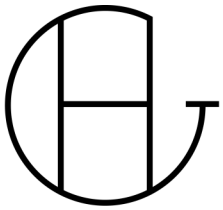


HoGent

Academiejaar 2012–2013

Geassocieerde faculteit Toegepaste Ingenieurswetenschappen
Valentin Vaerwyckweg 1 – 9000 Gent

Ontwikkeling van algoritmen voor rolnummerherkenning

Masterproef voorgedragen tot het behalen van het diploma van
Master in de industriële wetenschappen: informatica

Cédric VERSTRAETEN

Promotoren: *ing. Tim DE PAUW*
ir. Jeroen DEFOUR
ir. Jonas MAES

Woord vooraf

Mijn motivatie om voor dit onderwerp te kiezen, is vooral gekomen door het ouder worden en invloed van bepaalde belangrijke personen in mijn leven. Vroeger was ik vooral die speelse jongen, die veel dingen deed zonder er eerst bij na te denken en dit met de nodige gevolgen. Ik denk dat het grotendeels komt door het niet beter weten; ik vroeg mij bijvoorbeeld altijd af waarom ik wiskunde ooit nodig zou hebben. Het waren maar getalletjes die een ander getal maakten en daar stopte het dan ook.

Zoals elke jongen (van de jaren 90), vond ik de computer iets leuks; zonder mezelf hierbij veel vragen te stellen, gaf het al de spelletjes die ik leuk vond op het scherm weer. Achteraf vraag je je wel af hoe dat alles nu werkt. Op het eerste gezicht was er niet echt iemand die mij hierop een duidelijk antwoord kon geven. Bij het afsluiten van mijn secundair onderwijs, begon ik aan de opleiding Toegepaste Informatica aan het Departement Bedrijfskunde. Hier leerde ik wel wat ik wou, maar toch vroeg ik me nog steeds af waarom die wiskunde nodig was. Tot hier toe had ik nog steeds geen duidelijk antwoord gekregen, of tenminste niet een antwoord dat voor mij duidelijk genoeg was.

Je wordt ouder en je studeert af, je hebt de smaak te pakken maar je wilt toch nog net dat ietsje meer. Daarom koos ik voor het schakeljaar naar Toegepaste Ingenieurswetenschappen: Informatica. Opeens werd alles duidelijk; ik kreeg antwoorden op veel vragen, ik kon gemakkelijker verklaren waarom bepaalde dingen zijn zoals ze zijn. Hierdoor ben ik wiskunde meer gaan appreciëren, door mijn motivatie en interesse lukte het wel. Ondanks dit had ik nog steeds geen duidelijk beeld waar informatica en wiskunde nu precies te samen kwamen en dit op een zeer expliciete manier.

Toen ik te horen kreeg over het vakgebied Beeldverwerking was ik op slag verliefd. Dit is dan ook de hoofdreden waarom ik enorm gemotiveerd voor deze masterproef heb gekozen. Een tweede reden waarom ik bij het bedrijf ArcelorMittal naar deze scriptie heb gezocht, komt doordat ik er drie jaar terug, een vakantiejob als verpakker heb uitgevoerd. Dit hield het inpakken van rollen staal in die vervolgens naar de verschillende klanten werden opgestuurd d.m.v. boten, treinen of vrachtwagens. Ik was vooral geïnteresseerd in de logica van de grote massieve kranen die deze rollen staal verplaatsten. Het leuke is dat ik nu een scriptie schrijf waar ikzelf als arbeider gewerkt heb en dus de werkplaats maar al te goed ken. Uiteraard handelt deze scriptie over meer dan alleen de kranen in de zone verpakking. De verschillende kranen die gebruikt worden zijn verspreid over de verschillende productiehallen en behoren tot verschillende productieprocessen.

Om af te sluiten wil ik enkele mensen bedanken die een grote invloed hebben gehad bij het uitwerken van deze masterproef. Mijn interne promotor Tim De Pauw, die mij niet alleen tijdens deze masterproef zeer intensief gesteund en geholpen heeft maar

ook tijdens het schakeljaar en masterjaar. Ik kon met veel problemen bij hem terecht en kreeg hier telkens een zeer duidelijk antwoord op. Voor mij is hij meer dan een vaste waarde binnen de vakgroep Informatica.

Hiernaast wil ik graag mijn externe promotoren Jonas Maes en in het bijzonder Jeroen Defour bedanken die mij telkens met hulp en daad bijstond. Jeroen heeft me tijdens deze masterproef zeer goed begeleid en opgevolgd. Hij bekeek de methoden die ik voorstelde kritisch en leerde me dat werken in een industriële omgeving de nodige voorzorgsmaatregelen vereist, maar dat het natuurlijk ook zijn charmes heeft.

En als laatste wil ik mijn vriendin Kelly bedanken, die soms uren lang naar mijn problemen kon luisteren en hierbij zelf ook nadacht met een algemene blik. Hierdoor bekeek ik ook vaak de zaken op een andere manier.

Cédric Verstraeten
Gent, juni 2013

Inleiding

In de productiehallen van ArcelorMittal Gent, onderdeel van de grootste staalfabrikant ter wereld, worden rollen staal verplaatst aan de hand van automatische en bemande kranen. Om de veiligheid te verhogen en de kosten te drukken, wordt een deel van deze bemande kranen geautomatiseerd. Helaas loopt het verplaatsen van deze rollen weleens fout: soms verplaatst een bemande kraan de verkeerde rol of worden de labels, die de rollen identificeren, verwisseld. Dit heeft als gevolg grote vertragingen, een hoger kostenplaatje en imagoschade voor het bedrijf.

Op de automatische kranen is een camera aanwezig, die beelden maakt van de labels op de rollen. De bedoeling van deze masterproef is om automatisch de nummers op deze labels uit te lezen, zodat het aantal fouten tot een minimum kan worden herleid. De belichting, het label en de handelingen van arbeiders en kranen vormen hierbij de voornaamste uitdagingen.

Het herkenningsalgoritme dat ontwikkeld wordt maakt gebruik van de (camera)beelden van de automatische kranen en wordt ontwikkeld m.b.v. een computervisiebibliotheek. Het is opgesplitst in twee delen: tekstdetectie en tekstherkenning. Voor de tekstdetectie wordt een algoritme voorgesteld dat het rolnummer, van de rol staal, kan situeren. Om het rolnummer te herkennen wordt *optical character recognition* toegepast. Hiervoor wordt gebruik gemaakt van de bibliotheek *OpenCV* en de OCR-engine *Tesseract*.

Het algoritme voor de situering van het label wordt afzonderlijk geëvalueerd; in 82% van de gevallen wordt het label correct gesitueerd. Daarnaast wordt het algoritme geïntegreerd in het bestaande systeem van ArcelorMittal. Aan de hand van een uitgebreide evaluatie wordt aangetoond dat, na situering van het label, het rolnummer met 64% zekerheid kan worden bepaald.

In hoofdstuk 1 wordt de bestaande structuur, die binnen ArcelorMittal wordt gebruikt, besproken. Hoofdstuk 2 behandelt de problemen die het herkennen van het rolnummer bemoeilijken. Vervolgens worden in hoofdstuk 3 enkele beeldverwerkingstechnieken verduidelijkt. In hoofdstukken 4 en 5 wordt het algoritme beschreven. Hierna wordt in hoofdstuk 6 het algoritme geïmplementeerd in een offline GUI, die gebruikt wordt om het algoritme te kalibreren. Hoofdstuk 7 behandelt de integratie van het algoritme in het bestaande systeem van ArcelorMittal. Tot slot worden in hoofdstuk 8 de resultaten van de integratie uitgebreid besproken.

Abstract

In de productiehallen van ArcelorMittal Gent worden rollen staal verplaatst aan de hand van automatische en bemande kranen. Helaas loopt het verplaatsen van deze rollen weleens fout: soms verplaatst een bemande kraan de verkeerde rol of worden de labels, die de rollen identificeren, verwisseld. Dit heeft als gevolg grote vertragingen, een hoger kostenplaatje en imagoschade voor het bedrijf.

Om deze problemen op te lossen, wordt een oplossing voorgesteld die bestaat in de situering van het rolnummer gevolgd door het uitlezen ervan. Er wordt aangetoond dat de situering in 82% van de gevallen een correct resultaat oplevert. Toepassing in het bestaande systeem van ArcelorMittal Gent leert dat tot 64% van de gesitueerde rolnummers correct worden uitgelezen, en dat de voorgestelde oplossing geschikt is voor gebruik in een realtimeomgeving. Tot slot worden enkele technieken besproken die het resultaat kunnen verbeteren.

Inhoudstafel

Lijst van figuren

Lijst van tabellen

1 Situering

In het eerste hoofdstuk worden het basisprobleem en de doelstelling van deze masterproef verduidelijkt. Hierbij worden enkele belangrijke begrippen besproken die essentieel zijn in de volgende hoofdstukken. Daarnaast wordt een kort overzicht gegeven van de bestaande infrastructuur die binnen ArcelorMittal wordt gebruikt.

1.1 ArcelorMittal Gent

ArcelorMittal Gent (AMG) is één van de meest geïntegreerde staalproducenten binnen de ArcelorMittal Groep. Elke stap van het productieproces vindt plaats op het terrein van AMG, van de aanvoer van grondstoffen tot het bekleden van staal. De Groep produceert jaarlijks 110 miljoen ton ruw staal, waarvan 5 miljoen in Gent. Deze grote massa vlakke staalplaat gaat naar autoproducenten en de industriële sector; ook toepassingen zoals huishoudapparaten en meubilair worden gemaakt van hun staal.

Sinds juni 2006 maakt AMG deel uit van ArcelorMittal, de grootste staalgroep ter wereld. Deze staalreus heeft vestigingen in 27 landen en meer dan 310.000 werknemers wereldwijd. [1]

1.2 Productieproces

Deze paragraaf beschrijft in grote lijnen het productieproces. [2]

Het staal doorloopt tijdens de productie een uitgebreid proces. Zo wordt tijdens het hoogovenproces, vloeibaar ruwijzer gevormd uit een combinatie van ijzererts en cokes. Vervolgens wordt het ruwijzer gezuiverd, gelegeerd, eventueel ontgast en continu gegoten in slabs. In de warmwalserij worden de opgewarmde slabs gewalst tot warmrollen, en komen in koudwalserij terecht. Hier worden de warmrollen gebeitst, koudge-

walst, gegloeid, geskind, geïnspecteerd en eventueel in platen geknipt. De warm- en koudgewalste rollen kunnen vervolgens verzinkt en geverfd worden. Daarnaast worden ook plakken gevormd (Figuur 1.1).

Figuur 1.1: Plakken (bron: [56]).

Het doel van ArcelorMittal is om uitstekend staal te maken: zuiver oppervlak, uitstekende vlakheid, soepele vervormbaarheid en rekbaarheid, en een superieure lasbaarheid.

1.3 Kranen

Wanneer een rol staal van de warmwals in de koudwals terechtkomt, wordt deze voorzien van een rolnummer en doorheen de verschillende hallen en productielijnen getransporteerd door middel van kranen (Figuur 1.2). Doordat de volledige stock geïnformatiseerd is, kan de plaats van een rol bepaald worden. Dit geïnformatiseerd systeem - verder besproken in volgende paragraaf - is fundamenteel voor de correcte werking van de verschillende kranen. Wanneer een rol naar een productielijn of stockplaats moet worden verplaatst, raadpleegt een kraan het geïnformatiseerd systeem om de positie van de rol te bepalen. Op het eerste gezicht lijken deze kranen vrij eenvoudig. Maar ze kunnen meer dan alleen een rol opnemen en neerplaatsen. Ze kunnen bijvoorbeeld ook roteren (360°). Dit roteren is iets wat we in de volgende hoofdstukken vaak zullen tegenkomen, en verdient dus de nodige aandacht.

Als de rollen staal, door één of meerdere verplaatsingen, tot hun verwachte eindbestemmingen zijn gekomen, worden ze op treinen, boten en vrachtwagens geplaatst om naar hun klanten te verzenden.

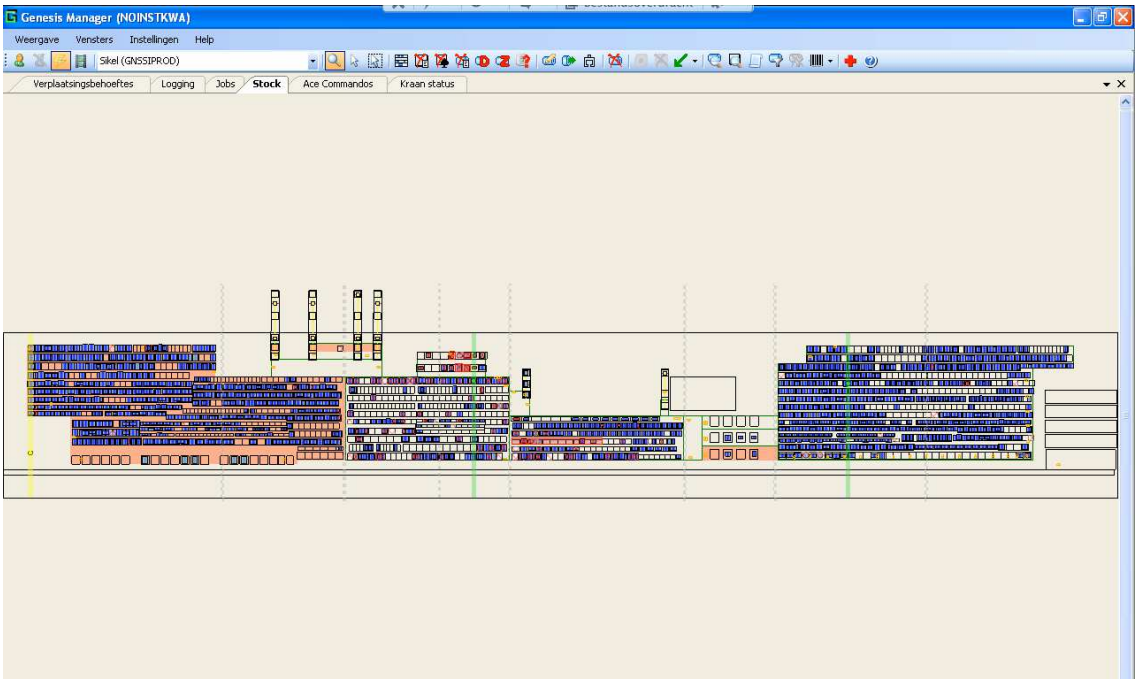
Deze kranen waren vroeger bemand en dus manueel bediend. Na verloop van tijd, door nieuwe technologieën, probeerde AMG om enkele zaken, waaronder de kranen, te automatiseren. Bij elke investering is het de bedoeling dat de gemaakte kosten vroeg of laat zullen renderen. In dit geval is dat het besparen van mankracht en minder strubbelingen zoals bijvoorbeeld rolverwisselingen (besproken in hoofdstuk 2).

Figuur 1.2: Kraan verplaatst rol staal (links) en bovenaanzicht met rolnummer (rechts).

Het doel is om de komende jaren steeds meer kranen te automatiseren. Maar het is onmogelijk om alle bemande kranen te vervangen door automatische kranen. Doordat sommige processen binnen ArcelorMittal zo complex of gevaarlijk zijn kunnen ze niet steeds door automatische kranen worden uitgevoerd.

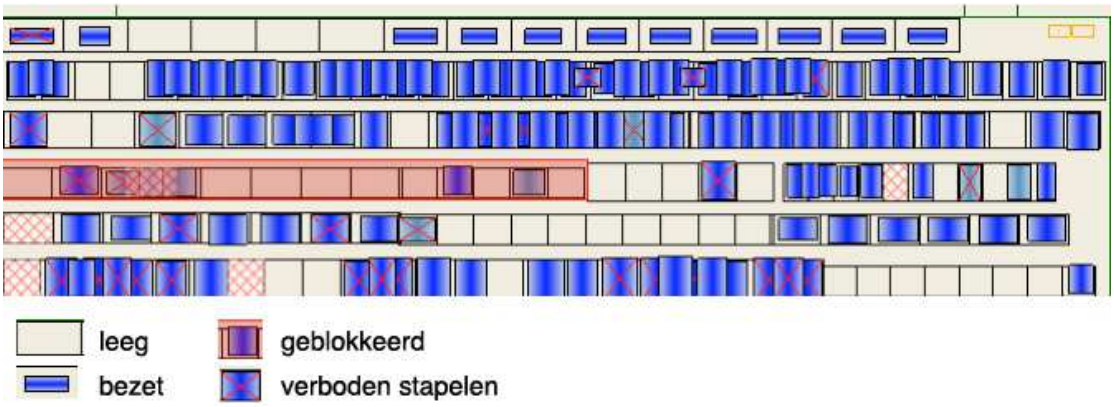
1.4 Het geïnformatiseerd systeem

Het geïnformatiseerd systeem (Figuur 1.3), waarvan de kranen gebruikmaken, geeft de volledige structuur van een productiehal weer. Het bevat alle verplaatsingsbehoeftes die door de kranen van de specifieke productiehal moeten worden uitgevoerd. Dit systeem, genaamd *Genesis*, is ontwikkeld door ArcelorMittal en is geschreven in de programmeertaal C#.



Figuur 1.3: Het geïnformatiseerd systeem.

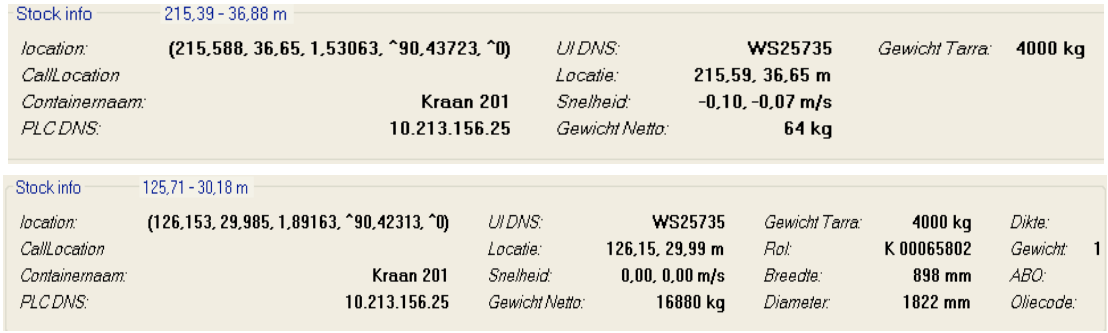
Figuur 1.4 geeft een gedeelte van een productiehal weer, zoals die zichtbaar is op de GUI van Genesis. Hierop worden de kranen, rollen staal en stockplaatsen in een realtime omgeving weergegeven. Deze realtime omgeving wordt mogelijk gemaakt door lasers of encoders die op de manuele en automatische kranen zijn gemonteerd. De lasers lezen barcodes, aanwezig op de muren van de productiehallen, en stellen de coördinaten van de huidige positie voor. Hierdoor kan men op elk moment de positie van elke kraan te weten komen.



Figuur 1.4: Een gedeelte van de productiehal Sikel.

1.4.1 Kranen

Wanneer een kraan een rol opneemt, kan dit bekeken worden door de eigenschappen van de kraan te raadplegen, zoals weergegeven in Figuur 1.5. Naast het gewicht, worden ook andere metrieken gemeten, zoals de snelheid van de kraan.



Figuur 1.5: Kraan zonder rol (B) en kraan met rol (O)

1.4.2 Verplaatsingsbehoeftes

Genesis is meer dan een GUI die de realtime omgeving van een productiehal weergeeft. Het bevat ook alle verplaatsingsbehoeftes, informatie over de rollen, logging van acties en de eigenschappen van de verschillende kranen. Een verplaatsingsbehoefte beschrijft de begin- en eindpositie van een rol. Zo is een mogelijke verplaatsingsbehoefte “verplaats rol 7667 van punt a naar punt b”.

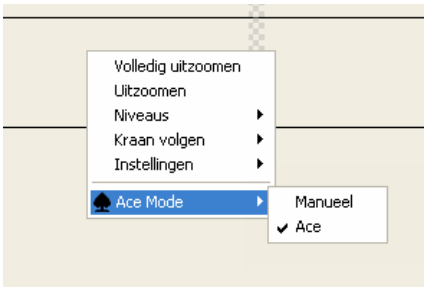
	Tijdstip	Process	Role	Hall	Category	Tekst
	28/02/2013 20:26:42.803	GnsLocalCoilStockManager	IAMTrace	SI	Ace	ZoneIDSegment received: ZoneID = OpAreaRCLOut, OkForAce = True
	28/02/2013 20:26:42.803	GnsLocalCoilStockManager	IAMTrace	SI	Ace	OpAreaRCLOut switched to True
	28/02/2013 20:26:43.614	GnsJobGenerator	SYMO	SI	Ace	Job K 00072249: +:Huidige pickup positie niet bereikbaar
	28/02/2013 20:26:43.614	GnsJobGenerator	SYMO	SI	Ace	Job K 00072249: +:Pickup niet ok voor ACE
	28/02/2013 20:26:43.614	GnsJobGenerator	SYMO	SI	Ace	Job K 00072249: +:Place niet ok voor ACE
	28/02/2013 20:26:43.646	GnsLocalCoilStockManager	IAM	SI	NeedsJobs	Job for K 00072249 to SCA 5 removed
	28/02/2013 20:26:43.661	GnsLocalCoilStockManager	IAM	SI	NeedsJobs	Job for K 00065113 to TUO 41 removed
	28/02/2013 20:26:43.661	GnsLocalCoilStockManager	IAM	SI	NeedsJobs	Job for K 00072249 to TUN 34 inserted
	28/02/2013 20:26:43.692	GnsJobGenerator	IAMTrace	SI	Ace	Copy properties manualModeRequested for OpAreaRCLOut to False
	28/02/2013 20:26:44.238	SchedulerService	IAMTrace	SI	Ace	Copy properties manualModeRequested for OpAreaRCLOut to False
	28/02/2013 20:26:48.622	AceService	IAMTrace	SI	Ace	LK201 Plc command completion Move K 00063495 From {X=196,17, Y=35,14} To TUO 37 (id =4580)
	28/02/2013 20:26:48.622	AceService	IAM	SI	Ace	LK201: Move K 00063495 from {X=196,17, Y=35,14}to {X=191,52, Y=35,12} completed
	28/02/2013 20:26:48.887	AceService	IAMTrace	SI	Ace	LK201 Plc command completion Move From null To null (id =4581)
	28/02/2013 20:26:48.918	AceService	IAM	SI	Ace	LK201 Moving Fanuk to: XYZ (1,8, 0, 0,24), ABC (23,8, 0, 90)
	28/02/2013 20:26:48.918	AceService	IAMTrace	SI	Ace	LK201 Send Move From null To null to PLC (id =4582)
	28/02/2013 20:26:50.135	AceService	IAMTrace	SI	Ace	LK201 Plc command completion Move From null To null (id =4582)

Figuur 1.6: Logging

De verplaatsingsbehoeftes worden naar de verschillende automatische kranen gestuurd. Doordat alle acties gearchiveerd worden in een logboek (Figuur 1.6), kan men eventuele fouten opsporen.

1.4.3 Automatic Crane Engine (ACE) mode

Deze modus (Figuur 1.7) bepaalt of de kranen in een productiehal automatisch of manueel opereren. Wanneer deze wordt ingesteld op manueel, stoppen alle automatische kranen in de geselecteerde productiehal onmiddellijk. De manuele modus wordt vooral gebruik tijdens de opstart van een productiehal, anders wordt deze maar sporadisch gebruikt.



Figuur 1.7: Ace mode

1.5 CamArc

Bij ArcelorMittal maakt men gebruik van het software pakket *CamArc*; dit staat voor Camera Archiveringssysteem. Deze software is op maat gemaakt en ontwikkeld door Cossilys [3]. CamArc maakt het onder andere mogelijk om de camerabeelden van de automatische kranen live te volgen of te herbekijken.

1.5.1 Technologische evolutie

Er worden al een aantal jaren camera's gebruikt in de warmwalserij. De camerabeelden worden live op schermen in de respectievelijke productiecabines getoond. Ze geven de operatoren informatie over het productiegebeuren.

Camerabeelden kunnen de operatoren waardevolle informatie opleveren. Er werd daarom geïnvesteerd in een nieuw opnamesysteem. Vroeger namen operatoren beelden op met een videorecorder met VHS-cassettes. Dat systeem had enkele belangrijke nadelen: de opnameduur was beperkt, elke camera had een eigen videorecorder nodig en de videobanden sleten snel door het veelvuldige gebruik. Wanneer men vergat de band te verwijderen, dan werden de beelden overschreven. Wilde men een bepaald beeld opzoeken, dan kostte dat veel tijd en moeite. Bovendien vergden de videorecorders veel onderhoud. Een alternatief was meer dan welkom.

Met de opkomst van de harddiskrecorders was het mogelijk om analoge camerasignalen te digitaliseren en te archiveren. Bovendien kon men op een harddiskrecorder verschillende camera's aansluiten. De recorders hadden ook een netwerkaansluiting, waarmee men de opgenomen beelden via een webbrowser kon opvragen.

Maar ook deze harddiskrecorders waren niet echt gebruiksvriendelijk. Het exacte tijdstip van het walsen en van de bijhorende opname waren vaak verschillend. Ook de digitale beeldkwaliteit liet soms te wensen over door het gebruikte compressiealgoritme en het beperkte aantal beelden per seconde.

De laatste jaren werden er steeds meer camera's gebruikt, waardoor de operatoren met meer beeldschermen moesten werken. Ook de onderhoudsdiensten waren vragende partij voor opgenomen beelden, om de oorzaak van bepaalde storingen te kunnen onderzoeken en een oplossing uit te werken. Tot slot zorgde ook de opkomst van netwerkcamera's, die rechtstreeks op het netwerk worden aangesloten, voor meer beeldmateriaal. De warmwalserij had dringend een nieuw systeem nodig, liefst één dat beelden kon linken aan het nummer van de rol.

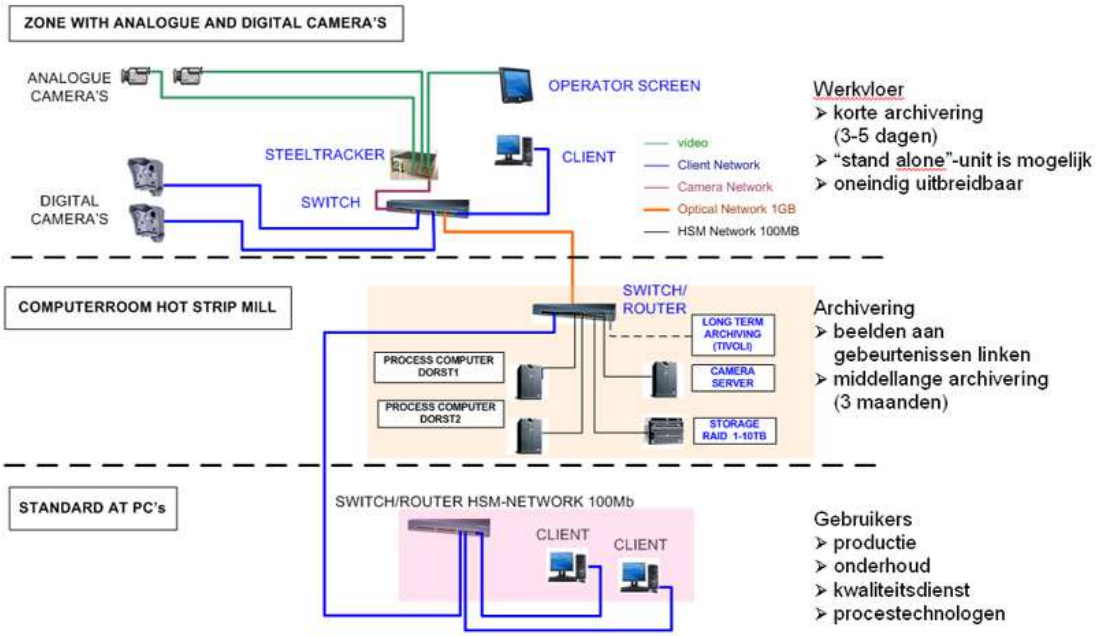
Om al die doelstellingen te verwezenlijken, zijn verschillende producenten en leveranciers geraadpleegd. De meeste leveranciers van archiveringssystemen boden hun standaardproduct aan, maar waren niet bereid om het aan te passen aan de specifieke noden van de warmwalserij. Uiteindelijk vond de afdeling een firma, het Franse bedrijf Cossilys[3], dat bereid was om specifieke software te ontwikkelen.

1.5.2 Infrastructuur

Het centrale element in het concept van CamArc is de "steeltracker" (Figuur 1.8). Het is een server die de analoge beelden digitaliseert en lokaal archiveert gedurende een aantal weken. Op één steeltracker kunnen 16 analoge camera's worden aangesloten. De lijntracking stuurt via de procescomputer de archivering van de beelden aan. Beelden die men voor een langere periode archiveert, staan op een cameraserver in de computerzaal van de warmwalserij.

Via de webbrowser, Internet Explorer, kan men opgenomen en live beelden raadplegen; andere webbrowsers zijn incompatibel. De gebruiker beschikt over verschillende zoekcriteria (rolnummer, periode en tijdstip) om beelden op te vragen, een hele verbetering.

Aanvankelijk werd het camerasysteem gebruikt om het proces te filmen vanaf de laadzijde van de ovens tot en met het warmbandtransport. Intussen wordt het systeem gebruikt in de volledige staalfabriek, in het plakkenpark (Figuur 1.1) en koudwals, waar de plakken (Figuur 1.1) uit de continugieterij worden opgeslagen voor ze naar de opwarmovens van de warmwalserij worden gebracht. [4]



Figuur 1.8: CamArc-netwerkinfrastructuur (bron: [4]).

1.5.3 Camera's op de automatische kranen

Ondertussen zijn een heleboel kranen geautomatiseerd in verschillende productiehallen, maar zoals hiervoor vermeld, geldt dit zeker niet voor alle kranen. Bij de automatische kranen worden camera's gebruikt. Deze camera's hebben foutopvolging als doel. Tijdens het opnemen van een rol staal gebruiken de automatische kranen 2D-scanners om de juiste positie t.o.v. de rol te bepalen. Net zoals bemande kranen, kunnen automatische kranen fouten maken. Denk maar aan een defecte scanner of sensor, die de oorzaak is van ongecontroleerd gedrag. Men kan dan aan de hand van deze camerabeelden, een diagnose stellen en een oplossing bedenken voor het probleem.

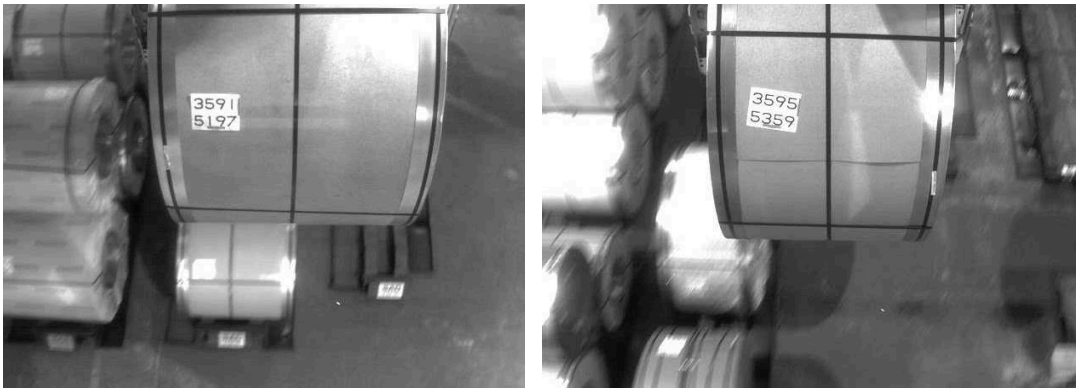
1.6 Herkenning

Door de camera's van de automatische kranen is het nu mogelijk om fouten te detecteren en op te lossen. Een bijkomende en nog belangrijkere controle bestaat in het nagaan of de opgenomen rol, de correcte rol is. Hiervoor controleert men het gewicht van de rol, maar in sommige situaties zijn er meer gegevens nodig, nl. het rolnummer bekijken; bijvoorbeeld bij het produceren van een serie rollen waarbij de rollen dezelfde afmeting hebben. Met een correcte rol wordt bedoeld dat de rol die opgenomen wordt overeenkomt met de rol die aangegeven is door Genesis. Deze veronderstelling is zeker niet vanzelfsprekend en moest bevestigd worden door de operatoren van de

bemande kranen. Er kunnen zogenaamde rolverwisselingen optreden door onoplettendheid van bemande kranen of rolnummers kunnen verwisseld worden tijdens het verpakken (meer hierover in hoofdstuk 2).

Bij de bemande kranen kan de operator het rolnummer op de rol vergelijken met het rolnummer aangegeven door Genesis. De automatische kranen hebben deze controle niet. Indien de automatische kranen ook over deze controle zouden beschikken, kan men jaarlijks een pak geld besparen en klachten van klanten vermijden (rolverwisselingen kunnen machines van klanten beschadigen). Daarnaast is dit een bijkomend argument om extra kranen te automatiseren. Er zou zelfs gezegd kunnen worden dat de taak "kraanman" overbodig wordt, maar dat is zoals hiervoor vermeld vrij onwaarschijnlijk.

Tijdens het analyseren (Figuur 1.9) van de verschillende camerabeelden heeft men opgemerkt dat de beelden niet alleen kunnen gebruikt worden voor foutopvolging. Doordat het rolnummer zeer frequent in de *viewport* van de camera komt, zouden de automatische kranen ook in staat zijn om deze rolnummers te herkennen en te besluiten of de rol al dan niet correct is.



Figuur 1.9: Camerabeelden; rolnummers.

Het herkennen van de rolnummers a.d.h.v. (camera)beelden is dan ook het doel van deze masterproef. Om dit mogelijk te maken, dient een herkenningsalgoritme ontwikkeld te worden, dat het rolnummer kan situeren en herkennen. Daarnaast zal er een evaluatietool ontwikkeld worden in de programmeertaal C#, om bestaande beelden te herevalueren. Uiteindelijk zal het herkenningsalgoritme in het huidige systeem van ArcelorMittal Gent worden geïntegreerd, en zal het algoritme uitgebreid geëvalueerd worden.

2 Analyse

Door verkeerde handelingen van bemande kranen kunnen rolverwisselingen optreden, waardoor de betrouwbaarheid van Genesis in het gedrang komt. Het is vanzelfsprekend dat rolverwisselingen te allen tijde vermeden moeten worden en naar de klant toe onaanvaardbaar zijn.

In dit hoofdstuk bekijken we de oorzaak van een rolverwisseling in detail. Daarnaast bekijken we hoe de rolnummers kunnen herkend worden en worden enkele problemen besproken die het herkennen bemoeilijken. Op basis van deze problemen zullen enkele methodes ontwikkeld worden, die het situeren en herkennen van het rolnummer mogelijk maken.

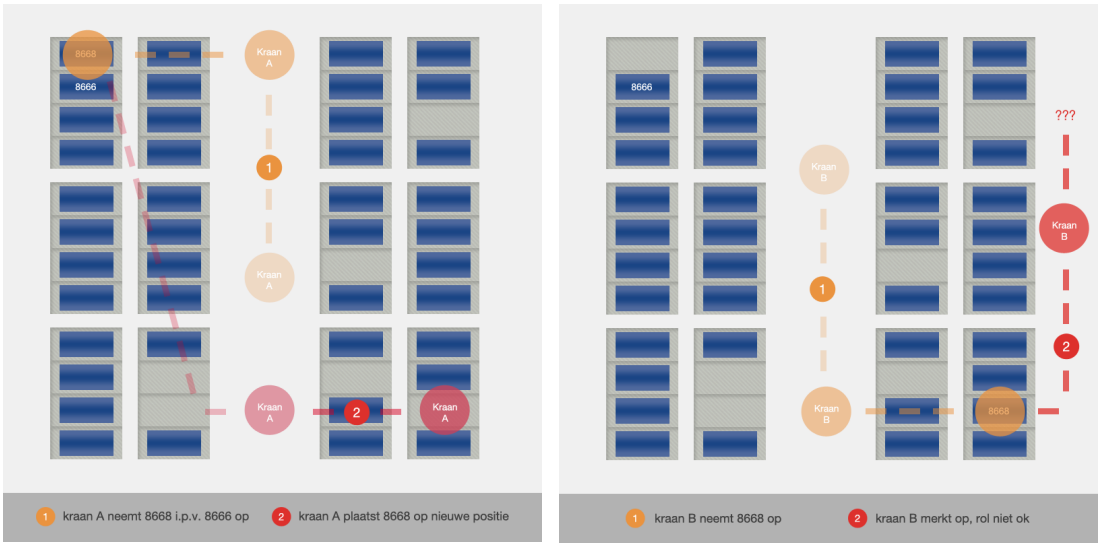
2.1 Rolverwisseling

2.1.1 Bij het verplaatsen

De automatische kranen nemen steeds een rol op, op de plaats die aangewezen is door Genesis. Aangezien de automatische en bemande kranen simultaan en onafhankelijk van elkaar werken, zetten de automatische kranen mogelijke fouten, gemaakt door de bemande kranen, onwetend verder.

Om meer inzicht te krijgen in de samenwerking van de kranen en de problemen die ze kunnen veroorzaken, nemen we als voorbeeld een bemande kraan die een verkeerde rol verplaatst en een rolverwisseling veroorzaakt. We illustreren in Figuur 2.1.

De kraanman van *kraan B* kijkt op het computerscherm naar het rolnummer van de op te nemen rol. Dankzij Genesis weet hij precies waar de rol zou moeten liggen; maar het is zeker niet vanzelfsprekend dat de rol die men wil verplaatsen effectief op die plaats ligt. Stel dat deze rol het rolnummer 8666 heeft. Het kan best zijn dat deze rol een uur voordien werd verplaatst door een andere kraan, *kraan A*.



Figuur 2.1: Rolverwisseling door kraan

We nemen aan dat dit effectief ook zo was. De vorige kraanman, van *kraan A*, die rol 8666 verplaatst heeft was op dat moment zijn shift aan het afwerken. *Kraan A* wou nog vlug die laatste rol 8666 verplaatsen, maar nam de verkeerde rol 866**8** op i.p.v. rol 866**6**, die één plaats verder lag.

Dit kon doordat het rolnummer 8666 niet leesbaar was of zelfs niet aanwezig. We gaan ervan uit dat de kraanman de correcte positie, gegeven door Genesis negeert of dat het geïnformatiseerd systeem tijdelijk niet beschikbaar is. Een belangrijke opmerking hierbij is dat Genesis in theorie de exacte positie van een kraan weet en dus eigenlijk ook weet als er een rolverplaatsing veroorzaakt wordt. In praktijk is dit niet zo, bijvoorbeeld bij een defecte decoder, waardoor Genesis er toch vanuit gaat dat de correcte rol wordt verplaatst.

Vervolgens verplaatst *kraan A* de verkeerde rol. Wanneer *kraan B* aankomt op de aangewezen positie, zal hij de rol opnemen en opmerken dat dit niet de rol 8666 is maar wel 8668.

Vervolgens zal *kraan B* de rol 8668 terug naar zijn correct positie brengen en mogelijks de volledige productiehal moeten afgaan, op zoek naar de rol met het rolnummer 8666. Stel bijvoorbeeld dat een verkeerde rol pas na de vijfde verplaatsing wordt opgemerkt, dan zou het systeem de posities moeten kunnen nagaan van de vijf vorige plaatsen.

Hierdoor is er veel geheugen en logica nodig om al deze plaatsen bij te houden. Voor sommige systemen is dit ontoelaatbaar of gewoon niet aanwezig, omdat men er nooit eerder ernstige problemen heeft van ondervonden. Gelukkig is dit bij ArcelorMittal niet zo.

Als een foute rol pas gedetecteerd wordt na de tiende verplaatsing, dan kan de kraanman toch gewoon snel op zoek gaan naar de tiende vorige plaats? Dat is zeker waar,

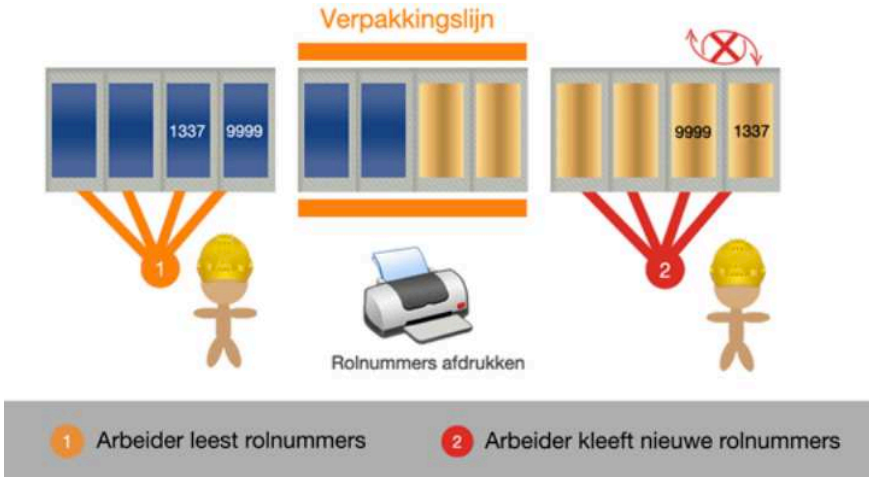
maar wat als Genesis uitvalt? Dit scenario is al voorgekomen en is dus op elk moment mogelijk. Om dit probleem op te lossen, zit er niets anders op dan manueel op zoek te gaan naar die ene rol, in de volledige productiehal of zelfs productiehallen. Het is niet verwonderlijk dat hiervoor beroep gedaan wordt op speciale teams, die zich enkel met dit soort problemen bezighouden.

Het is de bedoeling om met behulp van de rolnummerherkenning, rolverwisselingen op te sporen en te vermijden.

2.1.2 Bij het inpakken

Naast kranen kunnen ook arbeiders rolverwisselingen veroorzaken. Tijdens het productieproces worden de rollen staal van de ene productielijn naar de andere verplaatst. Dit geldt onder andere voor de verpakkingslijnen waar de rollen staal worden verpakt om het staal te beschermen tijdens het transporteren.

Rechts en onder het rolnummer zijn (kleine en identieke) barcodes aanwezig, die ook het rolnummer voorstellen. Zoals weergegeven op Figuur 2.2 wordt één van deze barcodes ingelezen door een arbeider. Aan de hand van het rolnummer (barcode) weet de computer aan de verpakkingslijn welke rol het is en hoe de rol moet worden verpakt. Vervolgens worden de verpakkingsdetails getoond op een scherm zodat ook de arbeiders weten hoe de rol moet worden verpakt. Daarnaast wordt tijdens het inlezen ook een nieuw rolnummer afgedrukt.



Figuur 2.2: Rolverwisseling door arbeiders

Wanneer de arbeiders een rol hebben verpakt, wordt extra bandijzer rond de verpakking voorzien. Hierna wordt, door een arbeider, het geprinte rolnummer boven op de verpakking geplakt.

De manier waarop dit rolnummer geplakt wordt kan voor twee problemen zorgen:

- het verkeerde rolnummer wordt op de rol geplakt, en

- het rolnummer wordt bovenaan de rol geplakt (meer hierover in volgende paragraaf).

Wanneer een rol voorzien wordt van een verkeerd rolnummer is er ook een rolverwisseling gebeurd. De rolverwisseling door arbeiders is gelijkaardig met die door kranen, maar toch is er een belangrijk verschil. Bij de rolverwisselingen door kranen lag de rol op de verkeerde plaats, maar was de rol effectief de rol die door het rol-nummer werd aangegeven. Bij de rolverwisselingen door arbeiders ligt de rol wel op de juiste plaats, maar komt het rolnummer niet overeen met de rol.

Het spreekt voor zich dat deze vorm van rolverwisseling complexer is. Er moet nu ook gecontroleerd worden of de rol effectief overeenstemt met het vermelde rolnummer. Dit kan men oplossen door de eigenschappen (diameter, breedte en/of gewicht) van de rol te raadplegen; soms is nog meer controle nodig. Tenslotte moeten ook deze soort rolverwisselingen worden opgemerkt en daarvoor kan de rolnummerherkenning een oplossing bieden.

We zullen zien dat de herkenning niet zo eenvoudig is als die op het eerst gezicht lijkt. Er zijn verschillende factoren en obstakels waar rekening mee moet worden gehouden.

2.2 Analyse van de camerabeelden

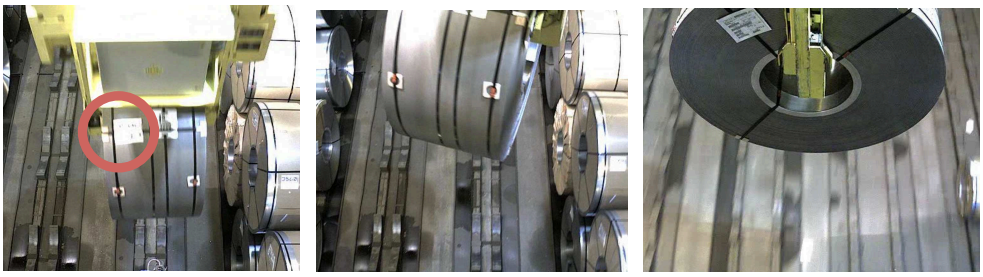
Om de rolnummerherkenning mogelijk te maken, zullen we de camerabeelden van de automatische kranen gebruiken. Daardoor kunnen we de rolverwisselingen zoals hiervoor vermeld detecteren. Het detecteren en herkennen van het rolnummer is niet zo eenvoudig omdat de kranen kunnen roteren. Ook is er veel belichting in de productiehallen, waardoor het rolnummer soms onzichtbaar wordt. We bespreken daarom deze moeilijkheden in deze paragraaf. Vervolgens zullen we in hoofdstuk 4 enkele methoden voorstellen die het herkennen van het rolnummer mogelijk maken. We zullen deze methodes evalueren aan de hand van onderstaande moeilijkheden, en op basis hiervan de meest gunstige methode selecteren.

2.2.1 Rotatie kranen

Een rol kan op twee manieren worden verplaatst, namelijk opnemen of opnemen en 90° roteren. Het roteren is afhankelijk van de verplaatsingsbehoefte van de te verplaatsen rol. Het is namelijk zo dat er naast horizontale ook verticale stockeringsplaatsen zijn, waardoor roteren vereist is (Figuur 1.4).

Figuur 2.3: 90° roteren (links) en zonder rotatie (rechts).

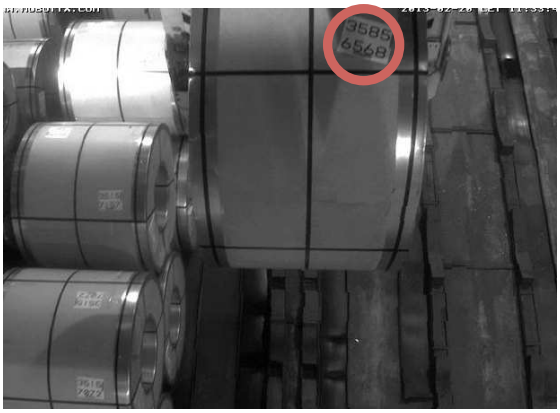
Het nadeel van het roteren is dat het rolnummer niet meer in beeld komt tijdens het verplaatsen van de rol. Het label is wel nog zichtbaar tijdens het opnemen, maar van zodra de verplaatsing start, wordt de rol 90° geroteerd en is het rolnummer uit het zicht van de camera (figuur 1.9). Vice versa is ook mogelijk.



Figuur 2.4: 90° roteren bij het verplaatsen van horizontale naar verticale plaats.

2.2.2 Rotatie label

Daarnaast kan het rolnummer ook schuin geplakt worden, waardoor dit de herkenning kan bemoeilijken.



Figuur 2.5: Rolnummer schuin geplakt.

2.2.3 Belichting

In de productiehallen is veel belichting aanwezig en dit in eerste plaats voor de veiligheid van de arbeiders. Deze belichting, onder andere aanwezig in het dak van de productiehallen, vormt een bijkomend probleem voor het herkennen van de rolnummers. Wanneer een rol verplaatst wordt door een kraan van punt a naar punt b, is het mogelijk dat de kraan zich tijdens het verplaatsen onder één of meerdere lichtspots bevindt. Deze spots overbelichten dan het rolnummer, waardoor het onleesbaar wordt voor de camera's van de geautomatiseerde kranen (Figuur 2.6). Het is ook mogelijk dat een kraan zelf over extra belichting beschikt en hierdoor zelf de rolnummers overbelicht, waardoor een rolnummer nooit herkend kan worden.

Figuur 2.6: Overbelichting van rolnummer door een lichtspot.

Figuur 2.7: Onderbelichting van rolnummer.

Het tegenovergestelde, onderbelichting, is ook mogelijk. Dit kan veroorzaakt worden door de schaduw van de kraan of door te weinig of geen belichting van een spot (Figuur 2.6).

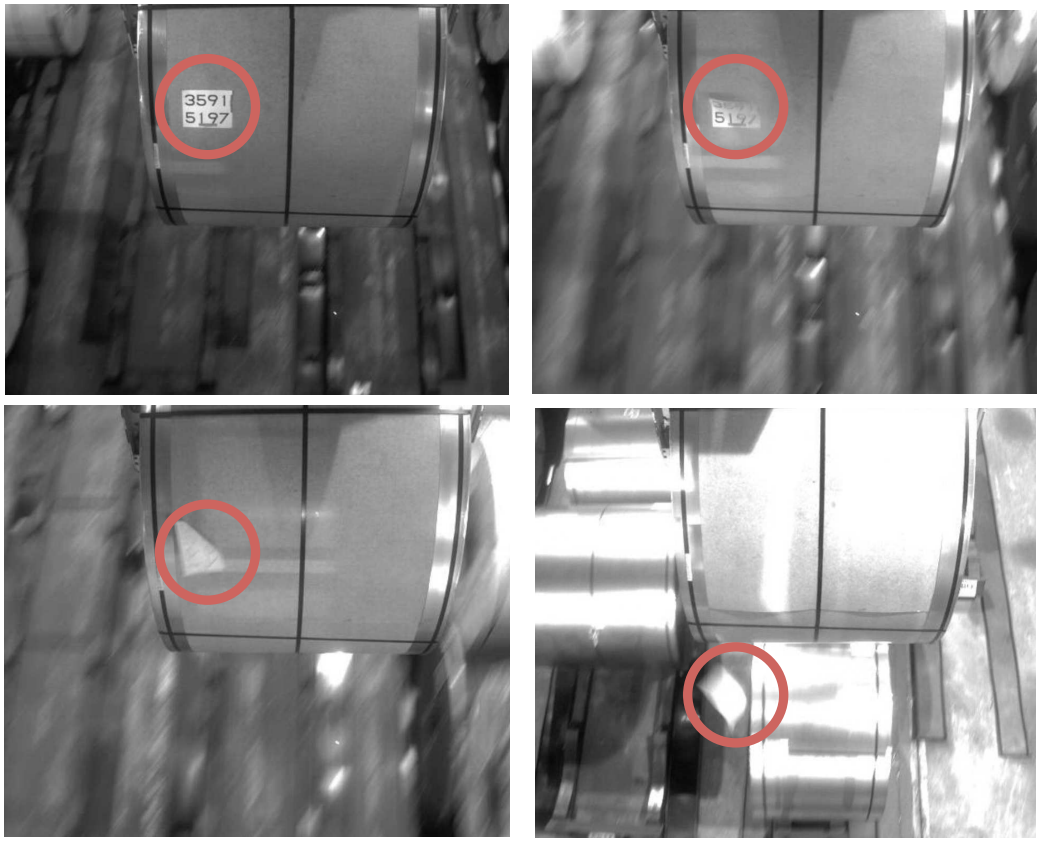
2.2.4 Positie rolnummer

De positie van het rolnummer is niet op voorhand bepaald en kan dus theoretisch gezien op elke mogelijke plaats van de rol geplaatst worden. De enige voorwaarde is dat het rolnummer zichtbaar is voor de kraanman, niet noodzakelijk voor de camera's van

de automatische kranen. De camera's van de automatische kranen zijn niet uitgelijnd. Hierdoor is het mogelijk dat het bovenste gedeelte van een rol staal tijdens het verplaatsen niet in beeld komt. Als dan het rolnummer bovenaan de rol geplaatst is, kan het voorkomen dat het rolnummer slechts gedeeltelijk of zelfs niet zichtbaar is voor de camera van de automatische kraan (Figuur 2.8).

Figuur 2.8: Label buiten het bereik van de camera.

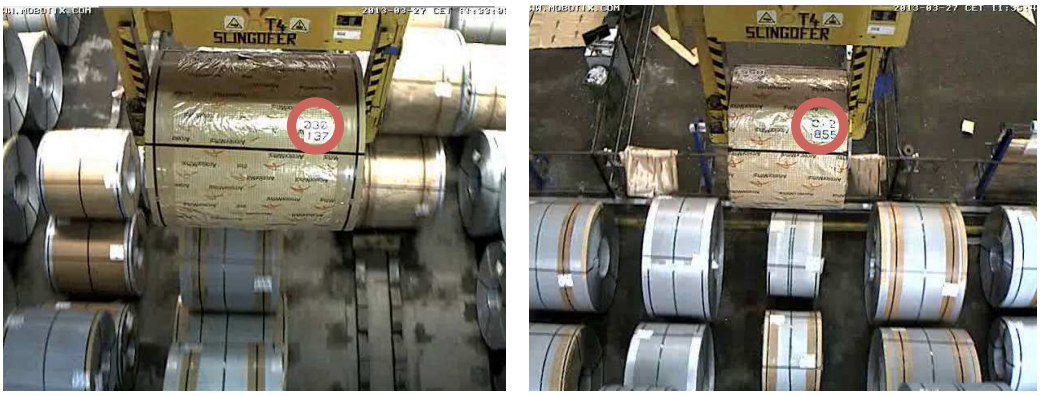
Het is ook mogelijk dat een rol zelfs geen rolnummer heeft. Dit kan door het vergeten plaatsen van het rolnummer – tijdens het verpakken - of door het verliezen tijdens het verplaatsen. Op Figuur 2.9 zien we een rolnummer dat verloren gaat tijdens het verplaatsen van de rol. Dit kan komen door het niet goed genoeg vastmaken van het label.



Figuur 2.9: Verliezen van rolnummer.

2.2.5 Verpakking

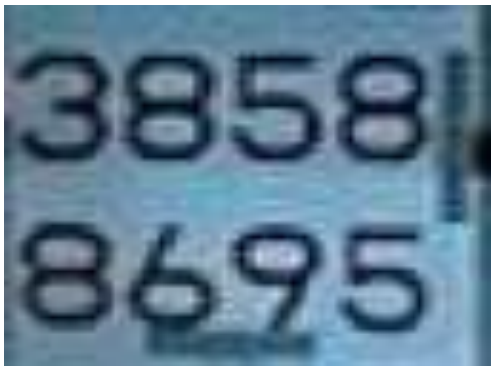
Rollen kunnen, als de klant dat wenst, verpakt worden met papier of karton. Papieren verpakking kan de herkenning van het rolnummer bemoeilijken (Figuur 2.10). De bedrukking “ArcelorMittal“ op de papieren verpakking zou voor valse herkenningen kunnen zorgen; het rolnummer en de bedrukking zijn allebei tekst, en hebben dus dezelfde eigenschappen.



Figuur 2.10: Verpakking met beschrijving “ArcelorMittal”.

2.2.6 Barcodes

Zoals vermeld in paragraaf 2.1.2 zijn er barcodes aanwezig op het rolnummer. Ook deze barcodes kunnen het herkennen bemoeilijken omdat deze vrij dicht bij het rolnummer zijn geplaatst. In Figuur 2.11 wordt duidelijk dat deze barcodes voor problemen kunnen zorgen. Zo kan het nummer “9” ook worden bekeken als het nummer “2”.



Figuur 2.11: Rolnummer en barcodes.

2.3 Alternatieve identificatie

Zoals hiervoor vermeld zijn er heel wat struikelblokken die het herkennen kunnen bemoeilijken. Een voor de hand liggende oplossing zou kunnen zijn om bijvoorbeeld een QR-code of een barcode te gebruiken i.p.v. een rolnummer; dit is voor een com-

puter eenvoudiger. Jammer genoeg kan dit niet omdat het rolnummer nog steeds door een kraanman moet kunnen gelezen worden. Het plaatsen zowel van een rolnummer als een vergrote barcode of QR-code is ook niet mogelijk.

2.4 Analyse van de testgegevens

In de vorige paragraaf hebben we de voornaamste problemen besproken. Naast deze ongunstige situaties zijn er natuurlijk ook veel gunstige situaties, waarbij het herkennen geen probleem is of zou mogen zijn. Zoals hiervoor vermeld, zullen we in hoofdstuk 4 enkele oplossingsmethoden voorstellen. Om deze methoden te kunnen evalueren, zullen we hiervoor elke methode uitgebreid moeten testen. Hiervoor wordt handmatig een evaluatieset opgesteld waarbij voor elke probleemklasse, een tiental frames worden geselecteerd uit verschillende opnames.

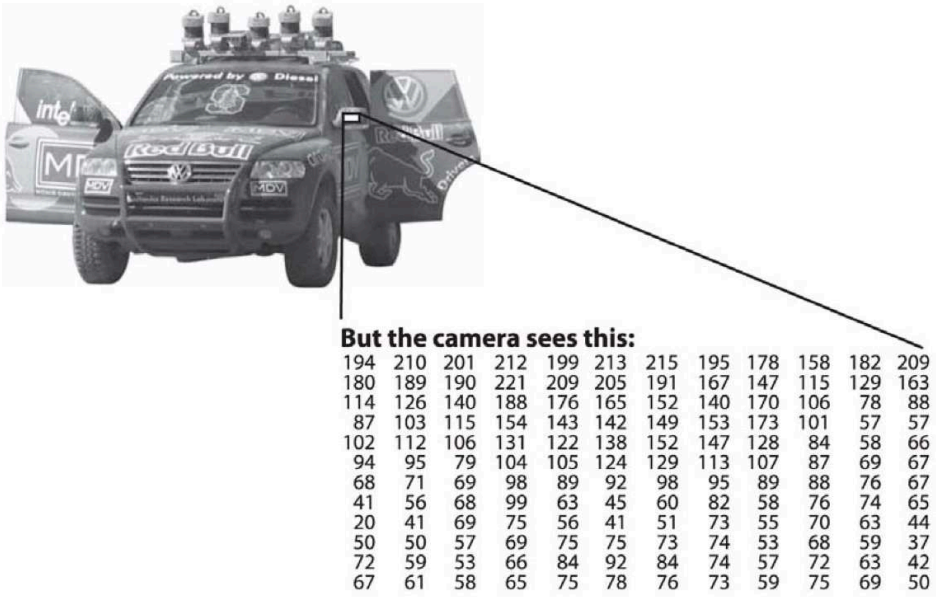
De evaluatieset bestaat uit volgende klassen:

- gunstige afbeeldingen
- afbeeldingen waarbij de kraan geroteerd is
- afbeeldingen met schuine rolnummers
- overbelichte afbeeldingen
- onderbelichte afbeeldingen
- afbeeldingen waarbij het rolnummer gedeeltelijk zichtbaar is
- afbeeldingen waarbij de rol verpakt is

2.5 Digitaal beeld

Digitale beelden kunnen gemaakt zijn door een computer of door een andere digitale bron. Ook wanneer een afbeelding wordt gescand, wordt de afdruk bewaard als een digitaal bestand. De voorstelling van zo'n digitaal beeld, in het geheugen van een computer, wordt weergegeven als een raster. Hierbij krijgt elke pixel van het raster een waarde of amplitude. Omdat digitale beelden fundamenteel zijn in deze masterproef, wordt kort de constructie van een digitaal beeld door een digitale camera besproken.

De weerkaatste lichtstralen van een beeld worden geprojecteerd op de beeldsensor. Bij een digitale camera is dit een CCD- of CMOS-sensor die bestaat uit een raster van lichtgevoelige pixels. Waarbij de grootte van het raster de resolutie bepaalt. Elk van die puntjes meet de lichtintensiteit in drie verschillende kanalen R, G en B. Elk kanaal stelt een getal voor tussen 0 en 255 (8 bits). De waarden worden op die manier gedigitaliseerd en kunnen door een computer geïnterpreteerd worden. De computer beschikt enkel over deze 3 bytes per pixel, en heeft geen besef van wat de afbeelding precies voorstelt. Een beeld is voor een computer - net zoals muziek, tekstdocumenten, video, enz. - een binaire stroom. Bij ons mensen gebeurt hetzelfde maar wij gebruiken onze hersenen om betekenis te geven aan de beelden die worden opgenomen door onze ogen. Figuur 2.12 toont dit aan voor een zwart-witafbeelding. Dit is een afbeelding met slechts één byte per pixel (grijswaarden dus). [5]



Figuur 2.12: De zijspiegel een raster van waarden (0-255) (bron: [6])

2.6 Beeldverwerking

Beeldverwerking is het verwerken van beelden om er de gewenste informatie uit af te leiden - in deze masterproef het rolnummer. Dezer dagen maakt bijna elk technisch vakgebied, rechtstreeks of onrechtstreeks, gebruik van digitale beeldverwerking – zoals de medische sector, defensie, auto-industrie, gamewereld enz. Tegenwoordig wordt vooral onderzoek gedaan naar patroonherkenning. Ook wordt er steeds meer en meer geïnvesteerd in ontwikkeling van goedkope beveiligingssystemen, die door onderzoek in beeldverwerking mogelijk gemaakt zijn. Daarnaast probeert men ook beeldverwerkingsalgoritmen in hardwarecomponenten te integreren, waardoor de uitvoeringstijd aanzienlijk daalt.

Om dit mogelijk te maken wordt gebruikgemaakt van verschillende beeldverwerkingstechnieken. Daarom zullen we in hoofdstuk 3 enkele belangrijke basisbewerkingen bespreken en een bestaande beeldverwerkingsbibliotheek selecteren. Vervolgens zullen we in hoofdstuk 4 deze verschillende technieken combineren en enkele methodes voorstellen die het herkennen van het rolnummer mogelijk maken.

2.7 Eigenschappen camera's

De resolutie van de camera's is 640×480. De camera's kunnen automatisch overschakelen van kleurbeeld naar grijswaardenbeeld (en vice versa). Door het gebruik van compressiealgoritmen wordt de bestandsgrootte van de camerabeelden sterk verminderd, met als gevolg dat ook de beeldkwaliteit afneemt. Bij CamArc wordt gebruikgemaakt van het MPEG4-D1 compressie-algoritme [7]. Dit is een lossy algoritme, maar de compressieartefacten die het oplevert, beïnvloeden het resultaat van het herkenningsalgoritme amper.

3 Basisbewerkingen bij beeldverwerking

In dit hoofdstuk worden enkele beeldverwerkingstechnieken besproken. Deze technieken komen in aanmerking voor de verschillende methoden die in hoofdstuk 4 zullen worden behandeld.

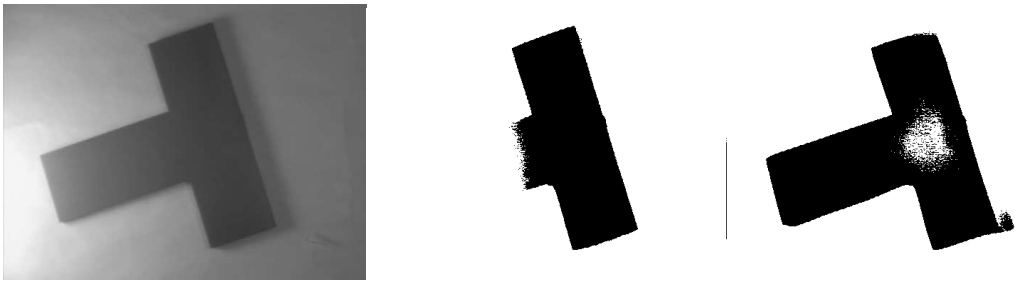
3.1 Thresholding

In digitale beeldverwerking probeert men vaak om een afbeelding te segmenteren. Dit is het verdelen van een afbeelding in deelgebieden van pixels, waarbij alle pixels van een deelgebied dezelfde eigenschap(pen) vertonen. Een belangrijke operatie hiervoor is *thresholding* en wordt gedefinieerd als:

$$g(x,y) = \begin{Bmatrix} 1 & if & f(x,y) > T \\ 0 & if & f(x,y) <= T \end{Bmatrix}$$

Aan de hand van een drempelwaarde (threshold) wordt de afbeelding in twee deelgebieden verdeeld (Figuur 3.1), nl. de achtergrond en voorgrond. De manier waarop deze drempelwaarde gekozen wordt is van groot belang. Een threshold wordt *globaal* genoemd als ze bepaald wordt door één drempelwaarde, *lokaal* als ze bepaald wordt per deelgebied en *adaptief* als ze gedefinieerd wordt door de x- en y-coördinaten. [8]

In veel algoritmen is thresholding een zeer cruciale stap. Hierdoor wordt vaak gekozen voor een adaptieve threshold, omdat deze dynamischer is dan een globale threshold. Er zijn veel verschillende methodes om thresholding mogelijk te maken. Een groot aantal hiervan zijn histogramgebaseerde methodes, zoals *Otsu's method*, waarbij pieken en dalen in het histogram worden geanalyseerd. [9]



Figuur 3.1: Originele afbeelding (links), globale threshold (midden) en adaptieve threshold (rechts). (bron: [10])

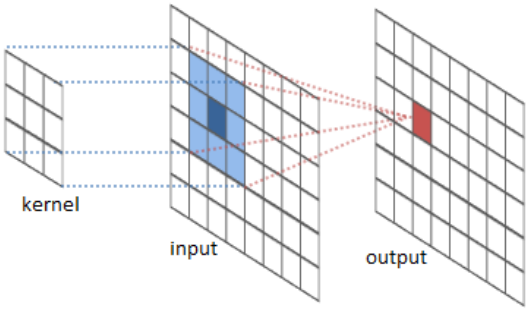
3.2 Discrete convolutie

Veel beeldverwerkingsalgoritmen – zoals randdetectie, vervaging, morfologische transformaties, etc. – maken gebruik van maskers. De maskers worden ook wel convolutiemaskers genoemd, omdat ze in combinatie met de convolutieoperator worden gebruikt. De convolutie van twee functies geeft als resultaat een nieuwe functie (bij randdetectie een afbeelding waarop enkel de randen zijn weergegeven). Omdat in deze masterproef met digitale afbeeldingen wordt gewerkt, en dus discrete waarden, zullen we hiervoor gebruikmaken van de discrete convolutie, gegeven door:

$$g(m,n) = f(m,n) * h(m,n) = \sum_{p=0}^{n-1} \sum_{q=0}^{m-1} f(p,q)h(n-p,m-q)$$

met m, het aantal kolommen en n, het aantal rijen. [11]

Hierbij zijn de functies $f(p,q)$ en , $h(n-p,m-q)$ respectievelijk de afbeelding en het masker. Om de discrete convolutie te berekenen wordt het masker over de afbeelding geschoven. De overlappende pixels van het masker met de afbeelding worden elk met hun overeenkomstige pixel vermenigvuldigd. De verschillende producten worden gesommeerd en de som wordt in het ankerpunt geplaatst (Figuur 3.2).



Figuur 3.2: Discrete convolutie. (bron: [10])

3.3 Vervagen

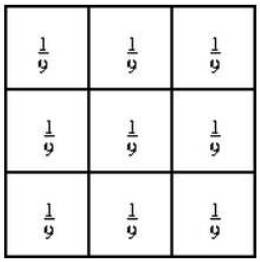
Een belangrijke toepassing van de convolutie is het vervagen van een afbeelding. Dit is het elimineren/filteren van ruis of het wegwerken van detail, wat handig kan zijn

om valse resultaten te vermijden. Hiervoor wordt een masker gebruikt, dat over de afbeelding wordt geschoven (discrete convolutie). Hoe de afbeelding wordt vervaagd is afhankelijk van het masker. Het vervagen noemt men ook wel filteren. We bespreken drie belangrijke filters:

- *Mean filter*
- *Gaussian filter*
- *Median filter*

3.3.1 Mean filter

Bij deze filter wordt het rekenkundig gemiddelde berekend van de overlappende pixels, en wordt het resultaat in het ankerpunt geplaatst (Figuur 3.3).



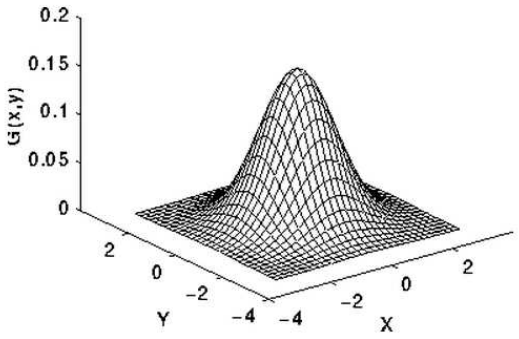
Figuur 3.3: Mean (3x3) filter. (bron: [12])

3.3.2 Gaussian filter

De Gaussian filter is gelijkaardig met de mean filter; enkel de gewichten van de maskers verschillen. Het idee van een gaussiaans masker is dat pixels dichter bij het ankerpunt meer invloed hebben dan pixels verder weg. De manier is, zoals de naam het aangeeft, gebaseerd op de gaussiaanse verdeling. De 2D gaussiaanse verdeling wordt gegeven door:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

met σ de standaardafwijking en het gemiddelde = 0 (de piek van de verdeling). De standaardafwijking bepaalt de steilheid of hoe wijd de verdeling is. Hoe groter de standaardafwijking wordt gekozen, hoe wijder de verdeling; verder gelegen pixels hebben nog steeds een grote invloed (maar iets minder dan het ankerpunt). Een grafische representatie van een 2D gaussiaanse verdeling wordt gegeven op Figuur 3.4 en de bijhorende filter(masker) op Figuur 3.5, met σ=1.



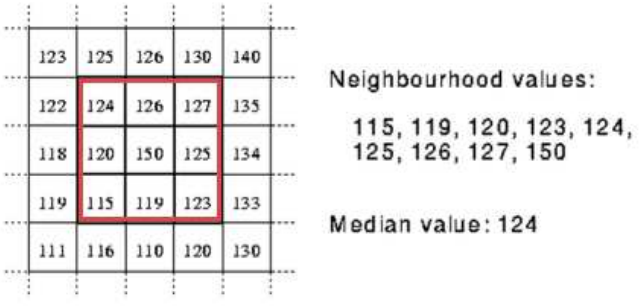
Figuur 3.4: 2D gaussiaanse verdeling. (bron: **[13]**)

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figuur 3.5: 2D gaussiaans masker. (bron: **[13]**)

3.3.3 Median filter

De volgende filter gebruikt een masker impliciet; het masker wordt in gedachten over de afbeelding geschoven. De pixels die overlappen met het masker komen in aanmerking. De overlappende pixels worden gesorteerd en de mediaan wordt bepaald. Vervolgens wordt de mediaan in het ankerpunt weggeschreven (Figuur 3.6). Hierna wordt het masker, fictief, verschoven en wordt de bewerking herhaald. De median filter kan gebruikt worden om uitschieters te verwijderen.



Figuur 3.6: Mediaan (3x3) filter. (bron: [14])

3.4 Mathematische morfologie

Maskers worden ook gebruikt bij de morfologische transformaties. *Morfologie* is een onderdeel uit de biologie, waarin men de vorm en structuur van planten en dieren bestudeert.

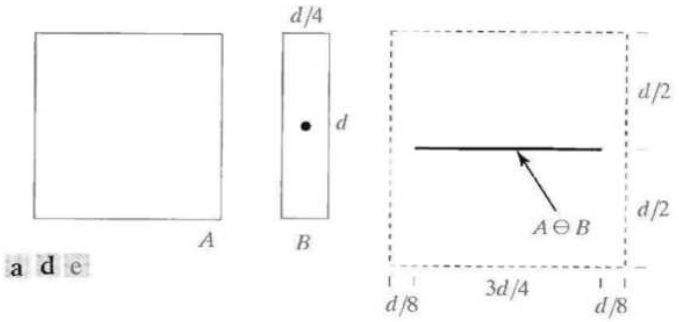
Daarnaast wordt *mathematische morfologie* gebruikt in digitale beeldverwerking om componenten, die aan bepaalde voorwaarden voldoen, te bepalen. De formulering die hierbij gebruikt wordt is de verzamelingenleer. De mathematische morfologie omvat enkele belangrijke operaties zoals *opening and closing*, *hole filling*, *boundary extraction*, *extraction of connected components*, *thinning*, *convex hull*, etc. De twee basisoperaties die hierbij gebruikt worden noemt men erosie en dilatatie.

3.4.1 Basisoperaties

3.4.1.1 Erosie

Erosie is een natuurverschijnsel waarbij een vast oppervlak wordt verplaatst of zelfs volledig verdwijnt. Bij beeldverwerking is dit verschijnsel gelijkaardig; en wordt een structuurelement/masker B gebruikt om een element A te slijten. De erosie van A door B wordt aangegeven door de verzameling van alle waarden z zodat B, verschoven over z, een deelverzameling is van A (Figuur 3.7). De erosie wordt gedefinieerd als:

$$A \odot B = \{z \mid [(B)z \subseteq A\}$$



(a) Set A. (d) Elongated structuring element. (e) Erosion of A using this element.

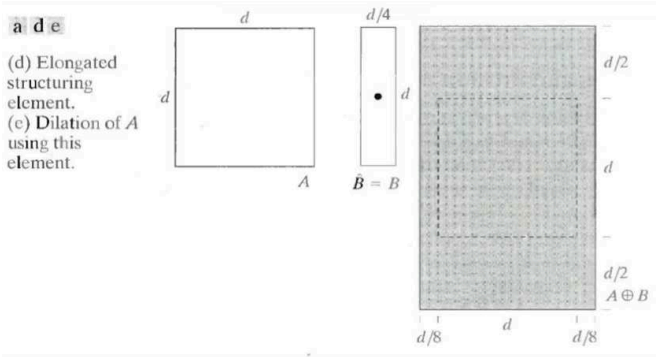
Figuur 3.7: Voor en na eroderen. (bron: [6])

Het eroderen van aan afbeeldingen kan toegepast worden om bijvoorbeeld ruis uit de afbeelding te verwijderen.

3.4.1.2 Dilateren

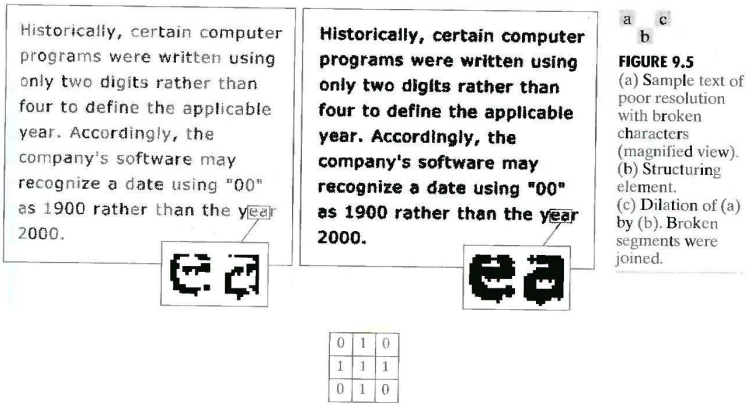
Het tegenovergestelde noemt men *dilateren*; er bestaat een dualiteit tussen eroderen en dilateren. Bij dilateren is het de bedoeling om een element A uit te breiden met een structuurelement/masker B. Hierbij gebeurt een reflectie van het structuurelement B rond zijn oorsprong en wordt de reflectie over een afstand z verschoven. De dilatatie van een element A en een structuurelement B is de verzameling van alle waarden z, waarbij A en B minstens één pixel overlappen (Figuur 3.8), en wordt dus gedefinieerd als:

$$A \oplus B = \{z \mid [(B)z \cap A] \subset A\}$$



Figuur 3.8: Voor en na dilateren. (bron: [6])

Dilatatie kan gebruikt worden om gaten op te vullen, en meer specifieker om gebroken karakters te reconstrueren door het gepaste (kruis)masker te gebruiken (Figuur 3.9).



Figuur 3.9: Herconstrueren van gebroken karakters. (bron: [6])

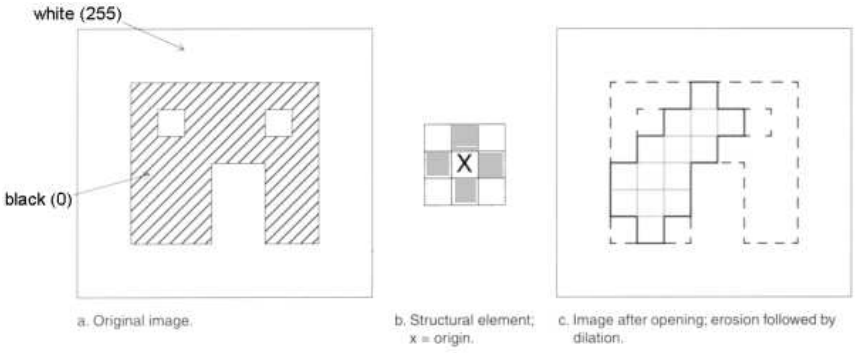
3.4.2 Opening en sluiting

Door de basisoperaties erosie en dilatatie te combineren krijgen we nieuwe operaties, nl. opening en sluiting. Een belangrijk detail is dat deze operaties meerdere keren na elkaar kunnen worden uitgevoerd, maar slechts één keer invloed hebben. Ze zijn net zoals dilatatie en erosie duaal. De opening en sluiting hebben enkele belangrijke toepassingen zoals bij vingerafdrukherkenning. [15]

3.4.2.1 Opening

Opening, erosie gevolgd door dilatatie (met hetzelfde masker B), zorgt ervoor dat uitsteeksels verwijderd worden en de contouren verzacht worden (Figuur 3.10). Het wordt gedefinieerd als:

$$A \bullet B = (A \odot B) \oplus B$$



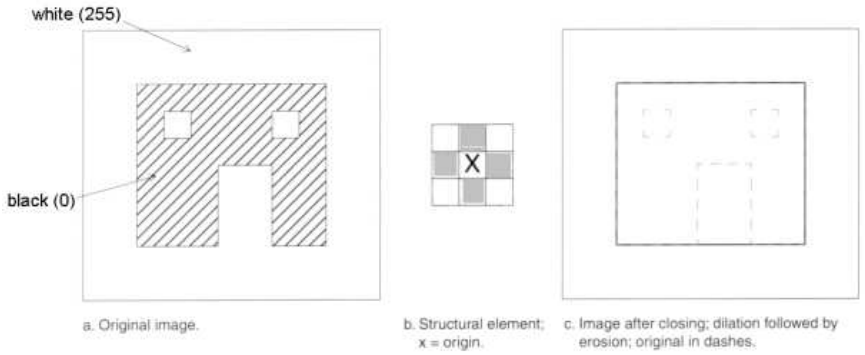
Figuur 3.10: Voorbeeld van opening. (bron: [6])

De opening kan ook gedefinieerd worden als, de unie van alle B objecten die zich volledig in A bevinden.

3.4.2.2 Sluiting

Sluiting, dilatatie gevolgd door erosie (met hetzelfde masker B), zorgt ervoor dat gebieden in elkaar overlopen; kleine gaatjes worden opgevuld, maar de originele grootte blijft bewaard (Figuur 3.11). Het wordt gedefinieerd als:

$$A \bullet B = (A \oplus B) \odot B$$



Figuur 3.11: Voorbeeld van sluiting. (bron: [6])

3.5 Randen

Een rand in een digitale afbeelding is een grens tussen twee gebieden of een set punten waar de intensiteit zeer sterk verandert. Randen kunnen ons waardevolle informatie opleveren, waardoor we belangrijke veronderstellingen kunnen maken. Ze worden dan ook zeer frequent gebruikt bij objectherkenning en bewegingsdetectie (bijvoorbeeld het detecteren van verkeersborden). De interesse naar efficiënte en foutloze randdetectie algoritmen is dan ook zeer groot. Er is al behoorlijk wat onderzoek naar gedaan. Uit onderzoek zijn verschillende efficiënte algoritmen ontstaan die ondertussen zeer populair zijn, zoals de *Canny edge detector*, die wordt besproken in 3.5.1. [16] [17]

In deze paragraaf zullen we enkele methodes bespreken om randen te detecteren, of concreter randpixels (pixels waar de intensiteit sterk verschilt). Vervolgens zullen we een methode voorstellen om randpixels te *linken*, om effectief de randen te detecteren. Het detecteren van randen gebeurd in algemene twee stappen:

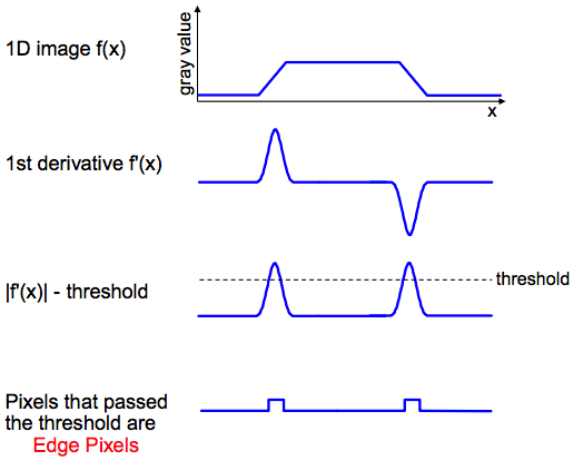
1. Bepalen van de randpixels
2. Linken van de verschillende randpixels

We bespreken eerst twee methoden om randpixels te detecteren, nl. de *gradiënt* en de *discrete laplaciaan*.

3.5.1 Gradiënt

Een van de eerste methoden om randpixels te detecteren gebruikte maskers om de eersteordeafgeleide, de gradiënt, te berekenen (Figuur 3.12). Wanneer de afgeleide

wordt berekend, d.m.v. de discrete convolutie met het masker, worden enkel veranderingen die sterk genoeg zijn (thresholding) beschouwd als randpixels.



Figuur 3.12: Bepalen van randen door eerste orde afgeleide. (bron: [18])

Zonder in detail te gaan kunnen we stellen dat er enkele bekende maskers bestaan om de gradiënt te berekenen, nl. de Roberts-, Prewitt- en Sobelmaskers (ook operators genoemd). Ze hebben verschillende voor- en nadelen; zo heeft het Sobelmasker als voordeel dat het niet gevoelig is voor Gaussiaanse ruis, maar wel nog steeds voor *salt and pepper*-ruis (Figuur 3.13). Gaussiaanse ruis wordt gedefinieerd door een gaussiaanse verdeling, met een bepaalde standaardafwijking. Salt and pepper-ruis daarentegen is zeer agressieve ruis (sterke intensiteitsuitschieters), waardoor deze voor problemen kan zorgen bij het detecteren van randen.

Figuur 3.13: Guassiaanse ruis (links) (bron: [6]) en salt and pepper ruis (rechts) (bron: [18]).

Een uitgebreide analyse van de verschillende maskers valt buiten de opzet van deze masterproef. We beperken ons daarom tot het Sobelmasker, omdat het aan de basis

ligt van een andere belangrijke methode om randpixels te detecteren; zie verder de Canny edge detector. We kunnen stellen dat met de twee Sobelmaskers (Figuur 3.14), op een eenvoudige manier de gradiënt kan worden berekend. De gradiënt geeft ons de sterkte en richting van de verandering in een pixel. Bijgevolg kunnen we pixels met een sterke verandering, boven een bepaalde drempelwaarde, beschouwen als randpixels.

$$G = \sqrt{Gx^2 + Gy^2} \approx |Gx| + |Gy| \qquad \Theta = \arctan(\frac{Gy}{Gx})$$

-1	-2	-1
0	0	0
1	2	1

(a) Convolution template S1

-1	0	-1
-2	0	2
-1	0	1

(b) Convolution template S2

Figuur 3.14: Horizontaal (Gx) en verticaal (Gy) Sobelmasker. (bron: [20])

Uit [20] leren we echter dat een optimaal randdetectiealgoritme in staat moet zijn om zo veel mogelijk ruis te elimineren en enkel echte randen te detecteren. Er zijn verschillende methodes om ruis te onderdrukken; in Figuur 3.15 wordt een verbeterde versie van de Sobeloperator weergegeven.

Figuur 3.15: Lena image (links), standaard Sobel (midden), Sobel met ruis-onderdrukking (rechts). (bron: [20])

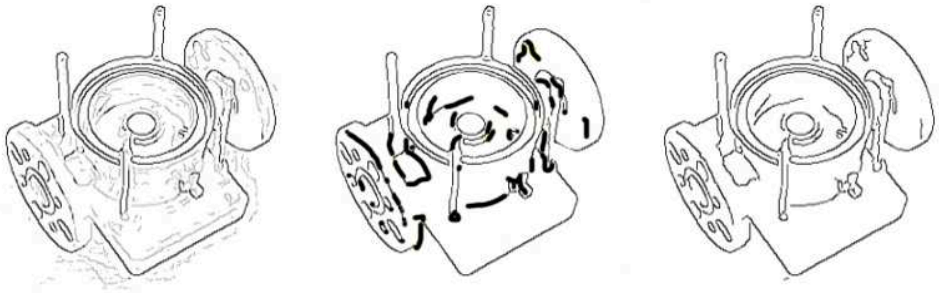
Een gelijkaardige, complexere maar vooral betere methode is de *Canny edge detector*, genoemd naar zijn uitvinder John Canny. De bedoeling van deze randdetector was vooral om de bestaande randdetectiealgoritmen te verbeteren. Canny specificeerde dat een randdetector aan de volgende drie voorwaarden moest voldoen:

- Foutpercentage: De randdetector mag enkel en alleen echte randen detecteren en moet in staat zijn om alle randen te vinden, m.a.w. de waarschijnlijkheid dat valse randen gedetecteerd worden moet zo laag mogelijk zijn.
- Lokalisatie: De gedetecteerde randen moeten zo dicht mogelijk bij de echte rand liggen.
- Antwoord: De rand detector mag niet meer dan één rand situeren waar slechts één enkele rand aanwezig is.

Het grote nadeel van deze methode is dat de berekeningen complex en zeer rekenintensief zijn t.o.v. de traditionele detectoren. [16] Door de gepaste threshold te gebruiken, worden enkel randen weergegeven met een hogere intensiteit dan de gekozen threshold.

Het algoritme bestaat uit vijf stappen:

1. Vervagen: de afbeelding wordt vervaagd om ruis te elimineren. Hiervoor wordt een Gaussiaans masker gebruikt.
2. Gradiënt: randen worden gevonden bij pixels waar de intensiteit sterk verandert.
3. Nonmaxima-suppressie: alleen lokale maxima mogen als randen gemarkeerd worden. Zwakke randen komen niet in aanmerking, en worden beschouwd als ruis.
4. Dubbele threshold: er worden twee thresholdwaarden opgegeven: een ondergrens en een bovengrens. Randen met intensiteiten lager dan de ondergrens worden onderdrukt, randen met intensiteiten groter dan de bovengrens worden gemarkeerd als sterke/echte randen.
5. Randhysteresis: randen met intensiteiten tussen het interval van de dubbele threshold worden enkel en alleen gemarkeerd als ze verbonden zijn met een echte rand. Hiervoor wordt een 8×8 masker gebruikt. De dikke zwarte randen op Figuur 3.16 liggen zeer dicht bij een echte rand en worden daarom ook beschouwd als een echte rand.



Figuur 3.16: Dubbele thresholding (links), rand hysteresis (midden), eind resultaat (rechts) (bron: [21])

3.5.2 Discrete Laplaciaan

Een andere manier om randpixels te bepalen is door het berekenen van de discrete Laplaciaan. De discrete Laplaciaan wordt gedefinieerd als de tweedeordeafgeleide:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

$$\frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \quad \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y)$$

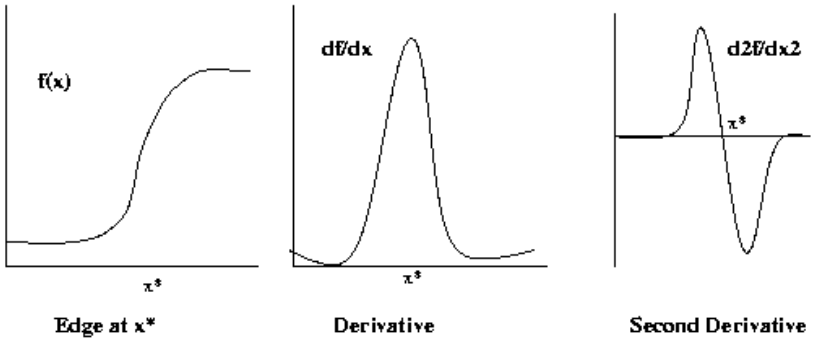
$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$
$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$\nabla^2 f(x,y) = f(x,y+1) + f(x,y-1) + f(x+1,y) + f(x-1,y) - 4f(x,y)$$

0	1	0
1	-4	1
0	1	0

Figuur 3.17: Laplaciaan (3x3) masker.

Ook bij het berekenen van de Laplaciaan wordt een masker (Figuur 3.17) gebruikt. Het grote verschil met de gradiënt is de manier waarop we bepalen of een pixel, een randpixel is. Bij de Laplaciaan berekenen we hiervoor de tweedeordeafgeleide en wordt gekeken waar nul overgangen, *zero crossings*, zijn (Figuur 3.18).

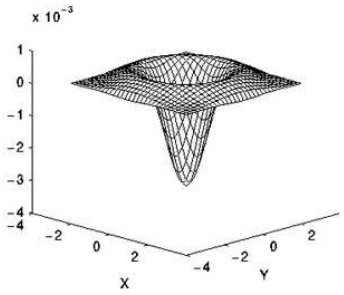


Figuur 3.18: Laplaciaan, tweedeordeafgeleide. (bron: [22])

Het nadeel van de Laplaciaan is dat de tweedeordeafgeleide zeer afhankelijk is van ruis (Figuur 3.21) en dat ze niet in staat is om de richting van de rand te bepalen. Het wegwerken van deze ruis kan men voorkomen door de afbeelding eerst te vervagen door middel van een Gaussiaans masker. Het Gaussiaans masker en Laplaciaan masker kunnen ook gecombineerd worden (Figuur 3.19). Men spreekt dan van de *Laplacian of Gaussian* (LoG), ook wel de *Marr-Hildreth-operator* genoemd (Figuur 3.20). [23]

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

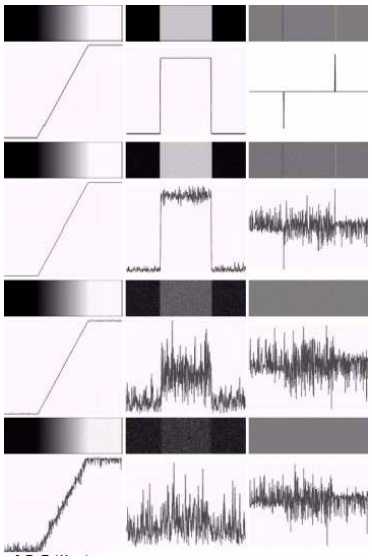
Figuur 3.19: LoG met σ, de standaardafwijking.



0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Figuur 3.20: LoG-masker. (bron: [23])

De voordelen t.o.v. de gradiëntoperatoren is dat ze dunnere randen oplevert en dat ze belangrijke *feature points* kan bepalen. [24] Ze kan naast het bepalen van randpixels ook gebruikt worden voor het verscherpen van een afbeelding.



Figuur 3.21: Laplaciaan zeer beïnvloedbaar voor ruis. (bron: [6])

3.5.3 Hough-transformatie

Verschillende methodes zijn voorgesteld om randpixels te detecteren. Zowel de gradient als de Laplaciaan kunnen ons hierbij helpen. Deze methodes bepalen of een pixel al dan niet een randpixel is, maar bepalen niet de relaties tussen de verschillende randpixels (*edge linking)*. Met andere woorden: de verschillende pixels weten niet of ze behoren tot dezelfde rand. Om dit probleem efficiënt op te lossen zijn enkele methoden bedacht. We bespreken de belangrijkste: de Hough-transformatie.

De bedoeling van de Hough-transformatie is om randen/lijnen in een afbeelding te situeren. Hierbij wordt een rand/lijn gedefinieerd als een rechte die door opeenvolgende randpixels gaat; het kan ook gebruikt worden om cirkels en ellipsen te detecteren. Er wordt bepaald of één of meerdere randpixels tot een rand behoren. We zullen vanaf nu het woord lijn gebruiken om verwarring te vermijden; met een lijn bedoelen

we dus ook een rand. Het idee dat hierbij wordt gebruikt is stemmen. De verschillende randpixels stemmen voor een bepaalde lijn. Als een lijn veel stemmen krijgt, wil dat zeggen dat ze verkozen is door verschillende randpixels. Wanneer een lijn genoeg stemmen/randpixels heeft, wordt de lijn aanvaard.

Een probleem hierbij is dat valse lijnen kunnen gedetecteerd worden door ruis (valse randpixels of stemmen), of dat niet alle randpixels op een lijn gevonden werden. Er zijn dan *gaps* tussen de verschillende randpixels. Gelukkig biedt de Houghtransformatie hier een oplossing voor.

De Hough-transformatie maakt gebruik van twee ruimtes:

- de afbeeldingsruimte (xy-ruimte), en
- de parameterruimte (mc-ruimte).

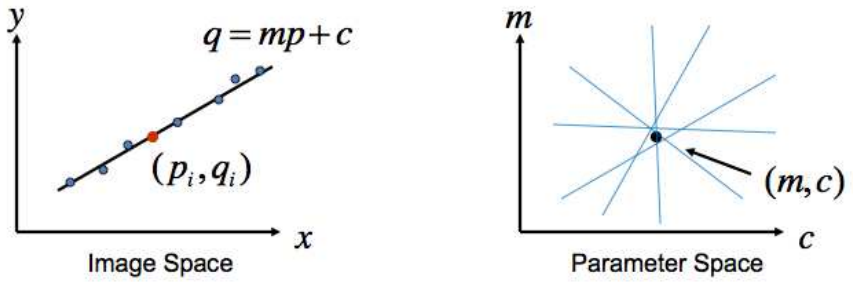
Een randpixel wordt in de afbeeldingsruimte gedefinieerd door een x- en y-coördinaat *(p,q)*, waarbij de vergelijking van een rechte door dat punt wordt gegeven door:

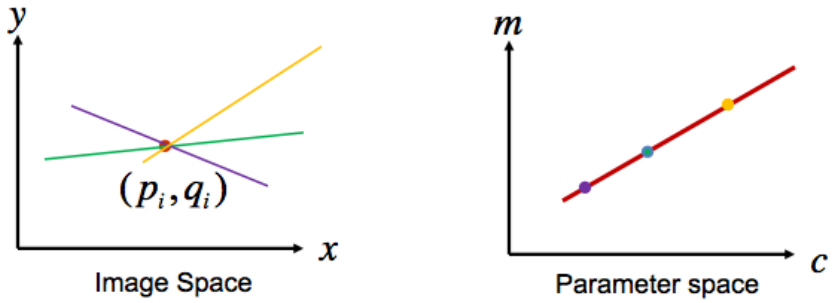
$$q = mp + c$$

Uit deze vergelijking volgt dat er oneindig veel rechten door de randpixel gaan, allemaal met verschillende waarden voor m en c maar wel door hetzelfde punt *(p,q)*. Als we de vergelijking herschrijven naar de vorm:

$$c = -mp + q$$

dan kunnen we voor het punt *(p,q)* een unieke rechte definiëren in de parameterruimte (mc-ruimte). Daarnaast zal elk ander punt/randpixel in de afbeeldingsruimte een rechte vertegenwoordigen in de parameterruimte; de omgekeerde redenering geldt ook. Wanneer twee randpixels parallel liggen ten opzichte van elkaar in de afbeeldingsruimte, zullen hun bijhorende rechten in de parameterruimte in een bepaald punt *(m,c)* kruisen (Figuur 3.22). Als dan veel rechten kruisen in de parameterruimte, wil dat zeggen dat veel randpixels parallel liggen ten opzichte van elkaar en dus op een rechte liggen in de afbeeldingsruimte, bepaald door de waarden m en c.



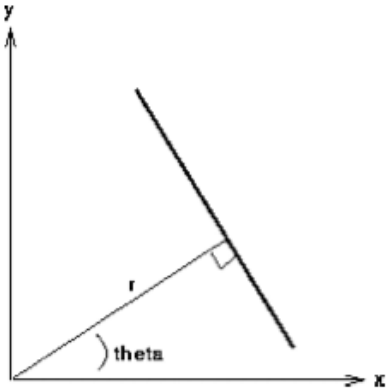


Figuur 3.22: Hough-transformatie met de afbeeldingsruimte en de parameter ruimte in functie van de interceptor. (bron: [25])

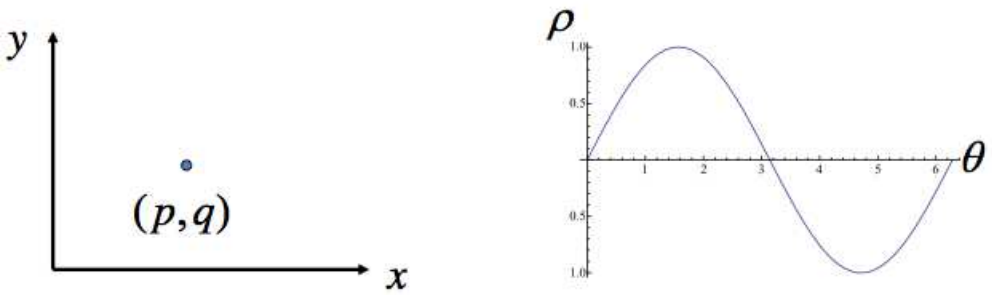
Een probleem bij deze notatie is het detecteren van verticale rechten: de waarde *m* (de helling) gaat naar oneindig en kan dus niet discreet worden voorgesteld. Daarom doet men beroep op een andere voorstelling, de normaalvergelijking van een rechte, gegeven door:

$$p\cos\theta + q\sin\theta = \rho$$

met θ de hoek tussen de x-as en de normaalvector van de rechte en ρ de kortste afstand van de oorsprong tot de rechte (Figuur 3.23).

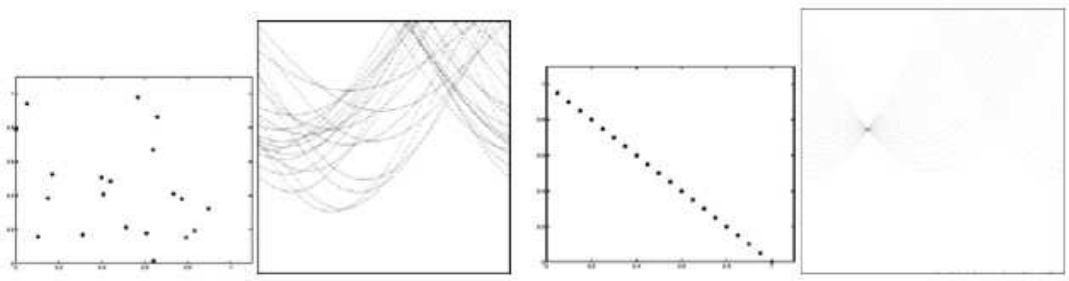


Figuur 3.23: Normaalvergelijking van een rechte. (bron: [25])



Figuur 3.24: Hough-transformatie met de xy-ruimte en de parameter ruimte in functie van de hoek θ. (bron: [25])

Het voordeel van deze voorstelling is dat nu ook verticale lijnen kunnen worden voorgesteld (Figuur 3.24). Het idee blijft hetzelfde: wanneer randpixels op dezelfde lijn liggen liggen in de afbeeldingsruimte zullen hun overeenkomstige sinusfuncties in de parameter ruimte elkaar snijden in een punt (θ,ρ) (Figuur 3.25). Aan de hand van dit punt kunnen we de overeenkomstige rechte in de afbeeldingsruimte construeren.



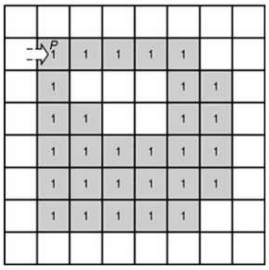
Figuur 3.25: Voorbeeld Hough-transformatie. (bron: [25])

De Hough-transformatie maakt het mogelijk om op een efficiënte manieren lijnen te detecteren; ze kan ook uitgebreid worden om cirkels of ellipsen te detecteren. De methode kan als basis dienen om rechthoeken te situeren in een afbeelding; bijvoorbeeld het label waarop het rolnummer gedrukt is.

3.6 Border following

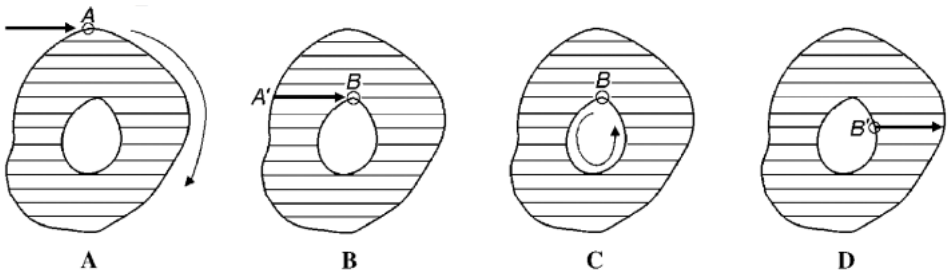
Border following [26] is één van de fundamentele technieken in het verwerken van binaire afbeeldingen. Het bepaalt de randen van de *connected components*. De rand van een component is belangrijke informatie die gebruikt kan worden voor objectherkenning of bijvoorbeeld als fingerprint voor het opslaan van afbeeldingen in een databank. Het heeft zeer veel toepassingen en is uitgebreid bestudeerd [27]. Verschillende algoritmen zijn voorgesteld. [26] [28]

Een border following-algoritme wordt steeds uitgevoerd op een binaire afbeelding. Hierbij wordt een verschil gemaakt tussen achtergrond- en objectpixels, respectievelijk 0- en 1-pixels (Figuur 3.26). Het algoritme volgt de verschillende objectpixels en is zo in staat om de randen van de verschillende componenten/objecten in een afbeelding te identificeren. De manier waarop het volgen gebeurt is eigen aan het algoritme.



Figuur 3.26: Border following. (bron: [29])

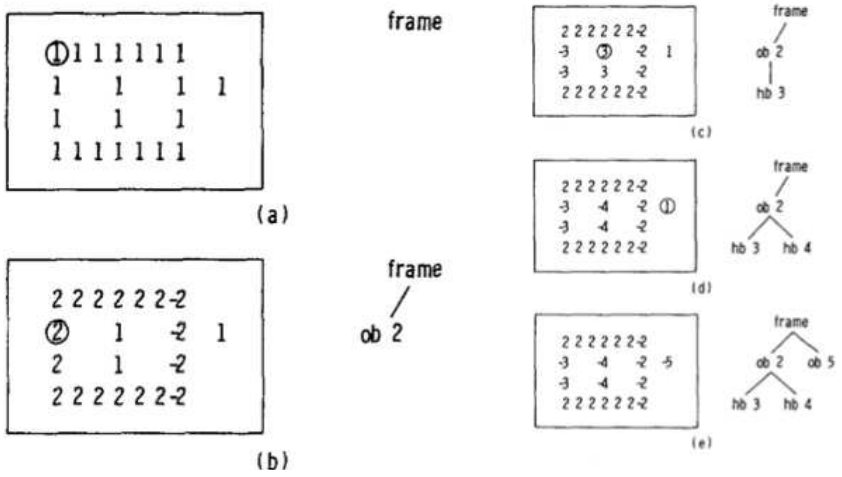
Buiten de 0- en 1-pixels wordt er ook onderscheid gemaakt in de soort rand. Er kunnen zowel *outer-borders* als *hole-borders* zijn (Figuur 3.27). Vaak is het voldoende om enkel de outer-borders van een component te bepalen, maar soms zijn ook de hole-borders nodig, zoals bij *optical character recognition*.



Figuur 3.27: Outer-border (A) en hole-border (C). (bron: [29])

In wat volgt wordt kort het border following-algoritme van Satoshi Suzuki en Keiichi Abe besproken [26] (Figuur 3.28).

Het algoritme start in de linkerbovenhoek van de afbeelding en overloopt de pixels van links naar rechts en van boven naar onder. De verschillende randen worden gelabeld met een randnummer. Het algoritme stopt als de pixel op de laatste rij en laatste kolom werd verwerkt.

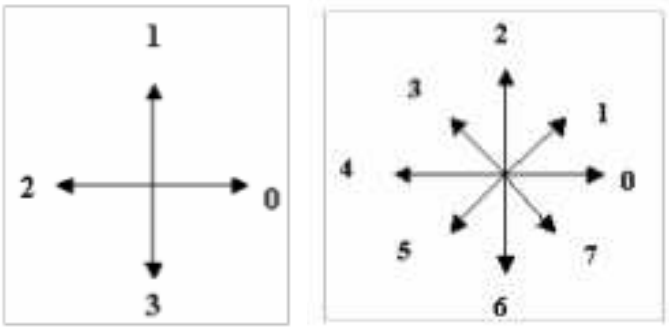


Figuur 3.28: Het border following-algoritme van S. Suzuki en K. Abe. (bron: [29])

Het algoritme voert een *scan* uit; van links naar rechts en van boven naar onder. Wanneer een eerste objectpixel (1-pixel) wordt aangetroffen (Figuur 3.28a), wordt een onderscheid gemaakt tussen de start van een hole-border of een outer-border (Figuur 3.29). Vervolgens wordt de border following, op de startpixel, uitgevoerd. Hierbij wordt in een 8- of 4-neighboorhood (Figuur 3.30), in tegenwijzerzin, gezocht naar een naburige objectpixel. Wanneer zo'n objectpixel gevonden is, wordt de pixel gelabeld met een randnummer en wordt dezelfde methode op de nieuwe pixel uitgevoerd, tot er geen naburige pixels meer zijn of tot opnieuw de startpixel gevonden wordt. Vervolgens wordt de scan verder uitgevoerd (op de pixel één positie verder van de

startpixel), en wordt gezocht naar een nieuwe startpixel. Wanneer een nieuwe startpixel gevonden is, wordt het randnummer geïncrementeerd en zal de border following opnieuw gestart worden vanuit deze nieuwe startpixel.

Figuur 3.29: Outer-border (links) en een hole-border (rechts). (bron: [29])



Figuur 3.30: 4- en 8-neighborhood. (bron: [29])

Merk op dat bij het labelen van de buitenste rand (Figuur 3.28b) er zowel positieve als negatieve randnummers zijn. Dit is één van de eigenschappen van het algoritme:

- Als de huidige rand tussen een 0-pixel *(i,j+1)* en een 1-pixel *(i,j)* is, dan krijgt de pixel *(i,j)* een negatief randnummer.
- In het andere geval krijgt de pixel *(i,j)* een positief randnummer als *(i,j)* nog niet eerder gevolgd is.

Het verschil tussen de positieve en negatieve randnummers is dat de negatieve randnummers niet opnieuw kunnen worden gekozen als startpixel. Wanneer de pixel rechtsonder wordt verwerkt, zullen alle randen gelabeld zijn. Vervolgens is men in staat om aan de hand van deze informatie bijvoorbeeld objecten te herkennen of belangrijke veronderstellingen te maken.

Een bijkomend voordeel van deze methode is dat deze informatie geeft over de topologische structuur. Zo is men in staat een hiërarchie op te bouwen van ouder- en grootouderborders. De methode kan eenvoudig uitgebreid geworden zodat enkel en alleen outer-borders worden bepaald. [26]

Figuur 3.31 geeft het border following-algoritme weer in pseudocode. Hierbij wordt verwezen naar Figuur 3.28.

Initieel is de afbeelding binair: de 0-pixels worden beschouwd als achtergrond en de 1-pixels als objecten. Tijdens het algoritme worden de 1-pixels overschreven met een number border (default NBD = 1). De functie *f(i,j)*, met *i* de rij en *j* de kolom, geeft

ons het randnummer van een pixel *(i,j)* terug. We starten links bovenaan de afbeelding.

1) Zoeken naar een nieuwe startpixel *(i,j)*
   1.1 Als *f(i,j)=1* en *f(i,j-1)=0,* verhoog NBD, *(i_2,j_2)←(i,j-1)* (outer-border).
   1.2 Anders als *f(i,j)=>1* en *f(i,j+1)=0, verhoog* NBD, *(i_2,j_2)←(i,j+1)* (hole-border).
   1.3 Anders ga naar stap 3.
2) Start de border following van uit de startpixel *(i,j)*.
   2.1 Zoek naar een niet 0-pixel in de neighborhood van *(i,j)* in wijzerzin startend bij pixel *(i_2,j_2)*. Wanneer een niet 0-pixel gevonden is noem het *(i_1,j_1)*, anders *f(i,j)* ← –NBD en ga naar stap 3.
   2.2 *(i_2,j_2)* ← *(i_1,j_1)* en *(i_3,j_3)* ← *(i,j)*.
   2.3 Zoek naar een niet 0-pixel in de neighboorhood van *(i_3,j_3)* in tegenwijzerzin startend bij de pixel *(i_2,j_2)*. Wanner een niet 0-pixel gevonden is noem het *(i_4,j_4)*.
   2.4 Aanpassen van f(i_3,j_3).
      a. Als de huidige rand tussen een 0-pixel *(i_3,j_3+1)* en een 1-pixel *(i_3,j_3)* is, dan *f(i,j)* ← –NBD.
      b. Anders *f(i,j)* ← NBD als *(i,j)* nog niet eerder gevolgd is.
   2.5 Als *(i_4,j_4)* = *(i,j)* en *(i_3,j_3)* = *(i_1,j_1)* opnieuw bij startpixel aangekomen, ga naar stap 3. Anders *(i_2,j_2)*←*(i_3,j_3)* en *(i_3,j_3)*←*(i_4,j_4)* en ga terug naar stap 2.3.
3) Ga naar stap 1 als niet laatste pixel van afbeelding, *(i,j)* ← *(i,j+1)*.

Figuur 3.31: Pseudocode algoritme van Satoshi Suzuki en Keiichi. (bron: [29])

3.7 Bibliotheken voor beeldverwerking

Om bovenstaande beeldverwerkingsbewerkingen te kunnen uitvoeren is er nood aan een computervisiebibliotheek. Er dient een keuze gemaakt te worden tussen verschillende bibliotheken, waarbij de verschillende voor- en nadelen van de bibliotheken worden bekeken en vergeleken. Er worden drie bibliotheken bekeken: VXL, LTI en OpenCV.

3.7.1 VXL

De *Vision "Something" Library* (VXL) [30] is een collectie van C++ bibliotheken voor onderzoek en implementatie van computervisiefunctionaliteit. Deze is ontworpen door TargetJr [31] met als doel een snel en betrouwbare software te ontwikkelen. VXL is geschreven in de programmeertaal C++ met het oog op platformonafhankelijkheid. Het is uitgegeven onder het VXL-licentie [32].

VXL bestaat uit vier core-bibliotheken:

- VNL (numerics): matrices, vectoren, etc.
- VIL (imaging): laden, opslaan en bewerken van afbeeldingen in verschillende formaten.
- VGL (geometry): geometrie voor punten, curves en andere objecten in 1D, 2D en 3D.
- VSL (streaming I/O).

Naast deze 4 core-bibliotheken bestaan er nog andere bibliotheken: numerieke algoritmen, video manipulatie, feature tracking, camera geometrie, etc.

De documentatie is beperkt. [33] De bibliotheek wordt nog steeds onderhouden [34]. De community is klein en er is relatief weinig informatie over te vinden.

3.7.2 LTI

LTI [35] is een objectgeoriëntieerde bibliotheek, geschreven in de programmeertaal C++, die bestaat uit algoritmen en datastructuren die frequent gebruikt worden bij computervisie. Hij is ontworpen aan de Aachen Universiteit van Technologie in Duitsland [36] in kader van enkele onderzoeksprojecten omtrent robotica, objectherkenning en gebarenherkenning. Hij is uitgegeven onder het LGPL-licentie.

Het doel van LTI is om het onderhoud en uitwisseling van code tussen de verschillende programmeurs te vereenvoudigen. Daarnaast werd LTI ontwikkeld met het oog op realtime applicaties.

De applicatie bestaat uit een honderdtal verschillende klassen: lineaire algebra, classificatie en clustering, beeldverwerking, etc.

De documentatie is uitgebreider dan bij VXL en is ook beter gestructureerd. [37] De community is vrij beperkt en de bibliotheek wordt ook nog steeds onderhouden.

3.7.3 OpenCV

OpenCV [38] is een open source computervisiebibliotheek en is geschreven in de programmeertalen C en C++. Hij werd ontworpen met het oog op efficiëntie en realtime applicaties. Het doel van OpenCV is om een eenvoudige API aan te bieden die het mogelijk laakt om snel gesofisticeerde computervisie software te ontwikkelen.

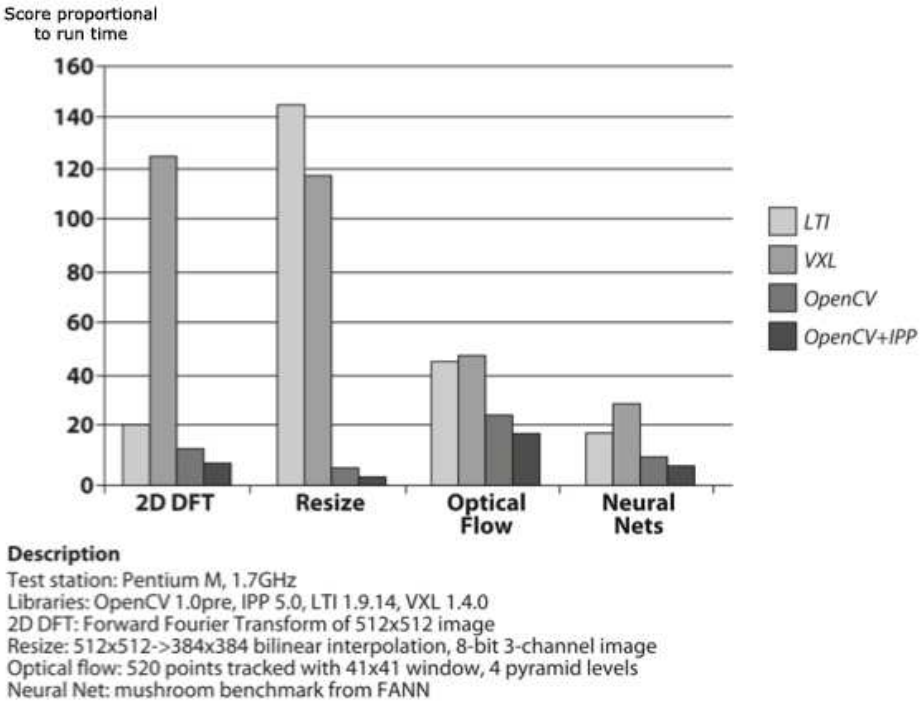
OpenCV bevat meer dan 500 functies over de verschillende afdelingen van computervisie: productinspectie, biologisch onderzoek, beveiliging, robotica, machineleer, etc.

De documentatie is zeer uitgebreid. OpenCV wordt gebruikt door verschillende grote bedrijven (IBM, Microsoft, Intel, Sony, Siemens en Google) en verschillende onderzoekscentra (Stanford, MIT, CMI, Cambridge en INRIA). De eerste release dateert van 1999 en in juni (2013) laatsleden verscheen versie 2.4.6.

In tegenstelling tot de andere bibliotheken is er voor OpenCV zeer goede documentatie beschikbaar, ook de community is zeer groot. Daarnaast mag worden verondersteld dat de bibliotheek in de nabije toekomst nog actief zal worden onderhouden.

3.7.4 Conclusie

In Figuur 3.32 worden de verwerkingstijden van de drie bibliotheken vergeleken in functie van vier verschillende bewerkingen. Hieruit kunnen we afleiden dat OpenCV zeer goed scoort ten opzichte van de andere bibliotheken.



Figuur 3.32: Vergelijking van de drie verschillende beeldverwerkingsbibliotheken. (bron: [39])

Eén van de vereisten van deze masterproef is dat de applicatie in C# geschreven wordt. Dit wil zeggen dat OpenCV en ook de vorige bibliotheken geport zouden moeten worden naar de programmeertaal C#. Gelukkig bestaan er voor de bibliotheek OpenCV, dankzij zijn populariteit, een aantal wrapper-bibliotheken. Hierdoor zijn we in staat om toch de functies van OpenCV te kunnen aanspreken vanuit de programmeertaal C#. We bespreken de belangrijkste en de meest uitgebreide, Emgu CV.

3.7.4.1 Emgu CV

Emgu CV [40] is een .NET-wrapper voor OpenCV en maakt het mogelijk dat OpenCV-functies kunnen aangesproken worden vanuit .NET-compatibele programmeertalen zoals C#, VB, etc. Hij is uitgegeven onder het GPL-licentie.

Ten opzichte van andere wrappers (*opencvdotnet*, *SharperCV*, ...) is de community vrij groot en is er veel documentatie over de wrapper beschikbaar. Emgu CV biedt (wrapped) meer functies aan dan andere wrappers en wordt regelmatiger geüpdatet naar recentere versies van OpenCV. We kiezen daarom om Emgu CV te gebruiken voor de uitwerking van de applicatie.

4 Situering label

Nu we de verschillende problemen besproken hebben en weten waarom de herkenning nodig is, zullen we in dit en het volgende hoofdstuk enkele oplossingsmethoden bekijken die we kunnen gebruiken om het rolnummer te herkennen. Hieruit kunnen we vervolgens een oplossingsmethode bepalen, die het meest geschikt is voor de herkenning van het rolnummer.

De herkenning kan opgesplitst worden in twee deelfasen:

1. Situering van het label (hoofdstuk 4).
2. Herkenning van het rolnummer (hoofdstuk 5).

Vervolgens zullen we de meest geschikte methoden voor elke fase combineren tot een volwaardig geheel dat in staat is om de rolnummers te herkennen.

4.1 Situering

Het is niet gewenst om op zoek te gaan naar het rolnummer in de volledige afbeelding, want dit zou ervoor zorgen dat

- ongewenste karakters gedetecteerd worden en
- de algoritmen voor het herkennen van het rolnummer inefficiënt worden.

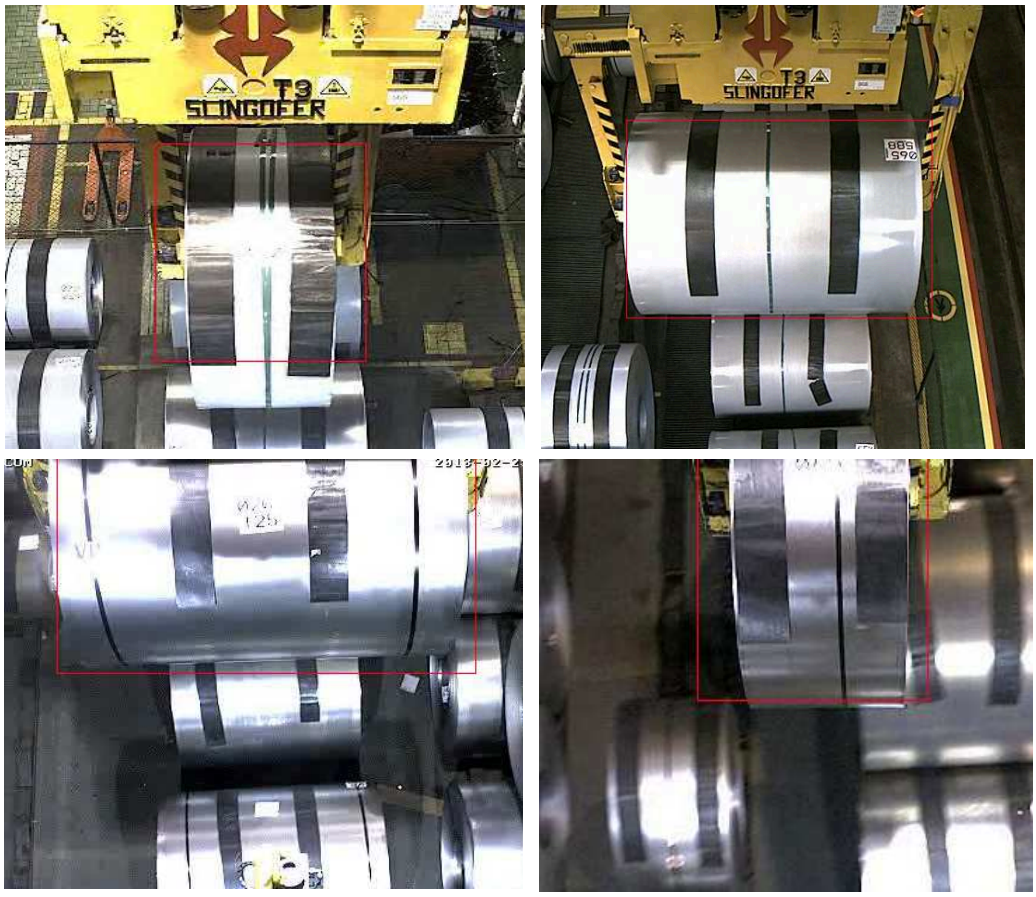
Daarom zullen we een bereik bepalen dat zo klein mogelijk is maar toch nog groot genoeg, zodat enkel en alleen het rolnummer herkend wordt. Dit bereik zullen we de *situering* van het label noemen. We bespreken in de volgende paragraaf enkele ideeën die dit mogelijk kunnen maken.

4.2 Situering rollen op camerabeelden

Wanneer we de camerabeelden analyseren, kunnen we opmerken dat een rol zich steeds in het midden van het (camera)beeld bevindt. We kunnen dit voordeel gebruiken om de situering te beperken. Hierdoor worden de operaties die we zullen gebruiken in het vervolg van deze masterproef efficiënter en preciezer. Alleen is de vraag waar deze ruimte zich bevindt. De hoek tussen de camera en de kraan is niet voor alle kranen gelijk, dus is het mogelijk dat hierdoor een rol hoger of lager in beeld komt dan bij een andere camera. Daarnaast hebben de rollen een verschillende diameter en breedte, waardoor rollen met een grotere diameter, zich dichter bij de camera zullen bevinden en dus ook een groter bereik nodig hebben. Uiteraard geldt dit principe ook voor rollen die breder zijn.

We zullen dit bereik handmatig bepalen voor de productiehal Sikel, omdat deze oorspronkelijk de vragende partij was voor de rolnummerherkenning. Vervolgens zal dit ook voor de productiehal in Gent gebeuren.

We zullen de camerabeelden van alle automatische kranen, in de productiehal Sikel, bekijken: LK201, LK202 en LK203 (waarbij LK staat voor loopkraan). Op Figuur 4.1 worden enkele camerabeelden weergegeven.



Figuur 4.1: Bepalen optimaal bereik.

Tabel 4.1: Optimaal bereik

Diameter / Breedte / Kraan	(±600mm,±600mm)	(±2000mm,±2000mm)
LK201	(250,130) (400,300) 150px×170px	(160,130) (490,330) **330px×200px**
LK202	(230,0) (380,150) 150px×150px	(140,0) (490,180) **350px×180px**
LK203	(200,130) (390,310) 190px×180px	(160,50) (490,260) **350px×210px**

Aangezien we geïnteresseerd zijn in het grootste bereik, kijken we enkel naar de rollen met de grootste diameter en breedte en dit voor de verschillende kranen. We raadplegen hiervoor de opname van de verplaatsing van een rol (met als diameter 2m en breedte 2m), en dit voor de drie automatische kranen. Vervolgens nemen we uit elke opname een frame en bepalen we handmatig de zone waarin de rol zich bevindt (Tabel 4.1).

We kunnen besluiten dat de camera van LK202 lager gericht is en daarom de rol hoger in beeld komt. We nemen de unie van de drie verschillende kranen om het grootste bereik te bepalen: (140,0) (490,330) = 350×330 (115.500 pixels). Aangezien de resolutie van de camerabeelden 640×480 (307.200 pixels) is, kunnen we besluiten dat 62% van het camerabeeld wordt uitgesloten.

Het bepalen van dit bereik is afhankelijk van de productiehal en ook van de verschillende kranen. We zouden dan ook kunnen voorzien dat per kraan een gebied wordt bepaald, waardoor de operaties nog efficiënter worden. Voor de eenvoud doen we dit niet. We verwachten echter geen noemenswaardige verbeteringen. Daarnaast is het verschil tussen de verschillende productiehallen wel groot. Daarom is het wel interessant om per productiehal een gebied te bepalen.

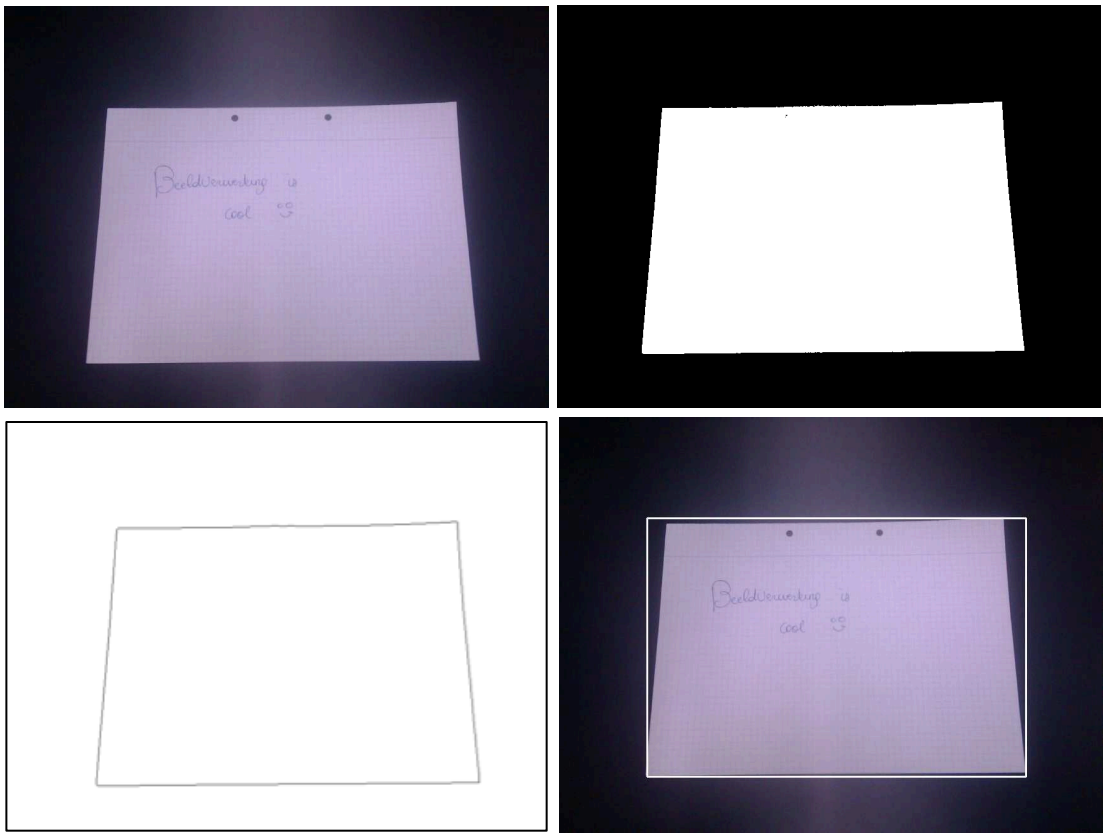
Omdat we tijdens deze masterproef camerabeelden van verschillende productiehallen analyseren (Gent en Sikel), zullen we voorlopig geen rekening houden met deze optimalisatie. De algoritmen worden (voorlopig) uitgevoerd op de volledige camerabeelden.

4.3 Randdetectie

Nu we de situering sterk beperkt hebben, kunnen we verder op zoek gaan naar het rolnummer. Zoals in 3.5 vermeld, kunnen randen ons belangrijke informatie opleveren. Als we het rolnummer van dichtbij bekijken, zien we dat het label waarop het rolnummer gedrukt is, steeds rechthoekig is en een witte kleur heeft (Figuur 4.2).

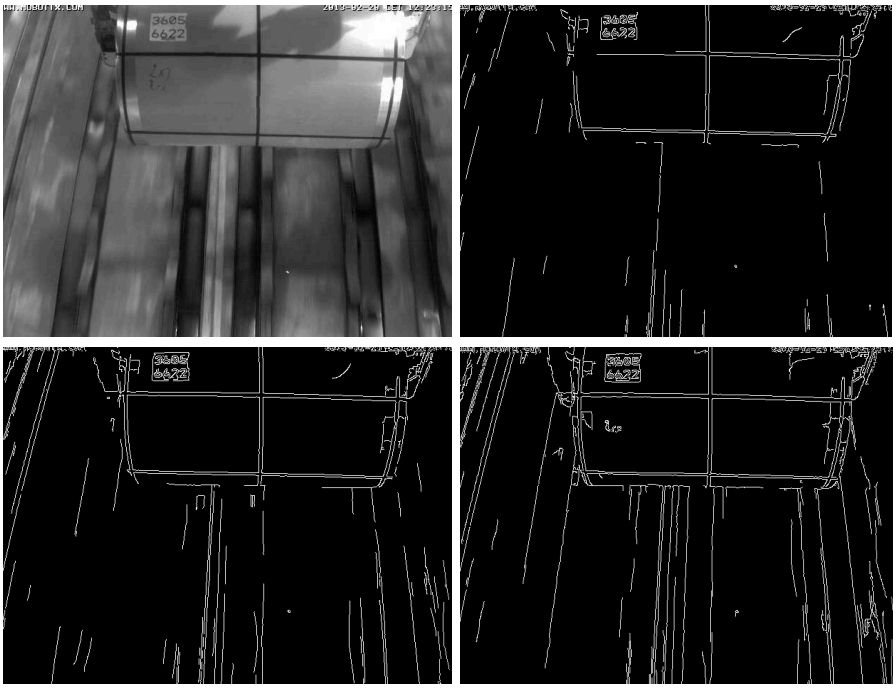
Figuur 4.2: Rolnummer.

Het detecteren van een witte rechthoek is eenvoudig en hiervoor kunnen we beroep doen op de Canny edge detector en de Hough-transformatie. De Canny edge detector zal eerst de randpixels, sterke veranderingen, bepalen. Vervolgens zal de Hough-transformatie, de effectieve lijnen detecteren en kunnen we besluiten dat de lijnen een rechthoek vormen, die aan de verhouding van het label voldoet (Figuur 4.3).



Figuur 4.3: Orginele afbeelding, Threshold Otsu's method [4], Canny edge detector, Hough-transformatie.

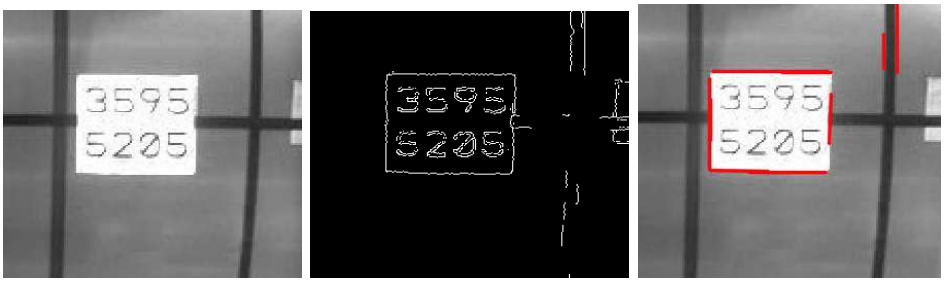
Als we de Canny edge detector toepassen op de (camera)beelden, worden de randen van het label gedetecteerd als het label goed zichtbaar is. Wanneer de sterke overgang tussen het label en de rol vermindert, krijgen we minder goede resultaten. Dit komt door de dubbele threshold; een lagere bovengrens zal ervoor zorgen dat het label ook gedetecteerd wordt bij minder sterke overgangen, maar zal er ook voor zorgen dat meer valse randen gedetecteerd worden (Figuur 4.4).



Figuur 4.4: Canny edge detector, zwakke overgangen minder accuraat.

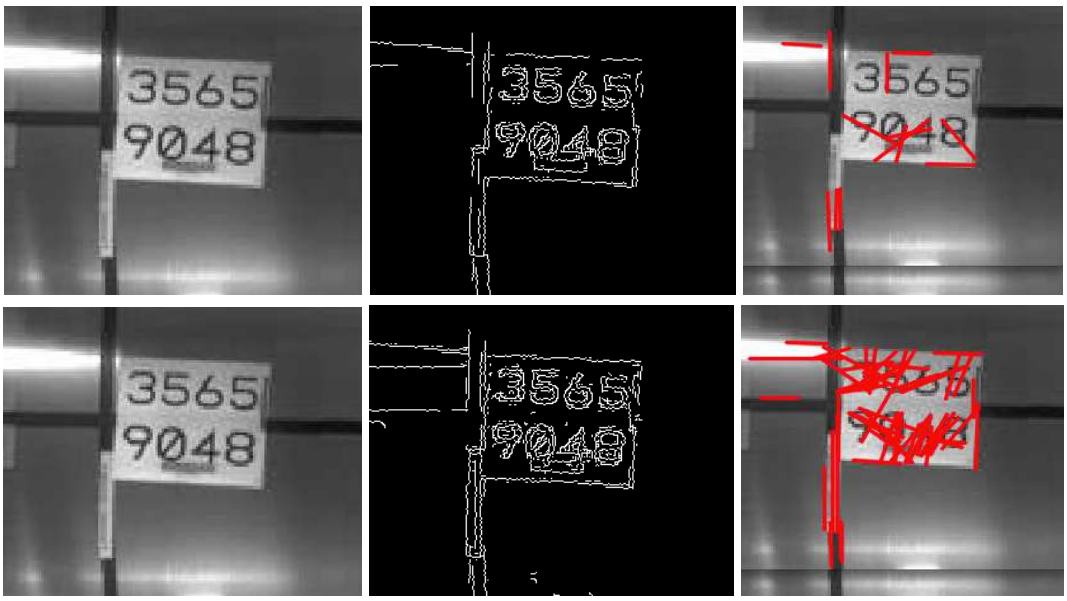
Als we hierna de lijnen proberen te detecteren m.b.v. de Hough-transformatie, krijgen we niet de resultaten die we verwachten. De randen van het label worden moeilijk gedetecteerd. De reden hiervoor is de lage resolutie, er zijn hierdoor veel *gaps* tussen de lijnstukken. Op Figuur 4.5 zijn de Canny edge detector en de Hough-transformatie toegepast met volgende parameters (proefondervindelijk vastgesteld):

- Canny edge detector: voor de dubbele threshold wordt een ondergrens=150 (O) en bovengrens=200 (B) gekozen.
- Hough-transformatie: minimale intersecties=1 (MI), minimum lengte=32 (ML) en maximum gaps=8 (MG).



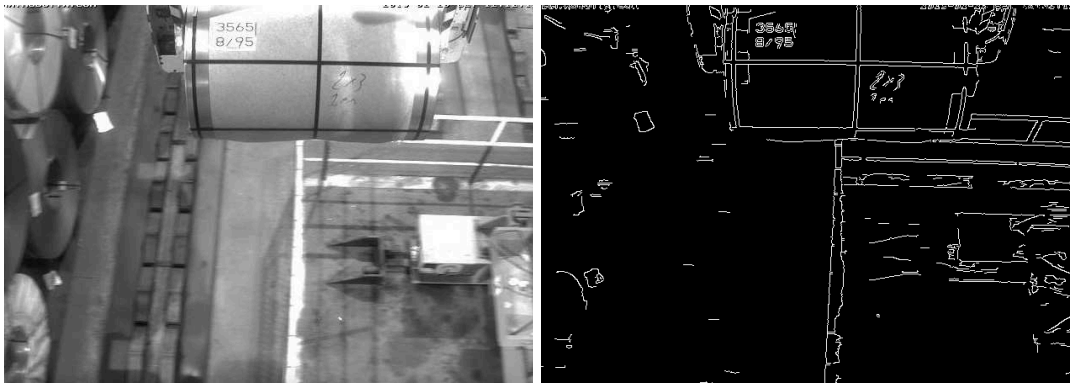
Figuur 4.5: Origineel label, Canny edge detector en Hough-transformatie.

De Canny edge detector kan ook bij minder sterke overgangen de randen van het label situeren. Maar hier is de Hough-transformatie niet in staat om de lijnen te bepalen. We passen dezelfde parameters toe als bij de vorige situatie en merken op dat de lijn niet volledig gedetecteerd wordt door de Canny edge detector. De Hough-transformatie geeft dan ook een bedroevend resultaat. Wanneer we de parameters van de Canny edge detector aanpassen naar O=100 en B=150, worden de randen opnieuw gedetecteerd. Het toepassen van de Hough-transformatie geeft ons nu nog slechtere lijnen (Figuur 4.6). Door de MG te verhogen zal de lijn bovenaan het label waarschijnlijk wel gedetecteerd worden, maar ook meer valse lijnen.



Figuur 4.6: Origineel label, Canny edge detector en Hough-transformatie.

Als het label overbelicht of onderbelicht is, zijn er geen sterke overgangen meer, wat wil zeggen dat de Canny edge detector de randen van het label helemaal niet kan detecteren (Figuur 4.7). Het heeft dan ook geen zin om de Hough-transformatie hierop op toe te passen. Het gebeurt vaak dat rollen overbelicht worden door spots en zoals we zien is bij overbelichting het rolnummer in veel gevallen nog steeds leesbaar.



Figuur 4.7: Canny edge detector, bij overbelicht label.

4.3.1 Resultaten

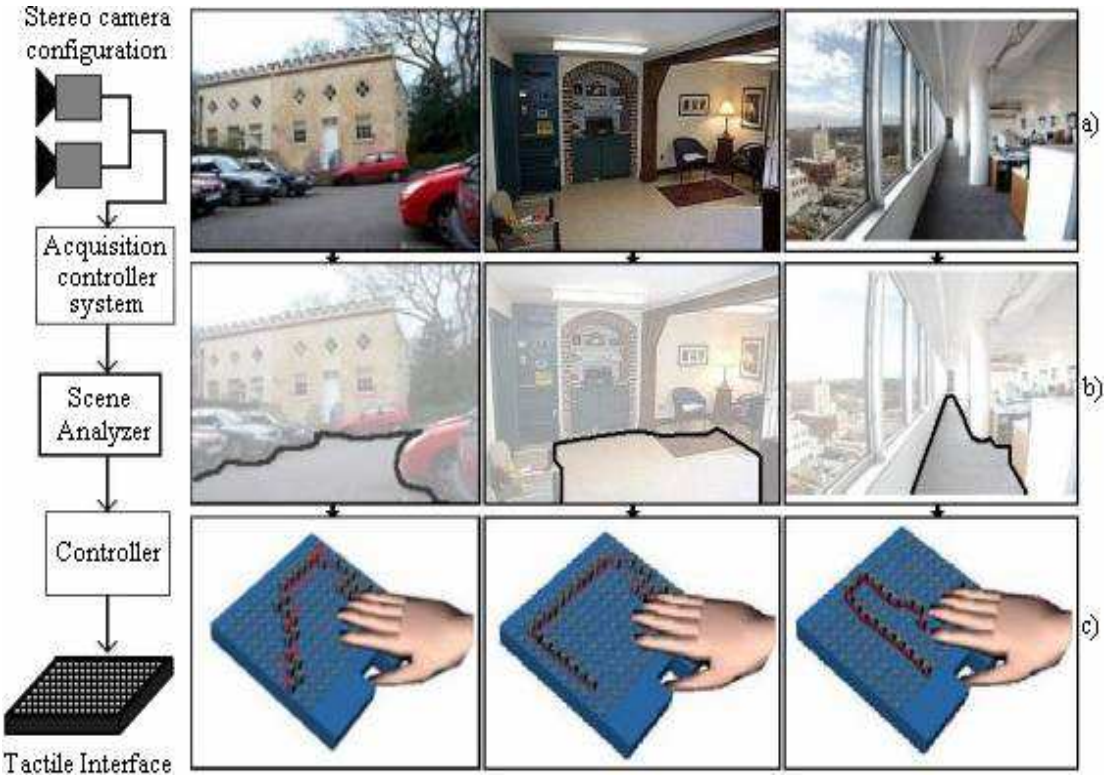
Het herkennen van de randen a.d.h.v. de randdetectie voldoet niet aan de verwachtingen. Door de lage resolutie en vooral de over- en onderbelichting wordt het herkennen van het label (de rechthoek) zo goed als onmogelijk. Ook de parameters van de Hough-transformatie moeten zeer nauwkeurig gekozen worden. Het evalueren van deze methode aan de hand van de evaluatieset wordt daarom niet gedaan.

4.4 Tekstdetectie

Uit de vorige methode, randdetectie, hebben we besloten dat deze te wensen overlaat bij over- en onderbelichting. We konden in deze omstandigheden het label niet detecteren, ook al was het rolnummer wel goed zichtbaar. Hierdoor zouden we beelden waarop het rolnummer wel goed leesbaar is, wat een positieve wending kan geven, negeren en als onbruikbaar verklaren.

We hebben tot nu toe het probleem van het herkennen van het rolnummer, verlegd naar het herkennen van het label, omdat het label een gunstige vorm had voor bepaalde beeldverwerkingsalgoritmen. Jammer genoeg is deze methode niet voldoende; daarom zullen we een andere methode bekijken.

Naast in het detecteren van randen is er ook een groeiende interesse in het detecteren van tekst in afbeeldingen en video. Ook hier is er veel onderzoek naar gedaan en zijn enkele algoritmen voorgesteld om dit soort problemen aan te pakken. Er worden zelfs wedstrijden, zoals *ICDAR 2011* [41], gehouden waarin verschillende algoritmen het tegen elkaar opnemen. Het detecteren en herkennen van tekst in een video-opname kan handig zijn om snel iets over de context van een video te weten te komen. Hierdoor kan men illegaal videomateriaal of plagiaat op het internet onderscheppen. Recent onderzoek heeft een tekstdetectie algoritme voor het project *Intelligent Glasses* voorgesteld (Figuur 4.8) [42]. Het doel van het algoritme is om blinde en slechtziende personen te helpen door tekst uit het dagelijkse leven te detecteren. Hierdoor zijn blinde en slechtziende mensen in staat om hun weg te vinden in grote steden of een menukaart te lezen. De slechtziende of blinde mensen kunnen via een tactiele interface de tekst lezen (brailleschrift).



Figuur 4.8: Intelligent Glasses project. (bron: [42])

Er dient een duidelijk onderscheid gemaakt te worden tussen tekstdetectie en tekstherkenning. Tekstdetectie bepaalt de plaatsen waar tekst aanwezig is en teksther-kenning vertaalt tekst gevonden in een afbeelding naar digitale tekst. Tekstdetectie is nodig opdat de teksther-kenning goed zou werken. We hebben er dan ook alle belang bij dat tekstdetectie algoritmen zo optimaal mogelijk werken.

De verschillende methodes die tekstdetectie mogelijk maken kunnen onderverdeeld worden in drie groepen:

- Connected components
- Randen
- Textuur

Zo stellen Yuan en Tan een methode voor om met behulp van de Canny edge detector paragrafen te detecteren uit nieuwskranten die onderhevig zijn aan ruis. [43] Alata en Al-Shabi gebruiken kleurreductie waarbij kleurenfrequenties onder een bepaalde threshold weggewerkt worden. [44] Hanif en Prevost maken gebruik van de textuur van karakters en neurale netwerken. [42] Pietikainen en Okun detecteren tekst in een complexe achtergrond door gebruik te maken van de Sobeloperator. [45] Dutta en Tan maken gebruik van de gradiënt en morfologische operaties (Figuur 4.9). [46]

Bijna al deze methodes gaan uit van volgende principes, namelijk dat tekst enkele belangrijke karakteristieken heeft:

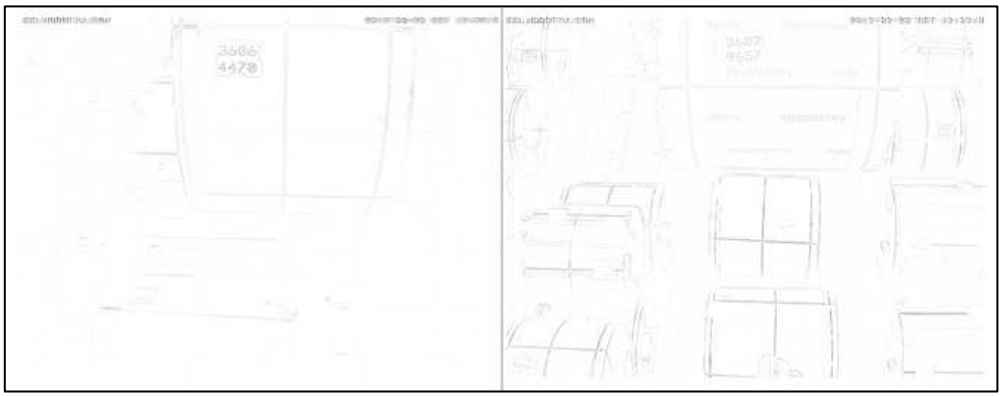
1. Geometrie
    - Tekst heeft vaak dezelfde grootte.
    - Omdat tekst leesbaar moet zijn, is ze gealigneerd.
    - Karakters in een tekst hebben een gelijke afstand.
2. Kleur
    - Karakters hebben vaak dezelfde kleur.
3. Randen
    - Doordat tekst goed leesbaar moet zijn, heeft ze sterke randen.

Omdat afbeeldingen vaak een complexe achtergrond hebben, is tekst detecteren een complexe opgave. Er bestaan methoden die met binaire, grijswaarden- of gekleurde beelden werken. Onderzoek toont aan dat tekst detecteren moeilijker is in gekleurde afbeeldingen, waardoor verschillende methode gebruikmaken van grijswaardenafbeeldingen. [45] Ook worden methodes die gebruikmaken van connected components vaak als onbetrouwbaar beschouwd omdat ze rekening houden met gelijkaardige eigenschappen zoals de kleur. Daarentegen zijn de textuurgebaseerde methodes zeer rekenintensief maar wel robuust. Het is dan ook vaak de reden waarom men voor methodes kiest die gebruikmaken van randen omdat deze operaties snel en eenvoudig zijn. [46]

Figuur 4.9: Tekst detectie met de gradiënt en morfologische operaties. (bron: **[46]**)

Opnieuw zijn randen zeer belangrijk en ook hier bij tekstdetectie kunnen ze ons een heleboel informatie opleveren. De discrete Laplaciaan (3.5), de tweedeordeafgeleide, kan belangrijke feature points bepalen. Het bepaalt de pixels waar sterke veranderingen worden waargenomen.

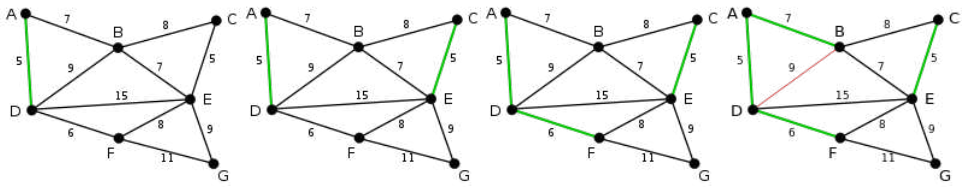
Als we de LoG (3.5) gebruiken op de (camera)beelden dan zien we volgend resultaat (Figuur 4.10). We zien dat de Laplaciaan zeer sterke intensiteiten geeft bij het rolnummer, wat wil zeggen dat hier sterke overgangen zijn. Dit is vanzelfsprekend, want mocht dit niet zo zijn, dan zouden we niet in staat zijn om het rolnummer te kunnen waarnemen.



Figuur 4.10: De Laplaciaan of Gaussian.

Zoals hiervoor vermeld is een eigenschap van tekst dat veel karakters zich dicht bij elkaar bevinden en dus veel randpixels vertonen. Wanneer we zoeken naar een beperkt gebied, waarin veel punten dicht bij elkaar liggen, dan zou dit een mogelijke plaats kunnen zijn voor het label, maar dit is uiteraard niet genoeg en vanzelfsprekend. Als we met dit idee even het beeldverwerkingsaspect vergeten en het probleem modelleren naar een meer algemene vorm, namelijk die van grafen, dan beschikken we meteen over een grote collectie efficiënte algoritmen.

Wanneer we het idee vertalen in de context van grafen, dan lijkt een zeer voor de hand liggende oplossing het algoritme voor clustering. Dit algoritme is een verfijnde versie van het Kruskal-algoritme, een oud en vernuftig algoritme dat een minimale overspannende boom (MOB) opstelt (Figuur 4.11). Op het eerste zicht lijkt het opstellen van een minimale overspannende boom wat ver gezocht. Daarom zullen we een korte inleiding geven en daarbij de aanleiding naar clustering toe.



Figuur 4.11: Het algoritme van Kruskal.

4.4.1 Kruskal

Het algoritme werkt als volgt [47]:

1. Maak een nieuwe graaf met dezelfde knopen als de originele graaf zonder de verbindingen.
2. Maak een verzameling S met alle verbindingen van de originele graaf.
3. Bepaal de verbinding met minimale gewicht uit de verzameling S.
4. Als de verbinding met minimale gewicht, twee knopen verbind die elkaar nog niet konden bereiken in de nieuwe graaf, dan voegen we deze verbinding toe in de nieuwe graaf.

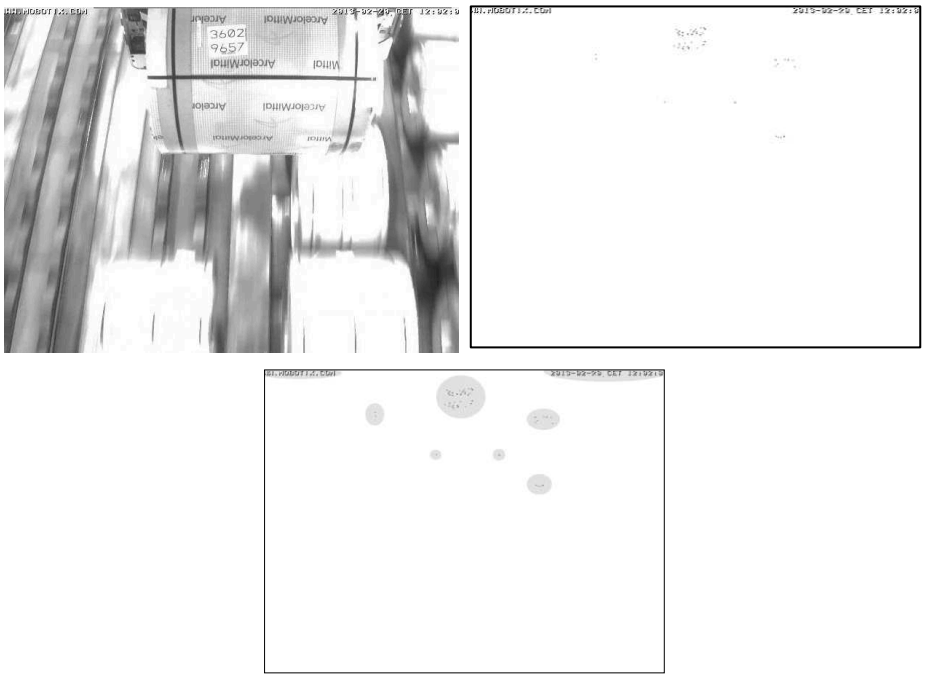
5. We verwijderen de verbinding met minimaal gewicht uit de verzameling S, en gaan terug naar 3, zolang niet $n - 1$ keer toegevoegd.
6. Na $n - 1$ keer toevoegen kunnen alle knopen elkaar bereiken en is er een MOB opgesteld.

Nu we het principe van Kruskal beet hebben, kunnen we het verder verfijnen. Een *cluster* wordt in de astronomie gedefinieerd als een groep sterrenstelsels die door de onderlinge zwaartekracht bij elkaar gehouden worden. Hierbij kan een cluster één of meerdere sterrenstelsels bevatten. In tegenstelling tot Kruskal, spreekt men bij clustering over de afstand tussen twee knopen en niet over het gewicht. De afstand is een vrij algemeen gegeven, en kan gedefinieerd worden als bijvoorbeeld de euclidische afstand, maar ook als het aantal mutaties tussen een paar DNA-sequenties. Wanneer we terug kijken naar het algoritme van Kruskal zien we dat na toevoegen van de $n - 1^{e}$ verbinding, we een minimaal overspannende boom hebben opgesteld, maar dus ook één cluster. Wat als we nu vroeger stoppen?

Stel dat we het algoritme zo aanpassen dat we stoppen bij het toevoegen van de $n - 2^{de}$ verbinding. We krijgen dan twee clusters, waarvan elke cluster de knopen bevat die het dichtst bij elkaar liggen. Het algoritme bepaalt dus het aantal clusters k door het aantal verbindingen $n - k$ die worden toegevoegd. Dit idee is bijna zoals wij het willen toepassen, maar er ontbreekt echter nog een belangrijk detail.

Wij zijn niet geïnteresseerd in het aantal clusters maar wel in de grootste cluster, nl. die het meeste knopen bevat die het dichtst bij elkaar gelegen zijn. Hiervoor voeren we een nieuwe conditie toe. We voegen namelijk enkel verbindingen toe waarvan hun afstand kleiner is dan een maximale afstand.

Als we het clusteringalgoritme toepassen op onze situatie, dan kunnen we voor alle punten gegeven door de Laplaciaan, een knoop aanmaken. Vervolgens bepalen we alle clusters en nemen we hieruit de grootste cluster (Figuur 4.12).



Figuur 4.12: Clusters.

Het bijhouden van een nieuwe gegevensstructuur, een graaf, lijkt wat overbodig. Gelukkig kunnen we hiervoor gebruikmaken van beeldverwerkingstechnieken zoals besproken in de volgende paragraaf

4.4.2 Adaptieve threshold

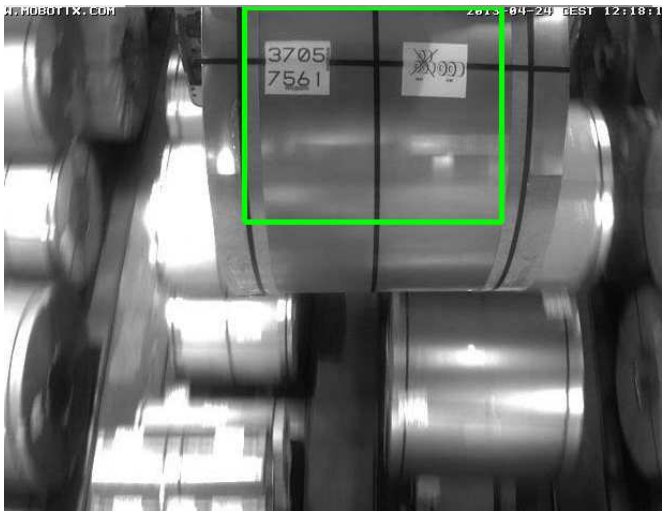
Voor we de beeldverwerkingstechniek (die de clustering mogelijk maakt) bespreken, behandelen we eerst nog een belangrijk detail. Door de Laplaciaan toe te passen krijgen we de sterke veranderingen in een afbeelding: hoe sterker de verandering, hoe hoger de intensiteit in dat punt (Figuur 4.10).

We hebben in het begin van deze paragraaf besproken dat tekst sterke veranderingen vertoont. Hierdoor zou een threshold kunnen bepaald worden, zodat enkel de sterkste veranderingen van de Laplaciaan overblijven. De threshold zorgt dat ruis, minder sterke veranderingen, wordt verwijderd en enkel de randpixels van het rolnummer worden behouden.

Het bepalen van zo'n threshold is niet triviaal. Bij onderbelichting zijn de overgangen immers zeer zwak en bij overbelichting sterk. Er dient een compromis gesloten te worden tussen deze twee extrema.

Zoals besproken in 3.1, kan een threshold ook adaptief bepaald worden. Door het nemen van een steekproef kunnen we een schatting maken voor de threshold. Aangezien we a priori weten waar een rol zich in de viewport van de camera zal bevinden (4.1.1), kunnen we in dit gebied een steekproef nemen. Voor deze steekproef berekenen we het rekenkundig gemiddelde van de grijsintensiteiten, en dit gemiddelde zullen we gebruiken als onze threshold (Figuur 4.13).

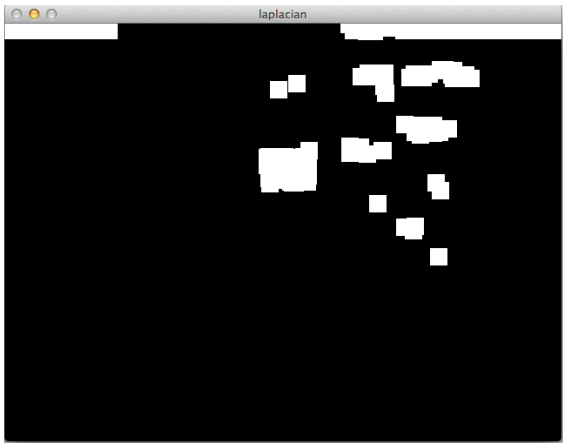
Het rekenkundig gemiddelde geeft een goede schatting over de te verwachten intensiteit. Wanneer een rol onderbelicht is, zal het rekenkundig gemiddelde zeer laag zijn, en worden door de threshold ook de minder sterke veranderingen getoond. (vice versa voor overbelichting).



Figuur 4.13: Streekproef, adaptieve threshold.

4.4.3 Morfologische transformatie

Door de closing-operatie toe te passen zullen de randpixels die dicht bij elkaar liggen een cluster vormen; de lege ruimte tussen de verschillende getallen wordt opgevuld (Figuur 4.14). We zien dat de cluster van het label niet de grootste cluster is, want de cluster van de *caption text* (uur en datum) is groter. Zoeken naar de grootste cluster is dus niet voldoende.



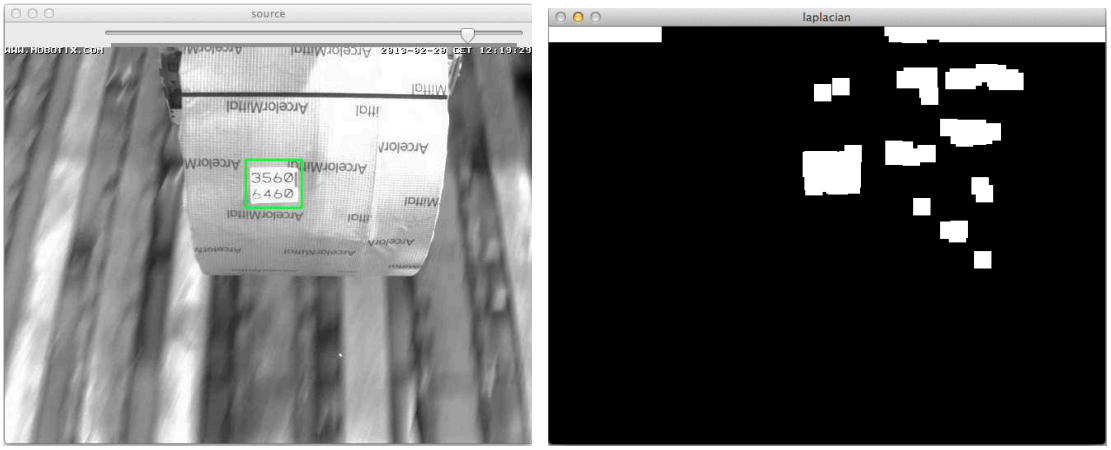
Figuur 4.14: Cluster.

Hiernaast zien we dat de cluster van het rolnummer en de vorm van het label zeer goed overeenkomen. Dit leidt ons terug naar vorige methode, namelijk vind de randen van het label. Dat is nu zeer eenvoudig, want de overgangen zijn zeer steil. Bo-

vendien werkt deze methode ook als het label overbelicht is, maar het rolnummer nog steeds zichtbaar is.

4.4.4 Border following

De methode die ons hierbij kan helpen is het border following-algoritme (3.4.5). Het algoritme zal de verschillende componenten/clusters bepalen. We kunnen vervolgens berekenen welke cluster het best overeenkomt met de afmetingen van het label (dat we verwachten). De positie van deze cluster wordt dan beschouwd als de best mogelijke kandidaat voor het label.



Figuur 4.15: Component van het rolnummer (groen).

Het bepalen van de best mogelijke cluster gebeurt als volgt:

1. Bepaal de uiterste punten van een cluster, en beschouw deze als punten van een omhullende rechthoek.
2. Bereken het verschil (D) tussen de afmetingen (breedte en hoogte) van de omhullende rechthoek en het label. Als het verschil groter is dan een vooropgestelde waarde, stop.
3. Bepaal de vulfactor, $V = \frac{opp.\ cluster}{opp.\ omhullende}$. Als $V < Vmax$, stop.
4. Bereken het verschil tussen de oppervlakte van het label en de oppervlakte van de cluster, en tel het verschil D erbij op. Vergelijk deze waarde met de voorlopig beste waarde. Als de waarde kleiner is, wordt de cluster de voorlopig beste cluster.
5. Stop als alle clusters afgewerkt zijn.

In de tweede stap zorgen we ervoor dat enkel clusters toegelaten worden met een kleine afwijking ten opzichte van de afmetingen van het label, dus niet clusters zoals Figuur 4.16d. Merk op dat hierdoor ook rolnummers die geroteerd zijn, in aanmerking kunnen komen. Stap 3 zorgt ervoor dat enkel clusters die effectief een rechthoek, al dan niet geroteerd, voorstellen worden geaccepteerd. Dit verwerpt valse clusters, zoals Figuur 4.16a en Figuur 4.16c. Vervolgens wordt in stap 4 een afstand bepaalt

voor de cluster; deze waarde bepaald hoe sterk de cluster van het label verschilt. De cluster (Figuur 4.16b) met de laagste afstand wordt gekozen.

Figuur 4.16: Clusters.

4.4.5 Resultaten

We zullen het algoritme evalueren a.d.h.v. een evaluatieset (2.4) en dit voor drie willekeurige kranen: de kraan LK203 uit productiehal Sikel en kranen 481 en 455 uit productiehal Gent. De beelden in de evaluatieset worden geëvalueerd door het clusteringsalgoritme en de resultaten (Tabel 4.2) zullen manueel bekeken worden. Daarnaast zullen we drie verschillende rolverplaatsingen bekijken.

Tabel 4.2: Evaluatie clusteringmethode a.d.h.v. evaluatieset

Klasse	Zichtbare rolnummers	Gedetecteerde rolnummers	Percentage herkend
Gunstig	38	38	100,00%
Rotatie	11	7	63,64%
Overbelicht	26	21	80,77%
Onderbelicht	21	12	57,14%
Gedeeltelijk	5	4	80,00%
Verpakte rollen	14	13	92,86%
	115	95	82,61%

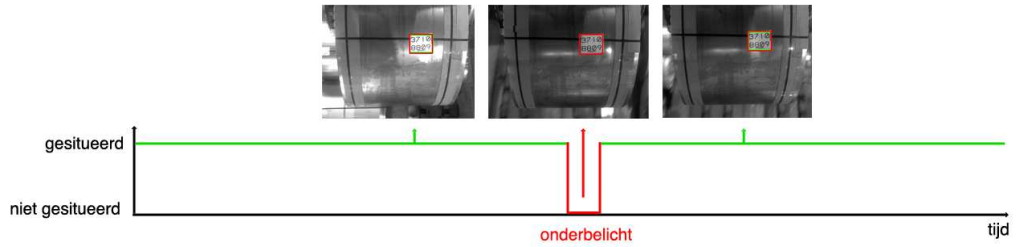
Naast het evalueren met de evaluatieset worden de rolverplaatsingen van drie verschillende rollen (Figuur 4.17) bekeken:

- niet beklede rol zonder overbelichting
- verpakte rol (papier) met over- en onderbelichting
- verpakte rol (karton) met over- en onderbelichting

De groene rechthoeken duiden aan waar het rolnummer gevonden is en de rode waar het rolnummer werkelijk is.

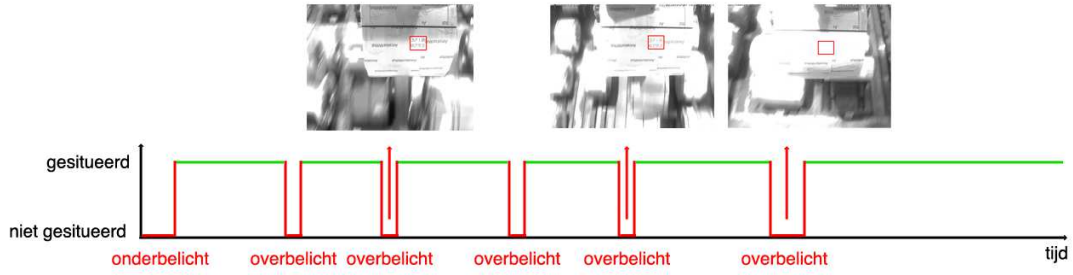
Figuur 4.17: Niet beklede rol, verpakte (papier) rol, verpakte (karton) rol.

Op Figuur 4.18 wordt de rolverplaatsing van een niet beklede rol geïllustreerd. Hieruit kunnen we besluiten dat in een gunstige situatie, zonder overbelichting en verpakking, het rolnummer zo goed als altijd wordt herkend. Enkel door onderbelichting wordt het label tijdelijk niet gesitueerd.



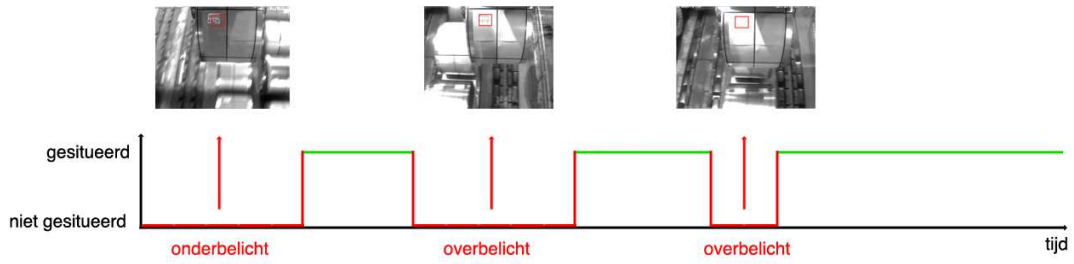
Figuur 4.18: Niet beklede rol zonder overbelichting.

Wanneer een rol overbelicht en verpakt is krijgen we minder goede resultaten (Figuur 4.19). De tekst op de verpakking kan zorgen voor valse herkenningen en de overbelichting kan het rolnummer onleesbaar maken.



Figuur 4.19: Verpakte (papier) rol met over- en onderbelichting.

Een rol die verpakt is met karton en zowel overbelicht als onderbelicht is toont ook minder goeie resultaten. Het valt op dat tijdens het opnemen van de rol, de rol onderbelicht is. Naar het einde toe krijgen we betere beelden waardoor het rolnummer gesitueerd wordt (Figuur 4.20).



Figuur 4.20: Verpakte (karton) rol met over- en onderbelichting.

4.5 Voting-heuristiek

We hebben een methode gevonden die ons toelaat om het rolnummer te situeren. Uit de resultaten kunnen we besluiten dat het algoritme tekortschiet wanneer het rolnummer onder- of overbelicht is. Wanneer het rolnummer onderbelicht is, kan het algoritme een andere "foute" cluster selecteren. Om deze foute detecties te vermijden, zullen we een heuristiek toepassen die rekening houdt met de frequenties. Clusters die meerdere keren geselecteerd zijn hebben een grotere waarschijnlijkheid dan clusters die maar één keer geselecteerd zijn.

Het zou ons kunnen helpen mochten we iets weten over vorige *matches*. Hierdoor kunnen we valse detecties vermijden en het herkenningsproces optimaliseren. We bespreken twee datastructuren die ons hierbij kunnen helpen.

4.5.1 Tabel

Een eenvoudige methode houdt de gevonden posities bij in een simpele tabel. Wanneer het clusteringalgoritme op een frame wordt toegepast, wordt een punt voor het rolnummer bepaald. Van dit punt wordt het dichtstbijzijnde punt (euclidische afstand) bepaald in de tabel. Wanneer de tabel niet leeg is, zal zeker een dichtstbijzijnd punt gevonden worden. Als dat dichtstbijzijnde punt dicht genoeg ligt, binnen een bepaald interval (L), dan beschouwen we het punt als gelijkwaardig en incrementeren we de frequentie van dat dichtstbijzijnde punt. Wanneer het dichtstbijzijnde punt te ver van het gevonden punt ligt, voegen we het punt aan de tabel toe met de frequentie waarde 1. Het interval wordt gebruikt omdat tijdens het verplaatsen, de rol kan slingeren t.o.v. de camera. Hierdoor kunnen de posities voor het rolnummer lichtjes afwijken. Een nadeel van deze eenvoudige datastructuur is dat de tabel telkens helemaal moet doorzocht worden.

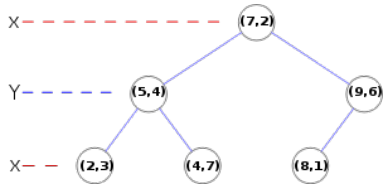
Als we vervolgens, naast de tabel, ook het punt bijhouden met de hoogste frequentie, het punt dat voorlopig het meest verkozen is door het clusteringalgoritme, dan kun-

nen we valse positieven uitschakelen. Wanneer bijvoorbeeld positie *p1*, 50 maal verkozen is en plots positie *p2* voor de eerste keer, dan is het vrij onwaarschijnlijk dat het rolnummer te vinden is op positie *p2*. We zullen vervolgens positie *p2* negeren en doen alsof het rolnummer op positie *p1* is gevonden.

Stel dat na een bepaalde tijd de frequentie van y groter wordt dan de frequentie van *p1*, dan zal positie *p2*, het (nieuwe) punt met de hoogste frequentie worden.

4.5.2 KD-Tree

In principe volstaat een tabel. Als we deze oplossing schaalbaar willen maken, om meer frames per seconde te kunnen verwerken, dan kunnen we beroep doen op een andere structuur, de KD-tree (Figuur 4.21) met K=2; een afbeelding is 2-dimensionaal. Het nadeel van de tabel is dat wanneer een nieuw punt bepaald werd door het clusteringalgoritme, de tabel helemaal moet doorzocht worden om het dichtstbijzijnde punt te vinden. Wanneer de punten worden opgeslagen in een KD-tree, is het mogelijk dat het dichtstbijzijnde punt vroeger wordt gevonden en dat dus niet zoals bij de tabel, alle punten in de datastructuur moeten worden overlopen.



Figuur 4.21: KD-tree.

De werkwijze is volledige dezelfde als bij de tabel; alleen is de manier waarop punten worden toegevoegd en het dichtstbijzijnde punt gevonden wordt, complexer.

4.5.3 Tabel of KD-tree?

Het voordeel van de KD-tree is dat het dichtstbijzijnde punt sneller gevonden kan worden dan in een simpele tabel. Het nadeel is dat de KD-tree een complexere structuur heeft dan een tabel. De complexe structuur van de KD-tree loont pas de moeite als er veel in de KD-tree gezocht zal worden en als er veel punten in de KD-tree zitten; anders volstaat zoeken in een kleine tabel.

Gelukkig kan deze tabel nooit zeer groot worden; de grootte van de tabel is beperkt door het aantal frames. Er kunnen voor een typische rolverplaatsing maar een honderdtal frames worden geëvalueerd. Het overlopen van een tabel (met een honderdtal elementen) is niets in vergelijking met het overlopen van elke pixel van een afbeelding. Daarom zullen we toch gebruikenmaken van een tabel en niet van een KD-tree.

In Figuur 4.22 wordt links met rood de meest voorkomende positie aangeduid, en met groen de nieuwe positie. We zien dat de nieuwe positie wordt genegeerd omdat

het punt van het rode gebied een hogere frequentie heeft dan het punt in het groene gebied.

Figuur 4.22: Voting, meest voorkomende punt.

5 OCR

Optical character recognition (OCR) of optische karakterherkenning is de conversie van geprinte of geschreven tekst naar digitale tekst (e-tekst). Het maakt het voor machines mogelijk om automatisch tekst te herkennen en te verwerken. Er zijn tal van voorbeelden waar OCR zijn praktisch nut bewijst – zoals bij nummerplaatherkenning, sorteren van brieven (bpost), digitaliseren van oude boeken, enz.

Tekstherkenning kan net zoals tekstdetectie zeer complex zijn. De te herkennen tekst kan immers in verschillende kleuren of schuin staan. Ook het onderscheid tussen karakters is niet altijd even duidelijk; denk maar aan de letter o en het cijfer 0. Ook hier zijn er verschillende methodes voorgesteld om herkenning mogelijk te maken.

Dezer dagen bestaan er dan ook zeer veel OCR-tools (met een nauwkeurigheid tussen de 71% en 98% [48]), die onderverdeeld kunnen worden in verschillende categorieën: Desktop OCR, Server OCR, Web OCR, etc. Jammer genoeg zijn veel van deze tools betalend en niet open source. Eén van de weinige tools die toch gratis en open source is, en toch een zeer hoge nauwkeurigheid en performantie heeft, is de OCR-engine *Tesseract*. Het is een engine en geen volledig softwarepakket, wat wil zeggen dat het mogelijk is om Tesseract te gebruiken in andere applicaties. Op basis van deze positieve eigenschappen komt Tesseract in aanmerking voor de rolnummerherkenning. We zullen daarom kort Tesseract toelichten en proberen om de rolnummers met deze OCR engine te herkennen. Vervolgens zullen we de resultaten van Tesseract uitvoerig bespreken.

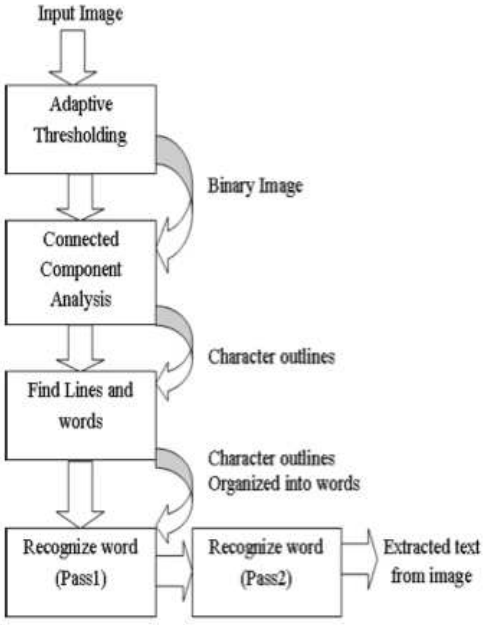
5.1 Tesseract

Tesseract is een open source OCR-engine en is begonnen als een doctoraatsonderzoek, met de bedoeling om een extra feature aan te bieden in scanners (deze integratie is echter nooit verwezenlijkt). Deze is oorspronkelijk ontwikkeld bij Hewlett-Packard

(HP) in 1984, door Ray Smith, en sinds 2005 vrijgegeven als open source. Momenteel wordt de engine onderhouden door Google.

5.1.1 Structuur

De werking van Tesseract wordt voorgesteld door een flowdiagram (Figuur 5.1). In de eerste fase wordt een *adaptive threshold* (3.1) toegepast op een grijswaarden- of kleurenafbeelding; onderzoek bevestigt een positieve wending bij het gebruiken van een grijswaardenafbeelding [48]. Hierna volgt de *connected component analysis*, waarbij de verschillende componenten of karakters bepaald worden. In de volgende fase worden de componenten omgevormd tot *blobs/clusters* die op hun beurt omgevormd worden tot tekstlijnen. Hierbij worden voor elke tekstlijn enkele zaken geanalyseerd: gelijkaardige tekstgrootte en vaste ruimte tussen de karakters. Aan de hand van deze eigenschappen worden de tekstlijnen verdeeld in woorden. In de laatste fase worden de woorden effectief herkend, waarbij twee ronden worden ingelast. In de eerste ronde worden de woorden voor het eerst herkend en worden de positieve herkenningen doorgegeven aan een *adaptive classifier* als training data. In de tweede stap wordt de adaptive classifier gebruikt om de woorden voor een tweede keer te herkennen. Deze laatste stap kan handig zijn, omdat bv. pas op het einde van de tekst (tijdens de eerste ronde) belangrijke informatie kan verkregen zijn die eigenlijk ook relevant was voor het begin van de tekst. Door tweemaal de tekst te overlopen weten we iets meer over de tekst en zullen we hem ook beter begrijpen. [49]



Figuur 5.1: Blokdiagram Tesseract. (bron: [48])

Tesseract ondersteunt ondertussen verschillende talen. Standaard wordt de taal Engels aangeboden in de vorm van een *trainingdata* bestand. Dit bestand bevat de karakterset, van verschillende lettertypes, die gebruikt wordt voor een specifieke taal. Daarnaast kan (optioneel) een woordenlijst worden aangeboden van woorden die frequent

gebruikt worden, om de herkenning te optimaliseren. Een bijkomend voordeel van Tesseract is dat er een mogelijkheid is om zelf een *trainingdata* bestand op te stellen; dit wordt besproken in de volgende paragraaf.

Wanneer een afbeelding wordt geëvalueerd geeft de engine een .txt-bestand terug. In dit bestand staan de herkende karakters. Er is ook een mogelijkheid om een *whitelist* op te stellen, deze lijst bevat dan de karakters die enkel mogen herkend worden tijdens het evalueren. Dit kan bijvoorbeeld gebruikt worden als we iets weten over de uitvoer, zoals enkel getallen bij het evalueren van een rolnummer.

5.1.2 Voorbeeld

Tesseract kan gebruikt worden om een gescand boek om te zetten naar een digitale versie, maar bijvoorbeeld ook om een handgeschreven lettertype te herkennen. Om de werking van Tesseract aan te tonen zullen we proberen om het handgeschreven lettertype in Figuur 5.2 te herkennen.

Figuur 5.2: Trainingset voor handgeschreven lettertype.

De herkenning kunnen we uitvoeren via command-line of een grafische interface die gebruikmaakt van de Tesseract-engine. Figuur 5.3 geeft de command-line opdracht weer om een afbeelding d.m.v. Tesseract te herkennen, hiervoor wordt gebruikgemaakt van de standaard *trainingdata*. Als eerste parameter wordt de afbeelding doorgegeven en als tweede de bestandsnaam naar waar het resultaat van de herkenning wordt uitgeschreven. De uitvoer wordt weergegeven in Figuur 5.3.

```
$ tesseract Handwriting-font.jpg output.txt

Tesseract Open Source OCR Engine v3.01 with Leptonica
```

Figuur 5.3: Herkennen afbeelding.

```
Ta
HAN
WQI%c?
LETTE@
wf&I”;n,$PROECT
```

Figuur 5.4: Resultaat herkenning.

Sommige karakters konden herkend worden zoals het eerste deel van “LETTER”, en andere konden zelfs niet gesitueerd worden. Het slechte resultaat is te wijten aan de trainingdata; deze ondersteunt (in de standaard trainingset) geen handgeschreven lettertypes. Zelfs al zou een handgeschreven lettertype ondersteund worden, dan nog kunnen de resultaten even slecht zijn. Daarom kan het in sommige situaties interessant zijn om Tesseract te trainen zodat hij bijvoorbeeld het handgeschreven lettertype uit Figuur 5.2 wel correct kan herkennen.

5.1.3 Training

Een bijkomend voordeel is dat Tesseract getraind kan worden, wat handig is om andere talen, naast het Engels, te herkennen. Germaanse talen gebruiken een gelijkaardige karakterset en kunnen dus gebruik maken van de standaard *trainingdata*. Daarentegen hebben andere talen, zoals Chinees of Japans, een compleet andere karakterset, waardoor een ander trainingdata-bestand nodig is. Het aanmaken van zo’n trainingdata bestand, voor een specifieke taal/lettertype, gebeurt door enkele opeenvolgende stappen te volgen.

Onderzoekers hebben geprobeerd om m.b.v. Tesseract Oudgrieks te herkennen. Hiervoor heeft men een trainingdata bestand moeten opstellen. De verschillende stappen en resultaten van het trainingsproces heeft men in een publicatie opgenomen. Het trainingsproces beschrijft men als complex omdat de kwaliteit van de beschikbare documentatie te wensen overlaat. [50] De documentatie is hoofdzakelijk te vinden op de officiële website van Tesseract. [51]

Voor onze toepassing zouden we ook de standaard trainingsdata kunnen gebruiken omdat er enkel getallen moeten worden herkend; (Arabische) cijfers zijn universeler dan letters. De standaard trainingsdata bestaan echter uit meerdere verschillende lettertypes, die verschillend zijn van het lettertype dat gebruikt wordt om de rolnummers op de labels te drukken. Het kan zijn dat hierdoor een slechte herkenning wordt gerealiseerd omdat de getallen in de trainingdata te sterk afwijken van de getallen op de rolnummers. Daarom kan het voordelig zijn om een eigen trainingsdata bestand op te stellen.

De verschillende stappen van het trainingsproces bestaan uit [51]:

1. Aanmaken *boxfile*.
2. Manueel verbeteren van *boxfile*.
3. Ingeven correcte *boxfile*.
4. Detecteren van de *characterset*.

5. Toelichten van het lettertype en eigenschappen.
6. Aanmaken *clusteringdata.*
7. Hernoemen en combineren van alle bestanden.
8. Verplaatsen van *trainingdata* bestand.

We bespreken kort, zonder in detail te treden, de belangrijkste stappen van het trainingsproces.

5.1.3.1 Boxfiles

In de eerste stap van het trainingsproces wordt een afbeelding (*trainingset*) ontworpen, die zal gebruikt worden om Tesseract het nieuwe lettertype aan te leren. Na het ontwerp zal Tesseract proberen om de verschillende componenten in de trainingset te situeren en te herkennen. Het resultaat van de herkenning en bijhorende gegevens (x- en y-coördinaten, breedte en hoogte) van de verschillende componenten worden in een boxfile opgeslagen.

Figuur 5.2 kan bijvoorbeeld gebruikt worden als trainingset voor het handgeschreven lettertype. Wanneer we de boxfile aan de hand van Tesseract OCR chopper [52] aanmaken, krijgen we volgend resultaat.


```
T 99 265 142 335 0
a 122 265 178 330 0
H 29 190 66 275 0
A 74 187 102 266 0
N 119 192 165 264 0
W 14 117 55 182 0
Q 58 118 95 186 0
I 117 117 124 184 0
% 138 125 222 263 0
c 224 127 247 188 0
N 252 124 286 182 0
L 58 66 78 108 0
E 92 66 116 107 0
T 128 66 157 107 0
T 161 68 186 112 0
E 189 68 211 108 0
@ 222 68 244 112 0
w 12 6 59 42 0
f 59 12 67 46 0
& 69 14 80 48 0
i 80 12 90 48 0
" 89 32 97 50 0
; 98 13 108 48 0
n 107 16 118 50 0
' 120 16 125 26 0
§ 126 19 130 26 0
P 146 5 164 60 0
R 173 6 198 58 0
O 204 7 232 56 0
E 278 4 294 58 0
C 299 1 330 59 0
T 339 5 362 60 0
```

Figuur 5.5: Boxfile voor handgeschreven lettertype.

We zien hetzelfde resultaat als in Figuur 5.4, alleen hebben we nu een grafische interface. Het is dan ook de bedoeling dat de boxfile wordt bijgewerkt en terug bij Tesseract wordt ingediend, om Tesseract te trainen; foute karakters en karakters die niet werden gesitueerd worden d.m.v. de grafische tool aangepast met de correcte waarde. Wanneer dit is gebeurd, zal een trainingsbestand worden aangemaakt dat informatie bevat over elk karakter (features); voor meer informatie welke features en hoe deze features worden bepaald, wordt er naar de documentatie van Tesseract verwezen. [48]

5.1.3.2 Font properties

Na het ingeven van de boxfile wordt extra informatie vergaard over het te herkennen lettertype. Zo worden de karakters die het lettertype bevat verzameld en moet een extra bestand worden aangemaakt met de naam van het lettertype en speciale eigenschappen (bold, italic, ...).

5.1.3.3 Feature extraction

Vervolgens worden de features uit het trainingsbestand gefilterd en over verschillende bestanden (*inttemp*, *shapetable*, *pffmtable* en *normproto*) verdeeld. De verschillende bestanden worden aangepast met een prefix die de taal van het trainingsbestand aangeeft. Vervolgens worden de bestanden gecombineerd tot een trainingdata bestand en wordt deze in de directory van Tesseract geplaatst.

In de volgende paragraaf zullen we Tesseract gebruiken om het rolnummer te herkennen. Hierbij zullen we een trainingset opstellen om Tesseract te trainen. Vervolgens worden enkele technieken besproken die de herkenning kunnen optimaliseren en worden de resultaten van de herkenningen, al dan niet getraind, vergeleken.

5.2 Toepassing

In deze paragraaf zullen we Tesseract gebruiken om het rolnummer te herkennen. We zullen in eerste instantie de rolnummers proberen te herkennen zoals deze worden afgeleverd door het tekstdetectie algoritme (4.4). Daarna zullen enkele bewerkingen worden voorgesteld die de herkenning positief kunnen beïnvloeden en wordt een trainingset opgesteld. Vervolgens zullen we de resultaten zonder en met nieuwe trainingset vergelijken en besluiten of het trainen van het lettertype al dan niet gunstig is.

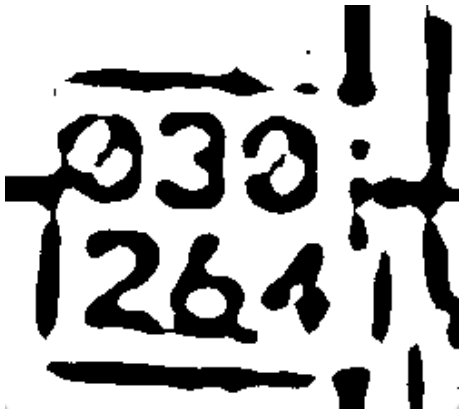
5.2.1 Standaard

Onderstaand rolnummer (Figuur 5.6) werd afgeleverd door het tekstdetectiealgoritme. Als we Tesseract uitvoeren op het rolnummer krijgen we een blanco resultaat; geen enkel component kon gesitueerd worden. Merk op dat dit door de adaptieve thresholding van Tesseract komt en niet door het trainingdata-bestand.

Figuur 5.6: Typisch rolnummer.

5.2.2 Adaptive thresholding

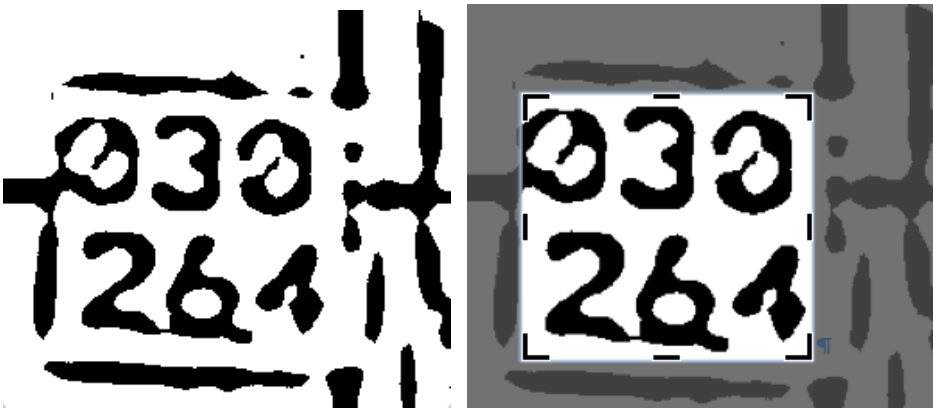
In Figuur 5.6 kon geen enkel component herkend worden. Wanneer we zelf een adaptieve threshold toepassen m.b.v. Emgu CV kunnen we het resultaat positief beïnvloeden. Wanneer de afbeelding, na een adaptieve threshold, wordt geëvalueerd, kan Tesseract wel de verschillende componenten evalueren. Het resultaat “1—5” van de herkenning is echter niet goed omdat er veel valse componenten gedetecteerd werden.



Figuur 5.7: Resultaten Tesseract met standaard trainingdata.

5.2.3 Auto croppen

Het zou goed zijn mochten we de valse componenten kunnen uitsluiten, zodat enkel het rolnummer nog zichtbaar is. Dit kan verwezenlijkt worden door *croppen* (bijsnijden). Croppen is het verwijderen van de uiterste stukken van een afbeelding. Wanneer we het rolnummer handmatig zouden croppen, krijgen we volgend resultaat (Figuur 5.8).



Figuur 5.8: Voor en na croppen.

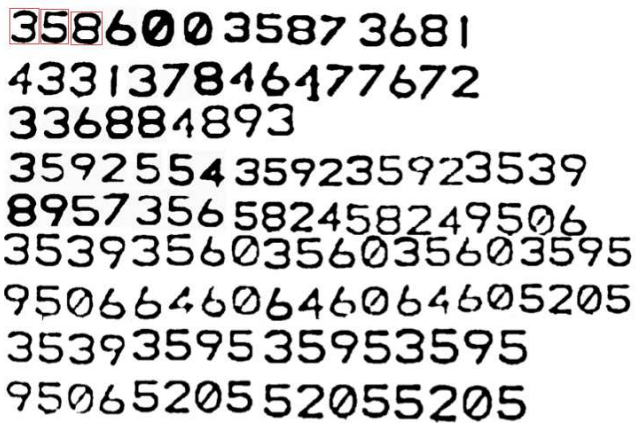
Het probleem hierbij is dat het croppen automatisch moet gebeuren. Wanneer we handmatig bovenstaande afbeelding croppen, gaan we eerst op zoek naar het meest linkse cijfer op de eerste regel (0), hierna naar het meest rechtse cijfer op de tweede

regel (4). Vervolgens bepalen we een bereik van (0) tot (4), met als gevolg dat het rolnummer in dit bereik valt.

We kunnen m.b.v. de technieken, besproken in Hoofdstuk 3, het automatisch croppen realiseren. Na de adaptieve threshold, kunnen de verschillende componenten bepaald worden d.m.v. het border following-algoritme. Aangezien de cijfers van het rolnummer steeds even groot zijn en we a priori de grootte van een cijfer weten, kunnen we de verschillende componenten filteren op basis van hun grootte. Vervolgens bepalen we een minimaal bereik dat alle componenten omvat. Als we opnieuw Tesseract toepassen krijgen we een beter resultaat “339 2614”.

5.2.4 Training

Om de herkenning te optimaliseren, kunnen we Tesseract trainen voor het lettertype van het rolnummer. Hierdoor is het mogelijk, dat betere resultaten worden verkregen. In Figuur 5.9 wordt een trainingset opgesteld, bestaande uit delen van verschillende rolnummers. Er werd getracht om een voldoende grote verzameling representatieve labels te beschouwen, waarin alle cijfers voldoende voorkomen. Vervolgens wordt de boxfile berekend, en worden de foute herkenningen aangepast (Figuur 5.10). Zo werd bijvoorbeeld het vijfde karakter “0” fout herkend.



Figuur 5.9: Trainingset rolnummer.

```
3 65 2829 170 2958 0
5 184 2827 288 2956 0
8 307 2822 420 2947 0
6 439 2817 560 2960 0
9 581 2813 709 2961 0
0 748 2818 860 2951 0
3 906 2829 1008 2950 0
5 1033 2829 1135 2946
...
```

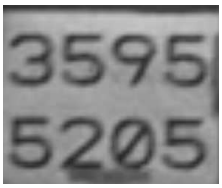
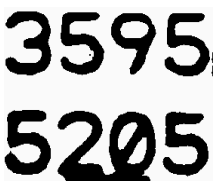
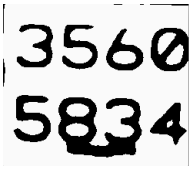
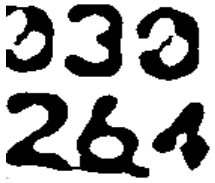
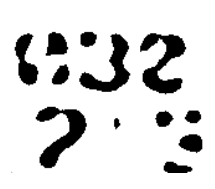
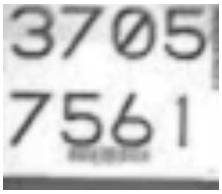
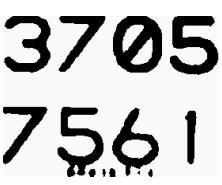
Figuur 5.10: Boxfile rolnummer.

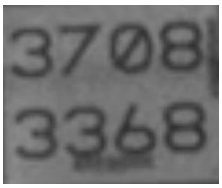
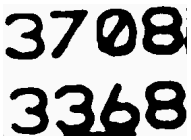
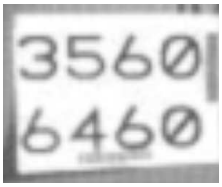
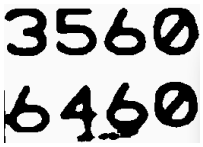
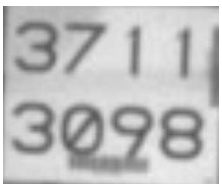
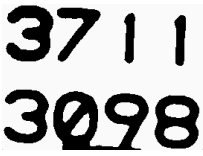
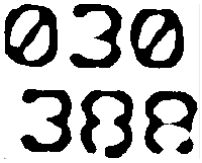
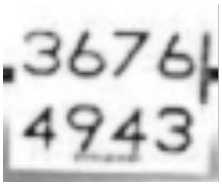
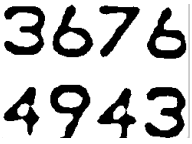
Wanneer de boxfile is aangepast, worden de volgende stappen van het trainingsproces uitgevoerd, zoals vermeld in 5.1.3. Uiteindelijk wordt een trainingdata bestand verkregen en wordt het bestand in de directory van Tesseract geplaatst. We bespreken de resultaten in de volgende paragraaf.

5.2.5 Resultaten

In wat volgt worden enkele rolnummers herkend die typisch zijn voor een rolverplaatsing. De herkenning wordt uitgevoerd op de standaardbeelden met de standaard trainingdata en op de bijgewerkte beelden met de nieuwe trainingdata.

Tabel 5.1: Resultaten rolnummerherkenning

Rolnummer	Bewerkt	Rolnummer (standaard trainingdata)	Bewerkt (nieuwe trainingdata)
		3593 552951	3595 5205
		3560 5834	53560 5834
			030 2614
			132 713
		3703 7591	3705 75061

		13709 43213	3708 3368
		35501 499	3560 6460
		524	3711 3038
		030 32311 1	030 388
		3676 4853	3676 4943

Uit de resultaten kunnen we afleiden dat, rolnummers die herkend werden zonder training, betere resultaten kunnen geven dan met training. Ook na training kunnen sommige rolnummers nog steeds niet correct herkend worden. Ondanks deze opmerkingen kan duidelijk vastgesteld worden dat het voordelig is om Tesseract te trainen.

Vaak worden bijkomende karakters gedetecteerd door Tesseract, door de adaptieve threshold (5.2.2), daarom zullen we in volgende paragraaf een oplossing voor dit probleem voorstellen.

5.3 LGD

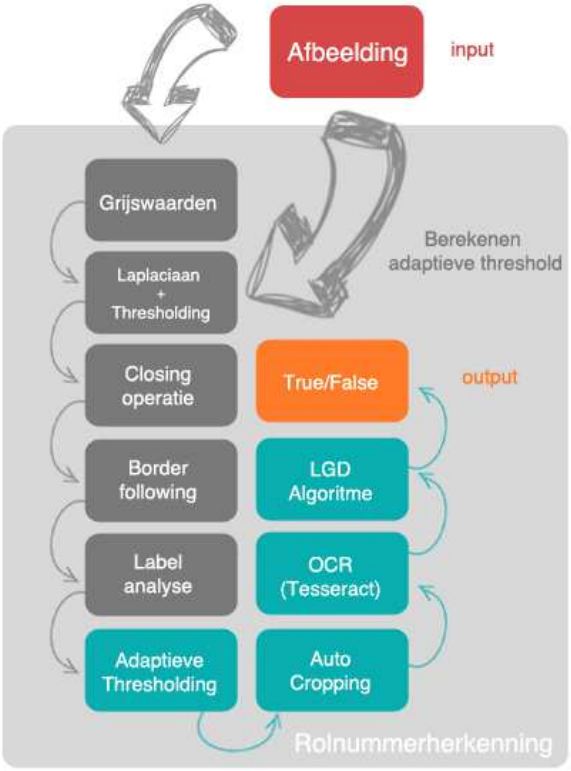
Het resultaat aangeleverd door de Tesseract engine moet vergeleken worden met het te verwachten rolnummer. Wanneer beide waarden gelijk zijn weten we dat de rol die opgenomen is, de correcte rol is. Het tegenovergestelde is niet waar, want door over- en onderbelichting kan het resultaat van Tesseract negatief beïnvloed worden. Hier-

door is het mogelijk dat het rolnummer gedeeltelijk of zelf niet herkend wordt. Daarnaast kunnen er ook valse karakters gedetecteerd worden waardoor er, naast het correcte resultaat, nog meerdere karakters zijn herkend. Het volstaat dus niet om gewoon de twee strings te vergelijken. Zoeken naar de overeenkomstige deelstring in beide strings geeft ons wel een goede oplossing. Hiervoor kan het algoritme van de langst gemeenschappelijke deelstring (LGD) [47] gebruikt worden.

Het algoritme geeft ons de lengte van de langst gemeenschappelijke deelstring in de twee strings. Wanneer deze lengte gelijk is aan de lengte van het te verwachten rolnummer, dan weten we dat beide waarden overeenkomen. Tijdens een rolverplaatsing zullen opeenvolgende camerabeelden geëvalueerd worden. Daarbij is het mogelijk dat als er in het begin van een rolverplaatsing genoeg zekerheid is, de te verplaatsen rol de correcte rol is en de volgende beelden niet meer moeten geëvalueerd worden. Hierdoor kan de evaluatie vroegtijdig worden afgesloten.

5.4 Rolnummerherkenning

Nu we alle componenten hebben, kunnen we ze samenbrengen tot een werkende oplossing. Figuur 5.11 geeft het schema van de rolnummerherkenning weer. We overlopen het kort. Een afbeelding wordt aan het algoritme aangeboden, en wordt omgezet naar grijswaarden. Vervolgens worden de sterke veranderingen bepaald door de Laplaciaan en wordt een threshold berekend op basis van de afbeelding. Deze adaptieve threshold wordt gebruikt om te bepalen hoe sterk de veranderingen moeten zijn om in aanmerking te komen voor het rolnummer. De closing-operatie zorgt ervoor dat de punten van het rolnummer een cluster vormen. Het border following-algoritme bepaalt alle connected components (clusters). De cluster die het best overeenkomt met het formaat van het rolnummer wordt gekozen, en wordt in de volgende stappen verder geanalyseerd. In het gebied van de cluster wordt een adaptive thresholding uitgevoerd en wordt het cropping-mechanisme toegepast om valse componenten te verwijderen. Tesseract wordt toegepast en het resultaat wordt d.m.v. het LGD-algoritme vergeleken. De functie geeft waar indien beide waarden overeenkomen.

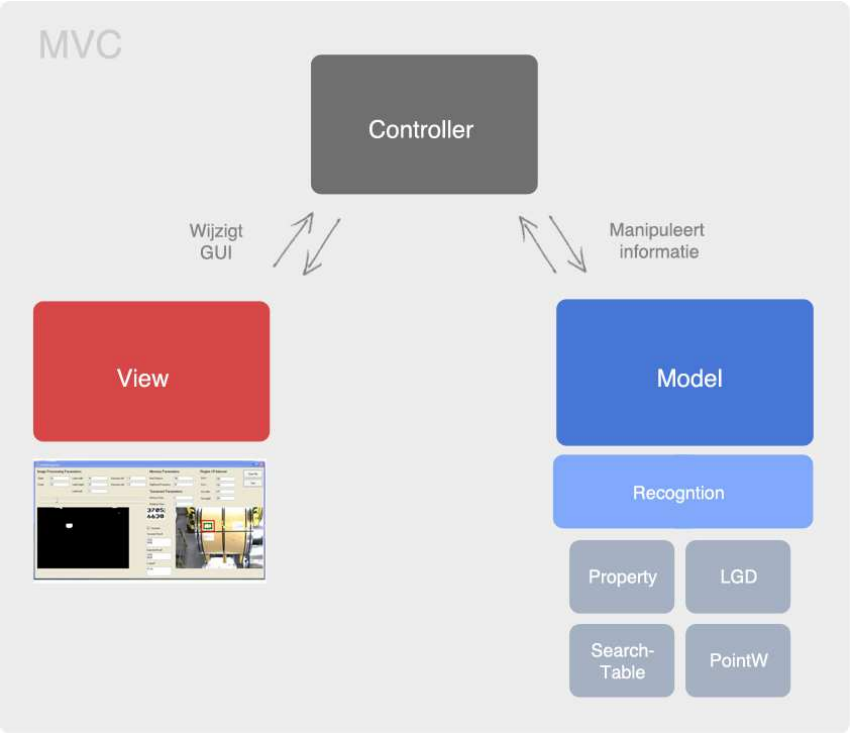


Figuur 5.11: Flowdiagram rolnummerherkenning.

In het volgende hoofdstuk wordt kort een applicatie besproken die voorgaande functionaliteiten zal bevatten. De applicatie zal gebruikt worden om camerabeelden te (her)evalueren en bepaalde fundamentele beeldverwerkingsparameters bij te werken.

6 Herkenningsapplicatie

In dit hoofdstuk wordt kort de herkenningsapplicatie besproken, die het mogelijk maakt om beelden te (her)evalueren. De applicatie, geschreven in C#, maakt gebruik van de algoritmen en heuristieken besproken in de vorige hoofdstukken. Er wordt hierbij gebruikgemaakt van het Model-View-Controller pattern (Figuur 7.1). Vervolgens wordt deze applicatie geïntegreerd in een service, die realtime beelden verwerkt.



Figuur 6.1: Flowdiagram applicatie.

6.1 Configuratie

Sommige van de algoritmen en heuristieken hebben parameters nodig. Deze parameters kunnen met behulp van deze applicatie experimenteel bepaald worden. Zo is men verplicht om enkele parameters in te geven via een XML-bestand (Figuur 6.2).

Afhankelijk van een productiehal of kraan kunnen de parameters veranderen. Zo is de grootte van het rolnummer in de productiehallen Gent en Sikel verschillend. Daarnaast bestaat het rolnummer in de productiehal uit acht cijfers en in Sikel uit zes cijfers. De *region of interest* kan ook verschillend verschillend zijn per productiehal.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Settings productionhall Gent -->
<parameters>
  <!-- Region of interest-->
  <roi_x>290</roi_x>
  <roi_y>160</roi_y>
  <roi_width>470</roi_width>
  ...
  <!-- Tesseract -->
  <min_char>6</min_char>
  <max_char>10</max_char>
</parameters>
```

Figuur 6.2: Configuratiebestand voor de productiehal Gent.

De parameters en hun default-waarden voor de productiehallen Gent en Sikel worden verder besproken in Tabel 6.1.

Tabel 6.1: Parameters rolnummerherkenning

Naam	Beschrijving	Gent	Sikel
roi_x	x-coördinaat van region of interest	290	290
roi_y	y-coördinaat van region of interest	0	160
roi_width	breedte van region of interest (pixels)	470	430
roi_height	hoogte van region of interest (pixels)	300	300
dilate	grootte structuurelement dilateren (pixels)	20	10
erode	grootte structuurelement eroderen (pixels)	31	20
guassian_smt	standaardafwijking voor Gaussian filter in horizontale dimensie.	1.5	9.5

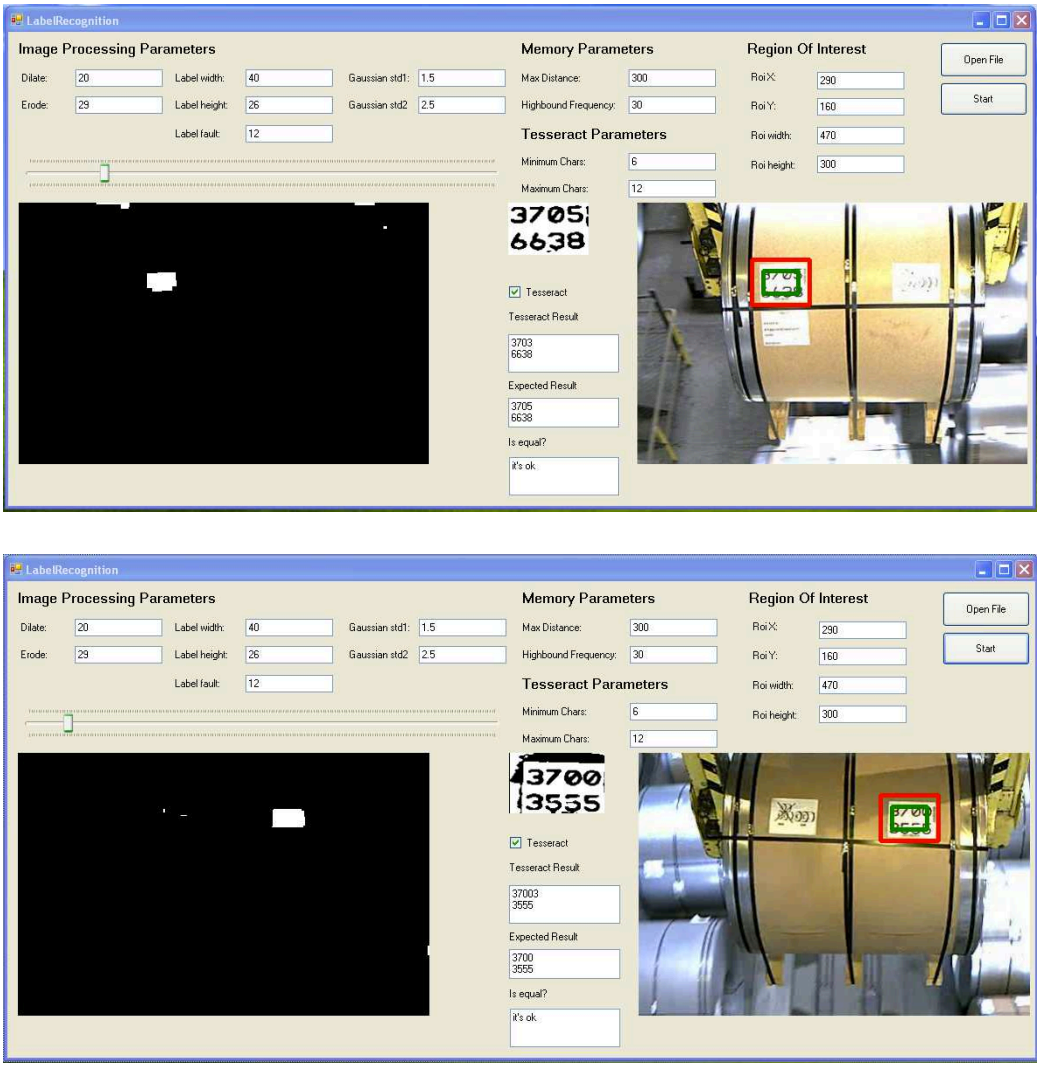
guassian_smt2	standaardafwijking voor Gaussian filter in verticale dimensie.	2.5	11.5
label_width	breedte van het te verwachten label (pixels)	40	20
label_height	hoogte van het te verwachten label (pixels)	30	15
label_fault	maximale afwijking van een cluster t.o.v. het label (pixels)	20	8
upper_limit_frequency	maximale frequentie dat een punt in de tabel kan toegekend krijgen	30	20
max_distance	maximale afwijking t.o.v. het dichtstbijzijnde punt (pixels)	30	30
min_char	minimaal aantal karakters dat door Tesseract moet worden herkend	6	4
max_char	maximaal aantal karakters dat door Tesseract moet worden herkend	12	7

6.2 GUI

In Figuur 6.3 wordt de GUI van de evaluatieapplicatie weergegeven. De gegevens van het XML-bestand worden geparsed en ingevuld in de correcte datavelden. Vervolgens kan een videobestand of afbeelding worden ingeladen en wordt het herkennings-algoritme uitgevoerd. Men kan steeds de verschillende parameters aanpassen om te experimenteren. De source code is beschikbaar op GitHub. [53]

Het rolnummer wordt geëvalueerd en het resultaat wordt in het tekstveld “Tesseract Result” geplaatst. Er is een mogelijkheid om het resultaat te verifiëren door het te verwachten rolnummer in te vullen in de tekstbox “Expected Result”. Vervolgens zullen de twee strings aan de hand van het algoritme van de langst gemeenschappelijke deelstring vergeleken worden.

Wanneer beide waarden “Expected en Tesseract Result”, bijvoorbeeld 3 keer, gelijk waren, wordt een melding “It’s ok” getoond. Deze functionaliteit is nodig omdat we niet kunnen afgaan op één beslissing. Dit zorgt ervoor dat valse herkenningen “die toevallig gelijk zijn aan de te verwachten waarde” genegeerd worden. Deze parameter zou ook opgenomen kunnen worden in het configuratiebestand, zodat per productie-hal of kraan een threshold kan worden ingesteld.

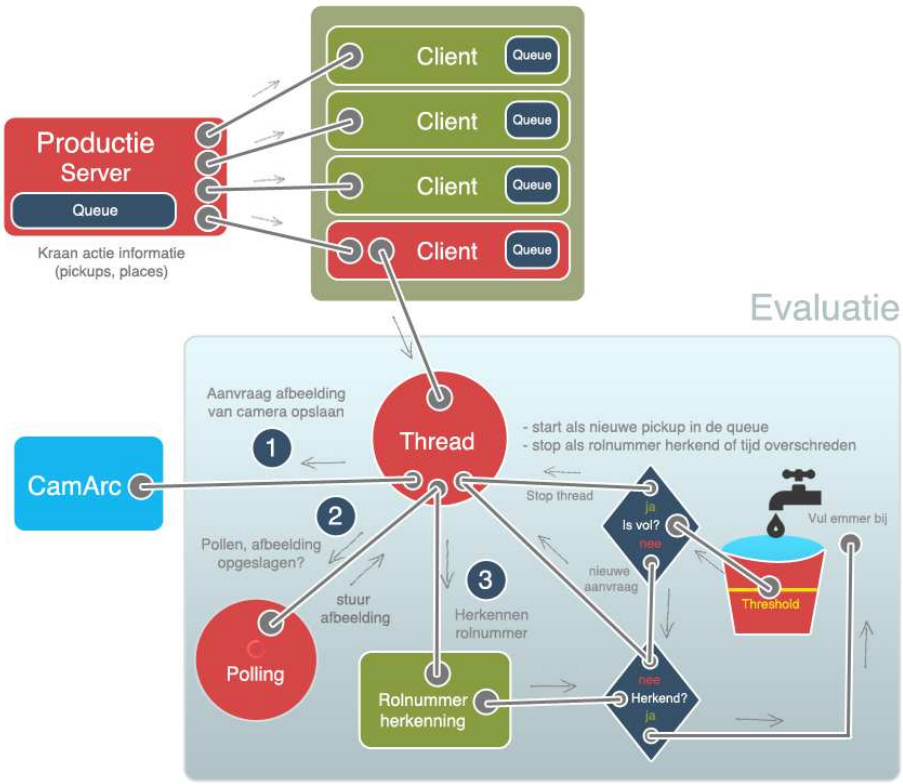


Figuur 6.3: Herkenning rolnummer.

De herkenningsapplicatie kan gebruikt worden bij het opstarten van een nieuwe productiehal of geautomatiseerde kraan, om de juiste parameters voor de rolnummerherkenning te bepalen. In het volgende hoofdstuk wordt de rolnummerherkenning geïntegreerd in een service die realtime beelden, uit een specifieke productiehal, zal afhandelen. De herkenningsapplicatie wordt hierbij gebruikt om de parameters voor deze productiehal te bepalen.

7 Integratie

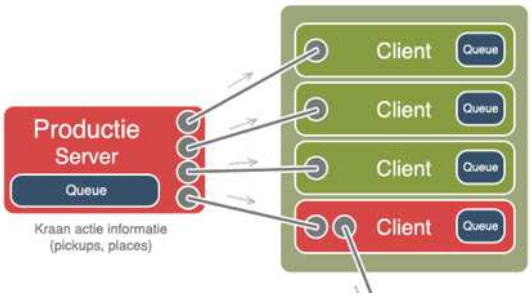
De verschillende camera's in een productiehal zijn aangesloten op een steeltracker waarvan de beelden beheerd worden door CamArc. In dit hoofdstuk wordt een service (Figuur 7.1) opgesteld die wanneer nodig beelden van een specifieke camera bij CamArc opvraagt. Vervolgens worden deze beelden geëvalueerd met de rolnummerherkenning en wordt het rolnummer geclassificeerd als al dan niet herkent.



Figuur 7.1: Flowdiagram herkenning.

Zoals in het eerste hoofdstuk vermeld houdt Genesis alle acties van de verschillende kranen bij. Zo weet Genesis wanneer een kraan roteert, een rol opneemt of neerplaatst. Deze acties worden in een queue bijgehouden door de productieserver. Wanneer een kraan bijvoorbeeld een rol opneemt, wordt deze actie in de queue van de productieserver geplaatst.

Verschillende toepassingen/services (clients) kunnen zich bij de productieserver aanmelden, zodat zij op de hoogte worden gebracht van de verschillende acties. Als een client is ingeschreven bij de productieserver, luistert hij naar de acties die door de productieserver worden uitgestuurd; het observer pattern. Als een client een nieuwe actie ontvangt, wordt de actie in zijn persoonlijke queue geplaatst en kan hij, naargelang zijn functie, de verschillende acties verwerken (Figuur 7.2).

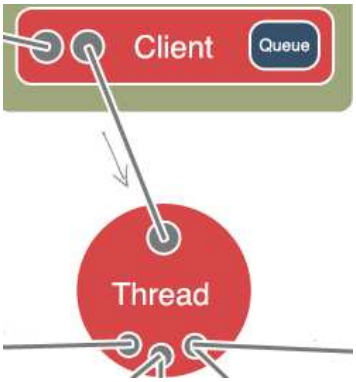


Figuur 7.2: Inschrijven bij de productieserver.

Wanneer de service een nieuwe actie ontvangt, wordt bekeken welke actie er precies is uitgevoerd: een *pickup* of een *place* (Figuur 7.3).

```
List<PickupInformation> CranePickups = new List<PickupInformation>();
private bool HandleActionMoveMaterial(SivaxMessageInfo msgInfo, ActionMoveMaterial action){
   lock (lockObject){
      if (action.To.SlotLevel == 99){ //pickup
          ..
      }
      else if (action.From.SlotLevel == 99){ //place
          ..
      }
   }
   return true;
}
```

Figuur 7.3: Afhandelen van een actie.



Figuur 7.4: Opstarten nieuwe rolnummerherkenning.

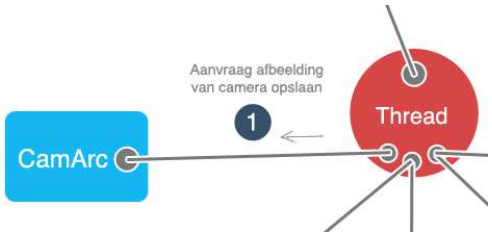
Wanneer een rol wordt opgenomen (pickup), wordt een nieuwe rolnummerherkenning opgestart (Figuur 7.4). Hiervoor wordt een thread aangemaakt met volgende informatie:

- een Beeldverwerkingsobject (Figuur 7.5)
  - een PickupInformation object
    - de naam van de kraan.
    - het nummer van de rol (Genesis).
    - het huidige tijdstip van de pickup.
  - een Recognition object
    - een LGD-object, om de resultaten te vergelijken.
    - een tabel die de gevonden coördinaten en frequenties bewaard.
    - de beeldverwerkingsfuncties.
  - een teller en logische waarde.

```
class ImageProcessingInformation {
   public bool EvaluationDone = false;
   public int EvaluationCount = 0;
   public string expected;
   public Recognition recognition;
   public PickupInformation pInfo;

   public ImageProcessingInformation(PickupInformation pInfo){
      this.pInfo = pInfo;

      // Parsing  rolnumber, remove prefix and suffix
      this.expected = ...;

      recognition = new Recognition(@"C:\...\config\parameters"+pInfo.Crane+".xml");
   }

   public bool recognize(string imagePath){
      return recognition.executeRecognition(imagePath, this.expected);
   }
}
```

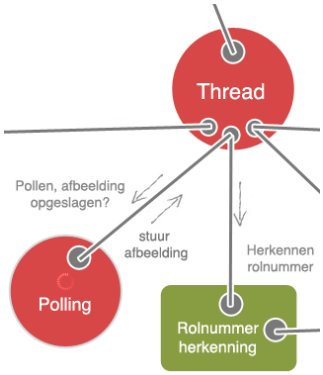
Figuur 7.5: Beeldverwerkingsobject.

Afhankelijk van de productiehal (of kraan) kan een ander configuratiebestand worden ingeladen (Hoofdstuk 6). Er wordt 15 seconden gewacht voor de thread wordt opgestart. Dit is namelijk de tijd die een kraan nodig heeft om een rol op te tillen; het rolnummer is niet zichtbaar tijdens het opnemen.



Figuur 7.6: Aanvraag camerabeeld bij CamArc.

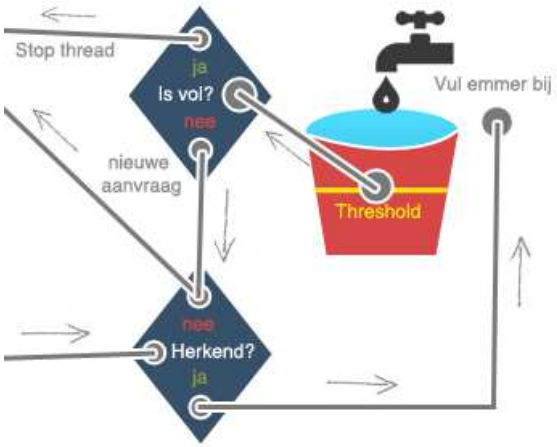
Vervolgens wordt elke seconde een aanvraag naar CamArc gestuurd om een afbeelding van de rolverplaatsing op te slaan. Deze aanvragen stoppen als het rolnummer herkent is of als de rol neergeplaatst is (Figuur 7.7).



Figuur 7.7: Pollen afbeelding beschikbaar.

Wanneer een aanvraag is ingediend, wordt een nieuwe thread opgestart die elke vijf sec. zal kijken of de afbeelding beschikbaar is. Als de afbeelding aanwezig is wordt deze naar de lokale schijf gekopieerd, verwerkt door het herkenningsalgoritme en opnieuw verwijdert (Figuur 7.7).

Vervolgens wordt de afbeelding verwerkt door de rolnummerherkenning. Als het resultaat van Tesseract bijvoorbeeld drie keer overeenkomt met het te verwachten rolnummer besluiten we dat het de juiste rol is. Wanneer op het einde van een rolverplaatsing, het rolnummer geen drie keer correct werd herkend, wordt de rol als niet herkend geclassificeerd (Figuur 7.8).




```
void ProcessImage(string imageId, string imagePath, ImageProcessingInformation imageInfo){
     if (!imageInfo.EvaluationDone){
          if (imageInfo.recognize(imageId, imagePath))
                  imageInfo.EvaluationCount++;

          // If the number is recognized three times, confirm recognition.
          if (imageInfo.EvaluationCount == 3)
                 imageInfo.EvaluationDone = true;
     }
}
```

Figuur 7.8: Herkennen van afbeelding.

Zoals hiervoor vermeld bevat het Recognition object de verschillende beeldverwerkingsfuncties. De functie *executeRecognition* zal eerst het rolnummer situeren en vervolgens Tesseract gebruiken om het rolnummer te herkennen (Hoofdstuk 4). Vervolgens wordt het resultaat van Tesseract vergeleken met het te verwachten rolnummer d.m.v. het LGD-algoritme. De functie geeft *true* terug als beide waarden overéénkomen.

8 Evaluatie

De integratie wordt ingevoerd en uitgebreid getest in de productiehal VF (Gent), waar twee kranen (LK4544 en LK4545) actief zijn. Er wordt gebruikgemaakt van het configuratiebestand voor de productiehallen in Gent (Figuur 6.2). Gedurende een weekend zijn 679 rolverplaatsingen, of 31016 beelden, geëvalueerd door het herkenningsalgoritme.

De resultaten zijn per kraan d.m.v. een logbestand handmatig geclassificeerd in drie categorieën:

- Herkend
- Niet herkend
- Niet bruikbaar

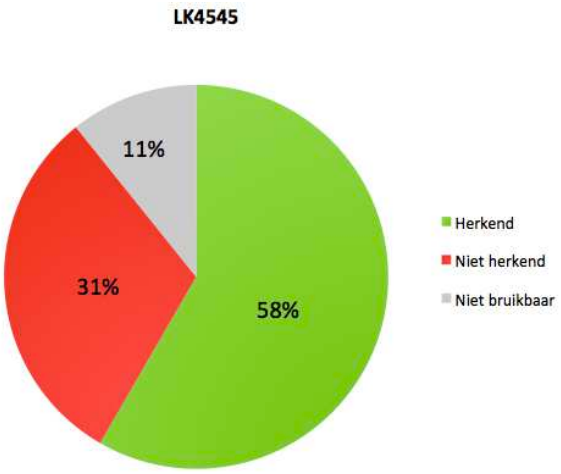
Een rolverplaatsing valt onder de categorie “Herkend” als deze drie keer herkend werd, anders onder de categorie “Niet herkend”.

Onder de categorie “Niet bruikbaar” vallen rolverplaatsingen waarbij de kranen roteren of waarbij het rolnummer (gedurende de volledige verplaatsing) buiten de viewport van de camera valt. Wanneer een kraan roteert, kunnen slechts een paar beelden geëvalueerd worden; daarom beslissen we om deze beelden buiten beschouwing te laten. Daarnaast kan het rolnummer ook omklappen door de snelheid van de kraan, waardoor het rolnummer niet zichtbaar is (Figuur 8.4). We bespreken de resultaten van de twee kranen afzonderlijk.

8.1 LK4545

Uit de resultaten (288 rolverplaatsingen) voor kraan LK4545 kunnen we afleiden dat de camera niet optimaal opgesteld staat voor de herkenning. 31 rolverplaatsingen (11%) konden niet geëvalueerd worden door de rolnummerherkenning, doordat de

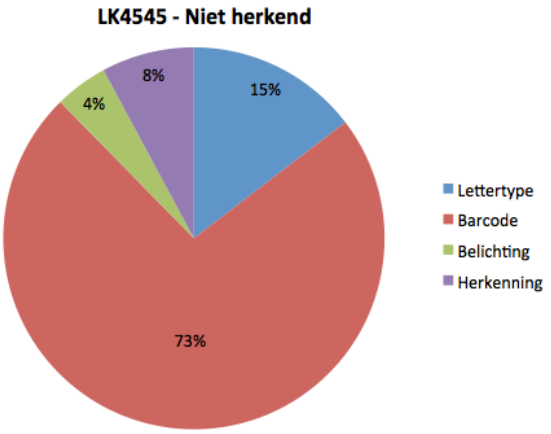
kranen roteren of omdat het label niet zichtbaar was; het label is te hoog op de rol geplaatst. (Figuur 8.1).



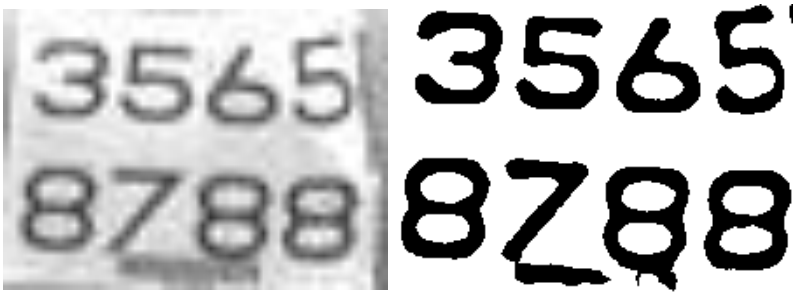
Figuur 8.1: Resultaten LK4544.

89 rolverplaatsingen (31%) werden niet herkend en dit om verschillende redenen (Figuur 8.2):

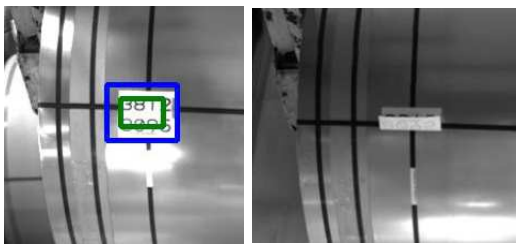
- Barcode (65 rolverplaatsingen): de barcode onder het rolnummer zorgt voor een vervorming van bepaalde cijfers. Hierdoor wordt bijvoorbeeld een “7” en een “9” vaak als een “2” gedetecteerd of een “1” als een “4” (Figuur 8.3).
- Lettertype (14 rolverplaatsingen): het lettertype dat gebruikt wordt voor het rolnummer is niet OCR-vriendelijk. Hierdoor gelijken karakters sterk op elkaar, bijvoorbeeld een “3” op een “8”.
- Belichting (7 rolverplaatsingen): bij sommige rolverplaatsingen is het rolnummer steeds onder- of overbelicht waardoor het zelden zichtbaar is.
- Herkenning (3 rolverplaatsingen): de rolnummerherkenning was niet in staat om het label te detecteren. Dit kan doordat het rolnummer te schuin geplaatst is of omdat de rol geen uniforme belichting heeft; de adaptieve threshold faalt.



Figuur 8.2: Oorzaak van niet herkende rolnummers bij LK4544.



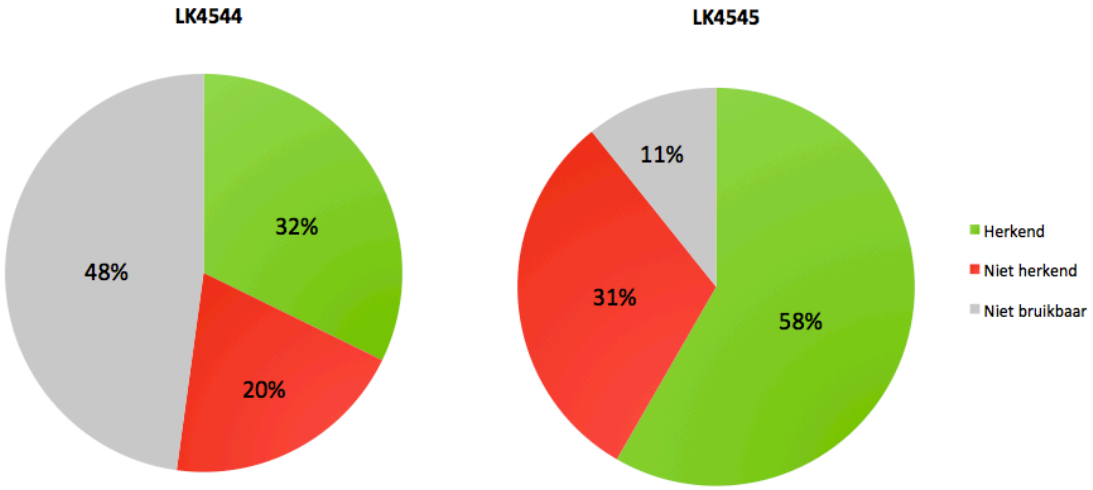
Figuur 8.3: Probleem met barcode.



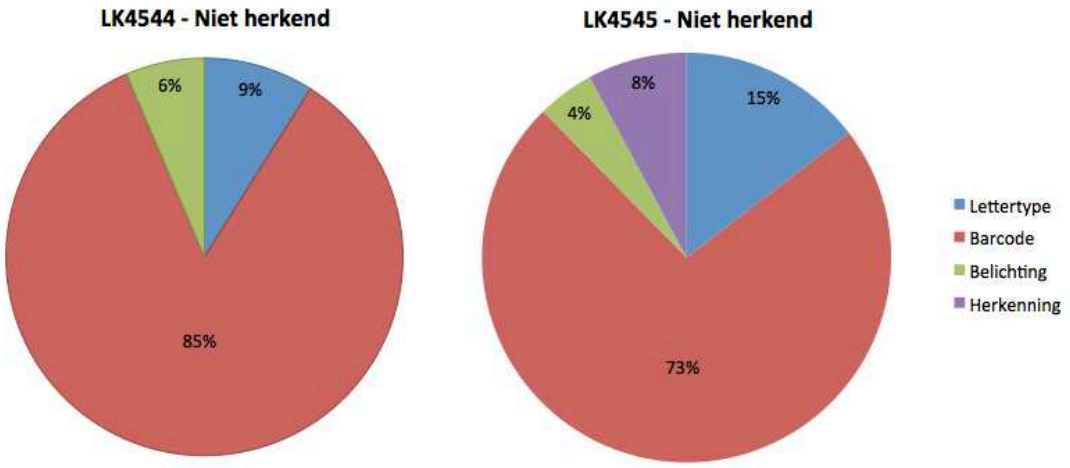
Figuur 8.4: Rolnummer omgeklapt door de snelheid.

8.2 LK4544

Uit de resultaten (391 rolverplaatsingen) voor kraan LK4544 kunnen we afleiden dat de camera veel minder goed opgesteld staat voor de herkenning, dan bij kraan LK4545 (Figuur 8.5). Bij het merendeel van de rolverplaatsingen valt het rolnummer buiten de viewport van de camera; hierdoor kunnen slechts 203 rolverplaatsingen (52%) geëvalueerd worden. 78 rolverplaatsingen konden niet herkend worden; dit is net zoals bij kraan LK4545 grotendeels te wijten aan de barcode en het lettertype van het rolnummer (Figuur 8.6).



Figuur 8.5: Resultaten LK4544 (links) en vergelijking met LK4545 (rechts).

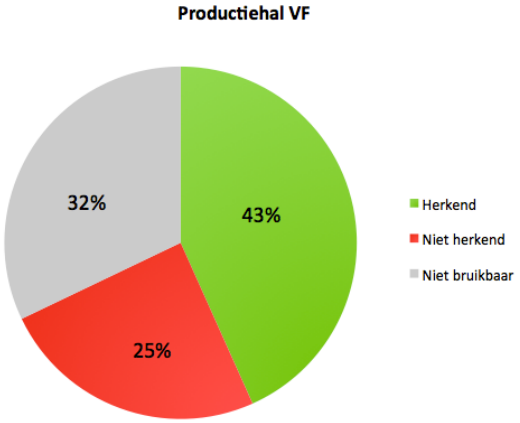


Figuur 8.6: Oorzaak van niet herkende rolnummers bij LK4544.

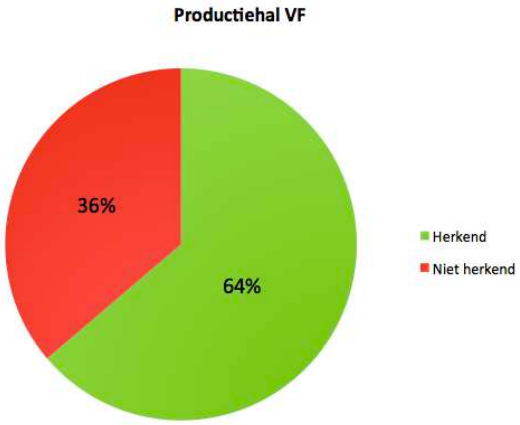
8.3 Productiehal VF

Wanneer we de resultaten van beide kranen samennemen, krijgen we de gewogen gemiddelden afgebeeld op Figuur 8.7. Als we geen rekening houden met de rolverplaatsingen in de categorie "Niet bepaald", dan hebben voor de productiehal VF een herkenningspercentage van 64% i.p.v. 43% (Figuur 8.8). Dit percentage is correcter aangezien de rolnummers van de rolverplaatsingen in deze categorie niet zichtbaar zijn en dus ook niet door een mens kunnen worden herkend.

De impact van een camera die slecht opgesteld staat is echter groot, zo zorgt de camera van kraan LK4544, dat bij ongeveer de helft van de rolverplaatsingen het rolnummer niet zichtbaar is. Als de camera goed opgesteld staat, kan een herkenningspercentage van 60% verwacht worden.

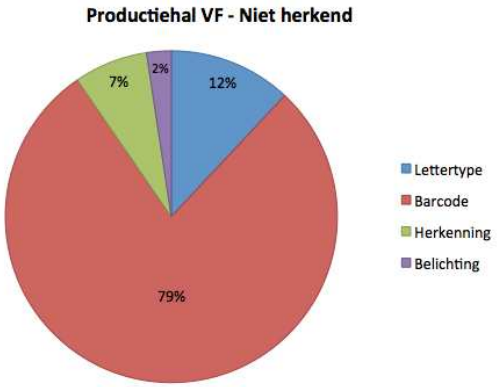


Figuur 8.7: Resultaten productiehal VF.



Figuur 8.8: Resultaten productiehal; enkel bruikbare rolverplaatsingen.

De barcode en het lettertype van het rolnummer zorgen, dat in ongeveer 90% van de gevallen het rolnummer niet herkend kan worden. De overige 10% is te wijten aan de complexe omgeving waarin de herkenning wordt uitgevoerd; onder- en overbelichting.



Figuur 8.9: Niet herkend productiehal VF.

8.4 Uitvoeringstijd

De uitvoeringstijd van de rolnummerherkenning, d.w.z. labelsituering plus OCR, wordt ook gemeten. De gemiddelde uitvoeringstijd berekend over 80 rolverplaatsingen, 5779 evaluaties, bedraagt **61 ms**. Deze meting werd uitgevoerd op een 2.66 GHz Intel Xeon machine (Tabel 8.1). Door de snelle uitvoeringstijd is de rolnummerherkenning bruikbaar in een realtime applicatie.

Tabel 8.1: Uitvoeringstijd

Aantal evaluaties	Minimum	Maximum	Standaardafwijking	Gemiddelde
5779	44 ms	84 ms	8.33 ms	61 ms

8.5 Optimalisatie

Uit Figuur 8.9 kunnen we besluiten dat het grootste probleem (79%) de barcode onderaan het rolnummer is. Ook door het lettertype kan het rolnummer niet altijd herkend worden (12%). Een oplossing voor deze twee problemen kan het herkenningspercentage verbeteren.

8.5.1 Barcode

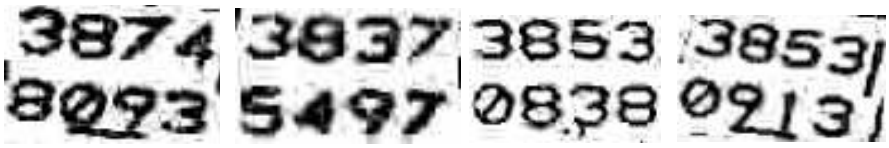
De onderste barcode verwijderen op het rolnummer kan een oplossing zijn. In theorie zou dit geen probleem zijn omdat er twee identieke barcodes op het rolnummer aanwezig zijn (2.1.2). In praktijk zijn er enkele struikelblokken:

1. Herontwerpen van het rolnummer.
2. Handelingen van de arbeiders.

Het herontwerpen van het rolnummer is mogelijk maar bij het invoeren van een nieuw rolnummer komt heel wat extra. Als we terug het rolnummer bekijken, dan zien we een verticale barcode in de rechterbovenhoek en een horizontale barcode onderaan. Dit is nodig zodat een arbeider steeds de barcode kan inlezen zonder zijn pols te veel te moeten draaien, bijvoorbeeld als een rolnummer schuin geplakt is. Op het eerste gezicht lijkt deze reden absurd, maar voor arbeiders die elke dag vaak dezelfde handeling moeten uitvoeren is deze opstelling veel eenvoudiger en minder belastend.

Een betere oplossing zou zijn om voldoende plaats tussen het rolnummer en de barcode te voorzien, zodat ze niet, zoals nu, bijna aan elkaar kleven.

Het probleem kan op het eerste gezicht ook softwarematig opgelost worden, door het cropping-mechanisme bij te sturen (5.2.3). Wanneer het rolnummer gesitueerd is, wordt het bereik van het rolnummer bijgewerkt d.m.v. croppen. Doordat de barcode zeer dicht bij het rolnummer staat, kan het zijn dat na het croppen nog een deel van de barcode aanwezig is. Het probleem is dat we niet a priori weten wanneer een deel van de barcode aanwezig is, zodat we bijvoorbeeld steeds extra zouden moeten croppen onderaan het rolnummer.



Figuur 8.10: Rolnummers met barcode.

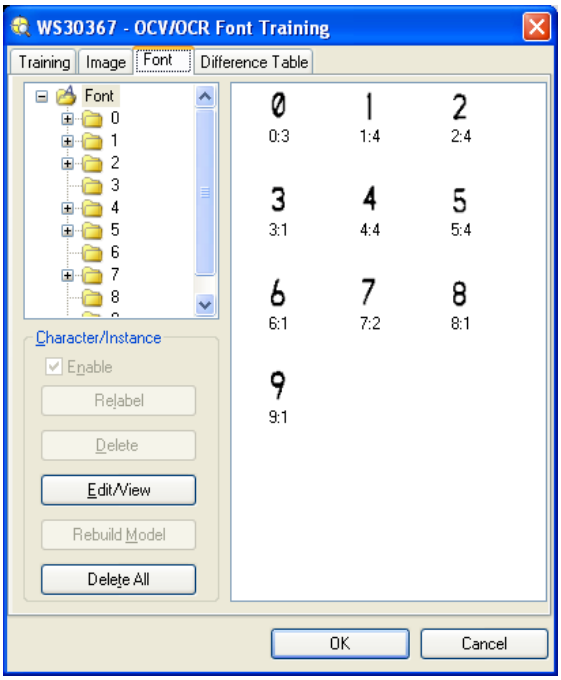
Als een deel van de barcode aanwezig is dan zou deze methode werken, maar soms is de barcode niet meer aanwezig en dan zou croppen een gedeelte van de onderste rolnummers verwijderen. Daarnaast zijn de rolnummers ook vaak schuin geplakt waardoor deze softwarematige oplossing zeker niet veilig is (Figuur 8.10).

8.5.2 OCR-lettertype

Het lettertype (*matrix60x40*) dat voor het rolnummer wordt gebruikt is geen OCR-lettertype. De verschillende karakters gelijken sterk op elkaar en hierdoor worden karakters vaak fout herkend. Tabel 8.2 is berekend met het programma *In-Sight Explorer* van de firma Cognex [54] en toont het verschil (%) aan tussen de verschillende karakters; welke berekening hiervoor gebruikt wordt is niet publiek (Figuur 8.11).

Tabel 8.2: Verschil tussen karakters (%)

-	0	1	2	3	4	5	6	7	8	9
1	71	-	-	-	-	-	-	-	-	-
2	54	84	-	-	-	-	-	-	-	-
3	50	73	39	-	-	-	-	-	-	-
4	73	64	73	72	-	-	-	-	-	-
5	51	63	71	41	84	-	-	-	-	-
6	68	82	81	76	83	38	-	-	-	-
7	67	86	41	68	87	88	90	-	-	-
8	41	63	67	33	82	22	52	83	-	-
9	68	84	76	66	72	60	76	60	55	-

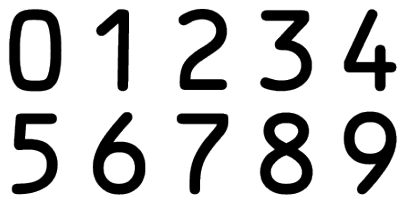


Figuur 8.11: Lettertype rolnummer.

Uit de tabel kunnen we afleiden dat “3” , “5” , “8” en “2”, “7” sterk op elkaar gelijken, dit volgt ook uit het logbestand van de integratie. Daarbij komt ook nog het probleem met de barcode, waardoor de resultaten er zeker niet op verbeteren.

Het lettertype van het rolnummer veranderen is een zeer drastische verandering omdat hiervoor goedkeuring moet gevraagd worden bij de verschillende partijen. De belangrijkste is de kraanman omdat hij het rolnummer moeten kunnen lezen. Daarbuiten moet deze verandering van bovenaf worden aangestuurd en dit kan zeer lang duren. Dit is dan ook de reden waarom er niet werd geëxperimenteerd met andere lettertypes.

Wanneer we de barcode verwijderen en een OCR-lettertype zouden toepassen, bijvoorbeeld OCR-B (Figuur 8.13), dan kunnen de resultaten van de herkenning positief beïnvloed worden. De problemen uit de categorieën: barcode (79%) en lettertype (12%), zullen hierdoor sterk afnemen. Als we voor deze problemen een verbetering van 70% of 90% veronderstellen, wordt voor het herkenningspercentage respectievelijk 86,70% of 93,25% verkregen.



Figuur 8.12: OCR-B.

8.5.3 Rolnummerherkenning

Bij het situeren van het label wordt gebruikgemaakt van een adaptieve threshold (4.4.2). De adaptieve threshold wordt berekend door een gedeelte van de afbeelding te analyseren. In sommige situaties faalt deze methode, zoals bijvoorbeeld bij een niet-uniforme belichting. Bij de berekening van de adaptieve threshold zal de threshold veel te hoog zijn voor de zwakke overgangen van het rolnummer (Figuur 8.13).

Figuur 8.13: Niet-uniforme belichting.

Een mogelijkheid om dit probleem op te lossen is door iteratief de afbeelding te evalueren met een lagere thresholdwaarde en te stoppen wanneer een (gunstige) component gevonden wordt.

9 Besluit

De bedoeling van deze masterproef was om automatisch de nummers van rollen staal te herkennen op camerabeelden. Er werd gestart met een bespreking van de bestaande structuur binnen ArcelorMittal Gent.

De camerabeelden kunnen onderhevig zijn aan vele problemen die het herkenningproces bemoeilijken, zoals de belichting in de productiehallen, het label en de handelingen van de kranen en arbeiders. Daarom werden deze vooraf opgelijst en geanalyseerd.

De voorgestelde oplossing maakt gebruik van technieken uit de beeldverwerking. Er werd aandacht besteed aan het situeren van randen en componenten. Deze werden dan ook uitgebreid onderzocht en besproken.

Aan de hand van deze technieken werd een tweefasige oplossing uitgewerkt. In eerste instantie past deze de beeldwerkingstechnieken toe om het rolnummer te situeren op de afbeelding. Vervolgens wordt het gevonden nummer uitgelezen aan de hand van optical character recognition.

De oplossing werd geïmplementeerd in een offline GUI. Deze laat toe om het herkenningsalgoritme te kalibreren voor een specifieke kraan of productiehal. Daarnaast werd het geheel geïntegreerd in de infrastructuur van ArcelorMittal Gent, wat aantoont dat de oplossing geschikt is voor gebruik in een realtimeomgeving.

Een meer gedetailleerde evaluatie leert dat de situering van het rolnummer slaagt met een betrouwbaarheid van 82%. Een correct gesitueerd nummer kan vervolgens in 64% van de gevallen worden uitgelezen.

Tot slot worden enkele technieken besproken om de prestaties van de oplossing te verbeteren. Niettemin levert de implementatie in zijn huidige vorm reeds zeer gunstige resultaten, en mag dus zeker worden teruggeblikt op een geslaagde masterproef.

Literatuurlijst

[1] ArcelorMittal, "Sterke speler in de regio," s.d. http://www.arcelormittal.com/gent/prg/selfware.pl?id_sitemap=1.

[2] ArcelorMittal, "Zo maakt ArcelorMittal Gent staal" http://www.arcelormittal.com/gent/repository/Publicaties/ZomaaktArcelorMittalGentstaal.pdf.

[3] Cossilys, http://www.cossilys21.com/.

[4] ArcelorMittal, "Big Brother in de warmwalserij," *Big Brother in de warmwalserij*, 2010 http://www.arcelormittal.com/gent/prg/selfware.pl?id_sitemap=960.

[5] M. Slembrouck, "Multicamera bewegingsgestuurde computerinterface," 2011 http://aleph.ugent.be/F/L9BDFBMBH9HK63KBS7IKVKRF8CKLL23FG8XMC791RDK2SU5R3T-03314?func=full-set-set&set_number=000803&set_entry=000001&format=999.

[6] R.C. Gonzalez and R.E. Woods, *Digital image processing*, Upper Saddle River, N.J.: Prentice Hall, 2008.

[7] MPEG-4 http://en.wikipedia.org/wiki/MPEG-4.

[8] B. Lovell, "Thresholding Techniques," 2003.

[9] A. Greensted, "Otsu Thresholding," *Otsu Thresholding (Java implementation)* http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html.

[10] P.C. Rossin, "Image Filtering & Convolution," (s.d.) http://www.cse.lehigh.edu/~spletzer/rip_f06/lectures/lec003_Filtering.pdf.

[11] A. C. Bovik, *The Essential Guide to Image Processing*, 2009.

[12] Mean filter, s.d. http://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm.

[13] Guassian filter, s.d. http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm.

[14] Median Filter, s.d. http://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm.

[15] M.A. Oliveira, N.J. Leite, A multiscale directional operator and morphological tools for reconnecting broken ridges in fingerprint images, Pattern Recognition, Volume 41, Issue 1, January 2008, Pages 367-377, ISSN 0031-3203

[16] Canny, John, "A Computational Approach to Edge Detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.PAMI-8, no.6, pp.679,698, Nov. 1986

[17] N. Neeta, J. Gaurav, G. Ashish, and J. Anshul, "Dynamic Thresholding Based Edge Detection," 2008.

[18] Edge Detection, http://www1.idc.ac.il/toky/imageProc-08/lectures/06_edgeX4.pdf.

[19] Salt and pepper noise, http://en.wikipedia.org/wiki/File:Noise_salt_and_pepper.png.

[20] Wenshuo Gao; Xiaoguang Zhang; Lei Yang; Huizhong Liu, "An improved Sobel edge detection," *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on* , vol.5, no., pp.67,71, 9-11 July 2010

[21] Canny, John, "A Computational Approach to Edge Detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.PAMI-8, no.6, pp.679,698, Nov. 1986

[22] CSE473/573 Fall 2010 Lecture Charts: Image Pre-preprocessing http://www.cse.buffalo.edu/courses/cse573/peter/lec/04.html.

[23] Gunn, S.R., "Edge detection error in the discrete Laplacian of Gaussian," *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on* , vol.2, no., pp.515,519 vol.2, 4-7 Oct 1998

[24] G.T. Shrivakshan and D.C. Chandrasekar, "A Comparison of various Edge Detection Techniques used in Image Processing," 2012 http://ijcsi.org/papers/IJCSI-9-5-1-269-276.pdf.

[25] P. Veelaert, "Hough Transform."

[26] Satoshi Suzuki, KeiichiA be, Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, and Image Processing, Volume 30, Issue 1, April 1985, Pages 32-46, ISSN 0734-189X

[27] M. Jitendra, "Contour and Texture Analysis for Image Segmentation," 2001 http://www.eng.utah.edu/~bresee/compvision/files/MalikBLS.pdf.

[28] Fu Chang; Chun-Jen Chen, "A component-labeling algorithm using contour tracing technique," *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on* , vol., no., pp.741,745, 3-6 Aug. 2003

[29] A. Talal Sleut, "A Chain Code Approach for Recognizing Basic Shapes."

[30] VXL http://vxl.sourceforge.net/.

[31] TargetJr http://www.targetjr.org/.

[32] VXL http://sourceforge.net/p/vxl/git/ci/master/tree/core/vxl_copyright.h.

[33] VXL (Book) http://public.kitware.com/vxl/doc/release/books/core/book.html.

[34] VXL (repository) http://sourceforge.net/projects/vxl/.

[35] LTI http://ltilib.sourceforge.net/doc/homepage/index.shtml.

[36] LTI Aachen University http://www.rwth-aachen.de/.

[37] LTI (documentatie) http://ltilib.sourceforge.net/doc/html/index.shtml.

[38] OpenCV website http://opencv.org/.

[39] G.R. Bradski and A. Kaehler, *Learning OpenCV: computer vision with the OpenCV library*, Sebastopol, CA: O'Reilly, 2008.

[40] Emgu CV website http://www.emgu.com/wiki/index.php/Main_Page.

[41] D. Karatzas, S. Robles Mestre, F. Nourbakhsh, and J. Mas, "ICDAR 2011 Robust Reading Competiti," 2011 http://www.cvc.uab.es/icdar2011competition/images/Report_RobustReading_Challenge1_final.pdf.

[42] S.M. Hanif and L. Prevost, "Texture based text detection in natural scene images," (s.d.) http://ceur-ws.org/Vol-415/paper2.pdf.

[43] Yuan, Q.; Tan, C.L., "Text extraction from gray scale document images using edge information," *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on* , vol., no., pp.302,306, 2001

[44] M. Alata and M. Al-Shabi, "Text detection and character recognition using fuzzy image processing," 2006 http://iris.elf.stuba.sk/JEEEC/data/pdf/5_106-3.pdf.

[45] M. Pietikäinen and O. Okun, "Text extraction from grey scale page images by simple edge detectors," (s.d.) http://www.ee.oulu.fi/mvg/files/pdf/pdf_148.pdf.

[46] A. Dutta, "Gradient based Approach for Text Detection in Video Frames" http://www.comp.nus.edu.sg/~tancl/publications/c2009/ICSIP2009-3.pdf.

[47] R. Stoop, *Algoritmen II*.

[48] C. Patel, A. Patel, and D. Patel, "Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study," 2012 http://research.ijcaonline.org/volume55/number10/pxc3882784.pdf.

[49] R. Smith, "An Overview of the Tesseract OCR Engine" http://research.google.com/pubs/archive/33418.pdf.

[50] N. White, "Training Tesseract for Ancient Greek OCR," 2013 http://www.eutypon.gr/eutypon/pdf/e2012-29/e29-a01.pdf.

[51] Google, "Tesseract Official Website" https://code.google.com/p/tesseract-ocr/.

[52] "Tesseract OCR Chopper" http://pp19dd.com/tesseract-ocr-chopper/.

[53] C. Verstraeten, "Rolnummerherkenning C# applicatie" https://github.com/cedricve/label-recognition-emgucv.

[54] Cognex website http://www.cognex.com/.

[55] R. Laganière, *OpenCV 2 computer vision application programming cookbook over 50 recipes to master this library of programming functions for real-time computer vision*, Birmingham, UK: Packt Pub., 2011 http://site.ebrary.com/id/10477260. Last accessed on February 22, 2013.

[56] ArcelorMittal, "Van GPS naar DGPS in het plakkenpark," s.d. http://www.arcelormittal.com/gent/prg/selfware.pl?id_sitemap=101&language=NL.

[57] L. Wang and J. Bai, "Threshold selection by clustering gray levels of boundary," 2003

[58] R. Stoop, *Algoritmen I.*

[59] Shivakumara, P.; Trung Quy Phan; Tan, C.L., "A Gradient Difference Based Technique for Video Text Detection," *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on* , vol., no., pp.156,160, 26-29 July 2009

[60] Trung Quy Phan; Shivakumara, P.; Tan, C.L., "A Laplacian Method for Video Text Detection," *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on* , vol., no., pp.66,70, 26-29 July 2009

[61] R. Velázquez, E. Pissaloux, and P. Le Polotec, "Towards a local spatial representation system for mobility assistance of the blind"