Embedded Vision

Dynamic Card Recognition

Datum: 21 maart 2016

Naam Student:	J. van Wijk
Studentnummer:	10065415
Examinator:	A. Andrioli (HHS)
Tweede Examinator:	P. Burghouwt (HHS)
Opdrachtgever:	J. Snijders (Alten)
Bedrijfsmentor:	A. Prins (Alten)

Samenvatting

Binnen veel automatiseringsprojecten wordt gebruik gemaakt van Computer Vision. Met het ontwikkelen van Computer Vision applicaties kan gezorgd worden dat in processen minder fouten worden gemaakt of dat processen minder arbeidsintensief worden.

Zonder het gebruik Computer Vision moeten we terugvallen op de afhankelijkheid van het menselijke oor, oog en verstand. Dit heeft o.a. als nadeel:

- Lees- en schrijffouten
- Interpretatiefouten (denken, analyseren)
- Uitvoeringsfouten

Het voordeel van Computer Vision is dat men de menselijke factoren zoals hierboven genoemd elimineert inclusief vermoeidheid, beperkte inzetbaarheid ect.

Om kennis te verbreden op het gebied van Computer Vision is een project gestart om d.m.v. moderne algoritmes speelkaarten te herkennen.

Voorwoord

Dit verslag is een weergave van mijn afstudeerstage bij Alten waar ik gewerkt heb aan het ontwikkelen van een dynamische speelkaarten herkenningssysteem.

In dit project ben ik begeleid door Jeroen Snijders (Business Manager) en André Prins (Technisch Consultant). In de vele gespreken met André heeft hij mij waar nodig nieuwe inzichten te geven tijdens het onderzoek.

Dit verslag geeft een duidelijker beeld hoe de techniek voor computer vision functioneert.

Mijn bijzondere dank gaat uit naar de collega's bij Alten die mij de mogelijkheid gegeven hebben om een uitdagende afstudeerstage bij hun uit te voeren, in het bijzonder André Prins voor zijn positieve input.

Ook wil ik Frans van Baardewijk bedanken voor zijn tomeloze inzet voor het corrigeren van mijn "Afrihollands".

Ian van Wijk
Capelle aan den IJssel, maart 2016

Inhoudsopgave

Lijst van figuren

Lijst van Tabellen

[1] http://kurosujun.deviantart.com/art/Petit-Cavalier-Playing-Cards-329993294

Inleiding

In dit verslag wordt de ontwikkeling omschreven van een systeem dat speelkaarten of een willekeurige andere set van kaarten kan herkennen, na het aanleren van de kaarten. Het idee is om een uniform detectiesysteem te maken dat voor meerdere spellen gebruikt kan worden.

Het systeem zal ontwikkeld worden op een Raspberry Pi (Pi), de Pi zal voorzien zijn van een camera module waarmee de beelden in het systeem geladen kunnen worden. Vervolgens zal door de Pi de beelden verwerkt en de verzamelde informatie getoond worden op een externe computer.

Het systeem zal uiteindelijk als demonstratiemodel gebruikt worden voor het consultancy bedrijf Alten, waarmee het op beurzen en events hun vaardigheden en kennis wil tonen op het gebied van Computer Vision. Hiermee hopen ze meer consultants in te gaan zetten op projecten waar beeldherkenning wordt toegepast. Denk bijvoorbeeld aan klanten als: Bosch Security Systems voor het analyseren van beveiligingsbeelden. Sorteren op grootte en kwaliteit van planten of vruchten. Het controleren van lasnaden.

1.1. Alten

Algemene informatie Alten

Alten is een bedrijf dat zich richt op technische consultancy en engineering, zij richten zich op het ondersteunen van opdrachtgevers bij de ontwikkeling van software. Alten is op te splitsen in vier expertises:

- **Consulting & Engineering**
  Zij bieden oplossingen voor problemen op basis van specifieke kennis en ervaring
- **Projecten**
  Ze bieden de mogelijkheid om de verantwoordelijkheid van projecten te dragen. Denk hierbij aan het intern uitvoeren van projecten voor opdrachtgevers.
- **R&D Outsourcing**
  Ze bieden de mogelijkheid om het volledige traject voor de ontwikkeling van producten te leiden en de release te verzorgen.
- **Training**
  Ze zijn een gespecialiseerde partner in training op het gebied van softwareontwikkeling.

Omdat Alten een consultancybureau is, is het belangrijk voor Alten om aan andere partijen te laten zien wat voor kennis zij in huis hebben. Alten verkoop aan klanten hun kennis. Met de kennis die hun consultants hebben, helpen zij klanten hun projecten te realiseren. Om die reden investeert Alten in het ontwikkelen van kennis van hun werknemers. Met deze kennisverbreding hopen zij sterker op de markt te staan. Om hun kennis te verkopen aan klanten, worden regelmatig prototypen ontwikkeld om te laten zien wat voor kennis zij in huis hebben. Met dat in gedachte is dit project tot stand gekomen. Alten hoop met dit project een nieuw prototype te creëren. De baten van dit project zijn indirect terug te zien, het is een kennisinvestering.

Rol van afstudeerder binnen Alten

Binnen Alten zal de afstudeerder horen bij de consultants. De afstudeerder zal tijdens dit project niet geplaats worden bij een bedrijf. Omdat het doel van het project is, om een demonstratiemodel te ontwikkelen, zal de afstudeerder geplaats worden bij de ADC (Alten Delivery Centre).

2. Probleemstelling

Het probleem dat tijdens dit project opgelost moet worden, is het dynamisch herkennen van verschillende speelkaarten. Het systeem moet de mogelijkheid bieden om willekeurig kaarten aan te leren en vervolgens te herkennen. Het systeem moet om het anders te stellen de speelkaarten parameteriseren en vervolgens in een willekeurige andere plaats in het beeld kunnen herkennen.

Er is reeds een poging gedaan voor het ontwikkelen van een systeem dat speelkaarten kan herkennen. Bij de ontwikkeling is gebruikt gemaakt van de Android SDK, zowel voor Tablet als Smartphone. Met dit systeem is het mogelijk om kaarten van het kaartspel “SET” te herkennen.

Figuur 1 - SET! Kaartspel

Het systeem functioneert goed, de verschillende kaarten worden herkend en het systeem is redelijk accuraat. Het systeem is echter heel statisch, het probleem ontstaat als er een nieuw kaartspel, denk bijvoorbeeld aan poker kaarten, toegevoegd moet worden. Voor elk nieuw kaartspel zal opnieuw ontwikkeling gedaan moeten worden voor het herkennen van de verschillende kaarten.

De techniek die het bestaande systeem gebruikt is het classificeren van de vorm, de vulling en de kleur op de kaart. Uiteindelijk wordt samen met die informatie bepaald welke kaart het is. Deze techniek wordt onder andere ook gebruikt bij een onderzoek dat aan de VU[11] gedaan is.

Een probleem wordt geïntroduceerd als er rekening gehouden moet worden met de dynamische aspecten van het nieuw te ontwikkelen systeem. Bij het nieuwe systeem mogen de kaarten niet vooraf geanalyseerd worden en wordt niet bepaald wat de vorm, kleur of vulling is van de “objecten” op de kaart. Het systeem mag niet aangepast worden voor een nieuwe set kaarten. Dit was wel het geval bij het voorgaande systeem; ook bij de techniek gebruikt voor het onderzoek aan de VU is dit het geval. Daarnaast moet het nieuwe systeem om kunnen gaan met kaarten waar de objecten een variatie van kleuren en vormen/patronen bevatten. Een voorbeeld van mogelijke kaarten is te zien in Figuur 2, in vergelijking met de “eenvoudige” SET! kaarten zijn de form en kleur van de kaarten veel moeilijker te herleiden.

Figuur 2 - Poker Speelkaarten

2.1. Doelstelling

Het eerste doel van de opdracht is om dynamisch speelkaarten te kunnen herkennen. Dit houdt in dat het systeem eenvoudig uit te breiden moet zijn een andere stel speelkaarten. Het systeem moet de mogelijkheid bieden om nieuwe kaarten aan te leren en te herkennen.

Het tweede doel van de opdracht is om de smartphone te vervangen door een Raspberry Pi met een camera module. Dit houdt in dat er nieuwe software voor een embedded systeem moet worden geschreven, duidelijke afgebakend van het OS.

Kortom, het doel is bereikt wanneer er een dynamisch prototype, werkend op een Raspberry Pi, is ontwikkeld dat in staat is om zelf nieuwe kaarten te herkennen en benoemen. Het is niet de bedoeling om de spelregels ook te programmeren.

2.2. Projectgrenzen

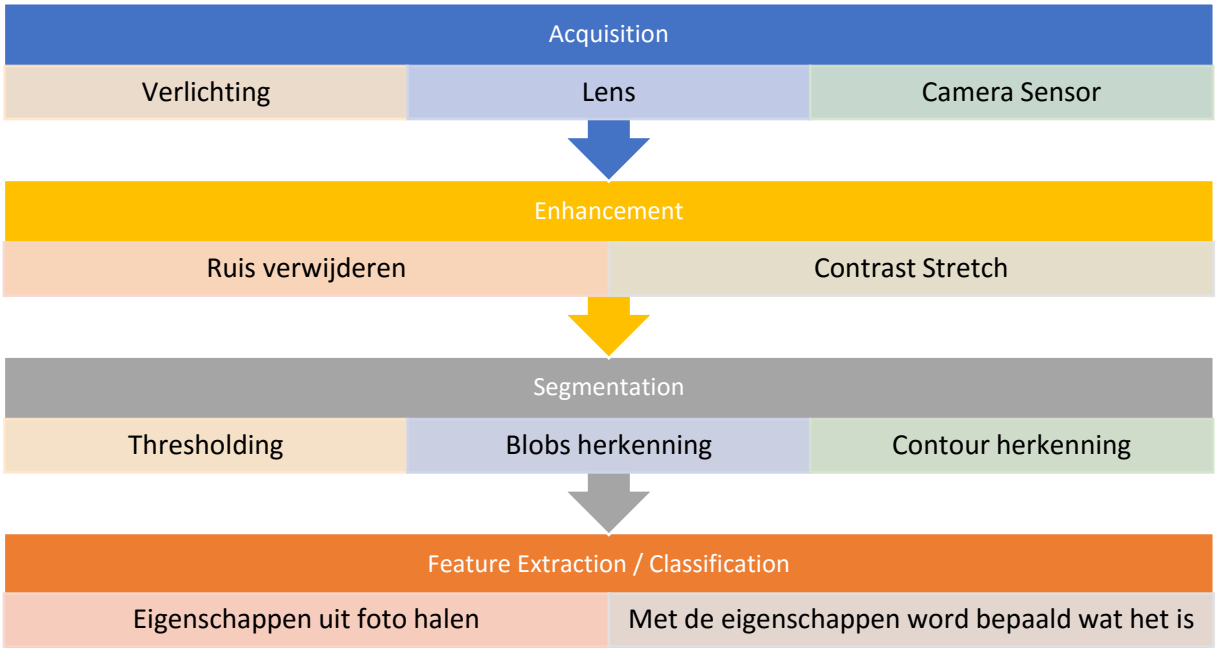
De focus van het project ligt voornamelijk bij het implementeren van een dynamisch-kaart-detectiesysteem. Het zal een onderzoek zijn om met bestaande libraries het project te kunnen ontwikkelen en te kijken welke algoritmes voor het herkennen van objecten het beste werken. Het systeem zal niet direct binnen een project gebruikt worden, maar moet een demonstratiemodel op leveren om de kennis te tonen op het gebied van Computer Vision aan (potentiele) klanten.

2.3. Achtergrond aanpak Vision Project

Voor de aanpak van dit project is allereerst een beperkte mate van achtergrondinformatie nodig. Daarom zal er eerst een hoofdstuk gewijden worden aan het onderbouwen van de project aanpak, vervolgens zal worden uitgelegd hoe dit in het project toegepast zal worden.

Over het algemeen zijn vision projecten op te delen in vier onderdelen.

Voor iedere onderdeel geldt, dat de voorbereidingen die getroffen worden bij de voorafgaande stap, voordelen bieden voor de volgende stap. Hierdoor worden prestatie verbeteringen versterkt in de volgende stappen. Om een voorbeeld te noemen: Als het object beter wordt belicht, waardoor de sensitiviteit van de sensor verlaagd kan worden, zal er minder ruis verwijderd hoeven te worden, waardoor minder rekenkracht nodig is.



Acquisition

Acquisition is verantwoordelijk voor zowel het fysieke aspect van computer vision als het binnenhalen van beelddata. In dit onderdeel wordt onder andere rekening gehouden met:

- de object belichting
- opstelling voor het nemen van foto's
- welke lens gebruikt wordt
- welke sensor / camera gebruikt zal worden
- inladen van beelddata in het systeem

Alle onderdelen samen moeten er uiteindelijk voor zorgen dat de beelden zonder te veel ruis of andere mankementen het systeem binnenkomen.

Bij de belichting van de objecten is het, net zoals bij 'normale' fotografie belangrijk dat de objecten niet direct belicht worden, maar indirect. Dit moet het licht beter verspreiden en een egaal licht op de kaarten laten vallen. Dit heeft voordelen voor latere stappen tijdens het proces voor het herkennen van de kaarten, omdat bij "spot" belichting elementen op de kaart onzichtbaar of onduidelijk kunnen worden.

Als extra onderdeel van de acquisition zal bij dit project ook rekening gehouden worden om na afloop van de Feature Extraction / Classification de beelden door te sturen naar een computer om de verwerkte beelden te tonen aan de eindgebruiker.

Enhancement

Bij de enhancement fase worden de beelden van de camera gepakt en wordt alle mogelijke ruis die ontstaan is tijdens de acquisition, zoveel mogelijk verwijderd. Denk hierbij aan ruis die ontstaan is door het gebruik van een hoge ISO-waarde. De reden voor het verwijderen van de ruis is dat bij latere stappen eventuele ruis voor verstoring kan zorgen bij het herkennen van de objecten.

Technieken

Smoothing Filters

Voor het verbeteren van de afbeelding zijn er smoothing filters. De meest gebruikte filters voor smoothing zijn: Mediaan, Gaussian en bilateraal. Het mediaan filter presteert het beste als het gaat om het verwijderen van “salt-and-pepper” of “speckle” ruis, waar de Gaussian blur weer beter is bij de voorbereidende stap voor de edge detection, dat bij de segmenteren naar voren zal komen. De bilateraal filtering is ook geschikt voor het herkennen van randen, maar dan meer op groot contrast randen.[2] Een groot nadeel van het bilateraal filter is dat het in vergelijking met andere filters een grotere executietijd heeft[10].

Pyramide

Voor het versnellen van het herkennen van de kaarten, kan gebruik gebruik gemaakt worden van een zogenaamde afbeelding pyramide[3]. Een afbeelding pyramide kan gebruikt worden voor het versnellen van het proces. Om een voorbeeld te noemen, tijdens het detecteren van kaarten in een verkleinde afbeelding kan veel tijd bespaard worden omdat de meest relevante informatie nog steeds beschikbaar is maar de meeste overbodige informatie achterwege gelaten wordt. Hiermee kan een grote tijdwinst behaald worden in het systeem. Na het detecteren/segmenteren van de objecten kan er voor gekozen worden om met de verhoudingen terug te gaan naar de originele afbeelding; hiermee kan bij de feature extraction / classification, waar de overbodige informatie wel noodzakelijk is, een hogere resolutie gebruikt worden bij het detecteren om zo een nauwkeurige resultaat te bereiken.

Naar verwachting zal het detectiesysteem een heel eenvoudige implementatie van deze techniek toepassen.

Segmentation

Na enhancement komt de foto in segmentation fase, die verantwoordelijk is voor het opknippen van de foto in verschillende sub-abeeldingen, ook wel bekend als blobs. Dit wordt gedaan door een thresholding uit te voeren op de foto. Hierdoor wordt de grijswaarde-foto omgezet in een zwart-wit foto. Na de eerste omzetting worden de zichtbare objecten opnieuw omgezet maar nu in blobs. Blobs zijn als het ware stukken in de foto die belangrijk zijn voor het herkennen van een object, hierdoor wordt de achtergrond gescheiden van de foto.

Technieken

Voor het segmenteren van de kaarten van de achtergrond zijn de volgende methodes onderzocht, namelijk:

- Canny Edge Detectie
- Labeling

Canny Edge Detectie

De herkenning via Canny maakt gebruik van de Otsu threshold waar vervolgens de Canny Edge detection wordt uitgevoerd. Na het herkennen van de randen/contouren (opgeslagen in vectoren van punten voor elke rand) van de verschillende objecten wordt vervolgens over alle contouren een “Ramer–Douglas–Peucker” algoritme uitgevoerd. Dit houdt kort gezegd in dat alle overbodige punten verwijderd wordt uit de vector.[15]

Na het uitvoeren van het Ramer-Doublas-Peucker algoritme wordt vervolgens gekeken of de verschillende contouren alleen vier punten bevatten, dit zou automatisch andere vormen moet uitfilteren zoals cirkels en dergelijke. Vervolgens wordt van de lange zijde en de korte zijde de lengte

bepaald aan de hand van de vier overgebleven punten. Aan de hand van de lengte en de breedte wordt de verhouding gecontroleerd op een rechthoekige vorm.

Labeling

Voor het segmenteren van de objecten met een labeling algoritme wordt allereerst een Otsu thresholding uitgevoerd. Vervolgens wordt van alle losse blobs een labeling algoritme uitgevoerd. Het nadeel van een labeling algoritme is dat dit algortime relatief vaak door de afbeelding moet gaan om te zorgen dat de afbeelding goed gelabeld is. Het zorgt voor veel vertraging, zoals aangetoond zal worden in hoofdstuk 4. Dit kan verbeterd worden door de afbeelding te verkleinen zoals omschreven bij de Enhancement.

Een ander nadeel van een labeling algoritme is dat na het detecteren opnieuw herkenning gedaan moet worden op de vormen van het object. Dit kan niet, zoals bij het Canny algoritme, direct worden bepaald aan de hand van de contouren.

Feature Extraction & Classification

Als laatste wordt naar ieder gesegmenteerd onderdeel afzonderlijk gekeken. Door alleen te kijken naar de verschillende gesegmenteerde objecten, kan een grote prestatie winst behaald worden. Dit komt door het verwijderen van de achtergrondinformatie.

Techniek

Voor het detecteren van speelkaarten zijn er een aantal mogelijkheden die gebruikt kunnen worden voor het dynamisch herkennen, de bestaande technieken zijn onder andere:

- SURF
- HoG & OpenTDL
- Absolute Difference comparison

SURF

De SURF-methode is een feature extraction algoritme dat uit de afbeelding keypoints haalt. De keypoints van de afbeelding zijn zo geselecteerd dat zij invariant zijn met verschillende afbeeldingen, ongeacht de rotatie en schaling van de afbeelding. Na het selecteren van de keypoints worden descriptors gemaakt van ieder keypoint. De descriptors geven van ieder keypoint de lokale rotatie en intensiteit weer. Met deze informatie kan aan de hand van de afbeelding een match gezocht worden met een template afbeelding. Het SURF-algoritme biedt een goede oplossing voor real-time applicaties. Zoals omschreven:

> *The main interest of the SURF-approach lies in its fast computation of operators using box filters, thus enabling real-time applications such as tracking and object recognition. [14]*

HoG of TLD

De TLD (Tracking-Learning-Detection) is een framework ontwikkeld voor ‘long-term’ tracking. Het is ontwikkeld met het doel om uit willekeurige complexe video streams een object te tracken waarbij tracking fouten regelmatig voorkomen en de detectie niet mag afnemen bij irrelevante informatie. Naar vermoedt, maakt het TLD-framework van ongeveer dezelfde techniek gebruik als HoG, de reden voor dit vermoeden is omschreven bij de uitvoering. De kracht van TLD licht in het lostrekken van het tracking en detection algoritme. Beide algoritmes maken gebruik van een learning algoritme. [20]

Voor het extraheren van de informatie uit een afbeelding aan de hand van de HoG algoritme, wordt allereerst de afbeelding gedeeld in kleinere “Region of Interests” (ROI). Iedere ROI heeft een overlappend gedeelte met zijn naburige ROIs. Vervolgen wordt van iedere ROI op een lokaal vlak de oriëntatie van de grijswaarde verandering bepaald. [8]

Absolute Difference comparison

Voor het gebruiken van de Abusolute Difference comparison (Abs Diff) is het noodzakelijk de kaart uit de afbeelding te knippen en te warpen in een vast formaat. Warpen houdt in dat de kaart zo vervormd wordt dat de kaart van een scheve hoek of lens distortion in een recht formaat geplaatst wordt. Vervolgens wordt een template object, dat ook gewarpt is in een standaardformaat, vergeleken met het “real-time” object. Bij vergelijking wordt op beide objecten een addaptive threshold uit gevoerd en vervolgens het absolute verschil berekend. Na het berekenen van het absolute verschil wordt een som bepaald van alle pixels, als de som een lage waarde heeft, betekent dit dat het hoogstwaarschijnlijk een match is. Zie Figuur 3 en 4 als voorbeeld. Deze techniek kan relatief goed omgaan met rotatie, maar mocht bij het warpen iets niet helemaal goed gaan, dan kan dit voor problemen zorgen bij het herkennen. [13]

Figuur 3 - Abs Diff (no match)

Figuur 4 - Abs Diff (match)

Colour Segmentation

Om samen met één van de voorgaand beschreven technieken de kleur te bepalen van de kaart, kan gebruik worden gemaakt van de volgende techniek. Voor het detecteren van de kleuren op de speelkaarten is er maar één techniek gevonden. De techniek houdt in dat de afbeelding geconverteerd wordt naar een HSV-kleurstructuur. Het voordeel dat de HSV- kleurstructuur heeft ten opzichte van de RGB- kleurstructuur is dat deze beter om kan gaan met kleurverschillen. Eenvoudig gesteld: bij RGB worden de kleuren gemengd en ontstaat de kleur, waardoor bij kleine veranderingen in kleurverzadiging al grote verschillen ontstaan binnen het RGB-histogram. HSV heeft een andere aanpak, daar wordt bijvoorbeeld de kleur groen aangegeven en vervolgens de verzadiging, hierdoor worden de kleurverschillen geëlimineerd door het anders belichten van de kaarten. Hierdoor kunnen

kaarten met dezelfde vorm, maar een andere kleur, toch onderscheiden worden van elkaar. Dit geeft een extra zekerheid bij het herkennen van de kaarten. [6] [7]

Keuze motivatie

Om toch een zo compleet mogelijke test te doen naar bestaande algoritmes, is er gekeken naar algoritmes die los van elkaar staan ten opzichte van de achterliggende technieken. De technieken waarnaar gekeken zijn, richten zich op lokale keypoint descriptors (SURF), lokale texture descriptors (HoG) of een eenvoudige binaire matching (Abs Diff). Zoveel SURF als HoG zijn beide de meest gangbare keuze binnen hun eigen categorie. Er is voor gekozen om Abs Diff ook toe te voegen aan het onderzoek omdat het naar verwachting minder resources nodig heeft om een goede match te vinden. [9]

In Tabel 1 is een overzicht gegeven van de verschillende technieken. [12]

Descriptor	**Binary**	**Invariance**		**Typical Use**	
		Scale	**Rotation**	**Finding Point Correspondences**	**Classification**
HOG	Nee	Nee	Nee	Nee	Ja
SURF	Nee	Ja	Ja	Ja	Ja
Abs Diff	Ja	Nee	Nee	Nee	Nee

Tabel 1 - Feature Extraction & Classification Compare

Dit project wordt zoals genoemd opgedeeld in vier verschillende onderdelen. De vier onderdelen worden afzonderlijk opgesplitst in meerdere onderdelen, zodat het gemakkelijker is om het overzicht te behouden, elkeen met zijn eigen iteratie proces.

3. Proces

3.1. Methodiek Keuze

Om de vorderingen van het project goed bij te houden, is er besloten een methodiek te kiezen. Er is gekeken naar twee opties:

- Waterval
- SCRUM

De nadruk bij de waterval methode licht bij het projectplan; vooraf moet een duidelijk plan en visie zijn dat bereikt moet worden. Er kan vooraf een redelijk accurate schatting gemaakt worden van het budget en het tijdschema. Vooraf wordt duidelijk een design gemaakt aan de hand van de systeemeisen, voordat het systeem ontwikkeld wordt.

In tegenstelling tot de waterval methode biedt SCRUM een flexibel ontwikkelmodel, een aanpasbare planning en een evolutionaire ontwikkeling. Softwareontwikkelaars werken aan kleine modules. Dit heeft het voordeel dat in een project omgeving snel ingespeeld kan worden op veranderingen in de eisen van het systeem. SCRUM is ideaal voor projecten waar het einddoel nog niet duidelijk vastgelegd is.

Omdat dit project uitgevoerd wordt door één elke student is het op het eerste gezicht niet noodzakelijk om een methodiek te kiezen. Toch is er voor gekozen om gebruik te maken van een mix van het methodiek SCRUM en Waterval.

De reden waarom er gekozen is om een “SCRUMFALL”-methode te gebruiken, is omdat het project, zoals in de voorgaande gedeelte omschreven is, op te delen is in vier onderdelen of wel vier fasen. Vooraf is al duidelijk wat ontwikkeld moet worden en daarom zou de waterval methode weer uitstekend geschikt zijn voor het project.

Net zoals in SCRUM zal om de twee weken gekeken worden wat de vorderingen zijn, ook wel bekend als periodes, en er zal ook steeds weer gekeken worden naar de taken die de komende periode afgerond moeten worden. Zo blijft er een goed overzicht van de vorderingen binnen het project en kan er waar nodig geanticipeerd worden.

3.2. Plan

Dit project wordt zoals genoemd opgedeeld in vier verschillende onderdelen. De vier onderdelen (fases) worden afzonderlijk opgesplitst in meerdere subonderdelen (periodes), zodat het overzicht beter behouden kan worden. Fases zijn kort gezegd demonstreerbare onderdelen van een deel van het systeem.

De 4 fases zijn:

- **Acquisition - Beelden in Raspberry Pi laden en tonen op computer**
  Het doel van deze fase is: beelden op de juiste manier binnen halen in het systeem. Hierbij wordt gekeken naar een opstelling waarop de camera geplaatst wordt. Optioneel wordt gekeken naar extra belichting en eventuele andere zaken die zich zouden kunnen voor doen. Naast de fysieke aspecten wordt ook gekeken hoe de foto geladen moet worden in de software. Omdat de Raspberry Pi geen beeldscherm heeft, wordt in deze stap tevens gekeken naar het tonen van de beelden.

  Ontwikkelingstijd: 1-2 weken

- **Enhancement – Verwijderen van eventuele ruis en verbetering van belichting**
  Bij deze fase moet het systeem in staat zijn eventuele ruis te verwijderen en correcties aan te brengen op onderbelichting.

  Ontwikkelingstijd: 2 weken
- **Segmentation – Verwijderen van achtergrond elementen en uitlichten eventuele kaarten**
  Bij deze fase worden alle verschillende elementen (blobs) uit de foto geknipt. Het is de bedoeling dat alle blobs bewaard moeten blijven en dat de achtergrond wordt verwijderd, zodat er alleen gekeken wordt naar de “region of interest” (ROI).

  Ontwikkelingstijd: 2-3 weken

Bij de ontwikkeling van de eerste drie fases is ervoor gekozen om per fase een periode te gebruiken. Hierdoor zijn de ontwikkelingstijden relatief kort. Het uiteindelijke systeem voor het herkennen van de kaarten zal pas in de laatste fase ontwikkeld worden. Deze laatste fase is dan ook opgedeeld in meerdere periodes.

- **Feature extraction / Classification – Dynamische herkenning**
  Als laatste fase moet het systeem helemaal functioneren volgens de eisen. Het systeem moet hier in staat zijn kaarten dynamisch te laden en te herkennen.
  - Periode 4.1
    Hier wordt gekeken naar de verschillende technieken die er al bestaan om dynamische objecten te herkennen. Het doel van deze periode is: een techniek te kiezen voor het herkennen van kaarten.

    Onderzoek tijd: 1 weken
  - Periode 4.2
    Start implementeren van techniek zoals gekozen in periode 4.1. Het idee is dat bij deze periode één kaart goed herkend zal worden. Hier wordt nog geen rekening gehouden met rotatie, kleur en schaling.

    Ontwikkelingstijd: 2 weken
  - Periode 4.3
    Vervolg implementeren periode 4.2. Functionaliteit wordt toegevoegd om rekening te houden met rotatie. Verder zal het systeem 2 kaarten moeten herkennen.

    Ontwikkelingstijd: 2 weken
  - Periode 4.4
    Zoals 4.3 met de volgende uitbreiding: herkenning van de kaarten ongeacht de afstand tot de lens en implementeren van de dynamische eigenschappen.

    Ontwikkelingstijd: 2 weken
  - Periode 4.5
    Het systeem moet nu dynamisch kaarten leren en markeren.

    Ontwikkelingstijd: 2 weken

Uit de voorgaande stuk is de volgende planning af te leiden:

Week	**Activiteit(en)**
36	Intro dagen / Plan van Aanpak Kickoff project meeting (Eindhoven)
37-38	Afronden Plan van Aanpak Fase: *Acquisition*
39-40	Fase: *Enhancement*
41-43	Fase: *Segmentation* Afspraak maken bedrijfsbezoek Verslag maken bedrijfsbezoek
44-46	Fase: *Feature extraction / Classification* Periode 4.1 Voortgangsverslag
46-48	Fase: *Feature extraction / Classification* Periode 4.2
49-50	Fase: *Feature extraction / Classification* Periode 4.3
51-52	Fase: *Feature extraction / Classification* Periode 4.4
53-1	Fase: *Feature extraction / Classification* Periode 4.5 Afronding / inleveren verslag
2	Afronding / Overdracht bij Alten

3.3. Risicoanalyse

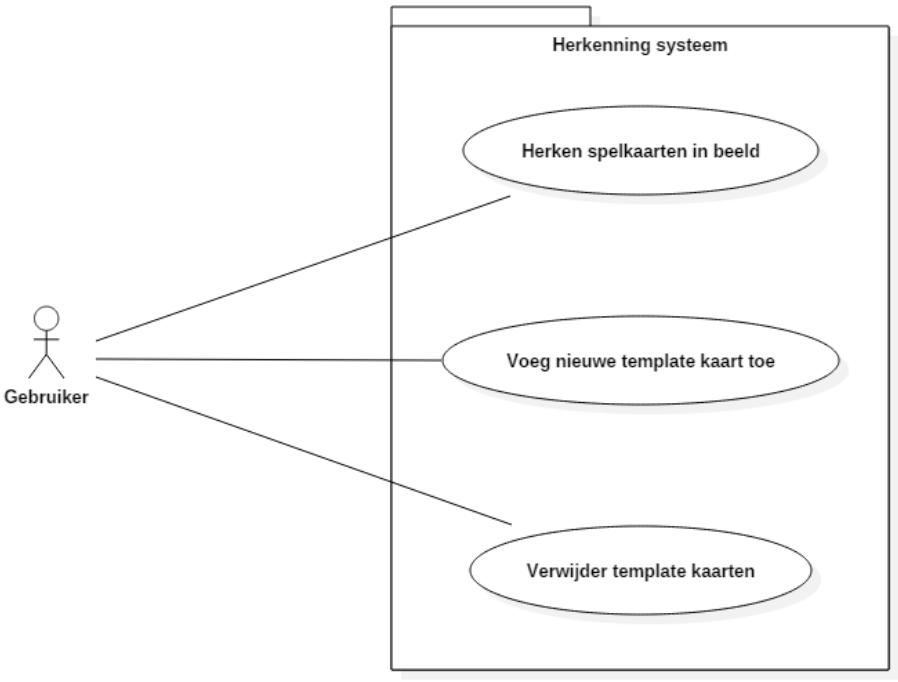
Ook binnen dit project zijn er risico's. Om die reden is er vooraf een lijst van risico's opgesteld. Door vooraf rekening te houden met de bekende risico's kunnen er op voorhand maatregelen genomen worden. In onderstaande tabel staan risico's opgesteld voor dit project.

Prioriteit	Risico omschrijving	Kans	Impact	Maatregel
200	Student heeft onvoldoende kennis van Computer Vision, waardoor project langer duurt dan verwacht	20%	10	Door aan het begin van het project een planning te maken kan bij het ontstaan van een probleem tijdig worden gereageerd. Er kan dan gekozen worden voor het gebruiken van een andere techniek of andere libraries. Het is belangrijk bij ieder eindoverleg van een periode een duidelijk inzicht te krijgen van de vorderingen. Hierdoor kan het risico van vertraging geminimaliseerd worden.
125	Het gebruik van bestaande libraries voor de communicatie van functionaliteiten in het system (buiten Computer Vision) werken niet	25%	5	Mede door een planning te maken, kan worden gekeken of er niet te veel tijd wordt besteed aan het uitzoeken van bestaande libraries. Door hierop tijdig te reageren kan worden gekeken naar andere oplossingen.

Tabel 2 - Risico's

4. Uitvoering

Het systeem op zich is relatief eenvoudig uit te beelden in één enkele use-case, zie Figuur 5. De use-case is gemaakt om zo de functionele eisen van het systeem vast te leggen, uit het oogpunt van de gebruiker. Voor iedere use-case is een omschrijving gemaakt deze omschrijvingen zijn te zien in Tabel 3, Tabel 4 en Tabel 5.



Figuur 5 - Use-Case Systeem

Herken speelkaarten in beeld	
Actoren	Gebruiker
Aannamen	Gebruiker heeft enige technische kennis voor het starten van proces op systeem.
Beschrijving	Op het moment dat de gebruiker het proces start, zal het systeem continu foto's nemen. Na het nemen van een foto zal het systeem het beeld analyseren en de gevonden resultaten tonen op een beeldscherm.
Uitzonderingen	1. Het object in beeld is niet bekend binnen het systeem. Het systeem zal het object negeren. 2. Het systeem herkend de kaart niet. Het systeem zal bij de volgende cyclus opnieuw proberen de kaart te herkennen. De gebruiker kan eventueel de kaart verplaatsen. 3. Het systeem herkend de verkeerde kaart. Het systeem zal bij de volgende cyclus proberen om de kaart toch goed te herkennen. De gebruiker kan eventueel de kaart verplaatsen.
Resultaat	Na iedere cyclus zal informatie gegeven worden van de eventuele objecten in beeld.

Tabel 3 - Use-Case Omschrijving - Herken Speelkaarten in beeld

Voeg nieuwe template kaart toe	
Actoren	Gebruiker
Aannamen	Gebruiker heeft enige technische kennis voor het starten van proces op systeem.
Beschrijving	Op het moment dat de gebruiker het proces start, zal het systeem eenmalig een foto nemen. Na het nemen van een foto zal het systeem het beeld analyseren en de template toevoegen aan het systeem.
Uitzonderingen	1. Het systeem herkend de kaart niet. De gebruiker kan eventueel de kaart verplaatsen en het proces opnieuw starten. 2. De kaart is reeds bekend binnen het systeem. Het systeem zal de kaart negeren. 3. Er liggen meerdere kaarten in beeld. Het systeem zal alle kaarten negeren.
Resultaat	Een nieuwe template wordt toegevoegd aan het systeem.

Tabel 4 - Use-Case Omschrijving - Voeg nieuwe template kaart toe

Verwijder template kaarten	
Actoren	Gebruiker
Aannamen	Gebruiker heeft enige technische kennis voor het starten van proces op systeem.
Beschrijving	Op het moment dat de gebruiker het proces start, zal het systeem alle templates verwijderen.
Uitzonderingen	1. Het systeem heeft geen templates geladen. Het proces wordt afgesloten.
Resultaat	Alle templates zijn verwijderd.

Tabel 5 - Use-Case Omschrijving - Verwijder template kaarten

4.1. Onderzoek libraries

Voor het detecteren van de verschillende kaarten is gekeken naar verschillende oplossingen om de omgeving in te richten. De oplossingen waarnaar onder andere gekeken zijn, zijn Computer Vision libraries. Voor de verschillende libraries is gelet op de omgeving waar het op moet draaien, zowel de documentatie van de library en als die van de community.

LibCCV

CCV is een minimale implementatie van een Computer Vision algoritme. CCV werkt op meerdere platforms, namelijk Raspberry Pi, Linux, Mac OSX en Windows. CCV in ontwikkeld in C. De documentatie en voorbeeld code is summier. Daarnaast heeft deze library geen implementatie van de acquisitie fase. Er kan gebruik worden gemaakt van eerdere ontwikkelingen.

SimpleCV

SimpleCV is een open source framework gebouw voor Computer Vision Applicaties. Met de mogelijkheid gebruik te maken van meerdere high-powered computer vision libraries zoals OpenCV. SimpleCV is ontwikkeld in Python. De documentatie en tutorials van SimpleCV zijn uitgebreider dan LibCCV, maar voor onder andere de feature extraction en classification mist er nog documentatie. De code kan draaien op zowel Linux, Windows en een Raspberry Pi.

Scikit-image

Scikit-image is een collectie van image processing algoritmen. Scikit bied een uitgebreide mogelijkehid aan image processing routines, maar biedt geen implementatie voor een camera module. Scikit is ontwikkeld in Python. De code werkt zowel op Windows, Linux en Raspberry Pi.

OpenCV

OpenCV is een algemeen bekend Computer Vision library. OpenCV is ontwikkeld in C/C++ en heeft een Python en Java interface. OpenCV kan gebruikt worden op zowel Windows als Linux platforms, OpenCV is ook beschikbaar op de Raspberry Pi. Ook heeft OpenCV, in tegenstelling tot andere libraries, een uitgebreide community; de documentatie is relatief uitgebreid en er zijn veel voorbeelden te vinden op internet.

Conclusie

De uiteindelijke keuze van de library is OpenCV. De reden voor deze keuze is omdat zoals eerder genoemd er een grote community achter zit en veel voorbeelden te vinden zijn. In Tabel 6 is een overzicht gegeven van de verschillende libraries.

	LibCCV[2]	**SimpleCV**[3]	**scikit-image**[4]	**OpenCV**[5]
Platform	Raspberry Pi, Linux, Windows, Mac OSX	Raspberry Pi, Linux, Windows, Mac OSX	Raspberry Pi, Linux, Windows, Mac OSX	Raspberry Pi, Linux, Windows, Mac OSX
Documentatie	-	+	- / +	+ +
Voorbeeld code	-	+	+	+ +
Tracking Impl.	?	+ +	-	+ + +
Ondersteuning Raspberry Pi Camera	-	+ [18]	- / ? [1]	+ + [16]
Community	-	- / +	- / +	+ + +
Programmeertaal	C	Python	Python	C++
Interfaces	-	-	-	Python, Java

Tabel 6 - Overzicht vergelijking CV libraries

4.2. Acquisition

Bij de aquisition komt belichten en de cameraopstelling aan bot. Naast de fysieke aspecten van het systeem wordt bij de acquisitie ook naar het inladen van de afbeeldingen in de library gekeken. Tijdens het onderzoek is gebleken dat het systeem ook een “server” applicatie nodig heeft voor het tonen van de beelden op een computerscherm. Dat zal allemaal verder in dit hoofdstuk omschreven worden.

Opstelling

Voor het nemen van de afbeeldingen is een opstelling nodig. Tijdens de ontwikkeling van de opstelling moeten rekening mee gehouden worden dat relatief eenvoudig extra verlichting geplaats kan worden. Ook moet rekening gehouden worden dat de camera verder of dichterbij het object geplaatst kan worden.

[2] http://libccv.org/
[3] http://simplecv.org/
[4] http://scikit-image.org/
[5] http://opencv.org/

Om die reden is er tijdens de ontwikkeling gekozen voor het gebruik van aluminium profielen. Aan de profielen kan relatief eenvoudig verlichting geplaatst worden. Daarnaast kan de opstelling eenvoudig worden aangepast door de hoogte te veranderen. In Figuur 6 is te zien hoe de uiteindelijke opstelling is gebouwd.

Figuur 6 - Opstelling

Afbeelding laden

Omdat er tijdens de ontwikkeling van het project gebruikt gemaakt zal worden van OpenCV, is voor het binnen halen van de afbeeldingen in het systeem een library nodig die kan samen werken met OpenCV. De library moet geschreven zijn in C/C++. Om die reden is er gekozen voor: RaspiCam. Er is onderzoek gedaan naar meerdere libraries, maar met de requirements van OpenCV en C/C++ is alleen de RaspiCam library beschikbaar. RaspiCam bied ook ondersteuning voor de OpenCV afbeeldingdatastructuur (cv::Mat).

Server Applicatie

Tijdens het onderzoek naar het laden van de afbeelding binnen het systeem, is gebleken dat er in de requirements een ondersteuning mist voor het tonen van de afbeeldingen. Om die reden is er gekeken naar mogelijke oplossingen voor het tonen van de afbeeldingen op een “server”. Het is van belang om te weten dat alle Vision operaties nog steeds uitgevoerd zullen worden op de Raspberry Pi. De “server” applicatie is alleen nodig voor het tonen van de afbeeldingen.

Voor het versturen van afbeeldingen van de Raspberry Pi naar de server zijn er verschillende oplossingen mogelijk. Tijdens het onderzoek naar verschillende mogelijkheden zal voornamelijk gelet worden op de snelheid, er zal geen rekening gehouden met beveiliging.

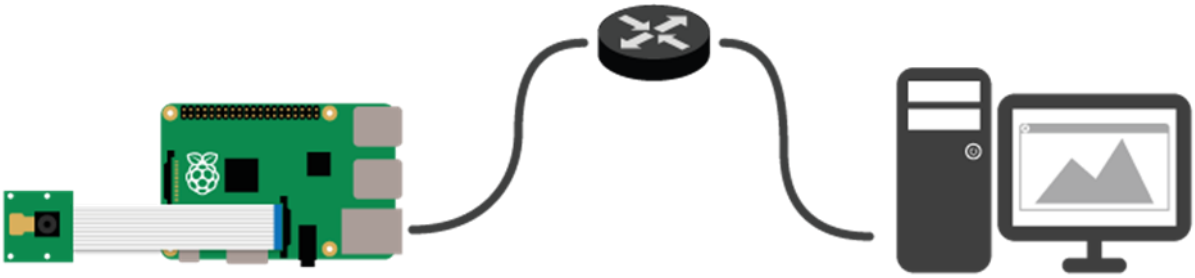
De mogelijke oplossingen die onderzocht zijn voor het gebruik zijn:

- POST via cURL[6]
- Socket library

[6] https://curl.haxx.se/libcurl/c/multi-post.html

Voor de POST oplossing is een server nodig die het POST bericht ontvangt en verwerkt. Vervolgens moet de connectie gemaakt worden met een webbrowser om de beelden te tonen aan de gebruiker. Door het gebruik van deze methode wordt veel tijd verspeeld aan het opzetten van connecties en het onderhouden van connecties. Daarnaast is het mogelijk, maar dat vraagt om meer ontwikkeltijd, om de geposte afbeelding door te sturen aan de browser. Om die reden is er besloten om naar een andere oplossing te kijken.

De beste optie voor het versturen van de afbeeldingen is via een Socket library. De Socket library geef de minste vertraging bij het versturen van de afbeelding. Met de Socket library wordt de Raspberry Pi binnen hetzelfde subnet geplaatst, maar in theorie zou dit ook op een willekeurig publiek IP-adres geplaats kunnen worden. Omdat het niet binnen de scope van het project noodzakelijk is, zal er geen rekening gehouden met security.



Figuur 7 - Netwerk opstelling

4.3. Enhancement

De Enhancement laag komt na de Acquisition laag. Bij de Enhancement wordt eventuele ruis, die ontstaat bij het nemen van de foto, verwijderd. Ook het verbeteren van de belichting kan hier worden gecorrigeerd, hoewel dit geen toegevoegde waarde heeft voor het herkennen van de objecten in beeld.

Bij dit project zijn geen enhancement's nodig geweest op het beeldmateriaal. De afbeeldingen zijn voldoende belicht en bevatten geen ruis. Een enhancement die wel uitgevoerd is op het detecteren van de verschillende kaarten, is het verkleinen van de afbeelding. Door het verkleinen van de afbeelding kan veel sneller een kaart gedetecteerd worden. Na het detecteren wordt aan de hand van de verhoudingen de juiste positie van de kaart bepaald in de grote afbeelding. Hiermee wordt een grote snelheidswinst behaald. In Tabel 7 is de snelheidswinst weergegevens.

	Labeling (No Resize)	*Labeling (Resize)*	*Canny (No Resize)*	*Canny (Resize)*
Gemiddelde duur	3624,1 ms	372,4 ms	217,6 ms	152,4 ms
Verschil	3251,7 ms		65,2 ms	

Tabel 7 - Snelheidswinst

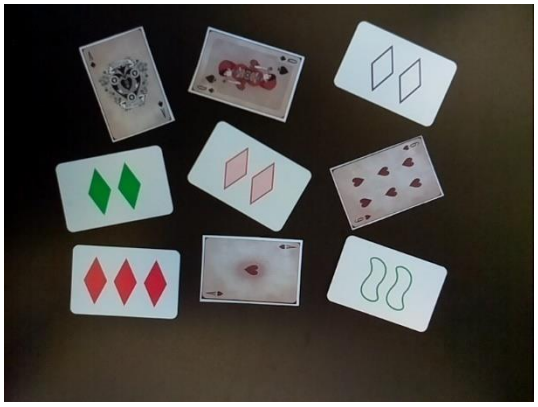
4.4. Segmentation

De Segmentation laag is verantwoordelijk voor het opdelen van de afbeelding in kleinere afbeeldingen. Iedere afbeelding afzonderlijk is een segment. Met de segmenten kan dan afzonderlijk gekeken worden naar een klein onderdeel van de afbeelding, om zo tijd te besparen met het onderzoeken van de verschillende kaarten in de Feature Extraction / Classification laag. Het doel is op een gerichte manier te zoeken naar de informatie in de afbeelding.

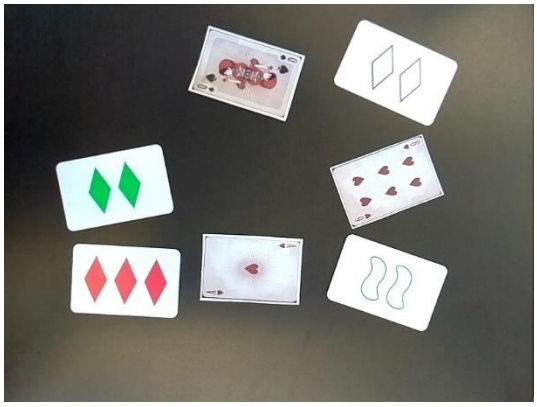
Voor het segmenteren van de afbeelding zijn twee mogelijke heden onderzocht namelijk:

- Canny Edge Detection
- Labeling

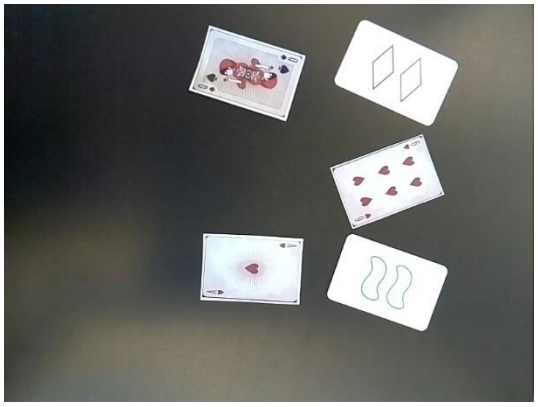
Voor beide functies is onderzoek gedaan naar de betrouwbaarheid van het segmenteren zowel als naar de snelheid waarmee. Tijdens de test is nog geen snelheid enhancement toegepast zoals omschreven in 4.3. De reden hiervoor is dat dan duidelijk te zien is welk algoritme beter presteert. In de test is onder andere onderzoek gedaan met één kaart, meerdere kaarten en andere type objecten in het gezichtsveld. In Tabel 8 is een overzicht te zien.



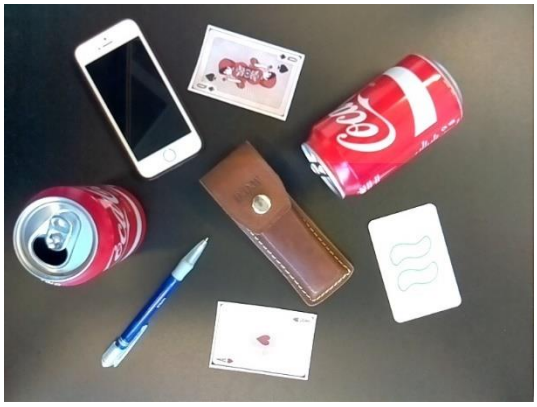
Figuur 8 - Segmentation 9 kaarten



Figuur 9 - Segmentation 7 kaarten



Figuur 10 - Segmentation 5 kaarten

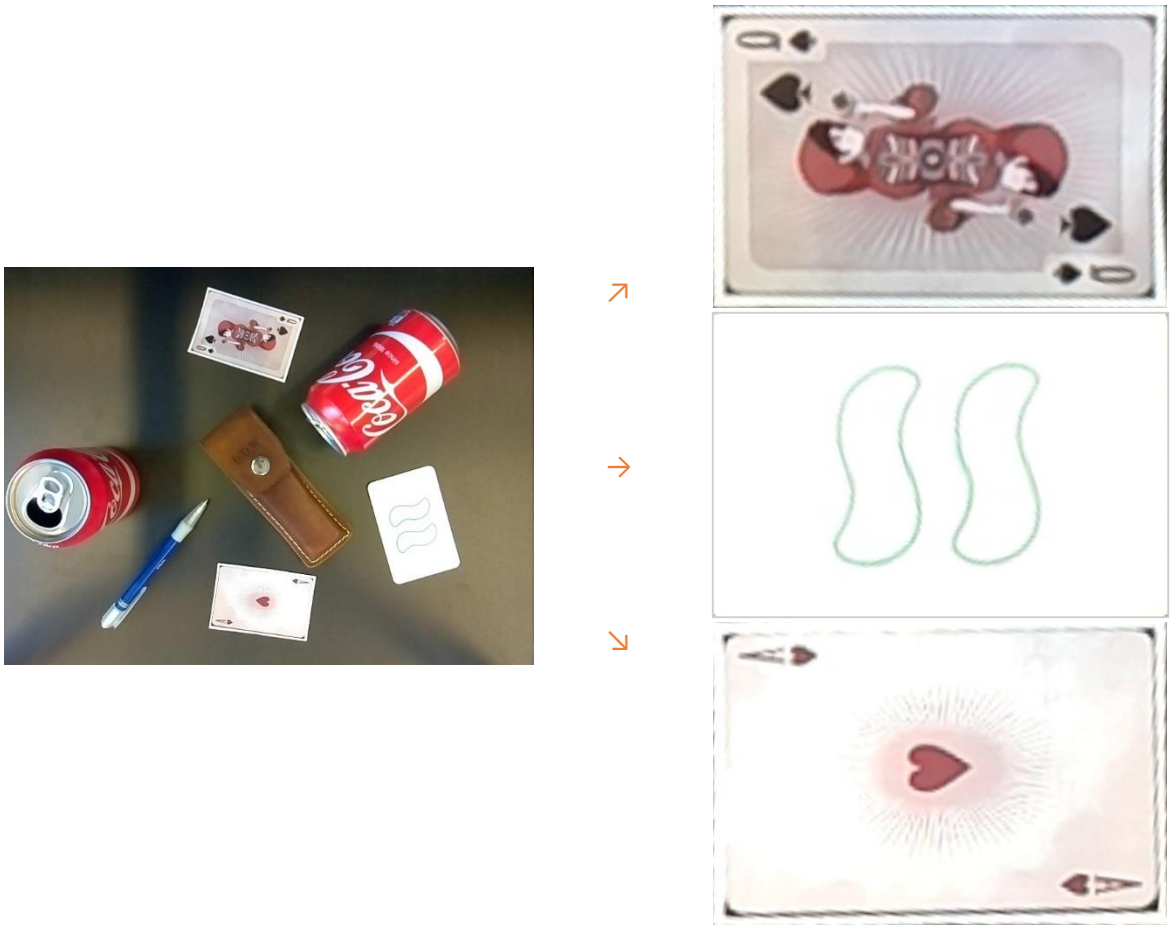


Figuur 11 - Segmentation 3 kaarten

	Figuur 8		*Figuur 9*		*Figuur 10*		*Figuur 11*	
	Canny	Label	Canny	Label	Canny	Label	Canny	Label
Aantal kaarten	9		7		5		3	
Andere Objecten	0		0		0		5	
Juiste objecten herkend	✓	✓	✓	✓	✓	✓	✓	✗
Gem. Duur herkenning (ms)	**311**	8375	290	6659	250	5934	225	4534

Tabel 8 - Segmentation Snelheidswinst

Uit het segmentatie proces komen de onderstaande gesegmenteerde afbeeldingen die vervolgens gebruikt kunnen worden bij de Feature Extraction / Classification.



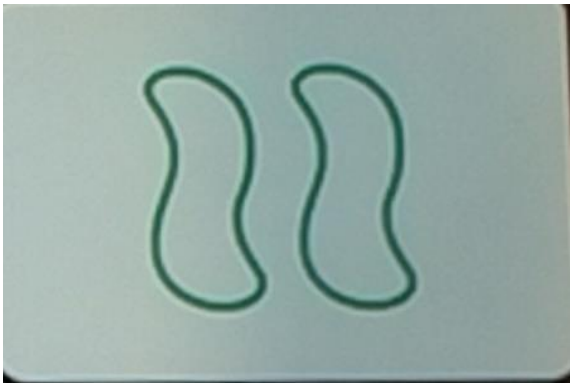
Figuur 12 - Resultaat Canny Edge Detection

4.5. Feature Extraction / Classification

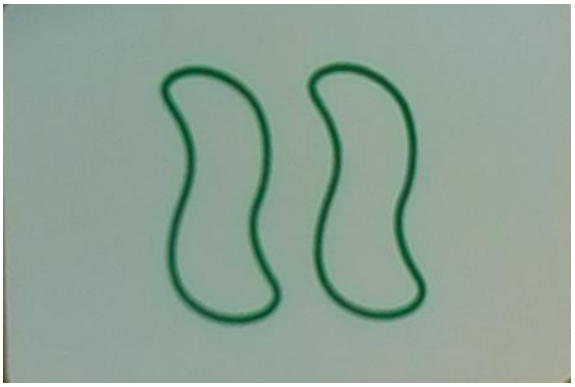
HoG (Histograms of Oriented Gradients) of TLD

Er is onderzoek gedaan naar TLD of HoG algoritme. Uit het onderzoek naar TLD is gebleken dat er een implementatie is namelijk: OpenTLD. OpenTLD is deels ontwikkeld in OpenCV, maar de herkenning is ontwikkeld in Matlab. Zoals omschreven in hoofdstuk 2 is het vermoeden dat TLD ongeveer dezelfde technieken gebruikt. Net zoals bij HoG kan het detectie algoritme van TLD slecht omgaan met rotatie en schaling, maar door het learning algoritme kan het systeem zichzelf verbeteren. Omdat OpenTLD matig tot slecht van commentaar voorzien is, is er besloten niet verder te gaan met het volledig implementeren van OpenTLD in OpenCV.

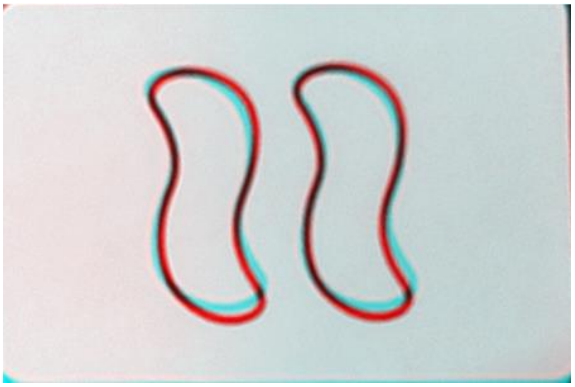
Voor het gebruiken van het HoG algoritme is een eenvoudige test gedaan. Er zijn twee foto's van dezelfde kaart gemaakt. Iedere foto is onder een iets andere hoek genomen. Zoals omschreven in het vorige hoofdstu,k wordt vervolgens de kaart gewarpt en in een vast formaat geplaatst. In onderstaand voorbeeld is te zien in Figuur 13 en 14 dat het verschil minimaal is, maar het blijkt dat HoG slecht om kan gaan met kleine rotaties of schalings verschillen, zoals te zien in Figuur 15.



Figuur 13 - SET Golf 2 Leeg (Template 1)



Figuur 14 - SET Golf 2 Leeg (Template 2)

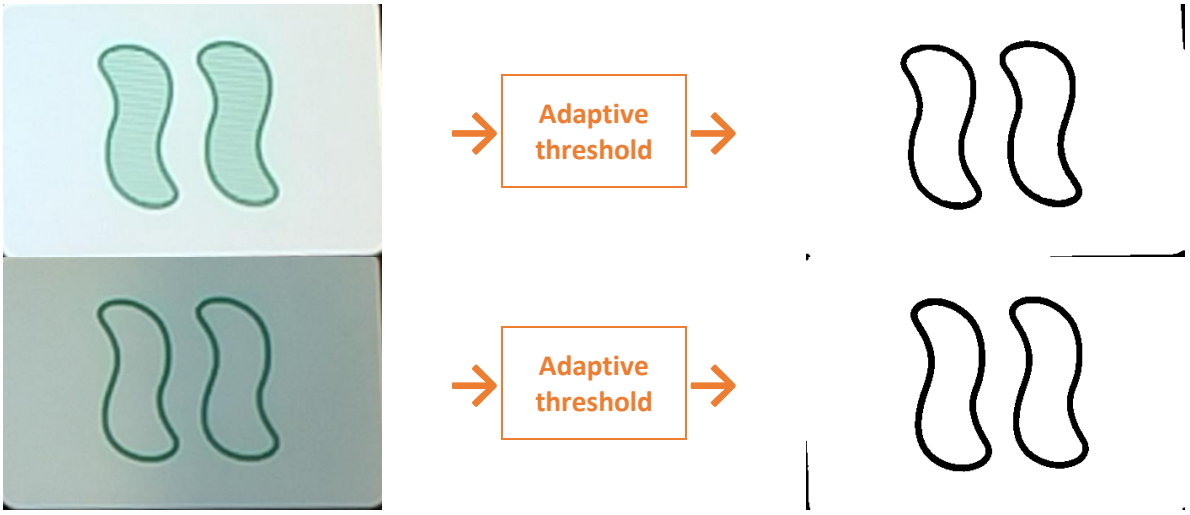


Figuur 15 - Mismatch HoG

Het probleem met de rotatie kan opgelost worden door gebruik te maken van een SVM-systeem (Support Vector Machine). Het SVM-systeem wordt vaak toegepast bij machine learning. Zoals te zien in het onderzoek “Histograms of Oriented Gradients for Human Detection”[19] waar zij het met het HoG algoritme mogelijk gemaakt hebben om mensen te tracken. Opvallend aan het systeem is dat mensen nooit in een vaste positie staan en daarom zal in eerste instantie het HoG algoritme niet geschikt zijn, maar door het gebruik van een dataset van 1800 verschillende afbeeldingen van mensen is het toch mogelijk. Het nadeel van HoG is dus dat er een relatief grote learning curve voor nodig is. Mede door te weinig kennis van de student op het gebied van SVM-systemen is er voor gekozen niet verder te kijken naar de HoG algoritme.

Absolute Difference comparision

Net zoals bij het HoG algoritme kan de Absdiff ook slecht om gaan met rotaties. Het systeem is met dezelfde voorbeeld kaart wel instaat een match te vinden. Dit komt omdat in vergelijking met andere template kaarten hier de meeste overlap in zit. Er ontstaat echter een probleem bij het herkennen van dezelfde kaart maar dan met een halve vulling, zie Figuur 16. Omdat de foto’s van de Raspberry Pi camera niet helemaal scherp zijn, wordt door het uitvoeren van een adaptive threshold de “Hassed” vulling uit gefilterd, door het toevoegen van deze template aan het systeem wordt het systeem onnauwkeuriger, er worden vaker mismatches herkend. Het systeem laat veel achtergrondinformatie vallen door het uitvoeren van de threshold.



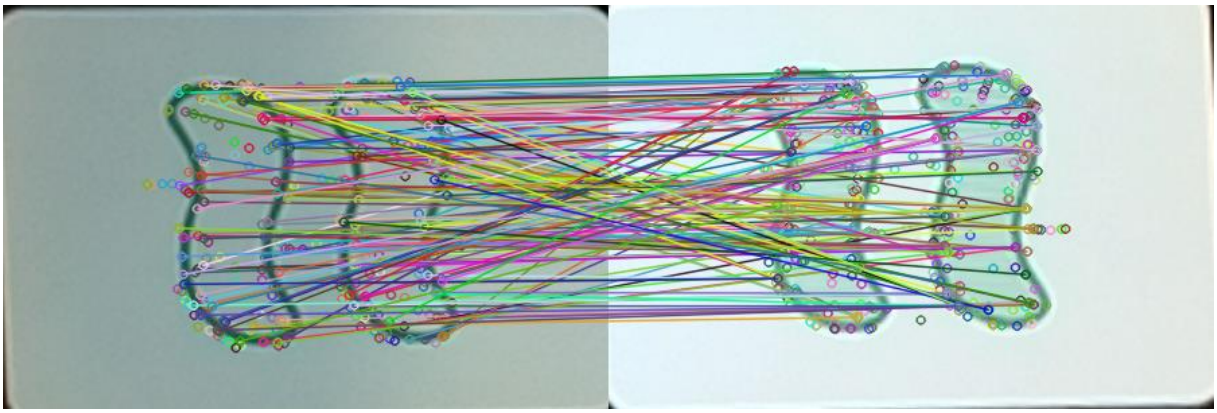
Figuur 16 - Probleem Abs Diff

Vanwege de onnauwkeurigheid van het algoritme, door het gebruik van de adaptive threshold waardoor veel informatie verloren gaat, is ervoor gekozen het Absdiff algoritme niet te gebruiken.

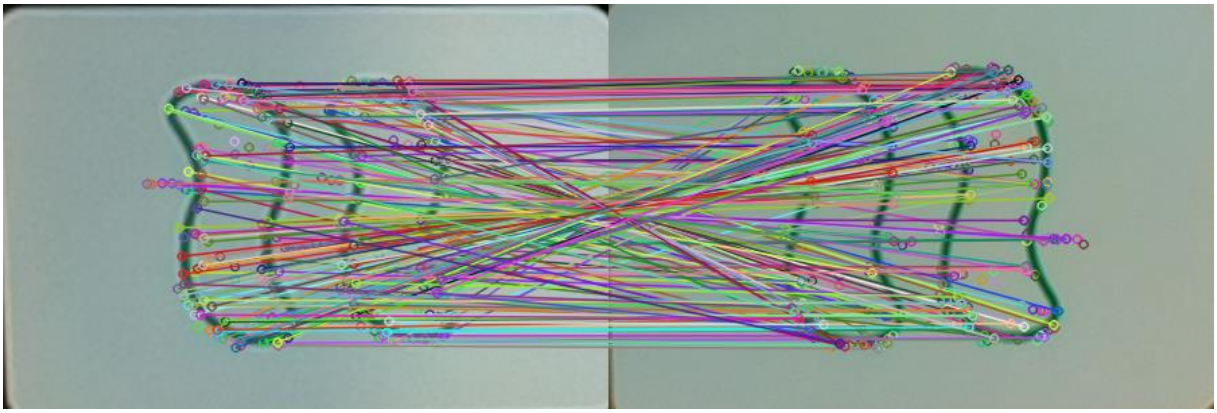
SURF (Speeded Up Robust Features)

Het voordeel van het SURF-algoritme in vergelijking met de HoG of Absdiff is dat dit algoritme wel goed omgaat met rotatie en schaling. Het SURF-algoritme zoekt in de afbeeldingen keypoints waaraan de template en de kaart vergeleken wordt. Bij de Absdiff is naar voren gekomen dat het algoritme belangrijke informatie verwijdert, waardoor het algoritme onbetrouwbaar wordt. Bij het SURF-algoritme is te zien dat dit niet gebeurt, in Figuur 17 en 18 is te zien dat de juiste kaart aanzienlijk meer matches bevat dan de verkeerde template. Om die reden is besloten om het SURF-algoritme te gebruiken.

De testen zijn uiteraard uitgevoerd op meerdere kaarten. Het volledige resultaat van de verschillende kaarten zal in het volgende hoofdstuk omschreven worden voor het SURF-algoritme.



Figuur 17 - SET Golf 2 Leeg <=> SET Golf 2 Halfvol (182 matches)

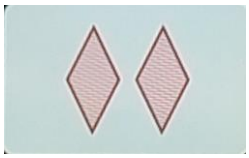
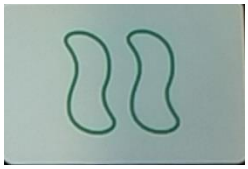


Figuur 18 - SET Golf 2 Leeg <=> SET Golf 2 Leeg (260 matches)

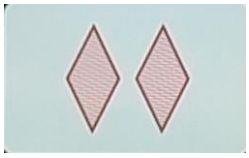
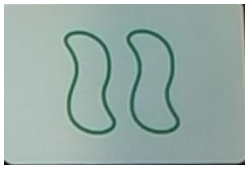
5. Resultaat

In de onderstaande tabellen staan de resultaten van diversen technieken. Gemeten is naar de weging van de template afbeelding (3) in verhouding met het aantal matches (2), waar vervolgens een nauwkeurigheid bepaald wordt door het systeem (4). Vervolgens is er gecontroleerd of de match overeenkomt de werkelijkheid (5).

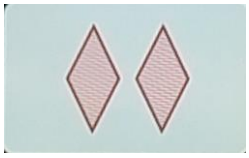
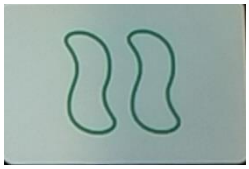
Kolom 6, 7 en 8 geven aan welke verkeerde match gemaakt is (6) en dat het algoritme, gezien de hogere nauwkeurigheid (8), bepaalt door de meerdere matches (7) ten opzichte van (2), een andere match aangeeft (6) dan de kaart (1).

Kaart (1)	Aantal Matches (2)	Totaal Template KeyPoints (3)	Nauwkeurigheid (4)	Correct Match (5)	Verkeerde Match (6)	Aantal Mismatches (7)	Nauwkeurigheid (8)
	416	1031	40%	✓			
	12	18	67%	✓			
	123	324	38%	✗		214	66%
	37	40	93%	✓			
	440	706	62%	✓			
	250	303	82%	✓			

Tabel 9 - Resultaat SURF

Kaart	Aantal Matches	Totaal Template Points	Nauwkeurigheid	Correct Match	Verkeerde Match	Aantal Mismatches	Nauwkeurigheid
	0	3744	0%	✗			
	0	3744	0%	✗			
	0	3744	0%	✗			
	0	3744	0%	✗			
	0	3744	0%	✗			
	0	3744	0%	✗			

Tabel 10 - Resultaat HoG

Kaart	Gewicht (lager is beter)	Totaal Mogelijke Gewicht	Nauwkeurigheid	Correct Match	Verkeerde Match	Gewicht (lager is beter)	Nauwkeurigheid
	24629	135000	82%	✓			
	2945	135000	98%	✓			
	20093	135000	85%	✗		10180	92%
	6975	135000	95%	✓			
	17867	135000	87%	✓			
	9844	135000	93%	✗		7735	94%

Tabel 11 - Resultaat Abs Diff

Uit de bovenstaande tabellen blijkt dat bij willekeurige invoer van kaarten het SURF-algoritme het beste naar voren komt.

6. Conclusie

Uit het onderzoek en de resultaten blijkt dat het HoG-algoritme niet direct inzetbaar is voor deze toepassing. Dit heeft te maken met het feit dat HoG niet zelfstandig kan functioneren, het heeft geen learning algoritme wat van essentieel belang is voor een dynamisch systeem.

Het Abs Diff-algoritme is verreweg het snelste waardoor hij minder resourses gebruikt, met als gevolg een slechtere kans op matches, mede door het feit dat hij essentiële informatie verwijdert uit de afbeelding.

Uit de resultaten blijkt dat het SURF-algoritme de grootste kans op matches oplevert, ondanks het feit dat hij langzamer is en meer rekenkracht nodig heeft. Dit laatste komt omdat hij vaker door de afbeelding heen gaat. Positieve bijkomstigheid ten opzichte van de andere algoritmes is dat SURF invariant is met betrekking tot schaling, vervorming en rotatie.

Om een grotere kans op matches te verkrijgen is het wenselijk om te kijken naar colour segmenation. In de mij toegewezen tijd is ben ik hier helaas niet aan toegekomen en wil ik dit adviseren voor nader onderzoek.

Gezien de grote kans op matches, is het voor nadere onderzoek nuttig om te kijken of er een versnelling mogelijk is bij het surf-algoritme zonder het aantal templates te verminderen.

Bij de segmantation fase in het onderzoek kunnen we concluderen dat Canny-Edge detection sneller resultaat levert dan de traditionele label techniek. Verder filtert de techniek alle overbodige objecten uit de afbeelding en plaats de kaart in een vast formaat dat voordelen biedt bij de feature extraction / classification.

7. Bibliografie

[1] *Accessing the Raspberry Pi Camera with OpenCV and Python*. (2015, Maart 30). Opgehaald van Adrian Rosebrock: http://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/

[2] al., G. B. (2015). Learning Image Processing with OpenCV. Packt Publishing.

[3] al., G. B. (2015). Learning Image Processing with OpenCV. Packt Publishing.

[4] Bay, H., Tuytelaars, T., & van Gool, L. (2006). SURF: Speed Up Robust Features. In *Computer Vision – ECCV 2006* (pp. 404-417). Springer Berlin Heidelberg.

[5] Baya, H., Essa, A., Tuytelaarsb, T., & Van Goola, L. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 346-359.

[6] Elfring, J. (2013, October 9). *Image Processing Using OpenCV*. Opgehaald van http://cstwiki.wtb.tue.nl/images/06-opencv.pdf

[7] *Histogram Comparison - OpenCV 2.4.12.0 documentation*. (sd). Opgehaald van http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html

[8] *Histogram of Oriented Gradients (HOG) Descriptor*. (sd). Opgehaald van https://software.intel.com/en-us/node/529070

[9] *How to match 2 HOG for object detection? - OpenCV Q&A Forum*. (2012, Julie 28). Opgehaald van http://answers.opencv.org/question/877/how-to-match-2-hog-for-object-detection/?answer=882#post-id-882

[10] *Image Filtering - OpenCV 2.4.12.0 documentation*. (sd). Opgehaald van http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#bilateralfilter

[11] Kool, W. (sd). *Automation of the SET Card Game*. Opgehaald van https://www.few.vu.nl/nl/Images/werkstuk-kool_tcm243-369064.pdf

[12] *Local Feature Detection and Extraction - MATLAB & Simulink - MathWorks Benelux*. (sd). Opgehaald van http://nl.mathworks.com/help/vision/ug/local-feature-detection-and-extraction.html#buk_bm9

[13] Nandi, A. (2013, Junie 16). *So I Suck At 24: Automating Card Games Using OpenCV and Python*. Opgehaald van http://arnab.org/blog/so-i-suck-24-automating-card-games-using-opencv-and-python

[14] Oyallon, E., & Rabin, J. (2015). An Analysis of the SURF method. *Image Processing On Line*, 176-218.

[15] *Ramer–Douglas–Peucker algorithm*. (sd). Opgehaald van https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm

[16] *RaspiCam: C++ API for using Raspberry camera with/without OpenCv*. (sd). Opgehaald van http://www.uco.es/investiga/grupos/ava/node/40

[17] Restoration in the Presence of Noise Only - Spatial Filtering. (2010). In R. Gonzalez, & R. Woods, *Digital Image Processing* (pp. 344-349). Pearson Education, Inc.

[18] *Simplecv + Camera board guide*. (sd). Opgehaald van https://www.raspberrypi.org/forums/viewtopic.php?t=57788

[19] Triggs, N. D. (sd). *Histograms of Oriented Gradients for Human Detection*. Opgehaald van http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

[20] Zdenek Kalal, K. M. (2010, Januari 1). *Tracking-Learning-Detection*. Opgehaald van http://vision.stanford.edu/teaching/cs231b_spring1415/papers/Kalal-PAMI.pdf