E03 / ELO / 07
Diepenbeek, 2003

Ontwerp van een database voor de analyse van videobeelden van appels.

Rapport over het eindwerk van
Wouter PAESEN en **Stany RASKIN**

Kandidaten voor de graad van Industrieel Ingenieur

Promotoren:

Dr. ir. Jan Genoe
Ing. Kristof Thewissen

Abstract

Dit eindwerk was een onderdeel van een HOBU-project met als doelstelling het ontwikkelen van een real-time fruitdetectie en -kwalificatiesysteem. Dit systeem moet kunnen gebruikt worden in automatische appelsorteer- en appelplukmachines.

De opdracht bestond erin een database te ontwikkelen van appels en de beelden hierin te analyseren. Voor de opbouw van de database zijn we vertrokken van realistische videobeelden, enerzijds van appels op de sorteerband en anderzijds van appels in de boomgaard net voor de pluk. De appelselecties in de videobeelden van de boomgaard dienden manueel te gebeuren. Bij de appels op de sorteerband was het mogelijk automatisch de selecties te bekomen door het grote kleurenverschil tussen appel en sorteerband. Al deze selecties van appels werden dan opgeslagen in de database.

De database is zo opgevat en uitgebouwd dat het mogelijk is om de inhoud ervan grondig te analyseren. Op een appelselectie uit de database dient eerst een preprocessing te worden uitgevoerd om een éénvormige voorstelling van de appelbeelden te krijgen. Preprocessing houdt o.a. in: herschalen, achtergrondinformatie uit het beeld halen, contrast en -belichtingsconditie normaliseren. Vervolgens dient een dimensiereductie te worden uitgevoerd door het extraheren van principale componenten. Verder werden aan de hand van histogrammen uitgebreide analyses op de beelden uitgevoerd. Hierdoor werd het mogelijk om het kwalificatiealgoritme te optimaliseren. Binnen het project zal deze optimalisatie nog van groot nut zijn wanneer deze in hardware moet worden geïmplementeerd om de real-time verwerking mogelijk te maken.

Voorwoord

Als afsluiter van ons opleiding als industrieel ingenieur dienen we aan de hand van een eindwerk te bewijzen wat we als ingenieur waard zijn. We hebben gekozen voor een eindwerk in het domein van de appelherherkening omdat dit de enige beschikbare keuze was die iets te maken had met artificial intelligence (AI). Dit was zeker iets dat ons beiden wel aansprak. Ook de uitdaging om mee te mogen te werken aan het HOBU-project dat het mogelijk zal maken real-time fruitpluk- en sorteermachines te realiseren heeft ons aangezet tot deze keuze. Met dit eindwerk kunnen wij dus ons steentje bijdragen in de technologische vooruitgang van de fruitsector.

We hebben veel steun gehad aan alle mensen die meegeholpen hebben aan de uitwerking van ons eindwerk. Het lijkt ons dan ook gepast op dit moment om een dankwoord te richten tot hen.

In de eerste plaats willen we onze promotoren dr. ir. Jan Genoe en ing. Kristof Thewissen bedanken omdat zij ons eindwerk zo goed opgevolgd en begeleid hebben. Verder richten we ook nog een dankwoordje tot lic. Jos Steverlinck voor zijn taalkundige ondersteuning en aan de projectmedewerkers voor hun nuttige tips en hulp. Ook willen we nog onze ouders bedanken. Zij hebben ons de kans gegeven om te studeren en hebben ons gesteund in de moeilijke momenten. Ook Ilse Verhoeven mag zeker niet vergeten worden.

Dit eindwerk heeft ons als toekomstig industrieel ingenieur een aantal vaardigheden bijgebracht zoals persoonlijke inzet en een vlotte individuele inbreng. Dit zal zeker nog van nut zijn in het verdere veloop van onze carrière.

Wouter en Stany

Inhoudsopgave

Lijst van tabellen

Lijst van figuren

Hoofdstuk 1

Situering van de afstudeeropdracht

1.1 Opdrachtgever

De opdrachtgever voor dit afstudeerwerk is de Katholieke Hogeschool Limburg [1], departement Industriële Wetenschappen en Technologie. Het afstudeerwerk maakt deel uit van een HOBU-project dat vanaf 1 september 2002 loopt aan de hogeschool.

1.2 HOBU-project

Naast het functioneren als onderwijsinstelling verrichten hogescholen in Vlaanderen vaak ook onderzoek. Traditioneel staan ze dicht bij het bedrijfsleven en zijn ze goed geplaatst om diensten aan de bedrijven te verlenen, ook aan kleinere bedrijven in minder hoogtechnologische sectoren.

In het verleden bleken de middelen waarover de hogescholen konden beschikken voor deze taken echter dikwijls te beperkt. Om hieraan in enige mate tegemoet te komen, besliste de Vlaamse regering eind 1996 een specifieke ondersteuning op te zetten om de interactie tussen de hogescholen en niet-technologiegedreven bedrijven te bevorderen, met als voornaamste doel de implementatie van kennis in de bedrijven. Deze ondersteuning wordt verzorgd door het "Instituut voor de Aanmoediging van Innovatie door Wetenschap en Technologie in Vlaanderen" of kortweg IWT [2]

In het kader van het zogenaamde HOBU-fonds werden sindsdien reeds zes oproepen voor projecten gelanceerd waarin in totaal 134 projecten werden geselecteerd, voor een totaalbedrag van 30,75 miljoen Euro. Telkens betreft dit projecten uitgevoerd door een hogeschool, in nauwe samenwerking met een groep bedrijven die zich verenigen in een gebruikerscommissie en met ondersteuning van een wetenschappelijke partner.

1.3 Real-time fruitdetectie en -kwalificatiesysteem

Als vervolg op het afstudeerwerk over "Hardware- en software-implementatie van een gezichtsdetectiesysteem" [3] dat in het academiejaar 2000-2001 in het departement IWT van de KHLim werd uitgevoerd, is een nieuw project opgestart dat probeert een oplossing te bieden op de vraag van de Limburgse fruitteeltsector naar automatische fruitsorteermachines.

De doelstelling van dit project is als volgt :

> In dit project zullen we het principale-componentenalgoritme [4, 5], dat in KULeuven Departement ESAT (Elektrotechniek) ontwikkeld werd om gezichten te herkennen

op videobeelden, aanpassen om appels te herkennen. Vervolgens zullen we dit implementeren in dedicated hardware zodat dit algoritme gebruikt kan worden in fruitpluk- en fruitsorteermachines. Een demonstrator van de door ons aangepaste fruitpluk- en sorteermachine zal aan de fruittelers en de fruitveilingen van de provincie Limburg getoond worden.

Samengevat houdt dit in dat we het bestaande principale-componentenalgoritme, *verkennen* en *vertalen* naar concreet uitgewerkte hardware voor toepassing in de fruitpluk en *verspreiden* naar de fruittelers van de provincie Limburg.

Voor een vlotte werking is het geheel opgedeeld in verschillende werkpakketten :

1. Databaseopbouw en principale-componentenextractie
2. Vertaling van de software naar een hardware-implementatie.
   (a) Aanpassing van de implementatie van de principale componenten.
   (b) Implementatie van de preprocessing
   (c) Implementatie van het histogram-equilibratiealgoritme in een virtex.
   (d) Verwerking van de bekomen principale componenten in een processor.
3. Ontwikkeling van een prototypebordje.
4. Inbouwen van het detectie- en selectiebordje in fruitpluk- en sorteermachines.
   (a) Inbouw in een fruitsorteermachine.
   (b) Inbouw in een fruitplukmachine.
5. Kenbaar maken van de resultaten aan de gehele fruitsector

Dit afstudeerwerk is de uitwerking van het eerste werkpakket.

1.4 Opdracht

Onderstaande tekst is de opdracht zoals die beschreven is in de projectaanvraag [6]

In dit werkpakket zullen we het bestaande algoritme aanpassen zodat het kan worden gebruikt voor het herkennen van appels, in plaats van het herkennen van gezichten. Hiervoor zullen we ons in dit project richten op twee variëteiten. Deze database wordt dan gebruikt om te bepalen welke matrixvermenigvuldiging moet worden uitgevoerd om het beeld te herleiden tot zijn principale componenten.

Ook in het algoritme zullen een aantal aanpassingen worden doorgevoerd, specifiek voor de toepassing op appels, die tot een vereenvoudiging leiden. Zo zullen we werken met een 30 pixel op 30 pixel frame in plaats van een 30 pixel op 40 pixel frame zoals dat voor gezichtsherkenning aangewezen is. Ook zullen we slechts de 20 meest relevante principale componenten uit ons beeld extraheren (in plaats van de 90 principale componenten voor de gezichtsherkenning van ESAT).

kwaliteit	(A)1	(A)3	(A)5
rijpheid	7	5	3

Figuur 1.1: Schematisch voorbeeld van één van de 40 variatiediagrammen beoordeeld door een fruitdeskundige

1.4.1 Opbouw van de database

Dit deelpakket vertrekt van vier digitale videotapes die opgenomen zullen worden onder de volgende omstandigheden:

- In een boomgaard, kort voor de appeloogst, met de camera op de positie zoals die zou zijn wanneer een plukmachine doorheen de gehele boomgaard zou rijden (dit voor twee appelvariëteiten);
- Op een bestaande fruitsorteermachine (ook voor twee appelvariëteiten).

We hebben bij de aanvang van het project, in overleg met de fruitdeskundigen, twee variëteiten gekozen. Één daarvan is van het groene ras (de Golden Delicious), omdat voor deze variëteit een herkenning op basis van de kleur alleen het moeilijkste is.

Op elk van de videotapes zijn minstens 2000 appels aanwezig. Deze tapes werden via een firewire IEEE 1394 verbinding op PC geplaatst en vervolgens zullen hierop manueel 2000 appels aangeduid worden die groter zijn dan 30 pixels op 30 pixels. Hierop worden dan dezelfde scaling en preprocessing toegepast die ook in het uiteindelijke systeem zullen worden toegepast.

Er zal ook een relevante database van niet-appels gemaakt worden uit dezelfde videotapes om het herkenningsalgoritme te toetsen.

1.4.2 Relevantieonderzoek van de principale componenten

Voor elke van de principale componenten zal onderzocht worden in welke mate de appelpopulatie varieert met deze principale component en in welke mate de niet-appelpopulatie varieert met dezelfde principale component. Deze analyse zal ons toelaten een eerste selectie van relevante principale componenten te maken.

1.4.3 Kwaliteitsanalyse door fruitdeskundigen

Voor elk van de relevante principale componenten uit de eerste selectie zal een variatiediagram worden gemaakt rond de gemiddelde appel.

Figuur 1.1 toont schematisch hoe een variatiediagram rond de gemiddelde appel er zou kunnen uitzien. Elke principaalcomponent geeft een variatie in beide richtingen rond deze waarde. Een fruitdeskundige uit de gebruikerscommissie kan op elk van deze variaties een kwaliteitsgraad en een rijpheidsgraad plakken. Mogelijk zullen we, eens de experimentele data ter beschikking zijn, in overleg met de fruitdeskundigen nog andere waardecijfers gebruiken. Op basis van deze kwaliteitsgraden en rijpheidsgraden zal de hardware die in dit project ontwikkeld wordt tezamen met de positie en grootte van de appel die herkend wordt ook een kwaliteitsgraad produceren.

- Deze kwaliteitsgraad kan door een plukmachine gebruikt worden om appels van een te lage kwaliteit niet te plukken, of de rijpheidsgraad kan gebruikt worden om onrijpe appels te laten hangen voor een volgende plukbeurt.
- Een automatische sorteermachine kan op basis van de kwaliteitsgraad, de rijpheidsgraad en de grootte van de appel de gewenste sortering uitvoeren.

Nadat de nodige gradatie door de fruitdeskundigen uitgevoerd is, verwerken we deze gegevens in een database. Met deze gegevens kunnen we een selectie maken van de 20 meest aangewezen principale componenten om een dubbel doel te bereiken:

- Het onderscheid maken tussen de klasse van de appels en de klasse van de niet-appels.
- Het objectief aangeven van een goed kwaliteitscijfer voor de appel.

Hoofdstuk 2

Overzicht van de beschikbare informatie

In dit hoofdstuk bespreken we de informatie en ervaring die in de hogeschool beschikbaar is in dit domein en de state of the art zoals die in de literatuur kan teruggevonden worden.

2.1 Huidige kennis die de literatuur biedt

Over reeds bestaande automatische appelplukmachines of appelplukmachines in ontwikkeling is bijna geen informatie te vinden, in tegenstelling tot de automatische sorteermachines waarover redelijk veel informatie te vinden is. Er werden hier namelijk op de hogeschool al een aantal eindwerken uitgevoerd die ook tot doel hadden om een automatische appelsortering te realiseren[7].

In deze eindwerken werd telkens de bestaande opstelling aangepast. Om de kleureninformatie te bekomen, maakte men wel steeds gebruik van een kleursensor. Door de weerkaatsing van het licht op de appels werd de kleureninformatie via een glasvezelkabel naar de kleursensor gebracht. Met een PLC werden dan de bewerkingen uitgevoerd om te bepalen tot welke klasse een bepaalde appel behoorde.

Eén van de problemen hierbij was dat de grootte van de brug waar de appel onderdoor moest afhankelijk was van de grootte van de appel. Indien men bijvoorbeeld de brug groter zou maken, zou de afstand voor de kleine appels tot de glasvezel te groot worden. Dit resulteerde in onnauwkeurige metingen. Deze slechte metingen leidden dan tot verkeerde klassebepaling van de appel. Een tweede probleem was dat de verwerkingssnelheid te laag was omdat men gebruik moest maken van een mechanische multiplexer. Dit was nodig omdat anders het aantal gebruikte kleursensors te groot zou worden.

Men heeft hier wel steeds verbeteringen op toegepast maar het gewenste resultaat werd nooit echt bereikt.

2.2 Punten in onderzoek waar vragen rijzen en onderzoek nodig is

Door technologische vooruitgang is het mogelijk geworden om nieuwe technologieën toe te passen. Zo zijn er in het academiejaar 1999-2000 aan de Katholieke Universiteit in Leuven Departement ESAT(Elektrotechniek) [8] twee gezichtsdetectietechnieken bestudeerd en uitgewerkt door twee studenten [4]. Beide technieken zijn uitgewerkt in Matlab[9]. Hierbij was het mogelijk om een gezicht te detecteren in een beeld. Dit gezichtsdetectiealgoritme is al toegepast in het eindwerk

van Veroniek Jacobs en Hans Vandermaesen [3] van twee jaar geleden. Het doel van dit eindwerk bestond erin om het algoritme op een bestaand hardwareplatform(ARIX) [10], dat een 'field programmable gate array' (FPGA) bevat, te implementeren.

2.2.1 Mogelijke wegen om met de huidige kennis de problemen te benaderen en op te lossen

In plaats van gebruik te maken van een kleursensor als meettoestel is het nu mogelijk een digitale kleurencamera te gebruiken. Deze mogelijkheid bestond al langer, maar hier werd niet voor gekozen omdat men een aantal argumenten had om deze methode niet te gebruiken. Men achtte zich toen nog niet in staat om de grote hoeveelheid informatie die in een beeld aanwezig is te verwerken. Dit is nu mogelijk door het uitvoeren van een dimensiereductie van gezichtsbeelden met behulp van de principale componentenanalyse.

Bij onze toepassing zullen er dus appels gedetecteerd moeten worden in een beeld. Hiervoor zal dus het algoritme aangepast en geoptimaliseerd moeten worden. We zullen vertrekken van het gezichtsdetectiealgoritme gebaseerd op probabilistische classificatie. Dit algoritme leert en optimaliseert zijn functionaliteit aan de hand van trainingsvoorbeelden van gezichten en niet-gezichten. Voor de appelherkenning zullen de trainingsdata bestaan uit een klasse van appels en een klasse van niet-appels. Onze database zal dus gebruikt worden om deze gegevens in op te slaan. Uit deze gegevens kunnen we dan bepalen welke matrixvermenigvuldiging uitgevoerd moet worden om het beeld te herleiden naar zijn principale componenten.

Het bestaande algoritme voor gezichtsdetectie uitgewerkt in Matlab heeft een goede performantie maar de snelheid van uitvoeren was te traag. Voor een beeld van 240 x 320 pixels duurt het ongeveer achttien minuten op een pentium II 450 MHz vooraleer het volledige beeld afgescand is op zoek naar gezichten. Om real-time toepassingen mogelijk te maken zal dit algoritme dus versneld moeten worden. In het eindwerk van Hans Vandermaesen en Veroniek Jacobs [3] is er al een grote stap gezet om dit mogelijk te maken. Zij hebben de meest tijdsintensieve delen(matrixvermenigvuldigingen) van het algoritme in de FPGA geïmplementeerd. Deze hardware implementatie was ongeveer 81 maal sneller dan de software implementatie. Hierdoor werd de globale verwerkingsnelheid met ongeveer 42% verhoogd. Een volledige hardware-implementatie zou dit nog kunnen verbeteren.

2.3 Beschrijving van het gezichtsdetectiealgoritme

Men kan het probleem van gezichtsdetectie terugbrengen tot een classificatieprobleem. De algemene aanpak voor het lokaliseren van een gezicht in een beeld is het exhaustief doorzoeken van het beeld. Hierbij gaat men vensters van een bepaalde grootte over het beeld schuiven en telkens controleren of de beeldinhoud van het venster al dan niet een gezicht voorstelt. Dit is een zeer tijdrovende operatie.

Het algoritme voor gezichtsdetectie waarvan we gaan vertrekken is in grote lijnen opgebouwd uit:

- Preprocessing: het doel hiervan is om een éénvormige voorstelling te krijgen van gezichtsbeelden en houdt volgende belangrijke bewerkingen in :
  - herschalen;
  - achtergrondinformatie uit beeld halen;
  - contrast normaliseren;
  - belichtingsconditie normaliseren.

- Het uitvoeren van een dimensiereductie van gezichtsbeelden met behulp van de principale componentenanalyse.
- Probalistische classificatiemethode.

2.4 Vereenvoudigingen aan het algoritme voor de detectie van de appels

- Een masker van $30 * 30$ pixels kan volstaan (i.p.v $30 * 40$ zoals bij gezichtsherkenning aangewezen is).
- We zullen slechts 20 principale componenten uit het beeld extraheren (i.p.v 90 voor de gezichtsherkening van ESAT).
- Bij de sorteermachine staat de camera op vaste afstand t.o.v sorteerband (rescaling valt weg).
- We hebben een uniforme achtergrond bij de sorteermachine zodat het herkennen van de appels sterk vereenvoudigd kan worden.

2.5 Criteria waarop de appels gesorteerd worden

- grootte

  Als parameter voor de grootte van de appel gebruikt men de gemiddelde waarde van een aantal metingen. De grootste waarde van de metingen bepaalt de lengte en de kleinste waarde bepaalt de diameter. Deze diameter kan variëren van 60 mm tot 120 mm. De onderverdeling naar diameter gebeurt per 5 mm. Zo zal een eerste onderverdeling diameters van 60 mm tot 65 mm omvatten (d.i. maat 60), een tweede diameters van 65 mm tot 70 mm, enz. De verdeling stopt echter bij 100 mm. Alle appels waarvan de diameter de 100 mm overschrijdt, worden ondergebracht in de verdeling boven de 100 mm. Zo geldt ook voor appels met een diameter kleiner dan 60 mm een toekenning onder de verdeling beneden de 60 mm.

- kleur en kwaliteit

  De appel is een natuurproduct en elke partij appels vertoont dus ook andere kenmerken waarbij de interpretatie van de overgangen tussen de verschillende kleuren subjectief benaderd worden. Daarom is het belangrijk om verschillende grens- en overgangswaarden voor appelsortering zelf in te stellen. De enige informatie die we hierover kunnen krijgen is de ervaring in een fruitbedrijf. Daarom hebben we een bezoek gebracht bij fruitteler (Baerts Constant) om hierover informatie in te winnen. Er wordt een verschil gemaakt tussen het rode ras (Jonagold) en de groene ras (Golden). Hieronder beschrijven we de informatie die we over het sorteren gevonden hebben voor het rode en het groene ras.

2.6 Rode ras *(Jonagold)*

Het rode ras wordt in klassen verdeeld naar de volgende criteria:

- verhouding van rode oppervlakte t.o.v. de totale oppervlakte (in % uitgedrukt);
- intensiteit van het rood (zeer donker rood, matig tot veel rood, weinig rood).

Figuur 2.1: Afbeelding van Jonagold appels. In de linkse kist bevinden zich appels van klasse A3, in de rechtse kist appels van klasse A2

Deze appels deelt men op in vijf klassen A1 tot A5. Uitsortering van de volgende fenomenen moet mogelijk zijn:

- bluts;
- stipvorming: door gebrek aan mineralen zoals calcium;
- grijsvorming;
- schurft;
- hagelschade;
- scaldvorming: door oxidatie van een zuur, vooral bij donkerrode-paarse appels;
- vogelpik;
- rotte plek.

2.7 Groene ras *(Golden)*

Het groene ras wordt in klassen verdeeld volgens de volgende criteria:

- achtergrondkleur;
- grijsvorming;
- aanwezigheid van blos.

De achtergrondkleur is de kleur die overheerst aan de kant van de appel die het minste zon gezien heeft bij het groeien. Met grijsvorming bedoelt men de grofheid van de schil die meestal het duidelijkst is rond het steeltje van de appel.

Figuur 2.2: linksonder ziet u een appel van het Golden Delicious ras klasse A3 en rechtsboven hetzelfde ras maar klasse A2. De grijsvorming rond de steel is goed zichtbaar.

Deze appels deelt men op in 3 klassen (A1, A2, A3) en dit op basis van achtergrondkleur en grijsvorming. Hier is nog een bijkomende klasse-indeling die de rijpheid weergeeft: groen “++” , groengeel “+”, lichtgeel “\”, Rijp (geel) “R”.

Dit is een belangrijk gegeven want dit bepaalt aan welke eisen het systeem moet voldoen. Door het feit dat er geen eenduidige indeling in klassen vastligt, moet het dus mogelijk zijn de indeling in klassen dynamisch in te stellen. Dit wil dus zeggen dat de grenzen om tot een bepaalde klasse te behoren aangepast moeten kunnen worden.

Hoofdstuk 3

Analyse van de opdracht

3.1 Voorbereidend werk

Om op een vlotte manier aan deze opdracht te kunnen beginnen, werd er in de maanden juli, augustus en september 2002 gewerkt aan een project om deze opdracht te ondersteunen. Om de videobeelden op een vlotte manier te kunnen verwerken, is er gespecialiseerde software nodig. Een probleem is dat er nergens een softwareleverancier gevonden is die de projectgroep van deze programmatuur kon voorzien. Om deze reden is het "snapshot" project opgericht.

Dit is een project met als doel te voorzien in programmatuur voor verwerking van videobeelden. Het verwerken van videobeelden moet hier gezien worden als het analyseren van videobeelden, niet als het bewerken van de beelden om een mooie montage te verkrijgen.

Om dit project op te kunnen bouwen, is er allereerst een onderscheid gemaakt tussen de delen die specifiek zijn voor het HOBU project en de delen die gebruik maken van algemene kennis. Deze algemene delen worden samengebracht in een softwarepakket dat onder de GPL-licentie [11] vrijgegeven wordt. De specifieke delen blijven niet-vrijgegeven kennis van de projectgroep.

Door de algemene delen vrij te geven worden er twee doelen bereikt :

- Allereerst wordt ervoor gezorgd dat andere projectgroepen gebruik kunnen maken van deze ondersteunende software, zodat al het werk dat hierin steekt niet steeds opnieuw gedaan moet worden.
- Daarbij komt nog dat anderen die de software gebruiken, toegang hebben tot de bron van het programma en eventuele fouten kunnen aanpassen. Zij kunnen deze aanpassingen aanbrengen aan het originele programma zodat uiteindelijk vele fouten uit het programma opgelost worden door collectief werk.

Gedetailleerde uitleg over deze software is op de website van het snapshot project [12] te vinden.

3.2 Beschikbare hulpmiddelen

Bij de aanvang van dit eindwerk zijn ons ter beschikking gesteld :

een grote hoeveelheid videobeelden : De videobeelden zijn verkregen zoals beschreven in paragraaf 1.4.1. Bij de aanvang van dit eindwerk zijn deze beelden gedigitaliseerd en opgeslagen op cd-rom's. Uit deze beelden zal informatie gedistilleerd moeten worden waarmee een goed

algoritme opgebouwd kan worden. Dezelfde beelden kunnen in een later stadium gebruikt worden om het opgestelde algoritme te toetsen.

een aantal PC's : Eén van deze PC's is qua verwerkingskracht superieur aan de twee andere. Deze zal dan ook gebruikt worden om rekenintensieve analysetaken uit te voeren. Om licentiekosten voor software te verminderen is er besloten om als besturingssysteem Debian GNU\Linux te installeren. Dit systeem is daarenboven ook nog veel stabieler dan andere commerciële besturingssystemen, wat uiteraard alleen maar voordelen heeft.

Een andere motivatie voor deze keuze is de beschikbaarheid van informatie en programmatuur. Verschillende functies die voor deze opdracht nodig zijn, werden reeds geïmplementeerd in andere softwarepaketten. Door de aard van de licentie waaronder deze programma's vallen, is het mogelijk om deze functies over te nemen. Dit geeft een belangrijke vermindering in ontwikkeltijd omdat we deze functionaliteiten niet meer zelf moeten programmeren en ook omdat de overgenomen functionaliteit minder fouten zal bevatten dan zelfgeschreven code.

inzage in andere eindwerkrapporten : Het betreft eindwerken die aan de Katholieke Hogeschool Limburg gelopen hebben en die over hetzelfde onderwerp handelen[7]. Ook de informatie van het eindwerk over gezichtsherkenning[3] mag gebruikt worden. Een samenvatting van de voor ons belangrijke informatie kunt u lezen in hoofdstuk 2.

3.3 Planning

Om de opdracht tot een goed eind te kunnen brengen, hebben we een aantal stappen gedefiniëerd. Deze stappen komen grotendeels overeen met de stappen in paragraaf 1.4. De afwijkingen hebben te maken met het feit dat er misschien ook nog andere methodes dan het principalecomponentenalgoritme gebruikt zullen worden.

In een eerste reeks van stappen zal er getracht worden een goed algoritme op te stellen om de eigenschappen van appels op een transportband te bepalen. Als dit algoritme volledig uitgewerkt is, gaan we over naar een systeem om karakteristieken van appels in een boomgaard te bepalen. Hierbij kunnen we heel waarschijnlijk gebruik maken van de kennis opgedaan bij de transportband.

3.3.1 Opbouw van de database

Om representatieve gegevens te vinden, moet er eerst een goede database zijn. Dit houdt in dat we uit de digitale videobeelden de relevante delen selecteren. Relevante delen van het beeld zijn in dit geval alle delen van het beeld waarin zich een appel bevindt. Hierdoor is het mogelijk om in volgende stappen enkel deze relevante delen te gebruiken.

Voor videobeelden op de transportband moet het mogelijk zijn de relevante delen automatisch te bepalen op basis van het grote verschil tussen de transportband en de appels. Bij beelden van de boomgaard zal dit iets moeilijker zijn. Waarschijnlijk zullen hierbij de relevante delen manueel aangeduid moeten worden.

3.3.2 Analyse van de beelden

Als alle relevante delen gekend zijn, zullen er analyses uitgevoerd worden op de videobeelden. De data die verkregen worden bij deze analyses, worden op een of andere manier opgeslagen om ze te kunnen gebruiken achteraf. De gegevens worden naar alle waarschijnlijkheid opgeslagen in tekstbestanden.

3.3.3 Onderzoek naar mogelijke algoritmes

Uit de resultaten van de geanalyseerde beelden zal er beslist moeten worden welke algoritmes toegepast kunnen worden. Natuurlijk de manier waarop de beelden geanalyseerd worden grotendeels bepaald gaat zijn door de algoritmes waarvan we denken dat ze bruikbaar zijn.

3.3.4 Implementatie en uittesten van de algoritmes

Zogezien we weten welke algoritmes er toegepast zullen worden, zal de implementatie van deze algoritmes uitgevoerd worden. Ook zullen er verschillende tests uitgevoerd worden om te controleren of de algoritmes echt wel de resultaten geven die ervan verwacht worden. Deze tests zullen gebruik maken van de originele videobeelden.

3.3.5 Uittesten van de algoritmes op onbekende gegevens

Als de algoritmes bevredigende resultaten geven op de geanalyseerde videobeelden, zullen er testen uitgevoerd worden op nieuwe, nog niet geanalyseerde, videobeelden. Uit de resultaten van deze tests zal er dan een beeld gevormd worden van de doeltreffendheid van het opgestelde systeem.

3.3.6 Optimalisatie van de algoritmes

Vanaf het moment dat er een werkend geheel bekomen is, zullen we nog proberen om de algoritmes te optimaliseren. Eventueel kan er eerst nog geoptimaliseerd worden om betere gegevens te bekomen (betere kwaliteitscijfers enz.). Er zal zeker getracht worden om de algoritmes zo te optimaliseren dat ze real-time uitgevoerd kunnen worden.

3.3.7 Uittesten van de geoptimaliseerde algoritmes

Om er zeker van te zijn dat we bij het optimaliseren van de algoritmes geen fouten geïntroduceerd hebben, worden ze aan dezelfde testen onderworpen als de niet-geoptimaliseerde algoritmes. Door de resultaten van beide tests te vergelijken kunnen we een goed beeld opbouwen van alle fouten en afwijkingen die er ontstaan zijn bij het optimaliseren.

Hoofdstuk 4

Opbouw van de programmakern

De ondersteunende software ontwikkelen heeft een groot deel van onze tijd in beslag genomen. We hebben deze software zo opgebouwd dat het mogelijk is om de functionaliteit eenvoudig uit te breiden door onafhankelijke modules te definieren. Op deze manier wordt een flexibel systeem verkregen dat ook nog toegepast kan worden in andere projecten die zich bezighouden met beeldanalyse. Het project is geschreven in C++.

De software is gelicentieerd als Vrije Software[11] wat wil zeggen dat iedereen hier gebruik van kan maken, en dat iedereen over de broncode kan beschikken.

In dit hoofdstuk bespreken we de opbouw van de software, het gebruik ervan, en de manier waarop de modules gemaakt moeten worden.

4.1 Globale structuur van het programma

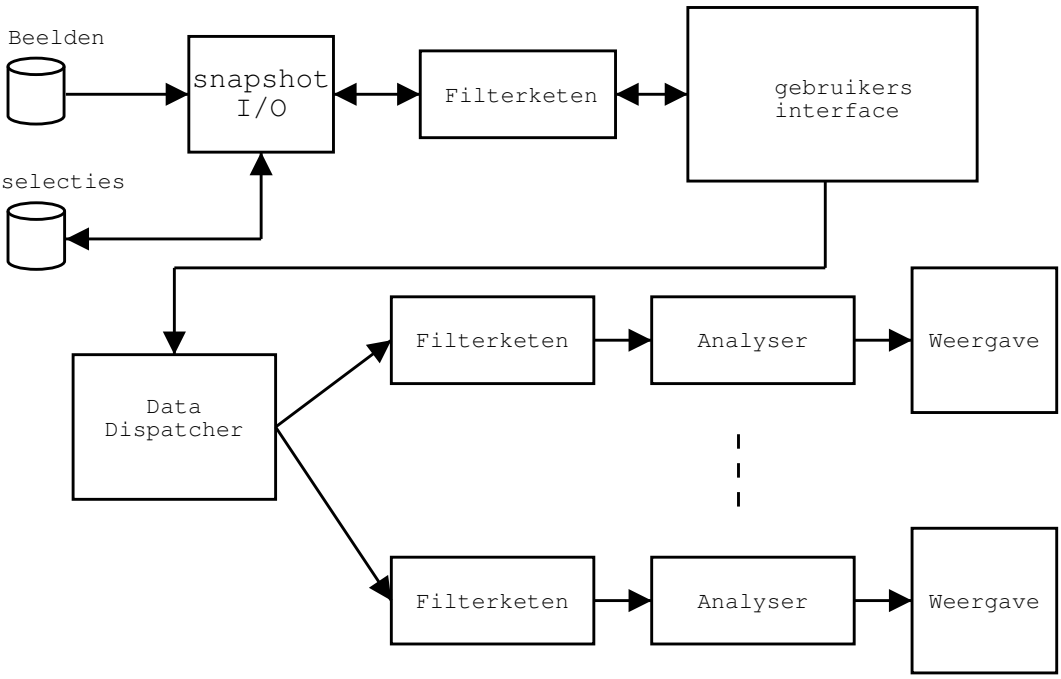
Oorspronkelijk was het de bedoeling om deze software enkel te gebruiken voor het aanduiden en extraheren van appels in de database met videobeelden. Tijdens het werken aan de software is het idee gegroeid om deze ook te gebruiken om de analyses op de beelden uit te voeren.

Daarom hebben we geprobeerd om de structuur van deze software zo flexibel mogelijk te houden. Hierdoor bestaan de mogelijkheden om met weinig werk een nieuw soort analyse op te zetten of eventueel op een nieuwe gegevensdatabase identieke analyses uit te voeren. Dit zou nuttig kunnen zijn bij het uitbreiden van het project naar andere fruitsoorten bijvoorbeeld.

De software heeft op dit moment twee grote functies :

- Het verwerken van volledige videobeelden en het aanduiden van de gebieden waar zich interessante gegevens bevinden kan zowel door een gebruiker gebeuren (manueel) als automatisch. De coordinaten van de aangeduide gebieden worden opgeslagen in een aparte database. We slaan niet de aangeduide gebieden op, omdat dit resulteert in een dubbele opslag van beeldgegevens.
- Het analyseren van de aangeduide gebieden.

De bedoeling van deze software is *niet* om een volledig pakket voor beeldanalyse te schrijven. Het is wel bedoeld om te helpen bij beeldanalyse, in die zin dat een grote hoeveelheid beelden geanalyseerd kan worden, zonder dat er tijd gestoken moet worden in de manier waarop beelden ingelezen worden, enz. De gebruiker krijgt een omgeving ter beschikking waarin hij analyses kan uitvoeren en de resultaten ervan kan laten zien, of kan opslaan.



Figuur 4.1: Globale structuur van snapshot.

Daar er betere pakketten beschikbaar zijn om statistische bewerkingen en dergelijke op de resulaten van analyses uit te voeren, is deze functionaliteit niet voorzien.

In figuur 4.1 is de algemene opbouw van het systeem weergegeven. De blokken worden in de volgende paragrafen wat beter toegelicht

4.1.1 Snapshot I/O

De I/O zorgt voor het uitlezen van twee bestanden. Het eerste bestand bevat de videobeelden en het tweede bestand bevat eventueel aangeduide gebieden, verderop selecties genoemd. Ieder videobeeld is gekenmerkt door twee gegevens : de naam van het videobestand en het volgnummer van het beeld in die videobestand. Deze twee gegevens worden dan ook gebruikt om voor iedere selectie aan te geven op welk beeld ze van toepassing zijn. De beeldgegevens worden samengebundeld met de selectiegegevens zodat ze intern in het programma altijd samen terug te vinden zijn. Deze bundel wordt door het I/O blok doorgegeven aan de filterketen.

Dit blok is niet bruikbaar voor het opslaan van gegevens van analyses omdat deze gegevens zo verschillend kunnen zijn dat het bijna onmogelijk is om hier een universele interface voor te voorzien. Opslag van analyseresultaten moet door moduleprogrammeurs zelf geïmplementeerd worden.

Het formaat waarin de beelden opgeslagen zijn, kan variëren. Het is wel raadzaam om voor een verliesvrij formaat te kiezen, om zo geen nuttige gegevens te verwaarlozen. We maken gebruik van de avifile bibliotheek[13] om beelden in te lezen. Alle bestandsformaten die door deze bibliotheek ondersteund worden, zijn in principe bruikbaar.

De database met selecties is een SQLite[14] database. SQLite is een bibliotheek waarmee men een volledige SQL database in een bestand kan steken. Het bestandsformaat is binair, maar de gegevens zijn er zeer eenvoudig uit te halen door middel van een meegeleverd programma dat deze bestanden kan lezen, of door een programma dat deze bibliotheek gebruikt.

4.1.2 Filterketen

Een filterketen is, zoals de naam het aangeeft, een aaneenschakeling van filters (waarover verderop in deze tekst meer). De gegevens die uit het I/O blok komen, worden door deze keten gestuurd voor ze op het scherm weergegeven worden. Dit maakt het mogelijk om bijvoorbeeld de invloed van belichtingscondities weg te werken zodat het beeld beter is voor de gebruiker.

4.1.3 Gebruikersinterface

De gebruikersinterface zorgt voor de interactie met de gebruiker. Met de muis kunnen gebieden aangeduid worden. Op deze gebieden kunnen dan bepaalde analyses uitgevoerd worden. Ook kan de gebruiker specifiëren welk videobestand moet worden geopend en in welk bestand de selecties moeten worden opgeslagen. Deze interface is ontwikkeld door gebruik te maken van de GTKmm[16] Grafische Interface ontwikkelset. Dit is een c++ ontwikkelset die oorspronklijk ontwikkeld is voor het populaire GIMP tekenprogramma. Ze wordt vooral gebruikt in GNU/Linux omgevingen, maar is ook bruikbaar in een Windows omgeving.

4.1.4 Data Dispatcher

Om ervoor te zorgen dat er nog meer bewerkingen toegepast kunnen worden op de beeldgegevens, hebben we een systeem gedefinieerd dat toelaat in aparte vensters gegevens weer te geven die afgeleid zijn van de beeldgegevens. Dit systeem is zo opgezet dat zo weinig mogelijk werk overgedaan moet worden. Dit wil zeggen dat eventuele functionaliteiten die al eerder in een filter gedefinieerd zouden zijn, opnieuw gebruikt kunnen worden voor de analysers. We hebben er ook voor gezorgd dat het uitvoeren van analyses de rest van het programma niet hindert, door voor iedere analyse een aparte thread [1] te gebruiken. Om ervoor te zorgen dat alle gegevens op de juiste plaats terechtkomen maken we gebruik van een dispatcher. De belangrijkste taken van deze dispatcher bestaan erin een lijst met geopende analyses bij te houden en ervoor te zorgen dat de beeldgegeven op het juiste moment aan de analysestructuren afgeleverd worden.

4.1.5 Analyser

Analysers zijn net als filters aparte modules, die opgebouwd zijn volgens eenzelfde logische structuur. Door een analyser te registreren bij de Data Dispatcher wordt de beschreven analyse steeds uitgevoerd als er nieuwe gegevens beschikbaar zijn. Wat de analyser precies doet, en op welke manier de resultaten hiervan voorgesteld worden is volledig overgelaten aan de moduleprogrammeur. Hoe een analyser opgebouwd moet worden, wordt beschreven in 4.4.

4.1.6 Weergave

Samen met een analyser kan de moduleprogrammeur een systeem voorzien dat de resultaten van een analyse in een apart venster weergeeft. Om flexibel te werken, is de manier waarop dit venster opgebouwd is eveneens overgelaten aan de moduleprogrammeur. De manier waarop deze weergave opgebouwd moet worden, is eveneens beschreven in 4.4.

[1] thread: draad van uitvoering. Door gebruik te maken van de mogelijkheden van het besturingssysteem is het mogelijk om in één programma meerdere paden van uitvoering te krijgen.

4.2 Frames en selecties

Om de beelden en de selecties samen te bundelen hebben we enkele klassen gedefinieerd. Een beeld dat van een beeldbestand gelezen wordt, wordt opgeslagen in een klasse “IFrame” (initieel frame). Hierbij wordt ook opgeslagen uit welk bestand het betreffende beeld komt, en welk volgnummer het beeld heeft. Deze klasse voorziet ook in de mogelijkheid om de RGB gegevens die we van het bestand lezen om te zetten naar de veel gebruikte HSI kleurenruimte en omgekeerd.

De klasse “MFrame” is van de “IFrame” klasse afgeleid en voegt de mogelijkheid toe om ook een lijst met selecties bij het beeld te voegen. Deze klasse wordt gebruikt door de gebruikersinterface om het beeld samen met de selecties te tekenen op het scherm.

Een derde klasse die we nodig hebben is “SFrame”. Ook deze is afgeleid van de “IFrame” klasse en stelt eigenlijk een uitgesneden kadertje voor. Daarom is het ook niet mogelijk om een “SFrame” klasse rechtstreeks te instantiëren. Het is enkel mogelijk een “SFrame” klasse te maken door een “MFrame” en een selectie te combineren. De “SFrame” klasse houdt de afmetingen bij van het “MFrame” waar ze uit afgeleid is, en de coördinaten in dit originele beeld waar de beeldgegevens zich precies bevinden.

Omdat al deze klassen afstammen van IFrame is het mogelijk om in een variabele met type IFrame* alle soorten van frames op te slaan.

4.3 Filters

Filters zijn functionele blokken in het programma die gebruikt kunnen worden om zowel de inhoud van het beeld dat ze verwerken als de selecties die bij het beeld horen aan te passen. Filters worden als onafhankelijke modules gedefineerd. Dit wil zeggen dat het aanpassen van een filter onafhankelijk van het hoofdprogramma kan gebeuren, maar om dit mogelijk te maken, moeten de filters wel volgens een vast patroon opgebouwd zijn.

In volgende paragrafen beschrijven we kort hoe een filter gedefinieerd moet worden. Voor meer gedetailleerde informatie verwijzen we naar de broncode en de documentatie van het snapshot project [12]. Een overzicht van de verschillende filters vindt u in 5

4.3.1 Definitie van een filter

Om een filter te definiëren, wordt een nieuwe klasse gedefinieerd die afgeleid is van de klasse FrameFilter. Door de interne definite van deze klasse moeten volgende methoden en eigenschappen door de moduleprogrammeur gedefinieerd worden :

- *constructor*
- *destructor*
- *Glib::ustring name();*

Deze methoden zorgen ervoor dat de filter geïnstantieerd kan worden, en ook uit het geheugen verwijderd kan worden. En de laatste zorgt ervoor dat snapshot eenvoudig kan terugvinden welk type filter een bepaalde instantie is. Ander methoden die de moduleprogrammeur *kan* implementeren zijn :

- *virtual void visualProcessRead();*

- *virtual void visualProcessWrite();*
- *virtual void selectionsProcessRead();*
- *virtual void selectionsProcessWrite();*

In deze vier methoden moeten de acties uitgevoerd worden die het gedrag van de filter bepalen. visualProcessRead en selectionsProcessRead zorgen voor het verwerken van respectievelijk de beeldgegevens en de selecties tijdens het lezen van een beeld. De andere twee worden uitgevoerd tijdens het terugschrijven van een beeld (in de hoofdketen).

In deze methodes kan de programmeur steeds verwijzen naar de variabele frame (van het type "IFrame*") om de beschikbare beeldgegevens te verkrijgen. Deze gegevens mogen gewijzigd worden, en er mag zelfs een volledig nieuw adres in frame geschreven worden.

De selectionsProcessRead en -Write methoden worden enkel aangeroepen door het programma als het beeld dat we verwerken van het type "MFrame" is. De programmeur moet er zelf voor zorgen dat de variabele frame omgezet wordt naar het type "MFrame". Dit kan doorgaans met een cast operatie gebeuren als volgt:

MFrame* mframe = dynamic_cast<MFrame*>(frame);

Indien mframe de waarde NULL zou bevatten is frame niet van het type MFrame en mag de verwerking van selecties afgebroken worden.

4.3.2 Filterdescriptor en parameters

Er is reeds eerder vermeld dat het mogelijk is om parameters te definiëren en er waarden aan toe te kennen. Dit wordt niet rechtstreeks in de filter gedaan, maar door een bijhorende klasse die afgeleid is van de klasse FilterDescriptor. Deze descriptor bevat alle gegevens en methoden om ervoor te zorgen dat de gebruiker van de filters, op een flexibele manier de parameters aan kan passen.

Er zijn talrijke voorbeelden in de bibliotheek "libstdfilters" die we zelf gebruikt hebben voor de analyses van de beelden. Het komt erop neer dat de programmeur van de filter ervoor moet zorgen dat het mogelijk is om via een zelf gekozen methode parameters te lezen en te schrijven. De descriptor vormt een omzetting tussen de globale methode die snapshot gebrukt, en de zelf gekozen methode die de programmeur gebruikt. Om ons niet te beperken tot types van parameters is ervoor gekozen om de parameters in snapshot zelf voor te stellen als tekenreeksen. Dus getallen worden niet binair voorgesteld maar door een opeenvolging van karakters zoals we die zelf op papier zouden schrijven. De programmeur van de descriptor moet er zelf voor zorgen dat deze tekenreeksen op een correcte manier omgezet worden naar het gepaste type (voor het schrijven van parameter) of andersom (voor het lezen van parameters).

Daarbij komt nog dat de descriptor ervoor zorgt dat snapshot nieuwe instanties van een filter kan aanmaken, de naam en een korte omschrijving van de filter kent. Ook kan via de descriptor een lijst met parameters opgevraagd worden waarin van iedere parameter het type, de naam en een korte omschrijving staat.

4.4 Analysers

Analysers worden gebruikt als de beeldgegevens niet gewijzigd worden, of als de resultaten van een bewerking in een apart venster weergegeven moeten worden. Ook hier geldt weer dat ze apart van snapshot gedefinieerd worden.

4.4.1 Opbouw van een analyser

De opbouw van een analyser is iets strikter omdat we ervoor gezord hebben dat de moduleprogrammeur zo weinig mogelijk rekening moet houden met de problemen die zich voordoen bij het gebruik van meerdere uitvoeringsaders. Om dit mogelijk te maken moet een analyser altijd afgeleid zijn van de klasse Analyser. De programmeur moet volgende zaken zelf voorzien :

- *constructor en destructor*
- *void rotate_ud();*
- *gboolean update_ud();*
- methodes om de resultaten van een analyse te bemachtigen

Constructor en destructor zorgen voor het opbouwen van een instantie en voor het afbreken van een instantie. De methode “rotate_ud” zorgt ervoor dat de interne buffer en de externe buffer omgewisseld worden. In de methode “update_ud” moet de eigenlijke analyse uitgevoerd worden.

De opslag van de resultaten van een analyse moet door de moduleprogrammeur zelf gedefinieerd worden. Om ervoor te zorgen dat er geen problemen optreden bij het lezen van de gegevens door een eventueel weergavevenster, worden er twee identieke buffers gebruikt. Één van deze buffers is voor intern gebruik (binnen de klasse), de andere is voor extern gebruik (bijvoorbeeld door een weergavevenster). In de methode “rotate_ud” moet de programmeur ervoor zorgen dat de interne en de externe buffer omgewisseld worden. De meest aangewezen methode is om een nieuwe klasse te definiëren waarin alle analysegegevens opgeslagen worden. Er kan dan een array met twee instanties van deze klasse gedefinieerd worden. De ene keer is het eerste lid van de array intern, en het tweede lid van de array extern. De volgende keer is het tweede lid van de array intern en het eerste lid extern.

Tijdens de uitvoering van “update_ud” kan de programmeur altijd met de methode “source_data()” de gegevens ophalen waarop een analyse uitgevoerd moet worden. Deze gegevens komen van de dispatcher en zijn ondertussen al door de filterketen, die bij de analyser hoort, gestuurd. De programmeur mag deze gegevens alleen gebruiken in het uitvoeringspad van “update_ud”. Ze mogen niet gewijzigd worden en moeten niet vrijgegeven worden door de programmeur.

Voorbeelden zijn beschikbaar in de bibliotheek “libstdanalysers” waarvan we in ons project regelmatig hebben gebruik gemaakt.

Ook een analyser kan parameters hebben. Om ervoor te zorgen dat een analyse opnieuw uitgevoerd wordt als de parameters wijzigen, is er een standaard opslagplaats voor parameters gedefinieerd. Deze opslag van parameters werkt weer volgens hetzelfde principe als bij de filters. Iedere parameter wordt gekenmerkt door een naam als tekenreeks. Ook de waarde die erbij hoort wordt als tekenreeks opgeslagen. De opslag van parameters en de toegang tot de parameters is reeds voorzien en de methoden hiervoor zijn ook al beschikbaar :

- *void set_property_value(property_name, value);*

  stelt een parameter in en voegt deze toe aan de lijst indien nodig.

- *void get_property_value(property_name, value);*

  zoekt een parameter op in de lijst. De opgezocht waarde wordt toegekend aan “value”. Indien de parameter niet in de lijst voorkomt, wordt “value” leeggemaakt.

4.4.2 Opbouw van een weergavevenster

Het opbouwen van een weergavevenster vereist wel dat de programmeur weet hoe hij met gtkmm [16] om kan gaan. Voor meer informatie verwijzen we naar de website van gtkmm. Een weergavevenster moet afgeleid worden van de klasse AnalyserWindow. Deze klasse voorziet weer in de oplossing van enkele problemen die zich voordoen bij het werken met meerdere uitvoeringsaders.

Er volgt een korte beschrijving van de methoden die door de moduleprogrammeur voorzien moeten worden.

construcor	In de constructor moeten de methode “init_interface” en “sync_interface” aangeroepen worden. Ook moet een nieuwe instantie van de gepaste analyser in de variabele “data” gestoken worden.
destructor	In de destructor kan er eventueel voor gezorgd worden dat gealloceerd geheugen vrijgegeven wordt. De analyser in “data” wordt automatisch vrijgegeven.
description(), name(), category()	Deze 3 mehodes geven allen een tekenreeks als resultaat. Hierin staan respectievelijk de omschrijving van wat het venster doet; de naam van het venster en de categorie waarin het venster valt. Doorgaans wordt een analyser nooit los van een venster gerefereerd. Daarom zijn naam, omschrijving en categorie normaalgezien kenmerkend voor de analyser die toegepast wordt in het venster.
init_interface()	Deze methode zorgt voor het opbouwen van alle elementen in het venster. Vooraleer er begonnen wordt, moet in deze methode de gelijknamige methode van de ouderklasse aangeroepen worden. (Bij een afleiding rechtstreeks van AnalyserWindow gebeurt dit door het statement “AnalyserWindow::init_interface();”.) Daarna mogen alle statements komen die zorgen voor de opbouw van het venster. Als laatste moet het statement “show_all();” uitgevoerd worden.
sync_interface()	Omdat het mogelijk is om de instellingen van een analyser (venster) uit een bestand te lezen, moet er een manier zijn om de knoppen die de instellingen bepalen gelijk te stellen met de waarden die in het bestand gespecifieerd zijn. Na het laden van de 1 instellingen wordt automatisch deze methode aangeroepen. Ze moet dan alle instellingen aflopen en de betreffende besturingselementen op de juiste waarde zetten.
update()	De “update” methode wordt door het systeem aangeroepen nadat de analyser uitgevoerd is. Deze methode moet dan de externe buffer van de analyser in “data” lezen, en deze op het scherm tekenen.

4.4.3 Parameters in een weergavevenster

Eerder is al vermeld dat een analyser parameters kan hebben en dat er standaardmethoden voorzien zijn om deze parameters in te stellen. Via het weergavevenster is het doorgaans mogelijk om ook deze parameters in te stellen, en zo de weergave of de aard van de analyse aan te passen. Het kan ook zijn dat een parameter specifiek aan het weergavevenster, en niet aan de analyser is. Toch is het raadzaam om alle parameters via de analyser op te slaan, omdat het mogelijk is om deze lijst

met parameters van een bestand te lezen, en op te slaan. Het weergavevenster kan met de methoden “set_property_value” en “get_property_value” van de analyser in “data” eigenschappen instellen en lezen.

Anderzijds is het ook zo dat bij het wijzigen van de parameters van een analyser, de analyse automatisch opnieuw gestart wordt, en als gevolgd daarvan de “update” methode van het betreffende weergavevenster ook opnieuw aangeroepen wordt. Bij het wijzigen van parameters moet dan geen rekening gehouden worden met het opnieuw starten van de analyse. De programmeur moet enkel de statements voorzien om bij het wijzigen van een besturingselement, de juiste parameter op de juiste waarde instellen, en om bij het uitvoeren van de methode “sync_interface” de juiste besturingselementen op de juiste waarde te zetten.

4.5 Sessies

Tot nu toe kunnen we functionele delen (filters en analysers) gebruiken en instellen. We kunnen ook via de gebruikersinterface van snapshot een combinatie van deze delen maken om zo een complexere analyse en filtering uit te voeren. Om niet iedere keer bij het starten van het programma deze samenstelling te moeten doen, is de mogelijkheid voorzien om alle instellingen op te slaan. Dit noemen we een sessie. Deze worden in een xml[17] formaat opgeslagen. Voordeel hiervan is dat het bestand eenvoudig leesbaar is, en eventueel manueel een sessie geconfigureerd kan worden door dit bestand aan te passen.

Volgende gegevens worden in een sessie opgeslagen :

- bestandsnaam van de geopende database;
- bestandsnaam van de geopende film;
- de hoofdfilterketen, samen met de volgorde van alle filters en de parameters van iedere filter;
- voor ieder filter die interne filterketens heeft, de volgorde en de parameters van iedere filter in deze keten. Dit is een recursief proces;
- voor iedere analyser de positie en grootte van het venster, de parameters van de analyser en de instellingen voor de bijhorende filterketen.

Hoofdstuk 5

Overzicht van de gebruikte filters

5.1 Preprocessing

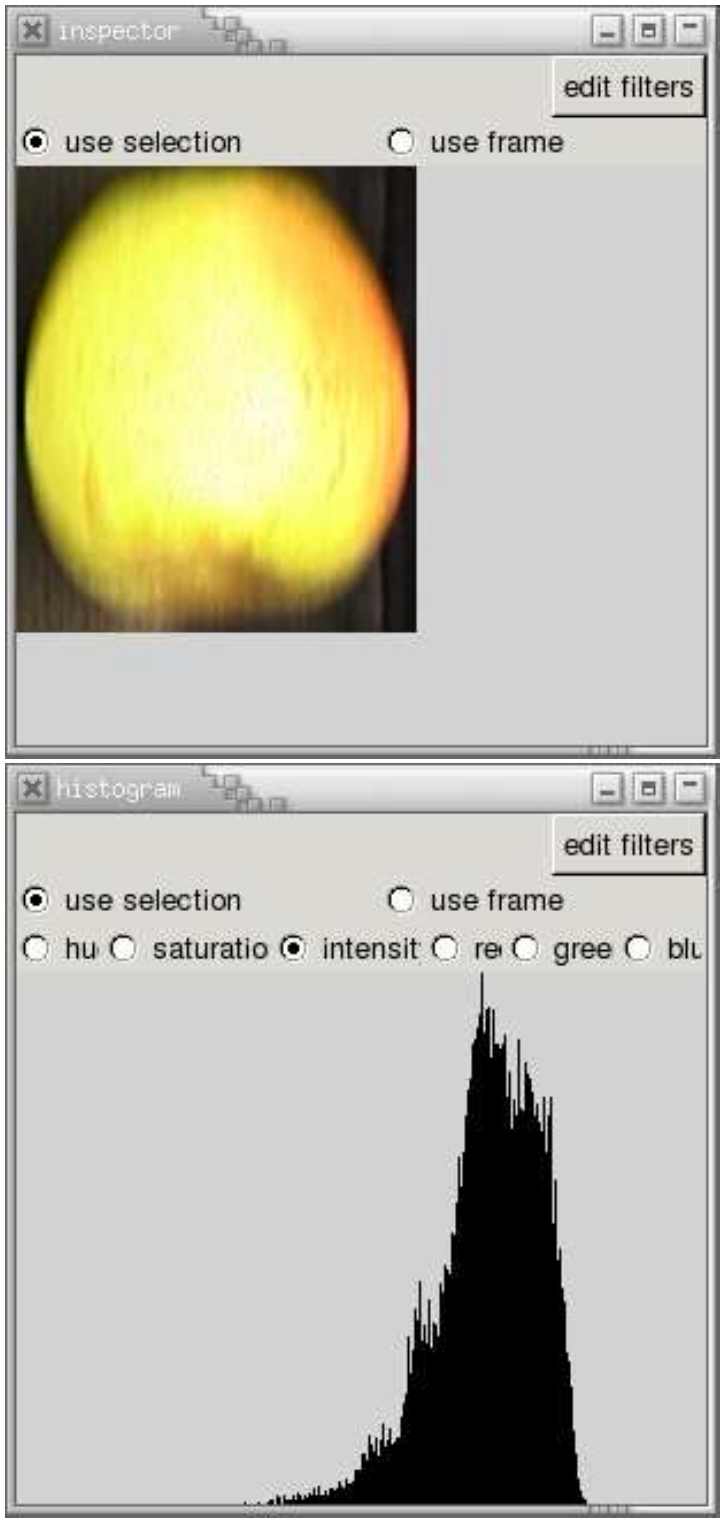
Preprocessing houdt o.a. in: herschalen, achtergrondinformatie uit het beeld halen, contrast en -belichtingsconditie normaliseren. Alle bewerkingen, die plaats vinden vóór de beeldanalyse, worden geacht te behoren tot de voorbereidingsfase. Het kan bijvoorbeeld zijn, dat de kwaliteit van de beeldinformatie minder goed is dan gewenst. Het beeld kan ruis bevatten door onvolkomenheden van de gebruikte sensoren. Het kan ook zijn dat het beeld onscherp is door toedoen van de camerafocus. Weer een ander defect kan optreden door beweging van het af te beelden object met een snelheid, die te groot is voor de gebruikte bemonsteringstijd. Verder kan de belichting slecht zijn of niet gelijkmatig verdeeld over het gehele beeldvlak. Een aantal filters die nu volgen kunnen gebruikt worden voor preprocessing. Voor contrast en -belichtingscondities te normaliseren is dit histogramequalisatie (zie 5.2). Om de achtergrondinformatie uit het beeld te halen kan maskering toegepast worden zoals beschreven in 5.3 en het herschalen kan door de fiter resize (zie 5.7).

5.2 Histogramequalisatie

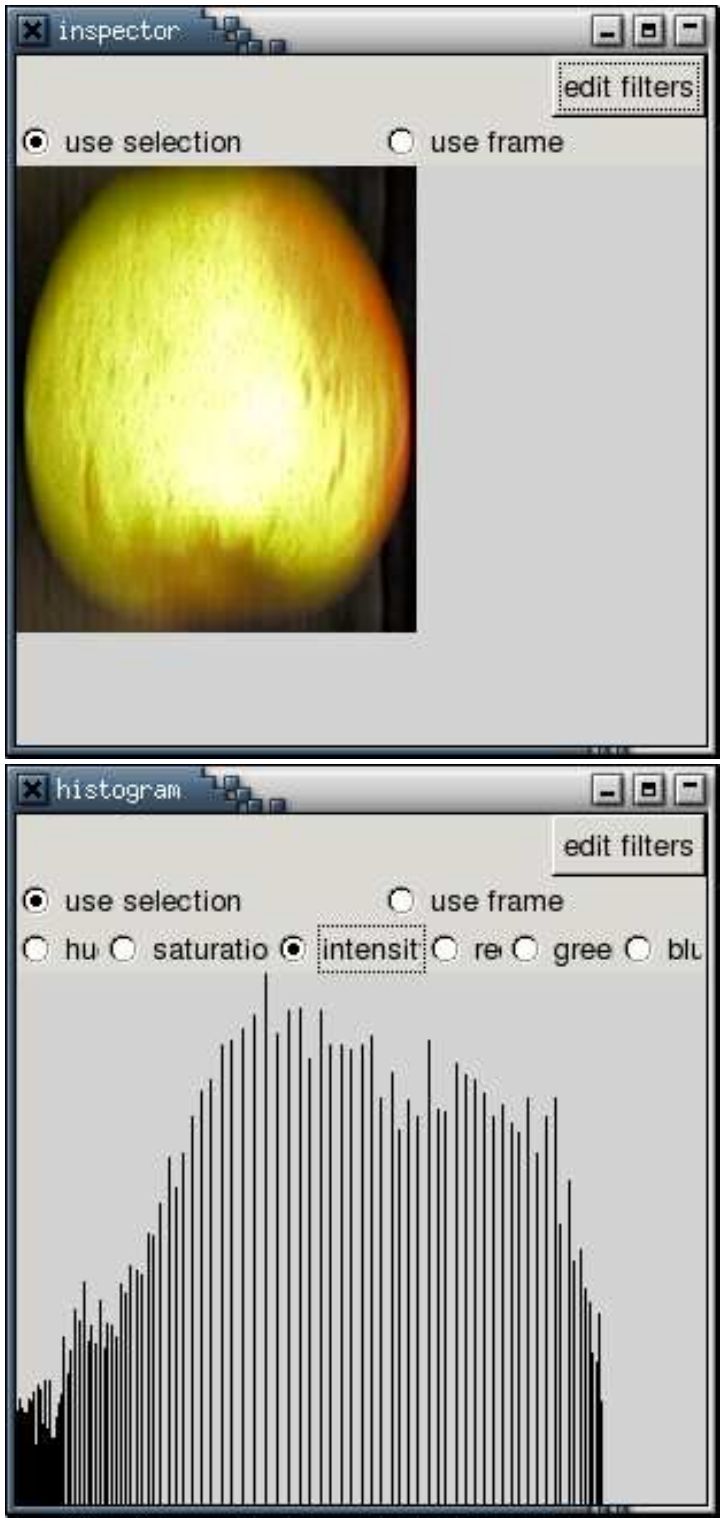
Histogramequalisatie is een techniek om het contrast in een afbeelding te vergroten waarbij het volledige bereik van intensiteitswaarden in een afbeelding gebruikt wordt.

Om bij een bitmap kleurenafbeelding histogramequalisatie toe te passen, maken we gebruik van de HSI-kleurruimte(hue, saturation, intensity). De huewaarde geeft de tint van de kleur (purper, geel, groen..) weer. Deze gaat van 0 tot 360. Hiervoor gebruiken we een 16-bits getal. Voor de satuatiewaarde en de intensiteitwaarde gebruiken we een 8-bits getal deze waarde gaat van 0 tot 255. De saturatie geeft de puurheid van de kleur weer, dit is de hoeveelheid wit licht die met de kleur gemengd is. De tint en de saturatie samen bepalen dus de kleurtoon. De intensiteit beschrijft de helderheid, deze is afhankelijk van de belichting. Deze intensiteitswaarde kan nu net zoals de grijswaarde gebruikt worden om de histogramequalisatie op toe te passen. Dit is enkel geldig omdat de intensiteitswaarden de kleur van de afbeelding niet beinvloeden.

In een afbeelding met veel contrast hebben we een histogram met een vlak verloop, in tegenstelling tot een afbeelding met weinig contrast is hierbij het histogram niet verdeeld over het volledige bereik van intensiteitswaarden. In figuur 5.1 zie je het histogram van een overbelichte appel. In het histogram zie je dit aan de intensiteitswaarden die allemaal redelijk groot zijn. Door hier nu histogramequalisatie op toe te passen kan het histogram afgevlakt worden en over het volledige intensiteitsbereik verspreid worden. Hierdoor kan een éénduidige voorstelling van alle appelbeelden



Figuur 5.1: Afbeeldig van een overbelichte appel met bijhorend histogram van de intensiteitswaarden voor equalisatie



Figuur 5.2: Afbeeldig van de appel uit figuur 5.1 toegepast op de intensiteitswaarden

bekomen worden. Dit kan nuttig zijn om verdere analyses op de beelden uit te voeren. Het tesultaat van deze bewerking op de appel in figuur 5.1 is weergegeven in figuur 5.2.

De waarschijnlijkheid dat een intensiteitwaarde r_k voorkomt in een beeld wordt benaderd door

$$P_r(r_k) = \frac{n_k}{n} \tag{5.1}$$

waarbij n het totaal aantal punten is en n_k het aantal punten in het beeld met intensiteitswaarde r_k voor k=0,1,2,...,255. De transformatie voor histogramequalisatie wordt gegeven door

$$\begin{aligned} S_k &= T(r_k) = \sum_{j=0}^{k} p_r(r_j) \\ &= \sum_{j=0}^{k} \frac{n_j}{n} \end{aligned} \tag{5.2}$$

Voor ieder punt in het beeld wordt de originele intensiteitswaarde vervangen door deze getransformeerde intensiteitswaarde.

5.2.1 Beschrijving van de uitwerking

De resultaten van de transformatie worden opgeslagen in een map. Deze heeft een unieke sleutel met bijhorende inhoud. Als sleutel gebruiken we de intensiteitswaarden(0-255) en als inhoud hoe dikwijls deze voorkomt in de afbeelding.

Dit wordt gedaan door alle punten van de afbeelding te overlopen. Hiervoor wordt er gebruik gemaakt van HSI-gegevens. Dit is een gegevensstructuur met enkelvoudige types voor hue1(int16[1]), saturation(int8[2]) en intensity(int8). Het beeld moet lijn per lijn overlopen worden en voor ieder punt moet er gekeken worden welke intensiteitswaarde deze heeft. Als deze nog niet opgeslagen is in de map dan wordt deze opgeslagen als unieke sleutel met als inhoud één. Wanneer deze waarde dan nog eens voorkomt in het beeld verhogen we de inhoud met één. Zo wordt er een map bekomen met als sleutel de intensiteitswaarden (r_k) en met als inhoud hoe dikwijls deze voorkomen. Deze map wordt dan nog gesorteerd op stijgende intensiteitswaarden.

Vervolgens wordt de inhoud van deze map cumulatief opgeteld en wordt iedere waarde gedeeld door het totaal aantal pixels(n). Dit geeft een map met als sleutel nog steeds de intensiteiswaarden die voorkomen in het beeld en als inhoud de genormaliseerde waarden van de cumulatieve som. Deze waarden liggen tussen 0 en 1 en moeten dus nog vermenigvuldigd worden met 255 om bruikbare transformatiewaarden te bekomen.

Om de transformatie toe te passen overlopen we het beeld opnieuw. We vervangen dan iedere intensiteitswaarde door zijn overeenkomstige getransformeerde waarde.

5.3 Maskering

Er bestaan uiteenlopende manieren om een afbeelding te bewerken en te analyseren. De bewerkingen gaan echter altijd punt per punt. Wat het programma in feite doet, is het verhogen of verlagen van de kleurwaarde van een punt volgens een bepaald principe. Bij een masker wordt aan bepaalde gebieden in het beeld een vaste kleur toegekend.

[1] 16 bit geheel getal
[2] 8 bit geheel getal

Omdat appels een ronde vorm hebben en we steeds werken met rechthoekige beelden, zal aan de uiterste randen steeds achtergrondinformatie aanwezig zijn. Vermits dit geen nuttige informatie is, is het beter deze informatie weg te filteren door een cirkelvormig masker over het beeld te leggen. Dit zorgt ervoor dat enkel de kleurwaarden binnen de cirkel onveranderd blijven en de punten die erbuiten liggen een vaste kleur krijgen.

5.3.1 Beschrijving van de uitwerking

We bepalen eerst de hoogte en de breedte van het frame. De kleinste waarde van deze twee wordt gebruikt als diameter van de cirkel. Zo ligt de cirkel steeds binnen het frame.

Het beeld wordt lijn per lijn overlopen en voor ieder punt wordt er nagegaan of dit binnen of buiten de cirkel valt. Het binnen of buiten de cirkel vallen van een punt wordt bepaald door de vergelijking van een cirkel. Nu kunnen via parameters een aantal dingen in gesteld worden:

- toepassen op HSI- of RGB-data;
- het beeld binnen of buiten het masker tonen;
- maskerkleur(alle kleuren);
- maskervorm(cirkel of ellips).

5.4 Drempelfilter

De drempelfilter bepaalt welke punten in het beeld tot de voorgrond behoren en welke tot de achtergrond behoren, door ze te vergelijken met een drempel. Ieder punt krijgt een vaste kleur toegewezen op basis van de klasse waartoe het behoort. Het is ook mogelijk om een derde “onzekere” klasse te maken, zodat punten die dicht bij de drempel liggen als onzeker gemarkeerd kunnen worden.

De drempel kan vast ingesteld worden, of kan berekend worden uit het toegeleverde beeld. De drempel kan ook als lopend gemiddelde van een aantal voorgaande drempelwaarden gekozen worden.

5.5 Split

Dit filter kan gebruikt worden om maar één enkel kleurkanaal over te houden. De waarden van dit kanaal worden zo omgevormd dat grijswaarde 0 overeenkomt met de laagste waarde van het kanaal (0 doorgaans), en dat grijswaarde 255 overeenkomt met de waarde in het kanaal (255 of 359).

Deze filter heeft één parameter “channel” waarmee het kanaal ingesteld kan worden dat men over wil houden.

5.6 Despeckle

Despeckle is een filter om ruis uit een binair beeld[3] of ternair beeld[4] beeld te verwijderen. Dit gebeurt door te onderzoeken hoeveel buren van dezelfde klasse ieder punt heeft. De gebruiker

[3] binair beeld : een beeld waarvan de punten maar twee verschillende waarden aan kunnen nemen.

[4] ternair beeld : een beeld waarvan de punten drie verschillende waarden kunnen hebben.

Tabel 5.1: Overzicht van de parameters van de *drempelfilter*

parameternaam	*omschrijving*
channel	het kleurkanaal waarop de drempel toegepast wordt (Rood, Groen, Blauw, Tint, Verzadiging of Intensiteit).
range	het aantal voorgaande beelden dat gebruikt word om de drempelwaarde te berekenen.
max_deviation	de maximale afwijking die de drempel van het huidige beeld mag hebben ten opzichte van de vorige drempel. Hiermee wordt voorkomen dat bijvoorbeeld volledig zwarte beelden mee in de drempel bepalen.
background_color	de kleur die aan de achtergrondpunten toegewezen wordt
foreground_color	de kleur die aan de voorgrondpunten toegewezen wordt
unsure_color	de kleur die aan de onzekere punten toegewezen wordt
fixed	"yes" of "no". Een vaststaande drempel of niet. Bij "no" wordt de drempel uit de beelden zelf berekend.
fixed_value	Bij een vast ingestelde drempel zijn kan hiermee de drempel ingesteld worden. Deze wordt opgegeven als percentage van de maximale waarde.
unsure_range	het percentage van de drempel dat als onzeker wordt genomen. Dit is zowel boven als onder de drempel, dus het onzeker bereik is het dubbele van de opgegeven waarde en ligt symmetrisch ten opzichte van de drempel. Het is ook mogelijk een vaste waarde op te geven door een getal zonder % teken in te voeren.

kan specifiëren hoeveel aanliggende punten minimaal van dezelfde klasse moeten zijn alvorens het beschouwde punt tot die klasse behoort. Dit getal kan gaan van één tot acht. Punten die door dit criterium niet tot voorgrond of onzeker behoren, worden automatisch naar de achtergrond verhuisd. Indien het aantal aanliggende punten groter moet zijn dan zeven, zal de filter automatisch alle punten naar de achtergrond verhuizen.

5.7 Resize

Het resize filter verkleint beelden tot een vaste afmeting, of tot een percentage van de originele grootte. Men kan voor zowel hoogte als breedte instellen hoeveel verkleind moet worden (in procent van de originele afmeting) of hoe groot de uiteindelijke afmeting moet zijn. Indien voor één van de parameters een 0 opgegeven wordt, wordt de parameter automatisch bepaald zodat de verhouding hoogte-breedte behouden blijft. Beelden vergroten is nog niet geïmplementeerd en als het beeld kleiner is dan de opgegeven afmetingen, of als er een percentage hoger dan 100% opgegeven wordt, zal de filter geen bewerkingen uitvoeren.

5.8 Rectangular_mask

Rectangular_mask past een rechthoekig masker toe op het beeld. Men kan dit masker opgeven door de coördinaten van de linkerbovenhoek op te geven (de nulpositie ligt hier in de rechterbovenhoek van het beeld). Men kan daarbij ook nog de breedte en de hoogte van de rechthoek opgeven.

Tabel 5.2: Overzicht van de parameters van de *despeckle-filter*

parameternaam	*omschrijving*
channel	Bepaalt welk kanaal in rekening genomen wordt voor de klasse te bepalen. Deze parameter kan rood, groen of blauw zijn.
foreground_color	De kleur die toegewezen wordt aan de voorgrond punten. Deze parameter bepaalt ook de minimale waarde van rood, groen of blauw voor de voorgrond klasse.
background_color	De kleur die toegewezen wordt aan de achterground punten. Ook hier geldt dat deze parameter de maximale waarde van rood, groen of blauw voor de achtergrond klasse bepaalt.
minimum_neighbours	Het kleinste aantal buren van eenzelfde klasse die een punt moet hebben om naar die klasse gepromoveerd te worden.

Tabel 5.3: Overzicht van de parameters van de *rectangular_mask filter*

parameternaam	*omschrijving*
left x, top y	De coördinaten van de linkerbovenhoek van het masker.
width, height	De hoogte en de breedte van het masker.
background_color	De kleur die aan de gemaskeerde punten toegewezen wordt.
keep	"inside" of "outside". Geeft aan of de binnenkant of de buitenkant van de rechthoek gemaskeerd moet worden. Indien de parameter op "inside" staat wil dit zeggen dat de buitenkant van de rechthoek gemaskeerd wordt. Om dan onnodig rekenwerk te vermijden wordt het beeld uitgeknipt tot een kleiner beeld.

5.9 Selectionfilter

Selectionfilter kan gebruikt worden om ongeldige selecties te verwijderen. Hierbij wordt dan vooral gedacht aan selecties die te klein zijn, of waarvan de hoogte-breedte verhouding zo is dat de selectie nooit geldig kan zijn. Ongeldige selecties worden uit de lijst met selecties verwijderd. Een criterium kan uitgeschakeld worden door -1 als waarde op te geven.

5.10 Growselections

Deze filter kan gebruikt worden om alle aanwezige selecties met een vast bedrag te vergroten. Dit kan apart ingesteld worden voor verticale ("grow-y") als horizontale ("grow-x") afmetingen. Negatieve waarden maken de selecties kleiner en bij het ingeven van "0" wordt er geen wijziging doorgevoerd.

Tabel 5.4: Overzicht van de parameters van *selectionfilter*

parameternaam	*omschrijving*
width-max, width-min	De minimale en de maximale breedte voor een selectie.
height-max, height-min	De minimale en de maximale hoogte.
ratio-min, ratio-max	De minimale en de maximale hoogte-breedte verhouding. Deze verhouding wordt altijd zo berekend dat ze het kleinst is (dus kleinste afmeting delen door grootste afmeting).

5.11 Componentscan

Dit filter past een "verbonden componenten" (connected components [18]) algoritme toe op een binair of ternair beeld.

5.11.1 Onderscheid maken tussen verschillende objecten

Om onderscheid te maken tussen verschillende componenten is een algoritme nodig dat aan iedere afzonderlijke component een uniek label toekent. Daar we het toekennen van labels in een digitaal systeem implementeren is het voldoende om gehele getallen als labels te gebruiken. Om eenvoudig te kunnen werken, converteren we ieder punt van het beeld dat opgeslagen is als drie bytes, naar een vier byte getal. Als de meest beduidende byte van een punt "0" is, wil dit zeggen dat het punt nog niet verwerkt is. Bij het toekennen van de labels we ervoor dat de meest beduidende byte een waarde verschillend van "0" heeft.

Verbonden componenten

Het algoritme dat we hiervoor gebruiken onderzoekt de relatie van iedere punten ten opzicht van zijn omgeving. Uit deze relatie worden alle punten in groepen ingedeeld. Groepen kunnen daarna op hun beurt weer samengenomen worden tot componenten of objecten. Een criterium dat gebruikt kan worden is dat alle punten tot dezelfde klasse behoren ("voorgrond" of "onzeker" in ons geval) en dat ze op een bepaalde manier met elkaar verbonden zijn.

Dit is een werkwijze die in veel geautomatiseerde systemen toegepast wordt. Er zijn verschillende soorten van relaties tussen punten onderling, maar de meest gebruikte zijn :

4-connectivity Deze relatie houdt enkel rekening met de punten die links, rechts, boven of onder van het beschouwde punt liggen. Stel dat we een verzameling punten $p(x,y)$ hebben waarbij ieder punt gekenmerkt wordt door zijn coördinaten x en y, dan kunnen we de omgeving van dat punt uitdrukken als :

$$N_4 = \{p(x+1,y), p(x-1,y), p(x,y+1), p(x,y-1)\} \tag{5.3}$$

8-connectivity Deze relatie houdt rekening met alle punten die raken aan het beschouwde punt, dus zowel horizontaal, verticaal als diagonaal.

$$\begin{aligned} N_8 = \{&p(x-1,y-1), p(x,y-1), p(x+1,y-1), p(x-1,y), \\ &p(x+1,y), p(x-1,y+1), p(x,y+1), p(x+1,y+1)\} \end{aligned} \tag{5.4}$$

Identificatie van verbonden componenten

Op basis van de relaties gedefinieerd in 5.11.1, kunnen we in een beeld zoeken naar objecten. Om het algoritme simpel te houden, en om mogelijk het algoritme later in seriële hardware te kunnen implementeren beschouwen we enkel de punten die reeds verwerkt zijn. Dus de punten met coordinaten $x+1$ of $y+1$ worden niet beschouwd. De relatie van het beschouwde punt tot deze punten wordt bepaald als we later de relatie van die punten tot hun eigen omgeving onderzoeken.

Verder is de afspraak dat punten die buiten het beeld vallen automatisch tot de achtergrond behoren. We kunnen dan volgend stappenplan volgen om de relatie te onderzoeken :

1. Behoort $p(x,y)$ tot de voorgrond ? Zo nee, ga door naar het volgende punt.

2. Indien alle punten in de omgeving tot de achtergrond behoren, kennen we een nog niet gebruikt label toe aan het beschouwde punt. Daarna gaan we door naar het volgende punt.

3. Indien alle punten in de omgeving, die tot de voorgrond behoren en reeds gelabeld zijn, hetzelfde label hebben, kennen we dit label toe aan het beschouwde punt en gaan we door naar het volgende punt;

4. Indien er meerdere (niet identieke) labels in de omgeving zijn, kennen we één van deze labels toe aan het beschouwde punt, en onthouden we dat de verschillende labels bij elkaar horen (equivalent zijn).

Na het voltooien van deze actie voor ieder punt in het beeld, hebben we een nieuw beeld gecreëerd waarin alle punten een specifiek label gekregen hebben. Ook hebben we een lijst van identieke labels. Dit is een eenvoudige lijst van koppels. Labels die samen in een koppel staan zijn equivalent.

Oplossen van de equivalenties

Een volgende stap is het oplossen van de lijst met koppels. We zouden tot een datastructuur moeten komen waarin we voor ieder label een representatief label kunnen opzoeken. Op die manier kunnen we uiteindelijk ieder label vervangen door zijn representatief label[5] en heeft iedere component een uniek label gekregen.

Om deze equivalenties op te lossen definiëren we een nieuwe lijst met koppels van labels. Bij deze lijst hoort een zoekfunctie $Z(L)$. Deze functie neemt als parameter een label, en geeft als resultaat een ander label. De functie is zo gedefiniëerd dat het label dat als parameter gebruikt wordt, en het label dat als resultaat gegeven wordt, equivalent zijn. De bedoeling van deze lijst is een boomstructuur op te bouwen. Het tweede lid in een koppel verwijst altijd naar de bovenliggende vertakking in de structuur. De wortel van deze boomstructuur is het representatief label voor alle andere labels in de boom. Om dit goed uit te kunnen voeren moet we 2 speciale labels definieren :

- Om aan te duiden dat een label niet in de lijst voorkomt, geeft de zoekfunctie $\emptyset$ als resultaat.
- Het opzoeken van een representatief label met de functie geeft als reslutaat ∞ terug.

De definitie van een label wordt dan uitgebreid tot :

$$L \in \{\emptyset, \infty\} \cup \Re \tag{5.5}$$

Voor het oplossen van de equivalenties vertrekken we vanuit de lijst met koppels die in 5.11.1 opgebouwd werd. Deze lijst wordt gesorteerd en dubbele koppels worden eruit verwijderd. We vertrekken vanuit een lege zoekboom en voeren voor ieder koppel de volgende stappen uit :

1. Zoek voor ieder label in het koppel (L_1, L_2), het geassocieerde label op in de zoekboom

$$A_1 = Z(L_1) \tag{5.6}$$
$$A_2 = Z(L_2) \tag{5.7}$$

   Nu zijn er 6 mogelijkheden :

[5] representatief label : in 5.11.1 werd reeds aangegeven dat sommige label equivalent kunnen zijn. Daarom duiden we in iedere set van equivalente labels één label aan dat de gehele set voorstelt. We noemen dit label "representatief label". Door bij een tweede stap van het scannen, ieder label te vervangen door het representatief label van de set, bekomen we dat de punten van iedere afzonderlijk component eenzelfde label hebben.

$A_1 = A_2 = \emptyset$: Geen van beide labels komt in de lijst voor. We kiezen L_1 als represenatatief label en voegen dit toe in de zoekboom zodat $Z(L_1) = \infty$. Ook L_2 voegen we toe zo dat $Z(L_2) = L_1$. *(uiteraard is het ook mogelijk om L_2 als represenatief label te kiezen)*

$A_1 = \emptyset$ en $A_2 \in \Re$: A_1 komt nog niet voor in de lijst, A_2 wel. We zoeken het representatief label L_∞ dat bij A_2 hoort en voegen A_1 aan de zoekboom toe, zo dat $Z(L_1) = L_\infty$.

$A_1 = \emptyset$ en $A_2 = \infty$: In principe is dit hetzelfde geval als het voorgaande puntje. Alleen weten we dat A_2 een representatief label is. We voegen L_1 toe aan de zoekboom zodat $Z(L_1) = L_2$.

$A_1 \in \Re$ en $A_2 \in \Re$: Hier weten we dat beide labels reeds in de boom voorkomen. We zoeken voor zowel A_1 als A_2 de representatieve labels ($L_{\infty 1}$,$L_{\infty 2}$) en passen de boom zo aan dat $Z(L_{\infty 1}) = L_{\infty 2}$.

$A_1 \in \Re$ en $A_2 = \infty$: Principieel het zelfde geval als voorheen, alleen moeten we maar 1 representatief label meer zoeken.

$A_1 = A_2 = \infty$: En deze optie maakt het nog eenvoudiger omdat beide labels represenatief zijn. We passen de zoekboom aan zodat $Z(L_1) = L_2$.

Als tweede stap verwerken we de lijst met toegekende labels. We controleren of al deze labels in de zoekboom voorkomen. Labels die nog niet in de zoekboom voorkomen hebben geen equivalenties en zijn dus automatisch representatief. Deze voegen we dus ook zo aan de zoekboom toe. Als deze stap voltooid is, hebben we een boomstructuur waarmee we voor iedere label een representatief label op kunnen zoeken. Het enige nadeel is dat we voor sommige labels meer dan één zoekbewerkingen moeten uitvoeren wat grote vertragingen kan opleveren bij grote beelden. Daarom doorlopen we de zoekboom nog een laatste keer en passen we deze zo aan dat we voor ieder label na éé zoekbewerking een representatief label vinden.

Als laatste stap overlopen we het beeld opnieuw en vervangen we ieder label door zijn representatief label.

5.11.2 Bepalen van de afmetingen van ieder object

De uiteindelijke bedoeling van dit filter is dat voor ieder afzonderlijk object een selectie gegeneerd wordt. Daarom moeten we voor ieder object nog de afmetingen en de positie bepalen. Hieruit kunnen we dan de nodige selecties afleiden. Voor ieder punt $p(x,y)$ in het beeld doen we het volgende :

1. Behoort het punt tot de achtergrond ? Zo ja, ga door naar het volgende punt.
2. Zoek de set van coordinaten $\{(x_{links}, y_{boven}), (x_{rechts}, y_{onder})\}$ die bij het label van het punt hoort.
3. Als $x < x_{links}$ stel $x_{links} = x$.
4. Als $x > x_{rechts}$ stel $x_{rechts} = x$.
5. Als $y < y_{boven}$ stel $y_{boven} = y$.
6. Als $y > y_{onder}$ stel $y_{onder} = y$.

Nu hebben we voor ieder label dat in de figuur voorkomt een koppel coördinaten. Deze coördinaten zijn de linkerbovenhoek en de rechteronderhoek van de selectie.

5.12 Selectionscan

Dit filter is specifiek ontwikkeld voor het zoeken van appels in beelden van de transportband in werking. Er worden intern drie filterketens gebruikt. Twee van die filterketens bevatten telkens drie filters. Deze zijn een drempelfilter, een despeckle filter en de componentscan filter. Van een binnenkomend beeld worden twee kopies gemaakt. Deze kopies worden dan door de twee componentscan filterketens gestuurd. Daarna wordt één beeld nog door derde keten gestuurd die ingesteld kan worden om de slechte selecties te verwijderen.

Het verschil tussen de twee componentscan filterketens is de drempel die toegepast wordt. Het is voldoende om één drempel toe te passen, en zo de verbonden componenten te vinden. Echter hier krijgen we problemen met appels die tegen elkaar liggen. Deze worden als één appel beschouwd. Daarom passen we een tweede, veel hogere drempel toe. Omdat de grens tussen twee appels altijd donkerder is (omdat de invalshoek van het licht op deze plaats kleiner is, is er minder reflectie) zullen aanliggende appels bij een hogere drempel doorgaans wel van elkaar onderscheiden worden. Het probleem bij een hogere drempel is dat de zijkanten van de appels niet in de selectie opgenomen worden, en we dus een deel van de informatie verliezen. Daarom maken we een combinatie van beide.

De selecties bij de lagere drempel zullen de selecties bij een hogere drempel altijd omvatten, omdat bij een hogere drempel de randen van een object niet tot de voorgrond behoren en het object dus kleiner geïdentificeerd wordt. Als we nu de selecties die voortkomen van de hoge drempel, proportioneel uitbreiden tot ze aan de omliggende selectie bij lage drempel raken, en enkel deze selecties gebruiken, bekomen we een algoritme dat redelijk goed onderscheid kan maken tussen verschillende appels op de sorteerband.

Deze werkwijze is enkel geschikt voor gebruik op de transportband, omdat we daar een redelijk uniforme achtergrond hebben. Op basis van de selecties die reeds manueel gemaakt zijn op de boomgaard, moet het ook mogelijk zijn om met andere technieken[21] appels te detecteren. Bij één van die methodes wordt de drempel niet berekend op basis van de gemiddelde waarde, maar op basis van het histogram. Ook het toepassen van de drempel gebeurt op basis van histogramgegevens. Deze methode kan gebruikt worden door een nieuwe filter te schrijven, die de histogram drempel methode toepast. De huidige “Threshold” filter moet dan vervangen worden door de nieuwe filter. Het enige programmeerwerkt dat hiervoor nodig is, is het schrijven van de nieuwe drempelfilter. Het vervangen van de filter kan gebeuren in het sessiebestand dat we eerder gebruikt hebben voor appeldetectie.

Hoofdstuk 6

Overzicht van de gebruikte analysers

6.1 Histogram Verwerking

Een belangrijk aspect bij beeldverwerking is de verdeling van de kleur over de pixels. Deze verdeling kunnen we bekijken met een histogram. Zo'n histogram kan gemaakt worden voor de drie basiskleuren rood,groen en blauw. Hierbij noemt men kleurwaarden die iedere kleur kan aannemen de kleurdiepte, dit zijn waardes tussen 0 en 255. Het histogram laat nu per kleur de verschillende kleurdieptes op de x-as zien (van zwart*(0)* tot volledig verzadigde kleur *(255)*, fel rood bijvoorbeeld), terwijl op de y-as het aantal pixels (per kleurdiepte) staat vermeld. Bij het histogram kan ook nog aangegeven worden wat de gemiddelde kleurdiepte de mediaan, en de modus is.

- het gemiddelde is de som van alle waarden gedeeld door het aantal pixels;
- de mediaan is de middelste kleurwaarde;
- de modus is de meest voorkomende kleurwaarde.

Het histogram van een digitale afbeelding is een discrete functie $h(r_k) = n_k$

- r_k is één van de gijswaarden k (k =0,1,...,255);
- n_k geeft aan hoeveel pixels in het beeld de intensiteitswaarde r_k hebben.

Het is meest gebruikelijk om het histogram te normaliseren door iedere waarde te delen door het totaal aantal punten(n) in het beeld. Het genormaliseerd histogram wordt dus $p(r_k) = n_k/n$. Dit geeft de waarschijnlijkheid weer dat een bepaalde kleurdiepte voorkomt. De som van alle componenten van het genormaliseerd histogram moet dus gelijk zijn aan één. Histogrammen zijn de basis voor verschillende verwerkingen van het beeld in het ruimtelijk domein. Een bijkomend voordeel van een histogram is dat het eenvoudig in software uitgerekend kan worden en zich ook goed leent tot hardware-implementatie. Dit maakt histogrammen interessant voor real-time beeldbewerking.

6.2 Beschrijving van de analysers

6.2.1 Standaard inspector

De inspector biedt de mogelijkheid om te zien op welk deel van het beeld de analyse uitgevoerd wordt. Hiervoor wordt, zoals bij iedere analyser, een apart venster geopen. Dit laat ook zien welke

filters al op het beeld uitgevoerd zijn voor de analyse. Het is dus eigenlijk de preprocessing die zichtbaar gemaakt wordt. In de bovenste afbeelding van figuur 5.1 van vorig hoofdstuk zie je de standaard inspector.

6.2.2 Standaard histogram

Voor ieder kanaal rood, groen en blauw voor RGB-data en hue, saturation en intensity voor HSI-data kan een histogram opgesteld worden. In het programma kan het kanaal ingesteld worden. Voor ieder frame of selectie kan dus het bijhorende histogram zichtbaar gemaakt worden. In de onderste afbeelding van figuur 5.1 van vorig hoofdstuk zie je het standaard histogram.

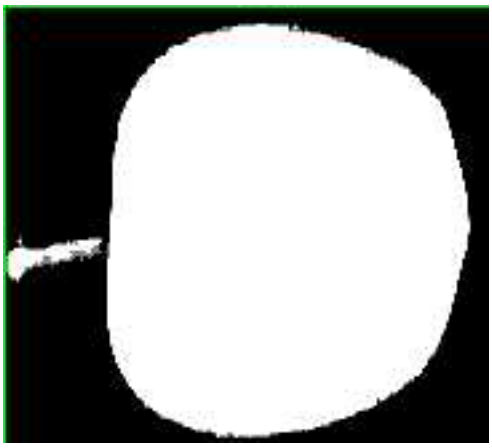
6.2.3 2D Histogram

Iedere kleur rood,groen en blauw hebben een kleurdiepte van 256, deze kunnen dus ieder een waarde hebben tussen 0 en 255. Als we nu rood en groen combineren zijn er $256 * 256$ mogelijkheden die kunnen voorkomen. Dit kan uitgezet worden in een 2D histogram. Dit geeft voor iedere mogelijke combinatie van kleurdiepten de frequentie van voorkomen in het beeld weer. Om dit in een 2D histogram te kunnen voorstellen, wordt de intensiteit van de getekende punten veranderd afhankelijk van de frequentie van voorkomen. Een combinatie die veel voorkomt in het beeld wordt weergegeven door een punt met hoge intensiteit en een combinatie die weinig voorkomt met een punt met lagere intensiteit. Er kan nu voor iedere kleurcombinatie rood,blauw en groen een 2D histogram opgesteld worden.

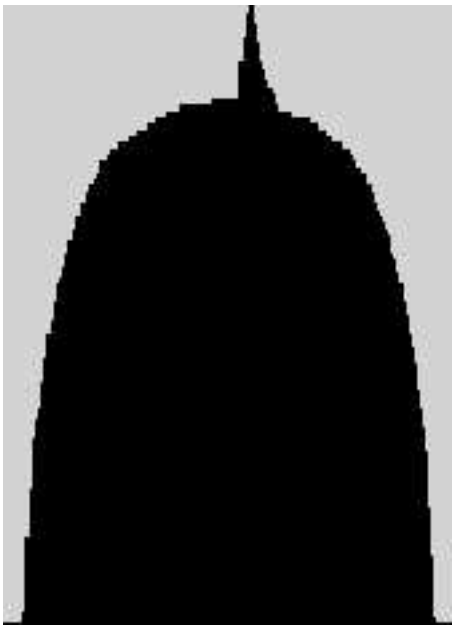
Deze analyser heeft diverse instellingen die aangepast kunnen worden om criteria te stellen aan de onderzochte data. 2 opties die we gebruikt hebben om histogramgegevens te verzamelen van appelen zijn “autoadd” en “useall” De optie “useall” zorgt ervoor dat bij het inladen van het beeld, de histogrammen van alle selecties gecombineerd worden tot één histogram. Dit histogram wordt vervolgens weergegeven. Door gebruik te maken van de optie “autoadd” worden de gegevens van alle reeds ingeladen histogrammen bijgehouden. Hierdoor kunnen we een gemiddeld histogram bekomen van alle selecties in één film. De gegevens van een histogram kunnen opgeslagen worden in een bestand, en naderhand kunnen deze eventueel terug ingeladen worden. Ook is het mogelijk om een verschilhistogram weer te geven. Hieruit kan dan voor ieder histogram de afwijking van het berekende histogram ten opzichte van het ingeladen histogram berekend worden.

6.2.4 Vormhistogram

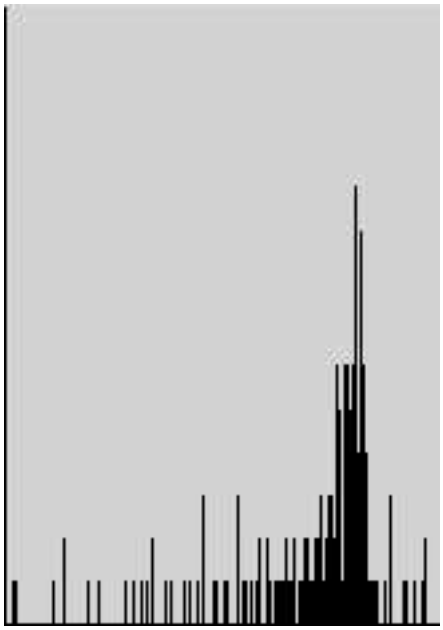
Voordat het vormhistogram toegepast kan worden moet de drempel (threshold) filter op het beeld toegepast worden. Hierdoor wordt een binair zwart-wit beeld bekomen. Dit is enkel mogelijk voor de appels die op de transportband liggen. Hierbij zorgt de drempel ervoor dat alle pixels van de appel een witte kleur krijgen. Als er nu een appel in het beeld geselecteerd wordt kunnen we hiervoor het vormhistogram berekenen (figuur 6.1). Voor iedere rij in het beeld wordt bijgehouden hoeveel witte pixels deze bevat, wat uiteindelijk een meetresulaat geeft dat de vorm van de appel in horizontale richting kenmerkt (figuur 6.2). Om nu het histogram hiervan op te stellen wordt voor ieder aantal witte pixels dat voorkomt *(lengte witte lijnen horizontaal)*, bijhouden hoe dikwijls deze voorkomen, het resultaat hiervan is een vormhistogram, zoals afgebeeld in figuur 6.3. Dit kan ook gedaan worden voor de kolommen in het beeld. Voor iedere kolom wordt dan weer bijgehouden hoeveel witte pixels deze bevat, wat allereest weer een vormbeschrijving geeft zoals in figuur 6.4. Het histogram hiervoor wordt op dezelfde manier bekomen als voor de horizontale richting. Dit laatste is afgebeeld in figuur 6.5.



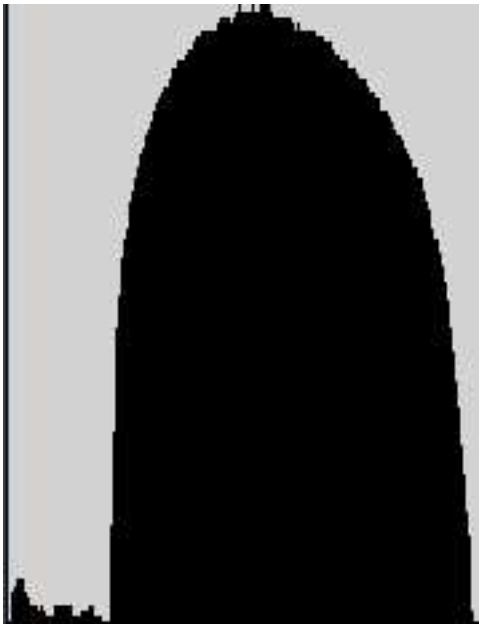
Figuur 6.1: Selectie van een appel waarop de threshold filter toegepast is



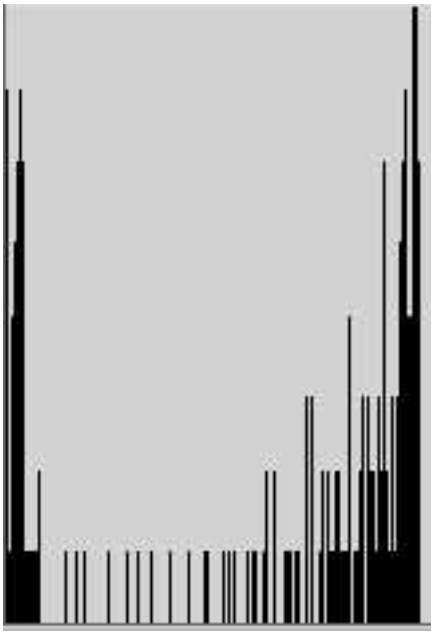
Figuur 6.2: Weergave van de vorm van de appel uit figuur 6.1 in horizontale richting



Figuur 6.3: Het vormhistogram van de appel uit figuur 6.1 in horizontale richting



Figuur 6.4: Weergave van de vorm van de appel uit figuur 6.1 in verticale richting



Figuur 6.5: Het vormhistogram van de appel uit figuur 6.1 in verticale richting

Hoofdstuk 7

Principale componenten in de kleurendistributies van appels

In dit hoofdstuk zullen we, op basis van een ruime statistiek van de pixels van de appels, de distributies van deze pixels in de kleurenruimte gaan bepalen. Hiervoor gaan we gebruik maken van de methodologie van de Principale Componenten Analyse [3, 4]. Verschillende benaderingen van deze kleurenruimte zijn mogelijk. In dit hoofdstuk en in het volgende hoofdstuk zullen we meestal de drie dimensionele RGB kleurenruimte bepalen. We zullen hiervoor eerst de relevante statistische grootheden dienen te bepalen. In eerste instantie zouden deze gegevens kunnen gecollecteerd worden aan de hand van een drie-dimensioneel histogram van de RGB-ruimte. Omdat een drie-dimensioneel histogram te sparce van natuur zou zijn en bovendien erg moeilijk te visualiseren is, maken we in deze analyse gebruik van de drie overeenkomende twee-dimensionele histograms. Deze drie overeenkomende twee-dimensionele histograms laten toe van dezelfde resultaten te bekomen als een volledige drie-dimensionele Principale Componenten Analyse.

Voorbeelden van de in dit eindwerk gebruikte twee-dimensionele histograms zijn terug te vinden in fig. 8.10. Ook in de literatuur zijn dergelijke histograms van appelbeelden terug te vinden, bijvoorbeeld bij Leemans *et al.*, zonder echter een volledige Principale Componenten Analyse uit te voeren [19].

7.1 Statistisch grootheden bepalen

Voor het 2D histogram kunnen de varianties, covariaties en correlatiecoëfficienten berekend worden, die we verder zullen gebruiken voor de PCA. Indien we slechts 2 dimensies hebben kunnen we met de variantie en covariatie een regressielijn bepalen. Voor meerdere dimensies is om hetzelfde te bekomen een PCA nodig. We bespreken hier daarom eerst het bepalen van een regressielijn in twee dimensies.

7.1.1 Variatie

De variantie is de gemiddelde kwadratische afwijking ten opzichte van het gemiddelde. Voor de berekening van variantie voor oorspronkelijke, ongegroepeerde gegevens in een datamatrix ga je als volgt te werk:

- bereken het gemiddelde van de variabele;

- trek iedere waarde van het gemiddelde af;
- kwadrateer de verschillen *(dit gebeurt omdat anders positieve verschillen tegen negatieve zouden wegvallen, het totaal van de afwijkingen zou dan nul zijn!)*;
- tel de gekwadrateerde verschillen op;
- variantie: deel het totaal van de gekwadrateerde verschillen door het aantal waarnemingen.

$$Var(rood) = \frac{1}{n-1}\sum_{i=0}^{n}(r_i-\overline{r})^2 \tag{7.1}$$

7.1.2 Covariantie

De covariantie is een maat voor de spreiding van twee gekoppelde variabelen. De covariantie van rood en groen wordt aangeduid met $Cov(rood,groen)$. Als $Cov(rood,groen)$ een positief getal is, dan is er sprake van positieve correlatie en als $Cov(rood,groen)$ een negatief getal is, dan is er sprake van negatieve correlatie. Als er geen sprake is van enige correlatie, dan geldt Cov(rood, groen) = 0. De berekening van de covariantie gaat dan als volgt:

- bereken de gemiddelde roodwaarde $\overline{r}$ en de gemidelde groenwaarde $\overline{g}$;
- bereken voor iedere roodwaarde r_i de deviatie $dr_i = r_i - \overline{r}$ en bereken voor iedere groenwaarde g_i de deviatie $dg_i = g_i - \overline{g}$;
- bereken de produkten van de deviaties, dus $(r_i-\overline{r})(g_i-\overline{g})$;
- bereken het gemiddelde van die produkten.

Dus:

$$Cov(rood,groen) = \frac{1}{n-1}\sum_{i=0}^{n}(r_i-\overline{r})(g_i-\overline{g}) \tag{7.2}$$

7.1.3 Correlatiecoëfficient

De correlatiecoëfficiënt is een getal dat de mate van correlatie tussen twee grootheden of variabelen aangeeft. Dit getal wordt aangeduid met de letter R en ligt tussen -1 en $+1$. In de grensgevallen $R=-1$ en $R=+1$ is er sprake van volledige correlatie. Bij $R=-1$ is dat volledige negatieve correlatie en bij $R=+1$ volledige positieve correlatie. Als er geen sprake is van enige correlatie, dan geldt $R=0$.

7.1.4 Berekening van de correlatiecoëfficiënt

Stel dat het verband tussen twee grootheden roodwaarden en groenwaarden wordt onderzocht en dat er n punten $(rood_i, groen_i)$ gegeven zijn. Voor de correlatiecoëfficiënt geldt dan:

$$R(rood,groen) = \frac{Cov(rood,groen)}{\sqrt{Var(rood)*Var(groen)}} \tag{7.3}$$

Met $Var(rood)$ en $Var(groen)$ de variatie, deze geven aan hoe sterk de rood- of groenwaarden van het gemiddelde afwijken.

7.1.5 Regressielijn

Bij het 2D histogram is het onmogelijk om een lijn te vinden die door alle punten gaat. Omdat de punten wel *ongeveer* op een lijn liggen, kan het interessant zijn om de lijn te zoeken die *zo goed mogelijk* bij het patroon van de punten past. Deze lijn noemen we de regressielijn of het regressiemodel.

Voordat we deze regressielijn kunnen gaan bepalen, moeten we weten wat we eigenlijk verstaan onder een *zo goed mogelijke* lijn. In de grafiek kan je zien dat de lijn door vrijwel geen van de meetpunten gaat. Er treedt dus meestal een afwijking op tussen de y die we hebben gemeten en de y die we verkrijgen uit ons regressiemodel. Het lijkt ons redelijk de lijn waarvoor de som van al deze (gekwadrateerde) afwijkingen het kleinst is, als de *beste* lijn te beschouwen.

Het bepalen van de regressielijn

Hier volgt de methode om de regressielijn $rood = a * groen + b$ te vinden volgens de kleinste-kwadratenmethode:

- bepaal $\overline{r}$ (het gemiddelde van alle ingestelde rood-waarden);
- bepaal $\overline{g}$ (het gemiddelde van alle gemeten groen-waarden);
- bepaal van alle rood-waarden het gekwadrateerde verschil met $\overline{r}$ tel ze op en deel door $n-1$ (het resultaat noemen we $Var(rood)$;
- bepaal van alle $groen - \overline{g}$ het produkt met de bijbehorende $rood - \overline{r}$ tel ze op en deel door $n-1$;
- bepaal $Var(rood)/Cov(rood, groen)$ (dit is a, de richtingscoëfficiënt van de regressielijn);
- bepaal $\overline{g} - a * \overline{r}$ (dit is b, het startgetal van de regressielijn).

Met deze regressielijn kan nu een transformatie van de toevalsveranderlijken (frequentie van voorkomen) doorgevoerd worden, zodanig dat deze een maximale variatie hebben en ongecorreleerd zijn. Dit noemt men de principale componenten.

Geometrisch komt dit neer op een assenstelselrotatie zodanig dat de nieuwe assen, geassosieerd met de principale componenten, wijzen in de richting van de grootste variatie. De eis dat de principale componenten ongecorreleerd moeten zijn, verplicht de keuze van een orthogonaal assenstelsel. Als nu een groot deel van de variatie (80 á 90 %) aan de eerste n principale componenten kan worden toegeschreven, dan kunnen we met slechts een weinig verlies aan informatie, de originele toevalsveranderlijken vervangen. Een principale componentenanalyse laat ons dus toe de dimensies te reduceren en dit met een controleerbaar verlies aan informatie. Hieruit blijkt het nut voor onze toepassing. Jammer genoeg geldt deze eenvoudige methode enkel voor twee gecorreleerde variabelen. In onze toepassing hebben we 3 variabelen. In sectie 7.2 wordt beschreven hoe we de principale componenten kunnen bepalen in onze toepassing.

7.2 Praktische uitwerking van de bepaling

Om de principale componenten te bepalen hebben we gebruik gemaakt van *GNU octave* [22]. Dit is een softwarepakket dat speciaal ontwikkeld is om als hulpmiddel te dienen bij het oplossen van wiskundige problemen. Eventueel zouden we Matlab hiervoor gebruikt kunnen hebben, maar omdat we er tot hiertoe al in geslaagd zijn om voor als ons werk *Vrije Software* [11] te gebruiken, proberen we deze trend voort te zetten.

7.2.1 Hulpfuncties

Om de principale compenentenbepaling flexibel uit te kunnen voeren hebben we een aantal functies opgezocht en gedefinieerd die als hulp kunnen dienen bij het uitwerken van het algoritme.

Inladen van een histogram

Om de histogrammen die eerder uit de database geëxtraheerd zijn, te gebruiken, moeten deze eerst en vooral geïmporteerd kunnen worden in octave. Hiervoor gebruiken we de functie *aload*. Met deze functie is het mogelijk om een matrix uit een bestand met getallen in te lezen. De histogrammen zijn opgeslagen in xml-formaat, maar het is perfect mogelijk om ze om te zetten naar een formaat met enkel opgesomde waarden. Dit kan gebeuren door alle lijnen waarin de tekens "<" of ">" voorkomen uit het bestand weg te laten. Deze bestanden kunnen we dan inlezen in een matrix door: *M = aload("bestandnaam","x resolutie","y resolutie");*

Gemiddelde waarden bepalen

Omdat er gewerkt wordt met histogrammen, komt het bepalen van het gemiddelde van de dataset overeen met het bepalen van het zwaartepunt van het histogram. We moeten wel rekening houden dat de waarde van een punt in het histogram het gewicht van dat punt voorstelt. Als de waarde van het punt in het histogram bepaald kan worden door de functie $V(x,y)$ kunnen we de zwaartepunten als volgt bepalen :

$$\overline{X} = \frac{1}{N}\sum_{x=1}^{x_{max}} x\{\sum_{y=1}^{y_{max}} V(x,y)\} \tag{7.4}$$

$$\overline{Y} = \frac{1}{N}\sum_{y=1}^{y_{max}} y\{\sum_{x=1}^{x_{max}} V(x,y)\} \tag{7.5}$$

Deze berekening kunnen we uitvoeren met de functie *mean*. Deze functie geeft zowel het zwaartepunt (gemiddelde) voor de verticale as terug als voor de horizontale as. Deze functie *mean* neemt 1 matrix als parameter, en interpreteerd deze matrix als histogram.

Variantiebepaling

Nadat het gemiddelde van iedere dataset (ieder histogram) bepaald is, moeten we voor iedere dataset de variantie bepalen, en de covariantie tussen de datasets in het histogram. Ook dit is in een aparte functie geïmplementeerd. Ook hier stellen we de waarde van ieder punt voor door de functie $V(x,y)$. Berekening van de varianties voor beide datasets (x en y) en de covariantie wordt dan :

$$Var(X) = \frac{1}{N-1}\sum_{x=1}^{x_{max}} (x-\overline{X})^2\{\sum_{y=1}^{y_{max}} V(x,y)\} \tag{7.6}$$

$$Var(Y) = \frac{1}{N-1}\sum_{y=1}^{y_{max}} (y-\overline{Y})^2\{\sum_{x=1}^{x_{max}} V(x,y)\} \tag{7.7}$$

$$Cov(X,Y) = \frac{1}{N-1}\sum_{y=1}^{y_{max}}\sum_{x=1}^{x_{max}} (x-\overline{X})(y-\overline{Y})V(x,y) \tag{7.8}$$

De functie *regressie* die hiervoor opgesteld is berekent deze 3 eigenschappen van het histogram. Als parameters van deze functie*regressie* dienen het histogram (als matrix) en de coördinaten van het zwaartepunt opgegeven te worden.

Maskering van extreme coördinaten

In deze functie worden alle punten waarvan één van de coördinaten 0 is gemaskeerd. Maskering gebeurt door de waarde van het punt op 0 te zetten. De reden waarom we dit doen is dat punten die geen intensiteit hebben ons geen informatie verschaffen. De punten die hier liggen zijn eerder punten die anders als achtergrond gecatalogeerd worden.

Ook maskeren we de punten waar één van de coördinaten maximaal is. De verantwoording hiervoor is dat deze punten zich op de rand van het lineair meetbereik van de camerasensoren bevinden. De waarden in deze punten zijn de integratie van alle punten die zich op deze coordinaat en erboven bevinden. *(* een punt met coördinaat $(10,320)$ zal door het beperkte meetbereik invloed hebben op het punt met coördinaat $(10,255)$ liggen. *)* Omdat deze meetfout de histogrammen verstoren, verwaarlozen we deze punten door ze te maskeren.

7.2.2 Opdeling in klassen

Als we de histogrammen visueel onderzoeken, merken we intuitief dat er meerdere gebieden van grote dichtheid zijn (dus veel punten met hoge waarde, dicht bij elkaar). Deze gebieden komen overeen met verschillende klasses en moeten we apart onderzoeken, omdat ze ons andere principale componenten zullen opleveren. Bovendien dienen experten in een latere fase ook punten aan te duiden die tot een andere klasse behoren. Deze punten zullen we ook allemaal op een gelijkaardige wijze groeperen en analyseren aan de hand van de principale componenten.

We delen deze gebieden zelf op in kleinere deelgebieden die we apart onderzoeken om principale componenten te bepalen. De opdeling van deze gebieden is eerder intuitief en vereist wat oefenwerk. De keuze van deze opdeling zal de resultaten beïnvloeden, maar een goede opdeling is deze die de resultaten minimaal beinvloed bij een kleine wijziging in de opdeling. In figuur 7.1 is een opdeling in klassen weergegeven.

Iedere klasse wordt afzonderlijk onderzocht voor principale componenten. Overeenstemmende gebieden in de 3 histogrammen worden samen onderzocht om zo een resultaat in drie dimensies te bekomen *(rood, groen en blauw)*

7.2.3 Analyse van een klasse

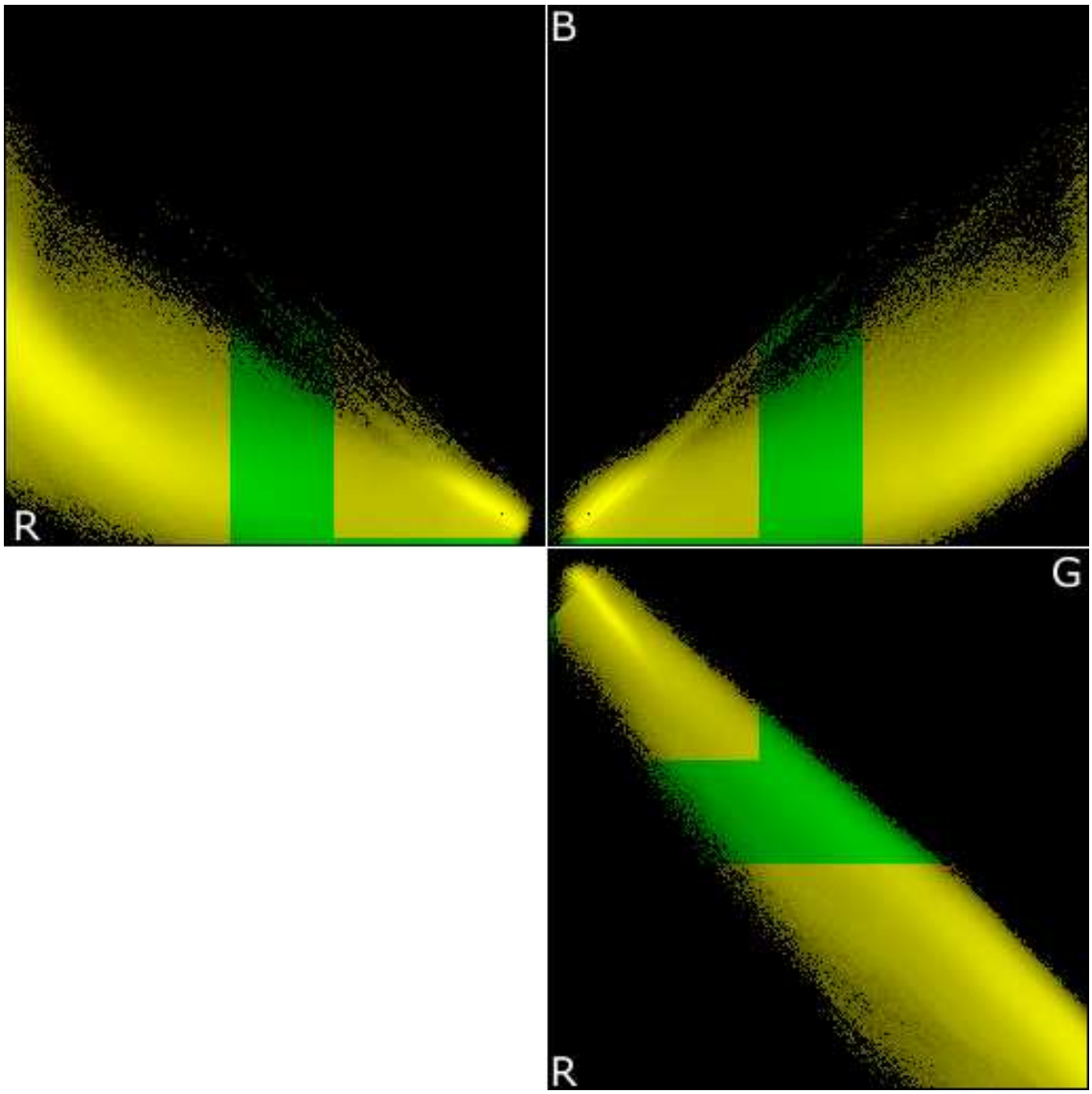
De eerste stap in het onderzoeken van een klasse is het bepalen van het gemiddelde voor zowel rood, groen als blauw. Omdat we 3 datasets hebben, vinden we voor iedere dimensie 2 gemiddelde waarden. De bepaling van de gemiddelde waarden gebeurt met de functie *mean* eerder beschreven.

Daarna onderzoeken we iedere dataset op varianties en covariantie. Dit kan gebeuren door de functie *regressie* toe te passen op ieder histogram. Na deze analyse bekomen we volgende gegevens:

- 2 gemiddelden voor iedere kleur;
- 2 varianties voor iedere kleur;
- 3 covarianties.

De twee gemiddelden en de twee varianties voor iedere dimensie kunnen uitgemiddeld worden zodat we voor iedere dimensie één gemiddelde en één variantie bekomen.

De drie bekomen gemiddelde bepalen de oorsprong van het nieuwe assenstelsel dat we zoeken. De rotatie bepalen we door eigenwaarden decompositie. De bekomen varianties en covarianties kunnen



Figuur 7.1: Opdeling van de 3 RGB histogrammen in klassen. Deze grafieken geven de logaritme van het gewicht van ieder punt weer, waar de logaritme groter is dan 1. De gele gebieden bestaan uit punten die duidelijk tot dezelfde klasse behoren. Elk van deze gele gebieden analyseren we afzonderlijk. In de groene gebieden is de klasse toekenning onduidelijk. De hoeveelheid punten in deze gebieden is bovendien verwaarloosbaar. We nemen deze punten niet mee in de berekeningen.

Tabel 7.1: Overzicht van analyseresultaten voor één klasse

gemiddelden		varianties		covarianties	
$\overline{R}$	214.65	$VAR(R)$	695.50	$COV(R,G)$	585.49
$\overline{G}$	220.08	$VAR(G)$	722.54	$COV(R,B)$	507.80
$\overline{B}$	65.19	$VAR(B)$	457.22	$COV(G,B)$	418.49

Tabel 7.2: Principale componenten voor één klasse

$$
\begin{aligned}
P^{-1} &= \begin{bmatrix} 0.70634 & -0.18997 & -0.68190 \\ 0.32561 & -0.76815 & 0.55129 \\ 0.62853 & 0.61143 & 0.48072 \end{bmatrix} \\
D &= \begin{bmatrix} 47.80068 & 0 & 0 \\ 0 & 174.02063 & 0 \\ 0 & 0 & 1653.43949 \end{bmatrix} \\
\% &= \begin{bmatrix} 2.549\% & 0 & 0 \\ 0 & 9.280\% & 0 \\ 0 & 0 & 88.171\% \end{bmatrix}
\end{aligned}
$$

we in een *covariantiematrix* schikken:

$$
\begin{bmatrix} \sigma_{rr} & \sigma_{rg} & \sigma_{rb} \\ \sigma_{gr} & \sigma_{gg} & \sigma_{gb} \\ \sigma_{br} & \sigma_{bg} & \sigma_{bb} \end{bmatrix}
$$

Met behulp van eigenwaarde ontbinding kunnen we uit deze covariantiematrix de principale componenten voor de beschouwde gegevens bepalen. Ook kan bepaald worden welk aandeel iedere principale component heeft in de gegevens. De manier waarop dit wiskundig opgelost kan worden is beschreven in bijlage A. In octave kan dit gebeuren met de ingebouwde functie *eig*.

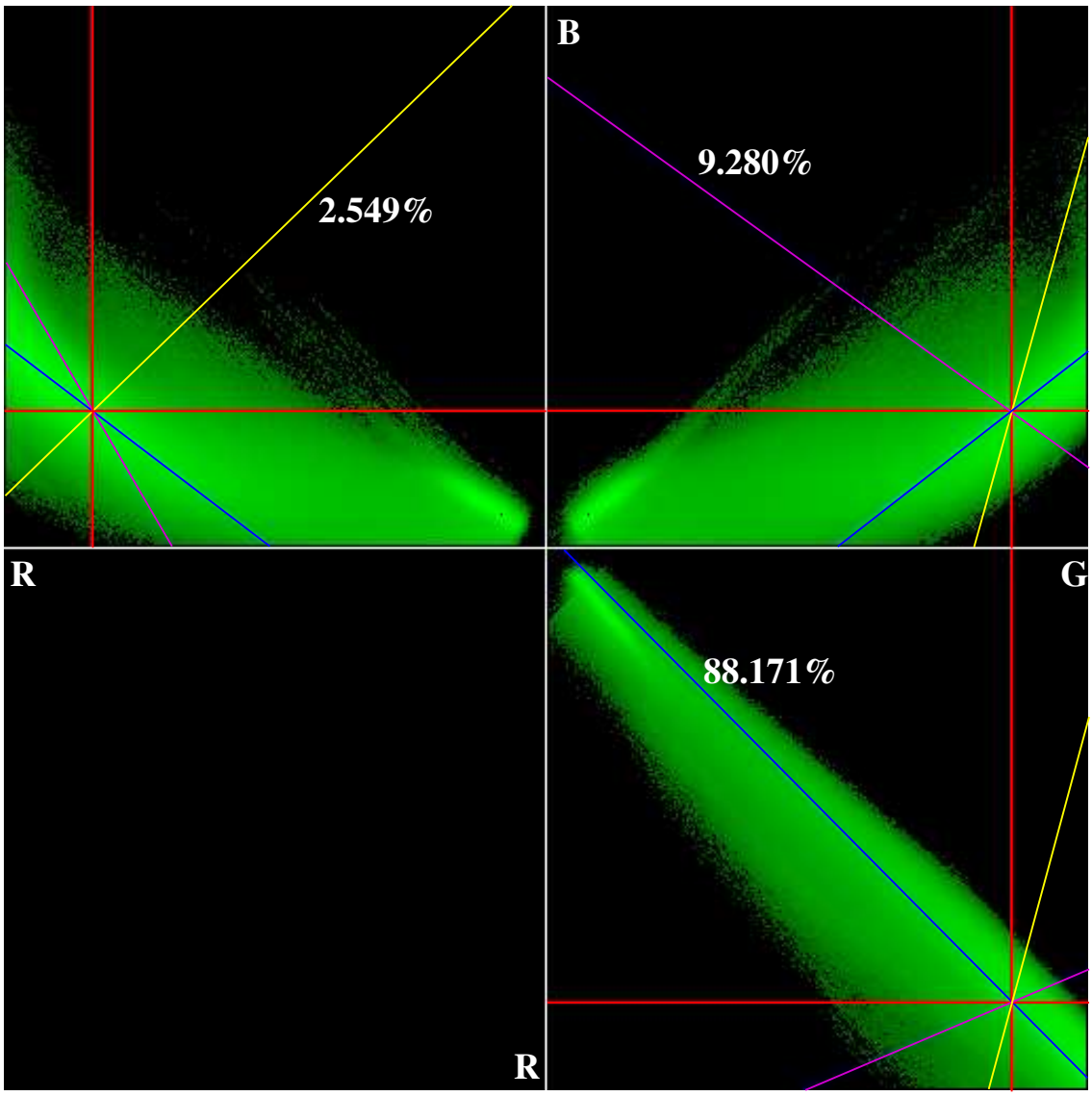
Deze analyse kan uitgevoerd worden voor iedere klasse die we afgezonderd hebben in de oorspronkelijke gegevens. In tabel 7.1 en tabel 7.2 vindt u de resultaten van deze analyse op de histogramman van één filmbestand. (De histogrammen die weergegeven zijn in figuur 7.1).

7.3 Vergelijking met Gauss curve fitting

Gauss curve fitting probeert een twee-dimensionele Gauss functie zo op de figuur te plaatsen dat de som van de kwadraten van de afstanden punt-curve zo klein mogelijk is. Indien we zeker weten dat onze punten normaal verdeeld zijn, zou deze methode ons ook een uitweg kunnen bieden voor het bepalen van de componenten. Echter niets zegt ons dat deze verdeling helemaal Gaussisch is. We hebben deze fit enkel uitgevoerd voor het rood-groen histogram.

We merken op dat het verloop van de richting van de hoofdassen en ook de variaties langs deze assen gelijkaardig zijn maar dat de gefitte gemiddelden veel sterker aan de randen van het histogram gelegen zijn. Ze komen dus niet overeen met de werkelijke gemiddelden.

De resultaten van deze analyse kunt u bekijken in figuur 7.3. Verder zijn de getallen ook nog weergegeven.



Figuur 7.2: Visuele weergave van de resultaten van een principale componenten analyse op een "golden" (monocolore) appel. De rode assenkruisen geven de ligging van het gemiddelde aan voor iedere kleur. De blauwe lijnen zijn de projectie van de as met maximale variantie (88%), de paarse lijnen stemmen overeen met de as die 9% van de variaties op zich neemt, en de gele lijnen stellen de minimale variantie-as voor.

$$Z_0 + Ae^{\frac{-1}{2(1-\sigma_{xy}^2)}\left(\left(\frac{x-\overline{x}}{\sigma_x}\right)^2+\left(\frac{y-\overline{y}}{\sigma_y}\right)^2-2\sigma_{xy}\frac{(y-\overline{y})(x-\overline{x})}{\sigma_x\cdot\sigma_y}\right)} \tag{7.9}$$

eerste fit			tweede fit		
Z_0	=	4705 ± 215	Z_0	=	3838.8 ± 106
A	=	$5.277 \cdot 10^5 \pm 3.08 \cdot 10^3$	A	=	$1.8153 \cdot 10^5 \pm 3.05 \cdot 10^3$
$\overline{x}$	=	25.591 ± 0.0562	$\overline{x}$	=	254.07 ± 1.53
σ_x	=	9.2197 ± 0.0422	σ_x	=	45.869 ± 0.3781
$\overline{y}$	=	26.923 ± 0.0691	$\overline{y}$	=	240.32 ± 1.47
σ_y	=	11.307 ± 0.0524	σ_y	=	44.808 ± 0.35
σ_{xy}	=	0.97025 ± 0.00103	σ_{xy}	=	0.97729 ± 0.00138

7.4 Verder uit te voeren analyses

De hierboven staande analyse laat een kwalificatie toe in twee klassen van de normale punten van een monocolore appel. Voor bicolore appels zullen in de toekomst gelijkaardige analyse uitgevoerd worden.

Met verdere analyses van specifieke defecten op monocolore appels is het mogelijk om, op basis van de verkregen principale componenten, punten van de appels in verschillende categorieën in te delen. Hiervoor moet er samen met een aantal fruitdeskundigen een database opgezet worden van specifieke defecten. Nadat deze database opgebouwd is, kunnen er gelijkaardige analyses uitgevoerd worden per defectklasse. Om dit de doen moet er in het snapshot pakket een functie voorzien worden om extra gegevens bij een selectie op te slaan. Door de opbouw van het pakket schatten we dat deze aanpassingen in hooguit 1 á 2 dagen uitgevoerd moeten kunnen 1 worden.

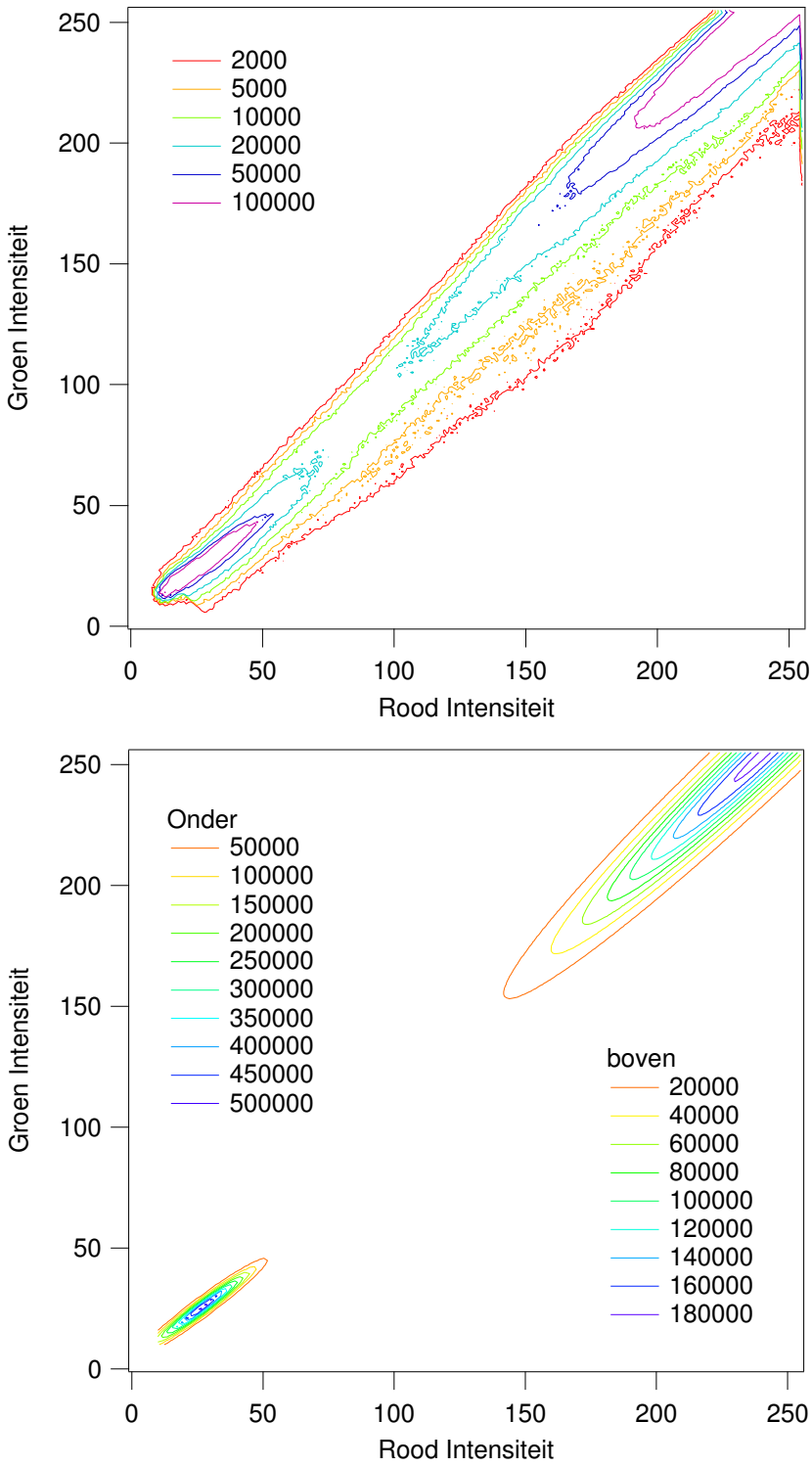
Een gelijkaardige werkwijze zal gevolgd moeten worden voor de bicolore appelsoorten.

Eens we, op basis van de database, alle normale en alle defect klasses geanalyseerd hebben, zal in het HOBU project de classificatie van onbekende appels kunnen starten.

We kunnen hierbij elk punt van de onbekende appel verplaatsen naar alle assenstellsels van alle principale componenten volgens de transformatie (A.1) in bijlage A. We definiëren nu een genormaliseerde afstand d_{norm} tot de oorsprong van de klasse als

$$d_{norm} = \sqrt{\left(\frac{X'}{\lambda_x}\right)^2 + \left(\frac{Y'}{\lambda_y}\right)^2 + \left(\frac{Z'}{\lambda_z}\right)^2} \tag{7.10}$$

In eerste instantie kunnen we nu elk punt indelen in de klasse waarvoor de genormaliseerde afstand d_{norm} tot de oorsprong van de klasse het kleinst is. In een later stadium kan er eventueel een complexer classificatie algoritme uitgewerkt worden, dat ook rekening houdt met andere parameters, zoals de klasse van de buren.



Figuur 7.3: Resultaten van de Gauss curve fit. De bovenste figuur stelt de gemeten distributie voor. De onderste figuur stelt de twee gefitte Gaussdistributies voor.

Hoofdstuk 8

Resultaten

8.1 Interpretatie van enkele typische histogrammen

We hebben enkele appelen bestudeerd samen met de histogrammen, de resultaten zijn veelbelovend

8.1.1 Histogrammen van een "golden" appel

In figuur 8.1 is een "golden" appel met grijsvorming en een rotte plek afgebeeld. Op de appel zijn 3 typische punten aangeduid *(rotte plek, grijsvorming en normale schil)*. Eronder zijn de histogrammen groen-blauw, rood-groen en rood-blauw afgebeeld, met daarop dezelfde punten. In het groen-blauw histogram zien we dat de goede plekken en de slechte plekken duidelijk in een ander gebied op het histogram liggen. Op basis van dit histogram is het dus mogelijk een punt te catalogeren als goed of als slecht. Waar de grens ligt moet nog nader onderzocht worden.

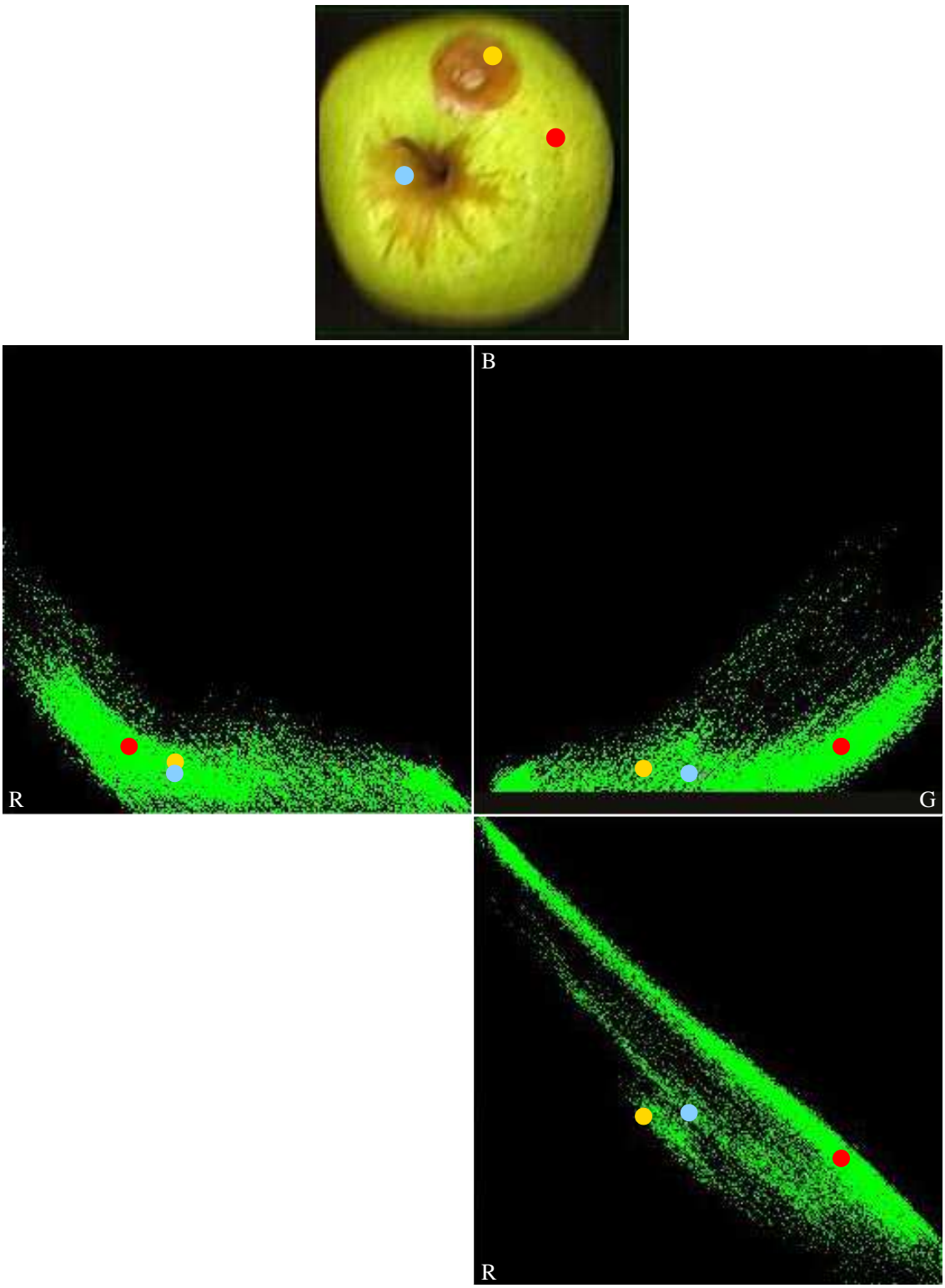
In het rood-groen histogram kunnen we drie langwerpige parallel liggende gebieden onderscheid. De drie aangeduide punten liggen allen in een ander gebied. Op basis van het rood-groen histogram moet het dus mogelijk zijn om de eerder slecht gecatalogeerde punten nog verder onder te verdelen in defectklassen. Omdat we op dit moment enkel histogrammen hebben van volledige appels kunnen we dit niet met volle zekerheid beweren, maar onderzoek van andere appelhistogrammen vertoont gelijkaardige resultaten.

Omdat de punten in het rood-blauw histogram eerder dicht bij elkaar liggen, gaat dit ons waarschijnlijk minder informatie verschaffen over de aard van een punt.

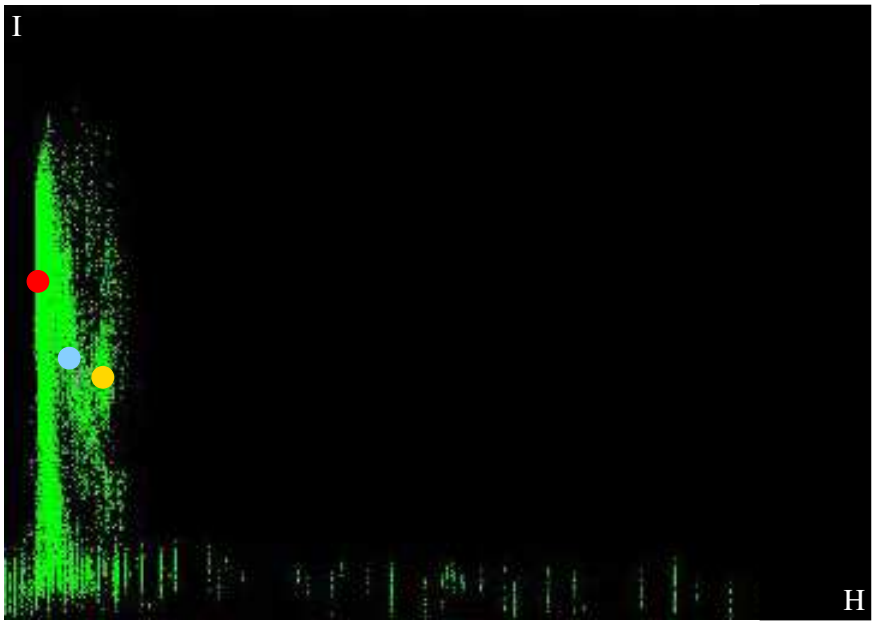
In figuur 8.2 is van dezelfde "golden" appel een histogram afgebeeld dat de tint vergelijkt met de intensiteit. De drie punten zijn ook op dit histogram afgebeeld. Hierop is te zien dat het ook mogelijk moet zijn om op basis van de tint een onderscheid te maken tussen goede en slechte punten. Maar ook de intensiteit geeft ons informatie over goede of slechte punten, hoewel deze intensiteit eerder afhankelijk zal zijn van de belichtingscondities. Mogelijk zijn deze invloeden te minimaliseren door een goede keuze van preprocessing.

8.1.2 Histogrammen van een volledig rotte appel

Figuur 8.3 geeft een volledig rotte appel weer met bijbehorende histogrammen. Ook de punten uit figuur 8.1 zijn op deze histogrammen afgebeeld. Het goede punt komt nu nergens voor op de histogrammen. De rotte plek komt nu op twee van de histogrammen redelijk veel voor en de grijsvorming komt maar op 1 histogram *(dit is in het rood-blauw histogram, dat waarschijnlijk toch niet gebruikt gaat worden voor de moncolore appels)*. Dit versterkt het vermoeden dat het

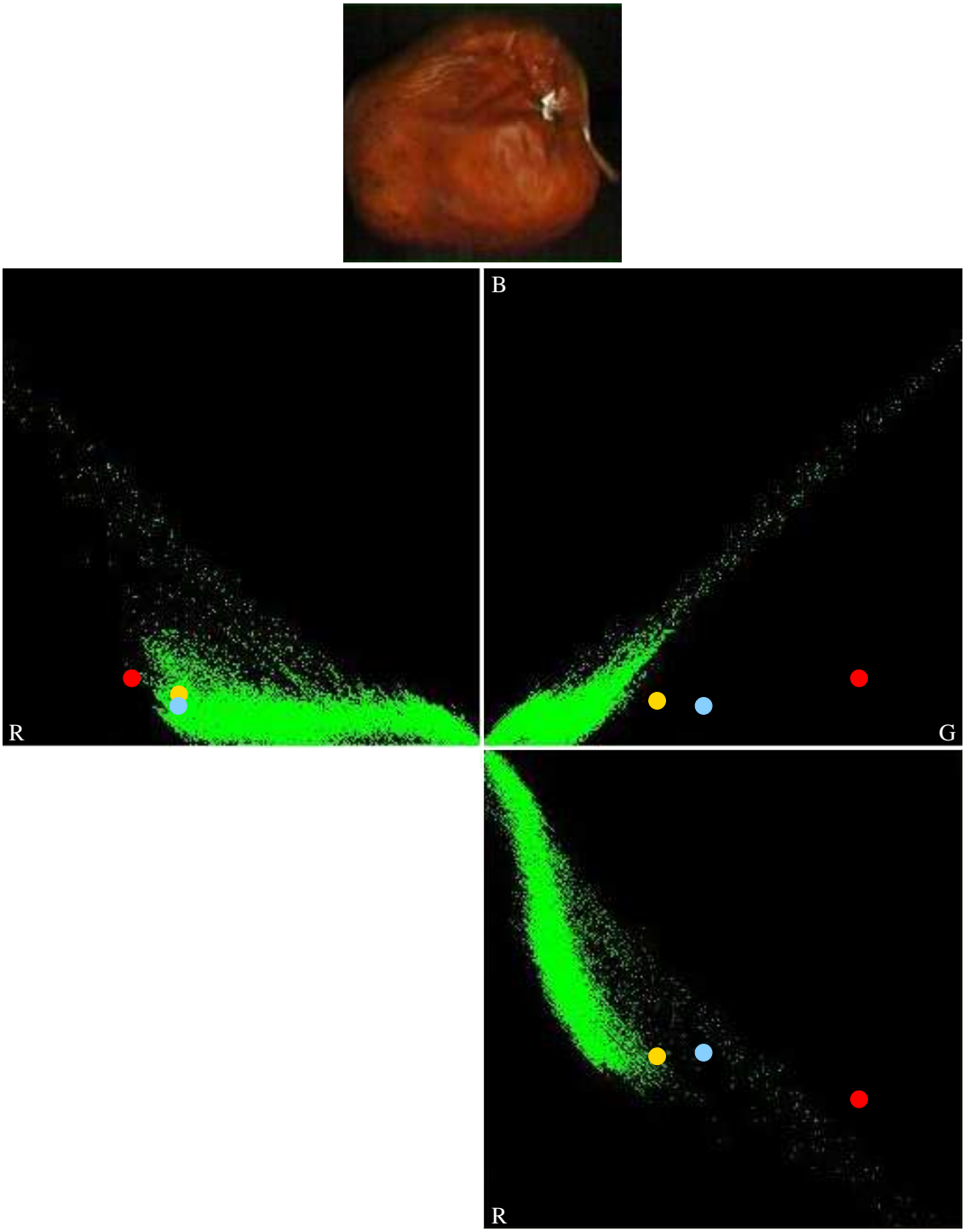


Figuur 8.1: Afbeelding van een “golden” appel. Op de appel zijn drie punten aangeduid die ook in de bijbehorende histogrammen aangeduid zijn.



Figuur 8.2: Tint intensiteit histogram van de appel in figuur 8.1. Ook de punten zijn weergegeven.

mogelijk moet zijn om punten te klassificeren op basis van het rood-groen en het groen-blauw histogram.

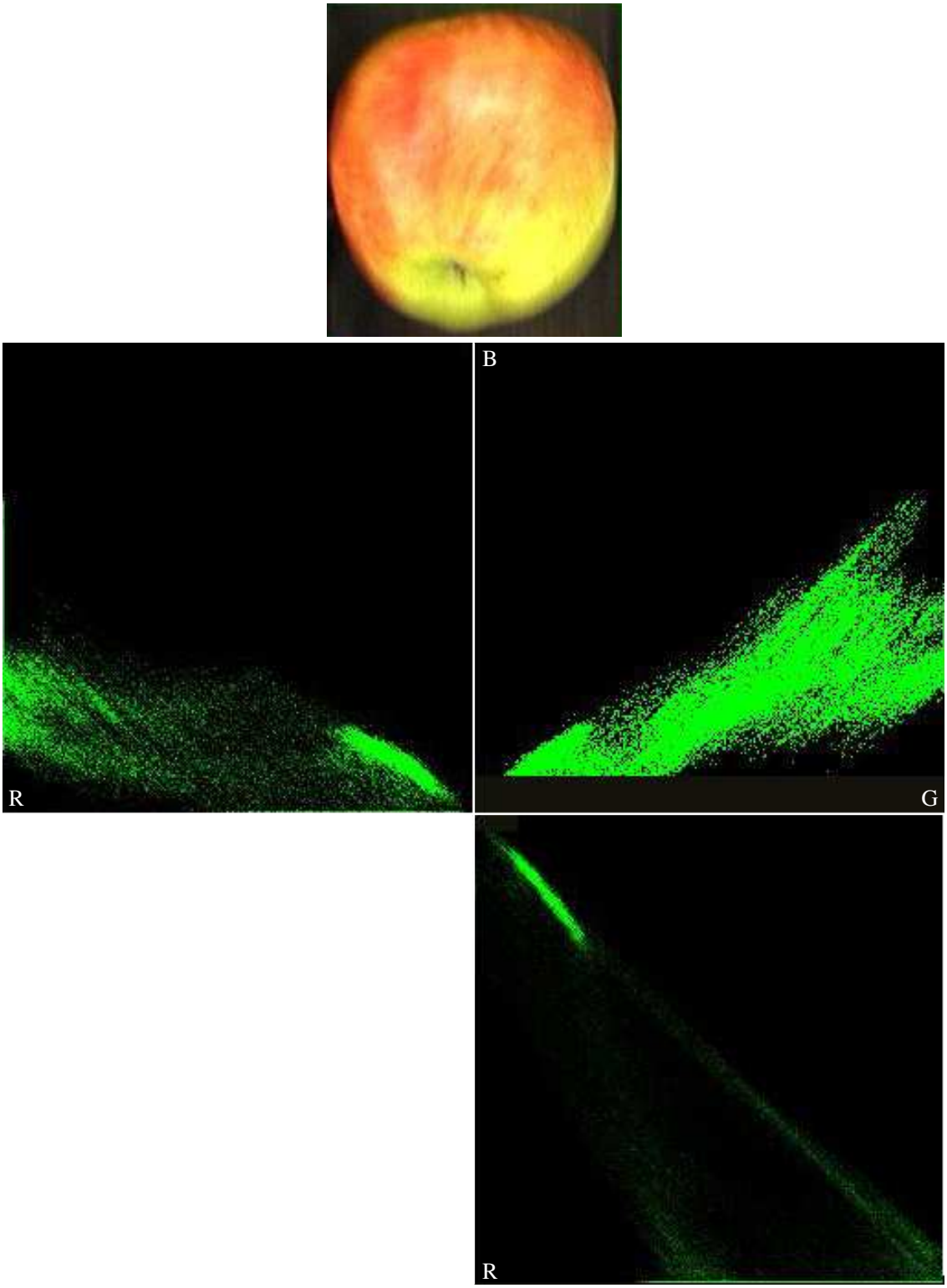


Figuur 8.3: Afbeelding van een rotte appel met bijhorende histogrammen. Ook de punten van figuur 8.1 zijn op het histogram aangeduid.

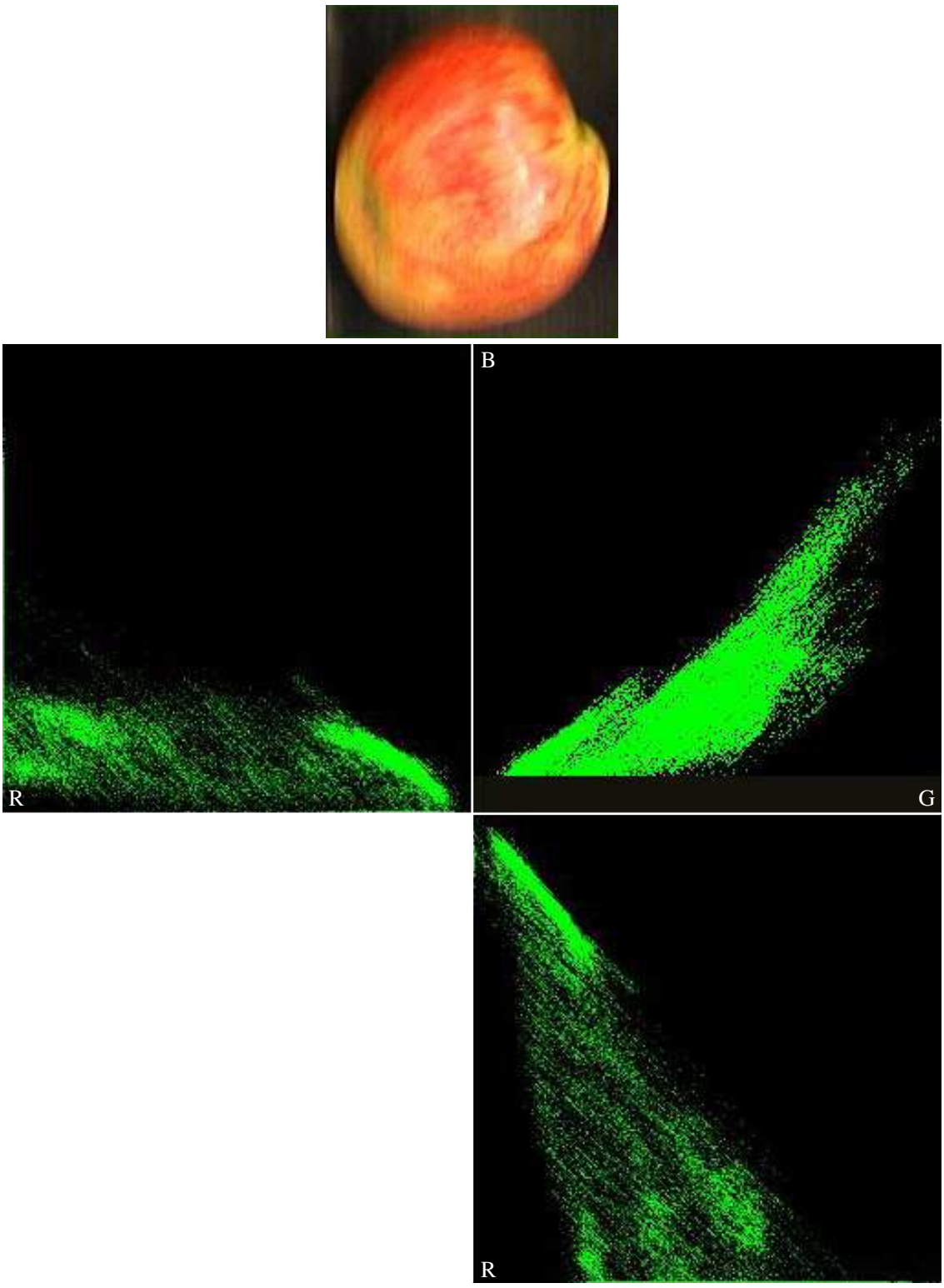
8.1.3 Histogrammen van een “jonagold” appel

Door de grotere complexiteit van de histogrammen hebben we ze nog niet grondig onderzocht. Uit een verkennend onderzoek kunnen we toch wel stellen dat bij de herkenning van de bicolore appels het rood-blauw histogram ook wel in rekening genomen moet worden. Figuren 8.4 en 8.5 geven histogrammen weer van jonagold appels waarop met we zelf geen defecten aantroffen. Figuur 8.5 heeft wel een groter aandeel aan rode schil dan figuur 8.4. Dit is te merken in de histogrammen waarin rood voorkomt. In deze histogrammen zijn meer heldere punten te vinden bij hogere rood-waarden.

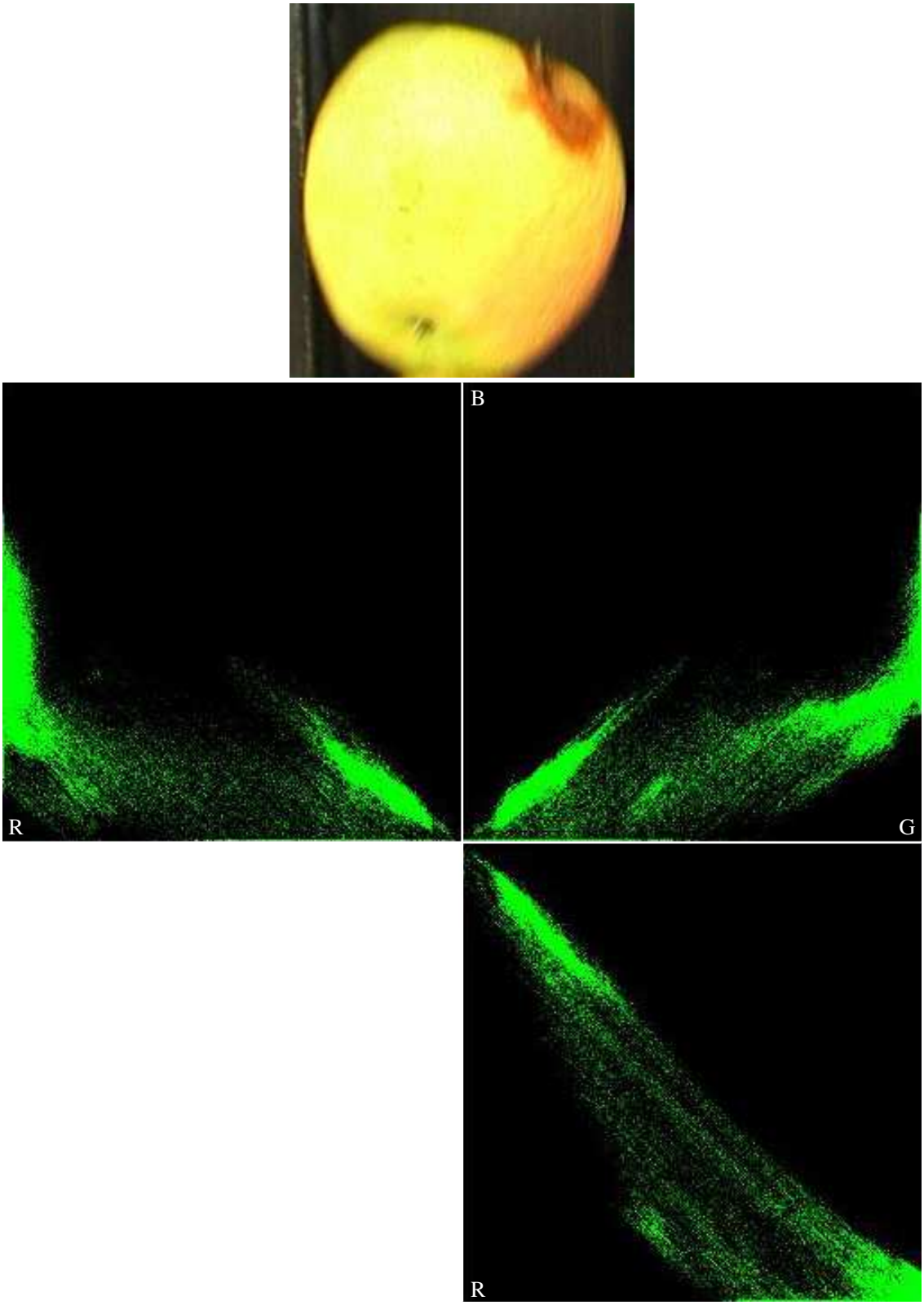
Figuur 8.6 en figuur 8.7 tonen beiden een jonagold met een rotte plek en figuur 8.8 geeft tenslotte een jonagold appel met een barst in de schil weer. Op het eerste zicht zijn hier geen echte over-duidelijke gebieden aan te duiden. Voor classificeren van jonagold appel (bicolore appels in het algemeen) is er dus nog meer onderzoek nodig.



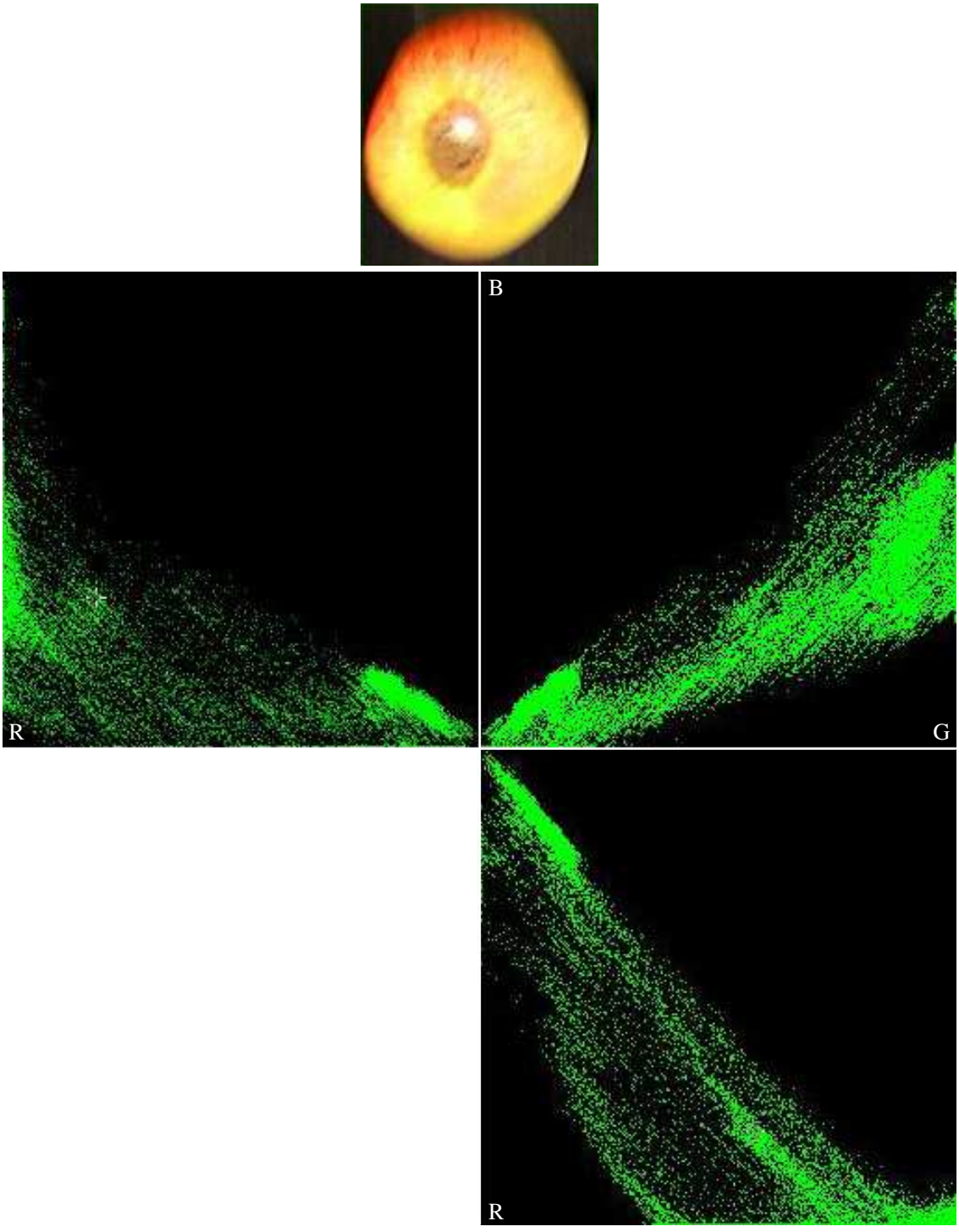
Figuur 8.4: Afbeelding van een “jonagold” appel met bijhorende histogrammen.



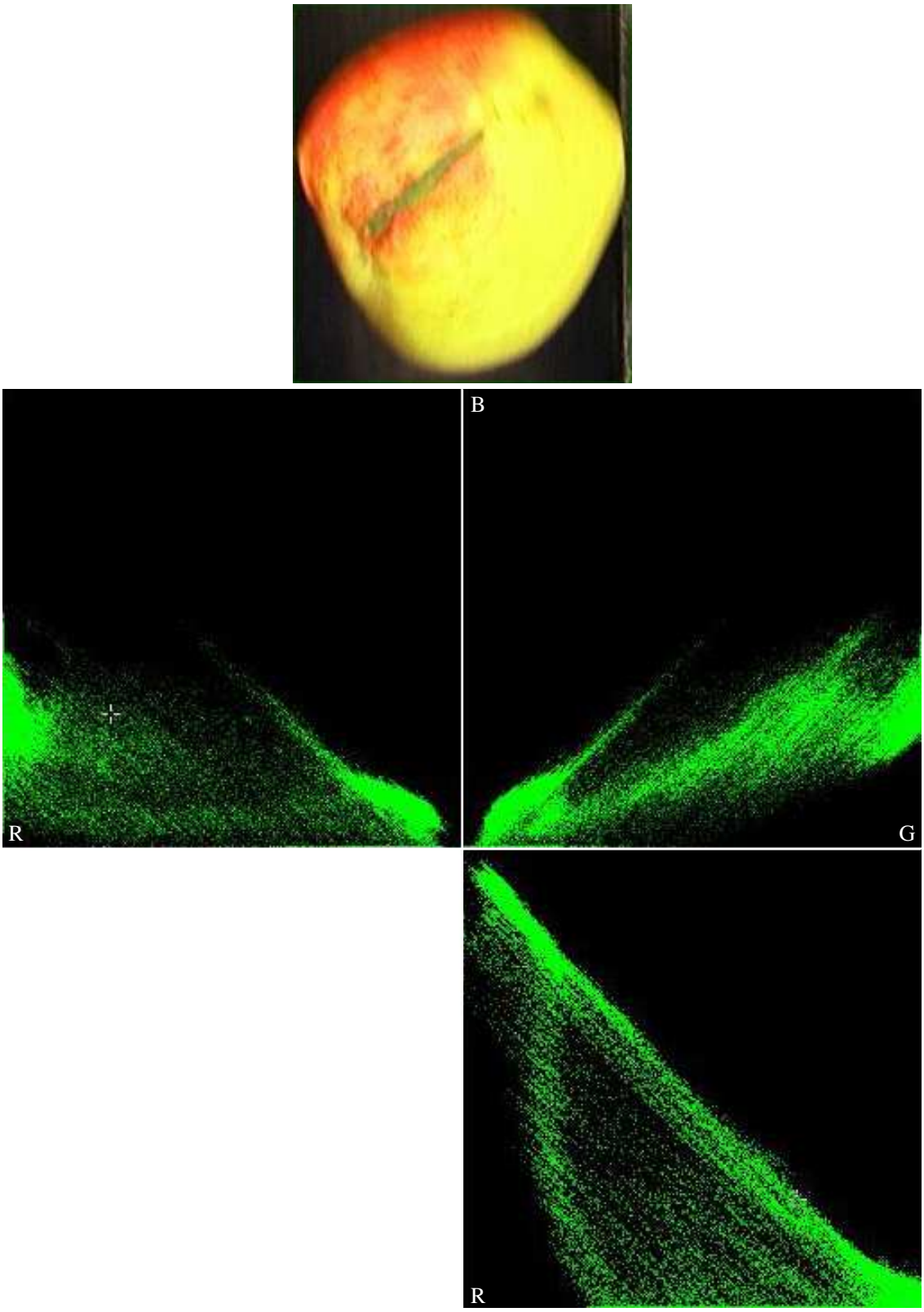
Figuur 8.5: Afbeelding van een “jonagold” appel met bijhorende histogrammen.



Figuur 8.6: Deze “jonagold” appel heeft een rotte plek. De histogrammen die erbij horen zijn ook afgebeeld.



Figuur 8.7: Deze “jonagold” appel heeft een rotte plek. De histogrammen die erbij horen zijn ook afgebeeld.



Figuur 8.8: Deze “jonagold” appel heeft een barst in de schil. De histogrammen die erbij horen zijn ook afgebeeld.

Figuur 8.9: Appels die tegen elkaar liggen

8.2 Resultaten van het automatisch selecteren van appels

8.2.1 Werking van het verbonden componenten algoritme

Automatisch selecteren van appels op de transbordband is uitvoerig beschreven in 5.12. We geven hier een kort overzicht van de resultaten die met deze procedure bekomen worden. Figuur 8.9 geeft originele beeld aan waarop dit voorbeeld toegepast is. We zien hier zowel een appel die volledig los ligt van omringende objecten, als 2 appels dit tegen elkaar liggen.

Figuur 8.10 geeft de resultaten weer van de twee filterketens in het “selectionscan” filter. Beide ketens hebben achtereenvolgens een “threshold” filter, een “despeckle” filter en een “componentscan” filter. Het enige verschil tussen de twee afbeeldingen is dat de bovenste een hoge drempelwaarde voor het thresholding filter heeft, en de onderste een lage drempelwaarde heeft.

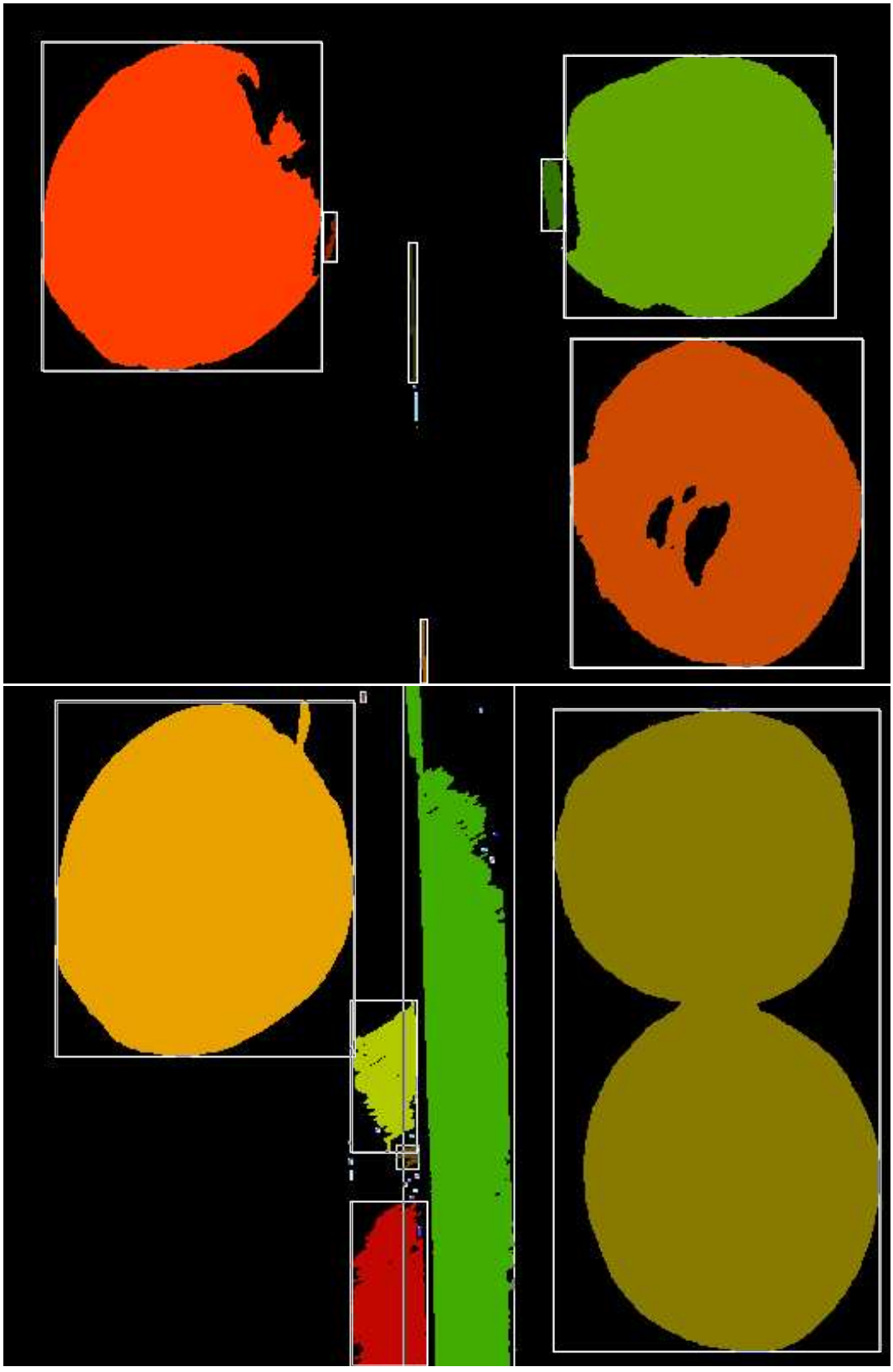
Op deze afbeeldingen is zeer goed te zien hoe er onderscheid gemaakt wordt tussen twee appels die tegen elkaar liggen. Bij de lage drempel worden de twee appels herkend als één groot blok, bij de hoge drempel wordt er wel degelijk onderscheid gemaakt tussen de twee appels. Wel is het zo dat de kadertjes bij de hoge drempel een deel van de appel wegsnijden. Op het resultaat van de *hoge* drempel wordt vervolgens nog een keer het filter “Selectionfilter” toegepast. Hierdoor worden de kaders die te ver afwijken van een vierkant, of de kaders die te klein zijn gewist. Er blijven dan nog drie kaders over in de bovenste afbeelding van figuur 8.10.

Deze drie kaders worden omsloten door de twee grote kaders in de onderste afbeelding. De “Selectionscan” filter zal vervolgens deze drie kaders proportioneel uitbreiden totdat ze ergens raken aan de omsluitende kader. Het uiteindelijke resultaat van he algoritme is afgebeeld in figuur 8.11. Er

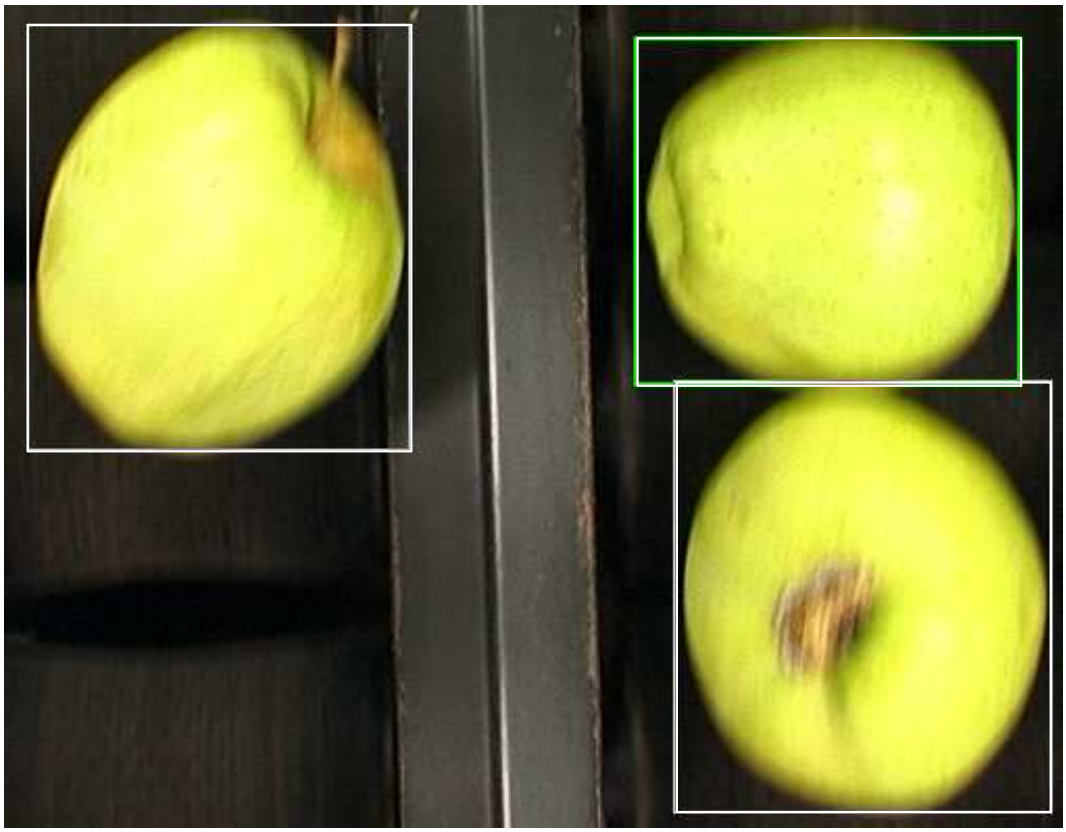
wordt dus wel degelijk onderscheid gemaakt tussen de verschillende appels die tegen elkaar liggen.

Beperkingen bij dit algoritme doen zich vooral voor bij aaneenliggende appels met een groot verschil in grootte. De rand van de grotere appel bedekt dan in feite het raakgebied van de appels. Dit wil zeggen dat er geen donkerder gebied is tussen de twee appels in, wat resulteert in het niet goed onderscheiden van de twee appels. Ook bij te felle belichting is er geen donker gebied tussen de appels, wat ook resulteert in een slechte werking. De kader die dan de twee appels aanduid in het hoge-drempel beel zal dan verwijderd worden, omdat deze te langwerpig is. De aanliggende appels zullen dan helemaal niet beschouwd worden als appels.

Omwille van dit probleem moet er steeds gezorgd worden voor een voldoende hoeveelheid aan gegevens. De niet gedetecteerd aanliggende appels mogen slechts een fractie uitmaken van het totaal aantal appels in de gegevensbank.



Figuur 8.10: Hier zien we het resultaat van achtereenvolgens een “threshold”-filter, een “despeckle”-filter en een “componentscan”-filter. In de bovenste afbeelding heeft het “thresold”-filter een hoge drempelwaarden, in de onderste afbeelding heeft het filter een lage drempelwaarde.



Figuur 8.11: Resultaat van selecties zoeken.

8.2.2 Enkele getallen

Om een voldoende aantal gegeven te verkrijgen voor de principale componten analyse hebben we één pc één nacht beelden laten analyseren op dit principe. *(De pc is dus van 17.00 tot 9.30 enkel en alleen bezig geweest met het analyseren van beelden. In deze analyse zat zowel het bepalen van de selecties als het berekenen van de drie 2D histogrammen van iedere bekomen selectie)*

Gedurende deze tijd heeft de pc zes cd's met videobeelden verwerkt. Hieruit hebben we 47992 selecties van appels bekomen. Als vergelijking kunnen we hierbij geven dat één enkele persoon twee weken gewerkt heeft aan het bepalen van 30423 selecties op beelden van appels in de boomgaard.

Besluit

De opbouw van de ondersteunende software heeft de meeste tijd in beslag genomen. Deze software die voor veel toepassingen kan gebruikt worden word vrijgegeven onder GPL [11]. De software verbonden met de appeldetectie en -kwalificatie wordt niet publiek vrijgegeven. Dit zijn de filters en analysers die we als aparte modules geprogrammeerd hebben. Ook de sessiebestanden, die de wijze aangeven waarop de filters en anlysers gebruikt worden, worden niet vrijgegeven.

Met het programma kunnen appels *(of andere interessante gebieden)* in videobeelden geselecteerd worden. Dit kan in de huidige toepassing zowel automatisch, voor beelden van appels op de transportband als handmatig, voor beelden van de boomgaard. Voor iedere bekomen selectie kunnen dan de drie bijhorende 2D histogrammen berekend worden. De goed doordachte opbouw van het programma maakt het mogelijk om op een snelle manier de resultaten van de analyses te bekomen. Hierdoor kunnen op korte tijd voldoende gegevens verzameld worden om Principale Componenten Analyse op te kunnen toepassen. Met de resultaten van deze PCA kan dan de aanwezigheid van appels in het beeld gedetecteerd worden en kan de relevante informatie uit het beeld geëxtrahereerd worden. De verwerking van deze gegevens wordt gedaan door een wiskundig softwarepakket GNU octave [22] omdat dit programma alle funties biedt die hiervoor nodig zijn. Het is dus onnodig om dit aan ons programma toe te voegen.

De bekomen 2D histogrammen hebben we intuitief opgedeeld in twee klassen. Dit omdat er duidelijk twee gebieden met hoge puntendichtheid te zien waren in het 2D histogram. Hierna hebben één van deze klassen onderzocht op principale componenten. Voor het rood-groen histogram hebben we bovendien een Gausscurve fitting toegepast. Bij normaal verdeelde punten zou dit een hulp kunnen zijn voor de bepaling van de principale componenten. De gefitte gemiddelden liggen , jammer genoeg, veel te dicht bij de rand van het histogram, wat gebruikt van deze methode uitsluit. We hebben ook enkel voor de monocolore appels de principale componenten gezocht. Dit maakt het nu al mogelijk monocolore appels te classificieren. Voor de bicolore appels zijn er gelijkaardige vooruitzichten, hoewel we dit nog niet met zekerheid kunnen zeggen. Wat nu precies de eigenschappen van deze klassen zijn moet nog uitgezocht worden. Verder zullen de principale componenten van specifieke appeldefecten een hulpmidden worden om de kwaliteit van de appels te bepalen.

Er werd gepoogd om het geleverde werk op een zo begrijpelijk mogelijke manier door te geven aan de mensen uit de projectgroep zodat de analyse op de database van appels verdergezet kan worden. Er zal ook zeker nog een database opgesteld moeten worden met verschillende soorten appeldefecten. Hiermee kunnen dan alle defectklasses geanalyseerd worden en zal in het HOBU project de classificatie van onbekende appels kunnen starten.

Bijlage A

Gebruik van eigenwaarden ontbinding voor principale componenten analyse

In 7.2.3 hebben we gebruik gemaakt van eigenwaarden ontbinding voor het bepalen van de rotatiematrix. Toepassing van deze rotatiematrix transformeert de originele gegevens naar principale componenten. Wat volgt is de verantwoording waarom we deze ontbinding mogen gebruiken om de rotatiematrix te bepalen. De varianties en covarianties die bekomen werden door analyse van de histogrammen kunnen we schikken in een *covariantiematrix* schikken:

$$\begin{bmatrix} \sigma_{rr} & \sigma_{rg} & \sigma_{rb} \\ \sigma_{gr} & \sigma_{gg} & \sigma_{gb} \\ \sigma_{br} & \sigma_{bg} & \sigma_{bb} \end{bmatrix}$$

Volgens het theorema van de eigenwaardenontbinding kunnen we deze matrix herschrijven : $\sigma = PDP^{-1}$. Het theorema zegt ook dat D een diagonaalmatrix is, en dat de kolommen van P orthogonale vectoren zijn. Omdat deze kolommen orthogonale vectoren zijn geldt dat $P^{-1} = P^T$.

Stel nu dat we een kolomvector (een punt uitgedrukt als vector) vermenigvuldigen met de covariantiematrix. Dit is dan equivalent met PDP^1 vermenigvuldigen met dezelfde kolomvector

$$\begin{bmatrix} \sigma_{rr} & \sigma_{rg} & \sigma_{rb} \\ \sigma_{gr} & \sigma_{gg} & \sigma_{gb} \\ \sigma_{br} & \sigma_{bg} & \sigma_{bb} \end{bmatrix} \begin{bmatrix} R_1 \\ G_1 \\ B_1 \end{bmatrix} = \begin{bmatrix} V_{1R} & V_{2R} & V_{3R} \\ V_{1G} & V_{2G} & V_{3G} \\ V_{1B} & V_{2B} & V_{3G} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} V_{1R} & V_{1G} & V_{1B} \\ V_{2R} & V_{2G} & V_{2B} \\ V_{3R} & V_{3G} & V_{3B} \end{bmatrix} \begin{bmatrix} R_1 \\ G_1 \\ B_1 \end{bmatrix}$$

Dus vermenigvuldigen met de covariantiematrix is identiek aan een rotatie P^{-1}, deze geroteerde vector vermenigvuldigen met de diagonaalmatrix D en daarna de omgekeerde rotatie P. Na de eerste rotatie werken we in een nieuw assenstelsel, waarin D de covariantiematrix is. Omdat D een diagonaalmatrix is kunnen we besluiten dat er in het nieuwe assenstelsel geen covarianties zijn.

Dit bewijst dat we door eigenwaarde ontbinding twee matrices vinden waaruit we de gezochte principale componenten kunnen afleiden. Matrix P bepaalt de assen (uitgedrukt in componenten van het originele assenstelsel) waarop de covarianties onbestaande zijn. Matrix D bepaalt de varianties op iedere as uitgedruk in P. de inverse van P bepaalt een rotatiematrix die we kunnen gebruiken om een punt in het originele assenstelsel te transformeren naar een punt in het nieuwe assenstelsel.

Assen waarop een grote variantie zit bevatten meer informatie dan assen waarop een kleine variantie zit. Het aandeel in informatie dat iedere as bevat kan berekend worden uit de variantie op die as.

$$\%_k = \frac{\lambda_k}{\sum_{i=0}^{n} \lambda_i}$$

De bekomen principale componenten transformatie kan dan uitgedrukt worden als:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = P^{-1} \left(\begin{bmatrix} R \\ G \\ B \end{bmatrix} - \begin{bmatrix} \overline{R} \\ \overline{G} \\ \overline{B} \end{bmatrix} \right) \tag{A.1}$$

Bijlage B

Korte gebruikersmanual voor het Snapshot programma

In dit appendix bespreken we in het kort hoe iemand die het Snapshot programma wenst te gebruiken het programma dient te installeren en er op een efficiënte wijze mee kan werken.

B.1 Installatie van het programma

De algemene code voor het Snapshot project kan bekomen worden op de site van het Snapshot project die gehost is op Sourceforge [12]. De specifieke code voor de analysers en filters die gebruikt worden binnen het HOBU project "Real-time fruitdetectie en -kwalificatiesysteem" wordt beheerd door de onderzoeksgroep "Digitale Systemen en Signalen" van de KHLim [1]. Om deze code te bekomen dient men met deze onderzoeksgroep afspraken te maken.

Deze code kan gecompileerd worden op een systeem dat voldoet aan de volgende minimale eisen:

- Linux distributie (Bij voorkeur Debian, woody 3.0[15])
- GTKmm (versie 1.3.22 of hoger) [16]
- Avifile (versie 0.7.15 of hoger) [13]
- SQLite (versie 2.6.1 of hoger) [14]
- libSigC++ (versie 1.1.11 of hoger) [23]
- libxml++ (versie 0.18.0 of hoger) [17, 24]

B.2 Gebruikershandleiding

Het eerste wat moet gebeuren, is de database die gebruikt gaat worden aanduiden. Deze database zal dan gebruikt worden om informatie over de selecties die gemaakt worden op te slaan. Open het "File" menu en klik op "set database" (of "new database", wat voorlopig hetzelfde is). Er komt nu een dialoogvenster op het scherm waarin een reeds bestaande database aangeduid kan worden, of waarin de naam en locatie van een nieuwe database aangeduid kan worden. De naam van de database moet altijd eindigen op *.sdb*. Nadat dit gebeurd is kan op "OK" geklikt worden om de

Tabel B.1: Overzicht muiskliks en de bijhorende acties

Hulptoets	Muisknop	Actie horende bij deze muisklik.
-	links	Start het opnemen van een selectie. Het loslaten bevestigt deze selectie
ctrl	links	Start het aanpassen van de bovenzijde en de linkerzijde van een selectie. Het loslaten bevestigt deze verandering
alt	links	Start het aanpassen van de onderzijde en de rechterzijde van een selectie. Het loslaten bevestigt deze verandering
ctrl-alt	links	Start het bewegen van een selectie. Het loslaten bevestigt deze verplaatsing.
-	rechts	Deze muisklik zal het volgende videobeeld tonen
shift	rechts	Deze muisklik zal het vorige videobeeld tonen
ctrl	rechts	Deze muisklik zal de volgende selectie in het beeld aantonen
ctrl-shift	rechts	Deze muisklik zal de vorige selectie in het beeld aantonen

actie uit te voeren. Bij het aanduiden van een niet bestaande database wordt deze automatisch gecreëerd.

Daarna moet er aangeduid worden welk videobestand het programma moet openen. Klik daarvoor op “new movie” in het “File” menu. Er komt weer een bestandsdialoog op het scherm. Hierin kan het videobestand dat men wenst te openen aangeduid worden. Bij videobestanden die *interlaced* zijn kan eventueel de optie “subframed mode” aangezet worden. Iedere frame wordt dan opgesplitst in 2 nieuwe frames : één frame met de even beeldlijnen, en één frame met de oneven beeldlijnen. Door op “OK” te klikken wordt het aangeduide bestand geopend.

Er is ook een mogelijkheid voorzien om filmbestanden te openen die reeds in de database gebruikt zijn. Kies daarvoor in het “File” menu op “Open movie”. Er komt nu een nieuw venstertje op het scherm met daarin een lijst met videobestanden die in de database voorkomen. Op “OK” klikken na het aanduiden van het gewenste bestand, zorgt er weerom voor dat het bestand geopend wordt.

Er is veel aandacht besteed bij de opbouw van het programma om de opname van selecties in de database zo efficiënt mogelijk te laten verlopen. Alle hiervoor belangrijke commando’s kunnen uitgevoerd worden aan de hand van de muisknoppen en 3 modifieer knoppen (shift, alt, ctrl) aan de linkerkant van het toetsenbord. Deze commando’s zijn opgenomen in tabel B.1.

Je moet je niet bekommeren om de opslag van de data. Dit gebeurt automatisch wanneer het programma een nieuw frame toont of wanneer je de film afsluit.

Bibliografie

[1] http://www.khlim.be

[2] http://www.iwt.be

Instituut voor de Aanmoediging van Innovatie door Wetenschap en Technologie in Vlaanderen.

[3] Hans Vandermaesen en Veroniek Jacobs, *Hardware- en software-implementatie van een gezichtsdetectiesysteem*, Eindwerk, Katholieke Hogeschool Limburg, Diepenbeek, 2001.

[4] Vincent Depoortere en Jan De Prins, *Ontwerp van een gezichtsdetectiesysteem*, eindwerk K.U. Leuven, 2000.

[5] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, second edition, pp 675–683, Prentice-Hall, New Jersey, 2001

[6] Jan Genoe, *Real-time fruitdetectie en kwalificatie systeem, aanvraag in het kader van het HOBU fonds 2002*, Departement industriële wetenschappen en technologie, Katholieke Hogeschool Limburg, Diepenbeek, 2002

[7] Bart Timmermans en Dirk Wenmakers, *Kleur- en kwaliteitssortering van appels met kleurensensor*, eindwerk, Katholieke Hogeschool Limburg, Diepenbeek, 1996; Vanessa Radoes en Erwin Vanoppen, *Realisatie van kwaliteitssortering van appels met kleurensensor*, eindwerk, Katholieke Hogeschool Limburg, Diepenbeek, 1997; Bob Maesen en Nico Martens, *Uitbouw van een Geïntegreerde kleur- en kwaliteitsunit voor appelsortering met behulp van een kleurensensor*, eindwerk, Katholieke Hogeschool Limburg, Diepenbeek, 1998; Ward Creyns, *Uitbouw van een Geïntegreerde kleur- en kwaliteitsunit voor appelsortering met behulp van een kleurensensor*, eindwerk, Katholieke Hogeschool Limburg, Diepenbeek, 1999; Walter Smits en Simon Appeltans, *Ontwikkeling van een kleurherkenningssysteem met Fuzzy Logic voor een appelsorteermachine*, eindwerk, Katholieke Hogeschool Limburg, Diepenbeek, 2001.

[8] http://www.esat.kuleuven.ac.be

[9] http://www.mathworks.com/products/matlab/

[10] Mirotech Microsystems Inc., *Arix Hardware overview*, Canada,

[11] http://www.gnu.org/licenses/gpl.html

GNU General Public License , Free Software Foundation, 2001

[12] http://snapshot.sourceforge.net

[13] http://avifile.sourceforge.net

Avifile videobibliotheek

[14] http://sqlite.sourceforge.net

SQLite SQL library engine

[15] http://www.debian.org/

Debian GNU/Linux, universeel besturingssysteem.

[16] http://gtkmm.sourceforge.net

[17] http://www.w3.org/XML/

Extensible Markup Language (XML)

[18] http://www.dai.ed.ac.uk/HIPR2/label.htm

Connected Components Labeling

[19] V. Leemans , H. Magein, M.-F. Destain, *Defect segmentation on Jonagold apples using colour vision and a Bayesian classification method*, Computers and Electronics in Agriculture 23, pp 43–53, 1999

[20] V. Leemans , H. Magein, M.-F. Destain, *Defects segmentation on Golden Delicious apples by using colour machine vision*, Computers and Electronics in Agriculture 20, pp 117–130, 1998

[21] http://www.ri.cmu.edu/pub_files/pub2/schneiderman_henry_2000_3/schneiderman_henry_2000_3.pdf

Henry Schneiderman ,Takeo Kanade, *A histogram-based method for detection of faces and cars*, Proceedings of the 2000 International Conference on Image Processing (ICIP '00) vol. 3, pp 504-507, 2000

http://www.freidok.uni-freiburg.de/volltexte/631/ ,
http://citeseer.nj.nec.com/573520.html

Sven Siggelkow, *Feature Histograms for Content-Based Image Retrieval*, Departement toegepaste wetenschappen, Universiteit Freiburg, Duitsland, januari 2003

[22] http://www.octave.org

GNU octave

[23] http://libsigc.sourceforge.net/

C++ signal library

[24] http://libxmlplusplus.sourceforge.net

C++ xml library