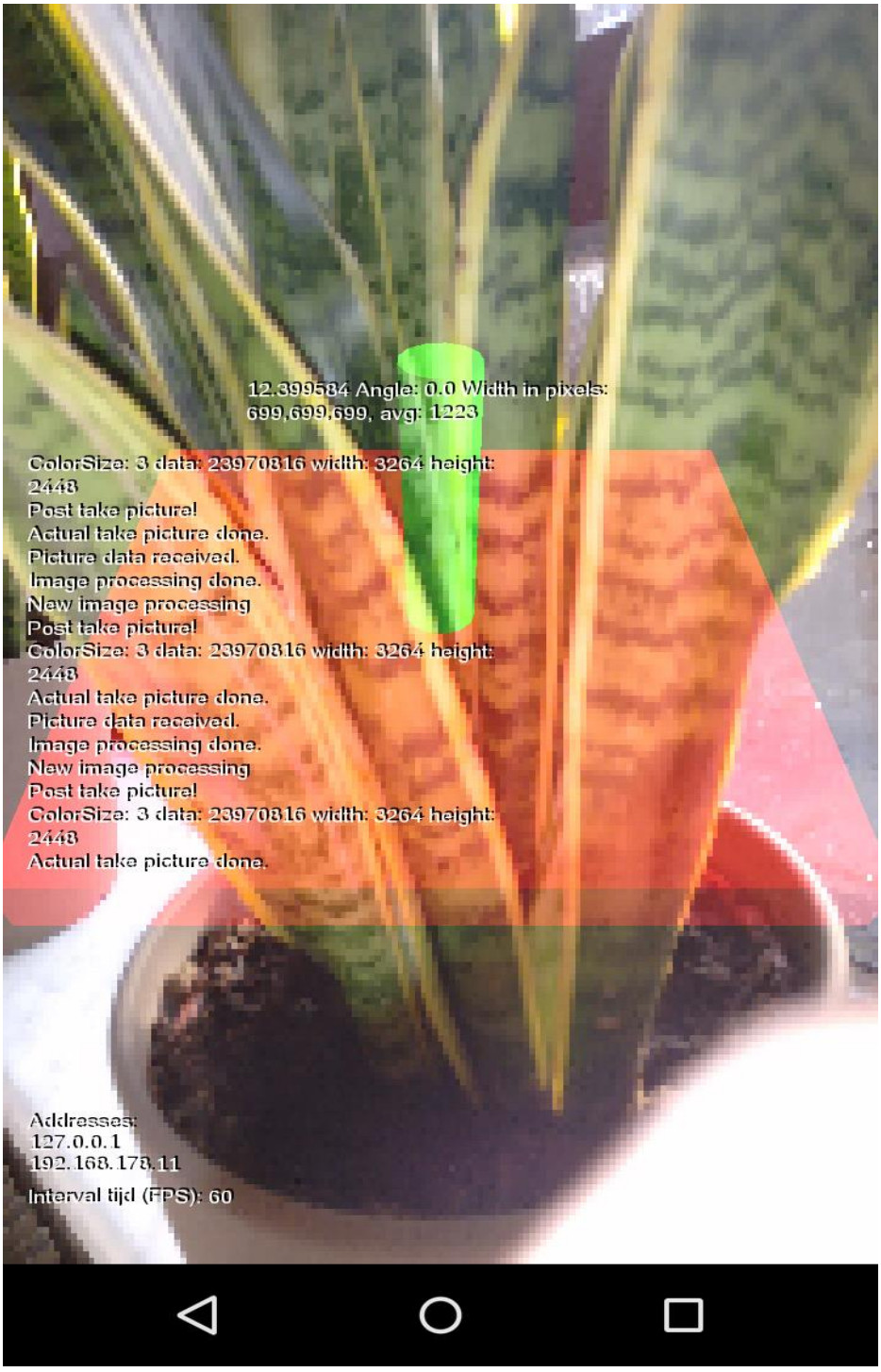
FIGUUR 14 EEN DIRECTE FOTO GEMAAKT DOOR DE CAMERA MET DE CONTACTLOZE KOPDIKTEMETER MODULE AANGESLOTEN.

Elke methode om de omgeving te tonen aan de gebruiker heeft voor-en nadelen:

- Met OpenGL is een duidelijk aanzicht te zien, maar dit aanzicht is niet altijd accuraat, op tijd zichtbaar of beschikbaar.
- Bij een directe feed van de camera wordt duidelijkheid van aanzicht ingeruild voor nauwkeurigheid, minder vertraging en beschikbaarheid.

Daarom is gekozen om beide technieken toe te passen, omdat ze elkaar goed aanvullen. De beelden van de OpenGL weergave en de camera worden over elkaar heen gelegd.

In Android is het mogelijk om een “preview” beeld te krijgen van de camera met een lagere resolutie, die vaker ververst dan een hoge resolutie foto. De “preview” functionaliteit is vergelijkbaar met video-opname. Dit is een Android specifieke functie en daarom is het moeilijk om de beelden te tonen d.m.v. OpenGL. Om deze reden is er geen gebruik gemaakt van deze modus. Voor andere platformen zijn er misschien wel alternatieven. De “preview” modus levert ongecomprimeerde beelden op, waardoor de vertraging veel lager kan zijn.



FIGUUR 15 HET UITEINDELIJKE SCHERM MET EEN FOTO GEMAAKT DOOR DE CAMERA OP DE ACHTERGROND EN TEKST EN EEN 3D WEERGAVE OP DE VOORGROND.

6.4.2 Feedback over het proces van de beeldanalyse

De gebruiker moet het verloop van het proces kunnen volgen (eis S.8). Zodra een foto gemaakt is krijgt de gebruiker een flits te zien. Deze flits wordt getoond door het beeld een fractie van een seconde (1 frame) wit te maken. Ook wordt er tekstuele informatie over het proces op het scherm getoond (eis S.7). Deze informatie bevat:

- De IP-adressen van de telefoon zelf. Met deze IP-adressen kan een verbinding gelegd worden van een externe computer.
- Het getekende beeld per seconde (oftewel intervaltijd of FPS).

- De laatste aantal regels van de debug-informatie.

Als er nog geen beeld van de camera binnen is gekomen, wordt er een standaardafbeelding gebruikt. Deze afbeelding wordt getoond in plaats van de camera-afbeelding.

6.4.3 Feedback over de uitvoer van de analyse

Met de output van de kopdiktemeter, verwacht de gebruiker feedback van de dikte van de steel (eis S.8). De applicatie genereert op basis van de analyse van de foto een uitkomst welke de diameter van de plant aangeeft. Deze uitkomst wordt door de applicatie na de verwerking (op de externe computer of de telefoon zelf) op het telefoonscherm weergegeven.

6.5 Configuratie en debugging

Aangezien dit project een innoverende applicatie omvat, dient er veel getest en gesleuteld te worden. Het is daarom handig om methodes in te bakken in de applicatie, waarmee bepaalde waardes aangepast of uitgelezen kunnen worden.

Deze extra functionaliteiten bestaan uit:

- Instellingen die ingesteld kunnen worden over de netwerkverbinding.
- Instellingen die ingesteld kunnen worden in het configuratie-bestand.
- Debugging informatie die gedetailleerde inzage in het proces geeft.
- Testroutines waarbij de juistheid van functies bekeken kan worden.
- Extra informatie die opgevraagd kan worden over de netwerkverbinding.

6.5.1 Instellingen over netwerkverbinding

De belangrijkste waardes die aangepast dienen te kunnen worden, zijn de specifieke lijnen uit de foto's die opgevraagd dienen te worden. Deze lijnen kunnen gespecificeerd worden over het netwerk d.m.v. een set commando's (eis M.3). Het maximum aantal mogelijk op te geven lijnen (op moment van schrijven is dat 32 lijnen maximaal) is aan te passen in de broncode.

6.5.2 Instellingen in configuratie-bestand

In sommige gevallen levert de camera donkere beelden op, omdat de omgeving onderbelicht is. In dat geval is het handig om een flitser actief te hebben. Voor onderzoeksdoeleinden heeft het de voorkeur om de flitser aan en uit te kunnen zetten (eis S.10). Als de flitser gebruikt dient te worden, moet dit aangeven worden in het configuratiebestand (“Config.java”) voordat het Android programma wordt gecompileerd.

Voor de netwerkcommunicatie kan het nodig zijn dat de Android telefoon zich als een Wifi hotspot gedraagt (eis C.2). De externe computer kan dan verbinden met de telefoon zonder tussenkomende router. Dit is tegenwoordig mogelijk in de meeste Android telefoons. Door in het configuratiebestand de *auto-hotspot feature* aan te zetten wordt bij het starten van de Android applicatie geprobeerd een Wifi hotspot op te zetten met enkele in de broncode gedefinieerde waardes (o.a. gebruikersnaam, wachtwoord en SSID).

De door de camera gemaakte afbeeldingen dienen eerst verkleind te worden, voordat ze naar de GPU worden verzonden. Als dat niet gebeurt moet er een onnodig grote afbeelding verzonden worden, wat de applicatie kan stilleggen. De factor van verkleining van deze afbeeldingen leidt tot verschillen in scherpte in de afbeelding. Een kleinere afbeelding leidt tot een snellere verzending naar de GPU. De factor kan aangepast worden in het configuratiebestand.

Als de camera van de Android telefoon draait, dan is mogelijk nodig om de afbeelding terug te draaien (eis S.2). Een aanpassing in het configuratiebestand maakt het mogelijk de oriëntatie van de afbeelding te veranderen. De oriëntatie van de camera blijkt echter moeilijk te corrigeren, omdat de diverse Android telefoons anders reageren op de Android specifieke commando's. Daarom wordt de oriëntatie van de

camera niet veranderd, maar wordt de data (de lijnen van de afbeelding) anders uitgelezen. Ook kan de afbeelding op het scherm geroteerd worden d.m.v. OpenGL..

Sommige *firewalls* kunnen bepaalde netwerkpoorten blokkeren. Daarom moeten de gebruikte netwerkpoorten voor de netwerkcommunicatie aangepast worden (in het configuratiebestand) (eis S.3). Er dient opgelet te worden dat dezelfde netwerkpoorten ook aangepast worden in de cliënt-applicatie op de externe computer.

6.5.3 Debugging informatie

Naast de diverse configuratie instellingen, is het belangrijk dat de gebruiker berichten krijgt over de lopende processen, zoals bijvoorbeeld of een foto zojuist gemaakt is (eis S.8). Dit gebeurt op twee manieren. Deze zogenaamde debug-berichten zijn uit te lezen met de Android *SDK*[4]. Met "ADB debugging[5]" kunnen de berichten opgevangen worden op PC via USB-kabel of via Wifi. Ook worden deze berichten op het scherm getoond (eis S.7).

6.5.4 Testroutines

De *loadtest* voor de beeldanalyse wordt automatisch gestart en gedraaid zolang er geen communicatie aanwezig is tussen de Android telefoon en de externe computer (eis C.1). Er zijn diverse variabelen in te stellen in het configuratiebestand om deze beeldanalyse aan te passen.

Een testfunctie is aanwezig om de nauwkeurigheid van de uitkomsten van de loadtest te meten (eis S.9). Deze testfunctie leest een JPEG of PNG afbeelding in en past de verwerking van de loadtest toe op deze afbeelding. Vervolgens worden de uitkomsten, bv. de diameter van de steel, in de "log" gezet. Deze testfunctie is het makkelijkst te draaien en te bekijken op een PC; de uitkomsten zijn identiek op een Android telefoon.

6.5.5 Extra informatie over netwerkverbinding

Het is mogelijk om de gemaakte foto in zijn geheel over het netwerk op te vragen (eis S.1). Dit kan zonder aanpassingen aan de applicatie door een commando aan te roepen over het netwerkverkeer. Het opvragen van de gehele foto kost enige tijd omdat het belastend is voor het netwerk (een gehele foto beslaat meerdere MegaBytes). Het duurt daardoor langer dan het opvragen van enkele lijnen van de foto.

Er is naar gekeken om accelerometer data van de Android telefoon te gebruiken om de relatieve verplaatsing op te vangen (eis S.4). Deze relatieve verplaatsing zou theoretisch gebruikt kunnen worden om met twee foto's diepte te berekenen, waarna de dikte van de steel berekend kan worden. Dit idee lijkt op *verzeiling*.

Verzeiling is een bekend begrip in de scheepsvaart. Een verzeiling is de verplaatsing van de positielijn op verschillende tijdstippen naar een gezamenlijk tijdstip. Dit wordt gedaan in de richting van de grondkoers (d.m.v. b.v. een kompas) over de afstand die afgelegd is tussen het tijdstip van de waarneming en het gezamenlijk tijdstip. Met gebruik van meerdere kenbare punten kan een kruispeiling gedaan worden. Omdat er vaak onzekerheden zijn in de koers en afgelegde afstand, is het belangrijk om de verzeilingsduur zo kort mogelijk te houden. [15]

Accelerometer data is beschikbaar via het netwerk. De accelerometer data kan gebruikt worden om het beeldanalyse proces te helpen. De accelerometer data wordt opgeleverd als gemiddelde over een periode in meter per seconde. Door deze waarde te vermenigvuldigen met de verstreken tijd die ook geleverd wordt in de netwerkcommunicatie kan men in theorie de verplaatsing berekenen. In de praktijk blijkt echter dat de accelerometer in het gros van de Android telefoons van te lage kwaliteit is en benodigde precisie ontbreekt.

[4] Zie voor de Android SDK https://developer.android.com/sdk/index.html
[5] Zie voor uitleg over Android debugging https://developer.android.com/tools/help/adb.html

Naast de accelerometer data, wordt ook de hoogte en breedte van de afbeelding over het netwerk gestuurd, evenals de lijnnummer (plaats op de y-as van de afbeelding) per aanwezige lijn.

Om feedback vanaf de PC te kunnen tonen is er een functie in de netwerkcommunicatie geïmplementeerd om een stukje tekst mee te sturen. Dit stukje tekst, op de PC opgesteld, kan bijvoorbeeld de uitkomst van de berekening op het scherm tonen. De positie, rotatie en dikte van de steel kan ook meegestuurd worden vanaf de PC.

7 Systeemontwerp

Bij het ontwerpen van de software voor de contactloze kopdiktemeter, zijn er beslissingen gemaakt over diverse onderdelen:

- Netwerk communicatie
  - De specificatie van de uitwisseling van gegevens over het netwerk.
- Lokale berekening
  - De werking van de beeldanalyse loadtest.
- Threading
  - De opzet en timing van de verschillende taken.

Daarna wordt de opzet van de structuur van de software uitgewerkt:

- Package indeling
- Design Patterns
- UML diagrammen

7.1 Netwerk communicatie

De netwerkcommunicatie volgt een set specifieke regels. Deze worden hier beschreven (eis M.4).

De netwerkcommunicatie bestaat uit drie onderdelen: de opdrachtinformatie, de uitvoer en de positiedata. Deze drie onderdelen verlopen iteratief. Een nieuwe opdrachtspecificatie kan worden opgegeven terwijl er uitvoer wordt opgehaald. De uitvoer wordt bepaald door de parameters van de opdrachtinformatie.

De opdrachtinformatie bevat:

- Een terugkoppeling over de breedte en hoogte van de camerabeelden
- Een verzending van:
  - De modus (1 voor verzoek om lijnen en 2 voor verzoek JPEG)
  - En indien lijnen worden opgevraagd
    - 32 numerieke waardes met lijnnummers

De uitvoerdata bevat:

- De uitvoerdata grootte in bytes (integer, 4 bytes)
- De modus van het behandelde verzoek (byte)
- Indien de modus een verzoek om lijnen betreft:
  - Het aantal te ontvangen lijnen (integer, 4 bytes)
- De payload[6] grootte in bytes (integer, 4 bytes)
- De tijd sinds vorige verzoek in milliseconden (integer, 4 bytes)
- Accelerometerdata X, Y en Z als numerieke waardes, vermenigvuldigd keer 1000 (drie keer: integer, 4 bytes)
- Indien de modus een verzoek om lijnen betreft:
  - Voor het aantal te ontvangen lijnen:
    - Het lijnnummer (integer, 4 bytes)
    - De data van de lijn (ter grootte van de eerder ontvangen payload grootte)
- Indien de modus een verzoek voor JPEG betreft:
  - De data van de JPEG (ter grootte van de eerder ontvangen payload grootte)

De positiedata bevat:

[6] Payload is een term voor de lading van een bericht. Deze lading is het hoofdonderdeel van het bericht. In dit geval gaat het om de kleurendata van de afbeelding.

- De X positie (waarde van -1000 tot 1000) (integer, 4 bytes)
- De Z positie (waarde van -1000 tot 1000) (integer, 4 bytes)

Zie de bijlage op pagina 72 voor de code van de Java-client. Deze code heeft commentaar beschikbaar zodat duidelijk wordt per stap wat er gebeurt.

7.2 Lokale berekening

Om het in de toekomst mogelijk te maken dat men de beeldanalyse op de Android telefoon zelf kan draaien, wordt er hiermee rekening gehouden bij het ontwerpen en documenteren van de code.

De code die de afbeeldingen analyseert staat in het bestand *LocalProcessor*.java in de *Engine* package.

> Zie de bijlage op pagina 77 voor de code van de lokale berekening. De code heeft commentaar bijgevoegd, zodat duidelijk wordt wat er per stap gebeurt.

7.2.1 Werking samengevat

De beeldanalyse die lokaal berekend wordt bestaat uit de volgende stappen:

1. De foto (JPEG afbeelding) wordt gedecomprimeerd.
2. Een of meerdere, vooraf vastgestelde lijnen worden geanalyseerd. De uitkomst is een ja/nee resultaat voor elke pixel. Een ja betekent dat de pixel binnen het gespecificeerde kleurbereik valt (b.v. een bepaalde groentint).
3. Er wordt geteld tot wanneer de eerste pixel in het kleurbereik valt voor elke lijn.
4. Er wordt geteld tot wanneer de laatste pixel in het kleurbereik valt voor elke lijn.
5. Uit deze twee waardes wordt de breedte berekend voor elke lijn.
6. Tevens wordt het midden berekend voor elke lijn.
7. Uit de middelpunten van de meerdere lijnen kan een hoek/schuinstand van de lijn berekend worden.
8. De hoek/schuinstand wordt gebruikt om de breedte bij te stellen voor elke lijn.
9. De gemiddelde breedte van elke lijn levert het resultaat op.

7.2.2 Ontbrekend

Er wordt in de lokale berekening (loadtest) geen gebruik gemaakt van de spiegels van de contactloze kopdiktemeter. In het geval dat de module met de spiegels wel wordt gebruikt, wordt de beeldanalyse niet significant zwaarder. De beeldanalyse van de opdrachtgever maakt wel gebruik van de module van de contactloze kopdiktemeter. Omdat de contactloze kopdiktemeter-module bij de loadtest niet gebruikt wordt, moet de steel op een constante afstand van de camera gehouden worden om een bruikbaar resultaat te leveren.

Ook wordt er bij de loadtest geen rekening gehouden met perspectivische verkorting of occlusie. Indien de steel in het midden van het beeld staat zal dat niet een groot probleem zijn. Het implementeren van een berekening om rekening te houden met de perspectivische verkorting zal de beeldanalyse niet significant zwaarder maken.

7.3 Threading

Om de gehele applicatie zo efficiënt en vloeiend mogelijk te laten draaien, wordt veel gebruik gemaakt van “threading” (eis M.5). Dat betekent dat veel van de taken parallel aan elkaar verlopen.

Er zijn verschillende threads gemaakt voor o.a.: OpenGL, de camera, de grote berekeningen (o.a. JPEG decompressie) en de transmissie over het netwerk. Voor de transmissie over het netwerk worden twee threads gebruikt.

7.3.1 Legenda

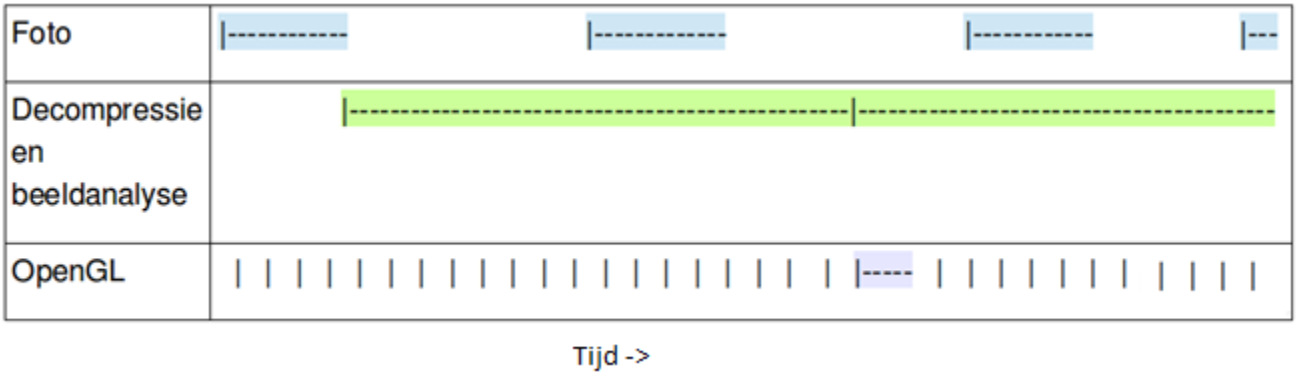
De threads staan in de linker kolom aangegeven. De foto thread houdt het maken en de compressie van de foto in. De beeldanalyse thread houdt het decomprimeren van de foto in en het analyseren van de foto. Beiden zijn optioneel. Dit is afhankelijk van de modus (lijnen of jpeg).

De OpenGL thread heeft ongeveer iedere 1/60ste van een seconde een kleine taak. Het verzenden van de foto naar de GPU voor weergave is een iets grotere taak.

| geeft het begin van een taak aan.

7.3.2 Lokale loadtest

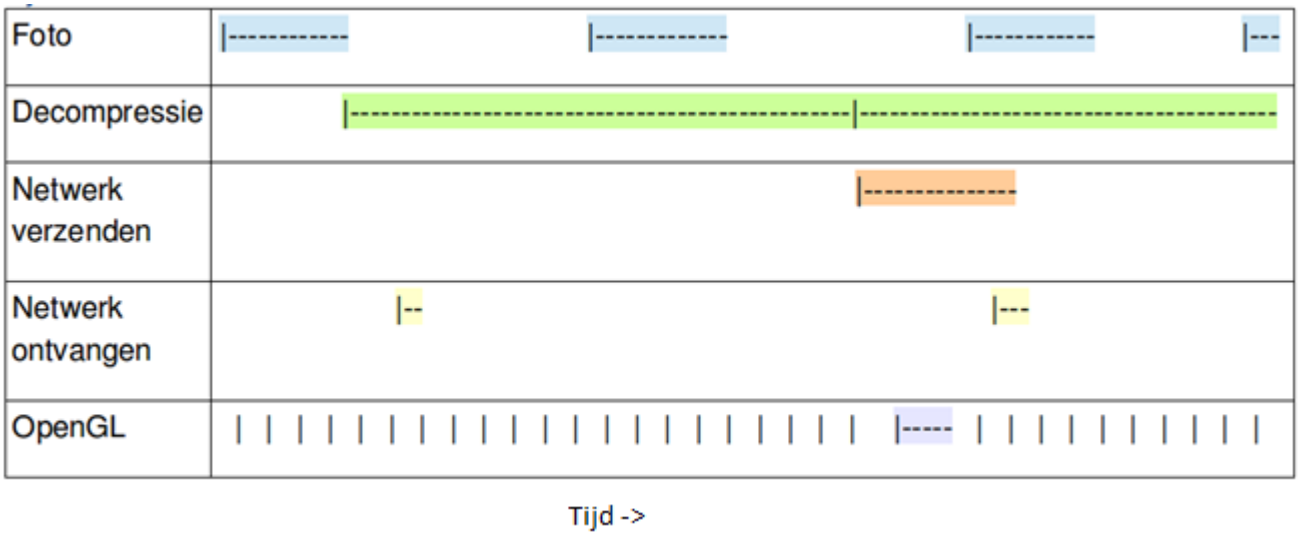
Het verwerken van de beelden loopt parallel aan het maken van de foto's. De eerste keer moet er gewacht worden tot de foto gemaakt is. Bij de volgende beeldanalyse staat er al een foto klaar.



FIGUUR 16 EEN OVERZICHT VAN DE PROCESSEN TIJDENS DE LOKALE LOADTEST.

7.3.3 Lijnen versturen

Onderstaand is te zien dat de taak om data te verzenden over het netwerk wordt gestart als de decompressie gedaan is. Omdat er een aparte thread wordt gebruikt voor het verzenden van de data, hoeft niet gewacht te worden met het starten van een nieuwe beeldanalysetaak tot het verzenden gedaan is.

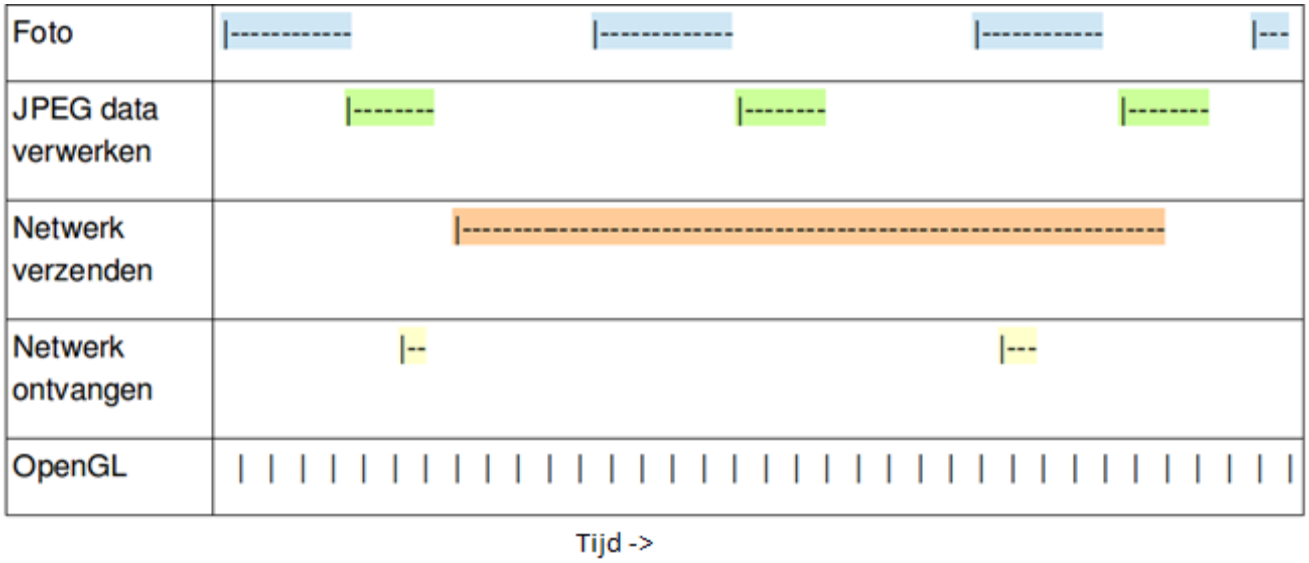


FIGUUR 17 EEN OVERZICHT VAN DE PROCESSEN TIJDENS HET VERSTUREN VAN LIJNEN.

7.3.4 JPEG versturen

De beeldverwerkingstaken duren een stuk minder lang, aangezien er geen beeldanalyse en decompressie wordt uitgevoerd bij het versturen van de gehele JPEG afbeelding. Daarentegen duurt het verzenden van de data veel langer.

In de OpenGL thread wordt geen foto verzonden naar de GPU, omdat er geen gedecomprimeerde foto aanwezig is.



FIGUUR 18 EEN OVERZICHT VAN DE PROCESSEN TIJDENS HET VERSTUREN VAN EEN JPEG.

7.4 Package indeling

Het gebruikte framework LibGDX is een platformonafhankelijk framework. Met LibGDX kan een programma een keer geschreven worden en dan naar meerdere platformen, zoals Android of iOS, gecompileerd worden. Er zijn soms bepaalde instructies nodig die wel platformafhankelijk zijn, zoals het gebruik van de camera. Daarom is de applicatie modulair opgebouwd.

De applicatie bestaat uit 3 modules: de “Core” module, de “Android” module en de “Desktop” module. De Core module bevat de kernfunctionaliteiten van de applicatie, die systeemonafhankelijk zijn. De Desktop en Android modules bevatten de systeemafhankelijke code, van Android en Desktop platformen respectievelijk, met verwijzingen naar de Core module. De Core module is platformonafhankelijk en kan dus losstaand niet uitgevoerd worden. Deze module bevat het gros van de functionaliteiten van de applicatie. Zo min mogelijk functionaliteiten zijn platformafhankelijk gedefinieerd.

De Core module is opgebouwd uit de volgende packages[7]:

- Claw
  - Camera
  - Debug
  - Engine
  - Graphics

[7] Een Java package is een mechanisme om Java classes te organiseren in *namespaces*, waarmee modulair programmeren in Java wordt geleverd. Een *namespace* identificeert een verzameling van namen eenduidig, zodat er geen ambiguïteit kan optreden wanneer een ander object toevallig dezelfde naam heeft.

        - Models
    - Network
    - Tests
    - Utils

De Android module bevat tevens de volgende packages voor platformafhankelijke functionaliteit:

- Claw
    - Android
        - Camera
        - Wifi

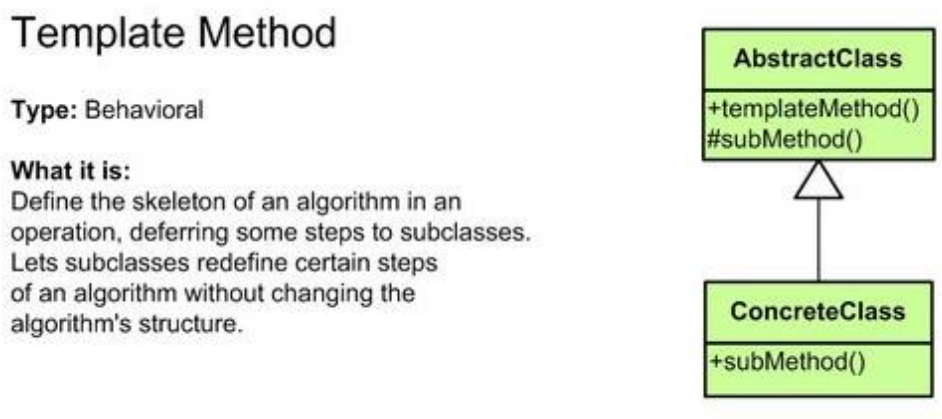
7.5 Design Patterns

Een ontwerppatroon, oftewel design pattern, is een software structuur om een bepaald ontwerpprobleem op te lossen. Een ontwerppatroon is geen algoritme, omdat er geen berekeningsprobleem wordt opgelost maar een ontwerpprobleem.

In de software van de contactloze kopdiktemeter zijn een aantal ontwerppatronen toegepast. In de gebruikte ontwerppatronen [16] is terug te zien dat dit project voornamelijk een grafische applicatie betreft, omdat bijvoorbeeld de meeste klassen niet vaak geïnstantieerd hoeven te worden.

7.5.1 Template Method

Het template method patroon definieert het skelet van een algoritme in een methode, genaamd de template methode, die sommige stappen doorstuurt naar subklassen. Het laat iemand sommige stappen va een algoritme herdefiniëren zonder wijzigingen te maken aan de structuur van het algoritme. [17]



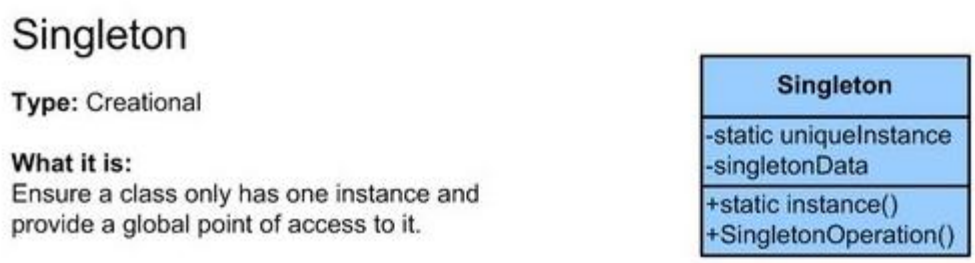
[18]

Gebruikt in klassen:

- DeviceCamera
- Box, Cylinder, Model

7.5.2 Singleton

Het singleton patroon is een software patroon dat het aantal van instanties limiteert van een klasse tot een enkel object. Het is voornamelijk handig te gebruiken wanneer een enkel object aangeroepen wordt verspreid over het systeem. Het concept is soms gegeneraliseerd naar systemen dat efficiënter opereren wanneer alleen een enkel object geïnstantieerd is. De term komt van het wiskunde concept van een singleton. [19]



[20]

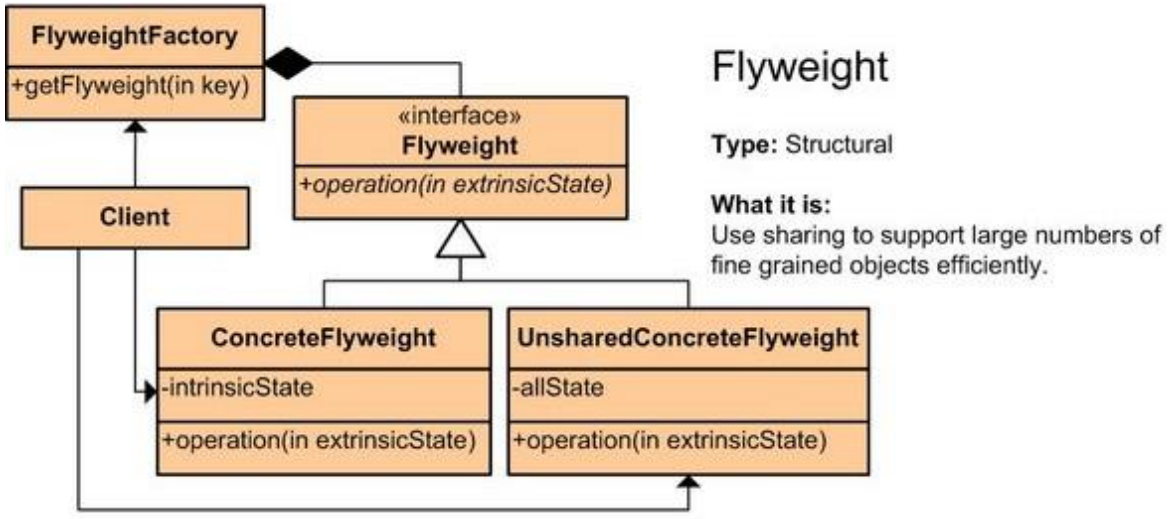
Gebruikt in klassen:

- GUI
- CameraImage
- DataManager
- ShaderManager

7.5.3 FlyWeight

Een flyweight is een object dat geheugengebruik minimaliseert door zoveel mogelijk data te delen met soortgelijke objecten. Het is een manier om grote aantallen objecten te gebruiken wanneer een simpele herhaaldelijke representatie te veel geheugen in beslag zou nemen. Vaak worden sommige delen van het object als externe data structuren gehouden en tijdelijk gegeven aan flyweight objecten als ze nodig zijn.

Een voorbeeld voor het gebruik van het flyweight patroon is in een word processor, waarbij karakters hergebruikt kunnen worden. [21]



[20]

Gebruikt in klassen:

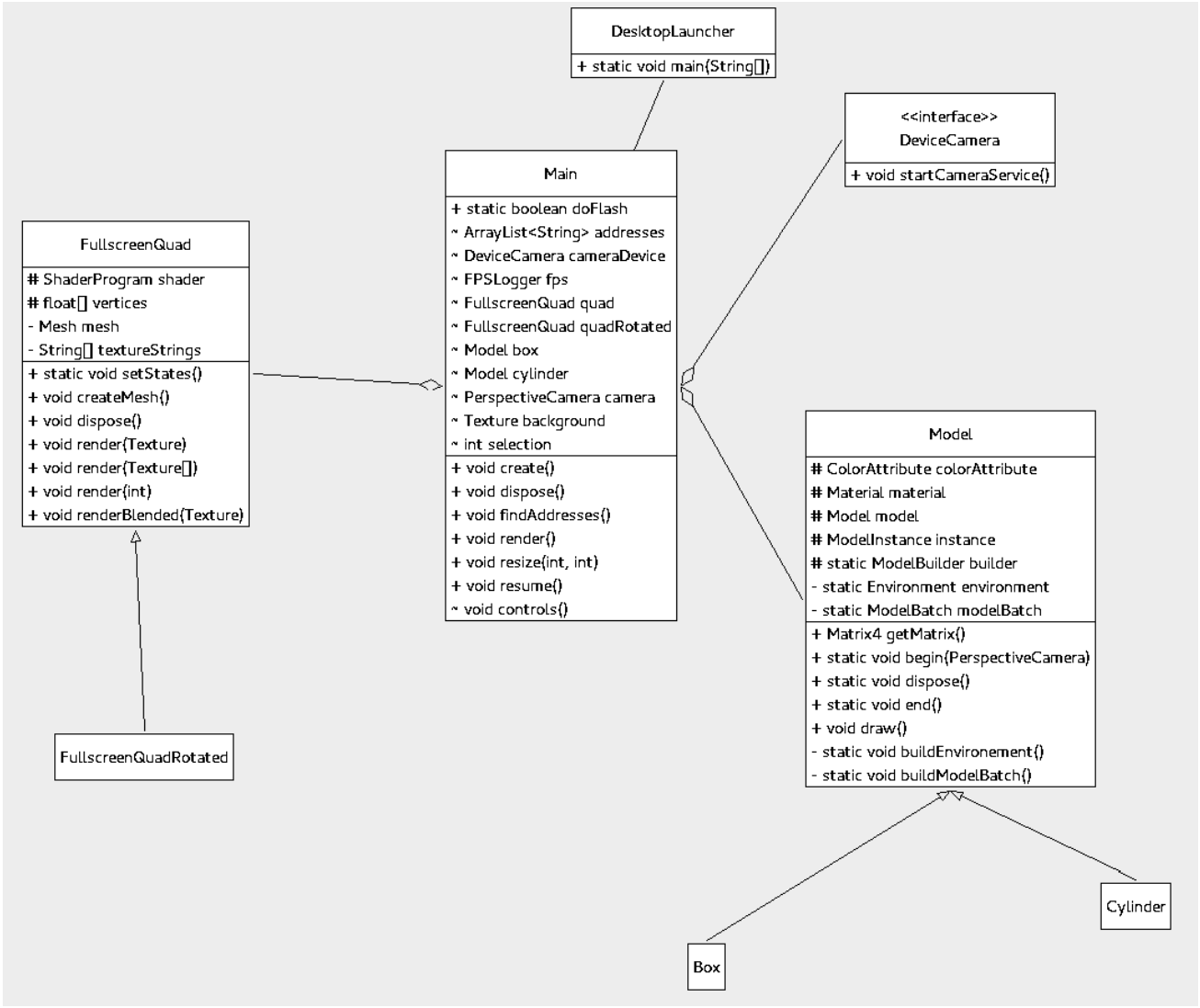
- ShaderManager

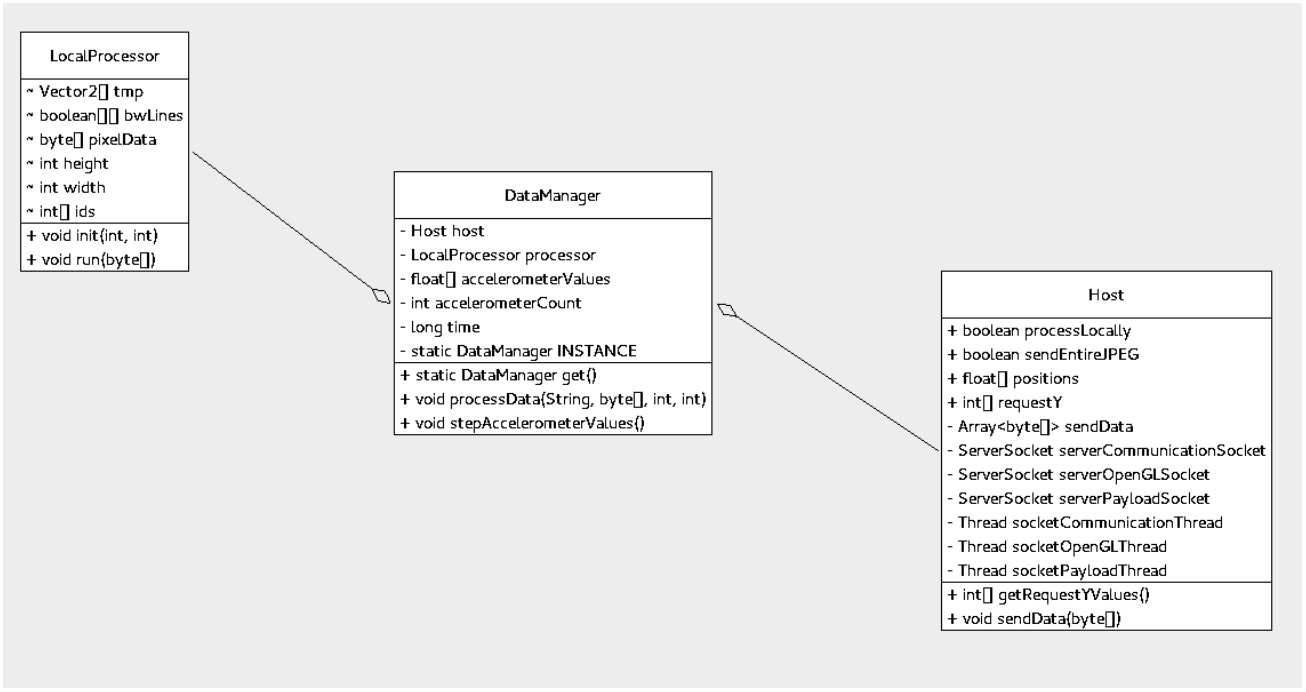
7.6 UML diagrammen

7.6.1 Klassendiagrammen

Op de volgende bladzijde staan de overzichten van de belangrijkste klassen. Het gaat daarbij o.a. om een topologie van de Main, Host, Client en DataManager klassen.

In de bijlage op pagina 88 staan de overige klassendiagrammen.





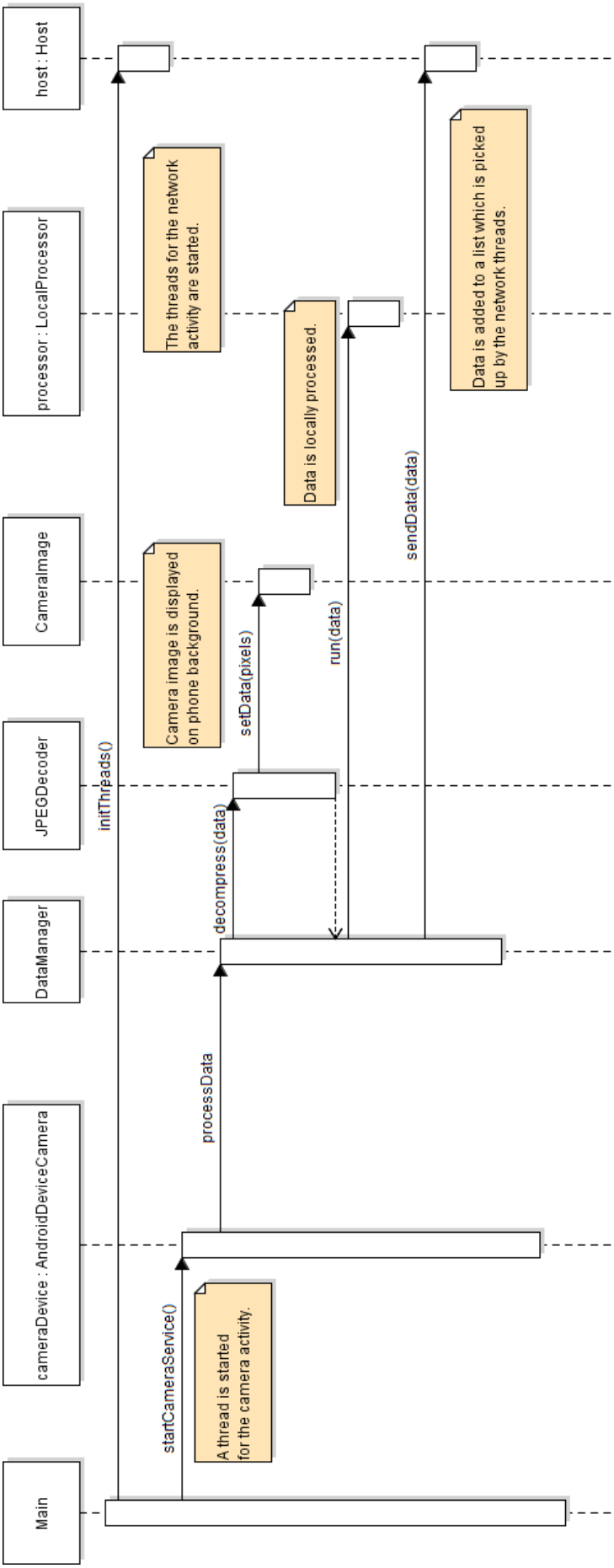
FIGUUR 19 EEN OVERZICHT VAN DE BELANGRIJKSTE KLASSEN VAN DE CONTACTLOZE KOPDIKTEMETER APPLICATIE.

De host-klasse verzendt informatie over het netwerk. De LocalProcessor voert de lokale beeldanalyse uit, indien nodig. De DataManager stuurt het gehele proces van foto tot verzending aan. De Main klasse is de hoofdklasse van de applicatie en daarmee het startpunt voor alle processen. De model-klasse wordt gebruikt voor de 3D-modellen. De FullscreenQuad-klasse stelt de applicatie in staat om een afbeelding.

7.6.2 Sequentiediagrammen

Op de volgende bladzijde is het belangrijkste sequentiediagram van de software te vinden. Dit diagram geeft een algemeen beeld over de werking van de applicatie.

In de bijlage op pagina 90 zijn de overige sequentiediagrammen te vinden.



FIGUUR 20 EEN OVERZICHT VAN HET MAKEN EN VERZENDEN VAN FOTO'S OP DE HOST-ZIJDE.

8 Testen

Er worden testen uitgevoerd om te kijken welke technieken de hoogste reactiesnelheid behalen of de meeste beelden opleveren (eis M.9). Daarnaast worden ook testen uitgevoerd om de nauwkeurigheid van de beeldanalyse te bepalen.

8.1 Testen van algemene functionaliteit

Voor de software voor de contactloze kopdiktemeter is een lijst met eisen opgezet. In dit hoofdstuk wordt gecontroleerd of aan de eisen van het project is voldaan.

Must have		Voldaan
M.1	Ondersteuning van de Android telefoon van het merk Wolfgang.	Ja
M.2	Ondersteuning voor de Dev-C++ ontwikkelomgeving.	Ja
M.3	Verzending van (een variabel in te stellen aantal) horizontale lijnen van de gemaakte afbeelding over het netwerk; minimaal 9 lijnen.	Ja
M.4	Beschrijving van de netwerkcommunicatie.	Ja
M.5	De Android-applicatie gebruik laten maken van threading.	Ja
M.6	De netwerkcommunicatie asynchroon laten verlopen o.a. d.m.v. meerdere communicatiekanalen.	Ja
M.7	Opstellen en uitvoeren van berekeningen om de uiteindelijke nauwkeurigheid in te schatten van de beeldanalyse.	Ja
M.8	Een nauwkeurigheid van 0,1mm/pixel voor de afbeelding van de camera.	Ja
M.9	Een snelheid van minstens 1 afbeelding per seconde.	Gemiddeld

Should have		Voldaan
S.1	De hele afbeelding van de camera uitlezen en verzenden over het netwerk.	Ja
S.2	Een manier om het uitlezen van de lijnen te draaien voor verzending (horizontaal/verticaal)	Ja
S.3	Het mogelijk maken om netwerkadressen en porten in te stellen voor de netwerkcommunicatie.	Ja
S.4	Toegang tot de accelerometer data van de telefoon.	Ja
S.5	Visualisatie van de afbeelding gemaakt door de camera op het scherm van de Android telefoon.	Ja
S.6	Visualisatie van de positie, oriëntatie en dikte van de steel in 3D met OpenGL op het scherm van de Android telefoon.	Ja
S.7	Tekstuele informatie getoond op het scherm van de Android telefoon, zoals netwerkadressen.	Ja
S.8	Debug-informatie die wordt gegeneerd tijdens de werking van de applicatie.	Ja
S.9	Een testroutine om de mogelijke nauwkeurigheid van de beeldanalyse te berekenen.	Ja
S.10	Het mogelijk maken om optioneel de flitser van de camera te gebruiken.	Ja

Could have		Voldaan
C.1	Een loadtest om te kijken of de beeldanalyse eventueel op de Android telefoon zelf kan draaien. Deze loadtest omvat een basis vorm van de beeldanalyse.	Ja
C.2	Een methode om automatisch een Wifi hotspot op te zetten op de Android telefoon.	Ja
C.3	Ondersteuning voor een verbinding met USB i.p.v. Wifi.	Ja

Won't have		Voldaan
W.1	Een volledige, complete beeldanalyse voor gebruik met de contactloze kopdiktemeter module.	Incompleet

In de bovenstaande tabel is te zien of er aan alle eisen is voldaan. De eisen zijn gecontroleerd gedurende de ontwikkeling en in dit verslag of in de code, waarbij voor specifieke eisen testen zijn uitgevoerd.

Eis M9 is een eis die lastiger te definiëren is en is daarom getest met een loadtest. Uit deze loadtest is gebleken dat er wel aan de eis wordt voldaan, maar niet ten alle tijden, omdat soms de hoeveelheid beelden onder de 1 afbeelding per seconde valt. Dit kan te maken hebben met de belasting van de telefoon door andere applicaties. Gemiddeld kan er echter worden geconcludeerd dat er aan deze eis wordt voldaan. Dit is verder uitgelegd in het hoofdstuk "Testen van snelheid en vertraging" op pagina 52.

De Won't have-eis hoeft niet voltooid te worden en is niet volledig aan voldaan bij de ontwikkeling van deze applicatie. Wel is er een beeldanalyse applicatie ontwikkeld om loadtesten mee uit te voeren. Deze kan echter niet worden beschouwd als een goede vervanging voor de originele beeldanalyse applicatie.

8.2 Testen van beeldanalyse

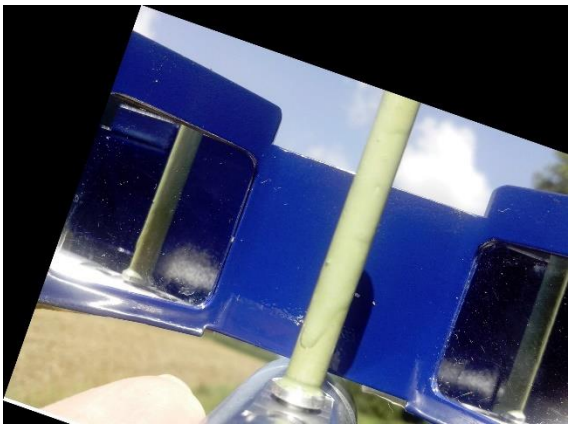
In dit hoofdstuk wordt er gekeken naar de nauwkeurigheid van de beeldanalyse, die onderliggend is aan de loadtest. Met de uitkomst van de beeldanalyse worden vervolgens ook de berekeningen van de nauwkeurigheid van de beeldanalyse geverifieerd.

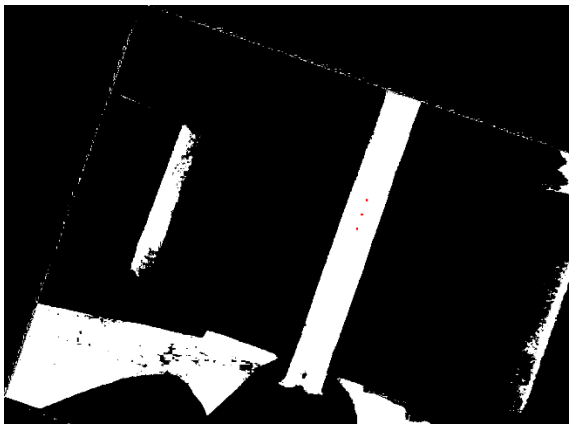
8.2.1 Omstandigheden

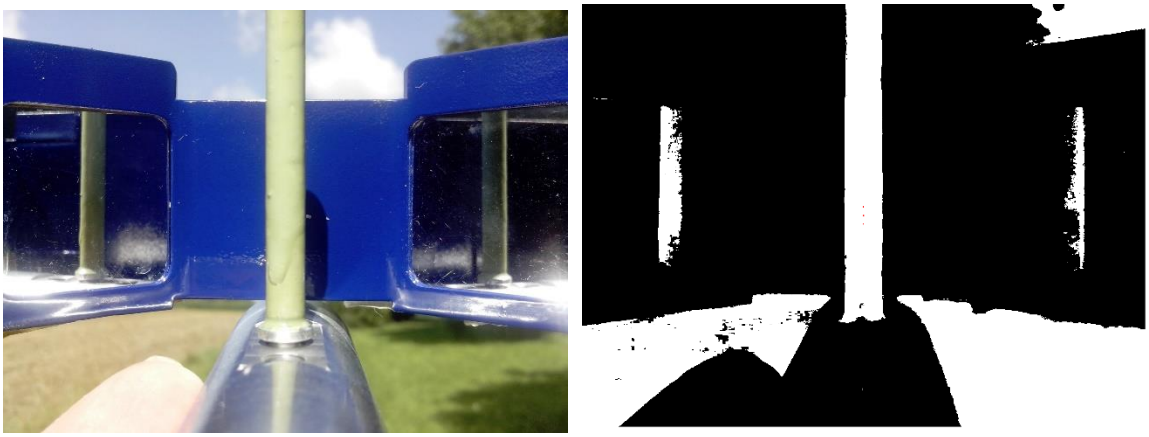
Het testen van de beedanalyse is gedaan op een computer d.m.v. een Java unittest. Het platform maakt in dit geval niet uit, aangezien een ander platform dezelfde resultaten zouden opleveren dankzij het gebruik van Java.

8.2.2 Test data en uitkomsten

De invoerafbeeldingen zijn verkregen van de opdrachtgever. De uitvoer is gegenereerd door de ImageProcessingTest klasse. Dezelfde onderliggende code wordt ook gebruikt door de *loadtest*.







FIGUUR 21 LINKS STAAN DE INVOERAFBEELDINGEN VAN DE BEELDANALYSE, RECHTS STAAN DE RESULTATEN ALS DEZE BEWERKT ZIJN DOOR DE BEELDANALYSE.

8.2.3 Analyse testresultaten

De segmentatie van de steel van de achtergrond lijkt succesvol te verlopen. Uit de gesegmenteerde afbeelding kan gemakkelijk de dikte van de steel uitgelezen worden. De stelen te zien in de spiegels worden nog niet goed genoeg gesegmenteerd. In het doel van deze test is dat echter niet van belang.

In de eerste twee bovenste afbeeldingen is te zien dat de steel scheef staat. De rode stippen geven drie meetpunten aan, waarmee de hoek gemeten wordt. Door de metingen verder van elkaar te spreiden kan een hogere nauwkeurigheid behaald worden. De gemeten hoek van de oriëntatie is in dit geval 18,8 graden, wat accuraat is.

Bij de onderste twee afbeeldingen wordt de dikte van de steel in pixels afgelezen. De dikte bedraagt in dit geval 220 pixels bij een resolutie van 3264 x 2448, waarbij de breedte van belang is. De diameter van de steel is 5,1 millimeter. Met 220 pixels geeft dat een nauwkeurigheid van 0,023mm/pixel. Daarmee wordt ruim voldaan aan de eis van minimaal 0,1mm/pixel.

8.3 Testen van snelheid en vertraging

In dit hoofdstuk worden testen verricht om de snelheid en vertraging van de software voor contactloze kopdiktemeter te meten.

8.3.1 Omstandigheden

Test hardware

Voor de volgende testen is een Motorola Moto G 2014 (2nd edition) Android telefoon gebruikt. Daarnaast is de werking van de applicatie ook getest op de Wolfgang telefoon, maar er zijn geen testresultaten vastgelegd. De Wolfgang telefoon heeft vergelijkbare specificaties voor relevante componenten, zoals de cameraresolutie en de processor.

De client draait op een Pentium dual-core PC met 3.2 Ghz. Het besturingssysteem is Ubuntu 14.04 LTS. Er is getest met de Java client.

Voor het opzetten van een USB-verbinding wordt een microUSB-kabel gebruikt.

Test software

Voor de volgende testen is gebruik gemaakt van de LibGDX log functionaliteit. Deze log functionaliteit geeft de exacte tijd weer wanneer een regel aan de log wordt toegevoegd. De regels die worden toegevoegd bevatten een beschrijving van de huidige stap in het proces, zoals dat zojuist een foto is gemaakt of dat de verzending over het netwerk is afgerond.

De tijden op de Android-telefoon worden gemeten d.m.v. ADB-debugging.

Test data

In de bijlage op pagina 64 zijn de log-bestanden weergegeven die zijn gebruikt om de tijden te meten. Aangezien de snelheid van de applicatie niet een hoge prioriteit heeft, is er niet gewerkt aan een complete en gemakkelijk bruikbare meetfunctionaliteit. Vandaar dat de tijden uit de logbestanden verkregen zijn.

De tijden zijn een momentopname en zijn dus niet een gemiddelde. Daardoor kunnen de gemiddelde tijden afwijken van de getoonde tijden. Tevens kunnen de metingen soms afwijken, doordat er gebruik wordt gemaakt van threading. Een voorbeeld hiervan is dat een uitgevoerde taak die later is uitgevoerd, toch voorrang krijgt van een oudere taak. Er is wel geprobeerd dit te minimaliseren.

Het is lastig om de tijden bij het gebruik van verzending (van JPEG of lijnen) over USB-verbinding te meten. De reden daarvoor is dat de ADB-logs niet uitgelezen kunnen worden via USB, als de USB-verbinding al gebruikt wordt voor een netwerkverbinding. Eventueel zou dan een ADB-verbinding via Wifi opgezet kunnen worden.

VERTRAGING

De tijd is gemeten van het maken van de foto tot na de verwerking van de foto. Het gaat hier om de tijd van het maken van de foto tot het verzenden van de foto. De tijd die in beslag wordt genomen door het netwerk wordt niet meegeteld.

TIJD TUSSEN FOTO'S

Bij deze test wordt de tijd gemeten van het maken van een foto tot het maken van de volgende foto. De foto's worden op een aparte thread gemaakt. Doordat de foto's op een aparte thread gemaakt worden, wordt er voor gezorgd dat er geen extra vertraging optreedt bij de andere processen. De snelheid van het maken van de foto's heeft geen invloed op de snelheid van het verzenden en ontvangen van de resultaten.

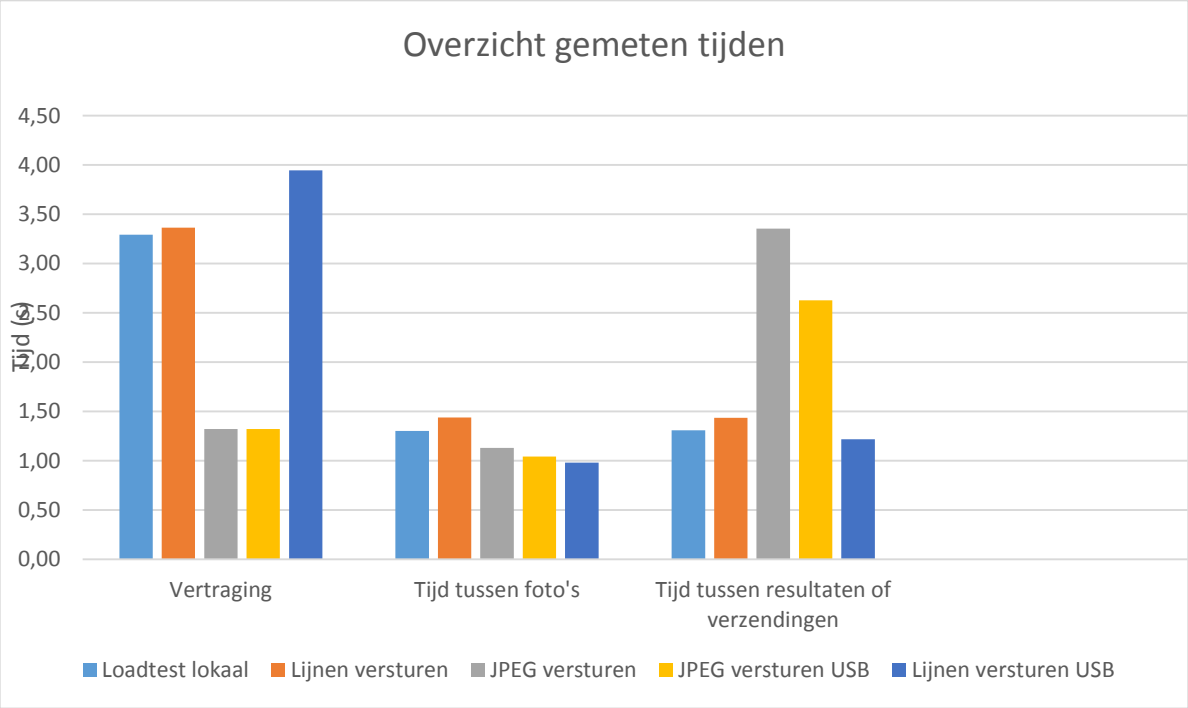
TIJD TUSSEN RESULTATEN EN OF VERZENDINGEN

Bij deze test wordt de tijd gemeten tussen de ontvangen of berekende resultaten. De tijden worden gemeten op de client-zijde indien van toepassing.

8.3.2 Uitkomsten van prestatie

De onderstaande grafiek toont de samengevoegde resultaten van de gemeten tijden in de testen. In de volgende hoofdstukken wordt getoond hoe deze resultaten zijn verkregen en berekend.

TABEL 2: OVERZICHT VAN DE GEMETEN TIJDEN



TABEL 3: TESTRESULTATEN

	Loadtest lokaal	Lijnen versturen	Lijnen versturen USB	JPEG versturen	JPEG versturen USB
Vertraging	3,29	3,37	3,95	1,32	1,32
Tijd tussen foto's	1,30	1,44	0,98	1,13	1,04
Tijd tussen resultaten of verzendingen	1,31	1,44	1,22	3,36	2,63

Voor het een beschrijving van het testraamwerk zie pagina 85 in de bijlage.

8.3.3 Analyse testresultaten

Uit de testen zijn meerdere conclusies te trekken:

Ten eerste is het uitvoeren van de beeldanalyse op de Android telefoon even snel als het uitvoeren van de beeldanalyse op een PC nadat de lijnen over het netwerk zijn verstuurd.

Ten tweede leidt het verzenden van de totale JPEG over het netwerk tot weinig belasting op de telefoon, wat leidt tot een lage vertraging van het maken van de foto tot het verzenden. Er is te zien dat het decomprimeren van de foto erg veel tijd in beslag neemt.

Vervolgens is de tijd tussen het maken van de foto's nagenoeg constant voor alle testen. Dit komt omdat de foto's op een aparte thread worden gemaakt.

Ten slotte neemt het versturen van de gehele JPEG veel tijd in beslag. Voor USB zijn de gemeten tijden (eis C.3) maar iets lager dan de gemeten tijden bij Wifi, wat onverwacht was. Dit komt waarschijnlijk omdat de USB verbinding afgeknepen wordt. Wellicht dat een ander besturingssysteem, bv. Windows, snellere resultaten zal opleveren. Een speciale applicatie gebruiken voor het USB-tethering, zoals PDANet [22], zou ook kunnen helpen.

9 Conclusie

In dit verslag is beschreven welke technieken het meest geschikt zijn om camerabeelden van een Android telefoon te faciliteren voor verdere beeldanalyse.

De techniek die gebruikt is om afbeeldingen te maken is een fotostream. Er worden met deze techniek continu foto's gemaakt, die (in het geval van de Wolfgang telefoon) gedecomprimeerd worden en over het netwerk worden verstuurd. Het decomprimeren is belastend voor de processor van de telefoon, waardoor de applicatie vertraagt. Indien mogelijk zou dit vermeden moeten worden om de applicatie sneller te maken. Helaas heeft de Wolfgang Android telefoon alleen de mogelijkheid om de foto's gecomprimeerd als JPEG op te leveren, waardoor decompressie onvermijdelijk is.

Om zo nauwkeurig en snel mogelijk de informatie uit te wisselen tussen de Android applicatie en computer, zijn er twee technieken toegepast: threading en asynchrone netwerkcommunicatie. Bij threading worden taken parallel uitgevoerd om de processor beter te benutten. Asynchrone netwerkcommunicatie zorgt ervoor dat opdrachten door host en client sneller kunnen worden afgehandeld terwijl dataoverdracht plaatsvindt.

Tijdens de ontwikkeling van de applicatie is onderzocht dat het ook mogelijk is lokaal op de Android telefoon de beeldverwerking te laten plaatsvinden. Om te bewijzen dat dit mogelijk is, is een loadtest geschreven aan de hand van een deel van de code van de beeldanalyse van de opdrachtgever. Bij deze loadtest worden lijnen van de camera afbeeldingen uitgelezen om beeldanalyse op uit te voeren. Tot het verwerken van 24 lijnen per seconde gaat dit in ieder geval goed, waarmee bewezen is dat er kan worden voldaan aan de eis om één afbeelding (van minimaal 9 lijnen) per seconde te kunnen analyseren. Er treedt geen extra vertraging op ten opzichte van beeldanalyse via een netwerkverbinding, aangezien de beeldanalyse gebeurt in de tijd dat de foto gemaakt, gecomprimeerd en gedecomprimeerd wordt.

Om de gebruiker te informeren van de resultaten van de analyse wordt de gemaakte foto na decompressie ingelezen en verkleind om verzonden te worden naar de GPU voor presentatie. Er is diverse andere (tekstuele) informatie op het scherm aanwezig, zoals bijvoorbeeld de log, om de voortgang van de analyse weer te geven. Behalve de uitkomst van de beeldanalyse en logs wordt ook het IP-adres van de Android telefoon vermeld.

Om de applicatie te configureren zijn diverse extra functionaliteiten toegevoegd, zoals de rotatie van de camera en getoonde afbeeldingen, flitser en een Wifi-hotspot. Een deel van deze functionaliteiten kan in een configuratiebestand ingesteld kan worden. Deze configuratiefunctionaliteiten moeten voor het compileren van het programma gewijzigd worden en kunnen niet zonder hercompilatie achteraf worden aangepast. Variabelen die vaker moeten worden aangepast, zoals het aantal en welke lijnen moeten worden verstuurd, kunnen over de netwerkverbinding vanaf de client-applicatie worden meegeven.

Voor debugging zijn twee testroutines ingebouwd in de applicatie: de loadtest en een test om de nauwkeurigheid van de beeldanalyse van de loadtest te bepalen. Daarnaast wordt er data over het netwerk verstuurd van de accelerometer (gemiddelde snelheid) en de tijdsduur van analyse. Voor verdere beeldanalyse kan over de netwerkverbinding vanaf de client-applicatie de gehele JPEG afbeelding worden opgevraagd in plaats van enkele lijnen.

10 Aanbevelingen

Gedurende het ontwikkelen van de applicatie zijn keuzes gemaakt voor technieken. Daarbij zijn er soms afwegingen gemaakt in verband met de beschikbaarheid van ontwikkeltools, de prestaties van hardware en de gewenste oplevertijd van de applicatie. In dit hoofdstuk worden aanbevelingen gedaan voor verdere ontwikkeling.

10.1 Device / apparaat

Nieuwere, duurdere Android telefoons hebben niet altijd een veel hogere resolutie beschikbaar. Er is dus niet per se een reden een dure telefoon te gebruiken.

De snelheid van het maken van de foto's kan erg variëren, evenals de aanwezige processorkracht. Dit kan betekenen dat de processen sneller kunnen verlopen of minder stroom gebruiken bij een betere telefoon.

Sommige toestellen (b.v. Nexus 5 en Nexus 6) hebben de optie de foto's op te leveren in RAW formaat, wat betekent dat er geen extra tijd voor de compressie en decompressie van de JPEG afbeelding hoeft te worden berekend. Dit maakt veel uit voor het aantal gemaakte foto's per seconde (snelheid) en de vertraging tussen het maken van de foto en uitlezen van de data (latency). De RAW ondersteuning in Android staat wel nog in de kinderschoenen.

De iPhone ondersteunt het TIFF formaat, wat ook een ongecomprimeerd formaat is en dus ook geschikt voor deze applicatie. De ontwikkelde applicatie is redelijk eenvoudig om te zetten naar een iOS applicatie. Het is wel nog nodig om platformafhankelijke functionaliteit, zoals de camera, toe te voegen aan de applicatie.

10.2 Lokale berekening

Er wordt aangeraden om de berekeningen lokaal op de Android telefoon te laten plaatsvinden en verder uit te ontwikkelen. Dit vermindert de complexiteit en maakt het geheel gebruiksvriendelijker. Tevens ontstaat er betere draagbaarheid, betere bereikbaarheid en minder storingsgevoeligheid.

Aanvankelijk zal het een redelijke klus zijn om de beeldanalyse zo aan te passen dat deze te draaien is op Android op een zuinige manier, maar uiteindelijk is het een duidelijke aanwinst.

10.3 Videostream en foto -combinatie

Het is mogelijk om het algoritme dat de positie, rotatie en dikte van de steel berekent te splitsen in twee delen. Aangezien de positie en rotatie hoogstwaarschijnlijk niet heel nauwkeurig hoeven te zijn, zou dit kunnen gebeuren d.m.v. een videostream (op 1280x720 resolutie) in plaats van een serie foto's met een veel lagere snelheid (op 3264x2448 resolutie).

Op deze manier kan als de rotatie en positie goed staan gezien aan de hand van de videostream, pas een foto gemaakt worden, waarmee de dikte van de steel berekend wordt.

Dit bespaart op rekentijd en zorgt voor een vloeiender draaiende applicatie, aangezien het aantal verwerkte beelden per seconde veel hoger ligt.

Dit is echter een compleet andere aanpak dan de aanpak die oorspronkelijk bedacht is. Er is met de heer Lambers besproken om deze methode niet nu toe te passen.

10.4 Videostream

Als in de praktijk de frequentie van de beelden hoger moet zijn en bovenstaande techniek niet uitgevoerd kan worden, is het misschien beter om toch gebruik te maken van een videostream. De vereiste hiervoor is dat een telefoon met video opname van een resolutie van 1920x1080. Tegenwoordig zijn er veel telefoons beschikbaar welke deze resolutie of een hogere resolutie ondersteunen. De Wolfgang telefoon die in dit project ondersteund diende te worden kan dat echter niet.

10.5 Verhoging nauwkeurigheid met meerdere metingen

In theorie kan de nauwkeurigheid van de metingen verhoogd worden door het gemiddelde van meerdere metingen te nemen. Een dergelijke methode heet *subpixel analyse* (Engels: subpixel analysis). In dit project is daar nog niet naar gekeken.

Er zijn voor en nadelen aan het gebruik van subpixel analyse:

Een voordeel is dat bij het gebruik van deze methode kunnen videobeelden in plaats van foto's gebruikt worden. De resolutie is namelijk niet de meest bepalende factor, maar wel de hoeveelheid afbeeldingen. Android stelt videobeelden ongecomprimeerd toegankelijk, terwijl foto's gecomprimeerd opgeleverd worden. In het geval van gebruik van videobeelden is geen decompressie nodig.

Een nadeel is dat in het slechtste scenario de nauwkeurigheid niet beter wordt. Meer metingen kunnen er wel voor zorgen dat het slechtste scenario minder snel voor komt. En dan is er voor dit project, als er videobeelden gebruikt worden, een te lage resolutie beschikbaar.

Met een meting d.m.v. meerdere beelden is de fout te reduceren tot 3 maal de standaard deviatie uit N beelden.[8]

10.6 Videobeelden met OpenCV

Bij het gebruiken van videobeelden in plaats van foto's kan het handig zijn om het OpenCV framework te gebruiken.

OpenCV (Open Source Computer Vision Library) is een open source *computer vision* en *machine vision* bibliotheek. OpenCV is gebouwd om een gemeenschappelijke infrastructuur op te zetten voor applicaties met visie. OpenCV is gelicenseerd volgens de BSD-licentie en OpenCV is daardoor vrij bruikbaar (ook voor commerciële toepassingen). Hierdoor is het interessant om toe te passen in een applicatie die mogelijk ook commercieel gaat worden ingezet (zoals de kopdiktemeter). [23]

10.7 Gebruik USB-kabel

Door de telefoon via een microUSB-kabel te verbinden met de externe computer kan er meer en sneller data uitgewisseld worden. De bandbreedte neemt significant toe, waardoor JPEG-afbeeldingen zelfs in hun geheel verzonden kunnen worden en sneller gedecomprimeerd kunnen worden op de externe computer.

Een dergelijke opzet is gemakkelijk te implementeren door *USB Tethering*[9] aan te zetten op de Android telefoon en de telefoon te verbinden met een USB-kabel. Er wordt dan een netwerkverbinding opgezet. De software van dit project is daarmee dan ook gewoon te gebruiken.

Er zou bijvoorbeeld een laptop met een goede, snelle processor in een rugtas verbonden kunnen worden met de Android telefoon d.mv. een microUSB-kabel. Er is dan geen sprake van een draadloze verbinding, waarmee ook geen problemen door interferentie met het draadloze signaal zouden kunnen optreden. Tevens wordt er bespaard op stroomconsumptie door gebruik te maken van een verbinding via USB.

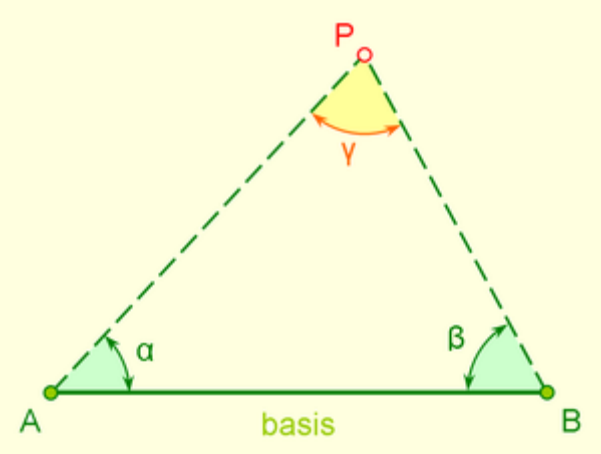
10.8 Voorwaartse insnijding

Een wellicht makkelijker alternatief voor achterwaartse insnijding, is voorwaartse insnijding. Voorwaartse insnijding wordt ook veel gebruikt in de landmeetkunde. Hierbij wordt uit twee bekende punten een derde onbekend punt berekend (de methode van de achterwaartse insnijding gebruikt minstens drie bekende punten). [24]

---

[8] Bron: Jan Lambers

[9] In Android te vinden door de volgende stappen in het menu te maken: Settings -> Wireless settings -> Tethering -> Tethering USB. Dit kan echter verschillen per Android versie.



FIGUUR 22 DE VOORWAARTSE INSNIJDING BEREKENT PUNT P D.M.V. PUNT A EN B.

Telefoon met stereoscopische camera

Een telefoon uitgerust met twee camera's die naar voren kijken zou een voorwaartse insnijding kunnen uitvoeren zonder extra hardware toevoegingen aan de telefoon. Tegenwoordig zijn er telefoons uitgerust met stereoscopische camera's die daar aan kunnen voldoen. Voorbeelden van zulke Android telefoons zijn de HTC Evo 3D, de LG Optimus 3D MAX, de LG Optimus 3D P920, de Sharp Aquos SH-12C of SH-80F of de Samsung SCH-B710. De software van de contactloze kopdiktemeter kan aangepast worden om gebruik te maken van beide camera's.

Nintendo 3DS

Noemenswaardig is dat de Nintendo 3DS spelcomputer ook twee camera's bevat die naar voren kijken. Door een interface te leggen van de SD-kaart-slot naar een computer zouden theoretisch in realtime de foto's opgevangen kunnen worden. Een alternatief zijn misschien SD-kaarten met Wifi ondersteuning. De resolutie van de Nintendo 3DS is onvoldoende voor een fotostream met een resolutie van 640x480 per camera. Een videostream zou misschien wel kunnen (met ook een resolutie van 640x480) als het gemiddelde van de metingen over meerdere frames wordt genomen.

De Nintendo 3DS ondersteunt ook *homebrew*, oftewel zelfgeschreven applicaties. [25]

11 Reflectie

11.1 Evaluatie

Deze reflectie gaat over het onderzoeken, het prototyping-proces en het uitwerken van software voor de contactloze kopdiktemeter.

Bij de voorbereiding is gekeken naar de methodiek de planning en de gebruikte computertechnieken. De gekozen methodiek (Prototyping), was goed en paste goed bij het project. De planning was niet uitgebreid genoeg. Bovendien was er niet voor alle taken voldoende tijd beschikbaar. Vaak duurde het onderzoeken en het maken van prototypes langer dan verwacht. De computertechnieken waren vooraf goed gekozen.

Op voorhand waren de eisen niet altijd duidelijk genoeg. Gaandeweg werden de eisen duidelijker, maar soms ook uitgebreider. Daardoor liep de planning soms ook mis. Het was beter geweest om de eisen vooraf meer nadrukkelijk te definiëren.

Daarnaast bleek achteraf dat het opstellen van het verslag meer tijd in beslag nam dan aanvankelijk gedacht. Hoofdzakelijk omdat het verslag niet op tijd af was moest er uitstel voor het afstuderen aangevraagd. Dit had voorkomen kunnen worden door eerder aan het verslag te beginnen.

Een volgende keer zou ik zorgen dat de eisen vanaf het begin vastliggen en duidelijk zijn. Daarmee wordt de planning ook stabieler, evenals de haalbaarheid van het project. In dit project waren de eisen vooraf niet geheel duidelijk, maar een betere analyse van de exacte eisen zou wel geholpen hebben.

Tevens zou ik voldoende tijd besteden aan het rapport en hier op tijd aan beginnen.

11.2 Competenties

11.2.1 Analyseren van het probleemdomein

“Het onderzoeken en beschrijven van het gegeven probleemdomein (van de opdrachtgever) in termen van het TI domein, en het vaststellen van de gewenste veranderingen. Het inwerken in het domein van de opdrachtgever, maakt onderdeel van deze taak uit.”

- Analyseren van de te implementeren techniek. Misschien moet het algoritme overgezet worden naar een andere taal; hier voor moet de werking van het algoritme van te voren goed begrepen worden.
- Het uitzoeken van een goede en gebruiksvriendelijke opstelling. Er worden enkele keuzes gemaakt, zoals het gebruik van OpenGL of een videostream voor de visualisatie van de steel.
- De prestatie van de verbinding tussen een Android telefoon en een externe computer in een realistisch scenario onderzoeken. Is het de meeste efficiënte en gewilde methode om het hoofdalgoritme via het draadloos netwerk uit te voeren?

11.2.2 Creëren en innoveren

“Nieuwe en/of ongebruikelijke oplossingen bedenken voor gegeven situaties. Een originele bijdrage leveren in een brainstormsessie. Nieuwe producten bedenken op basis van bestaande technologie.”

- Uitzoeken wat de efficiëntste methodes zijn om desbetreffend algoritme van de techniek uit te voeren. Hiervoor moet worden gekeken naar verschillende platformen en of het algoritme geschikt is voor deze platformen (b.v. of het algoritme parallel kan draaien onder OpenCL op een GPGPU).
- Construeren van een gebruiksvriendelijke ervaring voor de bediening van het complete systeem. Hier voor moeten enkele testen uitgevoerd worden aan de hand van criteria. De criteria zijn o.a. vertraging/gemak in gebruik en instapniveau. Aangezien de te ontwikkelen applicatie een nieuw doel dient, moeten er ook nieuwe oplossingen bedacht worden voor de bediening daar van.

- Op de langere termijn, kunnen er ideeën of problemen opduiken welke aangepakt moeten worden. Is het bijvoorbeeld gewenst dat er een hoes over de telefoon gaat, zodat deze niet nat wordt? In dat geval moet de bediening wellicht niet door middel van het touchscreen gebeuren.

11.2.3 Het realiseren van software

"Het programmeren van componenten van software."

- Overzetten van het hoofdalgoritme van de techniek naar een ander platform naar aanleiding van het onderzoek over welk platform (b.v. OpenCL) het meest geschikt is. Of het eventueel schrijven van een wrapper indien nodig en als het algoritme voldoende uitgeschreven is.
- Iteratief ontwikkelen van de software aan de hand van gemaakte keuzes. De keuzes voor de ontwikkeling zijn gemaakt op basis van het testen van prototypes. De ontwikkelde software wordt iteratief getest, en er wordt gestreefd naar zo hoog mogelijke efficiëntie en prestatie.
- Realiseren van een protocol voor de informatie-uitwisseling tussen de Android telefoon en de externe computer.
- Testen van het complete systeem en de prestatie ervan. Testen d.m.v. diverse testpersonen en zo nodig veranderingen aanbrengen.

12 Bibliografie

[1] 12 3 2015. [Online]. Available: http://www.inholland.nl/NR/rdonlyres/8D38969D-0CD4-4C7A-958D-98EC689E26EF/0/HetprojectGreenEyes.pdf.

[2] 3 4 2015. [Online]. Available: http://www.bloodshed.net/devcpp.html.

[3] T. C. S. K. e. A. J. Ruben van Gulik, „Contactloze diktebepaling van tomatenstengels door segmentatie en lijndetectie," 2014.

[4] 15 3 2015. [Online]. Available: http://www.javaworld.com/article/2077184/core-java/the-lean--mean--virtual-machine.html.

[5] 16 3 2015. [Online]. Available: https://en.wikipedia.org/wiki/Android_%28operating_system%29.

[6] 12 3 2015. [Online]. Available: http://www.glprogramming.com/blue/ch01.html.

[7] 5 4 2015. [Online]. Available: https://github.com/libgdx/libgdx/wiki/Introduction.

[8] 3 4 2015. [Online]. Available: https://nl.wikipedia.org/wiki/MoSCoW-methode.

[9] J. A. e. J. Ebbinge, Inleiding Landmeetkunde, Delft: de VSSD, 2003-2006.

[10] M. e. J.J.Derwael, „Drie dimentionale metingen van horizontale boringen," SCK/CEN, Studiegroep Omgeving, 1993.

[11] 4 4 2015. [Online]. Available: http://www.steves-digicams.com/knowledge-center/how-tos/digital-camera-operation/ccd-vs-cmos-whats-the-difference.html.

[12] 15 3 2015. [Online]. Available: https://en.wikipedia.org/wiki/JPEG.

[13] B. W. C. J. H. K. Gil Jin Yang, „Implementation of HTTP Live Streaming for an IP Camera using an," Dept. of Electrical and Information Engineering, Seoul National Univ. of Science, Zuid Korea, 2014.

[14] 1 4 2015. [Online]. Available: http://www.impulseadventure.com/photo/jpeg-huffman-coding.html.

[15] 2 4 2015. [Online]. Available: https://nl.wikipedia.org/wiki/Verzeiling.

[16] 5 4 2015. [Online]. Available: https://nl.wikipedia.org/wiki/Ontwerppatroon.

[17] 5 4 2015. [Online]. Available: https://en.wikipedia.org/wiki/Template_method_pattern.

[18] 4 4 2015. [Online]. Available: : http://www.celinio.net/techblog/wp-content/uploads/2009/09/designpatterns1.jpg.

[19] 5 4 2015. [Online]. Available: https://en.wikipedia.org/wiki/Singleton_pattern.

[20] 4 4 2015. [Online]. Available: : http://www.celinio.net/techblog/wp-content/uploads/2009/09/designpatterns2.jpg.

[21] 5 4 2015. [Online]. Available: https://en.wikipedia.org/wiki/Flyweight_pattern.

[22] 10 5 2015. [Online]. Available: https://play.google.com/store/apps/details?id=com.pdanet&hl=en.

[23] 13 3 2015. [Online]. Available: http://opencv.org/about.html.

[24] P. W. Schermerhorn, Landmeten en waterpassen, Delft: N.V. WED. J. Ahrend & Zoon, 1943.

[25] „Ninjhax,” 27 5 2015. [Online]. Available: http://smealum.net/ninjhax/.

[26] „RoboVM,” 5 3 2015. [Online]. Available: http://robovm.com/.

13 Bijlagen

13.1 Log-bestand voor JPEG verzending over Wifi

De werking van de telefoon heeft onderstaande log-bestanden opgeleverd. Deze bestanden zijn gebruikt om de testen op te baseren. Dit log-bestand beschrijft de processen, terwijl JPEG afbeeldingen over een Wifi-verbinding worden verzonden.

03-24 12:10:20.843: I/Claw(24220): Accept server socket.
03-24 12:10:20.843: I/Claw(24220): Test line count: 2168715
03-24 12:10:20.843: I/Claw(24220): Data to write: 2168736
03-24 12:10:21.346: I/Claw(24220): Picture data received, id: 58857
03-24 12:10:21.346: I/Claw(24220): Process data with ID: 58857
03-24 12:10:21.346: I/Claw(24220): DataManager done, id: 58857
03-24 12:10:21.348: I/Claw(24220): Do take picture, id: 17372
03-24 12:10:21.358: I/Claw(24220): Done processing with ID: 58857
03-24 12:10:21.769: I/Claw(24220): Actual take picture done, id: 17372
03-24 12:10:22.653: I/Claw(24220): Picture data received, id: 17372
03-24 12:10:22.653: I/Claw(24220): Process data with ID: 17372
03-24 12:10:22.654: I/Claw(24220): DataManager done, id: 17372
03-24 12:10:22.657: I/Claw(24220): Do take picture, id: 13108
03-24 12:10:22.669: I/Claw(24220): Done processing with ID: 17372
03-24 12:10:23.081: I/Claw(24220): Actual take picture done, id: 13108
03-24 12:10:23.992: I/Claw(24220): Picture data received, id: 13108
03-24 12:10:23.992: I/Claw(24220): Process data with ID: 13108
03-24 12:10:23.992: I/Claw(24220): DataManager done, id: 13108
03-24 12:10:23.993: I/Claw(24220): Do take picture, id: 63334
03-24 12:10:24.005: I/Claw(24220): Done processing with ID: 13108
03-24 12:10:24.250: I/Claw(24220): Sent data from server socket.
03-24 12:10:24.250: I/Claw(24220): Sending..
03-24 12:10:24.405: I/Claw(24220): Actual take picture done, id: 63334
03-24 12:10:24.632: I/Claw(24220): Accept server socket.
03-24 12:10:24.632: I/Claw(24220): Test line count: 2171435
03-24 12:10:24.632: I/Claw(24220): Data to write: 2171456
03-24 12:10:25.297: I/Claw(24220): Picture data received, id: 63334
03-24 12:10:25.297: I/Claw(24220): Process data with ID: 63334
03-24 12:10:25.298: I/Claw(24220): DataManager done, id: 63334
03-24 12:10:25.302: I/Claw(24220): Do take picture, id: 61278
03-24 12:10:25.310: I/Claw(24220): Done processing with ID: 63334
03-24 12:10:25.717: I/Claw(24220): Actual take picture done, id: 61278
03-24 12:10:26.610: I/Claw(24220): Picture data received, id: 61278
03-24 12:10:26.610: I/Claw(24220): Process data with ID: 61278
03-24 12:10:26.611: I/Claw(24220): DataManager done, id: 61278
03-24 12:10:26.619: I/Claw(24220): Do take picture, id: 21898
03-24 12:10:26.629: I/Claw(24220): Done processing with ID: 61278
03-24 12:10:27.035: I/Claw(24220): Actual take picture done, id: 21898
03-24 12:10:27.811: I/Claw(24220): Sent data from server socket.
03-24 12:10:27.811: I/Claw(24220): Sending..
03-24 12:10:27.920: I/Claw(24220): Picture data received, id: 21898
03-24 12:10:27.920: I/Claw(24220): Process data with ID: 21898
03-24 12:10:27.920: I/Claw(24220): DataManager done, id: 21898

03-24 12:10:27.926: I/Claw(24220): Do take picture, id: 13104
03-24 12:10:27.929: I/Claw(24220): Done processing with ID: 21898
03-24 12:10:28.309: I/Claw(24220): Accept server socket.
03-24 12:10:28.309: I/Claw(24220): Test line count: 2165355
03-24 12:10:28.309: I/Claw(24220): Data to write: 2165376
03-24 12:10:28.357: I/Claw(24220): Actual take picture done, id: 13104
03-24 12:10:29.260: I/Claw(24220): Picture data received, id: 13104
03-24 12:10:29.260: I/Claw(24220): Process data with ID: 13104
03-24 12:10:29.260: I/Claw(24220): DataManager done, id: 13104
03-24 12:10:29.266: I/Claw(24220): Do take picture, id: 7012
03-24 12:10:29.276: I/Claw(24220): Done processing with ID: 13104
03-24 12:10:29.674: I/Claw(24220): Actual take picture done, id: 7012
03-24 12:10:30.569: I/Claw(24220): Picture data received, id: 7012
03-24 12:10:30.569: I/Claw(24220): Process data with ID: 7012
03-24 12:10:30.570: I/Claw(24220): DataManager done, id: 7012
03-24 12:10:30.571: I/Claw(24220): Do take picture, id: 10184
03-24 12:10:30.583: I/Claw(24220): Done processing with ID: 7012
03-24 12:10:30.992: I/Claw(24220): Actual take picture done, id: 10184
03-24 12:10:31.450: I/Claw(24220): Sent data from server socket.
03-24 12:10:31.451: I/Claw(24220): Sending..
03-24 12:10:31.805: I/Claw(24220): Accept server socket.
03-24 12:10:31.805: I/Claw(24220): Test line count: 2163776
03-24 12:10:31.805: I/Claw(24220): Data to write: 2163797
03-24 12:10:31.886: I/Claw(24220): Picture data received, id: 10184
03-24 12:10:31.886: I/Claw(24220): Process data with ID: 10184
03-24 12:10:31.887: I/Claw(24220): DataManager done, id: 10184
03-24 12:10:31.895: I/Claw(24220): Done processing with ID: 10184
03-24 12:10:31.898: I/Claw(24220): Do take picture, id: 66695
03-24 12:10:32.309: I/Claw(24220): Actual take picture done, id: 66695
03-24 12:10:33.204: I/Claw(24220): Picture data received, id: 66695
03-24 12:10:33.204: I/Claw(24220): Process data with ID: 66695
03-24 12:10:33.204: I/Claw(24220): DataManager done, id: 66695
03-24 12:10:33.209: I/Claw(24220): Do take picture, id: 67387
03-24 12:10:33.212: I/Claw(24220): Done processing with ID: 66695
03-24 12:10:33.626: I/Claw(24220): Actual take picture done, id: 67387
03-24 12:10:34.535: I/Claw(24220): Picture data received, id: 67387
03-24 12:10:34.535: I/Claw(24220): Process data with ID: 67387
03-24 12:10:34.536: I/Claw(24220): DataManager done, id: 67387
03-24 12:10:34.539: I/Claw(24220): Do take picture, id: 81050
03-24 12:10:34.548: I/Claw(24220): Done processing with ID: 67387
03-24 12:10:34.949: I/Claw(24220): Actual take picture done, id: 81050
03-24 12:10:35.011: I/Claw(24220): Sent data from server socket.
03-24 12:10:35.011: I/Claw(24220): Sending..
03-24 12:10:35.284: I/Claw(24220): Accept server socket.
03-24 12:10:35.285: I/Claw(24220): Test line count: 2171148
03-24 12:10:35.285: I/Claw(24220): Data to write: 2171169
03-24 12:10:35.844: I/Claw(24220): Picture data received, id: 81050
03-24 12:10:35.844: I/Claw(24220): Process data with ID: 81050
03-24 12:10:35.844: I/Claw(24220): DataManager done, id: 81050

03-24 12:10:35.850: I/Claw(24220): Do take picture, id: 95303
03-24 12:10:35.858: I/Claw(24220): Done processing with ID: 81050
03-24 12:10:36.259: I/Claw(24220): Actual take picture done, id: 95303
03-24 12:10:37.155: I/Claw(24220): Picture data received, id: 95303
03-24 12:10:37.155: I/Claw(24220): Process data with ID: 95303
03-24 12:10:37.155: I/Claw(24220): DataManager done, id: 95303
03-24 12:10:37.159: I/Claw(24220): Do take picture, id: 68584
03-24 12:10:37.169: I/Claw(24220): Done processing with ID: 95303
03-24 12:10:37.590: I/Claw(24220): Actual take picture done, id: 68584
03-24 12:10:38.272: I/Claw(24220): Sent data from server socket.
03-24 12:10:38.272: I/Claw(24220): Sending..
03-24 12:10:38.473: I/Claw(24220): Picture data received, id: 68584
03-24 12:10:38.473: I/Claw(24220): Process data with ID: 68584
03-24 12:10:38.474: I/Claw(24220): DataManager done, id: 68584
03-24 12:10:38.478: I/Claw(24220): Do take picture, id: 97172
03-24 12:10:38.484: I/Claw(24220): Done processing with ID: 68584
03-24 12:10:38.598: I/Claw(24220): Accept server socket.
03-24 12:10:38.604: I/Claw(24220): Test line count: 2185500
03-24 12:10:38.607: I/Claw(24220): Data to write: 2185521
03-24 12:10:38.902: I/Claw(24220): Actual take picture done, id: 97172
03-24 12:10:39.795: I/Claw(24220): Picture data received, id: 97172
03-24 12:10:39.795: I/Claw(24220): Process data with ID: 97172
03-24 12:10:39.795: I/Claw(24220): DataManager done, id: 97172
03-24 12:10:39.805: I/Claw(24220): Do take picture, id: 21529
03-24 12:10:39.807: I/Claw(24220): Done processing with ID: 97172
03-24 12:10:40.213: I/Claw(24220): Actual take picture done, id: 21529
03-24 12:10:41.108: I/Claw(24220): Picture data received, id: 21529
03-24 12:10:41.108: I/Claw(24220): Process data with ID: 21529
03-24 12:10:41.109: I/Claw(24220): DataManager done, id: 21529
03-24 12:10:41.113: I/Claw(24220): Do take picture, id: 88410
03-24 12:10:41.118: I/Claw(24220): Done processing with ID: 21529
03-24 12:10:41.533: I/Claw(24220): Actual take picture done, id: 88410
03-24 12:10:41.598: I/Claw(24220): Sent data from server socket.
03-24 12:10:41.598: I/Claw(24220): Sending..
03-24 12:10:41.920: I/Claw(24220): Accept server socket.
03-24 12:10:41.920: I/Claw(24220): Test line count: 2171373
03-24 12:10:41.920: I/Claw(24220): Data to write: 2171394
03-24 12:10:42.418: I/Claw(24220): Picture data received, id: 88410
03-24 12:10:42.418: I/Claw(24220): Process data with ID: 88410
03-24 12:10:42.418: I/Claw(24220): DataManager done, id: 88410
03-24 12:10:42.421: I/Claw(24220): Do take picture, id: 14136
03-24 12:10:42.437: I/Claw(24220): Done processing with ID: 88410

13.2 Log-bestand voor lijnen verzenden over Wifi

De werking van de telefoon heeft onderstaande log-bestanden opgeleverd. Deze bestanden zijn gebruikt om de testen op te baseren. Dit log-bestand beschrijft de processen, terwijl er lijnen worden verzonden over een Wifi-verbinding.

03-24 12:09:39.046: I/Claw(24220): Picture data received, id: 25623
03-24 12:09:39.046: I/Claw(24220): Process data with ID: 25623

03-24 12:09:39.047: I/Claw(24220): DataManager done, id: 25623
03-24 12:09:39.054: I/Claw(24220): Do take picture, id: 13012
03-24 12:09:39.463: I/Claw(24220): Actual take picture done, id: 13012
03-24 12:09:39.738: I/Claw(24220): Count lines: 32
03-24 12:09:39.751: I/Claw(24220): Pointer: 313497
03-24 12:09:39.751: I/Claw(24220): Done processing with ID: 96312
03-24 12:09:39.759: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448
03-24 12:09:39.760: I/Claw(24220): Sending..
03-24 12:09:39.762: I/Claw(24220): Accept server socket.
03-24 12:09:39.763: I/Claw(24220): Test line count: 32
03-24 12:09:39.763: I/Claw(24220): Data to write: 313497
03-24 12:09:40.231: I/Claw(24220): Sent data from server socket.
03-24 12:09:40.371: I/Claw(24220): Picture data received, id: 13012
03-24 12:09:40.371: I/Claw(24220): Process data with ID: 13012
03-24 12:09:40.372: I/Claw(24220): DataManager done, id: 13012
03-24 12:09:40.378: I/Claw(24220): Do take picture, id: 56744
03-24 12:09:40.788: I/Claw(24220): Actual take picture done, id: 56744
03-24 12:09:41.037: I/Claw(24220): Count lines: 32
03-24 12:09:41.044: I/Claw(24220): Pointer: 313497
03-24 12:09:41.044: I/Claw(24220): Done processing with ID: 25623
03-24 12:09:41.051: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448
03-24 12:09:41.051: I/Claw(24220): Sending..
03-24 12:09:41.052: I/Claw(24220): Accept server socket.
03-24 12:09:41.052: I/Claw(24220): Test line count: 32
03-24 12:09:41.052: I/Claw(24220): Data to write: 313497
03-24 12:09:41.512: I/Claw(24220): Sent data from server socket.
03-24 12:09:41.713: I/Claw(24220): Picture data received, id: 56744
03-24 12:09:41.713: I/Claw(24220): Process data with ID: 56744
03-24 12:09:41.714: I/Claw(24220): DataManager done, id: 56744
03-24 12:09:41.718: I/Claw(24220): Do take picture, id: 18648
03-24 12:09:42.185: I/Claw(24220): Actual take picture done, id: 18648
03-24 12:09:42.412: I/Claw(24220): Count lines: 32
03-24 12:09:42.418: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448
03-24 12:09:42.419: I/Claw(24220): Pointer: 313497
03-24 12:09:42.419: I/Claw(24220): Done processing with ID: 13012
03-24 12:09:42.426: I/Claw(24220): Sending..
03-24 12:09:42.426: I/Claw(24220): Accept server socket.

03-24 12:09:42.427: I/Claw(24220): Test line count: 32
03-24 12:09:42.427: I/Claw(24220): Data to write: 313497

13.3 Log-bestand voor lokale beeldanalyse

De werking van de telefoon heeft onderstaande log-bestanden opgeleverd. Deze bestanden zijn gebruikt om de testen op te baseren. Dit log-bestand beschrijft de processen, terwijl de beeldanalyse lokaal plaatsvindt op de Android telefoon.

03-24 12:09:03.401: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448
03-24 12:09:03.625: I/Claw(24220): Actual take picture done, id: 71714
03-24 12:09:04.581: I/Claw(24220): Picture data received, id: 71714
03-24 12:09:04.582: I/Claw(24220): Process data with ID: 71714

03-24 12:09:04.582: I/Claw(24220): DataManager done, id: 71714
03-24 12:09:04.591: I/Claw(24220): Do take picture, id: 77570
03-24 12:09:05.069: I/Claw(24220): Actual take picture done, id: 77570
03-24 12:09:05.118: I/Claw(24220): Done processing with ID: 67532
03-24 12:09:05.121: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448
03-24 12:09:05.997: I/Claw(24220): Picture data received, id: 77570
03-24 12:09:05.997: I/Claw(24220): Process data with ID: 77570
03-24 12:09:05.998: I/Claw(24220): DataManager done, id: 77570
03-24 12:09:06.007: I/Claw(24220): Do take picture, id: 39434
03-24 12:09:06.428: I/Claw(24220): Done processing with ID: 71714
03-24 12:09:06.434: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448
03-24 12:09:06.489: I/Claw(24220): Actual take picture done, id: 39434
03-24 12:09:07.305: I/Claw(24220): Picture data received, id: 39434
03-24 12:09:07.305: I/Claw(24220): Process data with ID: 39434
03-24 12:09:07.305: I/Claw(24220): DataManager done, id: 39434
03-24 12:09:07.312: I/Claw(24220): Do take picture, id: 80687
03-24 12:09:07.720: I/Claw(24220): Actual take picture done, id: 80687
03-24 12:09:07.885: I/Claw(24220): Done processing with ID: 77570
03-24 12:09:07.893: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448
03-24 12:09:08.638: I/Claw(24220): Picture data received, id: 80687
03-24 12:09:08.638: I/Claw(24220): Process data with ID: 80687
03-24 12:09:08.639: I/Claw(24220): DataManager done, id: 80687
03-24 12:09:08.648: I/Claw(24220): Do take picture, id: 80234
03-24 12:09:09.097: I/Claw(24220): Actual take picture done, id: 80234
03-24 12:09:09.146: I/Claw(24220): Done processing with ID: 39434
03-24 12:09:09.152: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448
03-24 12:09:09.933: I/Claw(24220): Picture data received, id: 80234
03-24 12:09:09.934: I/Claw(24220): Process data with ID: 80234
03-24 12:09:09.934: I/Claw(24220): DataManager done, id: 80234
03-24 12:09:09.936: I/Claw(24220): Do take picture, id: 48085
03-24 12:09:10.346: I/Claw(24220): Actual take picture done, id: 48085
03-24 12:09:10.453: I/Claw(24220): Done processing with ID: 80687
03-24 12:09:10.453: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448
03-24 12:09:11.239: I/Claw(24220): Picture data received, id: 48085
03-24 12:09:11.239: I/Claw(24220): Process data with ID: 48085
03-24 12:09:11.239: I/Claw(24220): DataManager done, id: 48085
03-24 12:09:11.245: I/Claw(24220): Do take picture, id: 14170
03-24 12:09:11.681: I/Claw(24220): Actual take picture done, id: 14170
03-24 12:09:11.797: I/Claw(24220): Done processing with ID: 80234
03-24 12:09:11.803: I/Claw(24220): ColorSize: 3 data: 23970816 width: 3264 height: 2448

13.4 Logbestand voor JPEG verzenden over USB

De werking van de telefoon heeft onderstaande log-bestanden opgeleverd. Deze bestanden zijn gebruikt om de testen op te baseren. Dit log-bestand beschrijft de processen, terwijl gehele JPEG-afbeeldingen worden verzonden over een USB-verbinding.

05-09 19:18:47.805: I/Claw(24918): Actual take picture done, id: 23823
05-09 19:18:48.518: I/Claw(24918): Picture data received, id: 23823
05-09 19:18:48.518: I/Claw(24918): Process data, id: 23823
05-09 19:18:48.519: I/Claw(24918): DataManager started, id: 23823

05-09 19:18:48.551: I/Claw(24918): No receiver present, skipping payload..
05-09 19:18:48.551: I/Claw(24918): Sending..
05-09 19:18:48.552: I/Claw(24918): Do take picture, id: 98123
05-09 19:18:48.563: I/Claw(24918): Done processing, id: 23823
05-09 19:18:48.772: I/Claw(24918): Actual take picture done, id: 98123
05-09 19:18:49.468: I/Claw(24918): Picture data received, id: 98123
05-09 19:18:49.468: I/Claw(24918): Process data, id: 98123
05-09 19:18:49.469: I/Claw(24918): DataManager started, id: 98123
05-09 19:18:49.478: I/Claw(24918): Done processing, id: 98123
05-09 19:18:49.550: I/Claw(24918): No receiver present, skipping payload..
05-09 19:18:49.550: I/Claw(24918): Sending..
05-09 19:18:49.594: I/Claw(24918): Do take picture, id: 61495
05-09 19:18:49.805: I/Claw(24918): Actual take picture done, id: 61495
05-09 19:18:50.523: I/Claw(24918): Picture data received, id: 61495
05-09 19:18:50.523: I/Claw(24918): Process data, id: 61495
05-09 19:18:50.524: I/Claw(24918): DataManager started, id: 61495
05-09 19:18:50.527: I/Claw(24918): Do take picture, id: 13296
05-09 19:18:50.531: I/Claw(24918): Done processing, id: 61495
05-09 19:18:50.550: I/Claw(24918): No receiver present, skipping payload..
05-09 19:18:50.550: I/Claw(24918): Sending..
05-09 19:18:50.752: I/Claw(24918): Actual take picture done, id: 13296

13.5 Logbestand voor lijnen over USB

De werking van de telefoon heeft onderstaande log-bestanden opgeleverd. Deze bestanden zijn gebruikt om de testen op te baseren. Dit log-bestand beschrijft de processen, terwijl lijnen worden verstuurd over een USB-verbinding.

05-09 19:39:25.935: I/Claw(27153): Actual take picture done, id: 45526
05-09 19:39:26.123: I/Claw(27153): Count lines: 32
05-09 19:39:26.132: I/Claw(27153): ColorSize: 3 data: 23970816 width: 3264 height: 2448
05-09 19:39:26.155: I/Claw(27153): Sending..
05-09 19:39:26.157: I/Claw(27153): Accept server socket.
05-09 19:39:26.157: I/Claw(27153): Test line count: 32
05-09 19:39:26.158: I/Claw(27153): Data to write: 313497
05-09 19:39:26.190: I/Claw(27153): Pointer: 313497
05-09 19:39:26.190: I/Claw(27153): Done processing, id: 91844
05-09 19:39:26.397: I/Claw(27153): Sent data from server socket.
05-09 19:39:26.655: I/Claw(27153): Picture data received, id: 45526
05-09 19:39:26.655: I/Claw(27153): Process data, id: 45526
05-09 19:39:26.656: I/Claw(27153): DataManager started, id: 45526
05-09 19:39:26.661: I/Claw(27153): Do take picture, id: 92593
05-09 19:39:26.910: I/Claw(27153): Actual take picture done, id: 92593
05-09 19:39:27.174: I/Claw(27153): Count lines: 32
05-09 19:39:27.180: I/Claw(27153): Pointer: 313497
05-09 19:39:27.180: I/Claw(27153): Done processing, id: 1194
05-09 19:39:27.182: I/Claw(27153): Sending..
05-09 19:39:27.183: I/Claw(27153): Accept server socket.
05-09 19:39:27.183: I/Claw(27153): Test line count: 32
05-09 19:39:27.183: I/Claw(27153): Data to write: 313497
05-09 19:39:27.206: I/Claw(27153): ColorSize: 3 data: 23970816 width: 3264 height: 2448

05-09 19:39:27.639: I/Claw(27153): Picture data received, id: 92593
05-09 19:39:27.639: I/Claw(27153): Process data, id: 92593
05-09 19:39:27.642: I/Claw(27153): DataManager started, id: 92593
05-09 19:39:27.643: I/Claw(27153): Do take picture, id: 44480
05-09 19:39:27.686: I/Claw(27153): Sent data from server socket.
05-09 19:39:27.870: I/Claw(27153): Actual take picture done, id: 44480
05-09 19:39:28.215: I/Claw(27153): Count lines: 32
05-09 19:39:28.224: I/Claw(27153): ColorSize: 3 data: 23970816 width: 3264 height: 2448
05-09 19:39:28.241: I/Claw(27153): Pointer: 313497
05-09 19:39:28.241: I/Claw(27153): Done processing, id: 62983
05-09 19:39:28.242: I/Claw(27153): Sending..
05-09 19:39:28.242: I/Claw(27153): Accept server socket.
05-09 19:39:28.242: I/Claw(27153): Test line count: 32
05-09 19:39:28.242: I/Claw(27153): Data to write: 313497
05-09 19:39:28.429: I/Claw(27153): Sent data from server socket.
05-09 19:39:28.606: I/Claw(27153): Picture data received, id: 44480
05-09 19:39:28.606: I/Claw(27153): Process data, id: 44480
05-09 19:39:28.607: I/Claw(27153): DataManager started, id: 44480
05-09 19:39:28.607: I/Claw(27153): Do take picture, id: 55472
05-09 19:39:28.868: I/Claw(27153): Actual take picture done, id: 55472
05-09 19:39:29.280: I/Claw(27153): Count lines: 32
05-09 19:39:29.282: I/Claw(27153): ColorSize: 3 data: 23970816 width: 3264 height: 2448
05-09 19:39:29.287: I/Claw(27153): Pointer: 313497
05-09 19:39:29.287: I/Claw(27153): Done processing, id: 45526
05-09 19:39:29.288: I/Claw(27153): Sending..
05-09 19:39:29.288: I/Claw(27153): Accept server socket.
05-09 19:39:29.290: I/Claw(27153): Test line count: 32
05-09 19:39:29.290: I/Claw(27153): Data to write: 313497
05-09 19:39:29.631: I/Claw(27153): Sent data from server socket.
05-09 19:39:29.768: I/Claw(27153): Picture data received, id: 55472
05-09 19:39:29.768: I/Claw(27153): Process data, id: 55472
05-09 19:39:29.773: I/Claw(27153): DataManager started, id: 55472
05-09 19:39:29.781: I/Claw(27153): Do take picture, id: 95894
05-09 19:39:30.023: I/Claw(27153): Actual take picture done, id: 95894
05-09 19:39:30.575: I/Claw(27153): Count lines: 32
05-09 19:39:30.588: I/Claw(27153): Pointer: 313497
05-09 19:39:30.591: I/Claw(27153): Sending..
05-09 19:39:30.592: I/Claw(27153): Accept server socket.
05-09 19:39:30.594: I/Claw(27153): Test line count: 32
05-09 19:39:30.595: I/Claw(27153): Data to write: 313497
05-09 19:39:30.608: I/Claw(27153): Done processing, id: 92593
05-09 19:39:30.782: I/Claw(27153): Picture data received, id: 95894
05-09 19:39:30.782: I/Claw(27153): Process data, id: 95894
05-09 19:39:30.783: I/Claw(27153): DataManager started, id: 95894
05-09 19:39:30.793: I/Claw(27153): Do take picture, id: 41652
05-09 19:39:30.982: I/Claw(27153): Sent data from server socket.
05-09 19:39:31.053: I/Claw(27153): Actual take picture done, id: 41652
05-09 19:39:31.425: I/Claw(27153): Count lines: 32
05-09 19:39:31.432: I/Claw(27153): Pointer: 313497

05-09 19:39:31.434: I/Claw(27153): Sending..
05-09 19:39:31.438: I/Claw(27153): Done processing, id: 44480
05-09 19:39:31.444: I/Claw(27153): Accept server socket.
05-09 19:39:31.445: I/Claw(27153): Test line count: 32
05-09 19:39:31.445: I/Claw(27153): Data to write: 313497

13.6 Code voor Java-client

Onderstaand staat de klasse met code die verantwoordelijk is voor de client-functionaliteit van de software voor de contactloze kopdiktemeter. De client draait op de externe computer en verbindt met de Android telefoon.

De client-functionaliteit bestaat voornamelijk uit netwerkcommunicatiecode, aangezien de beeldanalyse door de opdrachtgever wordt aangeleverd.

```
public class Client {
    public static void startClientLines(final String ipAddress) {
        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                while(true) {
                    SocketHints hints = new SocketHints();
                    Socket socket;
                    try {
                        socket = Gdx.net.newClientSocket(Net.Protocol.TCP, ipAddress,
Config.COMMUNICATION_PORT, hints);

                        try {
                            int width = StreamUtil.getInt(socket.getInputStream()); //
mode
                            int height = StreamUtil.getInt(socket.getInputStream()); //
mode

                            LogHandler.log("Camera width: " + width + " and camera
height: " + height);

                            socket.getOutputStream().write(1); // mode
                            for (int i = 0; i < Config.MAX_LINES; i++) {
                                int sendId = MathUtils.random(0, 3000);
                                LogHandler.log("SendId: " + sendId);
StreamUtil.putInt(socket.getOutputStream(), sendId);
                            }
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                        LogHandler.log("We've sent the mode and line id's.");
                        break;

                    } catch (RuntimeException ex) {
                        LogHandler.log("Waiting..");
                        try {
                            Thread.sleep(1000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                        continue;
                    }
                }

                receiveData(ipAddress);
            }
        });
        thread.start();
    }

    public static void startClientEntireJPEG(final String ipAddress) {
        Thread thread = new Thread(new Runnable() {
```

```
            @Override
            public void run() {
                while (true) {
                    SocketHints hints = new SocketHints();
                    Socket socket;
                    try {
                        socket = Gdx.net.newClientSocket(Net.Protocol.TCP, ipAddress,
Config.COMMUNICATION_PORT, hints);

                        try {
                            int width = StreamUtil.getInt(socket.getInputStream()); //
mode
                            int height = StreamUtil.getInt(socket.getInputStream()); //
mode

                            LogHandler.log("Camera width: " + width + " and camera
height: " + height);

                            socket.getOutputStream().write(2); // mode
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                        LogHandler.log("We've sent the mode.");
                        break;

                    } catch (RuntimeException ex) {
                        LogHandler.log("Waiting..");
                        try {
                            Thread.sleep(1000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                        continue;
                    }
                }

                receiveData(ipAddress);
            }
        });
        thread.start();
    }

    // Receive data independent from set mode!
    private static void receiveData(final String ipAddress) {

        while(true) {
            Socket socket;
            SocketHints hints = new SocketHints();
            try {
                socket = Gdx.net.newClientSocket(Net.Protocol.TCP, ipAddress,
Config.PAYLOAD_PORT, hints);

            } catch (RuntimeException ex) {
                LogHandler.log("Waiting for socket..");
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                continue;
            }
```

```
LogHandler.log("------ Waiting for data.. ---------");

int dataSize = -1;
try {
    dataSize = StreamUtil.getInt(socket.getInputStream());
} catch (IOException e) {
    e.printStackTrace();
}
LogHandler.log("dataSize: " + dataSize);

int mode = -1;

try {
    mode = socket.getInputStream().read();
} catch (IOException e) {
    e.printStackTrace();
}
dataSize--;

LogHandler.log("Mode: " + mode + (mode == 1 ? "(lines)" : "(entire JPEG)"));

// Receive line id if mode is 1

int lineCount = -1;
if (mode == 1) {
    try {
        lineCount = StreamUtil.getInt(socket.getInputStream());
    } catch (IOException e) {
        e.printStackTrace();
    }
    LogHandler.log("lineCount: " + lineCount);
    dataSize -= 4;
}

// Receive payload size
int size = -1;
try {
    size = StreamUtil.getInt(socket.getInputStream());
} catch (IOException e) {
    e.printStackTrace();
}
LogHandler.log("Size: " + size);
dataSize -= 4;

// Receive time between previous frame
int diffTime = -1;
try {
    diffTime = StreamUtil.getInt(socket.getInputStream());
} catch (IOException e) {
    e.printStackTrace();
}
LogHandler.log("DiffTime (ms): " + diffTime);
dataSize -= 4;

int accelX = -1;
try {
    accelX = StreamUtil.getInt(socket.getInputStream());
} catch (IOException e) {
    e.printStackTrace();
}
```

```
LogHandler.log("Accelerometer X: " + accelX);
dataSize -= 4;

int accelY = -1;
try {
    accelY = StreamUtil.getInt(socket.getInputStream());
} catch (IOException e) {
    e.printStackTrace();
}
LogHandler.log("Accelerometer Y: " + accelY);
dataSize -= 4;

int accelZ = -1;
try {
    accelZ = StreamUtil.getInt(socket.getInputStream());
} catch (IOException e) {
    e.printStackTrace();
}
LogHandler.log("Accelerometer Z: " + accelZ);
dataSize -= 4;

if (size < 0 || mode < 0) {
    LogHandler.exit("Mode or size was faulty!");
    break;
}

if (mode == 1) {
    for (int k = 0; k < lineCount; k++) {
        int lineId = -1;
        try {
            lineId = StreamUtil.getInt(socket.getInputStream());
        } catch (IOException e) {
            e.printStackTrace();
        }
        LogHandler.log("lineId: " + lineId);
        dataSize -= 4;

        byte data[] = new byte[size];
        for (int i = 0; i < size; i++) {
            int value = -1;
            try {
                value = socket.getInputStream().read();
            } catch (IOException e) {
                e.printStackTrace();
            }

            if (value < 0) {
                LogHandler.log("End of transmission.");
                break;
            }

            data[i] = (byte) value;
        }
        dataSize -= size;

        // Do something with data...
    }
} else if (mode == 2) {
    if (size == 0) {
        LogHandler.log("Empty JPEG!");
    } else {
```

```
                    byte data[] = new byte[size];
                    for (int i = 0; i < size; i++) {
                        int value = -1;
                        try {
                            value = socket.getInputStream().read();
                        } catch (IOException e) {
                            e.printStackTrace();
                        }

                        if (value < 0) {
                            LogHandler.log("End of transmission.");
                            break;
                        }

                        data[i] = (byte) value;
                    }
                    dataSize -= size;

                    JPEGDecoder.get().decode(data);
                }
            } else {
                LogHandler.log("Mode not 1 or 2!");
            }
            LogHandler.log("Data left: " + dataSize);
            try {
                while(dataSize > 0) {
                    socket.getInputStream().read();
                    dataSize--;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }


            LogHandler.log("------------ Finished data.. ---------");
        }

        LogHandler.log("Ended reading.");
    }
}
```

13.7 Code voor Lokale berekening

Onderstaand staat de klasse met code verantwoordelijk voor de lokale berekening die uitgevoerd wordt op de Android telefoon. Momenteel betreft dit de *loadtest*.

In de code staat commentaar waarbij wordt aangegeven wat er bij bepaalde stappen gebeurt.

```
package net.kajos.claw.Engine;


import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.math.Vector2;
import net.kajos.claw.Config;
import net.kajos.claw.Graphics.GUI;
import net.kajos.claw.Utils.JPEGDecoder;
import net.kajos.claw.Utils.Time;

/**
 * Created by kajos on 24-2-15.
 */
public class LocalProcessor {
    byte[] pixelData;
    int width = 0;
    int height = 0;

    public LocalProcessor() {
    }

    public void init(int width, int height) {
        if (this.width != width || this.height != height) {
            this.width = width;
            this.height = height;

            // There are 3 lines to be analyzed
            // The middle of the screen
            // From the middle, Config.OFFSET_CENTER_Y lines up vertically
            // From the middle, Config.OFFSET_CENTER_Y lines down vertically
            ids[0] = this.height / 2 - Config.OFFSET_CENTER_Y;
            ids[1] = this.height / 2;
            ids[2] = this.height / 2 + Config.OFFSET_CENTER_Y;

            bwLines = new boolean[ids.length][this.width];

            tmp = new Vector2[ids.length];
            for (int i = 0; i < ids.length; i++) {
                tmp[i] = new Vector2();
            }
        }
    }

    // This array contains the line IDs
    // At the moment they are set in init, which is run at application start
    private int[] ids = new int[]{0,0,0};
    private boolean[][] bwLines;

    Vector2[] tmp;
    public void run(final byte[] data) {
        pixelData = JPEGDecoder.get().decode(data, ids);

        int nWidth = JPEGDecoder.get().getWidth();
        int nHeight = JPEGDecoder.get().getHeight();
```

```
init(nWidth, nHeight);

float line_m = -.1f/.48f;
float line_b = 0.425f;

int c = 0;
for (int i = 0; i < ids.length; i++) {
    for (int k = 0; k < width; k++, c+=4) {
        // Here the image analysis happens.
        // You get the rgb values with pixelData[c,c+1,c+2].
        // You can get the line ID with ids[i].
        float summed = 0f;

        float r = pixelData[c];
        summed += r;

        float g = pixelData[c+1];
        summed += g;

        float b = pixelData[c+2];
        summed += b;

        r /= summed;
        g /= summed;
        b /= summed;

        // The result is written to the array bwLines
        if (g > (line_m*r + line_b))
            bwLines[i][k] = true;
        else
            bwLines[i][k] = false;
    }
}

// The middle of the image is taken
// There is a boundary defined by Config.OFFSET_CENTER_X
// Outside this boundary no analysis happens
int start = width / 2 - Config.OFFSET_CENTER_X;
int end = width / 2 + Config.OFFSET_CENTER_X;

// The first encounters from the left are stored here
int[] first = new int[ids.length];
// The last encounters from the left are stored here
int[] last = new int[ids.length];
// The middle is calculated from first and last occurrences and store here
int[] medium = new int[ids.length];
// The size is calculated from first and last occurrences and store here
int[] size = new int[ids.length];

// First occurrence is found
for (int i = 0; i < ids.length; i++) {
    loop:
    for (int k = start; k < end; k++) {
        if (bwLines[i][k]) {
            first[i] = k;
            break loop;
        }
    }
}
// Last occurrence is found
for (int i = 0; i < ids.length; i++) {
```

```
            loop:
            for (int k = end-1; k >= start; k--) {
                if (bwLines[i][k]) {
                    last[i] = k;
                    break loop;
                }
            }
        }

        // Middle and size are calculated
        for (int i = 0; i < ids.length; i++) {
            medium[i] = (first[i] + last[i]) / 2;
            size[i] = last[i] - first[i];
        }

        if (size[0] == 0 || size[1] == 0 || size[2] == 0) {
            Gdx.app.postRunnable(new Runnable() {
                @Override
                public void run() {
                    GUI.get().setLabel(9, "No match found", 100, 800);
                }
            });
            return;

        }

        // Average medium from all lines is calculated here
        int avgMedium = 0;
        for (int i = 0; i < ids.length; i++) {
            medium[i] = (first[i] + last[i]) / 2;
            avgMedium += medium[i];
            tmp[i].set(medium[i], ids[i]);
        }
        avgMedium /= ids.length;


        float angle = tmp[0].sub(tmp[ids.length-1]).angle() - 270f; // angle between
first and last

        String string = "Angle: " + angle + "\nWidth in pixels: ";
        for(int i = 0;i<ids.length;i++)
            string+=size[i]+",";

        string+="\nAvg: "+avgMedium;

        final String gString = string;
        Gdx.app.postRunnable(new Runnable() {
            @Override
            public void run() {
                GUI.get().setLabel(9, gString, 200,800);
            }
        });

    }
}
```

13.8 Overzicht mogelijke technieken

In dit hoofdstuk wordt een overzicht gemaakt van de onderzochte technieken zijn voor gebruik in de software voor de contactloze kopdiktemeter.

13.8.1 Camera werking

Inleiding

De camera van de Android telefoon kan op diverse manieren benut worden. De meest geschikte methode hangt af van het doel en de beschikbare middelen. Het gaat om de acquisitie van de beelden, het formaat van de oplevering van de beelden en extra eigenschappen zoals het gebruik van een flitser.

Vastleggen foto van enkele lijnen

Van de 326x2448 pixels zijn er maar enkele lijnen van de afbeelding nodig om te gebruiken voor de beeldanalyse. Als een foto gemaakt kan worden bestaande uit een enkele lijn, zou dit sneller gaan. Dit belast de camera minder en heeft daarom de voorkeur. De pixels worden dan lijn voor lijn uitgelezen in plaats van de complete foto met het maximaal aantal beschikbare lijnen.

Echter is het maken van een foto, bestaande uit alleen een enkele lijn van pixels, niet mogelijk op de Wolfgang, als mede op de meeste andere Android telefoons, aangezien deze methode niet ondersteund wordt in de mogelijke foto-resoluties.

Videostream

De camera op Android telefoons ondersteunt videostreams. Videostreams werken meestal op een resolutie van ongeveer 1280x720 met 30 beelden per seconden.

Fotostream

De maximale resolutie van een foto op de Wolfgang telefoon is 3264x2448. De foto kan ongeveer 1 keer per seconde gemaakt worden.

Combinatie video- en fotostream

Een van de methodes die ik heb toegepast is om tussen het maken van de foto's in een videostream te maken. Dat gaat als volgt: de videostream start, streamt live video naar het beeldscherm, vervolgens wordt de stream gestopt en wordt een foto gemaakt, waarna de videostream weer gestart wordt.

Echter in de praktijk blijkt dat de CMOS-sensor niet genoeg tijd heeft voor een videostream. Het beeld hapert en wordt donkerder van aanzicht. Nog belangrijker is dat de fotostream er langer over doet om een foto te maken.

Daarom is er gekozen om de preview weer te geven d.m.v. de fotostream, wat betekent dat er minder beelden per seconde getoond worden, maar de fotostream niet langzamer verloopt.

Tevens wordt een 3D representatie getoond.

Formaat oplevering

De meeste Android telefoons maken foto's en comprimeren deze hardware- of softwarematig naar het JPEG-formaat.

Pas daarna is de foto beschikbaar en dus enkel in het JPEG-formaat.

Als alleen enkele lijnen uitgelezen dienen te worden, is het JPEG-formaat niet bruikbaar.

Daarom moet eerst de JPEG afbeelding weer gedecomprimeerd worden, om vervolgens de lijnen uit te lezen, waarna deze verzonden kunnen worden over het netwerk. Dit kost extra tijd en processorkracht.

Een van de weinige telefoons waar dit niet nodig is, is de Nexus 5. Deze ondersteunt ten tijde van schrijven het RAW-formaat bij het maken van foto's. Het RAW formaat is een ongecomprimeerd formaat.

Flitser

In sommige gevallen levert de camera donkere beelden op. In dat geval is het handig om een flitser actief te hebben. Voor onderzoeksdoeleinden is nodig om de flitser aan en uit te kunnen zetten.

13.8.2 Visualisatie

Inleiding

Om de Android applicatie makkelijk in gebruik te maken, is het nodig om de gebruiker feedback te geven over de staat van het proces. Een belangrijk onderdeel is om weer te geven wat de camera ziet. Dat kan op diverse manieren. Zo kan met OpenGL een omgeving getoond worden of een directe feed van de camera getoond worden.

OpenGL weergave

Om te laten zien hoe de plantensteel georiënteerd en gepositioneerd staat, wordt er gebruik gemaakt van OpenGL visualisatie. OpenGL wordt ondersteund door het Android platform in de vorm van OpenGL ES, een mobiele variant van OpenGL. In mijn geval maak ik gebruik van OpenGL ES 2.0; een ruim ondersteunde versie met veel mogelijkheden (voor b.v. optimalisatie).

Op de desktop wordt OpenGL ES in theorie geëmuleerd. OpenGL ES is namelijk voornamelijk een subset van de gewone OpenGL, welke aanwezig op desktop platformen.

Directe video weergave van de camera

Een mogelijke alternatieve methode om te laten zien waar de steel zich bevindt, is om simpelweg een live videostream van de camera te tonen op het scherm.

Aangezien er gewerkt wordt met spiegels, betekent dit wel dat het ingewikkelder is voor de gebruiker om het beeld te interpreteren.

Deze methode is uiteindelijk niet geïmplementeerd, omdat het niet mogelijk is gebleken om een live video stream naast de hoge resolutie fotostream open te hebben.

Directe foto weergave van de camera

Als de foto's gemaakt zijn, moeten deze eerst gedecomprimeerd worden om de individuele lijnen uit te lezen. Daarna kunnen de lijnen verzonden worden over het netwerk.

Na de decompressie, is het mogelijk om een klein geschaalde versie van de foto, te versturen naar de GPU van de Android telefoon, om deze aan de gebruiker te laten zien. Dit is een zelfde idee als bovenstaande, waarbij in plaats van video, foto's gebruikt worden.

Echter wordt deze afbeelding maar een enkele keer per seconde of langer ververst. Het is echter wel handig om een indicatie te hebben wat de computer ontvangt.

13.8.3 Berekening

Inleiding

Na de acquisitie van het beeld, dient het beeld gebruikt te worden als basis voor berekeningen. De berekeningen zorgen voor de oplevering van de positie, rotatie en dikte van de steel. Deze berekeningen kunnen lokaal en extern gebeuren. Tevens kunnen deze berekeningen op verschillende manieren geschieden.

Lokale berekening

Aangezien er een hoge vereiste is aan de resolutie, is de shuttertijd van de camera behoorlijk groot. Terwijl de camera bezig is om de foto te maken, is er veel ongebruikte tijd over. Deze ongebruikte tijd kan makkelijk ingevuld worden, door het beeldanalyse algoritme te verplaatsen naar de telefoon, in plaats van deze te draaien op een externe computer.

Hiermee kan dan de koppeling met de computer verwijderd worden en is enkel de telefoon nog vereist.

Het is hier voor wel nodig om het beeldanalyse algoritme in Java te schrijven.

Wellicht een bijkomend voordeel is dat de vertraging minder optreedt, aangezien er geen gegevens worden verstuurd over het netwerk.

Er kan wel verwacht worden dat het batterijverbruik toeneemt door de hogere consumptie van de CPU.

De opdrachtgever had de wens om de zware berekeningen en de Android telefon gescheiden te houden. Ik zal hier echter wel rekening mee houden in mijn code, zodat dit in de toekomst toch toegepast kan worden zonder veel moeite.

Externe berekening met netwerkcommunicatie

Een andere mogelijkheid is om de zware berekeningen uit te voeren op een externe computer. Hiervoor moet de data over het netwerk verzonden worden. In het geval dat enkele lijnen worden opgevraagd is dat niet een groot probleem, al is er wel meer vertraging aanwezig door het gebruiken van een (draadloze) netwerkverbinding.

Het gebruik van een draadloze verbinding kan tot meer energieverbruik leiden.

Nadat de berekening gemaakt is, kunnen de resultaten teruggestuurd worden naar de Android telefoon voor weergave.

Er moet rekening gehouden worden met meer vertraging door het gebruik van netwerkcommunicatie.

Accelerometer data

Een vraag van de opdrachtgever was om de accelerometer data uit te lezen vanaf de Android telefoon. Met deze data, kan de snelheid van de telefoon in 3 axis uitgelezen worden. Door de gemiddelde snelheid (m/s) te vermenigvuldigen met de tijdsduur (s), kan de verplaatsing (m) berekend worden.

D.m.v. de verplaatsing van de telefoon tussen de gemaakte foto's in, kan theoretisch de dikte van de steel berekend worden, wellicht zonder dat gebruik gemaakt hoeft te worden van de module die gekoppeld wordt aan de Android telefoon.

Het is wel de vraag, of de accelerometer data van de beschikbare Android telefoons nauwkeurig genoeg is voor een dergelijk proces als hierboven beschreven.

13.9 Overzicht toegepaste technieken

In dit hoofdstuk wordt een overzicht gemaakt van de technieken die uiteindelijk zijn gebruikt in de software voor de contactloze kopdiktemeter.

13.9.1 Fotostream

Voor het creëren van beeldmateriaal, is gekozen om een constante serie van foto's te maken, oftewel een fotostream. Het opnemen van video met hedendaagse telefoons levert beelden op met een lagere resolutie dan nodig voor de beeldanalyse.

Het lijkt onoverkomelijk dat het resultaat van een fotostream ook vertraging met zich mee brengt. De vertraging kan al snel lopen in de vijf seconden.

13.9.2 Threading

Om de applicatie zo snel mogelijk te laten verlopen, wordt er hevig gebruik gemaakt van threads. De threads zorgen er voor dat sommige processen parallel kunnen worden uitgevoerd. Als processen parallel worden uitgevoerd, staat de processor minder lang stil, te wachten op een resultaat. Tevens kan er beter worden gebruik gemaakt van de meerdere processorkernen tegenwoordig aanwezig in processoren.

Er zijn een aantal hoofdprocessen aanwezig. Allereerst moet de foto gemaakt worden, vervolgens moet de foto gedecomprimeerd worden, waarna de foto getoond moet worden en moet worden verzonden.

Er moet dus een thread zijn om de foto te maken en een thread om de foto te decomprimeren. De verkleining van de foto gebeurt op de zelfde thread als de decompressie. Het tonen van de foto gebeurt echter weer op de separate OpenGL thread. Het verzenden van de foto gebeurt uiteindelijk ook op een aparte netwerkthread.

Daarnaast is er nog een tweede netwerkthread voor de data aanvragen.

13.9.3 Asynchroon netwerkverkeer

Het merendeel van de applicatie, draait parallel in verschillende threads. Taken worden ook niet altijd sequentieel uitgevoerd. De fotodata wordt hierdoor opgeleverd in batches.

Om deze reden zijn er twee netwerkpoorten in gebruik: een voor communicatie van informatie zoals opdrachten en de ander voor data van de foto's.

Als een opdracht wordt gestuurd, is de kans groot dat deze pas een halve seconde later wordt uitgevoerd en dat de data niet direct aan de opdracht voldoet.

13.9.4 Weergave beeld

Aangezien een videostream niet te openen is naast de fotostream, is er gekozen om de foto na decompressie te schalen naar een kleiner formaat met lagere resolutie en deze weer te geven.

De gebruiker krijgt hierdoor een idee wat de camera ziet, zonder dat het ten koste gaat van de kwaliteit en snelheid van het proces.

Daarnaast wordt ook d.m.v. een OpenGL omgeving een schatting gegeven van de positie van de steel. Deze steel over het camerabeeld heen gelegd of eventueel wordt een optie gegeven tot het schakelen tussen beelden.

13.9.5 Berekening

De opdracht vereist dat de zware, “image vision” berekeningen op een externe computer uitgevoerd kunnen worden. Dit is dan ook mogelijk. Ik heb hiervoor tevens een client-applicatie opgeleverd welke is geschreven in de C programmeertaal d.mv. de Dev-C++ IDE. Communicatie over het netwerk is zoveel mogelijk geoptimaliseerd om snel te verlopen.

Daarnaast, als er geen netwerkcommunicatie plaatsvindt, draait de Android telefoon een loadtest om lokaal berekeningen uit te voeren. Dit proces leidt nog niet tot daadwerkelijke resultaten, maar biedt houvast voor toekomstige ontwikkelingen. Zie hiervoor het hoofdstuk Aanbevelingen op pagina 57.

13.9.6 Communicatie over netwerk

Er wordt diverse informatie over het netwerk verstuurd. Zo wordt eerst de grootte van de foto's verstuurd; breedte en hoogte. Daarna stuurt de cliënt de gewenste opdrachtmodus (JPEG of individuele lijnen), en indien nodig de gewenste lijnen.

De data die ontvangen wordt door de cliënt, bevat de modus, individuele lijnen of JPEG data en tevens accelerometer data en de verstreken tijd sinds de vorige data overdracht.

De accelerometerdata en verstreken tijd kan gebruikt worden voor berekeningen.

13.10 Testraamwerk

13.10.1 Vertraging

De tijd is gemeten van het maken van de foto tot na de verwerking van de foto. Het gaat hier om de tijd van het maken van de foto tot het verzenden van de foto. De tijd die in beslag wordt genomen door het netwerk wordt niet meegeteld.

Loadtest lokaal

03-24 12:09:04.591: I/Claw(24220): Do take picture, id: 77570
03-24 12:09:05.069: I/Claw(24220): Actual take picture done, id: 77570
03-24 12:09:05.997: I/Claw(24220): Picture data received, id: 77570
03-24 12:09:05.997: I/Claw(24220): Process data with ID: 77570
03-24 12:09:05.998: I/Claw(24220): DataManager started, id: 77570
03-24 12:09:07.885: I/Claw(24220): Done processing, id: 77570

Tijd van begin tot eind:

12:09:07.885 – 12:09:04.591 = 3.294 seconden

Lijnen versturen

03-24 12:09:39.054: I/Claw(24220): Do take picture, id: 13012
03-24 12:09:39.463: I/Claw(24220): Actual take picture done, id: 13012
03-24 12:09:40.371: I/Claw(24220): Picture data received, id: 13012
03-24 12:09:40.371: I/Claw(24220): Process data, id: 13012
03-24 12:09:40.372: I/Claw(24220): DataManager started, id: 13012
03-24 12:09:42.419: I/Claw(24220): Done processing, id: 13012

Tijd van begin tot eind:

12:09:42.419 – 12:09:39.054 = 3.365 seconden

JPEG versturen

03-24 12:10:21.348: I/Claw(24220): Do take picture, id: 17372
03-24 12:10:21.769: I/Claw(24220): Actual take picture done, id: 17372
03-24 12:10:22.653: I/Claw(24220): Picture data received, id: 17372
03-24 12:10:22.653: I/Claw(24220): Process data, id: 17372
03-24 12:10:22.654: I/Claw(24220): DataManager started, id: 17372
03-24 12:10:22.669: I/Claw(24220): Done processing, id: 17372

Tijd van begin tot eind:

12:10:22.669 – 12:10:21.348 = 1.321 seconden

JPEG versturen over USB

05-09 19:18:48.552: I/Claw(24918): Do take picture, id: 98123
05-09 19:18:48.563: I/Claw(24918): Done processing, id: 23823
05-09 19:18:48.772: I/Claw(24918): Actual take picture done, id: 98123
05-09 19:18:49.468: I/Claw(24918): Picture data received, id: 98123
05-09 19:18:49.468: I/Claw(24918): Process data, id: 98123
05-09 19:18:49.469: I/Claw(24918): DataManager started, id: 98123
05-09 19:18:49.478: I/Claw(24918): Done processing, id: 98123

Tijd van begin tot eind:

19:18:49.478 – 19:18:48.552 = 0.926 seconden

Lijnen versturen over USB

05-09 19:39:26.661: I/Claw(27153): Do take picture, id: 92593
05-09 19:39:26.910: I/Claw(27153): Actual take picture done, id: 92593
05-09 19:39:27.174: I/Claw(27153): Count lines: 32
05-09 19:39:27.180: I/Claw(27153): Pointer: 313497
05-09 19:39:27.180: I/Claw(27153): Done processing, id: 1194
05-09 19:39:27.182: I/Claw(27153): Sending..
05-09 19:39:27.183: I/Claw(27153): Accept server socket.
05-09 19:39:27.183: I/Claw(27153): Test line count: 32
05-09 19:39:27.183: I/Claw(27153): Data to write: 313497
05-09 19:39:27.206: I/Claw(27153): ColorSize: 3 data: 23970816 width: 3264 height: 2448
05-09 19:39:27.639: I/Claw(27153): Picture data received, id: 92593
05-09 19:39:27.639: I/Claw(27153): Process data, id: 92593
05-09 19:39:27.642: I/Claw(27153): DataManager started, id: 92593
05-09 19:39:30.608: I/Claw(27153): Done processing, id: 92593

Tijd van begin tot eind:

19:39:30.608 – 19:39:26.661 = 3.947 seconden

13.10.2 Tijd tussen foto's

Bij deze test wordt de tijd gemeten van het maken van een foto tot het maken van de volgende foto. De foto's worden op een aparte thread gemaakt. Doordat de foto's op een aparte thread gemaakt worden, wordt er voor gezorgd dat er geen extra vertraging optreedt bij de andere processen. De snelheid van het maken van de foto's heeft geen invloed op de snelheid van het verzenden en ontvangen van de resultaten.

Loadtest lokaal

11:05:48.220 – 11:05:46.916 = 1.304 seconden

Lijnen versturen

11:08:53.505 – 11:08:52.066 = 1.439 seconden

JPEG versturen

11:07:27.108 – 11:07:25.790 = 1.318 seconden

JPEG versturen USB

19:18:49.594 – 19:18:48.552 = 1.042 seconden

Lijnen versturen USB

19:39:27.643 – 19:39:26.661 = 0.982 seconden

13.10.3 Tijd tussen resultaten of verzendingen

Bij deze test wordt de tijd gemeten tussen de ontvangen of berekende resultaten. De tijden worden gemeten op in client-applicatie, behalve bij de lokale loadtest.

Loadtest lokaal

1.310 seconden

Lijnen versturen

1.436 seconden

JPEG versturen

3.356 seconden

JPEG versturen USB

2.628 seconden

Lijnen versturen USB

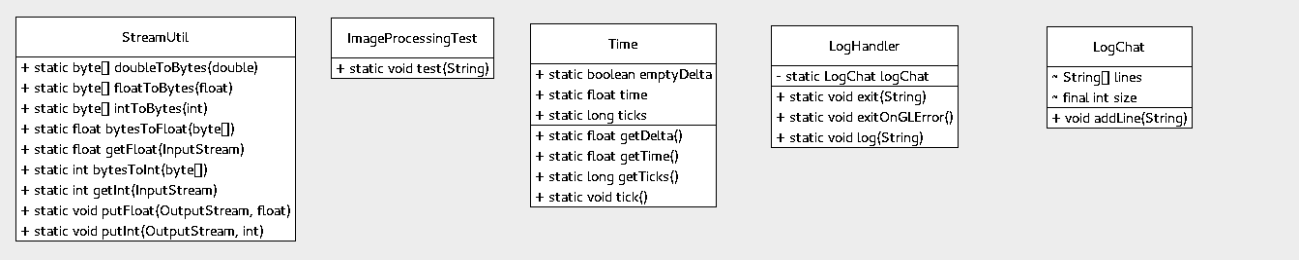
0.982 seconden

13.11 Overige UML klassen

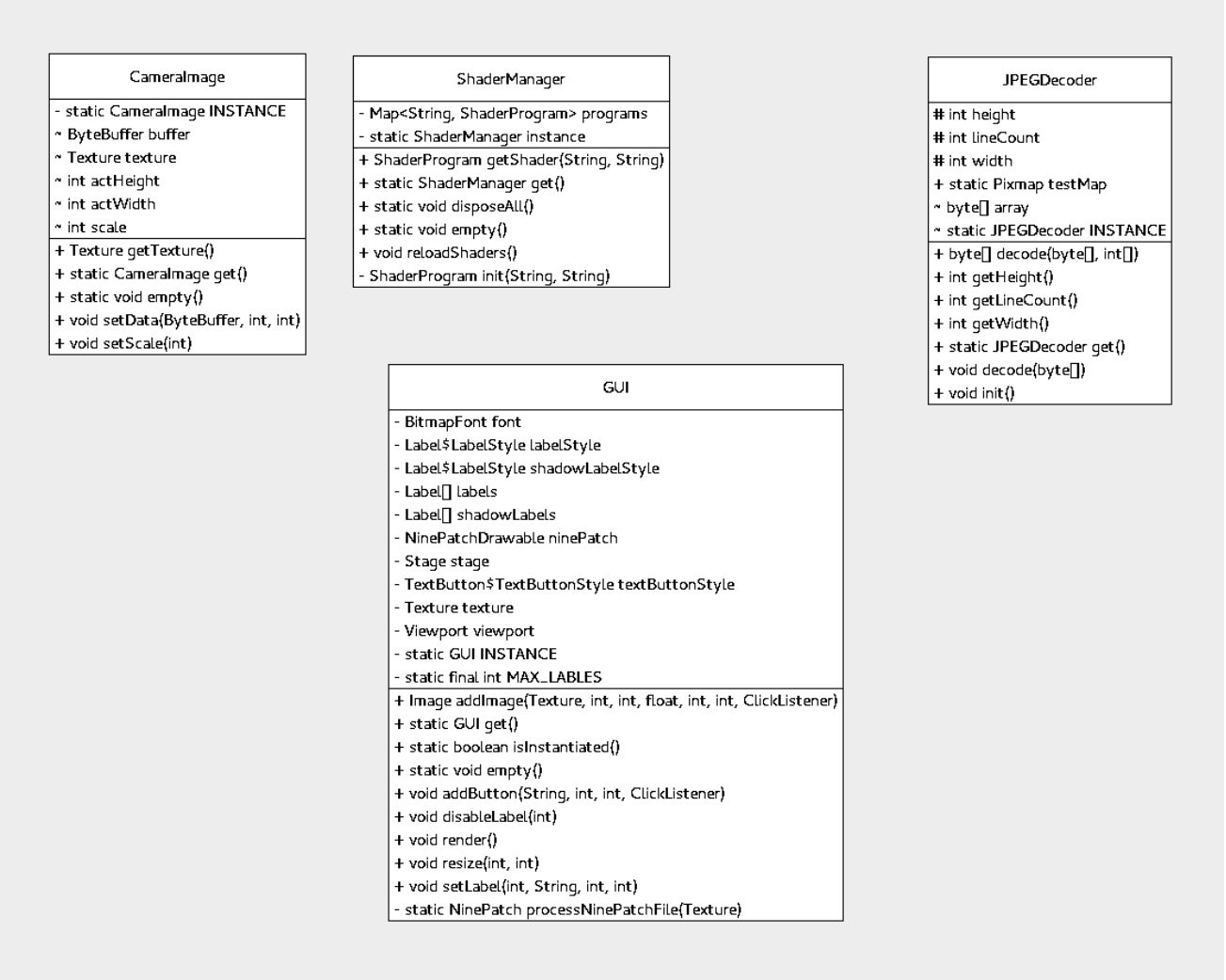
13.11.1 Klassendiagrammen

In onderstaande figuren staan de overzichten van de statische en singleton klassen.

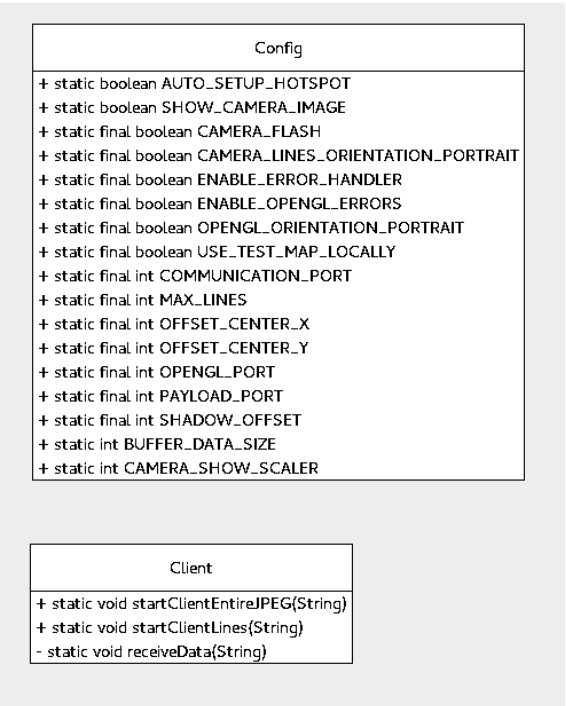
- De Config-klasse bevat alle configuratie-variabelen.
- De Client-klasse bevat de netwerkcommunicatie-code voor de client-applicatie.
- De CameraImage-klasse bevat de code om de afbeelding van de camera te tonen op het scherm. De ShaderManager-klasse behandelt de shaders voor OpenGL.
- De StreamUtil-klasse heeft enkele helper-functies.
- De ImageProcessingTest-klasse heeft een test voor de beeldanalyse. De Time-klasse houdt o.a. de tijd bij.
- De LogHandler en LogChat klassen zorgen voor het bijhouden en tonen van de log.



FIGUUR 23 OVERZICHTEN VAN VOORNAMELIJK STATISCHE OF SINGLETON KLASSEN.



FIGUUR 24 OVERZICHTEN VAN VOORNAMELIJK STATISCHE OF SINGLETON KLASSEN.

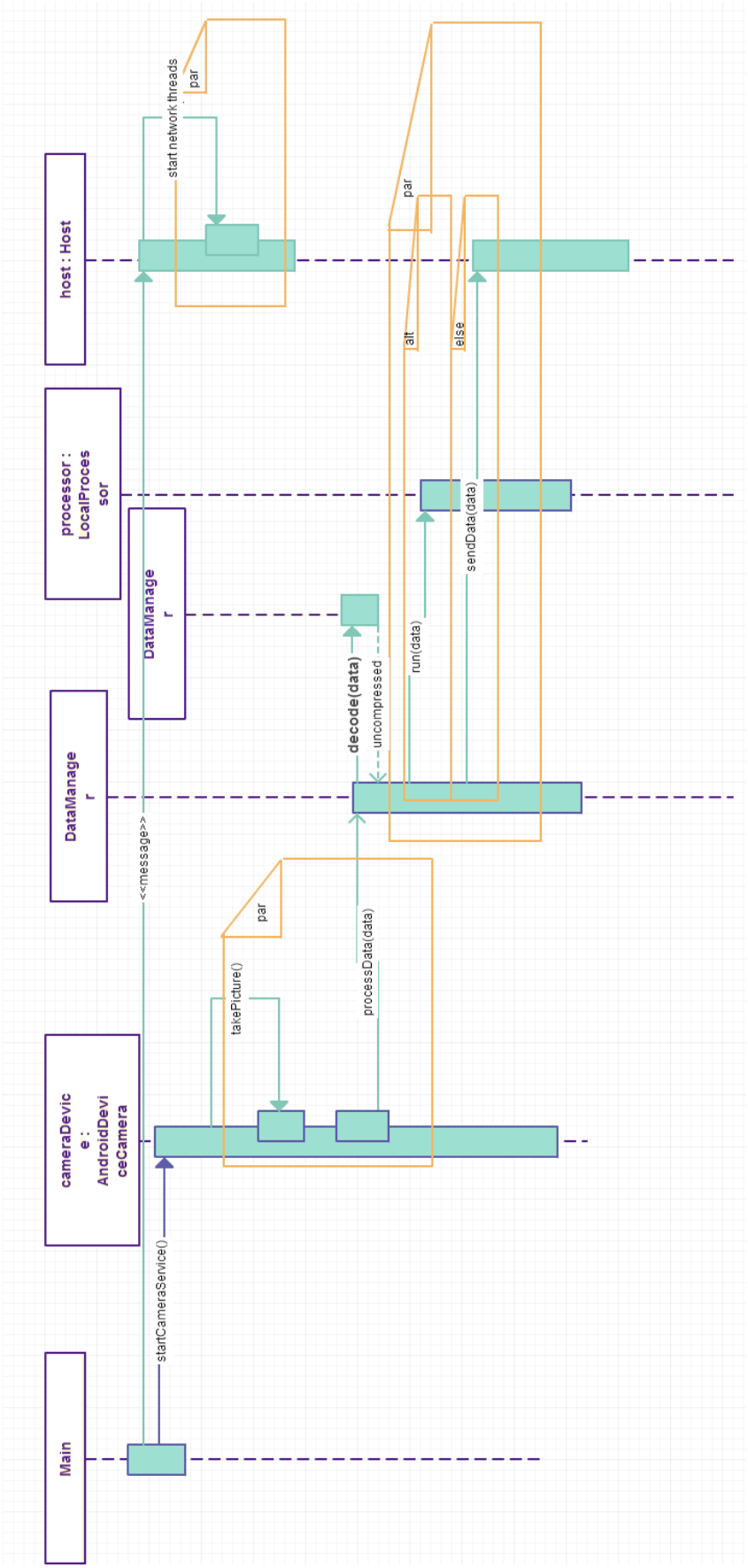


FIGUUR 25 OVERZICHTEN VAN VOORNAMELIJK STATISCHE OF SINGLETON KLASSEN.

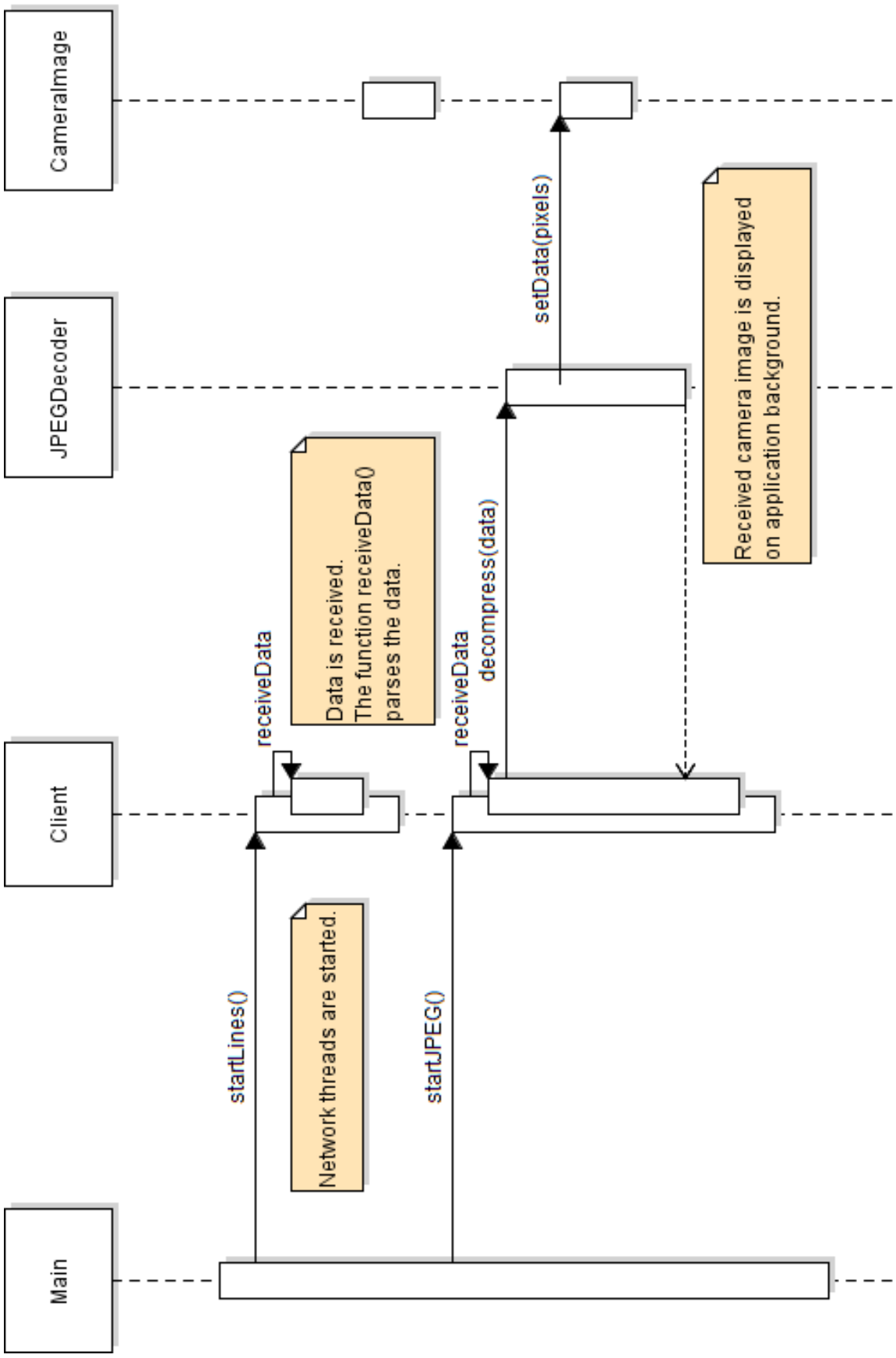
13.11.2 Sequentiediagrammen

In onderstaande figuren zijn sequentiediagrammen te zien van de volgende klassen:

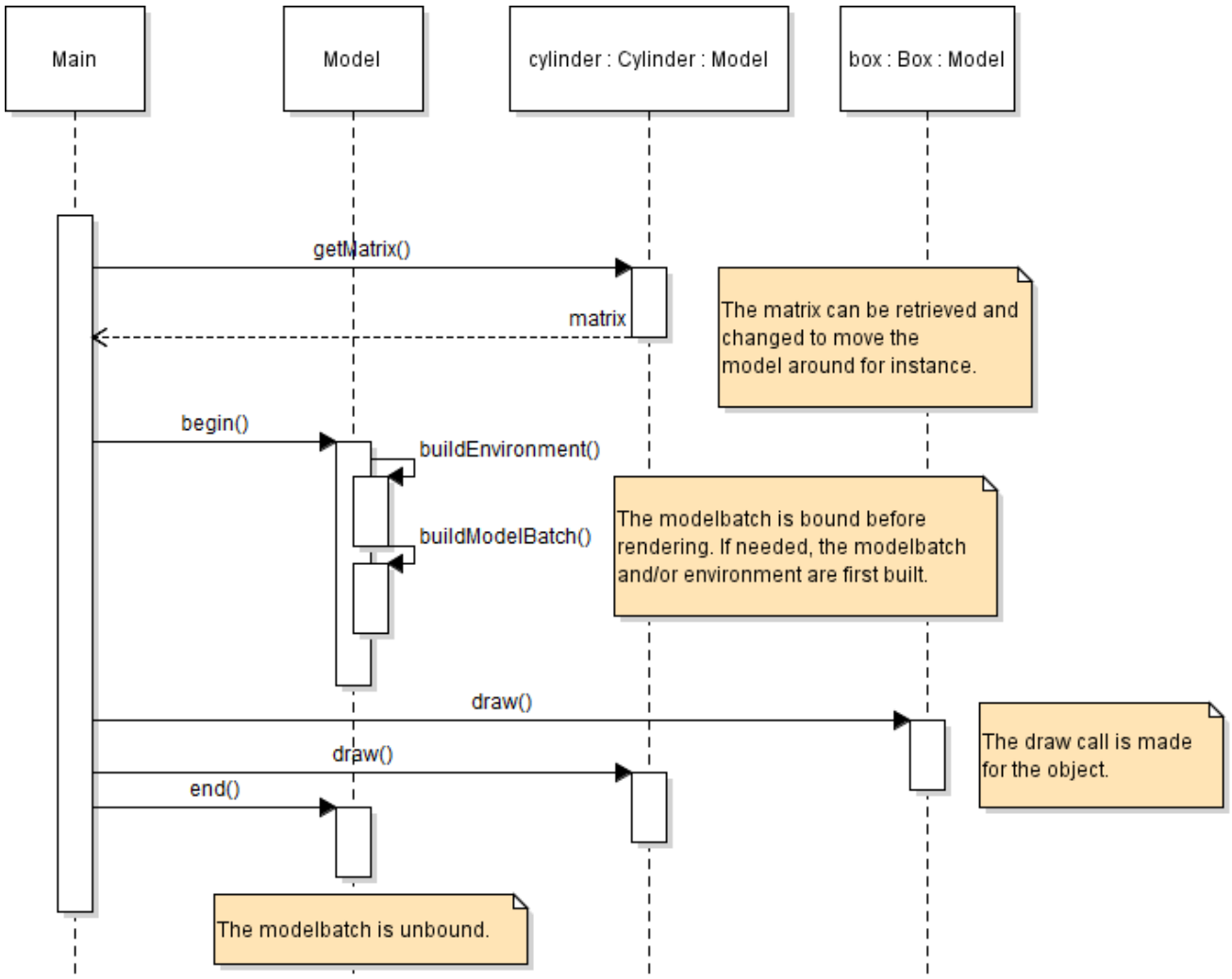
- Main-klasse: De hoofdklasse. Vanuit hieruit wordt het programma geinstantieerd.
- Client-klasse: De klasse die client-applicatie functionaliteit verzorgt.
- JPEGDecoder-klasse: Deze klasse is verantwoordelijk voor het decomprimeren van de JPEG-afbeeldingen.
- CameraImage-klasse: Deze klasse toont de camera-afbeeldingen op het scherm.
- Model-klasse: De abstracte klasse voor het opzetten en tonen van 3D objecten.
- Cylinder & Box–klassen: Instanties van de Model-klassen voor specifieke 3D objecten.
- CameraDevice-klasse: Verantwoordelijk voor het maken van de foto's. Een abstrace-klasse.
- AndroidCameraDevice-klasse: Een platformafhankelijke implementatie van de CameraDevice-klasse.
- DataManager-klaase: De klasse die de de processen van foto tot verzending of verwerking aanstuurt.
- LocalProcessor-klasse: Deze klasse verwerkt de foto lokaal d.m.v. beeldanalyse.
- Host-klasse: De klasse die de host-applicatie functionaliteit verzorgt.



FIGUUR 26 EEN OVERZICHT VAN HET MAKEN EN HET VERZENDEN VAN FOTO'S OP DE HOST-ZIJDE.



FIGUUR 27 EEN OVERZICHT VAN DE CLIENT-ZIJDE VAN START TOT ONTVANGST.



FIGUUR 28 EEN OVERZICHT VAN HET RENDERING-PROCES OP DE HOST-ZIJDE.

13.12 Gantt-chart

Projectplanner

Periodetrend: 1

Plan Werkelijk Werkelijk (meer dan gepland)

ACTIVITEIT	PLAN BEGIN	PLAN DUUR	WERKELIJK BEGIN	WERKELIJK DUUR
Beeldopnametechnieken	4	12	3	16
Communicatietechnieken	5	12	5	13
Mobiele beeldverwerking	7	10	8	10
Gebruikers feedback	5	10	6	11
Configuratie en debugging	4	6	4	8
Literatuuronderzoek	1	5	1	4
Systeemontwerp	4	12	3	9
Testen	7	10	6	12
Rapportverwerking	15	5	16	10