

2013•2014  
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN  
*master in de industriële wetenschappen: energie*

## Masterproef

Visioncamera gebaseerde kwaliteitscontrole van strippen

Promotor :  
ing. Joseph THEUNISSEN

Promotor :  
Ing. ERIC JOOSTEN  
Ing. PAUL MEYNEN

Frederik Scheelen , Wouter Ferson

*Proefschrift ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie*

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2013•2014  
Faculteit Industriële  
ingenieurswetenschappen  
*master in de industriële wetenschappen: energie*

## Masterproef

Visioncamera gebaseerde kwaliteitscontrole van strippen

Promotor :  
ing. Joseph THEUNISSEN

Promotor :  
Ing. ERIC JOOSTEN  
Ing. PAUL MEYNEN

Frederik Scheelen , Wouter Ferson

*Proefschrift ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie*

## Woord vooraf

Deze masterproef is het eindstation van onze opleiding voor het bekomen van de graad van Industrieel Ingenieur aan de associatie KU Leuven en UHasselt. We hebben deze thesis uitgewerkt bij Vandersanden Group nv. in Lanklaar. De kern van deze masterproef bevindt zich vooral in het visiegebeuren en dus het verwerken van camerabeelden met behulp van een softwareprogramma. Aangezien de industrie steeds meer geautomatiseerd wordt met behulp van visie en het hele visiegebeuren voor ons nog een totaal nieuw gegeven was, leek het ons een enorme uitdaging om deze masterproef tot een goed einde te brengen.

Gedurende het academiejaar hebben we leren werken in teamverband alsook contacten leren leggen met verschillende bedrijven om een bepaald gegeven te kunnen uitwerken. Aangezien we nog nooit in aanraking zijn gekomen met visiesystemen, was het noodzakelijk om het gebruik van enkele softwareprogramma's zoals *HALCON* en *Visual Studio* aan te leren. Verder in het academiejaar hebben we van *ACRO* de kans gekregen de ontwikkelde systeemprogramma's effectief uit te testen op een testopstelling met daarbij het vrije gebruik van de nodige materialen. Deze testen waren in het begin niet foutloos, waardoor duidelijk werd dat implementatie van de theorie naar de praktijk zeker geen evidentie is. Elke fout werd uiteindelijk stap per stap opgelost tot het bekomen van een werkend systeem. Dit alles heeft meegeholpen aan een leerrijke ervaring die zeker van pas zal komen bij het intreden van het bedrijfsleven.

Het volledige academiejaar konden we rekenen op de hulp en ondersteuning van een aantal personen. Via deze weg willen we die mensen nog eens extra bedanken. Allereerst zijn er onze promotoren Ing. Eric Joosten, Ing. Paul Meynen en Ing. Jos Theunissen die ons steeds hulp en ondersteuning hebben geboden bij de masterproef. We danken ook de medewerkers van *ACRO* met in het bijzonder Ing. Maarten Verheyen en Ing. Geert Moonen voor technische ondersteuning en tips i.v.m. visietechnieken, alsook voor het verkrijgen van het nodige materiaal voor de gebruikte testopstelling. Vervolgens willen we ook de directie, docenten en personeelsleden van deze associatie bedanken voor hun begeleiding en ondersteuning gedurende het volledige studietraject. Ten slotte, maar zeker niet minder belangrijk, bedanken we ook onze ouders voor de mogelijkheden die ze ons hebben gegeven, voor hun hulp en onvoorwaardelijke steun.



# Inhoudsopgave

Woord vooraf.....	1
Lijst van tabellen .....	5
Lijst van figuren.....	7
Abstract.....	11
Abstract in English.....	13
1 Inleiding .....	15
1.1 Situering .....	16
1.2 Probleemstelling.....	17
1.3 Doelstellingen.....	18
1.4 Materiaal en methode.....	19
2 Vandersanden NV – Bedrijfsprofiel .....	21
2.1 Historiek.....	21
2.2 Ambities en toekomstplannen.....	23
3 Het baksteenproces: van leem tot steen.....	25
4 Strippenfacer.....	27
4.1 Concepten .....	27
4.2 Bevestiging.....	30
4.3 Materiaalkeuze.....	32
4.3.1 Staal-S235.....	32
4.3.2 Roestvrij staal (RVS/Inox).....	34
4.3.3 Aluminium.....	34
4.4 Keuzes en bestelling .....	35
4.5 Implementatie.....	36
5 Het visiesysteem.....	37
5.1 Opzet van het visiesysteem .....	37
5.2 Onderdelen van het visiesysteem .....	38
5.2.1 Camerakeuze.....	38
5.2.2 Lens.....	40
5.2.3 Belichting.....	41
5.2.4 Lijnlaser.....	44
5.2.5 Het frame .....	45
5.3 Uitwerpsysteem (actuator) .....	47
5.4 Hoogtemeting.....	47
5.4.1 Lasertriangulatie .....	48
5.4.2 Hoogtemeting a.d.h.v. een smartcamera.....	49
5.4.3 Hoogtemeting a.d.h.v. belichtingscontrole (schaduwvorming) .....	49
6 Beeldverwerking .....	51
6.1 Keuze van het softwarepakket.....	51
6.2 Kalibratie.....	51
6.2.1 Werkwijze van kalibratie .....	52
6.2.2 Kalibratieplaat.....	54
6.3 Objectdetectie.....	55

6.3.1 Beelden inlezen .....	55
6.3.2 Distorsie .....	55
6.3.3 Decompose – beeldontleding .....	56
6.3.4 Threshold.....	57
6.3.5 Connectie en regioselectie.....	57
6.3.6 Softwarematige sensor .....	58
6.4 Kwaliteitscontrole van de steenstrip.....	59
6.4.1 Contourcontrole.....	59
6.4.2 Lengte- en breedtebepaling van de steenstrip.....	59
6.4.3 Rechthoekigheid en vormcontrole.....	60
6.4.4 Hoekcontrole.....	61
6.4.5 Spievormcontrole a.d.h.v. lasertriangulatie .....	65
6.4.6 Automatische reset bij foutdetectie.....	75
7 Gebruikersinterface in Visual Studio .....	77
7.1 Programmacode exporteren naar VB.Net .....	77
7.2 Het ontwerp .....	81
7.2.1 Het inlogscherm.....	81
7.2.2 Resultatenscherm .....	83
7.2.3 Instellingenscherm .....	84
7.3 De code.....	86
7.3.1 Code inlogscherm.....	86
7.3.2 Code resultatenscherm .....	88
7.3.3 Code instellingenscherm .....	90
8 Testopstelling visiesysteem.....	93
8.1 Testopstelling.....	93
8.1.1 Camera testopstelling.....	94
8.1.2 Lens testopstelling.....	95
8.1.3 Belichting testopstelling.....	95
8.1.4 Lijnlaser testopstelling.....	96
8.1.5 Frame testopstelling .....	96
8.2 Testresultaten .....	97
8.2.1 Nauwkeurigheid van lengte-en breedtemeting.....	97
8.2.2 Vorm- en hoekcontrole .....	98
8.2.3 Hoogtemetingen.....	99
8.2.4 Controle op spievormigheid .....	99
8.2.5 Snelheidstest en advies voor verhoging van de systeemsnelheid .....	100
9 Microsoft SQL Server (database).....	103
9.1 Database en tabel aanmaken .....	103
9.2 Connectie tussen Visual Studio en SQL Server .....	105
10 Conclusie .....	107
10.1 Algemene conclusie.....	107
10.2 Kostenraming .....	108
Literatuurlijst.....	109
Bijlagen.....	111

## Lijst van tabellen

Tabel 1: vergelijking tussen COGNEX en HALCON .....	51
Tabel 2: sorteren van rijen en kolommen in HALCON.....	69
Tabel 3: overzicht van bussystemen voor beeldtransfer .....	101
Tabel 4: kostenraming (schatting).....	108





## Lijst van figuren

Figuur 1: een baksteenstrip .....	18
Figuur 2: baksteenproces in 1925 .....	21
Figuur 3: modernisering van de fabriek.....	22
Figuur 4: hedendaags assortiment .....	22
Figuur 5: rollenzeef .....	25
Figuur 6: stapelwijze voor het bakproces.....	26
Figuur 7: strippenfacer ontwerp 1.....	28
Figuur 8: strippenfacer ontwerp 2.....	28
Figuur 9: strippenfacer ontwerp 3.....	28
Figuur 10: strippenfacer ontwerp 4.....	29
Figuur 11: detailtekening strippenfacer ontwerp 4.....	29
Figuur 12: strippenfacer ontwerp 5.....	29
Figuur 13: bevestigingsmogelijkheid 1.....	30
Figuur 14: bevestigingsmogelijkheid 2.....	31
Figuur 15: bevestigingsmogelijkheid 2.....	31
Figuur 16: bevestigingsmogelijkheid 3.....	31
Figuur 17: detailtekening bevestigingsmogelijkheid 3.....	31
Figuur 18: prijs referentieplaat in staal.....	33
Figuur 19: prijs referentiesplaat in RVS .....	34
Figuur 20: prijs referentiesplaat in aluminium .....	35
Figuur 21: proefopstelling strippenfacer.....	36
Figuur 22: factoren voor berekening sluitertijd.....	39
Figuur 23: variabelen bij de lens .....	41
Figuur 24: voorbeeld diffuse belichting.....	42
Figuur 25: voorbeeld ringbelichting .....	42
Figuur 26: voorbeeld directionele strijkbelichting.....	43
Figuur 27: voorbeeld gepolariseerd licht.....	43
Figuur 28: verschil tussen homogene en Gaussiaanse lijnlaser .....	45
Figuur 29: frame camera.....	46
Figuur 30: opstelling camera.....	46
Figuur 31: lasertriangulatie.....	48
Figuur 32: Gocator van LMI.....	49
Figuur 33: toepassing strijkbelichting.....	50
Figuur 34: kalibratiescherm - setup.....	52
Figuur 35: kalibratiebeeld.....	53
Figuur 36: kalibratieplaat.....	54
Figuur 37: genereren kalibratiepatroon.....	54
Figuur 38: realtime-beelden inlezen.....	55
Figuur 39: distorsie.....	55
Figuur 40: origineelbeeld met respectievelijk R-,G-,B- beeld.....	56
Figuur 41: thresholdoperatie.....	57
Figuur 42: smallest_rectangle steenstrip.....	60

Figuur 43: centerpunt van de steenstrip.....	61
Figuur 44: pixelrijen en –kolommen van een beeld .....	62
Figuur 45: oriënteren van de steenstrip.....	62
Figuur 46: oplegging van het rechthoekig masker met uitvergroting van de gefilterde hoek .....	64
Figuur 47: uitfiltering van laserlijn.....	66
Figuur 48: zichtbare lijnen bij strippenovergang.....	67
Figuur 49: merkerpunten laserlijn .....	68
Figuur 50: referentielijnen voor hoogtemeting.....	69
Figuur 51: procedure merkerpunten.....	70
Figuur 52: procedure afstand merkerpunt tot referentielijn .....	71
Figuur 53: procedure berekening gemiddelde hoogte.....	72
Figuur 54: groeperen en uitmiddelen.....	74
Figuur 55: procedure groeperen en uitmiddelen.....	74
Figuur 56: referentie toevoegen .....	77
Figuur 57: nieuwe procedure aanmaken.....	78
Figuur 58: de verschillende procedures.....	79
Figuur 59: keuze procedure.....	79
Figuur 60: exporteren van procedure .....	79
Figuur 61: lijst van exportbestanden van de procedures.....	80
Figuur 62: geëxporteerd bestand geopend in Visual Studio.....	80
Figuur 63: eerste scherm (loginscherm) in Visual Studio.....	81
Figuur 64: tweede scherm (resultatenscherm) in Visual Studio .....	83
Figuur 65: derde scherm (instellingenscherm) in Visual Studio .....	85
Figuur 66: code van het loginscherm.....	87
Figuur 67: standaard code van de login knop.....	88
Figuur 68: starten van de timer .....	88
Figuur 69: stoppen van de timer .....	89
Figuur 70: het leegmaken van de tekst vakken en het venster.....	89
Figuur 71: het eerste beeld van de camera weergeven.....	89
Figuur 72: het weergeven van de lengte en breedte van de strip.....	89
Figuur 73: het weergeven van een goedgekeurde contour inkeping.....	90
Figuur 74: het weergeven van een afgekeurde contour inkeping.....	90
Figuur 75: verwijzing naar de actuele threshold waardes .....	90
Figuur 76: code van threshold-ondergrens .....	91
Figuur 77: schuifbalk van threshold-ondergrens.....	91
Figuur 78: instellen standaard waarden threshold .....	91
Figuur 79: schuifbalken met standaard waarden.....	91
Figuur 80: testopstelling .....	94
Figuur 81: metalen referentieblok.....	97
Figuur 82: strip met een contourdefect.....	98
Figuur 83: gecontroleerde steenstrippen .....	98
Figuur 84: testopstelling met implementatie van rollenbaan .....	100
Figuur 85: connectie met Server.....	103
Figuur 86: database aanmaken .....	103
Figuur 87: tabel aanmaken .....	104

Figuur 88: maken van kolommen.....	104
Figuur 89: de aangemaakte tabel .....	104
Figuur 90: connectie database toevoegen .....	105
Figuur 91: nieuwe DataSet.....	105
Figuur 92: code voor connectie met database .....	106
Figuur 93: verwijzen naar DatabaseConnectie.....	106



## Abstract

Vandersanden Group in Lanklaar gebruikt een strippenzaag voor het maken van baksteenstrippen en ondervindt hierbij nieuwe uitdagingen. Door de stijgende vraag dient namelijk de productie te stijgen met behoud van kwaliteit. De huidige manuele kwaliteitscontrole heeft echter een negatieve invloed op die eis. Deze masterproef heeft als doel de controle van de strippen te automatiseren met behulp van een visiesysteem. Bijkomend onderzoekt de masterproef of een productiesnelheid van 40 strippen per minuut mogelijk is na automatisatie van de kwaliteitscontrole.

De beeldverwerking van het visiesysteem werd ontwikkeld met behulp van HALCON, de gebruikersinterface met MS Visual Studio. Deze interface maakt het voor de gebruiker mogelijk het visiesysteem te beheren en indien nodig de belangrijkste parameters aan te passen. De strippen dienen echter enkel langs de ruwe zijde een controle te ondergaan. Een kantelsysteem zorgt ervoor dat deze zijde altijd naar de camera van het visiesysteem wordt gericht.

De gewenste kwaliteit op het gebied van vorm, afmetingen en spievormigheid van de strippen wordt behaald door middel van het visiesysteem. Tevens heeft het systeem een meetnauwkeurigheid van  $\pm 0,5$  mm op de lengte en breedte van de strip. Een database (MS SQL Server) houdt deze afmetingen bij met het oog op eventuele toekomstige procesverbeteringen. Tot slot geven testen aan dat een productiesnelheid van 56 strippen per minuut haalbaar is na automatisatie van de kwaliteitscontrole.



## Abstract in English

Vandersanden Group in Lanklaar uses a special saw for making strips from bricks and hereby faces new challenges. Because of the growing demand, an increase in production while maintaining the quality is thereby required. However, the current manual quality control has a negative influence on this requirement. This master's thesis aims on an automatic quality control of the strips using a vision system. In addition it examines if a production speed of 40 strips per minute is possible after automation of this quality control.

The image processing of the vision system is developed using HALCON, the user interface with MS Visual Studio. This interface makes it possible for the user to operate the vision system and to adjust the most important parameters if necessary. The strips only have to be inspected along their rough side. An overturn system ensures that this side is always directed towards the camera of the vision system.

The desired quality in terms of contour shape, measurements and wedge shape is achieved by the vision system. This system also has a measurement precision of  $\pm 5$  mm on the length and width of the strips. The measurements are stored in a database (MS SQL Server) in view of potential process improvements in the future. Finally, the test results show that a production speed of 56 strips per minute is possible after automation of the quality control.





# 1 Inleiding

Vandersanden Group nv. in Lanklaar is een steenfabriek en produceert bakstenen in verschillende vormen, kleuren en maten. Uit deze bakstenen kunnen steenstrippen gezaagd worden welke tegenwoordig vaak bij nieuwbouw en renovaties worden gebruikt. Door de stijgende populariteit van deze strippen, moet vraag en aanbod in 'evenwicht' worden gehouden door middel van een productieverhoging. Vandersanden Group nv. wil deze productieverhoging realiseren door het productieproces van de steenstrippen te versnellen zonder kwaliteitsverlies. De kwaliteitscontrole dient dus efficiënt en sneller te gebeuren dan bij de huidige manuele controle. Het ontwerp van een visiesysteem dringt zich dus op en het is dit ontwerp wat de kern van deze masterproef vormt.

In dit eerste inleidende hoofdstuk wordt nog dieper ingegaan op de onderzoeksopzet van deze masterproef met aandacht voor de situering, probleemstelling, doelstellingen en methodiek. In het tweede hoofdstuk wordt kort de geschiedenis van Vandersanden Group nv. vermeld samen hun ambities en toekomstplannen. Vervolgens geeft het derde hoofdstuk meer inzicht op het productieproces van bakstenen. In hoofdstuk 4 worden het ontwerp en de testen van de strippenfacer besproken, een kantelsysteem om de strippen steeds met de gesneden kant op de transport te laten vallen. Men wenst namelijk enkel het ruwe oppervlak te controleren die op die manier naar de camera van het visiesysteem wordt gericht. Hoofdstuk 5 gaat dan weer dieper in op de opbouw van het visiesysteem en mogelijke oplossingen om de hoogte van de strippen te kunnen meten. Deze hoogtemeting is namelijk van belang bij de controle op spievormigheid. Vervolgens geeft hoofdstuk 6 de volledige bespreking van de beeldverwerking in HALCON, de kern van de kwaliteitscontrole. Dit hoofdstuk geeft een gedetailleerde bespreking over de programmatie van de verschillende facetten van de kwaliteitscontrole. In hoofdstuk 7 wordt dan weer getoond hoe de code uit HALCON naar MS Visual Studio kan worden geëxporteerd. Met behulp van MS Visual studio wordt dan een gebruikersinterface voor het visiesysteem ontworpen, wat eveneens in dit hoofdstuk is weergegeven. In hoofdstuk 8 worden verschillende testen op de werking van het visiesysteem geanalyseerd. Naast het behalen van de gewenste kwaliteit, gaan de testen ook na tot welke productiesnelheid het visiesysteem de kwaliteitscontrole nog op een goede manier kan uitvoeren. In hoofdstuk 9 wordt besproken hoe de gemeten lengte, breedte en (gemiddelde) hoogte van een gecontroleerde strip kan opgeslagen worden in een database (MS SQL Server). Tot slot volgt er nog een besluit en een kostenraming (schatting) in hoofdstuk 10.

## 1.1 Situering

Deze masterproef vindt, zoals eerder gezegd, plaats bij Vandersanden Steenfabrieken NV te Lanklaar. De opdracht bestaat erin een visiesysteem te ontwerpen voor het automatisch uitsorteren van slechte en goede steenstrippen aan de uitgang van de strippenzaag.

Vandersanden Steenfabrieken NV is een familiebedrijf dat door de jaren heen al heel wat verschillende productielocaties heeft opgericht. Op dit moment zijn er zowel twee in België als in Nederland gevestigd. Die zorgen voor een totale afzet van circa 60 miljoen straatbakstenen en 320 miljoen gevelstenen per jaar. In Lanklaar produceert men drie soorten stenen:

- de klassieke baksteen of gevelsteen;
- steenstrippen;
- handgemaakte profielstenen (bv. jaartalstenen).

De steenstrippen (zie figuur 1 in paragraaf 1.3) worden gemaakt van zelfgeproduceerde bakstenen. Uit elke baksteen kan men namelijk twee steenstrippen (met vooraf ingestelde dikte) halen door middel van een strippenzaag. Deze strippen moeten een kwaliteitscontrole ondergaan op basis van een visiesysteem.

Baksteenstrippen komen steeds meer op in de moderne bouwwereld. Veel mensen gebruiken deze strippen bij renovaties. Zo is het mogelijk om deze strippen in een gewenst patroon te schikken en te lijmen op een isolatielaag. De klant krijgt zo een nieuw gerenoveerd uitzicht van het huis en tevens een betere isolatie. Zeker bij de renovatie van zeer oude huizen is dat een creatieve maar ook effectieve oplossing.

De markt van baksteenstrippen blijft dus groeien. Dat leidt tot het volledig automatiseren van de productie. De kwaliteitscontrole is daarbij een belangrijk gegeven. Een slechte kwaliteit (barsten, misvorming) zou immers kunnen leiden tot het verlies van klanten, wat niet gewenst is voor een bedrijf. Door de automatisering kan de huidige manuele kwaliteitscontrole en uitsortering vervangen worden door een visiesysteem en kan er eventueel verder nagedacht worden over het geautomatiseerd verpakken van de steenstrippen, hetgeen een FTE (*full-time equivalent*) kan besparen. Deze masterproef is daarom zeker geschikt voor het onderzoek naar de implementatie van een dergelijk visiesysteem.

## 1.2 Probleemstelling

Baksteenstrippen zijn steeds meer in opmars. Dat zorgt natuurlijk voor een duidelijke stijging in productieaantallen. Met grotere productieaantallen is onafscheidelijk ook een betere/nauwkeurigere kwaliteitscontrole nodig. Een mindere kwaliteit kan immers zorgen voor barsten, breuken, ... maar ook de vorm van de steen moet binnen bepaalde grenzen blijven.

Momenteel worden de steenstrippen handmatig gecontroleerd en al dan niet verwijderd door een werknemer. De werkmans moet enerzijds het controleren en (eventueel) verwijderen van de steenstrip realiseren, anderzijds moet hij deze steenstrippen verpakken op een kort tijdsbestek. Er komen namelijk 40 strippen per minuut op de transportband terecht waardoor een grondige controle dus veel inspanning kost voor de arbeider. De steenstrippen die uit de strippenzaag komen, vallen van 5 cm hoogte op een transportband. Op deze transportband liggen dus zowel goede, slechte (kwaliteitsverlies, kleine barsten) als gebroken strippen. Gebroken strippen zijn eenvoudig te detecteren en te verwijderen. Minder kwalitatieve steenstrippen zijn echter moeilijker om eruit te halen. Spievorming en contourfouten zijn vaak moeilijker te detecteren. Door het automatiseren van dit proces kan een nauwkeurigere en constantere kwaliteitscontrole gebeuren. Ook vanuit financieel oogpunt is automatisering van dit proces interessant aangezien het op termijn mogelijk is de kost van een werknemer te besparen.

Een ander bekend fenomeen is het onrechtmatig verwijderen van goede strippen. Manuele controle is immers niet alleen gebaseerd op bepaalde criteria maar ook op het gevoel van de werknemer. Zo kan een strip door de ene worden goedgekeurd en door de andere worden afgekeurd. Dat alles zorgt bijgevolg voor een daling van de te verkopen producten met extra verliezen van energie, tijd en kost die nodig waren bij de productie van dergelijke steenstrippen. Economisch is dat dus ook een nadeel voor het bedrijf.

Tegenwoordig gebruiken veel bedrijven een visiesysteem voor het controleren van de gefabriceerde producten. Vandersanden Group nv. vindt dat ook een interessant systeem. Een visiesysteem kan in deze toepassing namelijk zorgen voor een nauwkeurige controle van de strippen en eventuele afvoer van de afgekeurde strippen. Deze handelingen moeten gebeuren in dezelfde tijd als voor de arbeider is voorzien. Wanneer de controle en eventuele verwijdering gebeurd is, moet de arbeider in hetzelfde tijdsbestek als vroeger enkel instaan voor de verpakking. Door het correct instellen van de taktijden is het nu dan ook mogelijk de productiesnelheid te verhogen, aangezien de werknemer geen kwaliteitscontrole meer hoeft te doen. Een visiesysteem voert de kwaliteitscontrole immers sneller uit dan een werknemer. Daarbij moet wel rekening worden gehouden dat de afvoer ook niet te snel gebeurt. De werknemer moet namelijk de nodige tijd krijgen om de strippen te verpakken (minimum één minuut voor 44 steenstrippen).

Een ander probleem is dat het niet haalbaar is om steeds de productie stop te zetten wanneer er iets getest zal moeten worden, waardoor het maken van een testopstelling zich opdringt en de manier van werken toch wat anders zal zijn (simulatie van omgevingsfactoren zoals een transportband).

### 1.3 Doelstellingen

De masterproef heeft als hoofddoel de kwaliteitscontrole van steenstrippen te automatiseren. Dit is echter op te delen in verschillende deeldoelstellingen. Als eerste moet de strip steeds met de ruwe zijde naar boven op de transportband vallen zodat de kwaliteitscontrole steeds ten opzichte van dit oppervlak gebeurt (enkelzijdige controle is voldoende). Momenteel vallen de strippen niet altijd op hun goede zijde waardoor bijgevolg een gecontroleerde val van de strippen moet gerealiseerd worden. Een tweede deeldoelstelling bestaat uit het automatisch controleren van de strip met behulp van een visiesysteem. Deze controle bepaalt wat er met de strip moet gebeuren: verwijderen of verpakken. Op dit moment wordt dat namelijk handmatig gedaan, waardoor een verhoging van de productiecapaciteit moeilijk realiseerbaar is. De werknemer moet bij een verhoging van de capaciteit (productiesnelheid) nog in staat zijn om de aanvoer van de strippen te kunnen volgen. De criteria waarop de volledige strip moet gecontroleerd worden zijn vorm, afmetingen en spievormigheid.

Bijkomend moeten ook de afmetingen (lengte, breedte en hoogte) van de strip bijgehouden worden in een database. Dat is nodig om de strippen die buiten de toleranties vallen te verwijderen en op deze manier kan men ook controleren of de strippenzaag goed is afgesteld. De strip kan namelijk twee verschillende diktes hebben: 20 mm of 30 mm.

De huidige productie bedraagt 40 strippen per minuut. Dit is de standaard waaraan het visiesysteem zeker moet voldoen. Het bedrijf wenst om in de toekomst een productie van (meer dan) 44 strippen per minuut te halen. Een grotere productie is zeker een voordeel, maar het moet haalbaar blijven voor de arbeiders die het moeten verpakken, tenzij het verpakkingsproces wordt geautomatiseerd.

#### **Samengevat zijn de doelstellingen dus de volgende:**

- **strippen gecontroleerd laten vallen;**
- **geautomatiseerde kwaliteitscontrole strippen (op basis van vorm, afmetingen en spievormigheid);**
- **productiecapaciteit van 40 strippen per minuut.**



*Figuur 1: een baksteenstrip*

## 1.4 Materiaal en methode

Om de masterproef succesvol te kunnen uitvoeren is er software, hardware en testapparatuur nodig. Afhankelijk van de te realiseren doelstelling worden bepaalde methodes en materialen toegepast en gebruikt.

Alvorens te starten met de effectieve programmatie van de kwaliteitscontrole, is het van belang er eerst voor te zorgen dat de strippen op de goede zijde vallen (ruwe zijde). Om dit te bereiken zal er een constructie met behulp van plaatmateriaal gebruikt worden (staal). Doordat het van staal is, zal het niet snel slijten. Plaatmateriaal is daarbij ook makkelijker om in verschillende vormen te plooien en zo een gewenste constructie te bekomen. Het is wel iets zwaarder dan bijvoorbeeld aluminium, maar staal is veel sterker en minder gevoelig aan slijtage vanwege de impact van de steenstrippen. Het onderdeel waarvoor dit materiaal dient, maakt Vandersanden NV indien mogelijk zelf. Wanneer blijkt dat dat niet mogelijk is, zal een ander bedrijf het onderdeel maken.

Voor het visiesysteem is het belangrijk eerst een keuze te maken uit de verschillende beschikbare softwarepakketten voor beeldverwerking. Met deze software zal dan een programma worden gemaakt om de stenen te kunnen controleren op zijn afmetingen, vorm en spievormigheid. Tevens dient het programma op basis van de controle de desbetreffende steenstrip dan goed- of af te keuren.

Tot slot dienen na het ontwerp van de kantelconstructie en de programmatie van de kwaliteitscontrole testen te gebeuren op een testopstelling. Deze testopstelling zal uitgerust moeten zijn met een camera, belichting... en zal moeten geconstrueerd worden zodat de werkelijke omgeving zo goed mogelijk wordt nagebootst om representatieve testresultaten te bekomen.



## 2 Vandersanden NV – Bedrijfsprofiel

Wat is Vandersanden NV en wat doen ze? Dat zijn belangrijke vragen om zicht te krijgen op de bedrijfsactiviteiten, wat van belang is voor het werken aan een (bedrijfs-)project. In hetgeen wat volgt, zal het ontstaan, de ambities en de toekomstplannen van het bedrijf beschreven worden.

### 2.1 Historiek

#### 1925: de geboorte van de steenbakkerij [1]

In Spouwen richtte Jaak Vandersanden een kleine steenbakkerij op. In die tijd werd er nog gewerkt met een veldpers en een veldoven. Deze ringoven werd in 1953 vervangen door een kameroven van Hoffman. Het drogen van de stenen gebeurde door ze gewoon onder afdakjes te leggen, dit in tegenstelling met de droogovens die nu gebruikt worden. Figuur 2 toont het baksteenproces in 1925.



*Figuur 2: baksteenproces in 1925 [1]*

#### 1962 – 1994: modernisering, uitbreiding en overnames [1]

Gedurende deze periode heeft Vandersanden NV zeker niet stil gezeten. Er werd immers een tweede steenfabriek geopend in Spouwen. Deze was ook voorzien van een tunneloven, een serieuze modernisering en verbetering aangezien men ook nu nog gebruik maakt van tunnelovens. Na het openen en ook weer sluiten van een fabriek in Veldwezelt, ontstond er een derde hypermoderne fabriek in Spouwen. Ook in Lanklaar had Vandersanden NV zijn stempel gedrukt. Twee steenfabrieken werden hier overgenomen en samen met de fabrieken in Spouwen werden ze gemechaniseerd, zoals te zien is in figuur 3 (volgende pagina). In 1994 werd ook de fabriek in Lanklaar gemoderniseerd waardoor hun capaciteit werd verdubbeld, mede dankzij twee extra ovens.



*Figuur 3: modernisering van de fabriek [1]*

### **1995-2001: De groene denkwijze [1]**

De eerste twee fabrieken in Spouwen werden vervangen door een tweede hypermoderne fabriek met een productiecapaciteit van 100 miljoen stenen per jaar. Dit was echter niet alles, want in Spouwen werd er nog een bedrijf gestart met milieubewuste technieken. Een aspect wat steeds belangrijker wordt. De technieken waarvan men gebruik maakte zijn de volgende:

- warmte-kracht-koppelingscentrale (WKK) → eigen elektriciteitsopwekking;
- rookgasfilters voor de fabrieksschouwen → zuivering van de verbrandingsgassen.

### **2001-2012: Familiebedrijf groeit uit tot Europese grootmacht in baksteenproductie [1]**

Ook in Nederland heeft Vandersanden NV naam gemaakt. Een steenfabriek (Duijs) in Hedikhuizen werd overgenomen. De overname werd een joint-venture met Huwa Baksteen. Moderne productie-eenheden zorgden hierbij voor een capaciteit van 500 miljoen gevelstenen in 2007.

In de periode 2007-2012 werd de samenwerking tussen Vandersanden NV en Huwa echter stopgezet en nam Vandersanden NV twee van de drie fabrieken in Nederland over. Er werd in deze periode echter ook een fabriek gesloten, de vestiging in Hekelgem (2011). Dit alles zorgt ervoor dat Vandersanden NV met zijn twee productielocaties in België en twee in Nederland een totale afzet van circa 60 miljoen straatbakstenen en 320 miljoen gevelstenen per jaar in een uitgebreid assortiment heeft, zoals figuur 4 aantoont. Hiermee is het dan ook het grootste baksteen producerende familiebedrijf van Europa.



*Figuur 4: hedendaags assortiment [1]*



## 2.2 Ambities en toekomstplannen

Vandersanden NV blijft gepassioneerd en hard verder werken om steeds innovatieve resultaten te bekomen. Hiervoor investeren ze in steeds betere materialen/machines. Zo zal er binnenkort een nieuwe hoekzaag op de werkvloer worden geïnstalleerd die de productie van de hoekstrippen opdrijft naar 40 stuks per minuut (basis *waalformaat*). Met de huidige hoekzaag bedraagt dit slechts vier stuks per minuut. De capaciteit zal dus maar liefst 10 keer groter worden.

Ook de strippenzaag wil men onder handen nemen door met behulp van een visiesysteem een kwaliteitscontrole te kunnen uitvoeren op de strippen, om arbeidsbesparingen en eventuele capaciteitsverhogingen mogelijk te kunnen maken. Het ontwerp van dit visiesysteem vormt dan ook, zoals eerder gezegd, de kern van deze masterproef.



### 3 Het baksteenproces: van leem tot steen

Vandersanden NV maakt verschillende soorten bakstenen. Een uitbreiding op hun assortiment zijn de baksteenstrippen (met eventueel E-Board toepassing). Deze worden echter gemaakt uit de zelfgemaakte bakstenen. De kwaliteitscontrole van deze baksteenstrippen zullen de kern van deze masterproef vormen.

Een baksteen bestaat uit verschillende grondstoffen. Een verschillend mengsel van de grondstoffen leidt logischerwijs naar verschillende soorten bakstenen. Het belangrijkste element van bakstenen is leem (ook zand kan nog in kleinere hoeveelheden worden toegevoegd). Eventueel voegt men er nog (zwarte) *shist* aan toe zodat de steen vanuit de kern beter kan bakken. Dit alles wordt eerst nog door een mechanisme geleid wat functioneert als zeef. Dit mechanisme (figuur 5) bestaat uit twee grote rollen waarbij de ene een gewone contour heeft en de andere gleuven op zijn omtrek heeft. Deze staan op een bepaalde (kleine) afstand van elkaar. Door het samenvoegsel (leem-shist) tussen de twee rollen te laten passeren, worden de grove stukken via de gleufbanen in één van de rollen uit het samenvoegsel verwijderd. Het haalt immers de grove stukken (die nog aanwezig zijn) eruit. Het wordt ook nog geleid onder een sterke magneet die het ijzer eruit haalt. Het samenvoegsel wordt vervolgens goed gemengd met toevoeging van water. Dit water is afvalwater wat men recupereert bij het zuiveren van de vormbakken. Eens het mengsel homogeen is, perst men dit door een ronde opening. Zo verkrijgt men een grote cilindrische eenheid. Hiervan worden kleine stukken afgeknipt die nodig zijn om de vormbakken te vullen. Vervolgens worden er ook nog andere stoffen (bv. mangaan) toegevoegd om later de gewenste kleur van de baksteen te bekomen.



*Figuur 5: rollenzeef*

Vervolgens wordt het bekomen mengsel in de vormbak 'geworpen'. Dit omdat men door het mengsel een bepaalde kracht mee te geven er voor kan zorgen dat het mengsel ook goed in de hoeken van de vormbakken kan geraken en de vorm van de baksteen beter wordt. Het te veel aan mengsel wordt door een snel lopende transportband van de vormbak verwijderd. Door het gebruik van een transportband wordt het afval dan ook onmiddellijk afgevoerd. Het afval recupereert men in het bedrijf bij de productie van andere bakstenen. Belangrijk is het nog op te merken dat de vormbak vooraf voorzien wordt van wat gedroogd zand. Zo kan men het mengsel gemakkelijker uit de vorm halen zonder dat er al te veel resten achterblijven.

Nu worden de 'stenen' van hun vormbak gescheiden en gaan ze gedurende twee dagen in een droger. Na deze periode worden de gedroogde vormen gestapeld (zie figuur 6) en doorheen een tunneloven geleid om het werkelijk bakproces te doorlopen. De tunneloven verwarmt de bakstenen geleidelijk tot een temperatuur rond de 1100°C. Vervolgens worden ze gekoeld door uitsluitend luchtkoeling. Dit proces moet geleidelijk aan gebeuren, anders zal de baksteen zeer snel scheuren (Quartspunt). Het geleidelijk verwarmen gebeurt door een koude lucht van de uitgang naar de ingang te dwingen. De lucht zal bijgevolg verwarmen en uiteindelijk de temperatuur laten stijgen. In de oven wordt er gebruik gemaakt van twee soorten branders: zijbranders en verticale branders. Dit om ervoor te zorgen dat alle stenen (ook de onderste) voldoende worden verwarmd. De warme lucht die eruit komt kan opnieuw gebruikt worden voor het drogen van de zand en het leem in de beginfase van het baksteenproces. Tot slot worden de afgewerkte bakstenen verpakt en gestockeerd.



***Figuur 6: stapelwijze voor het bakproces***

Er kunnen ook bakstenen met verschillende vormen op maat gemaakt worden op vraag van de klant (bijvoorbeeld ronde bakstenen). Deze worden, in tegenstelling tot de normale, handmatig gemaakt waardoor de kostprijs van deze bakstenen logischerwijs stijgt.

## 4 Strippenfacer

Het eerste deel van deze Masterproef, is zoals eerder gezegd, het gecontroleerd laten aanvoeren van de baksteenstrippen. Met 'gecontroleerd' wordt bedoeld dat alle strippen met hetzelfde vlak naar boven ligt alvorens het door het visiesysteem onder de loep wordt genomen. Het is namelijk zo dat wanneer men twee strippen zaagt uit één baksteen, elke baksteenstrip een ruwe zijde (contour van de originele baksteen) en een gezaagde kant heeft ('glad'). Wat men dus met de strippenfacer wil bereiken is dat elke baksteenstrip op de transportband valt met de ruwe zijde naar boven (naar de lens van de camera toe).

Hetgeen van zeer groot belang is, is de manier waarop de steenstrippen aankomen. De gesneden strippen worden samen met het overschot van de originele baksteen (het middenstuk) op een baksteenhouder gebracht. Deze houder bevindt zich op een aanpasbare hoogte t.o.v. van de transportband en is iets smaller dan het middenstuk van de baksteen. Hierdoor vallen de strippen als het ware (ongecontroleerd) op de transportband. In hetgeen wat volgt, worden verschillende oplossingsmogelijkheden geëvalueerd en uitgewerkt om uiteindelijk een systeem te bekomen dat zorgt voor de gewenste valbeweging

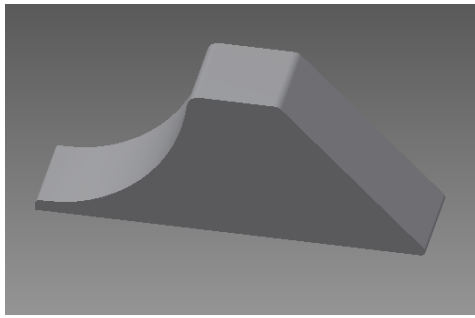
### Opmerking

**De aanvoer van steenstrippen gebeurt steeds per twee. Uit één baksteen worden namelijk steeds twee steenstrippen gesneden, uit de linker- en rechterzijde van de baksteen. In hetgeen wat volgt duidt '*linkse steenstrip*' op de steenstrip die gesneden is uit de linkerzijde van de baksteen en tevens links van de 'baksteenhouder' op de transportband valt. Dit alles is analoog voor de '*rechtse steenstrip*'.**

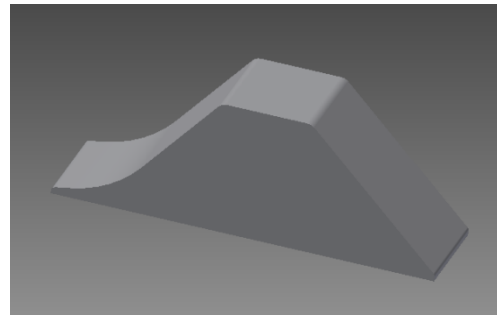
### 4.1 Concepten

In deze paragraaf worden verschillende concepten besproken op basis van een morfologisch overzicht. Dit is een methode die men in de ontwerperswereld gebruikt om zo ruim mogelijk na te denken over eventuele concepten. Nadien wordt dan een keuze gemaakt door de voor- en nadelen van elk concept tegen elkaar uit te spelen. De *beste* concepten zijn verder uitgewerkt in *Inventor* en worden in hetgeen wat volgt verder besproken.

De eerste ontwerpen (figuren 7- 9, volgende pagina) bestaan steeds uit één geheel, waardoor deze relatief eenvoudig te maken zijn en de kostprijs tevens ook laag blijft. Testen hebben echter uitgewezen dat de rechtse steenstrip steeds correct op de transportband valt, waarbij voor de linkse strip vaak het omgekeerde gebeurde. Deze ontwerpen zijn dus niet van toepassing voor het probleem, want het is cruciaal dat beide strippen correct op de transportband vallen.



***Figuur 7: strippenfacer ontwerp 1***



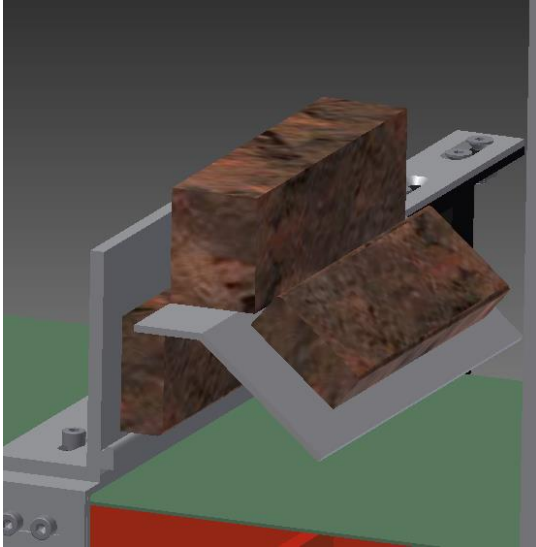
***Figuur 8: strippenfacer ontwerp 2***



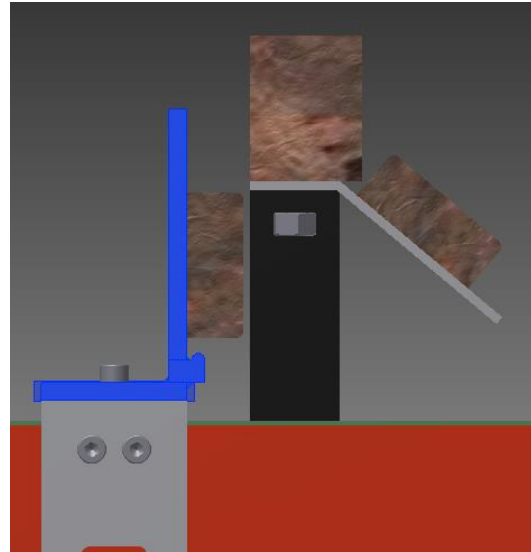
***Figuur 9: strippenfacer ontwerp 3***

Vervolgens zijn er twee andere ontwerpen die onderling veel gelijkenissen vertonen. Het eerste ontwerp hiervan is te zien op figuur 10 en figuur 11. Hierop is duidelijk te zien dat het ontwerp uit twee verschillende delen bestaat. Het rechtse gedeelte zorgt ervoor dat het middenstuk op de plaat blijft staan en dat de rechter strip correct op de transportband valt (ook vanwege de beweging van de transportband naar rechts). Het linkse gedeelte zorgt er dan weer voor dat de linkse strip eerst verticaal valt tot op een “bult” en vervolgens zal deze kantelen zoals aangegeven in figuur 11. Dit komt omdat het zwaartepunt van de strip rechts van de “bult” gelegen is en de neerwaartse kracht zo voor de gewenste kanteling zorgt ten opzichte van de “bult”.

Het linkerdeel bestaat uit twee platen waartussen een “bult” is vastgelast, zoals te zien is in figuur 11. Eerst was er in dit ontwerp echter gebruik gemaakt van een L-profiel waaraan de bult was vastgelast, maar het bedrijf *Elmech* kon de “bult” niet lassen op de gewenste hoogte vanwege de afronding die steeds aanwezig is bij een L-profiel. Het ontwerp kan tevens makkelijk bevestigd worden aan de transportband door middel van bouten (zie paragraaf 4.2), waardoor dit concept zeker een goede oplossing is voor het eerder besproken probleem (ongecontroleerde val).

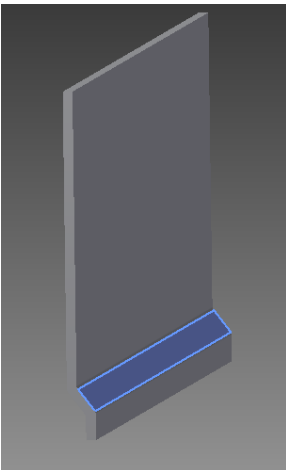


***Figuur 10: strippenfacer ontwerp 4***



***Figuur 11: detailtekening strippenfacer ontwerp 4***

De andere gelijkaardige mogelijkheid is te zien op figuur 12. De werking is identiek aan degene met de “bult”, enkel is nu de bult vervangen door een laagje rubber. Dit laagje zorgt ervoor dat de strip blijft “steken” vanwege de wrijving en op die manier ook de gewenste kantelbeweging zal maken. Er is echter gekozen voor het ontwerp met de bult, omdat testen hebben aangetoond dat dit principe minder goed werkt als het concept met de “bult”. Ook het feit dat het laagje rubber slijtage ondergaat en bijgevolg vervanging zich na x-aantal tijd opdringt, heeft een negatieve invloed gehad bij de conceptkeuze.

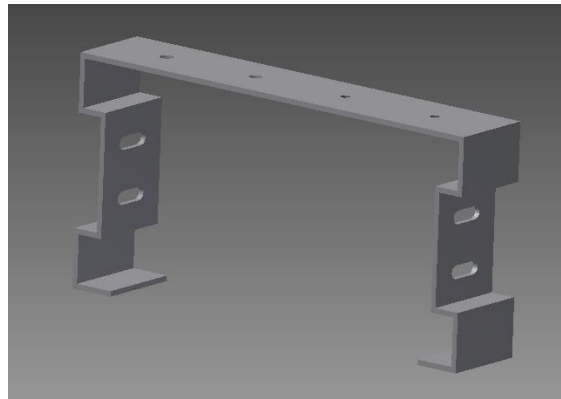


***Figuur 12: strippenfacer ontwerp 5***

## 4.2 Bevestiging

Er bestaan verschillende mogelijkheden om het gekozen ontwerp te monteren aan de transportband. De belangrijkste mogelijkheden worden in deze paragraaf besproken en uitgediept.

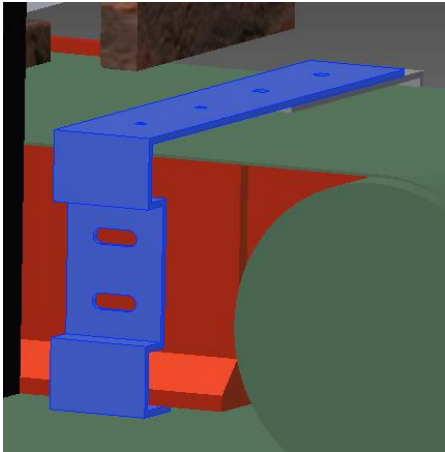
Een eerste mogelijkheid is te zien op figuur 13. Dit ontwerp bestaat uit plaatmateriaal dat gebogen wordt tot de gewenste vorm. Het ontwerp is ten eerste niet bruikbaar omdat het moeilijk op de transportband te bevestigen is. Het is enkel mogelijk om dit systeem via een uiteinde van de transportband door te schuiven tot de gewenste positie. Het monteren (en eventueel demonteren) van dit systeem vraagt dus veel tijd en deze is niet altijd beschikbaar. Ten tweede kon deze vorm door verschillende bedrijven niet geplooid worden, vanwege machinale beperkingen. Ook dit heeft ervoor gezorgd deze bevestigingsmogelijkheid te verwerpen en niet verder te gebruiken



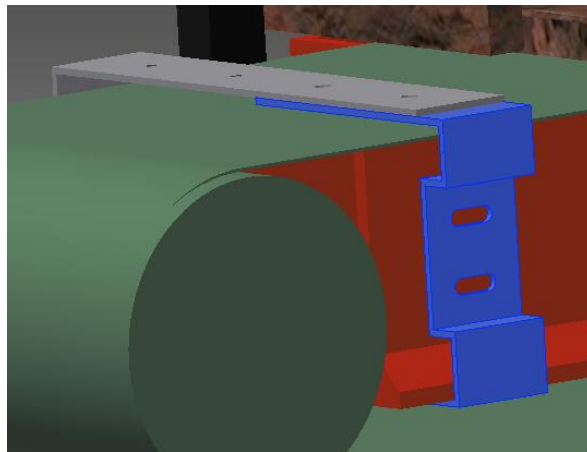
*Figuur 13: bevestigingsmogelijkheid 1*

De tweede bevestigingsmogelijkheid is een opsplitsing van de eerste en is te zien op figuur 14 en figuur 15 (volgende pagina). Hierbij is het wederom niet mogelijk om het materiaal te plooien tot de gewenste vorm. Een voordeel bij dit ontwerp ten opzichte van het vorige is echter wel dat deze eenvoudig op de transportband kan gemonteerd worden. De eerste bevestigingsmethode kon enkel op de transportband gezet worden door het model op de transportband te schuiven via een uiteinde van de transportband. Daardoor is bevestigingsmogelijkheid twee een stap in de goede richting om een goed bevestigingsprincipe te bekomen.



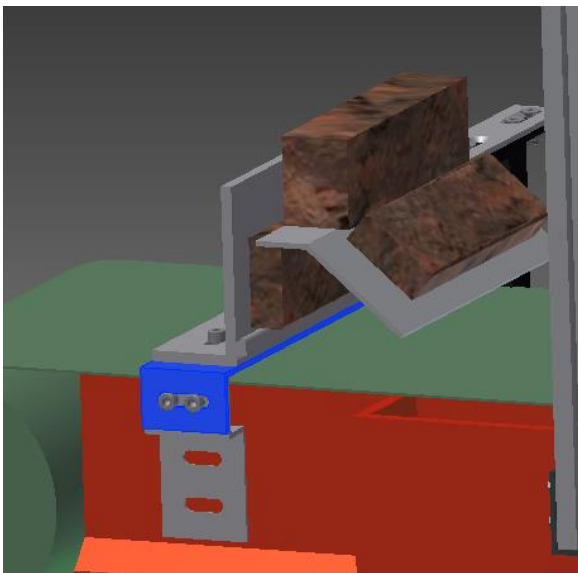


***Figuur 14: bevestigingsmogelijkheid 2***

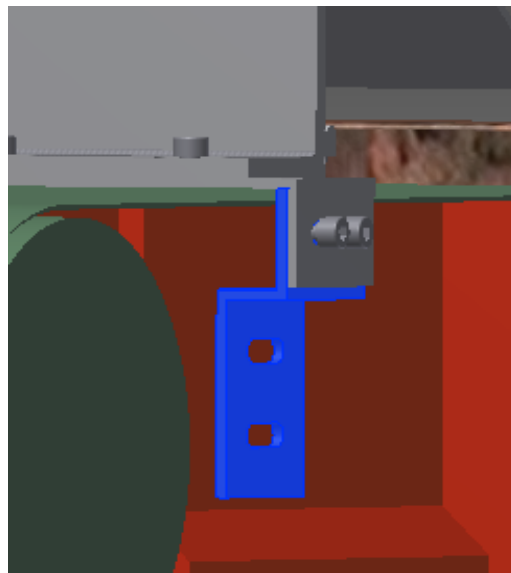


***Figuur 15: bevestigingsmogelijkheid 2***

Uiteindelijk is er een montageprincipe bedacht en uitgewerkt dat zowel effectief geplooid kan worden, als gemakkelijk te monteren is op de transportband. Figuur 16 en 17 tonen het concept van dit principe.



***Figuur 16: bevestigingsmogelijkheid 3***



***Figuur 17: detailtekening bevestigingsmogelijkheid 3***

Het “L-profiel met bult” wordt met behulp van schroeven vastgemaakt op een beugel (blauw gedeelte in figuur 16). Deze beugel dient vervolgens vastgemaakt te worden aan het tussenstuk (blauw gedeelte in figuur 17). Dit tussenstuk kan door middel van gleuven horizontaal verplaatst worden over de transportband zodat eenvoudige verstelling (bijregeling) mogelijk is bij de wisseling van bakstenen met een breedte van 50 mm naar 65 mm of andersom. Het systeem is tevens ook zo gemaakt dat er enkel bouten van *M8* nodig zijn. Enkel bij het monteren van de plaat, waar het middenstuk van de baksteen opkomt, is het nodig om verzonken bouten van het type *M8* te gebruiken.

## 4.3 Materiaalkeuze

Voor de materiaalkeuze van de strippenfacer was er keuze uit een breed gamma. Het materiaal moest echter voldoen aan een aantal voorwaarden voor gebruik in deze toepassing. Het materiaal moet immers een goede balans kennen tussen de volgende parameters:

- sterkte,
- gewicht,
- prijs.

Aan de hand van deze voorwaarden zijn er drie soorten materialen vergeleken met elkaar. Deze drie soorten zijn staal (S235), aluminium en roestvrij staal (RVS). In hetgeen wat volgt, staan de voor- en nadelen van elk materiaal vermeld en worden ze onderling met elkaar vergeleken om op die manier een gepaste keuze te kunnen maken voor zowel materiaal als leverancier.

### 4.3.1 Staal-S235

Staal is wellicht één van de meest gekende en gebruikte metalen voor plaatbewerking. Het is een legering van ijzer en koolstof en is verkrijgbaar in diverse vormen zoals buizen, kokers, profielen en platen. Staal kent een grote toepassingsgebied, enkele voorbeelden zijn: machines, voertuigen, snijgereedschap, gebouwen, etc.

Volgens *tosec* [2] staat S235 voor staal met een vloeigrens van 235 MPa. De vloeigrens bepaalt de sterkte oftewel de elasticiteit van het materiaal. Zo zal dit soort staal plastisch vervormen wanneer er een trekkracht van minstens 235 MPa op komt te staan. Deze trekkracht is voldoende sterk voor de strippenfacer, aangezien het systeem slechts een baksteenstrip van  $\pm 500$  gram moet opvangen. Standaardstaal voldoet dus aan de voorwaarde om de strip te kunnen opvangen (wat ook wel te verwachten was).

Vervolgens is het belangrijk om te weten hoe zwaar de strippenfacer zal zijn indien het wordt uitgevoerd in staal. Gegevens [3] tonen aan dat staal een soortelijke gewicht van ongeveer  $7800 \text{ kg/m}^3$  heeft (hierin zitten echter nog gradaties afhankelijk van het soort staal). Dit wil dus zeggen dat een blok waarvan de lengte, breedte en hoogte 1 meter bedraagt een massa van 7800 kg heeft. Om te weten hoeveel de strippenfacer weegt, is het noodzakelijk eerst het totale volume te berekenen. De berekeningen op de volgende pagina geven het totale volume van de strippenfacer weer door het volume van elk onderdeel apart op te tellen. Deze volumes worden bepaald aan de hand van de afmetingen (in mm) van alle betreffende onderdelen, waarvan de 2D-tekeningen terug te vinden zijn in bijlage A. Deze afmetingen zijn experimenteel bepaald aan de hand van de aanlevertijd van de strippen en de snelheid van de transportband. Dit om te vermijden dat achtereenvolgende strippen op elkaar vallen.

Bult:  $V_{Bult} = \left(16 \cdot 10 + \frac{\pi \cdot 3^2}{2}\right) \cdot 320 = 55723,89 \text{ mm}^3$

U-profiel strippenfacer:  $V_{U\text{-profiel strippenfacer}} = 450 \cdot 65 \cdot 4 = 117000 \text{ mm}^3$

Houder transportband:  $V_{Houder transportband} = 2 \cdot (150 \cdot 65 \cdot 4) = 78000 \text{ mm}^3$

Gatenplaat bij beugel:  $V_{Gatenplaat bij beugel} = 320 \cdot 65 \cdot 8 = 166400 \text{ mm}^3$

Geleidingsplaat:  $V_{Geleidingsplaat} = 320 \cdot 112 \cdot 8 = 286720 \text{ mm}^3$

Houder strippenfacer:  $V_{Houder strippenfacer} = 40 \cdot 450 \cdot 4 + 280 \cdot 60 \cdot 4 = 139200 \text{ mm}^3$

$$V_{totaal} = V_{Bult} + V_{U\text{-profiel strippenfacer}} + V_{Houder transportband} + V_{Gatenplaat bij beugel} + V_{Geleidingsplaat} + V_{Houder strippenfacer}$$

$$V_{totaal} = 55723,89 + 117000 + 78000 + 166400 + 286720 + 139200$$

$$V_{totaal} = 843043,89 \text{ mm}^3 = 843043,89 \cdot 10^{-9} \text{ m}^3$$

Het totaal volume van de strippenfacer is  $843043,89 \times 10^{-9} \text{ m}^3$ . Het gewicht wordt nu als volgt berekend:

$$\rho = \frac{m}{V} \rightarrow m = \rho \cdot V = 7800 \cdot 843043,89 \cdot 10^{-9} = 6,58 \text{ kg}$$

Wanneer er S235 als materiaal wordt gekozen, zal het totale gewicht van de strippenfacer dus 6,58kg zijn.

Ten slotte is er gekeken naar de prijs van staal [4]. Om de prijzen van de verschillende materialen te vergelijken, is er een standaard referentieplaat gebruikt van 500 x 1000 x 1 mm. Figuur 18 geeft de prijs voor de referentieplaat in staal.



**staalplaat 500 x 1000 x 1.00mm**  
 breedte x lengte x dikte, werkelijk gewicht: 4kg  
 transportgewicht: 15.00 kg  
 referentie: 104.002

€ 11,76 per plaat  
 (€ 9,72 excl. BTW)

Aantal:

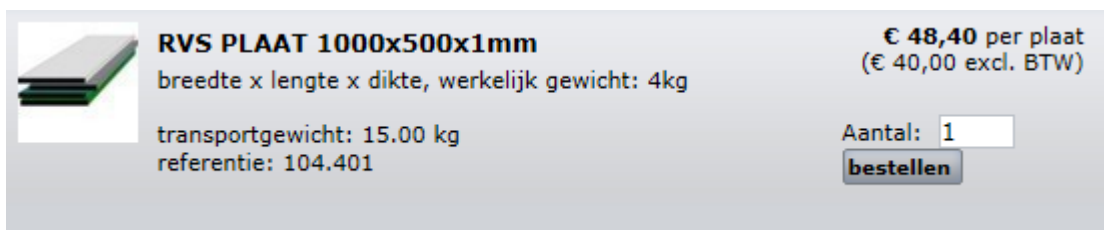
**Figuur 18: prijs referentieplaat in staal [4]**

### 4.3.2 Roestvrij staal (RVS/Inox)

Roestvrij staal (RVS) is een ijzerlegering met  $\geq 10,5\%$  chroom en  $\leq 1,2\%$  koolstof, deze legering zorgt voor een zelfherstellende oxidelaag die de corrosieweerstand verzekert [5]. Hierdoor zal het staal niet roesten wanneer het in contact komt met water.

Het belangrijkste verschil tussen een roestvrije staalsoort met gewoon koolstofstaal is de hoeveelheid chroom (Cr) in de legering. Chroom vormt een oxidelaag aan het oppervlak die het staal beschermt tegen de omgeving. Vanaf 12 gewichtsprocent chroom in het staal spreekt men van roestvast staal. Hoe meer chroom hoe hoger de weerstand tegen corrosie [5].

Aangezien het soortelijke gewicht en de sterkte (vloei-grens) van RVS in dezelfde grootteorde liggen als dat van staal (S235), voldoet ook dit materiaal aan de gewenste eisen voor de gewenste toepassing. Het verschil in prijs is echter wel significant tussen beiden. RVS namelijk ongeveer vier keer zo duur als gewoon staal (S235). Dit is het gevolg van de toegevoegde chroom die ervoor zorgt dat het staal niet roest. Figuur 19 geeft de prijs voor de referentieplaat gemaakt van roestvrij staal [6].



**RVS PLAAT 1000x500x1mm**  
breedte x lengte x dikte, werkelijk gewicht: 4kg  
transportgewicht: 15.00 kg  
referentie: 104.401

**€ 48,40** per plaat  
(€ 40,00 excl. BTW)

Aantal:   
**bestellen**

Figuur 19: prijs referentiesplaat in RVS [6]

### 4.3.3 Aluminium

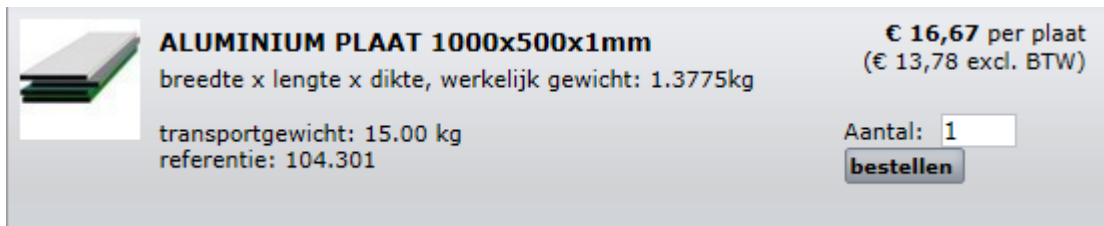
Aluminium is een materiaal dat bekend staat omwille van zijn lichtheid (gewicht) en de zilverwitte kleur. Het gewicht van aluminium bedraagt slechts een derde van dat van staal. Het is een slijtvast metaal en is bestendig tegen corrosie. Het soortelijke gewicht van aluminium [3] bedraagt  $2700 \text{ kg/m}^3$ . Het gewicht van de strippenfacer in aluminium bedraagt in dit geval dus:


$$m = \rho \cdot V = 2700 \cdot 843043,89 \cdot 10^{-9} = 2,28 \text{ kg}$$

Dit verschilt dus ongeveer met een factor drie van het gewicht van de stalen strippenfacer.

Een aluminiumlegering kent volgens *het mcbboek* [7] een treksterkte van  $400 \text{ N/mm}^2$ , deze waarde is beduidend lager dan de sterkte van staal of RVS die tussen de  $760 \text{ MPa}$  en  $860 \text{ MPa}$  bedraagt. Hierdoor zal aluminium dus veel sneller plooiën, buigen of breken. Dit wil dus ook zeggen dat er gemakkelijker bewerkingen op aluminium uitgevoerd kan worden, doordat dit soort materiaal minder sterk is.

Figuur 20 geeft de prijs weer van de referentieplaat, gemaakt uit aluminium. De prijs van aluminium [8] ligt tussen de prijs van staal (S235) en RVS. Het is iets duurder dan gewoon staal en ongeveer drie keer goedkoper dan roestvrij staal.



	<b>ALUMINIUM PLAAT 1000x500x1mm</b> breedte x lengte x dikte, werkelijk gewicht: 1.3775kg transportgewicht: 15.00 kg referentie: 104.301	<b>€ 16,67 per plaat</b> (€ 13,78 excl. BTW) Aantal: <input type="text" value="1"/> <b>bestellen</b>
---	---	---

*Figuur 20: prijs referentiesplaat in aluminium [8]*

#### 4.4 Keuzes en bestelling

Nu alle mogelijke ontwerpen, bevestigingsmogelijkheden en materiaalkeuzes bekend zijn, is het belangrijk om voor elk onderdeel de beste oplossing te kiezen en de volledige strippenfacer te realiseren. Als eerste werd er dan ook gekozen voor het vierde ontwerp van de strippenfacer, dit is degene met de bult (figuur 10). Dit model blijkt goed te werken na enkele handmatig uitgevoerde testen. Mochten de steenstrippen toch breken omdat ze vanop een hoogte van ongeveer 9 cm op het harde staal vallen, is het mogelijk een laagje rubber op de bult te implementeren zodat de botsing tegen de bult minder hard verloopt. Een tweede keuze is gevallen op de derde bevestigingsmogelijkheid (figuur 16) om de strippenfacer te kunnen monteren aan de transportband. De eenvoudige onderdelen, geconstrueerd uit plaatmateriaal, zijn immers eenvoudig met bouten vast te maken aan zowel de strippenfacer, als de transportband. Het aantal bewerkingen voor deze ontwerpen is beperkt tot het plooiën van de platen en het boren en draadtappen van enkele gaten. Als laatste is er gekozen om alle stukken in RVS te maken. RVS is een sterk materiaal dat een breed toepassingsgebied kent. Het grote voordeel van RVS is de extra chroom die ervoor zorgt dat het materiaal niet zal roesten bij contact met water. Aangezien de bakstenen nat gezaagd worden (zaagproces met toevoeging van water voor de koeling), zijn de gezaagde steenstrippen eveneens nat. Daardoor zullen ook de transportband en de strippenfacer vochtig/nat worden. In deze toepassing is RVS dus beter dan gewoon staal (S235), ondanks het feit dat RVS veel duurder is. Vervolgens was er nog de mogelijkheid om aluminium te gebruiken. Aluminium is goedkoper en minder zwaar dan RVS en is bovendien ook bestand tegen water. Toch werd in overleg met het bedrijf ervoor gekozen de strippenfacer uit te voeren in RVS.

Voor het maken van de strippenfacer zijn er twee bedrijven gecontacteerd, namelijk Elmech (Opglabbeek) en Jacometal nv. (Riemst). Beide bedrijven hebben een offerte gemaakt voor de strippenfacer. Elmech vroeg een bedrag van € 556,00 voor een uitvoering in RVS 304. Het bedrijf Jacometal kwam op een prijs van € 401,66 voor onbehandeld S235 als uitvoeringsmateriaal. In samenwerking met Vandersanden Group nv. zijn beide offertes besproken met als gevolg een bestelling bij Elmech. Ter vervollediging is de offerte van Jacometal nv. weergegeven in bijlage B. De offerte van Elmech bestond enkel uit een mail met vermelding van het totale bedrag (€556,00).

## 4.5 Implementatie

Doordat het niet meer mogelijk was om de productie van de strippen stil te leggen, is de strippenfacer (nog) niet geïmplementeerd in het proces. Er zijn wel nog enkele testen uitgevoerd met de strippenfacer (zie figuur 21) op een externe testplaats waar de omgeving zo goed mogelijk werd nagebootst. Deze testen wezen uit dat indien de (linkse) strip zeer kort langs de geleidingsplaat valt (de plaat met de 'bult'), de kantelbeweging correct gebeurt. Is de afstand tussen de strip en de geleidingsplaat echter wat groter, wordt de kans dat de strip de 'bult' schampt groter waardoor de strip niet altijd de gewenste (noodzakelijke) valbeweging maakt. Om dit probleem echter drastisch te verminderen (zonder steeds de geleidingsplaat opnieuw af te regelen), is het aangeraden een dunne lat (RVS) van 2-4 mm te lassen tegen de 'bult' zodat het contactoppervlakte met de strip groter wordt en zo de gewenste kantelbeweging toch gerealiseerd kan worden. Daarbij is het belangrijk dat de totale breedte van de 'bult' (met lat) steeds kleiner is dan de helft van de stripbreedte. Naast de valbeweging van de linkse steenstrip, diende ook de beweging van de rechtse strip getest te worden. Uit deze testen is gebleken dat deze strip wel altijd de gewenste valbeweging onderging.



*Figuur 21: proefopstelling strippenfacer*

Niet alleen de 'bult' op de geleidingsplaat dient dus gecorrigeerd te worden, ook de geleidingsplaat zelf dient een lichte aanpassing te ondergaan. Zo moet deze na implementatie tot op een bepaalde hoogte (van de bovenkant) worden korter gemaakt totdat de bovenkant ervan op een lagere hoogte gelegen is dan de afzettafel (>1 mm). Deze afzettafel is het gedeelte van de strippenfacer waarop het middenstuk van de baksteen gelegen is (rode aanduiding in figuur 21). De aanpassing dient gedaan te worden om zo contact met de grijper, die instaat voor het plaatsen van de steenstrippen (met middenstuk) op de afzettafel, te vermijden. De grijper opent namelijk na de gewenste plaatsing waardoor contact met de geleidingsplaat mogelijk is indien deze plaat te hoog is.

Tot slot moeten er ook nog enkele acties ondernomen worden met betrekking tot het monteren van de strippenfacer aan de transportband indien deze wel geïmplementeerd wordt in het proces. De transportband zelf moet lager worden gezet (10 -12 mm, afhankelijk van de afstelling) en er moeten gaten worden geboord in het frame van de transportband, zodat de strippenfacer hierop gemonteerd kan worden.

## 5 Het visiesysteem

In de huidige industrie maakt automatisering met behulp van visie een sterke opmars door. Ook Vandersanden Group nv. blijft niet bij de pakken zitten en volgt de trend van de huidige industrie. Dit uit zich in deze Masterproef met, zoals eerder gezegd, de implementatie van een visiesysteem voor de kwaliteitscontrole van de baksteenstrippen.

Een (standaard) werkend visiesysteem ontstaat door combinatie van verschillende onderdelen:

- industriële camera:
  - Afhankelijk van toepassing: resolutie, aantal beelden per seconden (framerate);
- lens: belangrijk voor het nodige gebied te kunnen waarnemen met de camera;
- belichting: zorgt voor een correcte weergave van het beeld (bv. helderheid);
- sensoren;
- software voor beeldverwerking;
- opstelling: frame om het systeem te kunnen bevestigen;
- actuatoren: in dit geval het uitwerpsysteem dat reageert op de output van het visiesysteem (optioneel, wordt pas in de toekomst aangebracht door Vandersanden NV zelf)

In hetgeen wat volgt komt de opzet van het visiesysteem aanbod samen met een diepgaandere bespreking van de verschillende systeemonderdelen.

### 5.1 Opzet van het visiesysteem

Met het visiesysteem wil men een automatisch controle van de baksteenstrippen bekomen. Een automatische controle is sneller maar ook accurater dan een manuele controle. Een werknemer kan namelijk af en toe een slechte strip over het hoofd zien en verpakken. Het kan ook zo zijn dat een werknemer een goede strip toch afkeurt wat de productie vermindert en dus economisch minder goed is voor het bedrijf.

Aangezien een visiesysteem sneller kan werken dan een manuele controle, kan ook de productiecapaciteit verhoogd worden. Hier moet echter rekening worden gehouden met de cyclustijden en het ontwerp van de strippenfacer (zie paragraaf 4.5) wat de productiesnelheid begrenst.

Het visiesysteem dient zoals eerder gezegd voor de automatische kwaliteitscontrole. Kwaliteit is echter een ruim begrip. Wat zorgt er namelijk voor dat men kan zeggen of een desbetreffende steenstrip kwaliteitsvol is of niet? Een werknemer kan immers vaak door middel van ervaring (verleden) de juiste keuze maken op het gebied van kwaliteitscontrole, maar een visiesysteem heeft geen 'ervaring' en zal bijgevolg geen rekening houden met eerder gecontroleerde strippen.

Deze eigenschap zorgt ervoor dat er strikte instructies moeten geprogrammeerd worden waarop het visiesysteem een juiste beslissing kan maken inzake de kwaliteitscontrole. De zaken waarop gecontroleerd moeten worden, zijn:

- Vorm (contour en hoeken),
- Afmetingen (lengte, breedte en hoogte),
- Spievormigheid.

Voldoen de steenstrippen niet aan deze opgegeven (geprogrammeerde) eisen, worden ze bestempeld als 'slechte' strippen en moeten ze bijgevolg verwijderd worden.

## 5.2 Onderdelen van het visiesysteem

### 5.2.1 Camerakeuze

De camera vormt het hart van het visiesysteem. Dit orgaan zorgt namelijk voor de beelden om de beeldverwerking op uit te laten voeren en daardoor een gepaste kwaliteitscontrole toelaat. Er zijn verschillende soorten (en types) camera's op de markt, elk met zijn eigen karakteristieken. Het is dus van belang om een camera te kiezen die toepasbaar is voor het beoogde doel (in een bepaalde omgeving), in dit geval de controle op steenstrippen. Voor deze toepassing zijn de belangrijkste eigenschappen van de camera waarmee rekening werd gehouden:

- de resolutie;
- de framerate (aantal beelden per seconden);
- de sluitertijd.

De **resolutie** [9] is een maat voor het detail in het beeld, het scheidend vermogen. Het is namelijk de minimale detecteerbare afstanden tussen twee lijnenparen in het beeld. Aangezien dit dus invloed heeft op de latere nauwkeurigheid van het visiesysteem, dient deze camera te voldoen aan een bepaalde graad van resolutiegrootte. Het bedrijf streeft naar een minimale nauwkeurigheid van  $\pm 1$  mm voor de afmetingen van de baksteenstrip (lengte en breedte). Om zeker aan deze nauwkeurigheid te voldoen is het van belang een camera te kiezen met een voldoende hoge resolutie, wenselijk hoger dan 640 x 480.

De **framerate** [9] van de camera is een maat voor het aantal beelden dat de camera kan nemen per seconde. Hoe hoger de framerate van de camera is, hoe meer beelden er per seconden kunnen genomen worden en dus ook hoe sneller de te controleren objecten de camera mogen passeren. Het is dus wel duidelijk dat dit een belangrijke parameter is voor de keuze van de camera. Aangezien de transportband op dit moment werkt op een snelheid van ongeveer 7 cm/s (0,07 m/s), is het echter niet nodig een camera te kiezen met een zeer hoge framerate. Voor deze masterproef is op basis van deze transportbandsnelheid dan ook geopteerd voor een framerate van 10 - 25 beelden per seconden.

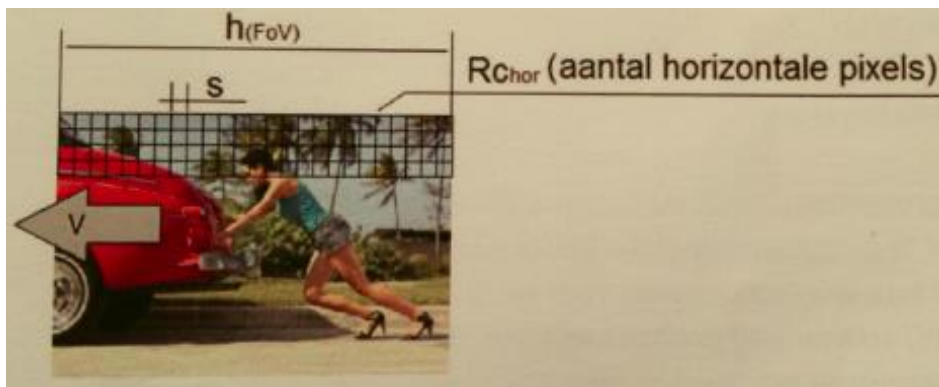
De **sluitertijd** [10] is een maat voor de opname van licht door de camera en bijgevolg dus ook voor de helderheid van het beeld. Een te hoge sluitertijd kan echter zorgen voor overbelichting en bijgevolg voor verzadiging van pixels. Dit geeft als effect een witte pixel. De sluitertijd is eveneens gelinkt aan *bewegingsonscherpte* van het beeld (motion blur). Wanneer een object op een bepaalde snelheid de camera passeert, kan hierdoor een onscherp (wazig) beeld ontstaan. Dit ontstaat wanneer een punt van het object de tijd heeft om licht 'uit te zenden' naar 2



verschillende pixels binnen een bepaalde tijd (sluiter tijd). De berekening van de maximale sluitertijd, om te zorgen dat het betreffende punt van het object slechts licht kan uitstralen naar één pixel, gebeurt op basis van vergelijking (1):

$$t [s] = \frac{s [m]}{v [\frac{m}{s}]} \text{ met } s [m] = \frac{h [m]}{R_{c,hor}} \quad (1)$$

In vergelijking (1) is  $v$  de snelheid waarmee het object de camera passeert (snelheid van de transportband). De horizontale afmeting (breedte) van de *field of view* (beeldbereik van de camera) is vertegenwoordigd door  $h$ . Het aantal pixels in horizontale richting waaruit de *field of view* bestaat, is weergegeven door  $R_{c,hor}$  (afhankelijk van resolutie). Ter verduidelijking toont figuur 22 dit alles nog eens.



**Figuur 22: factoren voor berekening sluitertijd [10]**

Indien de vergelijking nu wordt toegepast voor een camera met een resolutie van 2048 x 1536, is de maximaal toegelaten sluitertijd:

$$t [s] = \frac{s [m]}{v [\frac{m}{s}]} = \frac{\frac{h [m]}{R_{c,hor}}}{v [\frac{m}{s}]} = \frac{\frac{0,30 [m]}{2048 \text{ pixels}}}{0,07 [\frac{m}{s}]} = 2,09 \cdot 10^{-3} [s] \rightarrow 2,09 [ms]$$

**Opmerking:** de herkomst van de waarde voor de *field of view* is uitgelegd in paragraaf 5.2.2.

De maximale sluitertijd om voor dit systeem geen wazig beeld te krijgen, bedraagt dus 2,09 ms. De gekozen camera moet dus de capaciteit hebben om een lagere sluitertijd in te stellen dan de berekende maximale tijd.

## 5.2.2 Lens

De lens die wordt aangebracht op de camera, bepaalt mee de grootte van het gebied (*field of view*) dat kan worden opgenomen door de camera. Het is echter van belang dat de gehele oppervlakte van de steenstrip in dit beeld vervat zit om een doeltreffende kwaliteitscontrole te kunnen doen. Vanwege die reden dient de *field of view* minstens even groot te zijn als een steenstrip (nominale afmetingen).

De te controleren steenstrippen hebben een nominale lengte en breedte van 210 mm respectievelijk 50 mm (soms 60 mm), toch dient de *field of view* groter gekozen te worden. Elke steenstrip kan immers wat verschoven op de transportband liggen ten opzichte van elkaar vanwege de valbeweging op de strippenfacer. Rekening houdend met deze verschuivingen, is een *field of view* met een breedte ( $W$ ) en een hoogte ( $H$ ) van 29,8 mm respectievelijk 22,3 mm een goede oplossing.

Om nu een juiste keuze te kunnen maken voor de te gebruiken lens, zijn er ook nog twee andere parameters dan de *field of view* nodig:

- chipgrootte van de gebruikte camera,
- positie van de camera t.o.v. de transportband (werkafstand).

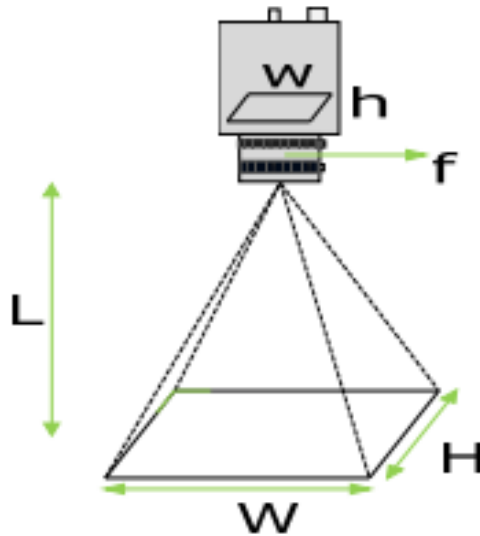
Kennis van deze parameters en het gebruik van vergelijking (2), kan de focale lengte (brandpuntsafstand) van de lens bepaald worden.

$$\frac{f}{L} = \frac{w}{W} = \frac{h}{H} \quad (2)$$

De variabelen in vergelijking (2) hebben de volgende betekenis:

- $f$ : focale lengte van de lens,
- $L$ : werkafstand,
- $w$ : chipbreedte,
- $h$ : chiphoogte,
- $W$ : breedte *field of view*,
- $H$ : hoogte *field of view*.

Aangezien eerst een camera wordt gekozen alvorens de lens te kiezen, zijn de chipbreedte en -hoogte gekende variabelen samen met de *field of view*. Enkel de focale lengte en de werkafstand blijven dus onbekend. De werkafstand is afhankelijk van de beschikbare ruimte om de camera te bevestigen. Eens de bevestiging is gebeurd, is ook deze variabele gekend en blijft enkel de focale lengte van de lens over als onbekende variabele. Oplossen van vergelijking (2) naar  $f$  geeft echter een oplossing van de focale lengte voor de te gebruiken lens. Op basis daarvan en rekening houdend met gepaste bevestigingsmogelijkheden op de camera, kan bijgevolg een geschikte lens gekozen worden. Ter verduidelijking is vergelijking (2) met aanduiding van de verschillende variabelen grafisch weergegeven in figuur 23 [11].



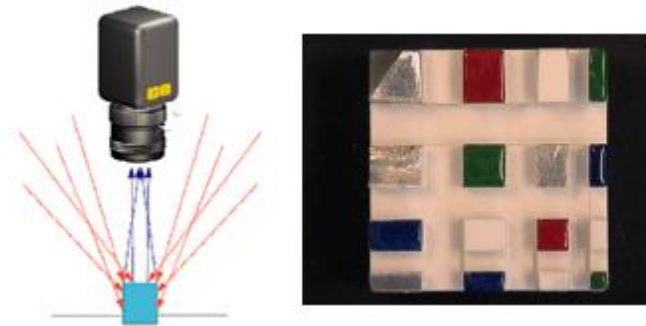
Figuur 23: variabelen bij de lens [11]

### 5.2.3 Belichting

De belichting is één van de belangrijkste organen van een visiesysteem. Toch wordt er vaak te weinig aandacht geschonken aan de keuze van dit orgaan wat tot complicaties in de beeldverwerking kan betekenen. Een minder goede belichting kan immers voor hinderende schaduwvorming zorgen. Deze schaduw moet dan met behulp van een beeldverwerkingssoftware (zie paragraaf 6.1) weggewerkt worden. Dit kan leiden tot zeer complexe programmering, minder nauwkeurige controles, grotere verwerkingstijden... welke niet gewenst zijn in een visiesysteem. Het gepaste belichtingstype kan deze problemen echter verhelpen en op die manier een grote meerwaarde bieden aan het totale visiesysteem. Een opsomming van enkele types geeft:

- diffuse belichting,
- ringbelichting,
- directionele strijkelichting,
- gepolariseerd licht.

**Diffuse belichting** [12] is gebaseerd op -zoals de benaming doet vermoeden- verspreid licht. Het licht komt dus van alle kanten waardoor het als eigenschap heeft minimale schaduwvorming en reflecties te vertonen. Een nadeel is echter dat er minder onderscheid mogelijk is in oppervlaktekenmerken vanwege dit type belichting. Ter verduidelijking is in figuur 24 een voorbeeld van diffuse belichting weergegeven.



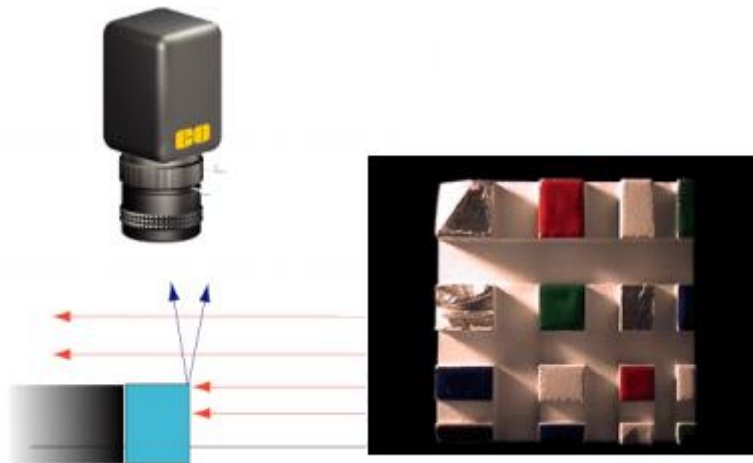
*Figuur 24: voorbeeld diffuse belichting [12]*

**Ringbelichting** [12] heeft eveneens als voordeel weinig schaduwvorming te geven. Daarnaast zorgt het ook voor een gelijkmatige belichting zodanig dat de belichte gebieden steeds evenveel licht ontvangen (gelijkmatig). Het nadeel van dit type belichting is het ontstaan van eventuele ringvormige glans (bij vlakke, glanzende oppervlakten). De montage gebeurt rechtstreeks op de lens, waardoor dure montagesstukken/-hulpstukken nodig zijn. Dit alles zorgt er dus voor dat dit een duurdere vorm van belichting is. In figuur 25 is een voorbeeld van deze vorm van belichting weergegeven.



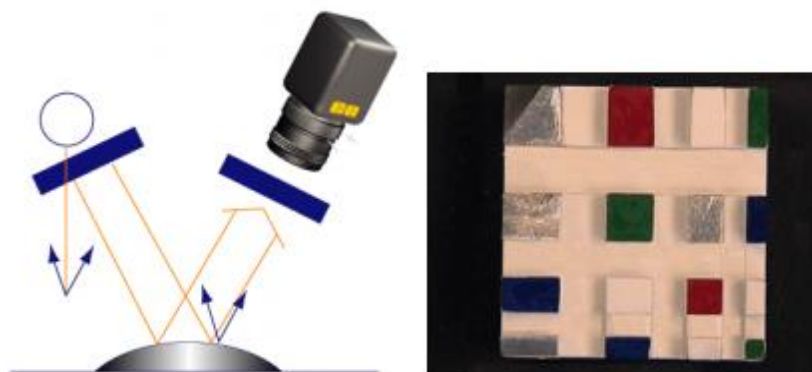
*Figuur 25: voorbeeld ringbelichting [12]*

Met behulp van **directionele strijkelichting** [12] kan men gemakkelijk oppervlaktedefecten ontdekken en de structuur van het oppervlak controleren. Het belangrijkste nadeel van dit type is echter de zware schaduwvorming die er ontstaat. Dit nadeel kan men mogelijk wel ombuigen naar een voordeel. Het is namelijk zo dat wanneer er zware schaduwvorming op een specifieke plaats optreedt, er in de buurt een ongelijkheid (bv. een bult of een gat) in het oppervlak aanwezig is. Op deze manier kan men dus een voorspelling doen van het oppervlak en op basis hiervan een soort kwaliteitscontrole uitvoeren. Een voorbeeld van directionele strijkelichting is weergegeven in figuur 26.



*Figuur 26: voorbeeld directionele strijkelichting [12]*

**Gepolariseerd licht** [12] zorgt voor een gelijkmatige belichting, zonder glans. Dit is mogelijk gemaakt door het gebruik van (polarisatie-) filters. Het nadeel wat hierdoor ontstaat is dat er een lagere helderheid van het beeld wordt bekomen vanwege het gebruik van deze filters. Het is dus van belang om voor elke toepassing een afweging te maken in verband met hoeveel licht het visiesysteem nodig heeft om optimaal te werken. Een voorbeeld in het gebruik van gepolariseerd licht is weergegeven in figuur 27.



*Figuur 27: voorbeeld gepolariseerd licht [12]*

De besproken types zijn slechts enkele types van de verschillende soorten belichtingen. Het zijn echter die types waarmee tijdens deze masterproef rekening is gehouden om een geschikte keuze te maken voor de desbetreffende toepassing. De keuze is dan ook gevallen op een gepolariseerde belichting. Testen hebben immers bevestigd dat dit type van belichting voldoende presteerde voor het beoogde doel, namelijk een kwaliteitscontrole voor baksteenstrippen. Deze gaf als resultaat eveneens een minimaal schaduwvorming wat belangrijk is om de te controleren steenstrip uit het beeld te halen. De belangrijkste reden voor deze keuze is echter het feit dat reflecties op de natte

transportband (vanwege het invallende licht) door de filter(s) worden tegengehouden. Deze zouden immers ongewenste *storingen* in het beeldresultaat kunnen veroorzaken.

De redenen om niet verder te gaan met de andere besproken belichtingsmethoden, zijn de volgende:

- diffuse belichting → te veel reflectie op natte achtergrond (transportband is nat);
- ringbelichting → duur en aangepaste lens nodig om de belichting hierop te monteren;
- directionele strijkelichting → te veel schaduwvorming voor de betreffende toepassing.

## 5.2.4 Lijnlaser

De lijnlaser is een laser dat geen puntbron is, maar een licht voortbrengt in de vorm van een lijn. Aan de hand van deze lijn kan dan door middel van verschillende soorten technieken een hoogtemeting gedaan worden. Een ruimschootse uitleg van deze technieken is gegeven in paragraaf 5.4.

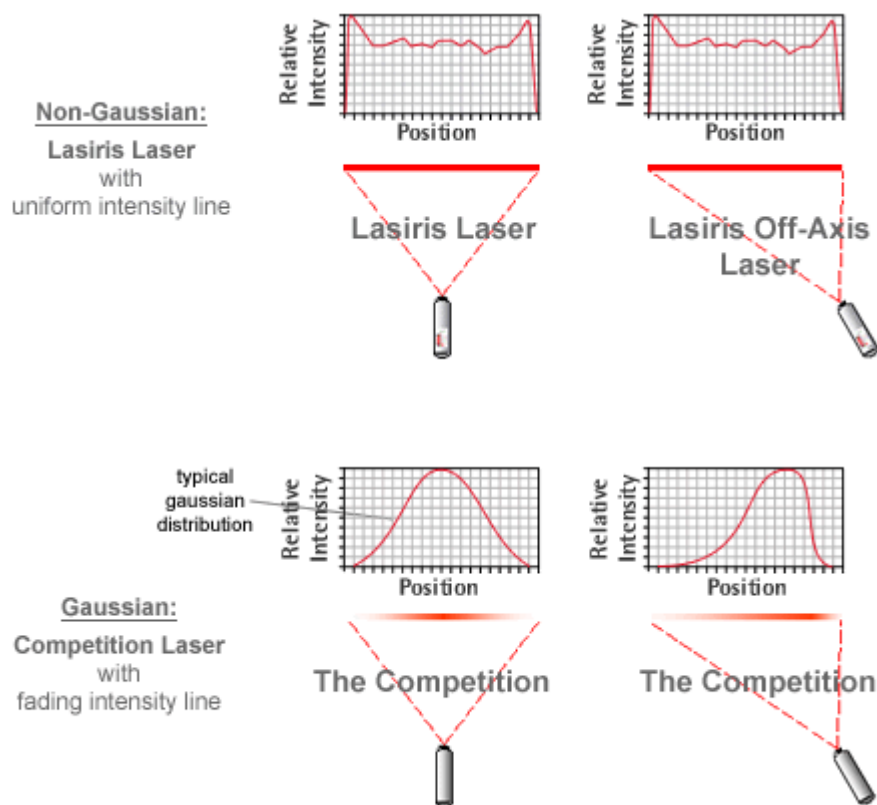
De keuze van lijnlaser bepaalt ook mee de haalbare nauwkeurigheid van de hoogtemeting. Het is immers van belang een scherpe visualisatie van de laserlijn in het beeld te krijgen om daarop dan technieken op hoogtemeting toe te kunnen passen. Met andere woorden moet de laser dus een bepaalde densiteit hebben, de weergave van een dunne scherpe laserlijn in plaats van een dikke zachte lijn.

In deze masterproef is de nauwkeurigheid van de hoogtemeting niet zo van belang. Men wil op basis van de hoogtemeting vooral een uitspraak kunnen doen over de spievormigheid van de strip. Hiervoor is namelijk geen exacte waarde van de hoogte nodig en dus ook geen grote nauwkeurigheid. Er is namelijk een nauwkeurigheid gewenst van twee millimeter op een nominale striphoogte van twintig (of dertig) millimeter. Tijdens de testfase (zie paragraaf 8.1.4) is er dan ook slechts gebruik gemaakt van een goedkope en eenvoudige lijnlaser, waarvan zijn type en andere gegevens niet zijn teruggevonden.

Voor de werkelijke implementatie van het visiesysteem op de strippenzaag, is er de optie om te kiezen voor een andere lijnlaser met betere eigenschappen. Voorbeelden van zulke lasers zijn (datasheets zijn weergegeven in bijlage C, D en E:

- *Z-laser* → ZM18B [13],
- *Z-laser* → ZM12B [14],
- *Z-laser* → ZD [15].

Uit deze drie voorbeelden valt de keuze voor deze specifieke toepassing op het type *ZM18B*. Deze laser genereert namelijk een felle lijn (tot 120 mW) dan de andere lasers (zie bijlage C). Tevens kan deze laser naast een *Gaussiaanse* lijn ook een homogene lijn genereren. Het is namelijk belangrijk dat de gegenereerde lijn ongeveer overal een even grote lichtintensiteit heeft. Bij een Gaussiaanse lijn is de intensiteit immers het grootst in het midden van de lijn en verminderd deze naarmate men verder van het midden afwijkt. Figuur 28 (volgende pagina) toont dit fenomeen ter verduidelijking.



*Figuur 28: verschil tussen homogene en Gaussiaanse lijnlaser [16]*

Het type *ZM12B* kan ook een homogene lijn genereren maar aangezien zijn lichtintensiteit lager ligt dan die van *ZM18B*, is het aangeraden de *ZM18B* te verkiezen boven de *ZM12B* omwille van de eerder vernoemde eigenschappen (intensiteit en lijntype).

### 5.2.5 Het frame

Een goed frame is van belang om een goede stabiliteit van het visiesysteem te garanderen. Op het frame wordt namelijk de camera bevestigd samen met de belichting en de laser (zie verder). Onstabielheid zoals trillingen kunnen het beeld sterk verzwakken, wat niet gebruikelijk is om een correcte kwaliteitscontrole te kunnen uitvoeren.

We hebben ook voor een universeel frame gekozen. Hiermee bedoelen we dat het frame zowel voor de tests als voor de het effectief in bedrijf stellen bruikbaar is. Zo moet men het kader slechts optillen en verplaatsen om een installatie van het kader, de camera... te bekomen (mits eventuele bijstellingen van de camera en dergelijke).

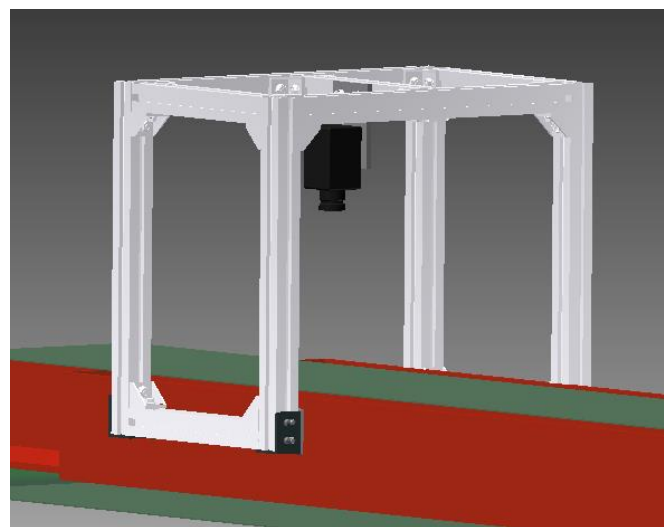
Het frame is geconstrueerd met aluminiumprofielen van Bosch-Rexroth. Deze hebben een doorsnede van 40x40 (mm) en een T-sleuf met een breedte van 10 mm. Deze profielen worden vaak gebruikt in toepassingen met visiesystemen. De profielen zijn immers relatief licht vanwege het gebruik van aluminium en de montage tot een bepaalde opstelling is vrij eenvoudig

met de bijbehorende T-moeren en bouten. Het resultaat van het frame is weergegeven in figuur 29 (ontwerp in *Autodesk Inventor*). Een totaal omhullende afscherming voor het omgevingslicht ontbreekt hier echter.



***Figuur 29: frame camera***

Dit frame kan met behulp van 'voetjes' die op de transportband worden bevestigd, geïnstalleerd worden op de transportband. Aangezien men werkt met gleuven en T-moeren, is het eenvoudig om eventueel nodige verstellingen te kunnen verrichten zodanig dat alles goed staat uitgelijnd. Dit geldt ook voor de camera. Het is namelijk van belang om de camera correct boven de transportband op te stellen. Hierbij is de hoogte ten opzichte van de transportband belangrijk, maar ook het feit dat de camera zo staat opgesteld dat de steenstrippen mooi in het beeld van de camera komen. Aangezien het frame en de andere onderdelen die de camera op het frame houden voorzien zijn van sleuven, is het eenvoudig om ervoor te zorgen dat aan al deze eisen wordt voldaan. De opstelling van het frame aan de transportband is weergegeven in figuur 30.



***Figuur 30: opstelling camera***



### 5.3 Uitwerpsysteem (actuator)

Het uitwerpsysteem dient voor de verwijdering van de afgekeurde steenstrippen. Dit proces kan op vele manieren gebeuren, maar eenvoudige en goedkope oplossing wordt bekomen door gebruik te maken van een pneumatische cilinder (ook andere actuatoren zijn mogelijk zoals een markeerspuit om de foute stenen aan te duiden). De keuze voor een pneumatische oplossing is bewust en wel om de volgende redenen:

- Perslucht wordt zeer vaak toegepast in het bedrijf;
- Zuiver;
- Beperkte onderhoud;
- Voldoende drukopbouw voor deze toepassing (beperkte kracht nodig)
- Goedkoper dan hydraulische en elektrische cilinders.

De cilinder moet aangestuurd worden indien er een afgekeurde strip gedetecteerd wordt. Dit gebeurt door middel van een PLC (Programmable Logic Controller). De PLC ontvangt immers een digitaal signaal (Digitale Input) van het visiesysteem indien er een steenstrip wordt afgekeurd. Bijgevolg reageert de PLC hierop door zelf een digitaal signaal te sturen (Digitale Output) naar de cilinder. De cilinder zal hierdoor uitschuiven en een impuls geven aan de afgekeurde steenstrip waardoor deze laatste van de transportband valt en wordt afgevoerd.

Hetgeen wat voorgaand is uitgelegd, is enkel het principe van het gewenste systeem. In de realiteit wordt er rekening gehouden met wanneer de cilinder precies actief moet worden. Dit aangezien de cilinder zich verder op de transportband bevindt dan het visiesysteem. Om dit probleem op te vangen, zal er dan ook gebruik moeten gemaakt worden van een zogenaamde *encoder*, die pulsen genereert naargelang de as van de transportband verdraait. Door de afstand van de cilinder t.o.v. het visiesysteem uit te drukken in het aantal pulsen, kan de PLC op het juist moment zijn digitaal signaal naar de cilinder doorsturen. Op deze manier bekomt men een strippenafvoer met enkel kwaliteitsvolle strippen op het gebied van vorm, afmetingen en spievormigheid.

Dit principe van uitwerpsysteem is echter zo voor het bedrijf gekozen, zodanig dat ze dit later zelf kunnen implementeren. Het is namelijk zo dat dit niet tot het doel van de masterproef behoort, maar eerder ligt in de lijn van de veranderingen die het bedrijf pas in de toekomst wil realiseren om een volledig automatische machine te bekomen. Voorlopig zal men nog blijven gebruik maken van een extra werkmans voor de verwijdering van de steenstrippen en is bijgevolg de implementatie van dit uitwerpsysteem geen echt doel voor de masterproef.

### 5.4 Hoogtemeting

Het is van belang dat de gezaagde steenstrippen geen spievorm krijgen. Dit zorgt namelijk voor een moeilijkere plaatsing van de strippen bij het bouwen van het huis. Ook voor het visuele aspect is dit nadelig (kwaliteitsverlies). Om de klant tevreden te houden is het dus van belang dat de ingestelde zaaghoogte van twee millimeter ook effectief zorgt voor strippen met een dikte

van twee millimeter. Lichte afwijkingen op de ingestelde hoogte zijn toegelaten aangezien het zeer ruwe karakter van de steenstrip ook voor kleine verschillen kan zorgen. De tolerantie waarbinnen het hoogteverschil moet blijven liggen, bedraagt twee millimeter.

Om spievorming te kunnen tegengaan, wordt de hoogte van de strip in elk opgenomen beeld van de camera gemeten. Dit wil dus zeggen dat als de camera een framerate heeft van 25 beelden per seconden, er elke seconden 25 hoogtemetingen gebeuren op de strip in kwestie. Aangezien de steenstrip op een transportband ligt, zal deze met een bepaalde snelheid voorbij de camera gaan waardoor elke meting op een andere plaats op de strip gebeurt.

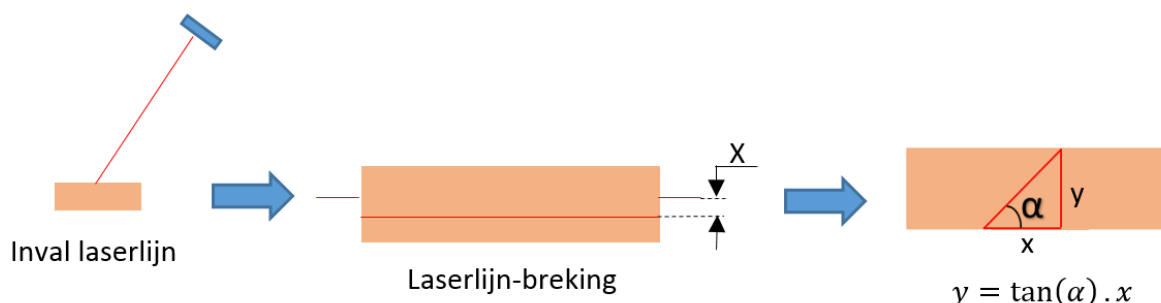
Hoogtemetingen kunnen op verschillende manieren ontstaan. In hetgeen wat volgt bespreken we drie methodes, waarvan we slechts één methode effectief zullen gebruiken: hoogtemeting met behulp van laser-triangulatie. Dit vanwege de relatief lage meerkost van de laser t.o.v. een smartcamera.

### 5.4.1 Lasertriangulatie

Bij deze methode gaan we de hoogte van de strip (bij elk opgenomen beeld van de camera) controleren aan de hand van de breking van een laserlijn. Een vereenvoudigde voorstelling van deze methode is weergegeven in figuur 31.

De basis van deze methode is de schuine opstelling van een laserlijn zodanig dat de laser schuin invalt op de steenstrip. Aangezien nu de laserlijn het eerst de steenstrip tegenkomt op zijn weg en pas later de transportband, ontstaat er een wegverschil tussen de laserlijn op de strip en de transportband. Door dit wegverschil ( $x$ ) kan men via berekeningen (bv. Stelling van Pythagoras, sinus/cosinus/tangens) de hoogte van de strip ( $y$ ) op die plaats meten. Kort samengevat gaat het visiesysteem de laserlijnen analyseren en op basis van die analyse wordt er dan een berekening gemaakt voor de hoogte. Op het resultaat van deze berekening kunnen we dan besluiten of de strip op het gebied van stripvorming wordt afgekeurd of goedgekeurd.

Het is zelfs zo dat indien we deze methode verder zouden verfijnen, we in principe een 3D-beeld kunnen construeren. Vandaar dat men deze methode ook vaak 3D-lasertriangulatie noemt.



**Figuur 31: lasertriangulatie**

## 5.4.2 Hoogtemeting a.d.h.v. een smartcamera

Bij deze methode wordt er gebruik gemaakt van een 3D-smartcamera (bv. Gocator van LMI [17], figuur 32). Die zijn uitgerust met zowel een laser als een camera en meet automatisch de verschillende hoogtes van de strip op. Deze hoogtes worden dan in de vorm van een array (lijst) teruggekoppeld naar de gebruiker.

Deze camera's zijn ook voorzien van een interface waardoor stand-alone werking mogelijk is. Dit wil zeggen dat er geen externe PC nodig is om de visie uit te voeren. De bewerkingen die je in deze interface kan doen zijn beperkter dan wanneer je bijvoorbeeld aan beeldverwerking zou doen met HALCON.

Het principe van deze camera's komt overeen met de hierboven besproken 3D-lasertriangulatie. Hier zit echter alles omvat in één apparaat en het aspect van belichting is niet noodzakelijk aangezien je alles analyseert via de laser. Zo kunnen ook de ruwe oppervlakten ontdekt worden in de steenstrippen door de gegevensstroom aan hoogtemetingen te gaan analyseren. Scheurtjes zijn moeilijker waarneembaar waardoor er weer een smartcamera moet gekozen worden met een hogere resolutie. Dit komt echter de kostprijs niet ten goede. Vanwege deze hoge kostprijs (kostprijs goedkope versie:  $\pm$  €4000) is dan ook in overleg met Vandersanden NV overeengekomen dit principe niet toe te passen in de Masterproef.

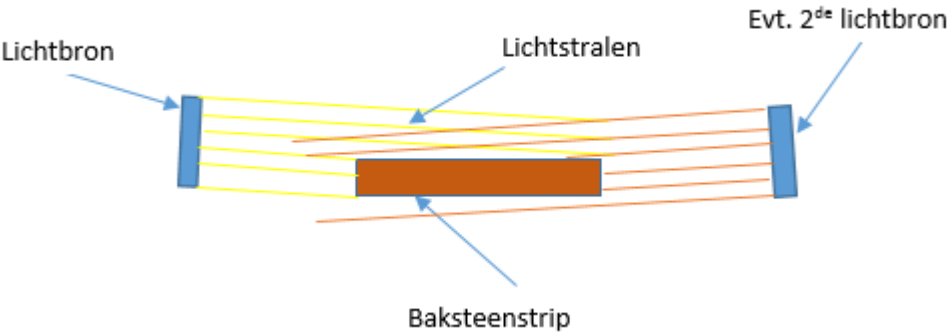


*Figuur 32: Gocator van LMI [17]*

## 5.4.3 Hoogtemeting a.d.h.v. belichtingscontrole (schaduwvorming)

Door de strijkbelijting, zoals weergegeven in figuur 33 afwisselend te gebruiken, worden de ruwheden op de bakstenen langs twee kanten belicht. Deze belichting zorgt ervoor dat de 'kuilen/putten', maar ook de uitsteeksels in de strip meer schaduwvorming veroorzaken. Deze schaduw kan men analyseren door te werken met de grijswaarden van de beelden. Op deze manier kan men een soort van pass/fail-systeem toepassen. Het moeilijke aan deze manier is dat de belichting accuraat moet worden afgeregeld en dat de tijdstippen voor het omschakelen van het licht zeer nauwkeurig moet plaatsvinden. Het feit dat de belichting steeds aan/uit gaat, is ook geen ideaal gegeven, hoewel dit probleem kan beperkt worden door gebruik te maken van LED-verlichting. Deze manier is dus niet nauwkeurig en doeltreffend genoeg voor toegepast te worden op deze masterproef aangezien ook het omgevingslicht nog een rol van betekenis kan spelen (dit kan echter opgelost worden door een afscherming). Ook het probleem met het meten van de hoogte is op deze manier niet opgelost waardoor nog bijkomende maatregelen dienen

getroffen te worden. Deze oplossingsmethode zal dus niet worden toegepast in de verdere uitwerking van de masterproef.



*Figuur 33: toepassing strijkbelichting*

## 6 Beeldverwerking

De beoogde kwaliteitscontrole wordt zoals eerder gezegd gerealiseerd door middel van een visiesysteem. In dit systeem is de beeldverwerking een belangrijk (zometer het belangrijkste) onderdeel. In dit hoofdstuk wordt dan ook een uiteenzetting gegeven over het gebruikte softwarepakket (vergelijking met andere softwarepakketten), enkele gebruikte functies/coderingen, simulatieresultaten en de testresultaten van de real-timeapplicatie. Hierbij is het dan ook van belang te vermelden dat we niet alle gebruikte programma-instructies zullen uitleggen, maar alleen diegene die van belang zijn om de programmaonderdelen uit te leggen.

### 6.1 Keuze van het softwarepakket

De beeldverwerking gebeurt door middel van een softwarepakket. Aangezien het bedrijf al eerder gebruik gemaakt heeft van de pakketten *COGNEX* en *HALCON* en deze ook wereldwijd bekend zijn als vooraanstaande softwarepakketten voor beeldverwerking, is er een vergelijking gemaakt tussen beiden (tabel 1). Op basis van de pro's en contra's, is de keuze dan ook gevallen op *HALCON*. Vooral het verkrijgen van een ontwikkelaarslicentie en de ter beschikking gestelde opleiding hebben positief bijgedragen aan de keuze voor *HALCON*. De procedures en technieken die verder in dit hoofdstuk zullen volgen, zijn dan ook tot stand gekomen met behulp van dit softwarepakket.

*Tabel 1: vergelijking tussen COGNEX en HALCON*

COGNEX	HALCON
Trial versie	Ontwikkelaarslicentie (ACRO)
Beperkte bibliotheek	Grote bibliotheek met voorbeelden
Universeel gebruik	Universeel gebruik
Geen basiskennis	Opleiding
Run-time licentie nodig bij implementatie	Run-time licentie nodig bij implementatie
Gebruik Vandersanden Group in Spouwen	Gebruik Vandersanden Group in Lanklaar

### 6.2 Kalibratie

Alvorens te starten met de effectieve beeldverwerking is het van belang om eerst de camera te kalibreren. Met kalibratie gebeurt er een nauwkeurige bepaling van de eigenschappen van de camera (maar ook van de lens), in een welbepaalde opstelling. Enkele van deze eigenschappen zijn bijvoorbeeld:

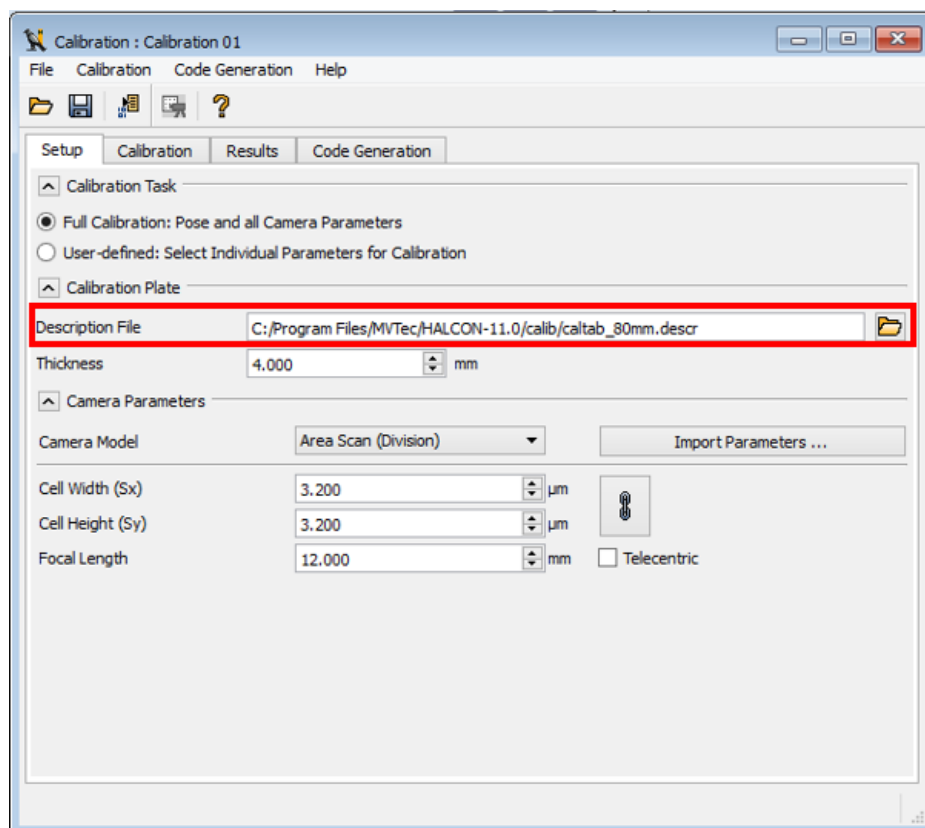
- fysieke afmetingen van een pixel (camera);
- camerapositie t.o.v. de *field of view* (camera);

- werkelijke focale lengte (lens);
- $C_x$  en  $C_y$ , centrum van het beeldvlak (= snijpunt van de optische as met het beeldvlak) [9];
- ...

Door kalibratie kan bijgevolg dus een nauwkeurigere beeldverwerking tot stand komen. Het verwijderen van distorsies uit het beeld is één van de methodes om de nauwkeurigheid van het beeld en dus de beeldverwerking te vergroten en gebeurt aan de hand van de bekomen parameters uit de kalibratie. Verdere bespreking van deze methode is weergegeven in paragraaf 6.3.2. . De werkwijze om een correcte kalibratie uit te voeren, is gegeven in de volgende deelparagraaf.

### 6.2.1 Werkwijze van kalibratie

HALCON biedt de gebruiker een *assistant manager* aan om de kalibratie eenvoudiger te laten verlopen. Deze *assistant manager* kan dus beschouwd worden als een programmawizard (stappenplan), toegepast op kalibratie. De eerste stap tot kalibratie is het ingeven van beginparameters. Deze parameters zijn volledig afhankelijk van de gebruikte opstelling (camera en lens) maar ook van de gebruikte kalibratieplaat. Deze kalibratieplaat vormt het hart van de gehele kalibratie zoals in de volgende stappen duidelijk zal worden. In figuur 34 is het kalibratiescherm weergegeven voor de eerste kalibratiestap.



**Figuur 34: kalibratiescherm - setup**

Men kan in figuur 34 de verschillende nodige beginparameters zien, namelijk:

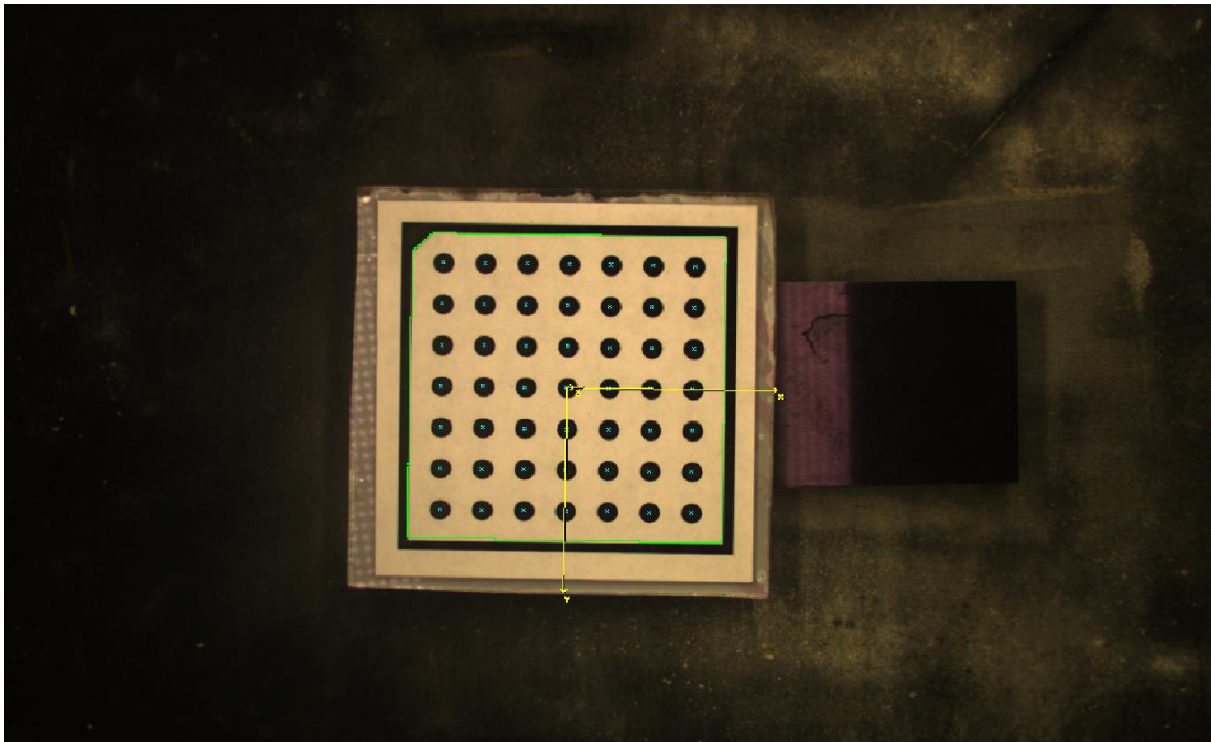
- dikte van de te gebruiken kalibratieplaat;
- fysische grootte van een pixel op de chip van de camera en x- en y-richting ( $S_x$  en  $S_y$ ) (datasheet);
- nominale focale lengte van de lens (datasheet).

Ook is het van belang om de *assistant manager* mee te geven welk type kalibratieplaat er voor de kalibratie zal gebruikt worden. De eigenschappen van een kalibratieplaat zijn vervat in een aparte file (.descr) en dient upgeload te worden in de *assistant manager* op de aangeduide plaats in figuur 34 (rode kader).

Na het ingeven van de gevraagde beginparameters, gaat men verder met stap twee van het kalibratieproces. Deze stap behelst onder andere de communicatie met de gebruikte camera waarbij de realisatie andermaal gebeurt aan de hand van de *assistant manager*. Eens de communicatie met de camera tot stand is gekomen, kan de effectieve kalibratie van de camera beginnen. Deze komt tot stand door toepassing van de volgende stappen:

1. positioneer de kalibratieplaat willekeurig in de *field of view* van het visiesysteem;
2. controleer of HALCON de kalibratieplaat detecteert;
3. neem een foto (*snap*) van het bekomen beeld;
4. herhaal voorgaande stappen totdat een minimum van 15 beelden bekomen werd (hoe meer beelden, hoe beter de kalibratie).

Een voorbeeld van een beeld is weergegeven in figuur 35. Op dit beeld kan men goed zien dat HALCON de kalibratieplaat gedetecteerd heeft. Zowel de punten als de (zwarte) contour van de plaat zijn immers aangeduid en tevens heeft HALCON hierop een assenstelsel toegepast.

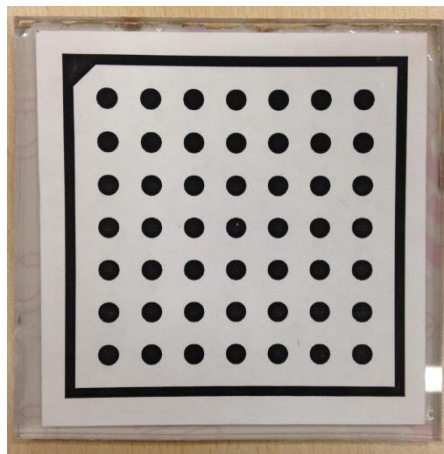


**Figuur 35: kalibratiebeeld**

Na het nemen van de beelden kan met behulp van de *assistant manager* de kalibratie van de camera voltooid worden. Het resultaat van deze kalibratie (parameters, positie) kunnen in een lijst worden opgeslagen maar kan ook als code gegenereerd worden om later in het beeldverwerkingsprogramma te gebruiken, bijvoorbeeld voor de implementatie van distorsieverwijdering.

## 6.2.2 Kalibratieplaat

Voorgaande deelparagraaf liet het belang van een kalibratieplaat al duidelijk uitschijnen. Deze platen zijn echter duur in aankoop (€300 - €500), waardoor men kan opteren om deze platen zelf te maken. In HALCON is het immers mogelijk om een kalibratiepatroon te genereren en op te slaan in een PostScript-bestand (.ps). Dit bestand kan men dan via een bestandsconverteer converteren naar een PDF-bestand. Dit bestand toont dan het gegenereerde kalibratiepatroon. Door dit dan af te drukken en bijgevolg op een glad oppervlak (bv. een stuk plexiglas) te plakken, bekomt men dus een zelfgemaakte kalibratieplaat die bruikbaar is voor de kalibratie van de camera. Een voorbeeld van een zelfgemaakte kalibratieplaat is weergegeven in figuur 36.



**Figuur 36: kalibratieplaat**

De procedure om in HALCON een kalibratiepatroon te genereren is weergegeven in figuur 37. Belangrijk hierbij op te merken is, dat naast het genereren van een PostScriptum-bestand ook een bestand wordt gegenereerd met de eigenschappen van het gegenereerde kalibratiepatroon (.descr). Het is dit bestand dat men zoals eerder gezegd in paragraaf 6.2.1 moet uploaden in de *assistant manager* zie figuur 34 om de kalibratie te kunnen starten.

```
1 | * Breedte van de caltab in meter
2 | Width := 0.175
3 | * Aantal dots per rij
4 | Rows := 7
5 | *
6 | MarkDist := Width/(Rows+1)
7 | *
8 | gen_caltab (Rows, Rows, MarkDist, 0.5, 'caltab_'+round(Width*1000)+'mm.descr', 'caltab_'+round(Width*1000)+'mm.ps')
```

**Figuur 37: genereren kalibratiepatroon**



## 6.3 Objectdetectie

### 6.3.1 Beelden inlezen

Een belangrijk item in de beeldverwerking is het opvragen van de (realtime)-beelden. Dit zijn de beelden die tijdens het proces worden gemaakt door de camera en waarop de beeldverwerking zal worden uitgevoerd. Figuur 38 geeft de procedure weer om realtime-beelden te kunnen inlezen. Op de plaats van de groene tekst in deze figuur, dient de eigenlijke beeldverwerking te komen.

```
open_framegrabber ('uEye', 1, 1, 0, 0, 0, 'default', 8, 'default', -1, 'false', 'default', '4', 0, -1, AcqHandle)
grab_image_start (AcqHandle, -1)
while (true)
    grab_image_async (Image, AcqHandle, -1)
    *
    *
    * Hier kom de code van het programma
    *
    *
endwhile
close_framegrabber (AcqHandle)
```

*Figuur 38: realtime-beelden inlezen*

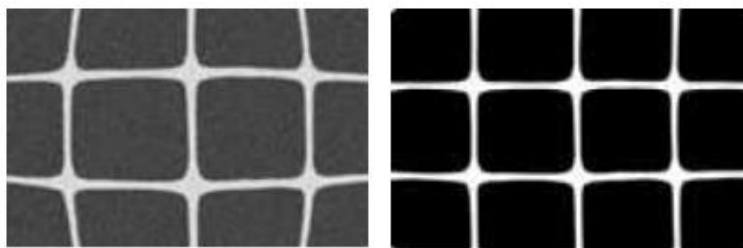
Om tot een werkend programma te geraken, moet er zeer veel getest en gesimuleerd worden. De simulaties zijn dan ook gebeurd op één foto, genomen van een (afgekeurde) baksteenstrip. Deze foto kan men inlezen door het volgende commando:

```
Read_image (: Image : Filename : ).
```

In hetgeen wat volgt (vanaf paragraaf 6.3.3), zullen de afbeeldingen waarop de bewerkingen worden weergegeven dan ook gebaseerd zijn op deze enkele foto. Een nadeel van deze aanpak is echter dat men geen gevarieerde input heeft en dat men bij het realtime-testen een grote kans heeft op noodzakelijke bijstellingen (aanpassingen) van het programma.

### 6.3.2 Distorsie

Alvorens met de effectieve beeldverwerking te starten, kan het van belang zijn het beeld van lensdistorsies te ontdoen. Distorsies zijn een bekend fenomeen in visiegebeuren en zijn vaak ongewenst. Ze zorgen namelijk voor een 'gebolde' weergave van het beeld, wat leidt tot onnauwkeurigheden wanneer er bepaalde zaken dienen gemeten te worden aan de hand van de opgenomen beelden. Een voorbeeld van dit fenomeen is weergegeven in figuur 39.



*Figuur 39: distorsie [9]*

De onnauwkeurigheden op metingen die ontstaan vanwege distorsies kunnen zoals eerder gezegd gecompenseerd worden door middel van kalibratie. Deze compensatie gebeurt dan aan de hand van de kalibratieparameters die men bekomt na kalibratie van de camera (zie paragraaf 6.2). De procedure om de compensatie te realiseren in HALCON bestaat uit de volgende twee commando's:

- *change\_radial\_distortion\_cam\_par ('fullsize', CameraParameters, 0, CamParOut);*
- *change\_radial\_distortion\_image (Image, Image, Image, CameraParameters, CamParOut).*

Alle bewerkingen van het beeldverwerkingsprogramma (uitgelegd in de volgende paragrafen) gebeuren op beelden waarop de distorsiecompensatie reeds gebeurd is.

### 6.3.3 Decompose – beeldontleding

Soms is het handig om een beeld te ontleden in zijn RGB-beelden. Zo verkrijgt men dus drie beelden waarvan het eerste beeld overeenkomt met de roodwaarde (R) van het originele beeld (uitgedrukt in grijswaardes), het tweede beeld overeenkomt met de groenwaarde (G) en het derde beeld met de blauwwaarde (B). Om dit visueel te kunnen voorstellen, is er een voorbeeld weergegeven in de volgende figuur (figuur 40).



*Figuur 40: origineelbeeld met respectievelijk R-,G-,B- beeld*

In bovenstaande figuur is het duidelijk zichtbaar dat in het R-beeld de grijswaarde het hoogst is voor het rode gebied van het origineel beeld. Hetzelfde fenomeen is zichtbaar bij de andere beelden maar dan voor respectievelijk het groene en het blauwe gebied.

De reden voor het gebruik van deze operatie is om de latere beeldverwerking eenvoudiger te laten verlopen. Het is namelijk zo dat de kleur van de transportband waarop de baksteenstrippen vallen, een groene kleur heeft. Door *decompose* zal de transportband veel lichter worden weergegeven in het G-beeld dan de elementen (steenstrippen) die er op liggen. De strippen kunnen bijgevolg makkelijker uit het beeld gehaald worden.

### 6.3.4 Threshold

Vorige paragraaf gaf al aan dat er ook gebruik moet gemaakt worden van een *threshold*. Hier is bewust gekozen voor een reguliere *threshold* boven een dynamische *threshold*. De reden hiervoor kan men zoeken in het feit dat er gewerkt wordt met een 'donkere kamer'. Het visiesysteem zal zich namelijk een soort van box bevinden (afscherming) waardoor het indringen van omgevingslicht dan ook zeer sterk beperkt wordt in het visiesysteem. Aangezien een dynamische *threshold* echter niet gebruikt wordt om de invloed van instabiele omgevingsparameters te verminderen, is deze in dit geval niet zozeer van toepassing en is een reguliere *threshold* meer dan voldoende.

De functie van een *threshold* om gewenste onderdelen uit een beeld te halen en overbodige elementen weg te 'filteren'. Zo kan men bijvoorbeeld een *threshold* instellen waarbij men zegt dat alle pixels in het grijswaardenbereik [125...250] moeten worden bijgehouden. Alle pixels die hier dan niet aan voldoen zullen dan logischerwijs uit het beeld weg gefilterd worden. Hier wordt meteen ook de reden van het *decompose*-operatie (zie vorige paragraaf) duidelijker. Aangezien de achtergrond (de transportband) veel lichter wordt weergegeven in het G-beeld, kan een *threshold* de donkerdere pixels (weergave van de steenstrippen) makkelijker eruit halen. In figuur 41 is er een algemeen voorbeeld gegeven om de werking van de *threshold* operatie weer te geven. Een *threshold* ontstaat met behulp van het volgende commando:

*Threshold ( Image : Region : MinGray,MaxGray : ).*



**Figuur 41: thresholdoperatie**

Het is belangrijk op te merken dat hoe lager de grijswaarde van een pixel is, hoe donkerder zijn weergave is. Vandaar ook dat het bereik zich eerder in de lager grijswaarden bevindt (grijswaarden gaan van 0 tot 256).

### 6.3.5 Connectie en regioselectie

Zoals vermeld in de vorige paragraaf, houden we na het gebruik van de *threshold* enkel die elementen over die relevant zijn voor de gevraagde toepassing. Na de *threshold* wordt er gebruik gemaakt van een instructie die alle aan elkaar grenzende pixels met elkaar verbind tot

zogenaamde regio's [18]. Deze regio's kunnen dan vervolgens door de functie *select\_shape* geselecteerd worden op basis van verschillende criteria. Voorbeelden van deze criteria zijn:

- regio-oppervlakte,
- rondheid,
- rechthoekigheid,
- gemiddelde grijswaarde,
- compactheid,
- ...

In bijlage F is de complete programmacode uit HALCON weergegeven waarin men ook de gebruikte criteria kan terugvinden.

Om de steenstrip volledig uit het beeld te filteren, is ervoor gekozen de functie *select\_shape* op basis van het oppervlaktecriterium uit te voeren. Hierdoor worden bijgevolg enkel die regio's geselecteerd (en bijgehouden) die voldoen aan de opgegeven oppervlaktegrenzen, welke proefondervindelijk zijn vastgelegd aan de hand van de nodige simulaties. Deze manier van werken is bijgevolg handig om ongewenste onzuiverheden die nog op de transportband kunnen aanwezig zijn, uit het beeld te filteren.

Wanneer dit alles gebeurd is, kan men ook nog gebruik maken van de functie *reduce\_domain*. Zo verkrijgt men het beeld waarop enkel de steenstrip te zien is. Dit is vooral gedaan in functie van de overzichtelijkheid.

### 6.3.6 Softwarematige sensor

In de voorbije deelparagrafen zijn de belangrijkste onderdelen voor de objectdetectie uitgelegd. Om echter tot een goed werkend systeem te komen, zijn er ook instructies nodig die het programma 'sturen'. Dit wil zeggen dat deze instructies nodig zijn om bepaalde onderdelen van de objectdetectie aan te spreken indien er aan een bepaalde voorwaarde wordt voldaan. Zo is er in de programmacode voorzien dat de beeldverwerking slechts start wanneer de steenstrip voorbij een bepaalde pixelrij van het beeld gepasseerd is (een beeld bestaat namelijk uit een groot aantal pixelrijen en -kolommen). Is er echter niet aan de vernoemde voorwaarde voldaan, dan zal de objectdetectie steeds opnieuw beginnen totdat hier wel aan voldaan is. Dit is dus het principe van een softwarematige sensor. Die voorkomt dat de beeldverwerking wordt gestart zelfs wanneer er geen steenstrip aanwezig is. Dit kan leiden tot foutmeldingen, maar ook tot zinloos tijdsverlies. Het is namelijk belangrijk om de bewerkingstijd steeds zo laag mogelijk te houden, zodanig dat het systeem de mogelijkheid heeft sneller te kunnen werken.

Eventueel had er voor dit alles gekozen kunnen worden voor een hardwarematige sensor. De reden waarom hiervoor niet is gekozen is het feit dat de extra programmacode geen extra meerkost vormt voor het systeem. Indien men kiest voor een hardwarematige sensor, moet men rekening houden met de kost van de sensor zelf, de communicatie van de sensor met het systeem (I/O-kaart) en de kans op falen van de sensor (slijtage, vervanging ...). Indien de software faalt kan men dit herprogrammeren zonder veel extra kosten. Omwille van de

aangehaalde redenen is er in dit geval dus gekozen voor een softwarematige sensor. Belangrijk hierbij op te merken is echter dat dit type sensor vooral interessant is indien men het gewenste *sensorsignaal* kan bekomen door enkel gebruik te maken van een beeld waarbij slechts een beperkte implementatie van extra code nodig is (voor de eenvoudigere sensortoepassingen).

De volledige programmacode van de objectdetectie is ook weer terug te vinden in de complete programmacode van HALCON, die zoals eerder gezegd weergegeven is in bijlage F.

## 6.4 Kwaliteitscontrole van de steenstrip

### 6.4.1 Contourcontrole

In dit onderdeel van de kwaliteitscontrole, beschrijven we de verschillende methodes om de contour van de steenstrip te controleren. Het zal onder andere gaan over de rechthoekigheid van de steenstrip, afmetingen, hoekcontrole, oppervlaktecontrole (afgebroken stukken)... Ook hier beschrijven we niet alle gebruikte programma-instructies, maar vooral de belangrijkste instructies die nodig zijn om de denkwijze achter de betreffende kwaliteitscontrole te verklaren.

### 6.4.2 Lengte- en breedtebepaling van de steenstrip

In elk productieproces is continuïteit van de producten een zeer belangrijk gegeven. Zo is het bijvoorbeeld dus van belang dat de afmetingen binnen welbepaalde grenzen blijven. Dit geldt ook voor de baksteenstrippen. Het cliënteel is immers niet gediend met strippen die sterk afwijkende afmetingen hebben. Om dit probleem aan te pakken, heeft Vandersanden NV toleranties ingesteld op de afmetingen (lengte en breedte) van de steenstrip. Zo mag elke steenstrip een maximale afwijking hebben van 1 mm op zijn nominale afmetingen.

Om de lengte en de breedte van de steenstrip te kunnen 'opmeten' maken we gebruik van de instructie *smallest\_rectangle2*. Deze instructie omhult de steenstrip als het ware met de kleinste mogelijke ('best passende') rechthoek. Door dan van deze virtuele rechthoek zowel de lengte als de breedte op te vragen, verkrijgen we de afmetingen in pixelwaarden (aantal pixels). Dit heeft dus als gevolg dat er nog een schaling moet gebeuren van aantal pixels naar millimeter. Deze schaling is gerealiseerd door een stuk metaal met gekende (nauwkeurige) afmetingen op te meten met behulp van de programmacode (dus in pixelwaarde). De vergelijking van het aantal pixels met het overeenkomstig aantal millimeter zorgt voor een schaling die voor alle andere metingen (dus ook op steenstrippen) van toepassing is. Ter vervollediging is in onderstaande figuur (figuur 42) een gesimuleerde steenstrip gegeven waarop *smallest\_rectangle* is uitgevoerd.



*Figuur 42: smallest\_rectangle steenstrip*

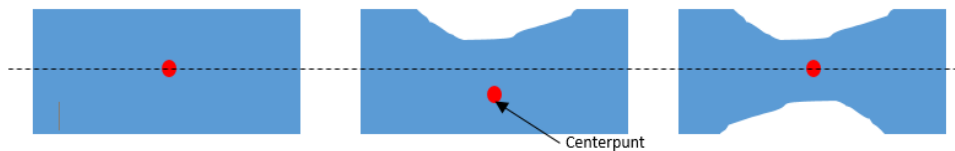
### 6.4.3 Rechthoekigheid en vormcontrole

Een andere controle op de kwaliteit van de steenstrip is de rechthoekigheid en de vormcontrole. Met rechthoekigheid bedoelen we de mate waarin de steenstrip overeenkomt met een perfecte rechthoek. Deze overeenkomst kan in HALCON bepaald worden door de instructie *Rectangularity*. Deze instructie geeft een waarde weer die indicatie biedt voor de eerder besproken overeenkomst. Deze waarde is gelegen tussen 0 en 1. Hoe dichter de waarde bij 1 ligt, hoe beter de overeenkomst. Door nu met *if-voorwaardes* te werken, kan de minimale waarde worden ingesteld waaraan de overeenkomst moet voldoen. Deze grens is experimenteel bepaald en is m.a.w. niet wiskundig bepaald. Het is echter nog van belang op te merken dat deze methode niet de enige is om een indicatie voor de rechthoekigheid te bekomen. Zo zou men ook de vier hoekpunten kunnen controleren op een bepaalde graad van rondheid. Via deze methode kan van dan eveneens een afwijking op de rechthoekigheid detecteren. Een te grote afronding zorgt voor een afwijking op de vorm van een rechthoek, wat dus de graad van overeenkomst negatief beïnvloedt.

De vormcontrole kan ook op verschillende manieren geschieden. Hier hebben we twee methodes bedacht, maar natuurlijk slechts één uitgewerkt. De eerste methode geschiedt op basis van oppervlaktevergelijking. Bij deze methode vergelijkt men immers de oppervlakte van de perfecte rechthoek (omhullende rechthoek van de steenstrip) met de oppervlakte van de steenstrip. Deze oppervlakte kan men bekomen door de *Area* van het object (steenstrip) op te vragen met bijvoorbeeld de functie *area\_center*. Met deze functie kan men namelijk zowel de oppervlakte opvragen als de centrumcoördinaten van het gekozen gebied (object/steenstrip). Indien nu de oppervlaktes met elkaar worden vergeleken en wederom enkele grenzen gekoppeld worden aan deze vergelijking, kan er dus een vormcontrole worden gerealiseerd. Het is namelijk zo dat wanneer de oppervlakte van de steenstrip te veel verschilt met die van de 'perfecte rechthoek', er stukken in het oppervlak van de steenstrip ontbreken en er dus bijgevolg een te grote afwijking is op de gewenste vorm van de steenstrip. Figuur 42 toont duidelijk dat aan de bovenzijden van de steenstrip er een serieuze afwijking is op de gewenste, rechthoekige vorm (inkeping).

De tweede methode voor de uitvoering van een vormcontrole geschiedt op basis van controle van het centerpunt van de steenstrip. Door na te gaan of de coördinaten van dit centrum 'sterk' afwijken van de centrumcoördinaten van de 'perfecte rechthoek', krijgen we een indicatie voor de vormcontrole. Het centrum van de steenstrip wordt immers bepaald door de oppervlakte ervan. Als deze oppervlakte bijgevolg afwijkt van het gewenste, krijgt men dus een verandering in de centrumcoördinaten en bijgevolg ook een afwijking van de 'gewenste' coördinaten. Nu is het echter zo dat deze methode niet wordt gebruikt voor de beeldverwerking van het

visiesysteem (wel de eerste methode), vanwege het niet sluitende resultaat. Het kan namelijk zijn dat zowel aan de bovenkant als aan de onderkant van de steenstrip een serieuze afwijking aanwezig is. Vanwege een vorm van symmetrie zal het centerpunt van de steenstrip nu toch nog ongeveer overeenkomen met het 'ideale' centerpunt. Door dit fenomeen zou de steenstrip op basis van vormcontrole toch nog goedgekeurd kunnen worden, hoewel er toch sterke afwijkingen aanwezig zijn op boven- en onderkant. Onderstaande figuur (figuur 43) demonstreert dit fenomeen nog eens ter verduidelijking.



*Figuur 43: centerpunt van de steenstrip*

#### 6.4.4 Hoekcontrole

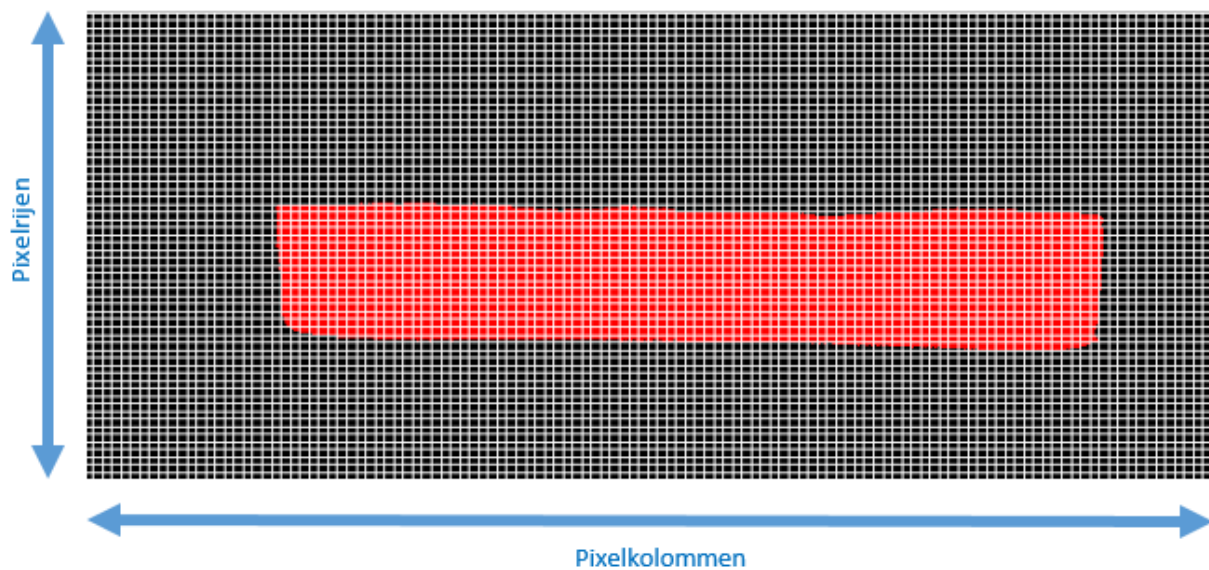
Een ander aspect in de kwaliteitscontrole van de strip is de hoekcontrole. Hier gaat men na of alle vier de hoeken van de steenstrip een goed vorm hebben. Voor de hoekvorm wilt dit vooral zeggen dat de afronding binnen bepaalde grenzen moet gelegen zijn alsook het feit dat hoekbreuken moeten vermeden worden. Deze zogenaamde grenzen of toleranties waaraan de hoeken moeten voldoen, worden experimenteel vastgelegd en tevens in samenspraak met Vandersanden NV.

Het principe van de hoekcontrole leunt sterk aan de eerder besproken controle op rechthoekigheid van de gehele steenstrip. Ook hier wordt er immers gebruik gemaakt van de rechthoeksfactoren van de hoeken. Deze factoren worden dan getoetst aan experimenteel bepaalde tolerantiegrenzen (eveneens in samenspraak met Vandersanden NV). Dit geeft als resultaat een indicatie voor de vorm en de correctheid van de verschillende hoeken. Om de rechthoeksfactoren van de verschillende hoeken nu te kunnen bepalen, moeten de hoeken los van het geheel worden bekeken. Een eerste belangrijke stap in dit proces is de correct oriënteren van de steenstrip in het beeld. Het is immers belangrijk dat deze oriëntatie voor elke steenstrip hetzelfde is, namelijk onder een hoek van  $0^\circ$  en dus horizontaal gelegen in het beeld. De reden waarom de oriëntatie zo belangrijk wordt geacht, is vanwege de verdere procedure om de rechthoekigheid van de hoeken te bepalen. De hoeken worden immers uit het beeld gefilterd door gebruik te maken van de eigenschappen van de volledige steenstrip. Het gaat hierbij vooral om de rijen en kolommen van de steenstrip die het in het desbetreffende beeld omvat (soort van coördinaten). Indien de oriëntatie van elke steenstrip niet ongeveer hetzelfde wordt weergegeven in het beeld (de steenstrippen liggen immers niet mooi recht langs elkaar op de transportband), zal de programmaprocedure foute bewerkingen uitvoeren. Dit zal duidelijker worden bij de gedetailleerdere uitleg van deze procedure (zie paragraaf 6.4.5). In de onderstaande figuur (figuur 44) wordt voor de duidelijkheid nog eens weergegeven dat de steenstrip in het programmabeeld een regio bevat bestaande uit een aantal pixelrijen en -kolommen.

Om de oriëntatiecorrectie in de programmacode te verwerken, zijn de volgende twee instructies van belang:

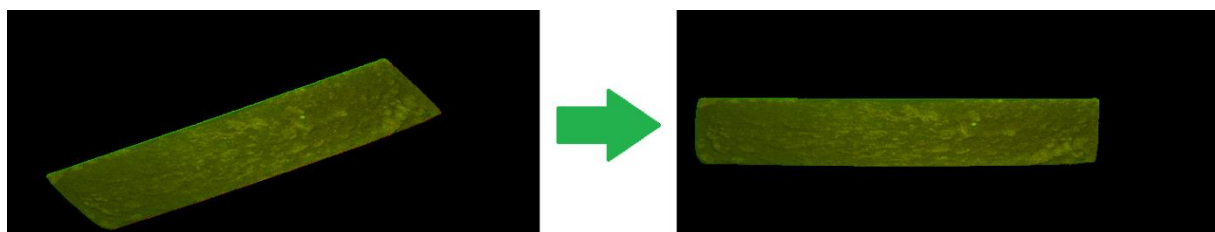
- *orientation\_region;*
- *rotate\_image.*

Het eerste commando geeft de hoek terug van het gewenste object/regio. *Rotate\_image* daarentegen gebruikt men om het beeld met een bepaalde hoek te draaien en geeft dan het resulterende beeld weer. Indien nu de output van de eerste instructie (een hoek) wordt gebruikt als input van de tweede instructie, wordt het beeld gedraaid zodanig dat het gewenste object terug onder een hoek van  $0^\circ$  staat. De gewenste oriëntatie is bijgevolg bekomen.



*Figuur 44: pixelrijen en -kolommen van een beeld*

De besproken procedures voor de oriëntatie van de steenstrippen zijn eveneens terug te vinden in de complete programmacode in bijlage F. In onderstaande figuur (figuur 45) is het resultaat van deze procedures nog eens visueel weergegeven.



*Figuur 45: oriënteren van de steenstrip*



Nu de gewenste oriëntatie van de steenstrip behaald is, kan de hoekcontrole effectief beginnen. De manier van werken of de procedure is als volgt:

- gebied van hoekpunt afbakenen;
- hoekpunt uit het beeld filteren (een apart object/regio van maken);
- rechthoeksfactor bepalen;
- controle met tolerantiegrenzen.

De eerste stap bij de hoekcontrole is het afbakenen van het hoekpunt. Dit kan op verschillende manieren gebeuren, maar hier is geselecteerd voor het afbakenen met behulp van een 'masker'. Het masker is een rechthoek en wordt geplaatst over het desbetreffende hoekpunt. Dit kan men realiseren met behulp van het commando *gen\_rectangle1*. Het commando construeert een rechthoek die bekomen wordt door de overstaande hoeken te definiëren aan de hand van de pixelrijen en -kolommen die de steenstrip in het beeld bevat. Wederom kan men in bijlag F deze procedure terugvinden, toegepast voor elk hoekpunt. De procedure toegepast op de linkerbovenhoek (gekeken op het beeld) is hier nogmaals weergegeven:

*gen\_rectangle1 (corner1, Rmin-20, Cmin-20, Rmin+100, Cmin+100).*

Het commando is als volgt te interpreteren:

- *corner1* → naam die gegeven wordt aan de betreffende region;
- *Rmin-20, Cmin-20* → coördinaat van het eerste maskerhoekpunt(links boven);
- *Rmin+100, Cmin+100* → coördinaat van het overstaande maskerhoekpunt (rechts onder).

Belangrijk om te weten is dat *Rmin* en *Cmin* de kleinste rijwaarde respectievelijk kolomwaarde voorstellen van de steenstrip in het beeld. De hoekpunten zijn dus gedefinieerd aan de hand van de pixelrijen en -kolommen van de steenstrip (in het beeld).

Nadat het masker is gegenereerd, is het nodig het hoekpunt uit het beeld te halen. Een mogelijkheid om dit te realiseren is door gebruik te maken van het commando *reduce\_domain*. Dit commando is al eerder ter sprake gekomen bij het uitfilteren van de steenstrip uit het globale beeld. Nu is het gebruikt om de steenstrip te reduceren tot de regio overkoepeld door het gegenereerde masker, dus het hoekpunt. Wederom kan in bijlage F de exacte procedure gevonden worden, toegepast op elk hoekpunt. De *reduce\_domain* toegepast op de linkerbovenhoek is de volgende:

*reduce\_domain (Baksteenstrip1, corner1, cornerZoom1).*

Dit commando is als volgt te interpreteren:

- *baksteenstrip* → 'originele' beeld van (uitgefilterde) steenstrip;
- *corner1* → gebied dat uit *Baksteenstrip1* moet gehaald worden;
- *cornerZoom1* → resultaat van de bewerking, het gereduceerde gebied.

Een voorbeeld van de werking van *reduce\_domain* is ter verduidelijking nog weergegeven in figuur 46.



**Figuur 46: oplegging van het rechthoekig masker met uitvergroting van de gefilterde hoek**

In bovenstaande figuur ziet men in het bovenste deel het rechthoekig masker dat op een hoekpunt wordt gelegd. Het onderste deel geeft het resultaat van *reduce\_domain* weer uitgevoerd op dit masker. Zo wordt alles wat omvat is door het masker afgezonderd uit het beeld en kunnen de nodige bewerkingen voor de hoekcontrole plaatsvinden.

Zoals vermeld in het begin van deze paragraaf, leunt het principe van de hoekcontrole sterk aan de controle op rechthoekigheid van de globale steenstrip. Daar werd gebruik gemaakt van het commando *rectangularity* om de rechthoeksfactor van de strip te bepalen. Ook hier past men deze manier van werken toe. Zo verkrijgt men de rechthoeksfactor van het desbetreffende hoekpunt. Deze waarde wordt dan vervolgens getoetst aan de opgelegde (experimenteel bepaalde) toleranties waarbinnen deze factor gelegen moet zijn. Is aan de voorwaarde voldaan, dan zal de hoek goedgekeurd worden. Is dit echter niet het geval, dan wordt de hoek bijgevolg afgekeurd en daarbij ook de volledige steenstrip.

Een volledige hoekcontrole bestaat er echter in elk hoekpunt te controleren. Door vier maal hetzelfde principe toe te passen, elke voor één hoekpunt, verkrijgen we een globale hoekcontrole voor de steenstrip. Belangrijk hierbij op te merken is dat wanneer de er een fout wordt ontdekt in het eerst gecontroleerde hoekpunt, de andere hoekpunten niet verder worden gecontroleerd. Aangezien de steenstrip toch wordt afgekeurd, zou dit immers tijdsverlies betekenen en zou het programma onnodige alle stappen uitvoeren (de mogelijk haalbare snelheid van het visiesysteem zal dan verlagen).

## 6.4.5 Spievormcontrole a.d.h.v. lasertriangulatie

### 6.4.5.1 Laserlijnerkenning

In paragraaf 5.4.1 werd het principe van lasertriangulatie, toegepast op hoogtemetingen, al uitgelegd. In dit deel wordt dat principe geïmplementeerd in de programmacode van de beeldverwerking. Het is namelijk van belang om de hoogte (dikte) van de strip te meten om op basis van die metingen conclusies te kunnen trekken over de spievormigheid van de steenstrip.

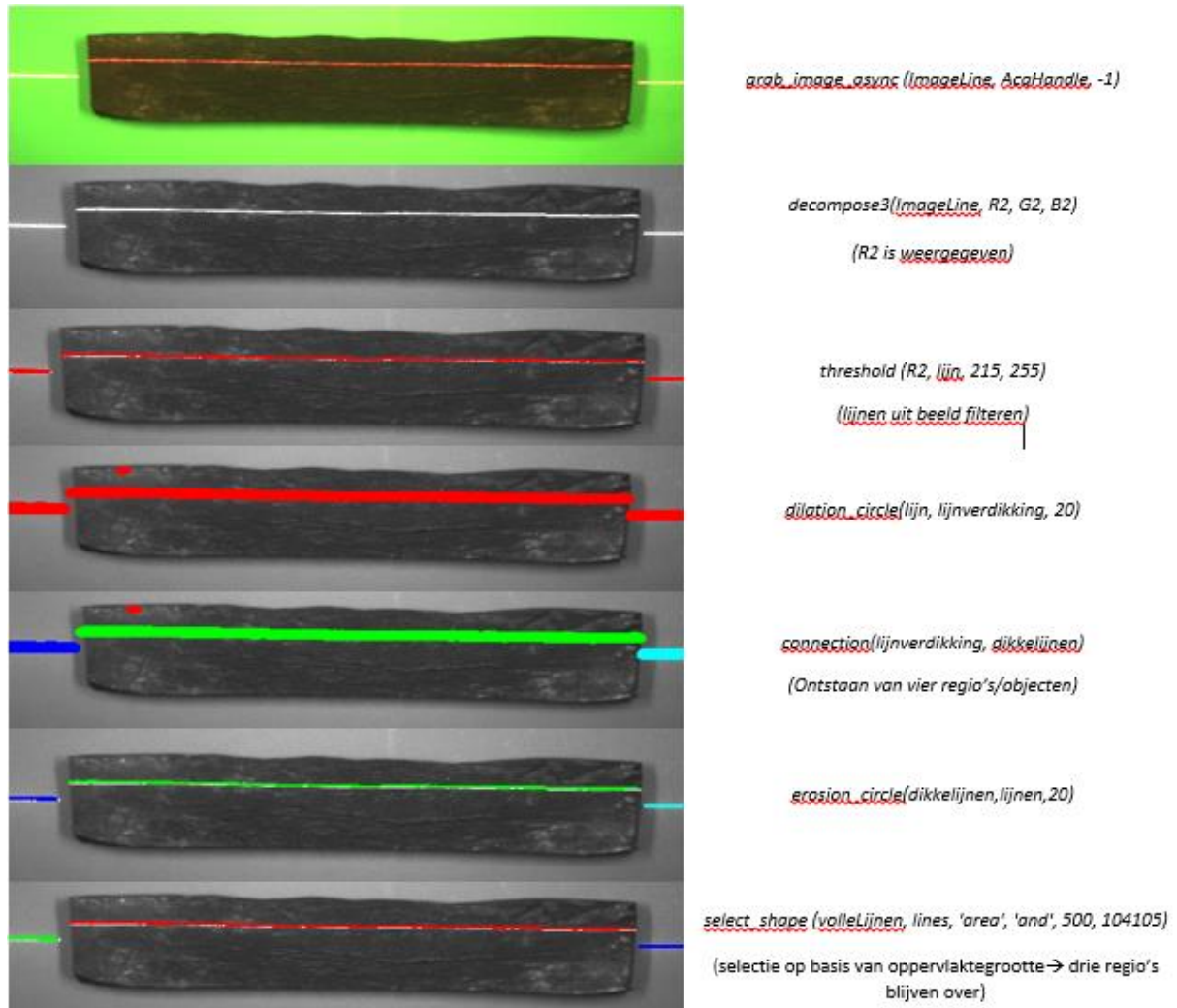
Het eerste wat er moet gebeuren om de triangulatie uit te kunnen voeren, is het detecteren van de laserlijn(en) in het beeld. Deze lijnen moeten uit het beeld gefilterd worden zodanig dat enkel hiermee verder wordt gegaan. De lijnerkenning of -filtering gebeurt op analoge wijze als bij de herkenning van de steenstrip, op uitzondering van andere parameterwaarden (*dilation*, *threshold* ...). De volgende procedures halen de laserlijn(en) uit het beeld (zie ook weer bijlage F):

- *grab\_image\_async (ImageLine, AcqHandle, -1);*
- *decompose3(ImageLine, R2, G2, B2);*
- *threshold (R2, lijn, 215, 255);*
- *dilation\_circle(lijn, lijnverdikking, 20);*
- *connection(lijnverdikking, dikkelijnen);*
- *erosion\_circle(dikkelijnen,lijnen,20);*
- *fill\_up(lijnen, volleLijnen);*
- *select\_shape (volleLijnen, lines, 'area', 'and', 500, 104105);*

Deze procedure begint met het (asynchroon) nemen van beelden via de camera (*grab\_image\_async*). Het beeld dat men verkrijgt is hier *ImageLine* genoemd. Deze zal men nu 'splitsen' in zijn rood-, groen- en blauwwaarde die elk een nieuw beeld genereren, respectievelijk *R2*, *G2* en *B2*. Aangezien de gebruikte laserlijn een rode kleur heeft, valt deze zeer goed op in het beeld *R2* (roodwaarde). Door hierop nu een gepaste *threshold* op uit te voeren (experimenteel), kan de laserlijn uit het beeld (*R2*) worden gefilterd.

Door kwalitatief mindere lijnlasers, maar ook door diepere groeven in de steenstrip, kan het voorvallen dat de laserlijn die zichtbaar is op de steenstrip meermaals kort onderbroken wordt. Dit kan voor problemen zorgen, aangezien deze lijn dan als verzameling van lijnstukken wordt gezien in plaats van één enkele. Om dit fenomeen zoveel mogelijk te beperken, is er gebruik gemaakt van het commando *dilation\_circle*. Die verdikt de lijnstukken met een opgegeven factor zodanig dat de verschillende lijnstukken elkaar onderling raken. Op die manier kan dan met het commando *connection*, dewelke ook eerder vernoemd is bij de steenstripherkenning, aanliggende pixels (van de verschillende lijnstukken) met elkaar verbonden worden zodat er één enkele regio ontstaat voor het gedeelte van de laserlijn dat invalt op de steenstrip. Hierop voeren we vervolgens het commando *erosion\_circle* uit. Deze doet net het omgekeerde van *dilation\_circle*. Zo bekomt men terug de originele dikte van de laserlijn, met dit verschil dat de lijn nu als één geheel wordt beschouwd in plaats van een verschillende aparte lijnstukken. Het commando *fill\_up* is hierbij een optioneel commando die ervoor zorgt dat de gefilterde regio's

(lijnen) opgevuld worden zodanig dat hierin geen gaten in voor komen. Tot slot zorgt het commando *select\_shape* ervoor dat die regio's uit het beeld (*R2*) worden geselecteerd die aan de opgegeven (experimenteel bepaalde) criteria voldoen (bv. grootte van het gebied). De totale procedure voor het uitfilteren van de laserlijn(en) is ook visueel weergegeven in figuur 47. Deze geeft in chronologische volgorde het resultaat van de verschillende bewerkingen weer.



**Figuur 47: uitfiltering van laserlijn**

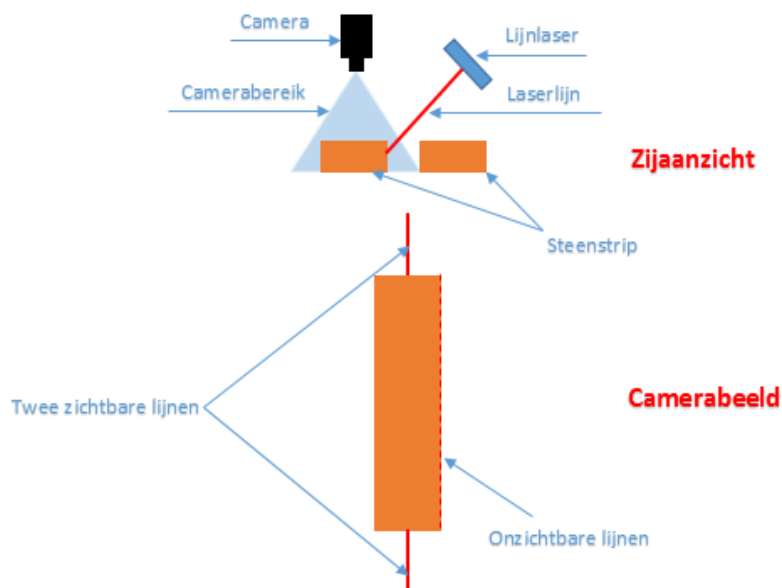
Figuur 47 toont dat bij de lasertriangulatie steeds maximum drie lijnen zichtbaar zijn (ook mede dankzij het commando *dilation\_circle* zoals eerder besproken). Bijgevolg zijn er dus steeds drie mogelijkheden waarmee men te maken krijgt op het gebied van aantal zichtbare lijnen:

- één zichtbare lijn,
- twee zichtbare lijnen,
- drie zichtbare lijnen.

Eén zichtbare lijn is mogelijk wanneer er geen steenstrip op de transportband aanwezig is. Indien dit het geval is, valt de laserlijn enkel in op de vlakke transportband waardoor er geen significante breking van het laserlicht ontstaat. Bijgevolg wordt er dus slechts één lijn

gedetecteerd. In dit geval dient het programma dan ook de volledige procedure waarin de spievormcontrole (in lengterichting) gebeurt over te slaan. Dit spaart veel rekenkracht uit, waardoor de het visiesysteem de mogelijkheid heeft sneller te werken (beeldverwerking gebeurt immers sneller. Ook zal het programma snel terugkeren naar zijn begintoestand, namelijk de start van de steenstripherkenning. Dit is nodig om een continue flow in het programma te krijgen. Het is namelijk van belang dat het systeem zo snel mogelijk reageert indien er een nieuwe steenstrip aanwezig is. Wanneer het programma immers nog bezig is met de verwerking van de spievormcontrole terwijl dit niet zinvol is (slechts één lijn zichtbaar), dan verliezen we teveel tijd voor het detecteren van de nieuwe steenstrip om zo de hele kwaliteitscontrole te starten voor de desbetreffende (nieuwe) strip.

Het geval waarin twee zichtbare lijnen door de camera in het beeld wordt opgenomen situeert zich vooral op het einde van kwaliteitscontrolecyclus. Bij de overgang van de ene steenstrip naar de andere kan de laserlijn gedeeltelijk invallen op de zijkant van de volgende steenstrip. Dit deel van de laserlijn zal dus amper zichtbaar zijn, waardoor bijgevolg slechts twee volwaardige laserlijnen door de camera opgenomen worden als zichtbare lijnen. Ook in dit geval moet net zoals in de situatie met één zichtbare lijn, de spievormcontrole (in lengterichting) worden overgeslagen. Ter verduidelijking is in figuur 48 het principe van deze gedachtegang visueel weergegeven.

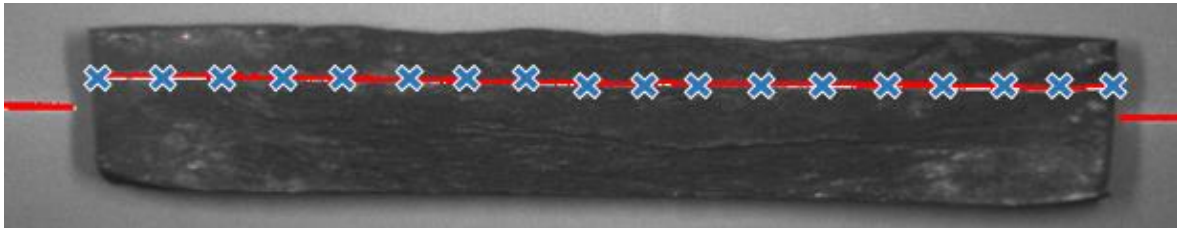


**Figuur 48: zichtbare lijnen bij strippenovergang**

De situatie waarin er drie zichtbare lijnen aanwezig zijn in camerabeeld is echter degene waarop de spievormcontrole dient te gebeuren. Zo bekommen we namelijk het beeld zoals weergegeven in figuur 31 (paragraaf 5.4.1). Voor de gedetailleerdere bespreking van de spievormcontrole is in hetgeen wat volgt dan ook uitgegaan van de situatie met drie zichtbare lijnen.

#### 6.4.5.2 Hoogtemeting a.d.h.v. lasertriangulatie

Het principe van de hoogtemeting aan de hand van lasertriangulatie is uitgelegd in paragraaf 5.4.1. Om de implementatie ervan in de programmacode te realiseren, moeten de drie zichtbare laserlijnen ontleed worden in een aantal *merkerpunten*. Deze punten zorgen voor een opdeling van de lijnen. Een voorbeeld van zulke opdeling is weergegeven in figuur 49 met behulp van kruisjes op de plaats van de merkerpunten.



*Figuur 49: merkerpunten laserlijn*

Op de positie van deze merkpunten bepaalt men nu de hoogte van de steenstrip aan de hand van lasertriangulatie. Om nu echter tot de effectieve realisatie van deze hoogtemeting te komen en te implementeren in de programmacode, is het volgende stappenplan gevolgd:

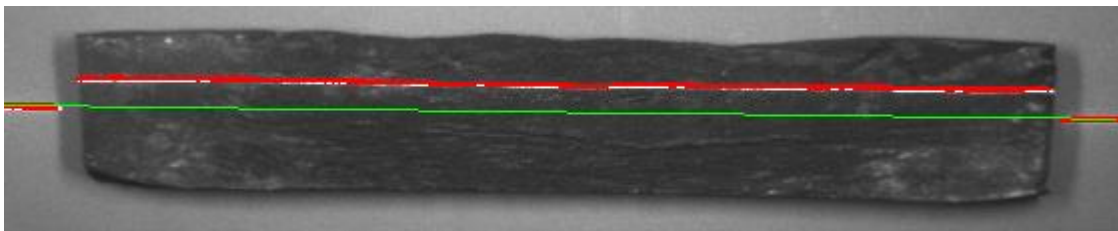
- lijneigenschappen opvragen (rijen en kolommen),
- eigenschappen sorteren,
- referentielijn construeren,
- merkpunten kiezen, creëren,
- hoogtes op de verschillende merkpunten bepalen,
- analyseren van bekomen hoogtes,
- goedkeuren of afkeuren.

Als eerste vraagt men dus de lijneigenschappen op van iedere lijn apart. Deze eigenschappen vertegenwoordigen de pixelrijen en -kolommen die de lijnen bezitten in het beeld, analoog aan de pixelrijen en -kolommen van de steenstrip zoals uitgelegd in paragraaf 6.4.4. Deze rijen en kolommen vormen als het ware de coördinaten van de verschillende pixels die de desbetreffende lijn vertegenwoordigen. Belangrijk is op te merken dat de rijen en kolommen in twee aparte *tuples* worden teruggegeven. Deze *tuples* zijn lijsten waarin de rijen of kolommen zijn opgeslagen. De wijze waarop deze rijen en kolommen zijn opgeslagen, vormen een probleem om onmiddellijk merkerpunten te kiezen en af te zonderen. Het is immers zo dat de rijen in de lijsten worden opgeslagen in oplopende volgorde. Voor de kolommen is dit echter niet het geval. Het is namelijk zo dat de laserlijnen worden weergegeven door allemaal pixels, waarvan er verschillende gelegen zijn op eenzelfde pixelrij. De desbetreffende pixels zijn dus van elkaar verschillend omdat ze door een andere pixelkolom gekarakteriseerd worden. Aangezien nu de lijst met pixelrijen oplopend gesorteerd is, is de lijst met pixelkolommen slechts oplopend gesorteerd (in groepjes) per rijnummer. Met andere woorden de sortering herbegint in de lijst met de kolommen indien het rijnummer verandert. Dit is ter verduidelijking nog eens weergegeven in tabel 2.

**Tabel 2: sorteren van rijen en kolommen in HALCON**

Sortering van rijen en kolommen in HALCON												
Rij	500	500	500	500	501	501	501	503	503	503	504	504
Kolom	1402	1450	1500	1542	1300	1359	1480	1320	1426	1954	1589	1965

Aan de hand van de verkregen eigenschappen (pixelrijen en -kolommen) van de verschillende lijnen, dient nu een referentielijn gemaakt te worden. Deze referentielijn maakt de verbinding tussen de meest linkse en rechtse lijn. Ze wordt bekomen door van zowel de linkse als de rechtse lijn de mediaan van de pixelrijen en -kolommen te nemen. Zo bekomt men voor elke lijn één enkele *pixelcoördinaat* (*Mediaan\_kolom*; *Mediaan\_rij*). Door de bekomen coördinaat van beide lijnen te verbinden met behulp van het commando *gen\_region\_line*, verkrijgt men het resultaat zoals weergegeven in figuur 50, waarbij de groene lijn de referentielijn voorstelt. Het is ten opzichte van deze referentielijn dat de hoogte door middel van lasertriangulatie wordt bepaald. De afstand tussen de gecreëerde merkerpunten en deze referentielijn is immers een maatstaf voor de hoogtemeting (principe is reeds uitgelegd in paragraaf 5.4).



**Figuur 50: referentielijnen voor hoogtemeting**

Nadat de referentielijn is geconstrueerd, bepaalt men de merkpunten op de middelste lijn (zoals in figuur 49). Het algoritme om deze merkpunten te bepalen, is eveneens weergegeven in de complete programmacode in bijlage F en aangegeven met de subtitel "*Bepalen van de merkpunten*". Voor de overzichtelijkheid is dit algoritme ook nog eens weergegeven in figuur 51 (volgende bladzijde).

```

gen_region_line(Rechterlijn, BeginrijRechts, BeginkolomRechts, EindrijRechts, EindkolomRech
tuple_sort(KolMidden, KolMiddenSort)
tuple_uniq (RijMidden, RijUniekSort)
tuple_uniq(KolMiddenSort, KolUniekSort)
tuple_length(KolUniekSort, KUlengte)
tuple_length(RijUniekSort, RUlengte)
tuple_gen_const(0,0, RijMiddenNieuw)
tuple_gen_const(0,0, KolMiddenNieuw)

for f:=0 to (KUlengte-1)/100 by 1
  tuple_gen_const(0,0, rijengroep)
  for g:=0 to (RUlengte-1)/10 by 1
    tuple_length(rijengroep, rijengroeplengte)
    yCo:=RijUniekSort[g*10]
    xCo:=KolUniekSort[f*100]
    test_region_point(midden, yCo, xCo, IsInside)
    if(IsInside==1)
      tuple_replace(rijengroep, rijengroeplengte, yCo, rijengroep)
    endif
  endfor
  tuple_length(RijMiddenNieuw, RijMiddenNieuwLengte)
  tuple_length(KolMiddenNieuw, KolMiddenNieuwLengte)
  if(rijengroeplengte!=0)
    tuple_mean(rijengroep, rijengroepMean)
    tuple_insert(RijMiddenNieuw, RijMiddenNieuwLengte, rijengroepMean, RijMiddenNieuw)
    tuple_insert(KolMiddenNieuw, KolMiddenNieuwLengte, xCo, KolMiddenNieuw)
  endif
endfor

tuple_length(KolMiddenNieuw, KolMiddenNieuwLengte)
tuple_length(RijMiddenNieuw, RijMiddenNieuwLengte)

```

**Figuur 51: procedure merkerpunten**

Het weergegeven algoritme (figuur 51) start met het beperken van het aantal pixelrijen en – kolommen van de middelste laserlijn tot enkel de unieke rijen en kolommen overblijven (commando *tuple\_uniq*). De verkregen lijst van de pixelrijen bevat zoals eerder vermeld een groot aantal identieke rijnummers. Door nu enkel de unieke rijnummers over te houden, bekomt men een veel kleinere lijst met enkel verschillende, unieke rijnummers. Hetzelfde geldt voor de lijst van de pixelkolommen. Dit alles maakt het eenvoudiger om later de merkpunten te bepalen en te creëren. Deze worden immers bepaald door na te gaan welke combinatie van unieke rijen en unieke kolommen een pixel genereert die vervat zit in de desbetreffende (middelste) laserlijn. Ook dit is weergegeven in het algoritme van figuur 51 en dit aan de hand van twee *for-lussen*.

De eerste *for-lus* kiest voor elke (lus-)cyclus een unieke kolom uit de eerder vernoemde lijst met unieke pixelkolommen van de middelste laserlijn. De tweede *for-lus* werkt echter met de unieke pixelrijen van diezelfde laserlijn. Deze lus gaat voor elke unieke pixelrij na of deze in combinatie met de gekozen pixelkolom uit de eerste *for-lus*, een pixel genereert die gelegen is in het gebied van de (middelste) laserlijn. Deze controle gebeurt aan de hand van het commando *test\_region\_point*. Die genereert voor een “juiste” combinatie een logische 1 en voor een “foute” een logische 0. Op basis daarvan kan men dus besluiten of men de gecontroleerde rij wil wegschrijven in een nieuwe lijst (indien logische 1). Dit wegschrijven gebeurt aan de hand van de *if-instructie* binnen de tweede *for-lus* (zie figuur 51). Indien er nu meerdere rijen zijn gevonden voor de desbetreffende kolom om een “juist” combinatie te vormen, bevat de nieuw



gegenereerde lijst bijgevolg meerdere plausibele unieke rijen. Om nu het aantal mogelijke merkerpunten te beperken, is er voor gekozen om een gemiddelde van de lijst met “goedgekeurde” rijen te nemen met behulp van het commando *tuple\_mean*. Het bekomen resultaat geeft de rij weer die in combinatie met de desbetreffende kolom (gekozen uit de eerste *for-lus*) de *coördinaten* van een merkerpunt gaat vormen. Deze *coördinaten* dienen bijgevolg weggeschreven te worden in twee lijsten, één voor de rij en één voor de kolom.

Indien men nu de vorige procedures uitvoert totdat de eerste *for-lus* (kolomkeuze) zijn einde bereikt, bekomt men dus twee lijsten waarin in de ene lijst de rijen en in de andere lijst de kolommen van de merkerpunten zijn gegeven. Deze lijsten zijn nu echter wel allemaal in correcte volgorde weggeschreven. Dit wil zeggen dat voor bijvoorbeeld het derde merkerpunt beide “coördinaten” zich op de derde plaats binnen de lijsten bevinden. Op deze manier kan er dus eenvoudiger gewerkt worden met de punten bij de hoogtemeting van de steenstrip.

### **Opmerking**

**In de programmacode van figuur 51 is duidelijk te zien dat men niet alle unieke rijen en kolommen controleert om een mogelijk merkerpunt te bekomen. Het aantal gecontroleerde rijen en kolommen zijn beiden aangepast met een bepaalde factor (factor 5 respectievelijk 100). Dit is slechts om de nodige rekenkracht te beperken en daarmee de snelheid van het programma te verhogen. Een nadeel kan zijn dat dit zorgt voor een iets lagere nauwkeurigheid. Testen geven echter aan dat deze verlaging in nauwkeurigheid klein en aanvaardbaar is ten opzichte van het inboeten in programmasnelheid en bijgevolg dus te verantwoorden is.**

Nu de merkerpunten zijn gekozen, kan men hiervan de afstand berekenen ten opzichte van de eerder geconstrueerde referentielijn. Zoals al gezegd, is deze afstand een maat voor de hoogte van de steenstrip op die bepaalde plaats. De procedure om de afstand van elk merkerpunt ten opzichte van de referentielijn te bekomen is weergegeven in figuur 52. In deze procedure is ook al de omvorming van de gevonden afstand naar de hoogte geïmplementeerd. Een aparte lijst neemt deze hoogtes op voor verdere gegevensverwerking.

```
for k:=0 to (KolMiddenNieuwLengte-1) by 1
  tuple_select (RijMiddenNieuw, k, PuntY)
  tuple_select (KolMiddenNieuw, k, PuntX)
  distance_pl (PuntY, PuntX, BeginrijLinks, BeginkolomLinks, EindrijRechts, EindkolomRechts, afstand)
  afstandMM:=afstand*pixelToMm
  hoogtePunt:=tan (laserHoek) *afstandMM
  tuple_insert (hoogtes, k, hoogtePunt, hoogtes)
endfor
```

***Figuur 52: procedure afstand merkerpunt tot referentielijn***

De verdere gegevensverwerking behelst het afleiden van een gemiddelde hoogte voor de gehele steenstrip en het analyseren van de verkregen hoogtes op spievormigheid van diezelfde steenstrip. Belangrijk is op te merken dat alle besproken procedures uit deze paragraaf als één cyclus gelden. Dit is dus de controle op één moment van het tijdsinterval waarin de steenstrip voorbij de lijnlaser passeert. Om de steenstrip volledig te controleren, moet deze controle dus gebeuren aan de hand van meerdere identieke cycli. Zo kan er uiteindelijk een algemene conclusie getrokken worden uit de analyse van de steenstrip en kan eveneens de gemiddelde

hoogte van de steenstrip berekend worden. De gemiddelde hoogte van de gehele steenstrip berekent men in twee stappen:

- Berekening van de gemiddelde hoogte op de plaats van de middelste laserlijn (van één cyclus);
- Gemiddelde berekenen van alle bekomen gemiddeldes gedurende de volledige hoogtemeting (meerdere cycli, dus op het einde van de algemene controle).

In figuur 53 is de procedure voor de berekening van de gemiddelde hoogte (groene kaders) van de steenstrip weergegeven evenals de analyse op de spievormigheid in lengterichting ervan (twee criteria, blauwe kader). De procedure houdt ook de hoogtes van het uiterst linkse en uiterst rechtse merkerpunt bij in aparte lijsten (rode kader). Deze zullen immers nodig zijn voor de controle op spievormigheid in zijdelingse richting (zie paragraaf 6.4.5.4).

```
tuple_mean(hoogtes,MeanHoogtes)
tuple_insert(GemHoogtes,cyclus,MeanHoogtes,GemHoogtes)
tuple_length(hoogtes,ElementenHoogtes)
tuple_select(hoogtes,0,ReferentieLinks)
tuple_select(hoogtes,ElementenHoogtes-1,ReferentieRechts)
tuple_insert(Left,cyclus,hoogtes[0],Left)
tuple_insert(Right,cyclus,hoogtes[ElementenHoogtes-1],Right)
cyclus:=cyclus+1
tuple_greater_equal_elem(hoogtes,17,Greatereq)
tuple_mean(Greatereq,equationMean)
if(equationMean>0.95 and equationMean<=1)
    GoedeStrip:=1
elseif(abs(ReferentieLinks-ReferentieRechts)<=4)
    GoedeStrip:=1
else
    GoedeStrip:=0
endif
if(-5<=MeanHoogtes and MeanHoogtes<=10)
    thirdTrigger:=1
elseif(GoedeStrip=0)
    secondTrigger:=1
endif
endif
tuple_mean(GemHoogtes,GemHoogteStrip)
```

**Figuur 53: procedure berekening gemiddelde hoogte**

De gemiddelde hoogte voor één cyclus wordt wederom berekend door middel van het commando *tuple\_mean*. Het resultaat hiervan schrijft men weg in een nieuwe lijst. Deze lijst wordt elke cyclus aangevuld met een nieuwe gemiddelde waarde voor de hoogte, resulterend uit de betreffende cyclus. Wanneer nu de volledige steenstrip voorbij de lijnlaser is gekomen en daarbij de laatste cyclus doorlopen is, bevat de lijst met gemiddelde hoogtes een resem waardes. De totale gemiddelde hoogte van de gehele steenstrip is dan het gemiddelde van deze waardes (*tuple\_mean*). Deze berekening gebeurt echter pas nadat de volledige hoogtemeting van de steenstrip is gebeurd. Dit wil dus zeggen dat het commando zich net na de *if-voorwaarde* voor drie zichtbare lijnen zie eerder bevindt (figuur 53).

### **6.4.5.3 Spievormigheid in lengterichting**

De analyse van de spievormigheid (in lengterichting) is gebaseerd op twee criteria en is zoals eerder vermeld, weergegeven in de blauwe kader in figuur 53 (paragraaf 6.4.5.2). Om aan de eerste voorwaarde te kunnen voldoen, wordt voor alle gevonden hoogtes van de verschillende merkpunten nagegaan of ze groter of gelijk zijn aan een bepaalde vooraf ingestelde waarde (bijvoorbeeld 17 mm. Dit is realiseerbaar met het commando *tuple\_greater\_equal\_elem*. Die geeft als resultaat eveneens een lijst terug met allemaal logische 1'en en 0'en . De logische 1 staat voor een hoogte die aan de voorwaarde heeft voldaan, voor de logische 0 is dit net omgekeerd. Het gemiddelde van deze lijst (een waarde tussen 0 en 1) geeft een indicatie voor het voldoen aan een bepaalde gewenste hoogte voor de steenstrip. Een hoge waarde van het gemiddelde wil zeggen dat de steenstrip over het algemeen voldoet aan de gewenste hoogtes, terwijl een lage waarde net het omgekeerde vertelt. In figuur 53 is de grenswaarde voor het gemiddelde van deze lijst op 0,95 gezet. Een waarde van 1 betekent dat er enkel steenstrippen goedgekeurd worden waarvan de opgemeten hoogtes altijd hoger zijn dan een opgegeven waarde. De reden waarom echter niet voor een instelwaarde 1 is gekozen, is omdat een steenstrip een zeer ruw profiel heeft en op een welbepaalde plaats toch ergens een hoogte kan hebben die lager is dan gewenst. Indien dit slechts een zeer klein gebied is, is het niet nodig deze steen af te keuren. Omwille van deze reden is er gekozen om de voorwaardefactor in te stellen op 0,95 in plaats van op 1.

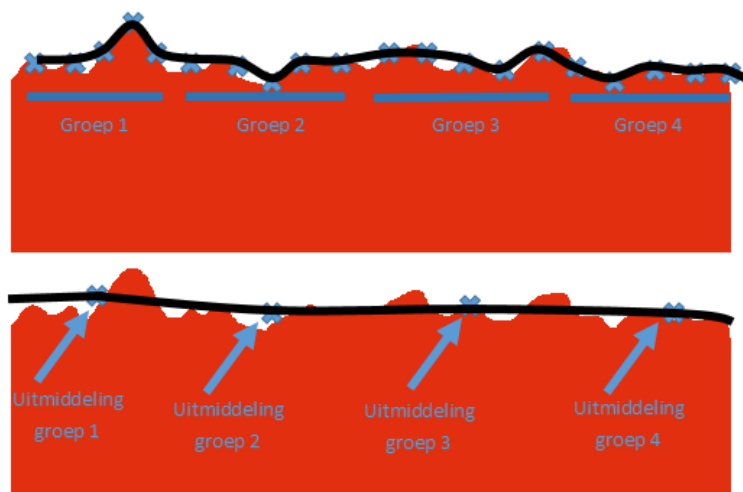
De tweede voorwaarde waaraan voldaan moet zijn om te slagen voor de analyse heeft ook rechtstreeks betrekking op de spievormigheid. Deze stelt namelijk dat het verschil in hoogte tussen het uiterst linkse en uiterst rechtse (merker-)punt een bepaalde waarde niet mag overschrijden. In dit geval is deze waarde experimenteel gekozen op 4 mm. Dit is ook zichtbaar in de programmacode van figuur 53. Indien ook aan deze voorwaarde is voldaan (en ook aan de eerste voorwaarde voor spievormigheid) kan gesteld worden dat de spievormigheid (in lengterichting) wordt goedgekeurd voor de betreffende steenstrip.

### **6.4.5.4 Zijwaartse spievormigheid**

Net zoals een controle op spievormigheid in de lengterichting, dient er ook een gelijkaardige controle te gebeuren in de zijwaartse richting (breedterichting) van de steenstrip. De controle op deze zijwaartse spievormigheid gebeurt aan de hand van de eerder bijgehouden waarden voor de hoogtes gemeten op de uiterst linkse en rechtse merkerpunten tijdens één cyclus (zie paragraaf 6.4.5.2). Deze hoogtes zijn zoals eerder gezegd, opgeslagen in aparte lijsten. Een zijwaartse spievormcontrole voor de steenstrip, die net volledig gepasseerd is, kan uitgevoerd worden door enkel de gegevens van die twee lijsten te verwerken op basis van een bepaald principe. Dit principe is gebaseerd op de volgende stappen:

- gegevens (hoogtemetingen) groeperen,
- bekomen groepen verwerken,
- verwerking analyseren op basis van drie criteria,
- zijwaartse spievormigheid goedkeuren of afkeuren.

De eerste stap in de spievormcontrole, is het groeperen van de gegevens. Dit wil zeggen dat het geheel aan gegevens, bekomen via de hoogtemetingen, een indeling in vijf groepen verkrijgt. Deze groepen vormen dus elk een lijst met als gegevens een aantal hoogtemetingen. Als tweede stap dienen deze “groepslijsten” verwerkt te worden tot bruikbare gegevens voor de zijwaarts spievormcontrole. Deze verwerking berekent het gemiddelde van elke groep en stockeert het resultaat in een nieuwe lijst. Het is op basis van deze lijst dat de controle tot stand komt. Het is namelijk zo dat door groepering van de gegevens en vervolgens het gemiddelde ervan te nemen, plaatselijke uitschieters (in hoogtemetingen) worden uitgemiddeld. Deze uitschieters zijn immers steeds mogelijk vanwege het ruwe oppervlak van de steenstrip, maar indien die slechts plaatselijk (zeer kleine oppervlakte) is, betekent dit niet dat de steenstrip per definitie dient verwijderd te worden. Deze steenstrippen voldoen namelijk nog steeds aan de kwaliteitseisen van het bedrijf waardoor het verwijderen van deze steenstrip zorgt voor nodeloze productverspilling. Een schematische weergave van het principe “grouperen en uitmiddelen” is weergegeven in figuur 54. De programmacode om deze procedure uit te voeren is weergegeven in figuur 55. Belangrijk is op te merken dat de voorgaande beschreven principes zowel voor de linkse zijde als de rechtse zijde van de steenstrip gelden, zoals ook weergegeven in de programmacode van figuur 55.



**Figuur 54: grouperen en uitmiddelen**

```

for i := 1 to 5 by 1
  for y:=((LeftLength-1)*(i-1)/5) to ((LeftLength-1)*i/5) by 1
    tuple_replace(GroupLinks,y-((LeftLength-1)*(i-1)/5),Left[y],GroupLinks)
  endfor
  tuple_mean(GroupLinks,LinksMean)
  tuple_insert(GroupsLinksMean,i-1,LinksMean,GroupsLinksMean)
  for z:=((RightLength-1)*(i-1)/5) to ((RightLength-1)*i/5) by 1
    tuple_replace(GroupRechts,z-((RightLength-1)*(i-1)/5),Right[z],GroupRechts)
  endfor
  tuple_mean(GroupRechts,RechtsMean)
  tuple_insert(GroupsRechtsMean,i-1,RechtsMean,GroupsRechtsMean)
endfor
tuple_length(GroupsLinksMean,lengthLgem)
tuple_length(GroupsRechtsMean,lengthRgem)
tuple_mean(Left,LeftMean)
tuple_mean(Right,RightMean)

```

**Figuur 55: procedure grouperen en uitmiddelen**

De volgende stap om tot een zijwaartse spievormcontrole te geraken, is het verwerken van de verkregen gegevens (uitmiddeling) en beoordelen op basis van drie criteria (allen toegepast in zijdelingse richting van de steenstrip):

- Hoogteverschil tussen eerste en laatste *uitgemiddelde* hoogte;
- Gemiddelde hoogte;
- Helling (spievormigheid).

De eerste twee criteria hebben vooral betrekking op de hoogte zelf en de toleranties hierop. Te grote hoogteverschillen mogen immers niet getolereerd worden. Het criterium in verband met de helling is echter specifiek toegespitst op de controle op spievormigheid. Hier gaat men elke *uitgemiddelde* hoogte vergelijken met de vorige en de volgende waarde in de lijst met uitgemiddelde hoogtes. Op deze manier kan een controle gebeuren naar de aanwezigheid van een helling en dus een controle op spievormigheid. De desbetreffende programmacode voor elk criterium is opnieuw terug te vinden in bijlage F met de subtitel "*Zijwaartse spievormcontrole*".

#### **6.4.6 Automatische reset bij foutdetectie**

De programmacode van de besproken beeldverwerking is ook uitgerust met een automatische *reset* bij een foutdetectie. Zo zal het programma gereset worden en terugkeren naar zijn begintoestand indien er een fout op de desbetreffende steenstrip is gedetecteerd. Een controle om na te gaan of het programma dient gereset te worden vanwege een fout is, bevindt zich op de volgende plaatsen in het programma:

- na hoekcontrole (vormcontrole en hoekcontrole is dus reeds gebeurd);
- na spievormcontrole in lengterichting.

De voornaamste reden voor implementatie van dergelijke *programmaresets* is de verbetering van de systeemsnelheid. Indien een steenstrip in een vroeg stadium van de globale kwaliteitscontrole wordt afgekeurd, zal het systeem immers resetten zonder de controles uit te voeren die in een laat stadium voorkomen. Dit bespaart op rekenkracht en bijgevolg op verwerkingstijd wat resulteert in een hogere systeemsnelheid van het visiesysteem. De implementatie van de automatische *reset* kan eveneens gevonden worden in de complete programmacode in bijlage F.



## 7 Gebruikersinterface in Visual Studio

De principiële werking van de beeldverwerking met zijn programmacode is in het vorige hoofdstuk volledig beschreven. In dit hoofdstuk zal beschreven worden hoe de gebruikersinterface voor het hanteren van het geschreven beeldverwerkingsprogramma tot stand is gekomen. Deze interface bevat verschillende elementen om op een eenvoudige manier programmaparameters, zoals *thresholdgrenzen*, aan te passen zonder effectief in de code zelf dingen te veranderen. De realisatie van deze interface is gebeurd door export van de programmacode (HALCON) naar de ontwikkelomgeving van Microsoft *Visual Studio*.

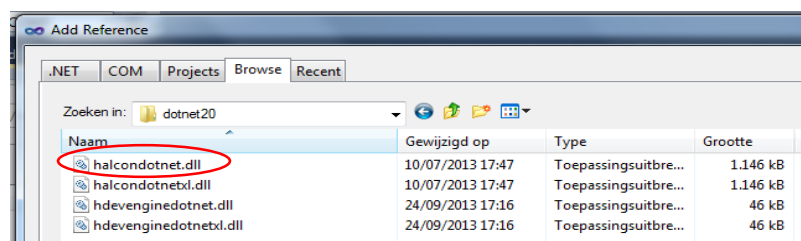
Visual Studio werkt met verschillende programmeertalen [19]:

- *Visual Basic.Net*,
- *C#*,
- *F#*,
- *Visual C++*.

Er is gekozen om de interface te maken op basis van Visual Basic.Net. Dit vanwege een bredere kennis van deze programmeertaal, welke tevens de standaardtaal van Visual Studio is. Het programma bestaat uit twee grote delen, namelijk het ontwerp van het scherm en de code om de knoppen, vensters, tekstboxen, ... aan te sturen. Het scherm vertegenwoordigt het visuele gedeelte van Visual Studio en bestaat uit verschillende hulpelementen zoals vensters, tekst vakjes, knoppen en schuifbalken die instaan voor het weergeven van het resultaat of het aanpassen van bepaalde parameters.

### 7.1 Programmacode exporteren naar VB.Net

Om de programmacode uit HALCON te kunnen gebruiken in Visual Studio, is het nodig om de “bibliotheek” van HALCON te koppelen aan Visual Studio. In deze bibliotheek zitten alle functies die beeldverwerking met HALCON mogelijk maken. Het toevoegen ervan in Visual Studio is mogelijk door bij “*My Projects*” een nieuwe referentie aan te maken, zoals figuur 56 aantoont. Door selectie van *halcondotnet.dll* en deze toe te voegen, kunnen alle functies van HALCON gebruikt worden in Visual Studio.



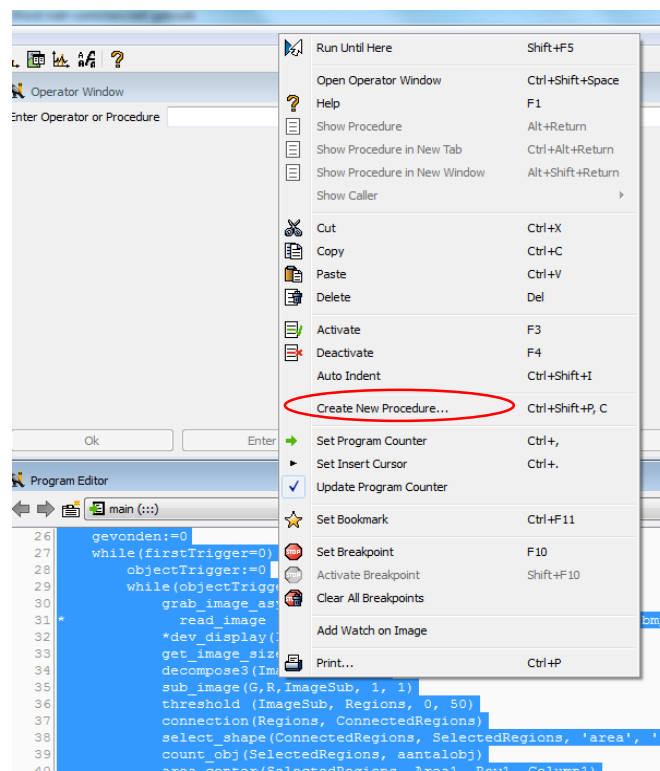
Figuur 56: referentie toevoegen

Het omzetten van de code vanuit HALCON naar Visual Studio gebeurt aan de hand van de volgende stappen:

1. Programmacode in HALCON opdelen en transformeren in een procedure;
2. Export van procedure naar de programmeertaal *Visual Basic.Net (VB.Net)*;
3. Code openen in Visual Studio;
4. Nodige code filteren en elke variabele declareren.

### **Stap 1: Maken van een procedure**

Om een procedure te maken in HALCON, is het nodig de gewenste code te selecteren en vervolgens bij het rechtermuisknopmenu te kiezen voor **Create New Procedure** zoals weergegeven in figuur 57. Het is ook mogelijk om de volledige code direct te exporteren, maar door procedures te maken van de belangrijkste stappen blijft de code overzichtelijk.

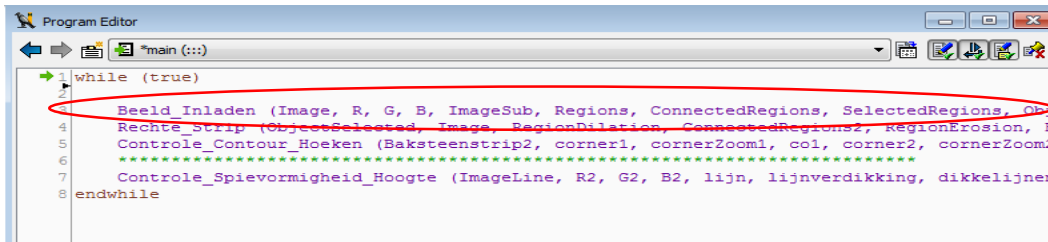


**Figuur 57: nieuwe procedure aanmaken**

Wanneer de volledige code is omgezet in procedures, is het mogelijk om ze één voor één om te zetten naar een andere programmeertaal (zie stap 2). Figuur 58 geeft nog eens de verschillende gecreëerde procedures weer, waarbij de volledige programmacode bestaat uit de volgende procedures:

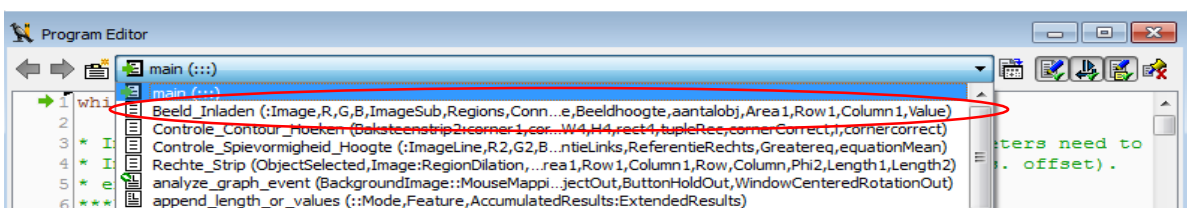
- *Beeld\_Inladen*,
- *Rechte\_Strip*,
- *Controle\_Contour\_Hoeken*,
- *Controle\_Spievormigheid\_Hoogte*.





**Figuur 58: de verschillende procedures**

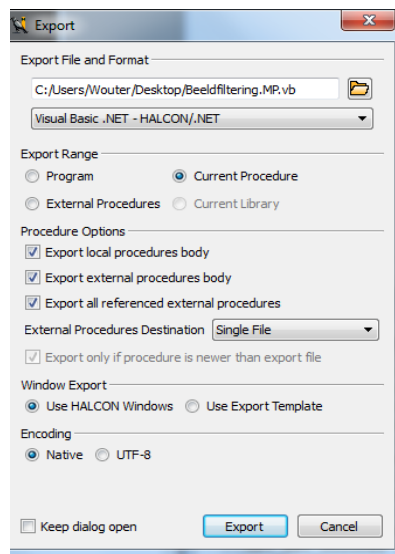
Om nu naar een aparte procedure te gaan moet men het venstertje “main” veranderen naar de desbetreffende procedure, zoals weergegeven in figuur 59. Als voorbeeld is hier de procedure “Beeld\_Inladen” gekozen.



**Figuur 59: keuze procedure**

## **Stap 2: Exporteren van procedure naar VB.Net**

Zoals eerder vermeld, is er gekozen voor Visual Basic.Net. Figuur 60 toont de instellingen voor het exporteren van een procedure. De *Export Range* moet op **Current Procedure** staan, zodat enkel de geopende procedure geëxporteerd wordt naar een gewenste plaats. Dit zorgt voor een beter overzicht in het programma. Door te klikken op *export*, wordt de code geëxporteerd naar de aangegeven bestandsplaats, in dit geval het bureaublad (“Desktop”).



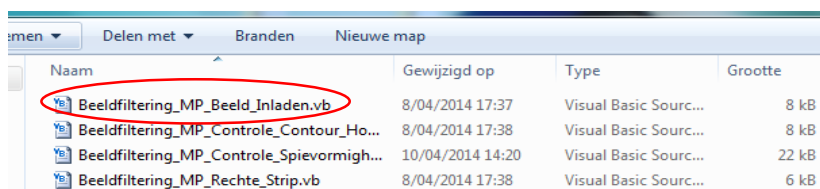
**Figuur 60: exporteren van procedure**

### Stap3: Code openen in Visual Studio

Figuur 61 toont een lijst met bestanden (.vb) van de geëxporteerde procedures. Wanneer het geëxporteerde bestand geopend wordt, verschijnt de code in Visual Studio zoals getoond in figuur 62. Visual Studio maakt van de code een module met dezelfde parameters als in HALCON. De code is daarom eenvoudig te vergelijken met die van HALCON. Om de code te laten werken is de regel *Imports HalconDotNet* noodzakelijk. Zonder deze regel zal Visual Studio de eerder vernoemde bibliotheek van HALCON niet kunnen vinden.

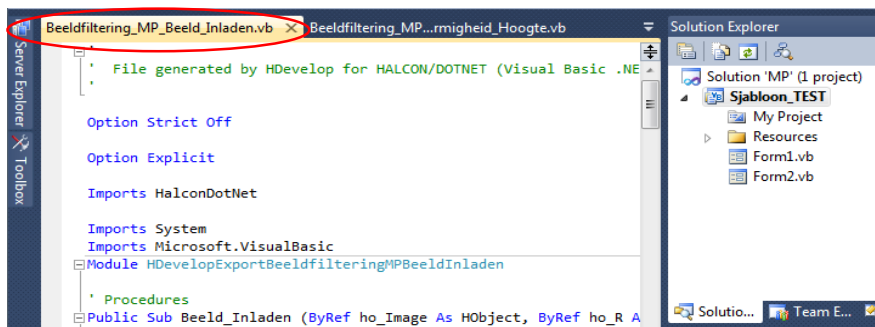
### Opmerking

Vaak staat de *solution platform* standaard ingesteld op *“x86”*, maar om het programma te laten werken moet deze omgezet worden naar *“any CPU”*.



Naam	Gewijzigd op	Type	Grootte
Beeldfiltering_MP_Beeld_Inladen.vb	8/04/2014 17:37	Visual Basic Sourc...	8 kB
Beeldfiltering_MP_Controlle_Contour_Ho...	8/04/2014 17:38	Visual Basic Sourc...	8 kB
Beeldfiltering_MP_Controlle_Spievormigh...	10/04/2014 14:20	Visual Basic Sourc...	22 kB
Beeldfiltering_MP_Rechte_Strip.vb	8/04/2014 17:38	Visual Basic Sourc...	6 kB

*Figuur 61: lijst van exportbestanden van de procedures*



*Figuur 62: geëxporteerd bestand geopend in Visual Studio*

### Stap 4: Declareren van variabelen en filteren van de code

HALCON is een programma dat alle geschreven code onmiddellijk uitvoert op het voorziene programmascherm. Bij Visual Studio moet men echter expliciet vermelden welke figuren/beelden men wil laten zien in het scherm. Daarom is het nodig om in de aangemaakte module in VS (Visual Studio) te zoeken naar de code die enkel noodzakelijk is voor de toepassing. Allereerst is het noodzakelijk het *“HALCON”* venster *“HWindowControl1”* volledig leeg te maken. Dit gebeurt met de code: ***HWindowControl1.HalconWindow.ClearWindow()***

Door deze instructie als eerste te zetten in de programmacode in Visual Studio, zal het venster *HWindowControl1* steeds worden leeggemaakt om nadien beelden te kunnen weer geven van nieuwe baksteenstrippen. Vervolgens is het de bedoeling om een beeld te laten zien in het leeggemaakte scherm. Dit gebeurt dan weer met de code: ***ho\_image.DispObj(HWindowControl1.HalconWindow)***

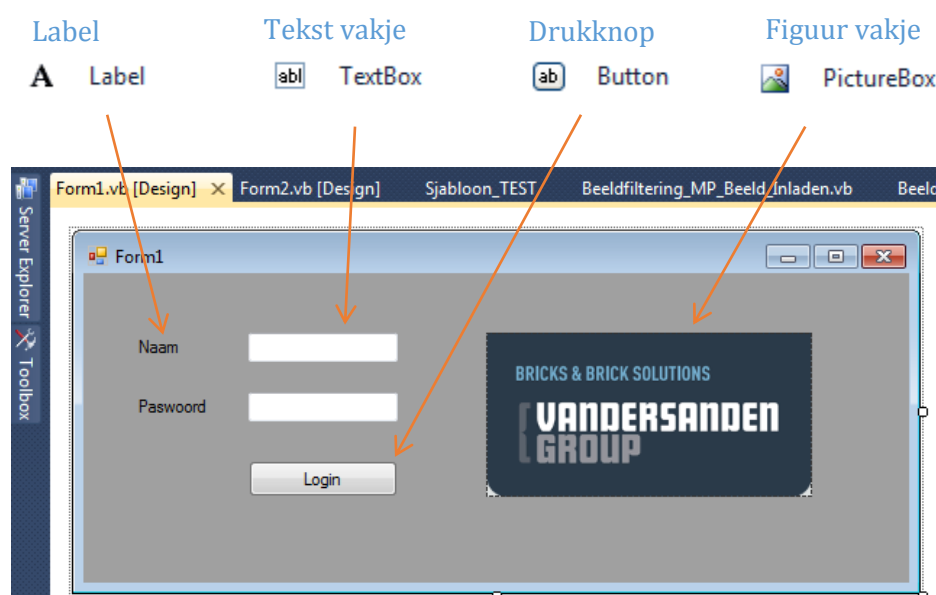
De figuur met de naam *ho\_image* verschijnt in het scherm. Hierbij is het op te merken dat Visual Studio de variabele *ho\_image* nog niet kent. Om deze te declareren moet men voor de functie de volgende code gebruiken: ***Dim ho\_image as HObject = nothing***. Deze code wil zeggen dat *ho\_image* gedeclareerd wordt als een *HALCON-object*. Wanneer er een variabele in de code staat die een getal of een reeks van getallen vertegenwoordigt, verandert de declaratie naar *HTuple* in plaats van *HObject*. Het is nodig om alle gebruikte variabelen in HALCON opnieuw te declareren in Visual Studio, omdat deze anders niet weet wat de variabele precies voorstelt.

## 7.2 Het ontwerp

Deze paragraaf beschrijft hoe de interface visueel oogt. Deze interface bestaat uit drie vensters (forms) met alle gebruikte functies. Het eerste venster is een login scherm, zodat enkel bevoegden het programma mogen gebruiken. Vervolgens staat het hoofdprogramma (kwaliteitscontrole) in het tweede scherm. Dit scherm laat de beelden van de camera zien en geeft weer of de strippen goed of slecht zijn. Het tweede scherm geeft ook de afmetingen van de strip weer. Een database in *SQL Server* verzamelt deze afmetingen (zie hoofdstuk 9). Ten slotte staat het derde en laatste scherm in voor de instellingen van programmaparameters als *thresholdgrenzen*, bevestigingshoek van de lijnlaser... .

### 7.2.1 Het inlogscherf

Zoals eerder vermeld, bezit Visual Studio verschillende hulptools zoals knoppen, tekstboxen, schuifbalken en figuurboxen. Alvorens effectief te kunnen programmeren, is het noodzakelijk een interfaceontwerp te maken met behulp van deze elementen. Figuur 63 geeft het inlogscherf van de interface weer, met aanduiding van de gebruikte tools.



Figuur 63: eerste scherm (loginscherf) in Visual Studio

## PictureBox [20]

Met behulp van een *PictureBox* is het mogelijk om een figuur of foto weer te geven. Deze functie is gemakkelijk voor het tonen van figuren met belangrijke gegevens, maar ook om andere informatie weer te geven. In sommige gevallen kan een figuur namelijk meer vertellen dan woorden. In het inlogscherf is de *PictureBox* gebruikt om het logo van Vandersanden weer te geven. Er zijn twee manieren om een foto te tonen. Een eerste methode is door in het menu **Properties** de figuur te zoeken bij **Image** en deze te selecteren. De figuur verschijnt dan meteen in het venster (inlogscherf). De tweede methode is door code te schrijven die de figuur oproept in het desbetreffende venster.

Een nuttige functie van de *PictureBox* is *AutoSize*. Deze bepaalt of het desbetreffende venster zich zal moeten aanpassen zodat het beeld volledig zichtbaar is. Een andere handige functie is *SizeMode*. Deze functie kan het beeld rekken, in het midden zetten of zoomen in het venster. Voordat het beeld wordt gebruikt, is het aangeraden om deze toe te voegen als een *Resource (bron)* aan het project. Daardoor kan het beeld zo vaak als nodig hergebruikt worden. Zo is het mogelijk om het beeld op verschillende locaties weer te geven.

## TextBox

In de *TextBox* is het mogelijk om tekst te typen wanneer het programma in *Run-mode* staat (werkend is), maar het kan ook gebruikt worden om waardes of tekst te laten verschijnen door middel van instructies (programmacode). Een *TextBox* bezit een aantal parameters die door middel van code of via het menu *Properties* (kunnen) worden aangepast. Enkele voorbeelden van deze parameters zijn:

- het lettertype,
- de achtergrondkleur van het tekstvak,
- de dikte van de tekst,
- de stijl van de tekst,
- het maximum aantal woorden.

## Label

*Labels* worden gebruikt om dezelfde redenen als tekstboxen, met dit verschil dat de inhoud van een label niet kan verandert worden door op die plaats iets te typen op het moment dat het programma bezig is. Dit is zoals eerder gezegd wel mogelijk bij tekstboxen. Daardoor is het nodig om de tekst op voorhand in de label te zetten of de tekst te veranderen aan de hand van een procedure (code).

## Button

Een *button* is een (druk)knop die een bepaalde actie uitvoert wanneer er op gedrukt wordt. Deze actie verschilt naargelang de toepassing. Zo kan een drukknop bijvoorbeeld gebruikt worden om van het ene scherm naar het andere te gaan, maar ook voor een bepaalde actie of controle uit te voeren.

## 7.2.2 Resultatenschermb

Figuur 64 toont het ontwerp van een tweede scherm, het resultatenschermb. Dit scherm geeft de belangrijkste informatie van de strippen weer, tijdens en na de kwaliteitscontrole. Deze informatie is onderverdeeld in twee grote delen, namelijk de weergave van de stripafmetingen en de resultaten van de steenstripcontrole. Het eerste deel visualiseert de lengte, de breedte en de hoogte van de strip in de bijhorende *textboxes*. Het tweede deel laat zien of de gecontroleerde strip voldoet aan de reeds genoemde eisen op basis van vorm, hoekvormgeving en spievormigheid. Ten slotte is het ook mogelijk om naar het instellingenschermb te gaan door te drukken op de knop "Instellingen".

### HWindowControl

In het venster "HWindowControl" komen de belangrijkste beelden tevoorschijn van de baksteenstrip, zoals het eerste beeld dat de camera maakt en het beeld waarop de strip gedraaid is (positiecompensatie). Beelden die ontstaan na bepaalde beeldverwerkingsinstructies zoals een *threshold*, *decompose*,... hebben geen visuele meerwaarde en worden bijgevolg niet in het venster weergegeven.

Door nu enkel de gewenste beelden te laten zien, zal niet alleen het programma sneller werken, maar oogt het venster ook veel overzichtelijker dan wanneer alle beelden worden getoond op een korte tijdspanne. Het besproken venster is gekoppeld aan de camera van het visiesysteem. De code om verschillende beelden te laten zien staat vermeld in paragraaf 7.3.2.



**Figuur 64:** tweede scherm (resultatenschermb) in Visual Studio

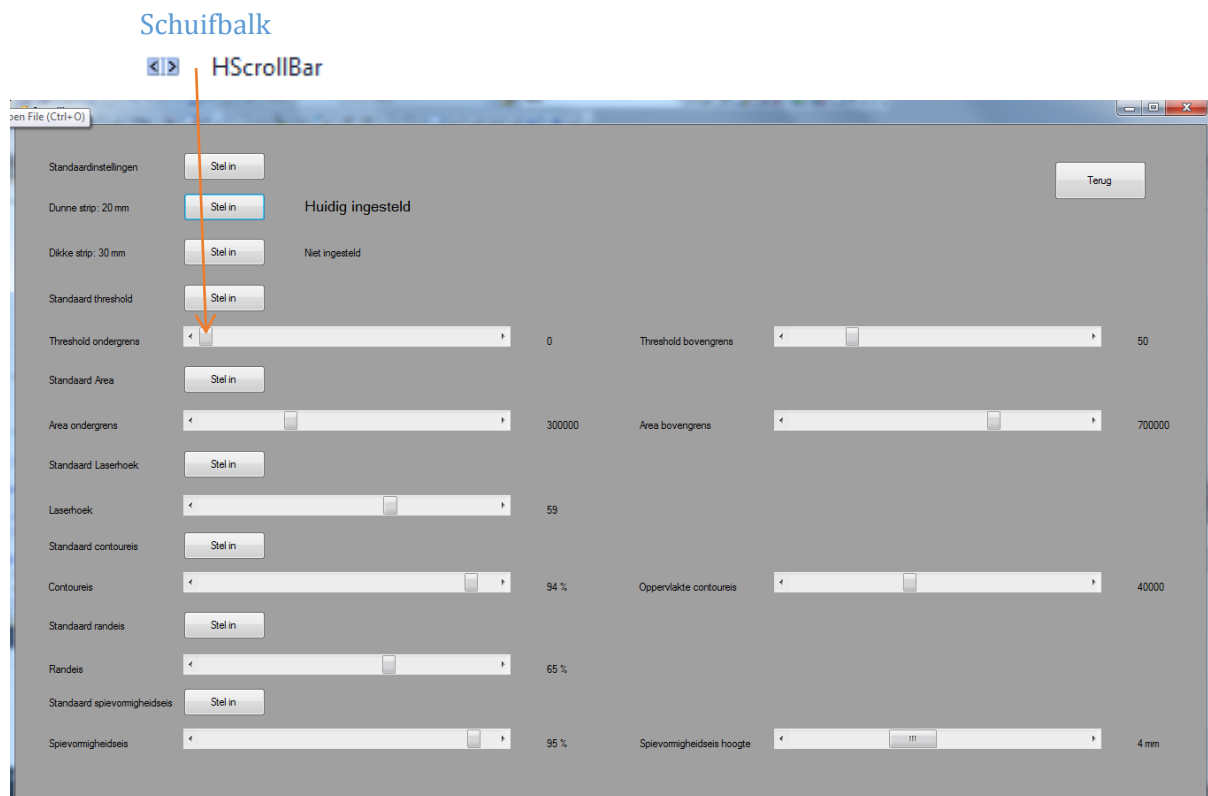
## **Timer**

*Timers* hebben de functie om op bepaalde tijdstippen een actie uit te voeren. *Timers* kunnen zowel op- als aftellen. Bij de meeste aftellende *timers* gebeurt er een actie wanneer de tijd op nul komt. Bij optellende *timers* is het mogelijk om op regelmatige tijdstippen acties uit te voeren. De *timer* kan ook terug op de oorspronkelijke beginwaarde worden gezet om zo opnieuw te kunnen gebruiken. Naast dit gegeven heeft de *timer* nog een belangrijke eigenschap die gebruikt wordt in deze masterproef. Een *timer* zorgt er namelijk voor dat men op verschillende knoppen kan drukken terwijl het programma in de *timer* loopt. Zo kan de kwaliteitscontrole op alle mogelijke tijdstippen onderbroken en/of terug opgezet worden. Om bijvoorbeeld de *threshold* te veranderen, is het nodig om het programma eerst te onderbreken en om die reden is er dan ook gebruik gemaakt van een *timer*. Het stoppen van een *timer* gebeurt met de commando `timer.stop()`. Door dit commando aan de *stop*-knop toe te kennen, is het mogelijk het programma te onderbreken. Anderzijds door de code `timer.start()` toe te kennen aan de *RUN*-knop, kan het programma opnieuw gestart worden (vanaf de plaats van onderbreking). Door het drukken op de *RUN*-knop, zal de *timer* en daarmee ook het programma blijven doorlopen tot er op de *Stop*-knop wordt gedrukt.

Wanneer de code voor de kwaliteitscontrole direct gekoppeld is aan een knop, zonder gebruik te maken van een *timer*, is het echter niet mogelijk om het programma te onderbreken door middel van een andere knop. Veronderstel namelijk eens dat een lus is geprogrammeerd in de *Start*-knop die steeds de kwaliteitscontrole uitvoert. Door op de knop te drukken, zal vanwege de luswerking het proces continu herhaald worden. Het programma zal dan oneindig lang dezelfde lus uitvoeren, zonder te kijken naar andere functies, knoppen, ... op het scherm (bv. *threshold*-instellingen). Doordat het programma blijft “hangen” in de lus, is het bijgevolg niet mogelijk om op de *Stop*-knop te drukken. Dit probleem geeft duidelijk het belang van een timer aan, waardoor implementatie ervan in deze masterproef dan ook een noodzaak was.

### **7.2.3 Instellingenschermb**

Het laatste scherm van de gebruikersinterface geeft de instellingen weer voor verschillende instelbare parameters. In dit scherm is er één nieuw element ten opzichte van de andere schermen, namelijk de schuifbalk. Figuur 65 toont het ontwerp van het betreffende instellingenschermb.



**Figuur 65: derde scherm (instellingenscherf) in Visual Studio**

## **HScrollBar** [21]

De schuifbalk *HScrollBar* is een element waarmee een waarde kan worden gekozen die gelegen is tussen een bepaalde onder- en bovengrens. Deze grenswaarden zijn afhankelijk van de toepassing en worden ingesteld bij *Minimum* en *Maximum* bij de *Properties* (eigenschappen) van de schuifbalk. De uiterst linkse en rechtse positie van het schuifblokje geven respectievelijk de onderste en de bovenste grens weer. De waarde van het schuifblokje kan op drie manieren aangepast worden naar de gewenste waarde (binnen de limieten). Een eerste methode is door het klikken op de schuifbalk zodat het blokje een vaste stap voor- of achteruit verplaatst. Deze stap is bekend onder de naam *LargeChange* in *Visual Studio* en kan ingesteld worden bij de *Properties* van de schuifbalk. Wanneer er tussen de linkse pijl en het schuifblokje van de schuifbalk wordt gedrukt, zal het blokje een verplaatsing (stap) naar links ondergaan.. Dit is analoog voor een verplaatsing naar de rechterkant. De tweede methode maakt gebruik van de pijlen aan het uiteinde van de schuifbalk. Wanneer er op de rechtse pijl gedrukt wordt, zal het schuifblokje een stap naar rechts verplaatsen. De grootte van deze stap is afhankelijk van de parameter *SmallChange* en is ook instelbaar bij *Properties*. Analoog zal het blokje dezelfde stap naar links verplaatsen wanneer er op de linkse pijl wordt gedrukt. De laatste methode maakt gebruik van het schuifblokje zelf. Door het blokje aan te klikken en vervolgens te slepen naar een gewenste positie in de schuifbalk, is het mogelijk om elke waarde tussen de limieten te bereiken.

In het instellingenscherf is het tevens mogelijk om de onder- en bovengrens van zowel de *threshold* als de *objectarea* (een maat voor de objectgrootte waarop het programma objecten in het beeld selecteert) te veranderen, alsook de waarde van de laserhoek. Het veranderen van deze waarden gebeurt eveneens door middel van schuifbalken.

Elke parameter heeft natuurlijk ook een standaardinstelling. Zo zullen de *threshold*-grenzen standaard ingesteld zijn op 0 (ondergrens) en 50 (bovengrens). Door op de knop “*Stel in*” te drukken met het label “*Standaard threshold*”, worden de grenzen aangepast aan de standaardinstelling. Op dezelfde manier kunnen ook de standaardwaarden van de *objectarea* en de laserhoek ingesteld worden. Ten slotte is er ook een knop die alle instellingen terugzet naar de standaardinstellingen, namelijk de “*Stel in*”-knop met het label “*Standaardinstellingen*”. Er is bewust gekozen om de criteria die handelen over de kwaliteit van de strippen ook op te nemen bij de instellingen. Zo kan Vandersanden zelf bepalen hoe streng ze willen zijn op gebied van de kwaliteit van de strippen.

## 7.3 De code

In deze paragraaf staat de verklaring van de gebruikte code in Visual Studio. De volledige code is terug te vinden in bijlage G, H en I voor respectievelijk *form 1*, *2* en *3* van het globale programma. Het grootste deel van de code bestaat uit de geëxporteerde procedures vanuit HALCON. Meer uitleg over deze code is te vinden in paragraaf 6. In deze paragraaf staat uitgelegd hoe de knoppen, tekst vakjes, timers, ... werken.

Om te beginnen is het belangrijk dat alle variabelen gedeclareerd zijn in VS (Visual Studio). Wanneer dit niet het geval is, zal VS een foutmelding geven bij het starten van het programma. Meer uitleg over de declaratie is te vinden bij stap 4 van paragraaf 7.1.

### 7.3.1 Code inlogscherf

De volledige code van het inlogscherf is terug te vinden in figuur 66. Wanneer er op de drukknop “login” wordt gedrukt (zie figuur 63) en de juiste gegevens zijn ingevuld in beide tekstvakken, zal het resultatenscherf geopend worden.



```

Public Class Login_Scherm

    Public i As Integer = 1

    Public Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LoginKnop.Click

        If Paswoord_Invoer.Text = "VDSG" Then
            If Naam_Invoer.Text = "VDS" Then

                Instellingen_Scherm.ThresholdOndergrens_Schuifbalk.Value = 0
                Instellingen_Scherm.ThresholdBovengrens_Schuifbalk.Value = 50
                Instellingen_Scherm.ThresholdOndergrensWaarde = Instellingen_Scherm.ThresholdOndergrens_Schuifbalk.Value
                Instellingen_Scherm.ThresholdBovengrensWaarde = Instellingen_Scherm.ThresholdBovengrens_Schuifbalk.Value
                Instellingen_Scherm.Threshold_OnderGrens.Text = Instellingen_Scherm.ThresholdOndergrensWaarde
                Instellingen_Scherm.Threshold_BovenGrens.Text = Instellingen_Scherm.ThresholdBovengrensWaarde

                Instellingen_Scherm.AreaOndergrens_Schuifbalk.Value = 300000
                Instellingen_Scherm.AreaBovengrens_Schuifbalk.Value = 700000
                Instellingen_Scherm.AreaOndergrensWaarde = Instellingen_Scherm.AreaOndergrens_Schuifbalk.Value
                Instellingen_Scherm.AreaBovengrensWaarde = Instellingen_Scherm.AreaBovengrens_Schuifbalk.Value
                Instellingen_Scherm.Area_OnderGrens.Text = Instellingen_Scherm.AreaOndergrensWaarde
                Instellingen_Scherm.Area_BovenGrens.Text = Instellingen_Scherm.AreaBovengrensWaarde

                Instellingen_Scherm.DunneStripIngesteld.ForeColor = Color.Black
                Instellingen_Scherm.DunneStripIngesteld.Font = New Font("Microsoft Sans Serif", 12.25, FontStyle.Regular)
                Instellingen_Scherm.DunneStripIngesteld.Text = "Huidig ingesteld"

                Instellingen_Scherm.DikkeStripIngesteld.ForeColor = Color.Black
                Instellingen_Scherm.DikkeStripIngesteld.Text = "Niet ingesteld"

                Instellingen_Scherm.LaserHoek_Schuifbalk.Value = 59
                Instellingen_Scherm.LaserHoekWaarde = Instellingen_Scherm.LaserHoek_Schuifbalk.Value
                Instellingen_Scherm.LaserHoek.Text = Instellingen_Scherm.LaserHoekWaarde

                Instellingen_Scherm.Contoureis_Schuifbalk.Value = 94
                Instellingen_Scherm.ContoureisWaarde = Instellingen_Scherm.Contoureis_Schuifbalk.Value
                Instellingen_Scherm.Contoureis.Text = Instellingen_Scherm.ContoureisWaarde & " %"

                Instellingen_Scherm.Randeis_Schuifbalk.Value = 65
                Instellingen_Scherm.RandeisWaarde = Instellingen_Scherm.Randeis_Schuifbalk.Value
                Instellingen_Scherm.Randeis.Text = Instellingen_Scherm.RandeisWaarde & " %"

                Instellingen_Scherm.Spievormigheidseis_Schuifbalk.Value = 95
                Instellingen_Scherm.SpievormigheidseisWaarde = Instellingen_Scherm.Spievormigheidseis_Schuifbalk.Value
                Instellingen_Scherm.Spievormigheidseis.Text = Instellingen_Scherm.SpievormigheidseisWaarde & " %"

                Instellingen_Scherm.vormEisOpp_Schuifbalk.Value = 40000
                Instellingen_Scherm.vormEisOppWaarde = Instellingen_Scherm.vormEisOpp_Schuifbalk.Value
                Instellingen_Scherm.vormEisOpp.Text = Instellingen_Scherm.vormEisOppWaarde

                Instellingen_Scherm.schaallengtepixel = 1056.0

                VisieControle_Scherm.Show()
                Me.Hide()

            Else
                MsgBox("Error! Foute login gegevens", MsgBoxStyle.Critical)
            End If
        Else
            MsgBox("Error! Foute login gegevens", MsgBoxStyle.Critical)
        End If

    End Sub
End Class

```

**Figuur 66: code van het loginscherm**

## **Knoppen toevoegen**

Het toevoegen van een knop gebeurt door het slepen van het icoontje “Button” (in de toolbox) naar het desbetreffende programmascherm. Bij de eigenschappen (*Properties*) van de knop is het mogelijk de naam en de tekst van de knop te veranderen. Zo kan de naam van de knop gewijzigd worden naar “LoginKnop” en de tekst naar “Login”, zoals werd weergegeven in figuur 63. Wanneer er dubbelgeklikt wordt op de pas aangemaakte knop, genereert *Visual Studio* automatisch enkele regels code in een nieuw venster. Figuur 67 toont de gegenereerde code.

```
Public Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LoginKnop.Click  
  
End Sub
```

**Figuur 67: standaard code van de login knop**

Tussen “Public Sub” en “End Sub” (zie figuur 66) is het mogelijk code te schrijven voor wat er moet gebeuren wanneer er op de knop *Login* gedrukt wordt. Het doel is om het volgende scherm (het resultatenscherm) te openen wanneer de juiste naam en paswoord zijn ingegeven in de voorziene vakken. Dit kan eenvoudig gebeuren door het gebruik van twee *if*-lussen. De eerste *if*-lus in figuur 66 kijkt naar de inhoud van het tekstvak “Paswoord\_Invoer”, Terwijl de tweede *if*-lus de inhoud van het tekstvak “Naam\_Invoer” controleert. *Naam\_Invoer* is het tekstvak waarbij het label “Naam” staat, *Paswoord\_Invoer* hoort bij het label “Paswoord”.

De eerste *if*-lus vergelijkt zoals eerder gezegd de inhoud (de tekst) van het tekstvak “Paswoord\_Invoer” met de tekst “VDSG”. Analoog vergelijkt de tweede *if*-lus de inhoud van het tekstvak “Naam\_Invoer” met de tekst “VDS”. Indien de tekst in minstens één van de twee tekstvakken niet klopt, voert *Visual Studio* de code uit van de bijbehorende *else*-lus. Deze lus maakt een nieuw venster met als mededeling: “Error! Foute login gegevens”. Wanneer beide tekstvakjes correct zijn ingevuld, wordt de code van de tweede *if*-lus uitgevoerd. Met deze code worden in het derde scherm (instellingscherm) “Instellingen\_Scherm” de eerder genoemde standaardinstellingen ingesteld. Doordat er steeds naar “Instellingen\_Scherm” wordt verwezen, begint de code eveneens steeds met “Instellingen\_Scherm”. Na de punt staat de aan te passen parameter, meer hierover is te vinden in de paragraaf 7.3.3. Ten slotte opent de code *VisieControle\_Scherm.show()* het tweede scherm (resultatenscherm) en *Me.hide()* sluit het huidige venster.

### **7.3.2 Code resultatenscherm**

Het resultatenscherm (zie figuur 64) bestaat uit de geëxporteerde code vanuit HALCON. Deze code staat zoals beschreven in paragraaf 7.2.2 onder “Timer1”. Dit is het eigenlijke hoofdprogramma. Figuren 68 en 69 laten respectievelijk zien hoe de *timer* gestart en gestopt wordt.

```
Private Sub RUNknop_Click_1(sender As System.Object, e As System.EventArgs) Handles RUNknop.Click  
    Timer1.Start()  
End Sub
```

**Figuur 68: starten van de timer**

```
Private Sub STOPknop_Click_1(sender As System.Object, e As System.EventArgs) Handles STOPknop.Click
    Timer1.Stop()
End Sub
```

**Figuur 69: stoppen van de timer**

Het effectieve hoofdprogramma begint met code voor het leegmaken van alle tekstvakken en het scherm *HWindowControl1*, zoals getoond in figuur 70.

```
HWindowControl1.HalconWindow.ClearWindow()
LengteStrip_Invoer.Text = ""
BreedteStrip_Invoer.Text = ""
Vorm_Invoer.Text = ""
Rand_Invoer.Text = ""
Spievorming_Invoer.Text = ""
HoogteStrip_Invoer.Text = ""
```

**Figuur 70: het leegmaken van de tekst vakken en het venster**

Vervolgens dient het (eerste) genomen camerabeeld weergegeven te worden in het daarvoor voorziene venster. Deze procedure is weergegeven in figuur 71.

```
HOperatorSet.GrabImageAsync(ho_image, hv_AcqHandle, New HTuple(-1))
HWindowControl1.HalconWindow.ClearWindow()
ho_image.DispObj(HWindowControl1.HalconWindow)
```

**Figuur 71: het eerste beeld van de camera weergeven**

Als volgende stap wordt de geëxporteerde code van HALCON uitgevoerd. Wanneer het programma de lengte en de breedte van de strip heeft berekend, zal de code uit figuur 72 ervoor zorgen dat deze afmetingen ingevuld worden in de respectievelijke tekstvakken *"LengteStrip\_Invoer"* en *"BreedteStrip\_Invoer"*. Op analoge wijze wordt de hoogte van de strip toegekend aan het tekstvak *"HoogteStrip\_Invoer"*.

```
stripbreedte = hv_stripbreedte
striplengte = hv_striplengte

LengteStrip_Invoer.Text = striplengte
BreedteStrip_Invoer.Text = stripbreedte
```

**Figuur 72: het weergeven van de lengte en breedte van de strip**

De kwaliteitscontroles (goed- of afkeuren) worden op soortgelijke manier weergegeven. Het grote verschil is dat er geen *integer*, maar een *string* wordt opgegeven in het tekstvakje. Wanneer bijvoorbeeld de vorm goedgekeurd is, wordt in het tekstvak *"Vorm\_Invoer"* de tekst *"Goedgekeurd"* in een groene kleur gezet. De code die hiervoor verantwoordelijk is, is getoond in figuur 73. Anderzijds wordt een strip met een slechte vorm visueel zichtbaar gemaakt door de rode tekst *"afgekeurd"*. Deze code is voor de volledigheid weergegeven in figuur 74.

```
hv_contourInkeping = New HTuple(1)
Vorm_Invoer.ForeColor = Color.Green
Vorm_Invoer.Text = "Goedgekeurd"
```

**Figuur 73: het weergeven van een goedgekeurde contour inkeping**

```
hv_contourInkeping = New HTuple(0)
Vorm_Invoer.ForeColor = Color.Red
Vorm_Invoer.Text = "Afgekeurd"
```

**Figuur 74: het weergeven van een afgekeurde contour inkeping**

De controles op de hoekvormigheid en de spievormigheid worden op analoge wijze zichtbaar gemaakt in het resultatenscherf en is bijgevolg niet verder uitgelegd.

De gebruikte thresholdgrenzen in het hoofdprogramma (code onder *timer1*) zijn handmatig in te stellen in het instellingenscherf. Om die grensinstellingen effectief te kunnen gebruiken, dient er nog verwezen te worden naar de actuele waarden van de onder- en bovengrens, ingesteld in het instellingenscherf ("*Instellingen\_Scherf*"). Figuur 75 geeft de code voor deze procedure weer.

```
HOperatorSet.Threshold(ho_imageSub, ho_Regions, Instellingen_Scherf.ThresholdOndergrenswaarde, Instellingen_Scherf.ThresholdBovengrenswaarde)
```

**Figuur 75: verwijzing naar de actuele threshold waarden**

### **Opmerking**

**De andere parameters, zoals de *objectarea* en de *laserhoek*, verwijzen op analoge wijze naar het instellingenscherf.**

Ten slotte staat er onderaan in de code van het resultatenscherf een verwijzing naar een database in *Microsoft SQL Server*. Deze database moet namelijk de berekende lengte, breedte en hoogte van een goedgekeurde strip opnemen in een tabel (zie hoofdstuk 9).

### **7.3.3 Code instellingenscherf**

Het derde scherf (instellingenscherf) in *Visual Studio* vertegenwoordigt de instellingen van het visiesysteem. In paragraaf 7.2.3 werd al aangehaald welke parameters dienen te kunnen worden aangepast. Deze paragraaf geeft diepgaandere uitleg over de code die deze aanpassing mogelijk maakt, samen met de visualisatie ervan in het scherf zelf.

## Parameteraanpassing met visualisatie (willekeurige instelling)

Voor de *threshold* zijn de standaardwaarden voor onder- en bovengrens respectievelijk 0 en 50. Deze grenzen zijn experimenteel bepaald zodat bijna alle kleursoorten van steenstrippen uit het beeld gefilterd kunnen worden. Voor extreme gevallen, dient deze parameter echter (handmatig) aangepast te worden. Figuur 76 toont de code om de gekozen waarde van de balk “*ThresholdOndergrens\_Schuifbalk*” (zie figuur 77) effectief in te stellen en te laten verschijnen langs de desbetreffende schuifbalk. De huidige waarde van de ondergrens is momenteel ingesteld op 125.

```
Private Sub HScrollBar1_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs) Handles ThresholdOndergrens_Schuifbalk.Scroll
    ThresholdOndergrensWaarde = ThresholdOndergrens_Schuifbalk.Value
    Threshold_OnderGrens.Text = ThresholdOndergrensWaarde
End Sub
```

**Figuur 76: code van threshold-ondergrens**



**Figuur 77: schuifbalk van threshold-ondergrens**

Ter vervollediging toont Figuur 78 de code om de standaardwaarden voor de *threshold* te visualiseren in de desbetreffende schuifbalken. In Figuur 79 is het resultaat van die code weergegeven.

### Opmerking

**De schuifbalken van de *threshold* zijn gelimiteerd van 0 tot en met 255. Tussen deze waarden zitten alle mogelijke grijswaarden zodat de stripherkenning steeds verbeterd kan worden indien nodig.**

```
Public Sub NormaleThreshold_Knop_Click(sender As System.Object, e As System.EventArgs) Handles NormaleThreshold_Knop.Click
    ThresholdOndergrens_Schuifbalk.Value = 0
    ThresholdBovengrens_Schuifbalk.Value = 50
    ThresholdOndergrensWaarde = ThresholdOndergrens_Schuifbalk.Value
    ThresholdBovengrensWaarde = ThresholdBovengrens_Schuifbalk.Value
    Threshold_OnderGrens.Text = ThresholdOndergrensWaarde
    Threshold_BovenGrens.Text = ThresholdBovengrensWaarde
End Sub
```

**Figuur 78: instellen standaard waarden threshold**



**Figuur 79: schuifbalken met standaard waarden**

Wederom kan op analoge wijze de grenzen van de *objectarea* en de waarde voor de laserhoek zelfstandig aangepast worden. Ook hier heeft elke variabele zijn standaardwaarde die ingesteld kan worden met de bijbehorende “*Stel in*”-knop.



## 8 Testopstelling visiesysteem

Net zoals de strippenfacer, dient het visiesysteem getest te worden alvorens een definitieve implementatie in het productieproces tot stand kan komen. Deze testen dienen uitgevoerd te worden op een testopstelling. De testopstelling is gemaakt op *ACRO*, waar ook de testen zijn uitgevoerd. De uitgevoerde testen betreffen de volgende punten:

- Nauwkeurigheid van lengte- en breedtemetingen;
- Vorm- en hoekcontrole;
- Hoogtemetingen;
- Controle op spievormigheid;
- Snelheidstest (nagaan of het systeem kan werken met de huidige snelheid van de transportband).

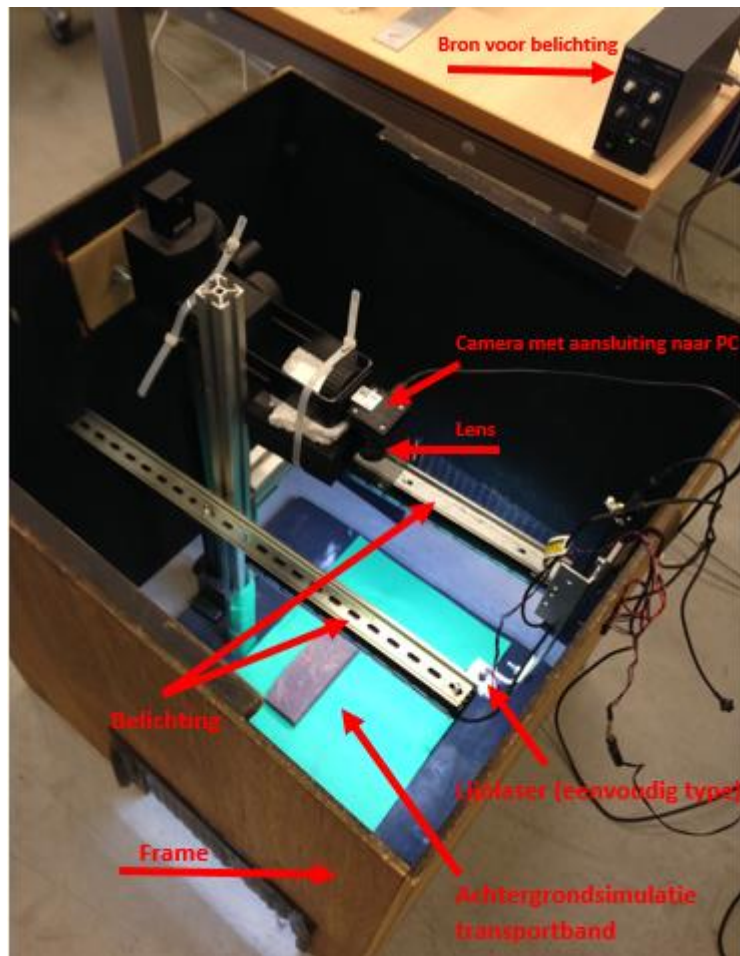
In de volgende paragrafen komen alle vernoemde testcriteria aan bod, maar vooreerst is er een beschrijving van de testopstelling gegeven.

### 8.1 Testopstelling

Zoals beschreven in paragraaf 5.2 bestaat het beoogde visiesysteem van deze masterproef uit de volgende onderdelen:

- Camera,
- Lens,
- Belichting,
- Laserlijn,
- Frame.

Ook de testopstelling dient uit deze onderdelen te bestaan. Belangrijk hierbij op te merken is dat de testopstelling slechts een benadering zal zijn van het toekomstig geïmplementeerde visiesysteem. Voor de testopstelling zijn immers geen aankopen verricht, maar is er gebruik gemaakt van recuperatiematerialen. Toch is het zo dat deze materialen zo zijn gekozen dat de werkelijkheid zo goed mogelijk benaderd werd waardoor de testresultaten zeker relevant en aanneembaar zullen zijn voor het toekomstige visiesysteem. In figuur 80 (volgende pagina) is de gebruikte testopstelling weergegeven met aanduiding van de verschillende componenten. De keuze van deze componenten is in de volgende deelparagrafen verder uitgediept.



*Figuur 80: testopstelling*

### 8.1.1 Camera testopstelling

In paragraaf 5.2 werden de belangrijkste eigenschappen van de camera al uitgelegd:

- Resolutie: 2048 x 1536;
- Framerate: 10 - 25 fps (*frames per second*);
- Sluiterijd: instelbaar tot <2,09 ms.

Bijgevolg is er op ACRO gezocht naar een camera die aan deze 'eisen' voldeed. Hierbij is de keuze gevallen op de *uEye UI-1460LE-C*. Volgens zijn datasheet (zie bijlage J) voldoet deze aan alle gevraagde criteria:

- resolutie: 2048 x 1536,
- framerate: 11 fps,
- sluitertijd: 57 - 1250 [ $\mu$ s].



### 8.1.2 Lens testopstelling

De lens van een visiesysteem wordt bepaald aan de hand van verschillende parameters zoals besproken in paragraaf 5.2.2. Om zo goed mogelijk aan het toekomstig geïmplementeerd visiesysteem te voldoen, werd bij de keuze van de lens rekening gehouden met een realistische werkafstand van de camera (zie paragraaf 5.2.2 voor uitleg). De keuze hing natuurlijk ook af van het beschikbare assortiment lenzen op ACRO. Het was immers niet ideaal om voor een testopstelling nieuwe lenzen aan te kopen. Deze zijn namelijk net zo duur en vaak nog duurder dan de camera zelf. Er dient echter ook rekening mee gehouden te worden dat de lens effectief past op de gekozen camera. Dit alles in beschouwing genomen heeft er toe geleid een lens te kiezen met een focale lengte van 12 mm: *Kowa LM12NCL*. De datasheet ervan is gegeven in bijlage K. De toe te passen werkafstand kan men bepalen met behulp van vergelijking (2) (paragraaf 5.2.2):

$$\frac{f}{L} = \frac{w}{W} = \frac{h}{H} \quad (2)$$

De waarden van de parameters uit vergelijking (2) zijn in dit geval:

- $f$ : 12,0 [mm],
- Chipafmetingen camera (zie bijlage J): 1/2",
  - $w$ : 6,554 [mm],
  - $h$ : 4,915 [mm],
- $W$ : 29,8 [mm]
- $H$ : 22,3 [mm]

Waaruit volgt dat een werkafstand  $L$  wordt bekomen van:

$$L = \frac{f \times W}{w} = \frac{12,0 \text{ [mm]} \times 29,8 \text{ [mm]}}{6,554 \text{ [mm]}} = \frac{12,0 \text{ [mm]} \times 29,8 \text{ [mm]}}{6,554 \text{ [mm]}} = 54,56 \text{ [mm]} \approx 55 \text{ [mm]}$$

### 8.1.3 Belichting testopstelling

Zoals besproken in paragraaf 5.2.3 is ervoor gekozen een gepolariseerde belichting te gebruiken. Dit is ook gebruikt in de testopstelling. De belichting bestaat uit twee LED-bars elk voorzien van een polarisatiefilm. Door ook een polarisatiefilm voor de cameralens aan te brengen, verkrijgt men een complete gepolariseerde belichting (zie ook uitleg in paragraaf 5.2.3). Herinner dat dit type van belichting gekozen is om de weerkaatsing op de natte transportband te compenseren. Deze zorgen namelijk voor overbelichte stukken in het beeld, wat zich uit in witte pixels. Deze zijn hinderlijk voor deze toepassing en dienen bijgevolg zoveel mogelijk verwijderd of gecompenseerd te worden. De datasheet van belichtingselementen (*LDL2-266X30SW-WD Bar Light* van *CCS Inc.*) is terug te vinden in bijlage L.

### 8.1.4 Lijnlaser testopstelling

De laserlijn vormt het hart van de hoogtemeting en controle op spievormigheid. Aangezien de nauwkeurigheid van deze metingen niet hoog zijn (gevraagd was een nauwkeurigheid van  $\pm 2$  mm) kan er voor de testopstelling gebruik gemaakt worden van een goedkope maar ook minder kwaliteitsvolle lijnlaser. Bij voorbaat kan er dus al gesteld worden dat de laserlijn die voor de werkelijke implementatie van het visiesysteem in het productieproces, kwaliteitsvoller moet zijn om een eventueel nauwkeuriger visiesysteem te bekomen. Het is namelijk zo dat minder kwaliteitsvolle lijnlasers een dikkere laserlijn creëren waarvan de densiteit minder scherp is. Een dikke lijn hangt onafscheidelijk samen met een onnauwkeurigere hoogtemeting. Het is namelijk moeilijker in te schatten of de bovenkant van de lijn, het midden of de onderkant ervan moet genomen worden voor de werking bij de hoogtemeting. Hoe dunner de lijn dus zal zijn, hoe minder dit probleem zal doorwegen en bijgevolg dus hoe nauwkeuriger het visiesysteem. Ook is het zo dat kwaliteitsvollere lijnlasers naast het genereren van dunnere laserlijnen, ook lijnen genereren met een hogere densiteit waardoor deze scherper zijn en beter worden opgenomen door de camera. Echter aangezien de nauwkeurigheid niet centraal staat als doel voor de hoogtemeting en de controle op spievormigheid, is er voor de testopstelling toch gebruik gemaakt van een eenvoudige lijnlaser (geen typenummer op lijnlaser of in database van ACRO).

### 8.1.5 Frame testopstelling

Het frame is de basis voor het visiesysteem. Alle componenten van het systeem worden hieraan immers gemonteerd. Zoals al beschreven in paragraaf 5.2.5 zal het frame van het uiteindelijke visiesysteem bestaan uit aluminiumprofielen, welke dicht dient gemaakt te worden om geen invloeden van het omgevingslicht te verkrijgen. Voor de testopstelling ziet dit frame er echter 'totaal' anders uit. Dit frame bestaat slechts uit één statief om de camera te positioneren en een kleine constructie (bestaande uit aluminiumprofielen) om de belichting op te kunnen stellen. De basis van het statief is toegelegd met groenachtig papier om een benaderend kleureffect van de werkelijke transportband te vormen. Verder is dit geheel omgeven door een box waarvan de binnenkant zwartgeverfd is, een zogenaamde *blackbox*. Over deze box is dan een doek gelegd waardoor het geheel van de box en het doek ervoor zorgt dat het omgevingslicht geen invloed kan uitoefenen op het visiesysteem. In de box is in twee tegenoverstaande vlakken eveneens een uitsparing voorzien voor een *rollenbaan*. Deze *rollenbaan* zal bij de snelheidstest dienst doen als een soort van transportband. Dit zal echter uitvoeriger worden besproken in paragraaf 8.2.5 (snelheidstest).

Hoewel het frame een andere constructie heeft dan die van het uiteindelijke visiesysteem, volstaat dit toch om de gewenste testen uit te voeren en resultaten te bekomen die zowel voor de testopstelling als voor het uiteindelijke visiesysteem representatief zullen zijn.

## 8.2 Testresultaten

### 8.2.1 Nauwkeurigheid van lengte-en breedtemeting

De lengte- en breedtemeting is eerst getest op een aantal stukjes metaal met bekende afmetingen alvorens de testen te starten op steenstrippen. Dit is gedaan omdat een steenstrip een ongelijke contour heeft waardoor nauwkeurig nameten quasi onmogelijk is. Op een stuk metaal met gekende afmetingen is dit nameten echter wel eenvoudig realiseerbaar. Het eerst gebruikte stuk metaal (figuur 81) had de volgende afmetingen:

- Lengte: 150,00 mm;
- Breedte: 50,00 mm;
- Hoogte: 20,00 mm.



*Figuur 81: metalen referentieblok*

De meetresultaten van het visiesysteem op dit metaal hadden steeds een nauwkeurigheid van minimum 0,5 mm ten opzichte van de nominale maat van zowel lengte als breedte. Testen op een stuk metaal met andere gekende afmetingen gaven eveneens een resultaat met een nauwkeurigheid van  $\pm 0,5$  mm.

Hieruit kan men al besluiten dat het visiesysteem het eerder gestelde doel om een meetnauwkeurigheid op lengte en breedte te hebben van  $\pm 1$  mm, haalt. Natuurlijk zijn er ook testen gebeurd op een reeks steenstrippen om na te gaan of de meting ook hier zijn efficiëntie haalt. Hieruit bleek dat de meetnauwkeurigheid op de strippen  $\pm 1$  mm was. Deze vermindering op het gebied van nauwkeurigheid is ontstaan door het onnauwkeurig kunnen nameten van de steenstrippen. Hierdoor kan men namelijk geen eenduidige uitspraak doen over de nauwkeurigheid van het visiesysteem zelf. Toch is er, rekening houdend met onnauwkeurig nameten, nog steeds een meetnauwkeurigheid van  $\pm 1$  mm vastgesteld. Het beoogde doel op het gebied van lengte-en breedtemeting is dus behaald met dit visiesysteem.

## 8.2.2 Vorm- en hoekcontrole

De testen in verband met de vorm-en hoekcontrole zijn wel onmiddellijk gebeurd op een steenstrip. Om uit de testresultaten een conclusie te kunnen trekken, zijn er verschillende steenstrippen soorten stenen getest:

- Strip met afgebroken hoek;
- Strip met een contourdefect (zie figuur 82);
- Strip dat kwaliteitscontrole bij Vandersanden NV heeft doorstaan.



*Figuur 82: strip met een contourdefect*

Op basis van de controles op deze types van stenen, gaf het systeem steeds de gewenste resultaten. Dit wil zeggen dat voor de eerste twee types de steen moest afgekeurd worden, terwijl het derde type een goedkeuring verdiende. De testen zijn op verschillende steenstrippen gebeurd waardoor geconcludeerd kan worden dat het systeem het gewenste doel heeft bereikt op het gebied van vorm- en hoekcontrole.

Belangrijk is op te merken dat de parameters die de procedure voor het controleren van de strip op vorm en hoeken, kunnen aangepast worden aan de wens van de gebruiker. Op deze manier is het visiesysteem zeer soepel in dit type kwaliteitscontrole en kan steeds het gewenste effect bekomen worden. Ook de testen zijn gebeurd door middel van verandering van die parameters.

In onderstaande figuur (figuur 83) zijn de drie gecontroleerde types van steenstrippen nog eens weergegeven met aanduiding van goed- of afkeuring.



*Figuur 83: gecontroleerde steenstrippen*

### 8.2.3 Hoogtemetingen

Het doel van deze testen was vooral het nagaan of een nauwkeurigheid van 2 mm op de nominale hoogte (dikte) kan behaald worden. De nauwkeurigheid mag hier dus een stuk lager liggen dan die van de lengte- en breedtemeting. Dit omdat de hoogtemetingen in eerste instantie spievormigheid moeten detecteren en slechts in tweede instantie een indicatie van de steenstripdikte moeten weergeven. Spievormigheid is namelijk een hoogteverloop en wordt gedetecteerd aan de hand van relatieve (geen absolute) hoogtewaardes, wat impliceert dat een grote nauwkeurigheid hier niet per se van toepassing dient te zijn.

Het nameten van de gemeten hoogtes is ook hier een onnauwkeurige aangelegenheid vanwege het ruwe en ongelijke oppervlak. Daarbij komt nog het feit dat het visiesysteem enkel een gemiddelde hoogte teruggeeft, wat het probleem van onnauwkeurig nameten bevestigt. Om die redenen is hier op dezelfde manier tewerk gegaan als in de testen op lengte en breedte, namelijk eerst de hoogtemeting testen op een stuk metaal met gekende hoogte (dikte). Dit stuk metaal had een dikte van 20,00 mm.

Uit testen en nameting bleek dat het visiesysteem een nauwkeurigheid van  $\pm 1,5$  mm op de nominale hoogte behaalde. De gewenste nauwkeurigheid is hiermee dan ook bereikt. Eveneens bleek uit de testen dat een correcte afstelling van de hoek waaronder de lijnlaser staat, een belangrijke invloed heeft op de nauwkeurigheid van de hoogtemetingen. Een kleine fout in het opmeten van deze hoek kan immers een grote fout opleveren in de hoogtemetingen. Een instellingsfout van  $2^\circ$  op de hoekinstelling gaf namelijk een onnauwkeurigheid van bijna 2 mm op de desbetreffende hoogtemeting. Een verdere verbetering van de nauwkeurigheid kan ook gebeuren door een kwalitatief betere lijnlaser te voorzien (hoge lichtdensiteit, helder).

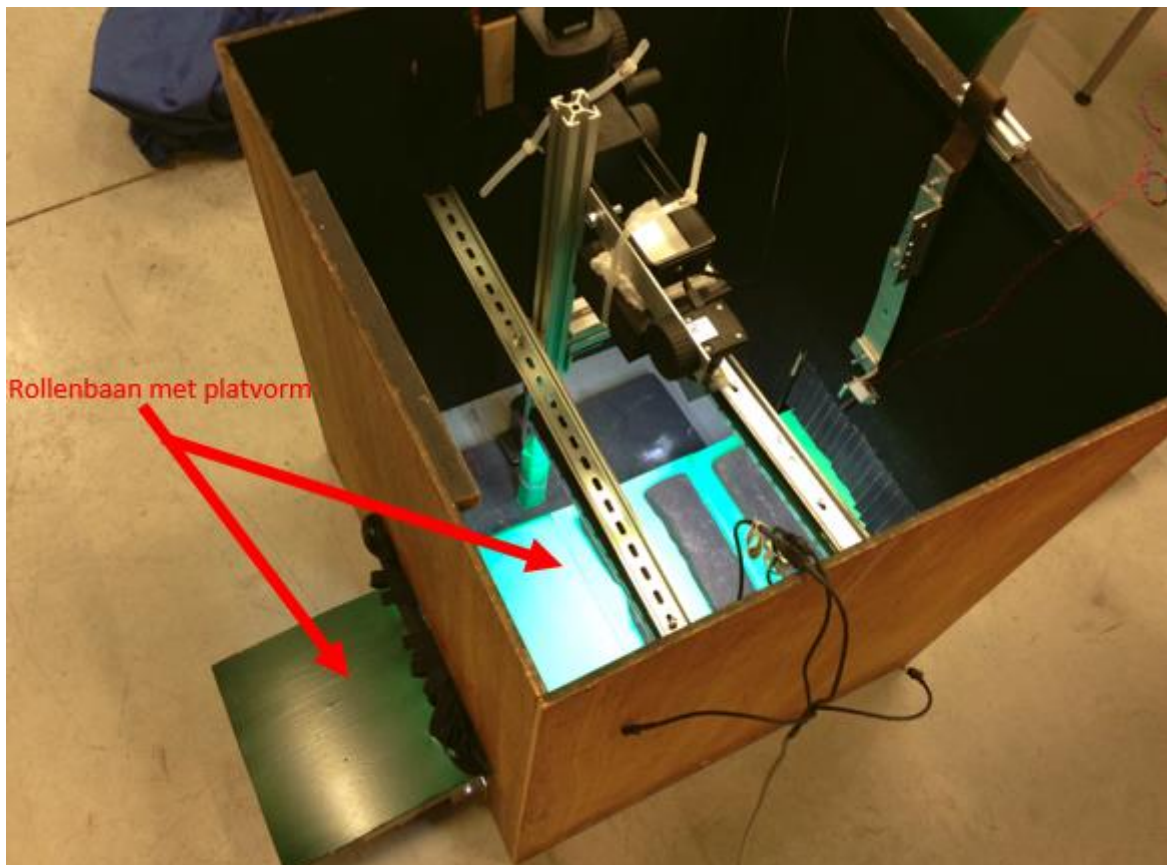
### 8.2.4 Controle op spievormigheid

Het doel van deze kwaliteitscontrole is om stenen met een spievormig verloop te detecteren en af te keuren. Tijdens de testen waren er echter te weinig spievormige steenstrippen beschikbaar waardoor ook houten spieën de test hebben ondergaan. Afhankelijk van de instelwaarden voor de strengheid van de spievormcontrole, werd de spievormige steenstrip tijdens de testen er steeds uitgehaald. Proeven op de houtvormige spieën hebben dit resultaat alleen maar bevestigd.

Toch dient er eerst een veel uitgebreidere test gedaan te worden alvorens een definitieve implementatie van het visiesysteem in de productielijn kan plaatsvinden. Een minimum van 200 stripcontroles **met** handmatige dubbelcontrole is vereist om gegronde conclusies te trekken over dit deel van de kwaliteitscontrole. Dit dient in het bedrijf zelf te gebeuren, net voor de definitieve implementatie. Op die manier kan men de instelwaarden ook verder proefondervindelijk afstellen tot de gewenste criteria op spievormgoedkeuring bekomen worden.

## 8.2.5 Snelheidstest en advies voor verhoging van de systeemsnelheid

Met deze testen is nagegaan of het visiesysteem de gewenste productiesnelheid van 40 steenstrippen per minuut haalt en of het systeem mogelijk hogere snelheden aankan met het oog op de toekomst. Om deze test te kunnen uitvoeren, moet de beweging van de huidige transportband zo goed mogelijk nagebootst worden in de testopstelling. Daarom is er gebruik gemaakt van een rollenbaan waarover een platvorm met een gelijkaardige kleur als de transportband horizontaal beweegt. Door steenstrippen op dit platvorm te plaatsen en het platvorm een snelheid mee te geven, komen de strippen eveneens met die snelheid voorbij de camera. Indien nu deze snelheid minstens zo hoog is als de snelheid van de werkelijke transportband, kunnen uit de testresultaten conclusies getrokken worden die eveneens representatief zijn voor het huidige productieproces. Figuur 84 toont de aangepaste testopstelling met implementatie van de rollenbaan.



*Figuur 84: testopstelling met implementatie van rollenbaan*

Uit de testen is gebleken dat met de huidige testopstelling een transportbandsnelheid van 4,5 cm/s mogelijk is opdat het visiesysteem de stenen kan detecteren. Het doel was echter een productiesnelheid van 40 strippen per seconden te behalen wat neerkomt op ongeveer 7 cm/s. Mogelijke verklaringen in vertragingen van de testopstelling kunnen teruggevonden worden in de gebruikte componenten. Zo werd het programma doorlopen met behulp van een laptop met

een minder goede processor (Intel®Core™ i5-2450M; 2,5GHz). Een vaste computer (desktop) of laptop met een hogere kloksnelheid (processor) kan het programma (beeldverwerking) tegen een hogere snelheid doorlopen, waardoor eveneens de systeemsnelheid stijgt. Dit is getest met een eigen laptop, welke een processor met een kloksnelheid van 2,40 GHz bevat (Intel®Core™ i7-3630QM). De resultaten van deze testen gaven aan dat door gebruik van de ‘verbeterde’ processor een transportbandsnelheid van 10 cm/s haalbaar is en dus in principe een kwaliteitscontrole bij een productiesnelheid van ongeveer 54 steenstrippen per minuut mogelijk is. Dit is echter een schatting van het aantal steenstrippen, aangezien dit aantal (even aantal vanwege twee strippen per baksteen) slechts berekend is d.m.v. de regel van drie:

$$40 \frac{\text{strippen}}{\text{min}} \rightarrow 7 \left[ \frac{\text{cm}}{\text{s}} \right] \xrightarrow{\text{Regel van drie}} 10 \left[ \frac{\text{cm}}{\text{s}} \right] \rightarrow 56 \frac{\text{strippen}}{\text{min}} \quad (3)$$

Met het gebruik van deze laptop is het beoogde doel van 40 strippen per minuut dus wel gehaald, waardoor de oorzaak van de mindere resultaten van de eerdere testen (lage verwerkingssnelheid) is bevestigd.

Naast de processorsnelheid, kan ook de transfersnelheid van de beelden van camera naar computer een belangrijke factor zijn indien een (nog) hogere systeemsnelheid gewenst is. Het verbeteren van de transfersnelheid is echter pas interessant wanneer de tijd voor het inlezen van beelden een groot deel van de totale verwerkingstijd inneemt. Indien dit niet het geval is, is het niet zinvol een aanpassing van transfertype te overwegen, tenzij datatransfer over lange afstanden dient te gebeuren. Sommige bussystemen kunnen immers eveneens goed functioneren op langere afstanden (bv. 100 m bij GigE). In deze masterproef stuurt de gebruikte camera zijn beelden door via USB. Dit is namelijk eenvoudig te implementeren (*Plug-and-play*) en dus interessant voor het gebruik in de testfase (externe testopstellingen). Tabel 3 toont ter vervollediging nog een overzicht van mogelijke bussystemen, elk met zijn eigenschappen.

**Tabel 3: overzicht van bussystemen voor beeldentransfer [22]**

	Cameralink	USB 2.0	IEEE1394	GigE
Topology	Master-slave	Master-slave	Peer to Peer	Peer to Peer
Bandwidth (Mbytes/s netto)	680	30	(a) 32, (b)64	85
Cable Length (m)	10	5, 10	5, 10	100m
Devices per bus	1	127	63	IP Space
Multi mastered	no	no	yes	yes
PC Load	Low(framegrabber)	Low to middle	Very Low	Low to middle

Tot slot kan ook geopteerd worden een camera met een hogere framerate en sluitertijd te kiezen om een goede kwaliteitscontrole te garanderen bij hogere snelheden. Hoe sneller (kleiner) de sluitertijd van de camera, hoe minder kans op blur-effecten (zie paragraaf 5.2.1) en dus hoe nauwkeuriger de kwaliteitscontrole kan gebeuren. Ook kan door het licht van de belichting te versterken, een lagere sluitertijd worden ingesteld om ook op die manier dit effect weg te nemen (juist afstellen zodat het gewenste beeld nog steeds wordt verkregen). Een hogere framerate heeft ook een effect op het verminderen van blur-effecten aangezien de camera in de mogelijkheid is sneller beelden te nemen.



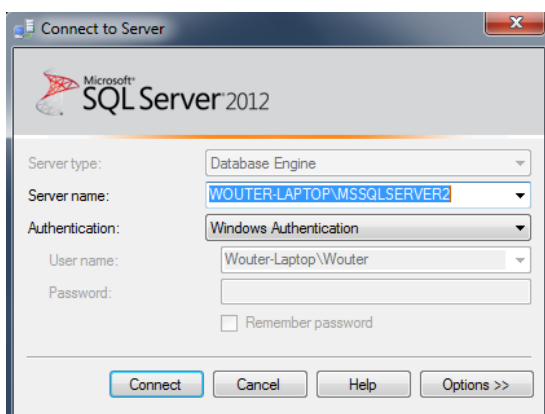


## 9 Microsoft SQL Server (database)

Om de afmetingen van een goede strip bij te houden, is er gebruik gemaakt van een database via Microsoft SQL Server. Dit is het programma dat Vandersanden Group nv. ook gebruikt voor het opslaan van gegevens in een database, vandaar de keuze voor *MS SQL Server*.

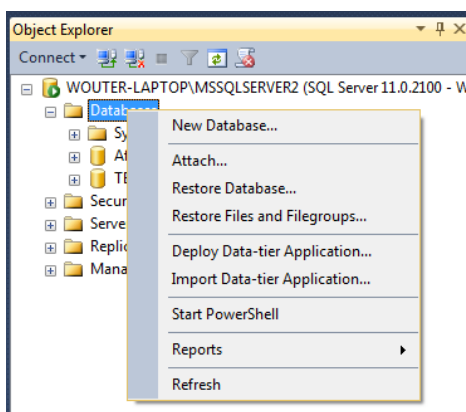
### 9.1 Database en tabel aanmaken

Het maken van een nieuwe database in *SQL Server* gebeurt in verschillende stappen. De eerste stap is het maken van een connectie met een server. Figuur 85 toont het connectiescherm dat automatisch geopend wanneer het programma opstart.



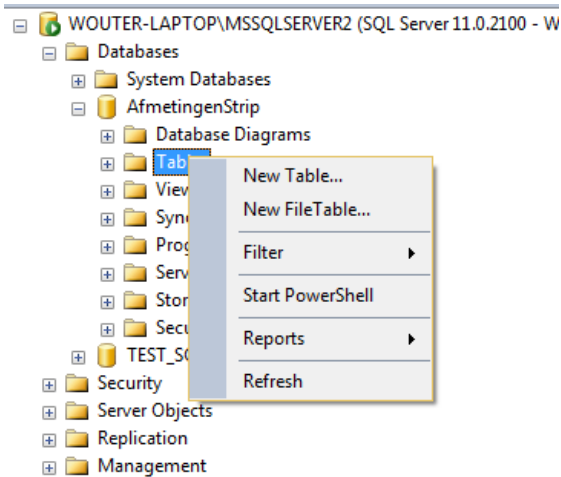
**Figuur 85: connectie met Server**

De naam van de server kan zelf gekozen worden, maar automatisch kiest SQL Server een standaard naam (in dit geval “WOUTER-LAPTOP\MSSQLSERVER2”). Deze naam wordt later gebruikt in Visual Studio om te verwijzen naar deze database. Vervolgens moet er een nieuwe database aangemaakt worden (zie figuur 86).



**Figuur 86: database aanmaken**

In de desbetreffende database moet er een nieuwe tabel (zie figuur 87) worden gemaakt met vier kolommen. De eerste kolom houdt het aantal steenstrippen bij (stripnummer) en vormt de *Primary Key* van de tabel (zie gele sleutel in figuur 88). Een Primary Key wordt toegekend aan een kolom om de uniciteit van zijn gegevens aan te geven. Zo zal elke goede strip namelijk een uniek nummer krijgen in de database. De drie andere kolommen geven respectievelijk de lengte, breedte en hoogte van de strip weer.



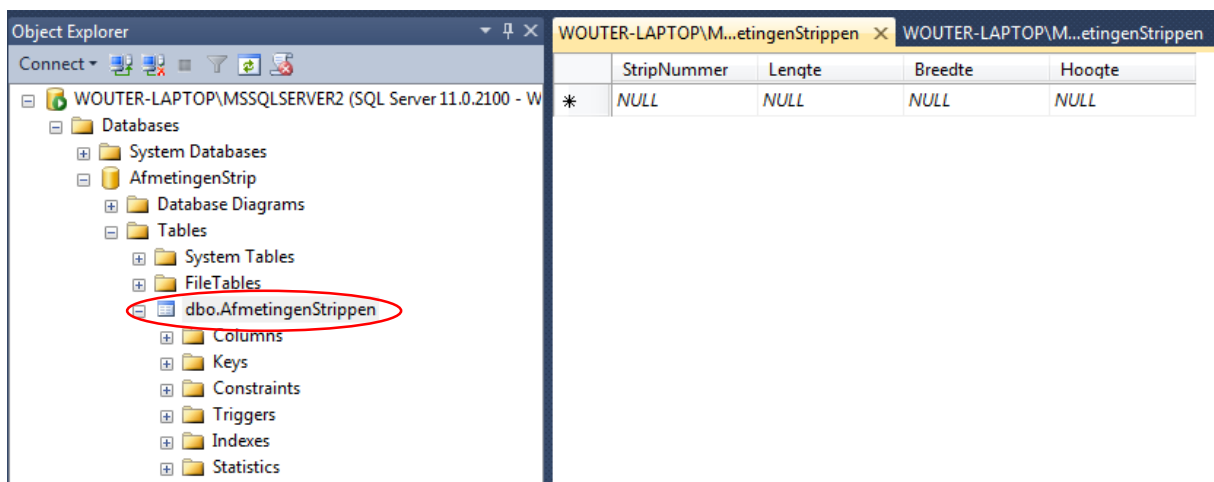
**Figuur 87: tabel aanmaken**

The screenshot shows the 'Columns in Table' dialog box for the 'AfmetingenStrip' table. The dialog has columns for 'Column Name', 'Data Type', and 'Allow Nulls'. The 'StripNummer' column is selected and highlighted in blue. The 'Allow Nulls' checkbox for 'StripNummer' is unchecked, while the checkboxes for 'Lengte', 'Breedte', and 'Hoogte' are checked.

Column Name	Data Type	Allow Nulls
StripNummer	int	<input type="checkbox"/>
Lengte	real	<input checked="" type="checkbox"/>
Breedte	real	<input checked="" type="checkbox"/>
Hoogte	real	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

**Figuur 88: maken van kolommen**

Door het sluiten van de aangemaakte tabel, zal SQL vragen om een naam te geven aan de tabel. Deze naam zal later wederom gebruikt worden in Visual Studio om beide programma's met elkaar te koppelen. Figuur 89 toont de aangemaakte tabel ("AfmetingenStrippen"). In deze tabel zijn echter nog geen enkele gegevens opgeslagen, waardoor elke kolom de waarde "NULL" krijgt.

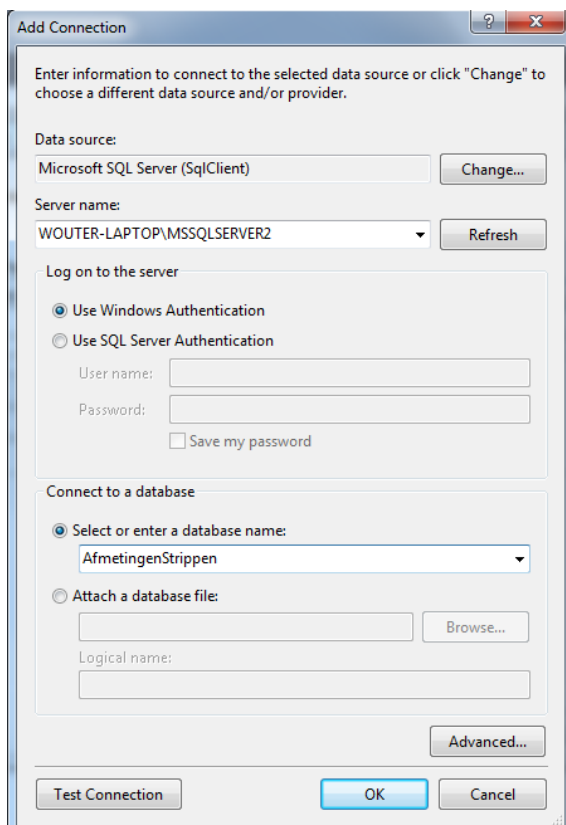


**Figuur 89: de aangemaakte tabel**

## 9.2 Connectie tussen Visual Studio en SQL Server

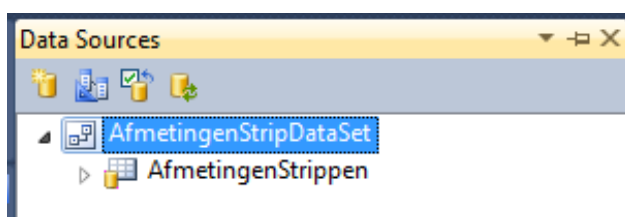
De vorige paragraaf beschreef de werkwijze voor het aanmaken van een database en een tabel in MS SQL Server. In hetgeen wat volgt, wordt de procedure voor het koppelen van de database met Visual Studio beschreven.

In Visual Studio moet er een nieuwe *data source* worden aangemaakt, die verwijst naar de desbetreffende database in SQL Server. Door het kiezen van SQL Server in de *Data Source Configuration Wizard* wordt er een scherm geopend, zoals getoond in figuur 90.



**Figuur 90: connectie database toevoegen**

In dit scherm (figuur 90) wordt verwezen naar de *Server name* (bovenste rode aanduiding) en de *database name* (onderste rode aanduiding) die zijn aangemaakt in SQL Server. Door de *Wizard* te voltooien, komt er een nieuwe *DataSet* ("AfmetingenStripDataSet") tevoorschijn bij *Data Sources* (zie figuur 91).



**Figuur 91: nieuwe DataSet**

Om nu de afmetingen in de database te zetten, moet er eerst een nieuwe klasse worden aangemaakt in Visual Studio. De naam van deze klasse is "DataConnectie" en hierin wordt een programma geschreven die de gegevens verstuurt naar de database. Figuur 92 toont de code voor dit programma (procedure).

```

Imports System.Data.SqlClient
Imports System.Globalization

Public Class DatabaseConnectie
    Dim connectieString As String
    Dim connectie As SqlConnection
    Dim aantal As Integer

    Public Sub New()
        connectieString = My.Settings.AfmetingenStripConnectieSQL
        connectie = New SqlConnection(connectieString)
    End Sub

    Public Sub addAfmetingenSQL(ByVal Lengte As Double, ByVal Breedte As Double, ByVal Hoogte As Double)

        Dim query = "insert into AfmetingenStrippen(Lengte,Breedte,Hoogte) " & " values(@Lengte,@Breedte,@Hoogte)"

        Debug.WriteLine(query)
        Dim commando = New SqlCommand(query, connectie)
        commando.Parameters.Add(New SqlParameter("@Lengte", Lengte))
        commando.Parameters.Add(New SqlParameter("@Breedte", Breedte))
        commando.Parameters.Add(New SqlParameter("@Hoogte", Hoogte))

        commando.Connection.Open()
        Dim aantal = commando.ExecuteNonQuery()

        MsgBox("Records toegevoegd: " & aantal & " x")

        commando.Connection.Close()

    End Sub
End Class

```

**Figuur 92: code voor connectie met database**

De eerste *public sub new()* zorgt voor de communicatie met de database en de *Public Sub addAfmetingenSQL(...)* stuurt elke afmeting naar de correcte tabel van de SQL database. In het hoofdprogramma moeten de afmetingen (lengte, breedte en hoogte) staan tussen de haakjes van dit aangemaakte programma. De volgende code geeft weer hoe de afmetingen moeten worden ingegeven in het hoofdprogramma:

*sql.addAfmetingenSQL(striplengte, stripbreedte, striphoogte).*

Ten slotte is het noodzakelijk de procedure, zoals getoond in figuur 93, in het hoofdprogramma te zetten om een verwijzing naar de klasse *DatabaseConnectie* te bekomen. De volledige code in *Visual Studio* voor de connectie met de database (*MS SQL Server*) is weergegeven in bijlage M.

```

Public Sub New()
    InitializeComponent()
    sql = New DatabaseConnectie()
End Sub

```

**Figuur 93: verwijzen naar DatabaseConnectie**

# 10 Conclusie

## 10.1 Algemene conclusie

Deze masterproef heeft aangetoond dat een automatische kwaliteitscontrole voor de desbetreffende strippenzaag zeker een haalbaar gegeven is. Zowel de voorbereiding van de controle (strippenfacer) als de kwaliteitscontrole zelf voldoen aan de opgelegde eisen. De strippenfacer zorgt namelijk voor een gecontroleerde aanvoer van de strippen (gezaagde zijde op transportband) om een zo representatief mogelijk kwaliteitscontrole te kunnen uitvoeren. Deze controle voert de volgende deelopdrachten uit op de desbetreffende steenstrip:

- lengte- en breedtemeting met een nauwkeurigheid van  $\pm 0,5$  mm;
- vorm- en hoekcontrole op basis van aanpasbare kwaliteitseisen;
- hoogtemeting met nauwkeurigheid van  $\pm 2$  mm;
- spievormcontrole (aanpasbare kwaliteitseisen).

De gewenste productiesnelheid van 40 strippen per minuut is met dit systeem tevens geen probleem indien men gebruik maakt van een computer met een hogere processorsnelheid (bv. Intel®Core™ i7-3630QM). Hierbij kan men gaan tot een productiesnelheid van 56 strippen per minuut (ideale geval). Indien in de testopstelling gebruik wordt gemaakt van een gewone laptop (Intel®Core™ i5-2450M; 2,5GHz) kan een productiesnelheid van 28 strippen per minuut gehaald worden. Het gebruik van een camera met een hogere framerate en snellere sluitertijd kan dan weer zorgen voor het behoud van een nauwkeurige kwaliteitscontrole bij hogere productiesnelheden (minder kans op blur-effecten en de mogelijkheid tot hoger aantal beelden per seconden). Voor de testopstelling was deze camera echter niet voorradig waardoor een kwalitatief minder camera is gebruikt. Toch is ook met deze camera en een computer met een goede processorsnelheid de gewenste productiesnelheid haalbaar waardoor alle resultaten representatief zijn voor toekomstige implementatie in het effectieve productieproces.

De gemeten lengte- en breedte van elke goede strip worden ook opgeslagen in een database (MS SQL Server). Op deze manier heeft men een beter zicht op groottes van de strippen en kan men op basis van die gegevens in de toekomst de nodige aanpassingen doen om het proces nog te verbeteren. Dit is dus vooral een toekomstgerichte applicatie binnen deze masterproef.

Er kan dus besloten worden dat de gestelde doelstellingen zijn gehaald. Zowel de strippenaanvoer m.b.v. de strippenfacer en de kwaliteitscontrole op basis van een visiesysteem geven het gewenste resultaat en hierbij ook de gewenste nauwkeurigheid.

## 10.2 Kostenraming

Om een zicht te bekomen op het te verwachten kostenplaatje van het gehele kwaliteitssysteem, geeft tabel 4 een overzicht van de gebruikte elementen en hun kostprijs. Dit overzicht geeft zowel de kostprijs van de testopstelling weer als de totale kostprijs indien het systeem effectief wordt geïmplementeerd, aangezien beiden toch een verschillend kostenplaatje hebben. Bij de effectieve implementatie is namelijk een *RUN-time* licentie van *HALCON* nodig in plaats van een ontwikkelaarslicentie. Ook dient een vaste pc te worden aangekocht om het programma op te laten werken met een voldoende hoge verwerkingssnelheid (bv. 3,2 GHz). Naast deze grote kosten, dienen ook nog de aluminiumprofielen te worden aangekocht voor de constructie van het frame, samen met de benodigde bevestigingsonderdelen zoals bouten, moeren, *T-moeren*...

Tot slot is het ook mogelijk om een kwalitatievere camera en lijnlaser te kiezen. Dit betreft een camera met een hogere framerate (maar lagere resolutie) en snellere sluitertijd en een lijnlaser die een felle, dichte laserlijn kan genereren. Beiden zorgen naast de kwaliteitsinjectie ook voor een verhoging van de kostprijs. Dit alles is wederom weergegeven in tabel 4 zodat Vandersanden Group nv. op basis hiervan zelf nog kan bepalen welke onderdelen (types) ze willen hebben in het globale systeem.

**Tabel 4: kostenraming (schatting)**

Kostenraming testopstelling					
Component	Merk/bedrijf	Type	aantal	eenheidsprijs (€)	Totale prijs (€)
Camera	uEye	UI-1465LE-C-HQ	1	€ 313.85	€ 313.85
Lens	Kowa	LM12NCL	1	€ 98.46	€ 98.46
Belichting	CCS Inc.	LDL2-266X30SW-WD Bar Light	2	1387	€ 2 774.00
Spanningsbron	CCS Inc.	PD2-3204-2	1	378	€ 378.00
Laser	/	/	1	/	/
Ontwikkelaarslicentie HALCON	MVTec.	Halcon 11.0	1	0	€ 0.00
Frame	/	/	1	/	/
				<b>Subtotaal</b>	<b>€ 3 564.31</b>
Schatting extra kosten bij implementatie + strippenfacer					
Component	Merk/bedrijf	Type	aantal	Geschatte eenheidsprijs (€)	Geschatte totale prijs (€)
Run-time-licentie HALCON	MVTec.	Halcon 11.0	1	€ 1 500.00	€ 1 500.00
Vaste PC	/	/	1	€ 2 000.00	€ 2 000.00
<b>Frame</b>					
Aluminiumprofielen	Coomach nv.	Profiel 30x30 (6m)	1	€8.22/m	€ 49.32
<b>Bevestigingsonderdelen frame (a.d.h.v. profieltype)</b>					
Inbusbout	Coomach nv.	Op maat voor gekozen profieltype	42	€ 0.18	€ 7.56
Inklikmoer	Coomach nv.	Op maat voor gekozen profieltype	42	€ 0.36	€ 15.12
Hoekverbinding	Coomach nv.	Op maat voor gekozen profieltype	20	€ 1.85	€ 37.00
				<b>Subtotaal</b>	<b>€ 3 609.00</b>
Strippenfacer	Elmech	Roestvrij staal	1	€ 566.00	€ 566.00
				<b>Totale (geschatte) prijs</b>	<b>€ 7 739.31</b>
Schatting kosten voor gebruik van enkele alternatieve componenten					
Component	Merk/bedrijf	Type	aantal	Geschatte eenheidsprijs (€)	Geschatte totale prijs (€)
Camera	uEye	UI-1540LE-M-GL	1	€ 221.54	€ 221.54
Laser (+ bevestiging)	Z-laser	ZM18B	1	€ 550.00	€ 550.00

## Literatuurlijst

- [1] „Historiek Vandersanden GROUP,” Vandersanden GROUP, [Online]. Available: <http://www.vandersandengroup.be/group/nl-be/historiek>. [Geopend 28 Augustus 2013].
- [2] „Staal,” Tosec, [Online]. Available: <http://www.tosec.nl/wiki/staal/>. [Geopend 2 Mei 2014].
- [3] „Dichtheid tabel,” Osbexact, 19 Juni 2006. [Online]. Available: [http://osbexact.nl/pages/189/Dichtheid\\_\(tabel\).html](http://osbexact.nl/pages/189/Dichtheid_(tabel).html). [Geopend 2 Mei 2014].
- [4] „Staalplaat kostprijs,” Ijzershop, 2010. [Online]. Available: <http://www.ijzershop.nl/301-staalplaat>. [Geopend 3 Mei 2014].
- [5] „Roestvrij staal,” Euro-Inox, [Online]. Available: [http://www.euro-inox.org/pdf/map/What\\_is\\_Stainless\\_Steel\\_NL.pdf](http://www.euro-inox.org/pdf/map/What_is_Stainless_Steel_NL.pdf). [Geopend 3 Mei 2014].
- [6] „RVS-plaat kostprijs,” Ijzershop, 2010. [Online]. Available: <http://www.ijzershop.nl/305-rvs-plaat>. [Geopend 3 Mei 2014].
- [7] „Treksterkte Aluminium,” mcbboek, [Online]. Available: [http://mcbboek.nl/MCB\\_h07/Overzicht\\_van\\_de\\_belangrijkste\\_eigenschappen\\_van\\_aluminium.htm](http://mcbboek.nl/MCB_h07/Overzicht_van_de_belangrijkste_eigenschappen_van_aluminium.htm). [Geopend 4 Mei 2014].
- [8] „Aluminium plaat kostprijs,” Ijzershop, 2010. [Online]. Available: <http://www.ijzershop.nl/304-alu-plaat>. [Geopend 5 Mei 2014].
- [9] J. Baeten, „Visietechnologie - het optisch systeem,” KHLim, [Online]. Available: [http://websites.khlim.be/jbaeten/visie/VisieTech2\\_lens.pdf](http://websites.khlim.be/jbaeten/visie/VisieTech2_lens.pdf). [Geopend 3 Mei 2014].
- [10] Data Vision, „Sluiterijd,” in *DATA VISION hardware handboek*, pp. 16-17.
- [11] K. Donné en W. Beckers, „Machinevisie: praktische aspecten,” in *Elektrotechniek en Automatisering Capita Select*, Gent, Academia Press, 2008, pp. 633-675.
- [12] J. Baeten, „Visietechnologie - Belichting,” KHLim, [Online]. Available: [http://websites.khlim.be/jbaeten/visie/VisieTech1\\_belichting.pdf](http://websites.khlim.be/jbaeten/visie/VisieTech1_belichting.pdf). [Geopend 22 April 2014].
- [13] Z-Laser, „Info over ZM18B,” Z-Laser, [Online]. Available: <http://www.z-laser.com/en/products/product/machine-vision-lasers/zm18b/zm18/>. [Geopend 22 Mei 2014].
- [14] Z-Laser, „Info ZM12B,” Z-Laser, [Online]. Available: <http://www.z-laser.com/en/products/product/machine-vision-lasers/zm12b/zm12/>. [Geopend 22 Mei 2014].

- [15] Z-Laser, „Info over ZD-type,” Z-Laser, [Online]. Available: <http://www.z-laser.com/en/products/product/machine-vision-lasers/zd/more-lasers-for-machine-vision/>. [Geopend 22 Mei 2014].
- [16] Coherent, „Lasiris™ Line Advantage,” Coherent, 21 Mei 2014. [Online]. Available: <https://www.coherent.com/products/?1829/Lasiris-Line-Advantage>. [Geopend 22 Mei 2014].
- [17] „Gocator van LMI,” LMI Technologies, [Online]. Available: <http://www.lmi3d.com/products/gocator/>. [Geopend 10 November 2014].
- [18] F. Scheelen en M. Verheyen, „Ontwerp van een visiegebaseerde afvalsorteermachine voor MDF en houtafval,” Diepenbeek, 2008.
- [19] „Microsoft Visual Studio,” Wikipedia, [Online]. Available: [http://nl.wikipedia.org/wiki/Visual\\_Studio](http://nl.wikipedia.org/wiki/Visual_Studio). [Geopend 12 Mei 2014].
- [20] „Displaying Images: Using The PictureBox Control,” Microsoft Developer Network, [Online]. Available: [http://msdn.microsoft.com/en-us/library/ms172590\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/ms172590(v=vs.90).aspx). [Geopend 1 Mei 2014].
- [21] „Using the Horizontal and Vertical Scroll Bar Controls,” Microsoft Developer Network, [Online]. Available: [http://msdn.microsoft.com/en-us/library/aa263200\(v=vs.60\).aspx](http://msdn.microsoft.com/en-us/library/aa263200(v=vs.60).aspx). [Geopend 1 Mei 2014].
- [22] G. Deconinck en S. P. (red.), „Camera-computerinterface,” in *Elektrotechniek en Automatisering Capita Selecta*, Gent, Academia Press, 2011, pp. 652-653.

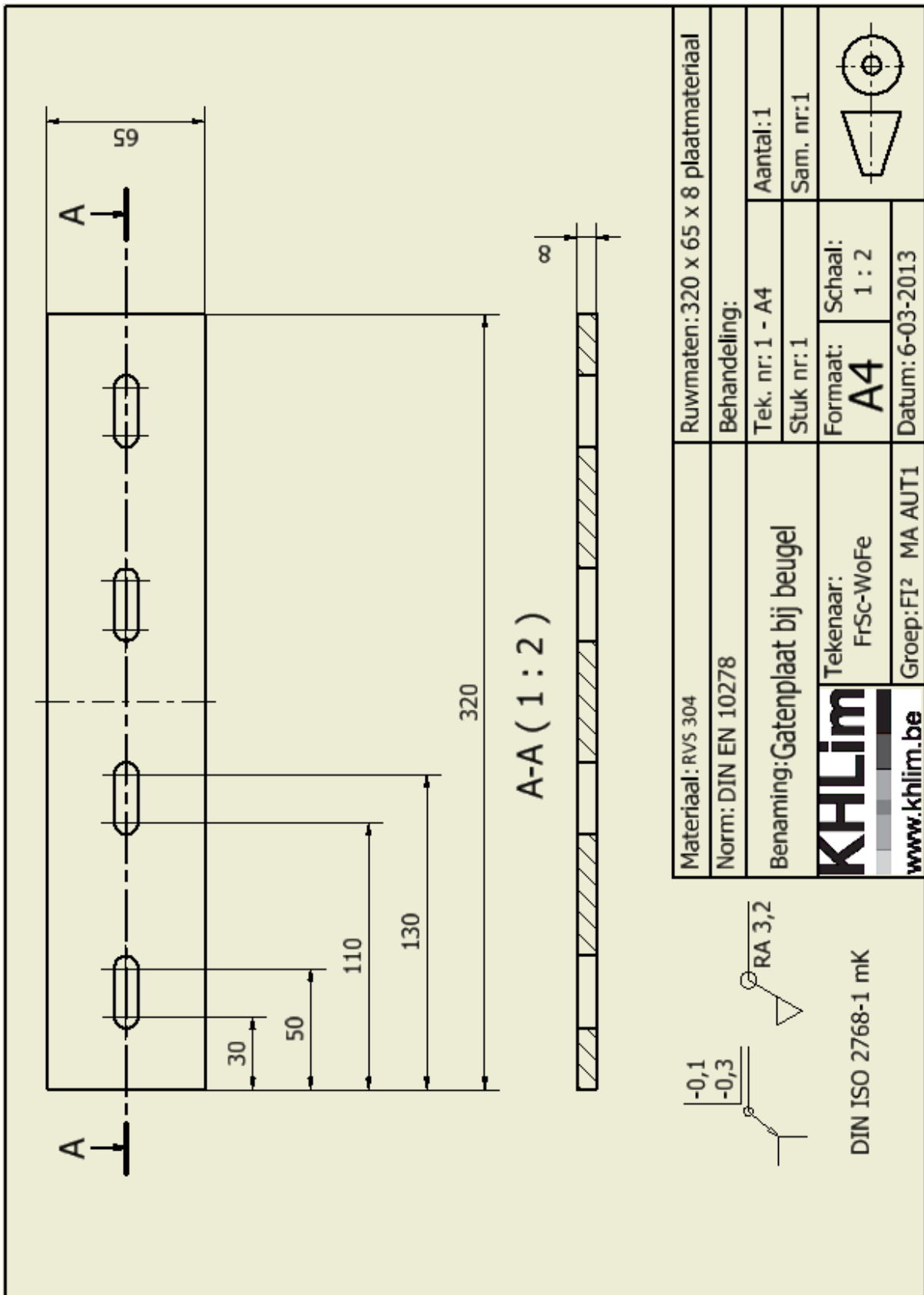


## Bijlagen

Bijlage A: 2D-tekeningen van strippenfacer .....	113
Bijlage B: prijsofferte Jacometal BVBA (strippenfacer) .....	120
Bijlage C: datasheet ZM18B .....	121
Bijlage D: Datasheet ZM12B .....	123
Bijlage E: datasheet ZD .....	125
Bijlage F: Complete programmacode in HALCON.....	127
Bijlage G: programmacode van form 1 in Visual Studio.....	133
Bijlage H: programmacode van form 2 in Visual Studio .....	135
Bijlage I: programmacode van form 3 in Visual Studio .....	151
Bijlage J: datasheet uEye UI-1460LE-C .....	154
Bijlage K: datasheet Kowa LM12NCL (lens).....	156
Bijlage L: datasheet belichtingselementen (catalogoog).....	157
Bijlage M: code in Visual Studio voor connectie met database .....	159



# Bijlage A: 2D-tekeningen van strippenfacer

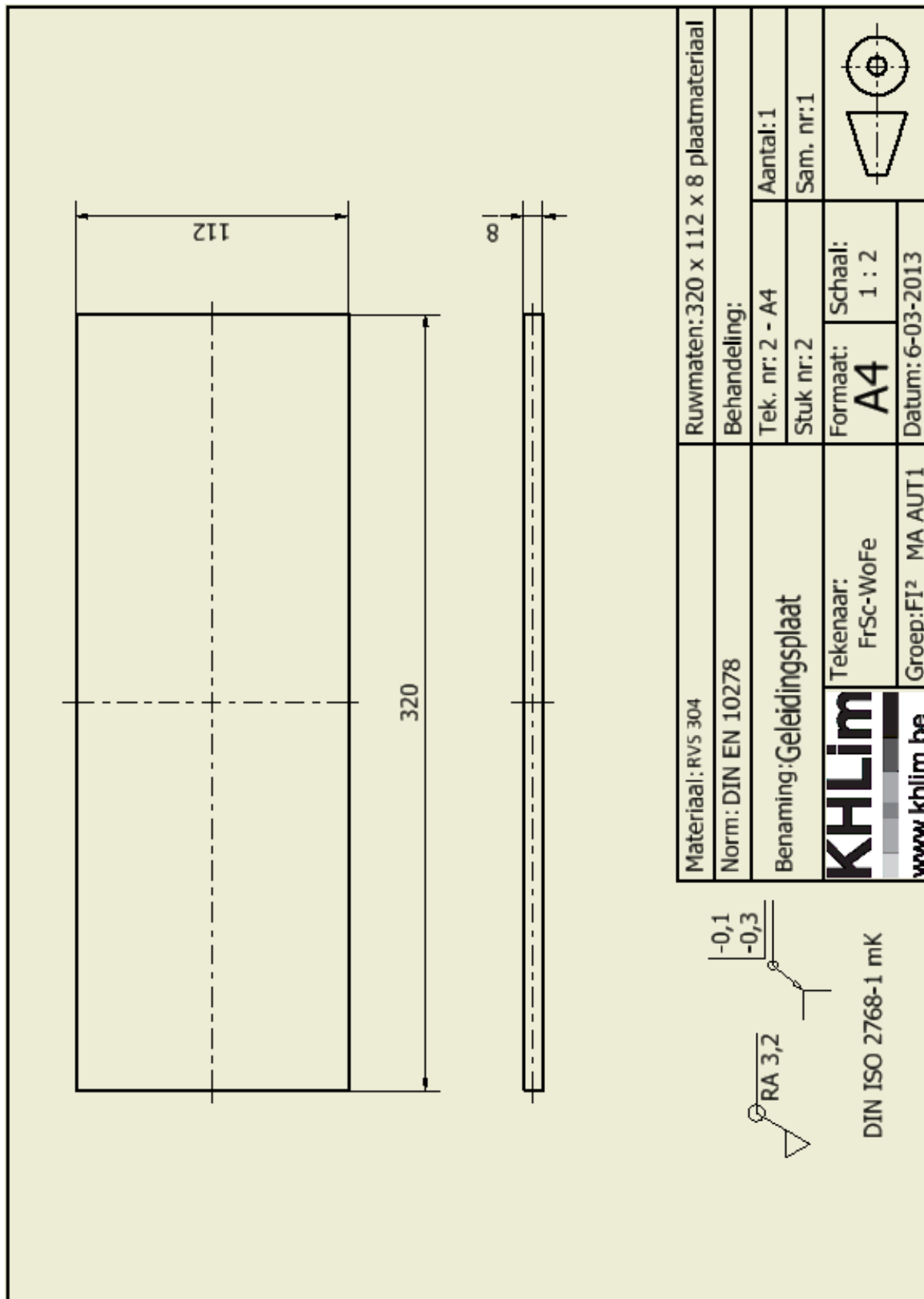


Materiaal: RVS 304	Ruwmaten: 320 x 65 x 8 plaatmateriaal	
Norm: DIN EN 10278	Behandeling:	
Benaming: Gatenplaat bij beugel	Tek. nr: 1 - A4	Aantal: 1
	Stuk nr: 1	Sam. nr: 1
<b>KHLim</b> Tekenaar: FrSc-Wofe www.khlim.be	Formaat: A4	Schaal: 1 : 2
	Groep: FI <sup>2</sup> MA AUT1	Datum: 6-03-2013

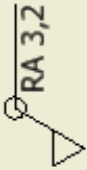

-0,1  
-0,3

RA 3,2

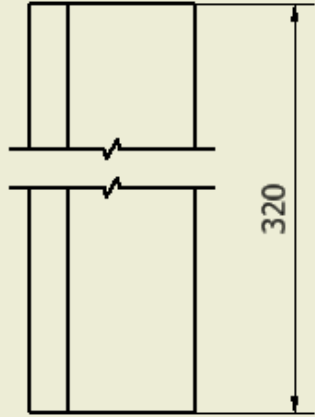
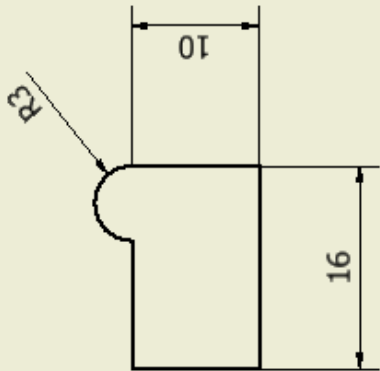
DIN ISO 2768-1 mK



Materiaal: RVS 304	Ruwmaten: 320 x 112 x 8 plaatmateriaal	
Norm: DIN EN 10278	Behandeling:	
Benaming: Geleidingsplaat	Tek. nr: 2 - A4	Aantal: 1
	Stuk nr: 2	Sam. nr: 1
<b>KHLim</b> Tekenaar: FrSc-WoFe www.khlim.be	Formaat: A4	Schaal: 1 : 2
	Groep: FI <sup>2</sup> MA AUT1	Datum: 6-03-2013

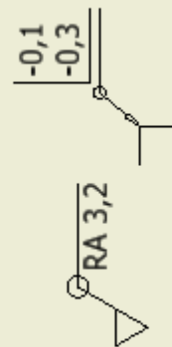
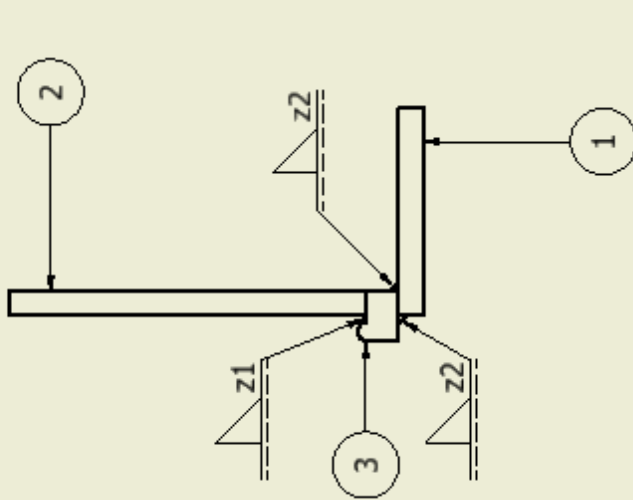

 RA 3,2  

 -0,1  
 -0,3  
 DIN ISO 2768-1 mK



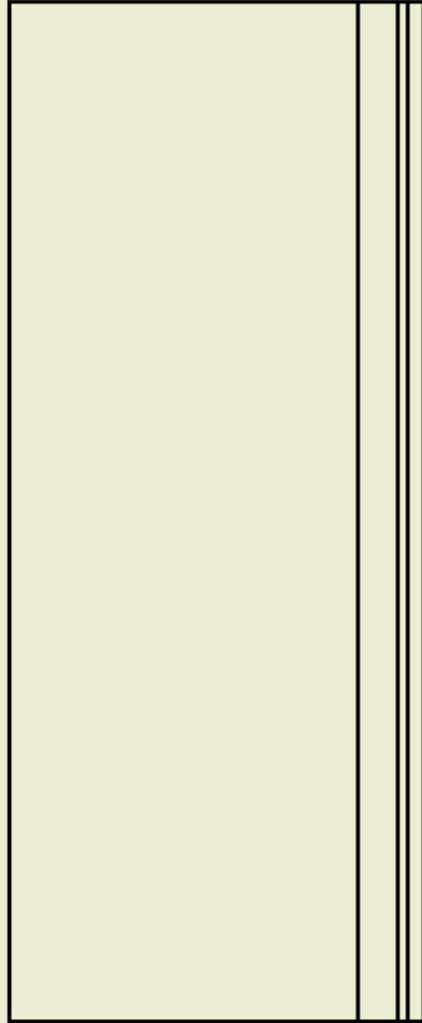


DIN ISO 2768-1 mK

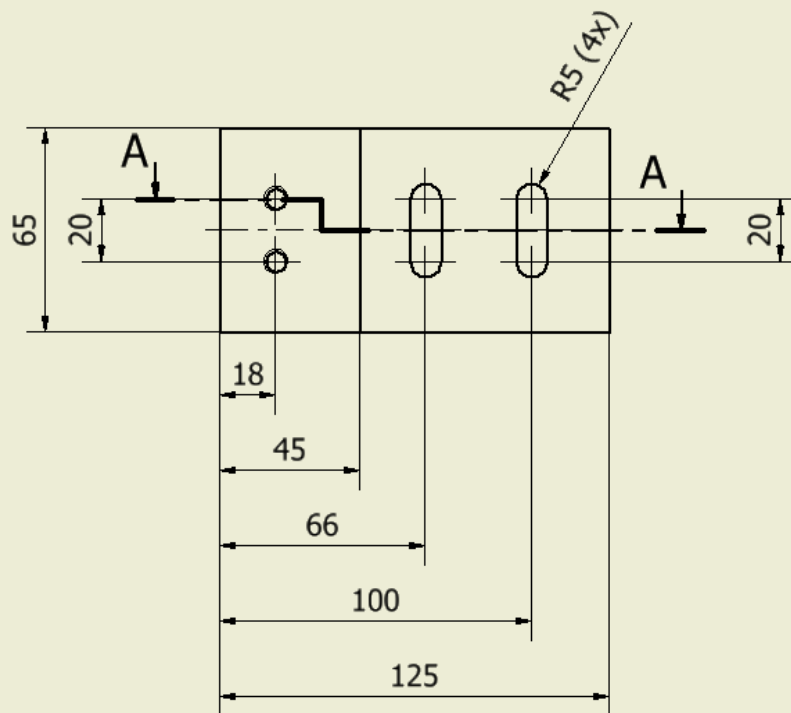
Materiaal: RVS 304	Ruwmaten: 320 x 18 x 15	
Norm: DIN EN 10278	Behandeling:	
Benaming: Bult	Tek. nr: 3 - A4	Aantal: 1
	Stuk nr: 3	Sam. nr: 1
<b>KHLIM</b> Tekenaar: FrSc-WoFe www.khlim.be	Formaat: <b>A4</b>	Schaal: 2 : 1
	Groep: FI <sup>2</sup> MA AUT1	Datum: 6-03-2013



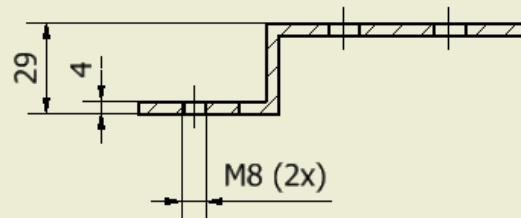
DIN ISO 2768-1 mK



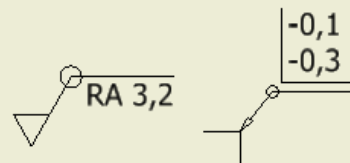
Materiaal: RVS 304	Ruwmaten:	
Norm: DIN EN 10278	Behandeling:	
Benaming: L-profiel stripfenacer	Tek. nr:	Aantal: 1
	Stuk nr:	Sam. nr: 1
<b>KHLIM</b> Tekenaar: FrSc-WoFe www.khlim.be	Formaat: <b>A4</b>	Schaal:
	Groep: FI <sup>2</sup> MA AUT1	Datum: 6-03-2013


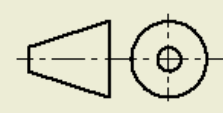


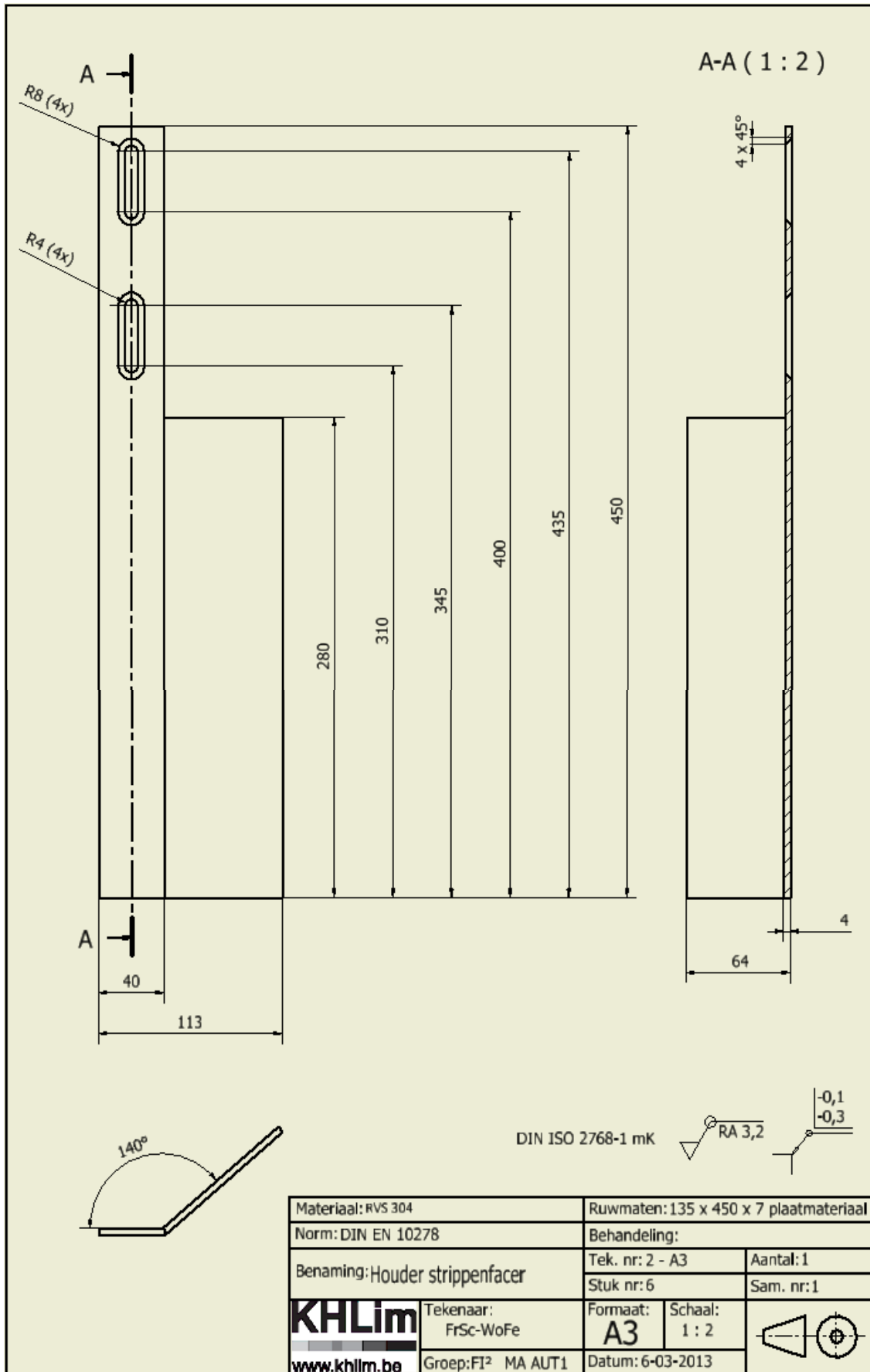
A-A ( 1 : 2 )



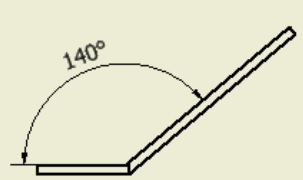
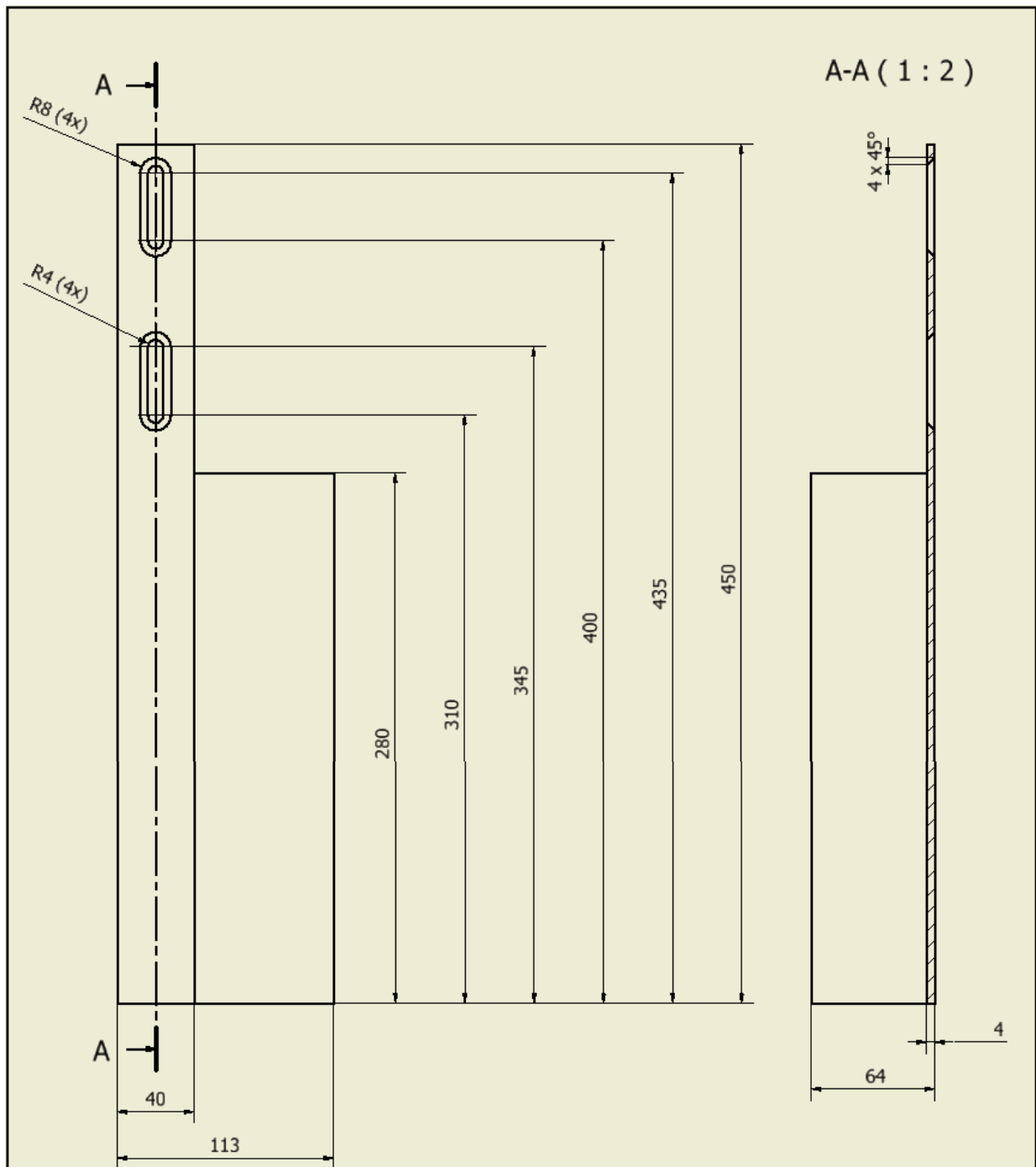
DIN ISO 2768-1 mK



Materiaal: RVS 304		Ruwmaten: 160 x 65 x 4 plaatmateriaal	
Norm: DIN EN 10278		Behandeling:	
Benaming: Houder transportband		Tek. nr: 4 - A4	Aantal: 2
		Stuk nr: 4	Sam. nr: 1
 <a href="http://www.khlim.be">www.khlim.be</a>	Tekenaar: FrSc-WoFe	Formaat: <b>A4</b>	Schaal: 1 : 2
	Groep: FI <sup>2</sup> MA AUT1	Datum: 6-03-2013	
			

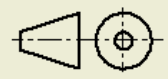






DIN ISO 2768-1 mK RA 3,2 -0,1  
-0,3

Materiaal: RVS 304		Ruwmaten: 135 x 450 x 7 plaatmateriaal	
Norm: DIN EN 10278		Behandeling:	
Benaming: Houder strippenfacer		Tek. nr: 2 - A3	Aantal: 1
		Stuk nr: 6	Sam. nr: 1
<b>KHLim</b> www.khlim.be	Tekenaar: FrSc-WoFe	Formaat: <b>A3</b>	Schaal: 1 : 2
	Groep: FI <sup>2</sup> MA AUT1	Datum: 6-03-2013	



## Bijlage B: prijsofferte Jacometal BVBA (strippenfacer)



**JACOMETALBVBA**  
 METAALCONSTRUCTIE  
 INRICHTING SCHIETSTANDEN

**WouterFerson**  
**projectstrippenfacervds**

B-

### OFFERTE (Blz. 1/1)

Datum	Documentnr	Klant nr.	Uw ref.	Ingave	
27/02/2014	OK 532	000222	strippenfacervds	MICHEL	
Omschrijving	Aantal	Eenhpr.	%	Totaal	BTW
<b>STRIPPENFACER VDS:</b>					
* bult: eenheidsprijs incl. oplassen:	1,00	117,20		117,20	21
* houder transportband:	2,00	34,85		69,70	21
* I-profiel strippenfacer: uit plaat geplooids, dus met radius:	1,00	55,21		55,21	21
* houder strippenfacer	1,00	58,44		58,44	21
* u-profiel strippenfacer	1,00	31,40		31,40	21
- s 235 onbehandeld					
- levertermijn: +/- 2 weken					
			Totaal excl.	331,95	
			Totaal BTW	69,71	
			Te betalen	EUR	<b>401,66</b>
			<b>Datum en handtekening</b>		

Jacometal bvba - Resekervelt 17 - 3770 Riemst - Tel: +32 (0)12 39 18 88 - Fax: +32 (0)12 39 18 87

BTW-nummer: BE 0841.265.954 - RPR Tongeren

Bijlage C: datasheet ZM18B



The graphic features a top section with a photograph of a hand holding a laser tool emitting red and green beams onto a circuit board. To the right, the text 'Z-LASER' is displayed in white on a dark blue background with a red horizontal line. Below this, the model name 'ZM18B' is centered in white. A list of technical specifications follows, each preceded by a right-pointing arrow. To the left of the specifications, a dark blue box lists various application areas. The central part of the graphic shows a close-up of the ZM18B laser device, which is a small, cylindrical tool with a black body and a silver-colored tip, mounted on a white surface. At the bottom, a row of four small images illustrates the laser's use on different materials: metal rods, wood logs, a tire tread, and a textured fabric. In the bottom right corner, there is a circular logo for TUV SUD ISO 9001.

# Z-LASER

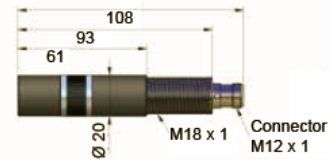
## ZM18B

- » Red or infrared wavelengths available
- » Optical output power up to 120mW / 160mW IR
- » Gaussian/uniform line, dot optics and various DOE optics available
- » 5 to 30VDC supply voltage with reverse polarity protection
- » Simple, external hand focusing mechanism
- » IP67 rated, water-proof and dust-proof
- » M18 thread mounted for simple and versatile mounting
- » LED laser operation indicator
- » Overvoltage protection with surge/spike protection

Machine Vision  
Automotive  
Wood  
Food & Beverage  
Medical  
Metal  
Stone  
Textile / Leather



# ZM18B



Mechanical specifications	
Dimensions	108mm x Ø 20mm (focusable version) / 92mm x Ø 20mm (fixed focus)
Housing	M18 industry housing, chromed brass Optic head: Anodised aluminium
IP rating	IP67
Weight	75g (focusable version)
Electrical isolation	Potential-free housing
Connection	M12 plug, 4-pin
Electrical specifications	
Supply voltage	5 to 30VDC
Operation mode	APC with current limiting
Modulation	No
Protection	Reverse polarity and transient protection / ESD, over temperature protection and LED pre-failure indicator
Optical specifications	
Wavelength	635nm, 640nm, 643nm, 685nm, 785nm, 808nm, 830nm, 850nm, 980nm
Output power	Up to 120mW / up to 160mW IR
Wavelength vs. temperature	Typ. 0.20 - 0.30nm / °C depending on wavelength
Power stability	±3% over operating temperature range
Focus range	100mm up to ∞
Pointing stability	< 15µrad / °C
Line (Gaussian profile)	3°, 5°, 10°, 15°, 20°, 30°, 90°, symmetrical or asymmetrical
Line (homogeneous intensity profile)	10°, 20°, 30°, 45°, 60°, 75°, 90°
Dot	Elliptical or circular
Environmental conditions	
Case temperature	-10°C up to +50°C (heat dissipation e.g. with mounting H8-M18)
Storage temperature	-10°C up to +80°C
Humidity	Max. 90%, non-condensing
MTTF at 25°C	> 30,000h (635nm to 685nm), > 100,000h (785nm to 980nm)

CE-Conformity according to the directives 2004/108/EC and 73/23/ECC excluding connection type.

<p><b>Accessories</b></p>	<p><b>Gaussian profile</b></p>	<p><b>Optics</b></p>
	<p><b>Homogeneous profile</b></p>	<p><b>Order code</b></p> <p>Z X M18B - X - X - X</p> <p>Power   Wavelength   Optic F = hand focusable without F = fixed focus</p> <p>Product family name</p>

© Z-LASER / Merzhauser Str. 134, 79100 Freiburg / Germany / Tel: +49 761 2964444 / info@z-laser.de / www.Z-LASER.com  
July 2013 / Subject to change



**Z-LASER**

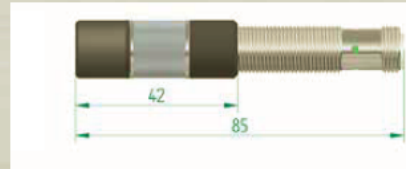
**ZM12B**

- » Red or infrared wavelengths available
- » Optical output power up to 80mW
- » 5 to 30VDC supply voltage with reverse polarity protection
- » Simple, external hand focusing mechanism
- » Compact M12 thread mounted for simple and versatile mounting
- » Potential-free housing

Machine Vision  
Automotive  
Wood  
Food & Beverage  
Metal  
Stone  
Textile



# ZM12B



Mechanical specifications	
Dimensions	85mm x Ø 15mm (focusable version) / 81mm x Ø 15mm (fixed focus)
Housing	M12 industry housing, chromed brass, Optic head: Anodised aluminium
IP rating	IP67
Weight	67g (fixed focus)
Electrical isolation	Potential-free housing
Connection	M12 plug, 4-pin
Electrical specifications	
Supply voltage	5 to 30VDC
Operation mode	APC with current limiting
Modulation	No
Protection	Reverse polarity and transient protection / ESD, over temperature protection and LED pre-failure indicator
Optical specifications	
Wavelength	635nm, 640nm, 643nm, 660nm, 785nm, 830nm, 850nm
Output power	Up to 80mW
Wavelength vs. temperature	Typ. 0.20 - 0.30nm / °C depending on wavelength
Power stability	± 3% over operating temperature range
Focus range	100mm up to 10.000mm
Pointing stability	< 15µrad / °C
Boresight error	< ± 10mrad
Line (Gaussian profile)	5°, 10°, 15°, 20°, 30°, 90°, symmetrical
Line (homogeneous intensity profile)	10°, 20°, 30°, 40°, 45°, 60°
Dot	Elliptical or circular
Environmental conditions	
Case temperature	-10°C up to +50°C (heat dissipation e.g. with mounting H8-M12)
Storage temperature	-10°C up to +70°C
Humidity	Max. 90%, non-condensing
MTTF at 25°C	> 30.000h (635nm up to 660nm), >100.000h (785nm up to 850nm)

CE-Conformity according to the directives 2004/108/EC and 73/23/ECC excluding connection type.

Accessories	Gaussian profile	Optics		
	<th>Homogeneous profile</th> <td> <th>Order code</th> </td>	Homogeneous profile	<th>Order code</th>	Order code
		<p>Z X M12B - X - X - X</p> <p>Power   Wavelength</p> <p>F = hand focusable without F = fixed focus</p> <p>Product family name</p>		

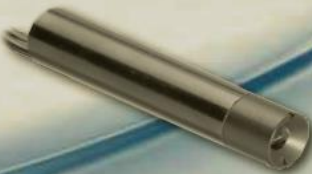
© Z-LASER / Merzhauser Str. 134, 79100 Freiburg / Germany / Tel: +49 761 2964444 / info@z-laser.de / www.Z-LASER.com  
July 2013 / Subject to change

Bijlage E: datasheet ZD



- » Universal mini laser module for line, cross and dot projection
- » Optical output power up to 15mW, red
- » 3 to 6VDC supply voltage (optionally: 24VDC) with reverse polarity protection
- » Dimensions: Ø 11mm x 52mm

- Wood
- Textile
- Metal



# ZD



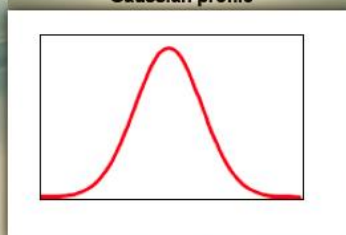
Mechanical specifications	
Dimensions	52mm x Ø 11mm
Housing	Brass, black chromed
IP rating	IP40
Electrical isolation	Potential-free housing
Connection	2m cable with Texas connector Optionally: Texas socket or cable up to 2m with open leads
Electrical specifications	
Supply voltage	3 to 6VDC (optionally: 24VDC)
Operation mode	APC with current limiting
Modulation	No
Protection	Reverse polarity protection
Optical specifications	
Wavelength	635nm, 650nm
Output power	Up to 15mW
Adjustable focus	No
Available optics	line Gaussian, dot elliptical, DOE cross
Environmental conditions	
Case temperature	-10°C to +40°C

CE-Conformity according to the directives 2004/108/EC and 73/23/ECC excluding connection type.

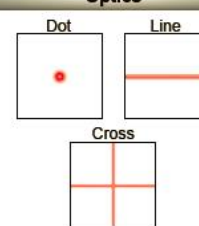
## Accessories



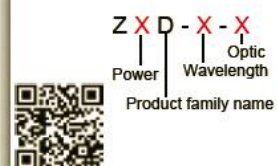
## Gaussian profile



## Optics



## Order code





## Bijlage F: Complete programmacode in HALCON

```
* *****
* *****Kwaliteitscontrole steenstrippen Vandersanden Group nv.*****
* *****
* *****Auteurs: Frederik Scheelen & Wouter Ferson*****
* *****
*
* Framegrabber sluiten + instellingen + parameterdeclaraties
*
close_framegrabber (AcqHandle)
dev_update_window ('off')
laserHoek := 58.6
*
* hoek omzetten in radialen
hoek := (2*3.14)*laserHoek/360
*
schaallengtePixel := 1054.0
schaallengte := 150.0
pixelToMm := schaallengte/schaallengtePixel
SpievormigheidseisHoogteWaarde := 4
dev_set_colored (12)
dev_set_line_width (1)
dev_set_shape ('original')
dev_set_lut ('default')
dev_set_paint (['default'])
dev_set_window_extents (0, 0, 669, 278)
dev_set_part (0, 0, 1535, 2047)
CameraParameters :=
[0.0130995, -19.3093, 3.2011e-006, 3.2e-006, 1143.27, 811.703, 2048, 1536]
CameraPose := [-0.0266767, -0.010406, 0.57925, 0.880209, 1.25083, 0.909382, 0]
*
* Framegrabber openen en beeld nemen
*
open_framegrabber ('uEye', 1, 1, 0, 0, 0, 0, 'default', 8, 'default', -1,
'false', 'default', '4', 0, -1, AcqHandle)
set_framegrabber_param (AcqHandle, 'frame_rate', 5.60689)
set_framegrabber_param (AcqHandle, 'gain_boost', 'enable')
set_framegrabber_param (AcqHandle, 'gain_factor_g', 150)
grab_image_start (AcqHandle, -1)
while (true)
  firstTrigger := 0
  gevonden := 0
  while (firstTrigger=0)
    objectTrigger := 0
    while (objectTrigger=0)
      grab_image_async (Image, AcqHandle, -1)
      *
      * Distorsie wegnemen uit het beeld
      *
      change_radial_distortion_cam_par ('fullsize', CameraParameters, 0,
CamParOut)
      change_radial_distortion_image (Image, Image, Image, CameraParameters,
CamParOut)
      *
      * Steenstrip uit beeld halen + beeld rechtzetten
      *
      get_image_size (Image, Beeldbreedte, Beeldhoogte)
      decompose3 (Image, R, G, B)
      sub_image (G, R, ImageSub, 1, 1)
      threshold (ImageSub, Regions, 0, 50)
```

```

connection (Regions, ConnectedRegions)
select_shape (ConnectedRegions, SelectedRegions, 'area', 'and', 200000,
700000)
count_obj (SelectedRegions, aantalobj)
area_center (SelectedRegions, Areal, Row1, Column1)
if (aantalobj>1)
    if (Row1[0]>=(Beeldhoogte/2) and Row1[0]<=((Beeldhoogte/2)+100) and
Row1[1]>=(Beeldhoogte-400))
        select_obj (SelectedRegions, ObjectSelected, 1)
        objectTrigger := 1
    else
        select_obj (SelectedRegions, ObjectSelected, 2)
        objectTrigger := 1
    endif
elseif (aantalobj=1)
    if (Row1[0]>=(Beeldhoogte/2) and Row1[0]<=((Beeldhoogte/2)+300))
        select_obj (SelectedRegions, ObjectSelected, 1)
        objectTrigger := 1
    endif
endif
endif
endwhile
region_features (ObjectSelected, 'area', Value)
area_center (ObjectSelected, Areal, Row1, Column1)
if (200000<=Value and Value<=700000 and firstTrigger=0)
    gevonden := 1
else
    gevonden := 0
endif
endif
if (Row1>=(Beeldhoogte/2) and Row1<=((Beeldhoogte/2)+300) and gevonden=1)
    firstTrigger := 1
else
    firstTrigger := 0
endif
endif
endwhile
dilation_circle (ObjectSelected, RegionDilation, 20)
connection (RegionDilation, ConnectedRegions2)
erosion_circle (ConnectedRegions2, RegionErosion, 20)
reduce_domain (Image, RegionErosion, Baksteenstrip)
orientation_region (Baksteenstrip, Phi)
Hoek_graden := (360/(2*3.14))*Phi
if (Hoek_graden>90)
    Hoek_graden := Hoek_graden-180
elseif (Hoek_graden<-90)
    Hoek_graden := Hoek_graden+180
endif
rotate_image (Baksteenstrip, ImageRotate, -Hoek_graden, 'constant')
decompose3 (ImageRotate, R, G, B)
sub_image (G, R, ImageSub, 1, 1)
threshold (ImageSub, Regions, 0, 50)
connection (Regions, ConnectedRegions)
select_shape (ConnectedRegions, SelectedRegions, 'area', 'and', 200000,
700000)
area_center (SelectedRegions, Areal, Row1, Column1)
if (aantalobj>1)
    if (Row1[0]>=(Beeldhoogte/2) and Row1[0]<=((Beeldhoogte/2)+300) and
Row1[1]>=(Beeldhoogte-400))
        select_obj (SelectedRegions, ObjectSelected, 1)
    else
        select_obj (SelectedRegions, ObjectSelected, 2)
    endif
elseif (aantalobj=1)
    if (Row1[0]>=(Beeldhoogte/2) and Row1[0]<=((Beeldhoogte/2)+300))
        select_obj (SelectedRegions, ObjectSelected, 1)
    endif
endif
endif
dilation_circle (ObjectSelected, RegionDilation, 20)
connection (RegionDilation, ConnectedRegions2)
erosion_circle (ConnectedRegions2, Baksteenstrip2, 20)
smallest_rectangle2 (Baksteenstrip2, Row, Column, Phi2, Length1, Length2)

```

```

dev_clear_window ()
dev_display (Baksteenstrip2)
dev_set_draw ('margin')
*
* Vormcontrole, lengte- en breedtemetingen, hoekcontrole
*
area_center (Baksteenstrip2, Area, R1, C1)
get_region_contour (Baksteenstrip2, Rows, Columns)
striplengte := (Length1*2)*pixelToMm
stripbreedte := (Length2*2)*pixelToMm
rectangularity (Baksteenstrip2, Rechthoekigheid)
vormEisOpp := 40000
OppervlakteOrig := (Length1*2)*(Length2*2)
if (((OppervlakteOrig - vormEisOpp)<Area) and Rechthoekigheid>=90)
  contourInkeping := true
else
  contourInkeping := false
endif
tuple_min (Rows, Rmin)
tuple_max (Rows, Rmax)
tuple_min (Columns, Cmin)
tuple_max (Columns, Cmax)
gen_rectangle1 (corner1, Rmin-20, Cmin-20, Rmin+100, Cmin+100)
area_center (corner1, AreaC1, CenterRowCorner1, CenterColumnCorner1)
reduce_domain (Baksteenstrip2, corner1, cornerZoom1)
smallest_rectangle2 (cornerZoom1, x1, y1, phi1, W1, H1)
gen_rectangle2 (co1, x1, y1, phi1, W1, H1)
rectangularity (cornerZoom1, rect1)
gen_rectangle1 (corner2, Rmax-100, Cmin-20, Rmax+20, Cmin+100)
area_center (corner2, AreaC2, CenterRowCorner2, CenterColumnCorner2)
reduce_domain (Baksteenstrip2, corner2, cornerZoom2)
smallest_rectangle2 (cornerZoom2, x2, y2, phi2, W2, H2)
gen_rectangle2 (co2, x2, y2, phi2, W2, H2)
rectangularity (cornerZoom2, rect2)
gen_rectangle1 (corner3, Rmin-20, Cmax-100, Rmin+100, Cmax+20)
area_center (corner3, AreaC3, CenterRowCorner3, CenterColumnCorner3)
reduce_domain (Baksteenstrip2, corner3, cornerZoom3)
smallest_rectangle2 (cornerZoom3, x3, y3, phi3, W3, H3)
gen_rectangle2 (co3, x3, y3, phi3, W3, H3)
rectangularity (cornerZoom3, rect3)
gen_rectangle1 (corner4, Rmax-100, Cmax-100, Rmax+20, Cmax+20)
area_center (corner4, AreaC4, CenterRowCorner4, CenterColumnCorner4)
reduce_domain (Baksteenstrip2, corner4, cornerZoom4)
smallest_rectangle2 (cornerZoom4, x4, y4, phi4, W4, H4)
gen_rectangle2 (co4, x4, y4, phi4, W4, H4)
rectangularity (cornerZoom4, rect4)
tupleRec := [rect1,rect2,rect3,rect4]
cornerCorrect := true
for i := 0 to |tupleRec| - 1 by 1
  if (cornerCorrect=true)
    if (tupleRec[i]<0.65)
      cornerCorrect := false
    elseif (tupleRec[i]>=0.65)
      cornerCorrect := true
    endif
  endif
endif
endfor
*
* *****Hoogtemeting + spievormigheidscontrole*****
*
* Laserlijn uit beeld halen + groeperen in 1, 2 of 3 zichtbare lijnen
*
if ((contourInkeping=true) or (cornerCorrect=true))
  dev_clear_window ()
  dev_set_draw ('fill')
  secondTrigger := 0
  thirdTrigger := 0
  cyclus := 0
  tuple_gen_const (0, 0, Left)

```

```

tuple_gen_const (0, 0, Right)
tuple_gen_const (0, 0, GemHoogtes)
tuple_gen_const (0, 0, GroupLinks)
tuple_gen_const (0, 0, GroupRechts)
tuple_gen_const (0, 0, GroupsLinksMean)
tuple_gen_const (0, 0, GroupsRechtsMean)
while (secondTrigger=0)
  grab_image_async (ImageLine, AcqHandle, -1)
  decompose3 (ImageLine, R2, G2, B2)
  threshold (R2, lijn, 215, 255)
  dilation_circle (lijn, lijnverdikking, 20)
  connection (lijnverdikking, dikkelijnen)
  erosion_circle (dikkelijnen, lijnen, 20)
  fill_up (lijnen, volleLijnen)
  select_shape (volleLijnen, lines, 'area', 'and', 500, 104105)
  dev_clear_window ()
  dev_display (lines)
  count_obj (lines, Number)
  *
  * *****Zijwaartse spievormcontrole*****
  *
  if (Number=1 or Number=2 or thirdTrigger=1)
    tuple_length (Left, LeftLength)
    tuple_length (Right, RightLength)
    select_obj (lines, midden, 1)
    get_region_points (midden, RijMidden, KolMidden)
    if (LeftLength>0 and RightLength>0)
      for i := 1 to 5 by 1
        for y := ((LeftLength-1)*(i-1)/5) to ((LeftLength-1)*i/5) by 1
          tuple_replace (GroupLinks, y-((LeftLength-1)*(i-1)/5), Left[y],
GroupLinks)
        endfor
        tuple_mean (GroupLinks, LinksMean)
        tuple_insert (GroupsLinksMean, i-1, LinksMean, GroupsLinksMean)
        for z := ((RightLength-1)*(i-1)/5) to ((RightLength-1)*i/5) by 1
          tuple_replace (GroupRechts, z-((RightLength-1)*(i-1)/5), Right[z],
GroupRechts)
        endfor
        tuple_mean (GroupRechts, RechtsMean)
        tuple_insert (GroupsRechtsMean, i-1, RechtsMean, GroupsRechtsMean)
        endfor
        tuple_length (GroupsLinksMean, lengthLgem)
        tuple_length (GroupsRechtsMean, lengthRgem)
        tuple_mean (Left, LeftMean)
        tuple_mean (Right, RightMean)
        try
          if (Left[0]>(Left[LeftLength-1]+SpievormigheidseisHoogteWaarde) or
(Left[0]+SpievormigheidseisHoogteWaarde)<Left[LeftLength-1])
            GoedeStrip := 0
          elseif
(Right[0]>(Right[RightLength-1]+SpievormigheidseisHoogteWaarde) or
(Right[0]+SpievormigheidseisHoogteWaarde)<Right[RightLength-1])
            GoedeStrip := 0
          elseif (LeftMean<(Left[0]-SpievormigheidseisHoogteWaarde) or
RightMean<(Right[0]-SpievormigheidseisHoogteWaarde))
            GoedeStrip := 0
          elseif
(((GroupsLinksMean[0]<(GroupsLinksMean[1]-1)) and (GroupsLinksMean[1]<(GroupsLinksM
ean[2]-1)) and (GroupsLinksMean[2]<(GroupsLinksMean[3]-1)) and (GroupsLinksMean[3]<(
GroupsLinksMean[4]-1)))
            GoedeStrip := 0
          elseif
(((GroupsLinksMean[0]>(GroupsLinksMean[1]+1)) and (GroupsLinksMean[1]>(GroupsLinksM
ean[2]+1)) and (GroupsLinksMean[2]>(GroupsLinksMean[3]+1)) and (GroupsLinksMean[3]>(
GroupsLinksMean[4]+1)))
            GoedeStrip := 0
          elseif
(((GroupsRechtsMean[0]>(GroupsRechtsMean[1]+1)) and (GroupsRechtsMean[1]>(GroupsRec
htsMean[2]+1)) and (GroupsRechtsMean[2]>(GroupsRechtsMean[3]+1)) and (GroupsRechtsMe

```

```

an[3]>(GroupsRechtsMean[4]+1))
    GoedeStrip := 0
  elseif
    ((GroupsRechtsMean[0]>(GroupsRechtsMean[1]+1)) and (GroupsRechtsMean[1]>(GroupsRechtsMean[2]+1)) and (GroupsRechtsMean[2]>(GroupsRechtsMean[3]+1)) and (GroupsRechtsMean[3]>(GroupsRechtsMean[4]+1)))
    GoedeStrip := 0
  else
    GoedeStrip := 1
  endif
  catch (Exception1)
  endtry
endif
tuple_remove (Left, [0:LeftLength-1], Left)
tuple_remove (Right, [0:RightLength-1], Right)
tuple_length (Left, LeftLength)
tuple_length (Right, RightLength)
secondTrigger := 1
thirdTrigger := 0
*
* *****
*
* Hoogtemeting en spievormcontrole in langsrichting
*
elseif (Number=3)
  select_obj (lines, links, 2)
  get_region_points (links, RijLinks, KolLinks)
  select_obj (lines, rechts, 3)
  get_region_points (rechts, RijRechts, KolRechts)
  select_obj (lines, midden, 1)
  get_region_points (midden, RijMidden, KolMidden)
  tuple_sort (KolLinks, sortKL)
  tuple_sort (KolRechts, sortKR)
  tuple_median (RijLinks, MediaanLinksR)
  tuple_median (sortKL, MediaanLinksK)
  tuple_median (sortKR, MediaanRechtsK)
  tuple_median (RijRechts, MediaanRechtsR)
  tuple_length (KolMidden, ElementenMiddellijn)
  tuple_length (KolLinks, ElementenLinks)
  tuple_length (KolRechts, ElementenRechts)
  tuple_select (sortKL, ElementenLinks-1, EindkolomLinks)
  tuple_select (sortKR, 0, BeginkolomRechts)
  tuple_gen_const (0, 0, hoogtes)
  BeginkolomLinks := MediaanLinksK
  BeginrijLinks := MediaanLinksR
  EindrijLinks := MediaanLinksR
  EindkolomRechts := MediaanRechtsK
  EindrijRechts := MediaanRechtsR
  BeginrijRechts := MediaanRechtsR
  dev_set_color ('blue')
  gen_region_line (Verbindingslijn, BeginrijLinks, BeginkolomLinks,
EindrijRechts, EindkolomRechts)
  dev_set_color ('green')
  gen_region_line (Linkerlijn, BeginrijLinks, BeginkolomLinks,
EindrijLinks, EindkolomLinks)
  dev_set_color ('yellow')
  gen_region_line (Rechterlijn, BeginrijRechts, BeginkolomRechts,
EindrijRechts, EindkolomRechts)
  tuple_sort (KolMidden, KolMiddenSort)
  tuple_uniq (RijMidden, RijUniekSort)
  tuple_uniq (KolMiddenSort, KolUniekSort)
  tuple_length (KolUniekSort, KUlengte)
  tuple_length (RijUniekSort, RUlengte)
  tuple_gen_const (0, 0, RijMiddenNieuw)
  tuple_gen_const (0, 0, KolMiddenNieuw)
*
* *****Bepalen van de merkerpunten*****
*
for f := 0 to (KUlengte-1)/100 by 1

```

```

tuple_gen_const (0, 0, rijengroep)
for g := 0 to (Rulengte-1)/5 by 1
  tuple_length (rijengroep, rijengroeplengte)
  yCo := RijUniekSort[g*5]
  xCo := KolUniekSort[f*100]
  test_region_point (midden, yCo, xCo, IsInside)
  if (IsInside==1)
    tuple_replace (rijengroep, rijengroeplengte, yCo, rijengroep)
  endif
endfor
tuple_length (RijMiddenNieuw, RijMiddenNieuwLengte)
tuple_length (KolMiddenNieuw, KolMiddenNieuwLengte)
if (rijengroeplengte!=0)
  tuple_mean (rijengroep, rijengroepMean)
  tuple_insert (RijMiddenNieuw, RijMiddenNieuwLengte, rijengroepMean,
RijMiddenNieuw)
  tuple_insert (KolMiddenNieuw, KolMiddenNieuwLengte, xCo,
KolMiddenNieuw)
endif
endfor
*
* *****
*
tuple_length (KolMiddenNieuw, KolMiddenNieuwlengte)
tuple_length (RijMiddenNieuw, RijMiddenNieuwlengte)
for k := 0 to (KolMiddenNieuwlengte-1) by 1
  tuple_select (RijMiddenNieuw, k, PuntY)
  tuple_select (KolMiddenNieuw, k, PuntX)
  distance_pl (PuntY, PuntX, BeginrijLinks, BeginkolomLinks,
EindrijRechts, EindkolomRechts, afstand)
  afstandMM := afstand*pixelToMm
  hoogtePunt := tan(hoek)*afstandMM
  tuple_insert (hoogtes, k, hoogtePunt, hoogtes)
endfor
tuple_mean (hoogtes, MeanHoogtes)
tuple_insert (GemHoogtes, cyclus, MeanHoogtes, GemHoogtes)
tuple_length (hoogtes, ElementenHoogtes)
tuple_select (hoogtes, 0, ReferentieLinks)
tuple_select (hoogtes, ElementenHoogtes-1, ReferentieRechts)
tuple_insert (Left, cyclus, hoogtes[0], Left)
tuple_insert (Right, cyclus, hoogtes[ElementenHoogtes-1], Right)
cyclus := cyclus+1
tuple_greater_equal_elem (hoogtes, 17, Greaterreq)
tuple_mean (Greaterreq, equationMean)
if (equationMean>0.95 and equationMean<=1)
  GoedeStrip := 1
elseif (abs(ReferentieLinks-
ReferentieRechts)<=SpievormigheidseisHoogteWaarde)
  GoedeStrip := 1
else
  GoedeStrip := 0
endif
if (-5<=MeanHoogtes and MeanHoogtes<=10)
  thirdTrigger := 1
elseif (GoedeStrip=0)
  secondTrigger := 1
endif
endif
try
  tuple_mean (GemHoogtes, GemHoogteStrip)
catch (Exception)
endtry
endwhile
endif
endwhile
*
* *****
* *****
* *****

```

## Bijlage G: programmacode van form 1 in Visual Studio

```
Public Class Login_Scherm

    Public i As Integer = 1

    Public Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles LoginKnop.Click

        If Paswoord_Invoer.Text = "VDSG" Then
            If Naam_Invoer.Text = "VDS" Then

                Instellingen_Scherm.ThresholdOndergrens_Schuifbalk.Value = 0
                Instellingen_Scherm.ThresholdBovengrens_Schuifbalk.Value = 50
                Instellingen_Scherm.ThresholdOndergrensWaarde =
Instellingen_Scherm.ThresholdOndergrens_Schuifbalk.Value
                Instellingen_Scherm.ThresholdBovengrensWaarde =
Instellingen_Scherm.ThresholdBovengrens_Schuifbalk.Value
                Instellingen_Scherm.Threshold_OnderGrens.Text =
Instellingen_Scherm.ThresholdOndergrensWaarde
                Instellingen_Scherm.Threshold_BovenGrens.Text =
Instellingen_Scherm.ThresholdBovengrensWaarde

                Instellingen_Scherm.AreaOndergrens_Schuifbalk.Value = 300000
                Instellingen_Scherm.AreaBovengrens_Schuifbalk.Value = 700000
                Instellingen_Scherm.AreaOndergrensWaarde =
Instellingen_Scherm.AreaOndergrens_Schuifbalk.Value
                Instellingen_Scherm.AreaBovengrensWaarde =
Instellingen_Scherm.AreaBovengrens_Schuifbalk.Value
                Instellingen_Scherm.Area_OnderGrens.Text =
Instellingen_Scherm.AreaOndergrensWaarde
                Instellingen_Scherm.Area_BovenGrens.Text =
Instellingen_Scherm.AreaBovengrensWaarde

                Instellingen_Scherm.DunneStripIngesteld.ForeColor = Color.Black
                Instellingen_Scherm.DunneStripIngesteld.Font = New Font("Microsoft
Sans Serif", 12.25, FontStyle.Regular)
                Instellingen_Scherm.DunneStripIngesteld.Text = "Huidig ingesteld"

                Instellingen_Scherm.DikkeStripIngesteld.ForeColor = Color.Black
                Instellingen_Scherm.DikkeStripIngesteld.Text = "Niet ingesteld"

                Instellingen_Scherm.LaserHoek_Schuifbalk.Value = 59
                Instellingen_Scherm.LaserHoekWaarde =
Instellingen_Scherm.LaserHoek_Schuifbalk.Value
                Instellingen_Scherm.LaserHoek.Text =
Instellingen_Scherm.LaserHoekWaarde

                Instellingen_Scherm.Contoureis_Schuifbalk.Value = 94
                Instellingen_Scherm.ContoureisWaarde =
Instellingen_Scherm.Contoureis_Schuifbalk.Value
                Instellingen_Scherm.Contoureis.Text =
Instellingen_Scherm.ContoureisWaarde & " %"

                Instellingen_Scherm.Randeis_Schuifbalk.Value = 65
                Instellingen_Scherm.RandeisWaarde =
Instellingen_Scherm.Randeis_Schuifbalk.Value
                Instellingen_Scherm.Randeis.Text = Instellingen_Scherm.RandeisWaarde &
" %"

            End If
        End If
    End Sub
End Class
```

```

        Instellingen_Scherm.Spievormigheidseis_Schuifbalk.Value = 95
        Instellingen_Scherm.SpievormigheidseisWaarde =
Instellingen_Scherm.Spievormigheidseis_Schuifbalk.Value
        Instellingen_Scherm.Spievormigheidseis.Text =
Instellingen_Scherm.SpievormigheidseisWaarde & " %"

        Instellingen_Scherm.SpievormigheidseisHoogte_Schuifbalk.Value = 4
        Instellingen_Scherm.SpievormigheidseisHoogteWaarde =
Instellingen_Scherm.SpievormigheidseisHoogte_Schuifbalk.Value
        Instellingen_Scherm.SpievormigheidseisHoogte.Text =
Instellingen_Scherm.SpievormigheidseisHoogteWaarde & " mm"

        Instellingen_Scherm.vormEisOpp_Schuifbalk.Value = 40000
Instellingen_Scherm.vormEisOppWaarde = Instellingen_Scherm.vormEisOpp_Schuifbalk.Value
        Instellingen_Scherm.vormEisOpp.Text =
Instellingen_Scherm.vormEisOppWaarde

        Instellingen_Scherm.schaallengtepixel = 1056.0

        VisieControle_Scherm.Show()
        Me.Hide()

    Else
        MsgBox("Error! Foute login gegevens", MsgBoxStyle.Critical)
    End If
Else
    MsgBox("Error! Foute login gegevens", MsgBoxStyle.Critical)
End If

End Sub

Private Sub PictureBox1_Click(sender As System.Object, e As System.EventArgs)
Handles PictureBox1.Click

    End Sub
End Class

```



## Bijlage H: programmacode van form 2 in Visual Studio

```
Option Strict Off
```

```
Option Explicit On
```

```
Imports HalconDotNet
```

```
Imports System
```

```
Imports Microsoft.VisualBasic
```

```
Imports System.Windows.Forms.VisualStyleElement
```

```
Public Class VisieControle_Scherm
```

```
    Private HWindow As Object
```

```
    Dim acqHandle As HalconDotNet.HTuple
```

```
    Dim hv_AcqHandle As New HTuple
```

```
    Dim hv_ExpDefaultWinHandle As HalconDotNet.HTuple
```

```
    Dim Window As HalconDotNet.HWindow
```

```
    Dim hv_schaallengtePixel As HTuple = Nothing
```

```
    Dim hv_schaalLengte As HTuple = Nothing
```

```
    Dim hv_pixelToMm As HTuple = Nothing
```

```
    Dim hv_firstTrigger As HTuple = Nothing
```

```
    Dim hv_gevonden As HTuple = Nothing
```

```
    Dim hv_objectTrigger As HTuple = Nothing
```

```
    Dim hv_Beeldbreedte As HTuple = Nothing, hv_Beeldhoogte As HTuple = Nothing
```

```
    Dim ho_R As HObject = Nothing
```

```
    Dim ho_G As HObject = Nothing
```

```
    Dim ho_B As HObject = Nothing
```

```
    Dim ho_image As HObject = Nothing
```

```
    Dim ho_imageSub As HObject = Nothing
```

```
    Dim ho_Region As HObject = Nothing
```

```
    Dim ho_ConnectedRegions As HObject = Nothing
```

```
    Dim ho_SelectedRegions As HObject = Nothing
```

```
    Dim ho_ObjectSelectedOut As HObject = Nothing
```

```
    Dim ho_Baksteenstrip2 As HObject = Nothing
```

```
    Dim ho_ConnectedRegions2 As HObject = Nothing
```

```
    Dim hv_Rechthoekigheid As HTuple = Nothing
```

```
    Dim ho_ObjectSelected As HObject = Nothing
```

```
    Dim hv_aantalObj As HTuple = Nothing
```

```
    Dim hv_Area As HTuple = Nothing, hv_Row As HTuple = Nothing, hv_Column As HTuple = Nothing
```

```
    Dim hv_Area1 As HTuple = Nothing, hv_Row1 As HTuple = Nothing, hv_Column1 As HTuple = Nothing
```

```
    Dim ho_RegionErosion As HObject = Nothing
```

```
    Dim ho_RegionDilation As HObject = Nothing
```

```
    Dim hv_Hoek_graden As HTuple = Nothing
```

```
    Dim hv_Value As HTuple = Nothing
```

```
    Dim ho_ImageRotate As HObject = Nothing
```

```
    Dim hv_R1 As HTuple = Nothing, hv_C1 As HTuple = Nothing
```

```
    Dim ho_Regions As HObject = Nothing
```

```
    Dim ho_Baksteenstrip As HObject = Nothing
```

```
    Dim hv_Length1 As HTuple = Nothing
```

```
    Dim hv_Length2 As HTuple = Nothing
```

```
    Dim hv_Rows As HTuple = Nothing
```

```
    Dim hv_Columns As HTuple = Nothing
```

```
    Dim hv_striplengte As HTuple = Nothing
```

```
    Dim hv_stripbreedte As HTuple = Nothing
```

```

Dim hv_Phi As HTuple = Nothing
Dim hv_OppervlakteOrig As HTuple = Nothing
Dim hv_contourInkeping As HTuple = Nothing
Dim hv_Rmin As HTuple = Nothing
Dim hv_Rmax As HTuple = Nothing
Dim hv_Cmin As HTuple = Nothing
Dim hv_Cmax As HTuple = Nothing
Dim ho_corner1 As HObject = Nothing, ho_corner2 As HObject = Nothing, ho_corner3
As HObject = Nothing, ho_corner4 As HObject = Nothing
Dim hv_AreaC1 As HTuple = Nothing, hv_AreaC2 As HTuple = Nothing, hv_AreaC3 As
HTuple = Nothing, hv_AreaC4 As HTuple = Nothing
Dim hv_centerRowCorner1 As HTuple = Nothing, hv_CenterRowCorner2 As HTuple =
Nothing, hv_CenterRowCorner3 As HTuple = Nothing, hv_CenterRowCorner4 As HTuple =
Nothing
Dim hv_CenterColumnCorner1 As HTuple = Nothing, hv_CenterColumnCorner2 As HTuple =
Nothing, hv_CenterColumnCorner3 As HTuple = Nothing, hv_CenterColumnCorner4 As HTuple
= Nothing
Dim ho_cornerZoom1 As HObject = Nothing, ho_cornerZoom2 As HObject = Nothing,
ho_cornerZoom3 As HObject = Nothing, ho_cornerZoom4 As HObject = Nothing
Dim ho_cornerZoom1m As HObject = Nothing, ho_cornerZoom2m As HObject = Nothing,
ho_cornerZoom3m As HObject = Nothing, ho_cornerZoom4m As HObject = Nothing
Dim ho_co1 As HObject = Nothing, ho_co2 As HObject = Nothing, ho_co3 As HObject =
Nothing, ho_co4 As HObject = Nothing
Dim hv_x1 As HTuple = Nothing, hv_x2 As HTuple = Nothing, hv_x3 As HTuple =
Nothing, hv_x4 As HTuple = Nothing
Dim hv_y1 As HTuple = Nothing, hv_y2 As HTuple = Nothing, hv_y3 As HTuple =
Nothing, hv_y4 As HTuple = Nothing
Dim hv_phi1 As HTuple = Nothing, hv_phi2 As HTuple = Nothing, hv_phi3 As HTuple =
Nothing, hv_phi4 As HTuple = Nothing
Dim hv_W1 As HTuple = Nothing, hv_W2 As HTuple = Nothing, hv_W3 As HTuple =
Nothing, hv_W4 As HTuple = Nothing
Dim hv_H1 As HTuple = Nothing, hv_H2 As HTuple = Nothing, hv_H3 As HTuple =
Nothing, hv_H4 As HTuple = Nothing
Dim hv_rect1 As HTuple = Nothing, hv_rect2 As HTuple = Nothing, hv_rect3 As HTuple
= Nothing, hv_rect4 As HTuple = Nothing
Dim hv_tupleRec As HTuple = Nothing
Dim hv_cornerCorrect As HTuple = Nothing
Dim hv_secondTrigger As HTuple = Nothing
Dim hv_thirdTrigger As HTuple = Nothing
Dim hv_cyclus As HTuple = Nothing
Dim hv_Left As HTuple = Nothing
Dim hv_Right As HTuple = Nothing
Dim hv_GroupLinks As HTuple = Nothing
Dim hv_GroupRechts As HTuple = Nothing
Dim hv_GroupsLinksMean As HTuple = Nothing
Dim hv_GroupsRechtsMean As HTuple = Nothing
Dim ho_ImageLine As HObject = Nothing
Dim ho_R2 As HObject = Nothing
Dim ho_G2 As HObject = Nothing
Dim ho_B2 As HObject = Nothing
Dim ho_lijn As HObject = Nothing
Dim ho_lijnverdikking As HObject = Nothing
Dim ho_dikkelijnen As HObject = Nothing
Dim ho_lijnen As HObject = Nothing
Dim ho_volleLijnen As HObject = Nothing
Dim ho_lines As HObject = Nothing
Dim hv_Number As HTuple = Nothing
Dim hv_LeftLength As HTuple = Nothing
Dim hv_RightLength As HTuple = Nothing
Dim ho_midden As HObject = Nothing
Dim hv_RijMidden As HTuple = Nothing
Dim hv_KolMidden As HTuple = Nothing

```

```

Dim hv_LinksMean As HTuple = Nothing
Dim hv_RechtsMean As HTuple = Nothing
Dim hv_lengthLgem As HTuple = Nothing
Dim hv_lengthRgem As HTuple = Nothing
Dim hv_LeftMean As HTuple = Nothing
Dim hv_RightMean As HTuple = Nothing
Dim hv_GoedeStrip As HTuple = Nothing
Dim ho_links As HObject = Nothing
Dim ho_rechts As HObject = Nothing
Dim hv_rijLinks As HTuple = Nothing
Dim hv_rijRechts As HTuple = Nothing
Dim hv_KolLinks As HTuple = Nothing
Dim hv_KolRechts As HTuple = Nothing
Dim hv_sortKL As HTuple = Nothing
Dim hv_sortKR As HTuple = Nothing
Dim hv_MediaanLinksR As HTuple = Nothing
Dim hv_MediaanRechtsK As HTuple = Nothing
Dim hv_MediaanLinksK As HTuple = Nothing
Dim hv_MediaanRechtsR As HTuple = Nothing
Dim hv_ElementenMiddellijn As HTuple = Nothing
Dim hv_ElementenLinks As HTuple = Nothing
Dim hv_ElementenRechts As HTuple = Nothing
Dim hv_EindkolomLinks As HTuple = Nothing
Dim hv_BeginKolomRechts As HTuple = Nothing
Dim hv_hoogtes As HTuple = Nothing
Dim hv_BeginKolomLinks As HTuple = Nothing
Dim hv_BeginrijLinks As HTuple = Nothing
Dim hv_EindrijLinks As HTuple = Nothing
Dim hv_EindkolomRechts As HTuple = Nothing
Dim hv_EindrijRechts As HTuple = Nothing
Dim hv_BeginrijRechts As HTuple = Nothing
Dim ho_Verbindingslijn As HObject = Nothing
Dim ho_Linkerlijn As HObject = Nothing
Dim ho_Rechterlijn As HObject = Nothing
Dim hv_KolMiddenSort As HTuple = Nothing
Dim hv_RijUniekSort As HTuple = Nothing
Dim hv_KolUniekSort As HTuple = Nothing
Dim hv_KUlengthe As HTuple = Nothing
Dim hv_RUlengthe As HTuple = Nothing
Dim hv_RijMiddenNieuw As HTuple = Nothing
Dim hv_KolMiddenNieuw As HTuple = Nothing
Dim hv_rijengroep As HTuple = Nothing
Dim hv_rijengroepLengte As HTuple = Nothing
Dim hv_yCo As HTuple = Nothing
Dim hv_xCo As HTuple = Nothing
Dim hv_IsInside As HTuple = Nothing
Dim hv_RijMiddenNieuwLengte As HTuple = Nothing
Dim hv_KolMiddenNieuwLengte As HTuple = Nothing
Dim hv_rijengroepMean As HTuple = Nothing
Dim hv_PuntX As HTuple = Nothing
Dim hv_PuntY As HTuple = Nothing
Dim hv_afstandCM As HTuple = Nothing
Dim hv_afstand As HTuple = Nothing
Dim hv_hoogtePunt As HTuple = Nothing
Dim hv_MeanHoogtes As HTuple = Nothing
Dim hv_ElementenHoogtes As HTuple = Nothing
Dim hv_ReferentieLinks As HTuple = Nothing
Dim hv_ReferentieRechts As HTuple = Nothing
Dim hv_Greatereq As HTuple = Nothing
Dim hv_equationMean As HTuple = Nothing
Dim hv_GemHoogtes As HTuple = Nothing
Dim hv_ElementenGemHoogtes As HTuple = Nothing

```

```

Dim hv_GemHoogteStrip As HTuple = Nothing
Dim hv_CameraParameters As HTuple = Nothing
Dim hv_CamParOut As HTuple = Nothing
Dim hv_Exception As HTuple = Nothing
Dim hv_Exception1 As HTuple = Nothing
Dim hv_CameraPose As HTuple = Nothing
Dim hv_hoek As HTuple = Nothing
Dim AantalGoedeStrippen As Integer = Nothing

Dim stripbreedte As Double
Dim striplengte As Double
Dim striphoogte As Double
Dim MagWerken As Boolean = False

Dim hv_WindowHandle As HTuple
Dim framegrabber As HFramegrabber

Private sql As DatabaseConnectie

Private Sub RUNknop_Click_1(sender As System.Object, e As System.EventArgs)
Handles RUNknop.Click
    Timer1.Start()
End Sub

Private Sub STOPknop_Click_1(sender As System.Object, e As System.EventArgs)
Handles STOPknop.Click
    Timer1.Stop()
End Sub

Private Sub Timer1_Tick_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
    'hoek omzetten in radialen
    HWindowControl1.HalconWindow.ClearWindow()
    LengteStrip_Invoer.Text = ""
    BreedteStrip_Invoer.Text = ""
    Vorm_Invoer.Text = ""
    Rand_Invoer.Text = ""
    Spievorming_Invoer.Text = ""
    HoogteStrip_Invoer.Text = ""
    striplengte = 0
    striphoogte = 0
    stripbreedte = 0
    hv_GemHoogteStrip = New HTuple(0)
    hv_stripbreedte = New HTuple(0)
    hv_striplengte = New HTuple(0)
    hv_cornerCorrect = New HTuple(0)
    hv_contourInkeping = New HTuple(0)
    hv_GoedeStrip = New HTuple(0)
    hv_hoek = ((((((New HTuple(2)).TupleMult(New
HTuple(3.14))))).TupleMult(Instellingen_Scherm.LaserHoekWaarde))).TupleDiv(New
HTuple(360))
    hv_schaallengtePixel = New HTuple(Instellingen_Scherm.schaallengtepixel)
    hv_schaalLengte = New HTuple(150.0)
    hv_pixelToMm = hv_schaalLengte.TupleDiv(hv_schaallengtePixel)
    hv_CameraParameters = (((((((New HTuple(0.0130995)).TupleConcat(-
19.3093)).TupleConcat( _
0.0000032011)).TupleConcat(0.0000032)).TupleConcat(1143.27)).TupleConcat(811.703)).Tup
leConcat(2048)).TupleConcat(1536)
    hv_CameraPose = (((((((New HTuple(-0.0266767)).TupleConcat(-
0.010406)).TupleConcat( _

```

```
0.57925)).TupleConcat(0.880209)).TupleConcat(1.25083)).TupleConcat(0.909382)).TupleConcat(0)
```

```

    If (MagWerken = False) Then
        HOperatorSet.OpenFramegrabber(New HTuple("uEye"), New HTuple(1), New
HTuple(1), _
        New HTuple(0), New HTuple(0), New HTuple(0), New HTuple(0), New
HTuple("default"), _
        New HTuple(8), New HTuple("default"), New HTuple(-1), New
HTuple("false"), _
        New HTuple("default"), New HTuple("4"), New HTuple(0), New HTuple(-1),
hv_AcqHandle)
        HOperatorSet.SetFramegrabberParam(hv_AcqHandle, New HTuple("frame_rate"),
New HTuple(5.60689))
        HOperatorSet.SetFramegrabberParam(hv_AcqHandle, New HTuple("gain_boost"),
New HTuple("enable"))
        HOperatorSet.SetFramegrabberParam(hv_AcqHandle, New
HTuple("gain_factor_g"), New HTuple(150))
        HOperatorSet.GrabImageStart(hv_AcqHandle, New HTuple(-1))
        MagWerken = True
    End If

```

```

hv_firstTrigger = New HTuple(0)
hv_gevonden = New HTuple(0)

```

```

Do While New HTuple(hv_firstTrigger.TupleEqual(New HTuple(0))).I()
    hv_objectTrigger = New HTuple(0)

```

```

    Do While New HTuple(hv_objectTrigger.TupleEqual(New HTuple(0))).I()

```

```

        HOperatorSet.GrabImageAsync(ho_image, hv_AcqHandle, New HTuple(-1))

```

```

        HWindowControl1.HalconWindow.ClearWindow()
        ho_image.DispObj(HWindowControl1.HalconWindow)

```

```

        HOperatorSet.ChangeRadialDistortionCamPar(New HTuple("fullsize"),
hv_CameraParameters, New HTuple(0), hv_CamParOut)
        HOperatorSet.ChangeRadialDistortionImage(ho_image, ho_image, ho_image,
hv_CameraParameters, hv_CamParOut)
        HOperatorSet.GetSize(ho_image, hv_Beeldbreedte, hv_Beeldhoogte)
        HOperatorSet.Decompose3(ho_image, ho_R, ho_G, ho_B)
        HOperatorSet.SubImage(ho_G, ho_R, ho_imageSub, New HTuple(1), New
HTuple(1))

```

```

        HOperatorSet.Threshold(ho_imageSub, ho_Regions,
Instellingen_Scherm.ThresholdOndergrenswaarde,
Instellingen_Scherm.ThresholdBovengrenswaarde)
        HOperatorSet.Connection(ho_Regions, ho_ConnectedRegions)
        HOperatorSet.SelectShape(ho_ConnectedRegions, ho_SelectedRegions, New
HTuple("area"), New HTuple("and"), New
HTuple(Instellingen_Scherm.AreaOndergrenswaarde), New
HTuple(Instellingen_Scherm.AreaBovengrenswaarde))
        HOperatorSet.CountObj(ho_SelectedRegions, hv_aantalObj)
        HOperatorSet.AreaCenter(ho_SelectedRegions, hv_Area1, hv_Row1,
hv_Column1)

```

```

        If New HTuple(hv_aantalObj.TupleGreater(New HTuple(1))).I() Then

```

```

            If (((New HTuple(((hv_Row1.TupleSelect(New
HTuple(0)))).TupleGreaterEqual( _

```

```

        hv_Beeldhoogte.TupleDiv(New HTuple(2))))).TupleAnd(New
HTuple(((hv_Row1.TupleSelect( _
        New HTuple(0))))).TupleLessEqual(((hv_Beeldhoogte.TupleDiv(New
HTuple(2))))).TupleAdd( _
        New HTuple(300)))))).TupleAnd(New
HTuple(((hv_Row1.TupleSelect(New HTuple(1))))).TupleGreaterEqual( _
        hv_Beeldhoogte.TupleSub(New HTuple(400))))).I() Then

        HOperatorSet.SelectObj(ho_SelectedRegions, ho_ObjectSelected,
New HTuple(1))
        hv_objectTrigger = New HTuple(1)

    Else
        HOperatorSet.SelectObj(ho_SelectedRegions, ho_ObjectSelected,
New HTuple(2))
        hv_objectTrigger = New HTuple(1)
    End If

    ElseIf New HTuple(hv_aantalObj.TupleEqual(New HTuple(1))).I() Then

        If ((New HTuple(((hv_Row1.TupleSelect(New
HTuple(0))))).TupleGreaterEqual( _
        hv_Beeldhoogte.TupleDiv(New HTuple(2)))))).TupleAnd(New
HTuple(((hv_Row1.TupleSelect( _
        New HTuple(0))))).TupleLessEqual(((hv_Beeldhoogte.TupleDiv(New
HTuple(2))))).TupleAdd( _
        New HTuple(300)))))).I() Then

            HOperatorSet.SelectObj(ho_SelectedRegions, ho_ObjectSelected,
New HTuple(1))
            hv_objectTrigger = New HTuple(1)
        End If
    End If

    Loop

    HOperatorSet.RegionFeatures(ho_ObjectSelected, New HTuple("area"),
hv_Value)
    HOperatorSet.AreaCenter(ho_ObjectSelected, hv_Area1, hv_Row1, hv_Column1)
    If (((New HTuple((New
HTuple(Instellingen_Scherm.AreaOndergrenswaarde)).TupleLessEqual(hv_Value))))).TupleAnd
(New HTuple( _
        hv_Value.TupleLessEqual(New
HTuple(Instellingen_Scherm.AreaBovengrenswaarde))))).TupleAnd(New
HTuple(hv_firstTrigger.TupleEqual( _
        New HTuple(0))))).I() Then

        hv_gevonden = New HTuple(1)
    Else
        hv_gevonden = New HTuple(0)
    End If

    'Volgende code gebeurt wanneer strip op een bepaalde afstand staat
    If (((New HTuple(hv_Row1.TupleGreaterEqual(hv_Beeldhoogte.TupleDiv(New
HTuple(2)))))).TupleAnd( _
        New HTuple(hv_Row1.TupleLessEqual(((hv_Beeldhoogte.TupleDiv(New
HTuple(2))))).TupleAdd( _
        New HTuple(300)))))).TupleAnd(New HTuple(hv_gevonden.TupleEqual(New
HTuple(1))))).I() Then

        hv_firstTrigger = New HTuple(1)
    Else

```

```

        hv_firstTrigger = New HTuple(0)
    End If

    Loop

        'Strip rechtzetten.

        HOperatorSet.CopyObj(ho_ObjectSelected, ho_ObjectSelectedOut, New HTuple(1),
New HTuple(-1))
        HOperatorSet.DilationCircle(ho_ObjectSelectedOut, ho_RegionDilation, New
HTuple(20))
        HOperatorSet.Connection(ho_RegionDilation, ho_ConnectedRegions2)
        HOperatorSet.ErosionCircle(ho_ConnectedRegions2, ho_RegionErosion, New
HTuple(20))
        HOperatorSet.ReduceDomain(ho_image, ho_RegionErosion, ho_Baksteenstrip)
        HOperatorSet.OrientationRegion(ho_Baksteenstrip, hv_Phi)
        hv_Hoek_graden = (((New HTuple(360)).TupleDiv((New HTuple(2)).TupleMult(New
HTuple(3.14))))).TupleMult(hv_Phi)

        If New HTuple(hv_Hoek_graden.TupleGreater(New HTuple(90))).I() Then

            hv_Hoek_graden = hv_Hoek_graden.TupleSub(New HTuple(180))
        End If

        HOperatorSet.RotateImage(ho_Baksteenstrip, ho_ImageRotate,
hv_Hoek_graden.TupleNeg(), New HTuple("constant"))
        HOperatorSet.Decompose3(ho_ImageRotate, ho_R, ho_G, ho_B)
        HOperatorSet.SubImage(ho_G, ho_R, ho_imageSub, New HTuple(1), New HTuple(1))
        HOperatorSet.Threshold(ho_imageSub, ho_Regions, New HTuple(0), New HTuple(50))
        HOperatorSet.Connection(ho_Regions, ho_ConnectedRegions)
        HOperatorSet.SelectShape(ho_ConnectedRegions, ho_SelectedRegions, New
HTuple("area"), New HTuple("and"), New
HTuple(Instellingen_Scherm.AreaOndergrensWaarde), New
HTuple(Instellingen_Scherm.AreaBovengrensWaarde))
        HOperatorSet.AreaCenter(ho_SelectedRegions, hv_Area1, hv_Row1, hv_Column1)

        If New HTuple(hv_aantalObj.TupleGreater(New HTuple(1))).I() Then

            If (((New HTuple(((hv_Row1.TupleSelect(New
HTuple(0))))).TupleGreaterEqual(hv_Beeldhoogte.TupleDiv( _
New HTuple(2))))).TupleAnd(New HTuple(((hv_Row1.TupleSelect(New
HTuple(0))))).TupleLessEqual( _
((hv_Beeldhoogte.TupleDiv(New HTuple(2))))).TupleAdd(New
HTuple(300)))))).TupleAnd( _
New HTuple(((hv_Row1.TupleSelect(New
HTuple(1))))).TupleGreaterEqual(hv_Beeldhoogte.TupleSub( _
New HTuple(400))))).I() Then

                HOperatorSet.SelectObj(ho_SelectedRegions, ho_ObjectSelectedOut, New
HTuple(1))

            Else

                HOperatorSet.SelectObj(ho_SelectedRegions, ho_ObjectSelectedOut, New
HTuple(2))

            End If

        ElseIf New HTuple(hv_aantalObj.TupleEqual(New HTuple(1))).I() Then

            If ((New HTuple(((hv_Row1.TupleSelect(New
HTuple(0))))).TupleGreaterEqual(hv_Beeldhoogte.TupleDiv( _
New HTuple(2))))).TupleAnd(New HTuple(((hv_Row1.TupleSelect(New
HTuple(0))))).TupleLessEqual( _

```

```

                ((hv_Beeldhoogte.TupleDiv(New HTuple(2))))).TupleAdd(New
HTuple(300))))).I() Then

                HOperatorSet.SelectObj(ho_SelectedRegions, ho_ObjectSelectedOut, New
HTuple(1))
                End If
            End If

            HOperatorSet.DilationCircle(ho_ObjectSelectedOut, ho_RegionDilation, New
HTuple(20))
            HOperatorSet.Connection(ho_RegionDilation, ho_ConnectedRegions2)
            HOperatorSet.ErosionCircle(ho_ConnectedRegions2, ho_Baksteenstrip2, New
HTuple(20))
            HOperatorSet.SmallestRectangle2(ho_Baksteenstrip2, hv_Row, hv_Column, hv_phi2,
hv_Length1, hv_Length2)

            HWindowControl1.HalconWindow.ClearWindow()
            ho_Baksteenstrip.DispObj(HWindowControl1.HalconWindow)

            'Contour- en hoekcontrole.

            HOperatorSet.AreaCenter(ho_Baksteenstrip2, hv_Area, hv_R1, hv_C1)
            HOperatorSet.GetRegionContour(ho_Baksteenstrip2, hv_Rows, hv_Columns)
            hv_striplengte = ((hv_Length1.TupleMult(New
HTuple(2))))).TupleMult(hv_pixelToMm)
            hv_stripbreedte = ((hv_Length2.TupleMult(New
HTuple(2))))).TupleMult(hv_pixelToMm)
            HOperatorSet.Rectangularity(ho_Baksteenstrip2, hv_Rechthoekigheid)

            stripbreedte = hv_stripbreedte
            striplengte = hv_striplengte

            LengteStrip_Invoer.Text = striplengte
            BreedteStrip_Invoer.Text = stripbreedte

            hv_OppervlakteOrig = ((hv_Length1.TupleMult(New
HTuple(2))))).TupleMult(hv_Length2.TupleMult(New HTuple(2)))

            If ((((((New HTuple((New HTuple(hv_OppervlakteOrig -
Instellingen_Scherm.vormEisOppwaarde)).TupleLess(hv_Area)))))).TupleAnd(New
HTuple(hv_Rechthoekigheid.TupleGreaterEqual(Instellingen_Scherm.ContoureisWaarde /
100))))).I() Then

                hv_contourInkeping = New HTuple(1)
                Vorm_Invoer.ForeColor = Color.Green
                Vorm_Invoer.Text = "Goedgekeurd"

            Else

                hv_contourInkeping = New HTuple(0)
                Vorm_Invoer.ForeColor = Color.Red
                Vorm_Invoer.Text = "Afgekeurd"

            End If

            HOperatorSet.TupleMin(hv_Rows, hv_Rmin)
            HOperatorSet.TupleMax(hv_Rows, hv_Rmax)
            HOperatorSet.TupleMin(hv_Columns, hv_Cmin)
            HOperatorSet.TupleMax(hv_Columns, hv_Cmax)

            'Eerste hoek.

```



```

        HOperatorSet.GenRectangle1(ho_corner1, hv_Rmin.TupleSub(New HTuple(20)),
hv_Cmin.TupleSub( _
        New HTuple(20)), hv_Rmin.TupleAdd(New HTuple(100)), hv_Cmin.TupleAdd(New
HTuple(100)))
        HOperatorSet.AreaCenter(ho_corner1, hv_AreaC1, hv_centerRowCorner1,
hv_CenterColumnCorner1)
        HOperatorSet.ReduceDomain(ho_Baksteenstrip2, ho_corner1, ho_cornerZoom1)
        HOperatorSet.SmallestRectangle2(ho_cornerZoom1, hv_x1, hv_y1, hv_phi1, hv_W1,
hv_H1)
        HOperatorSet.GenRectangle2(ho_co1, hv_x1, hv_y1, hv_phi1, hv_W1, hv_H1)
        HOperatorSet.Rectangularity(ho_cornerZoom1, hv_rect1)

        'Tweede hoek.
        HOperatorSet.GenRectangle1(ho_corner2, hv_Rmax.TupleSub(New HTuple(100)),
hv_Cmin.TupleSub( _
        New HTuple(20)), hv_Rmax.TupleAdd(New HTuple(20)), hv_Cmin.TupleAdd(New
HTuple(100)))
        HOperatorSet.AreaCenter(ho_corner2, hv_AreaC2, hv_CenterRowCorner2,
hv_CenterColumnCorner2)
        HOperatorSet.ReduceDomain(ho_Baksteenstrip2, ho_corner2, ho_cornerZoom2)
        HOperatorSet.SmallestRectangle2(ho_cornerZoom2, hv_x2, hv_y2, hv_phi2, hv_W2,
hv_H2)
        HOperatorSet.GenRectangle2(ho_co2, hv_x2, hv_y2, hv_phi2, hv_W2, hv_H2)
        HOperatorSet.Rectangularity(ho_cornerZoom2, hv_rect2)

        'Derde hoek.
        HOperatorSet.GenRectangle1(ho_corner3, hv_Rmin.TupleSub(New HTuple(20)),
hv_Cmax.TupleSub( _
        New HTuple(100)), hv_Rmin.TupleAdd(New HTuple(100)), hv_Cmax.TupleAdd(New
HTuple(20)))
        HOperatorSet.AreaCenter(ho_corner3, hv_AreaC3, hv_CenterRowCorner3,
hv_CenterColumnCorner3)
        HOperatorSet.ReduceDomain(ho_Baksteenstrip2, ho_corner3, ho_cornerZoom3)
        HOperatorSet.SmallestRectangle2(ho_cornerZoom3, hv_x3, hv_y3, hv_phi3, hv_W3,
hv_H3)
        HOperatorSet.GenRectangle2(ho_co3, hv_x3, hv_y3, hv_phi3, hv_W3, hv_H3)
        HOperatorSet.Rectangularity(ho_cornerZoom3, hv_rect3)

        'Vierde hoek.
        HOperatorSet.GenRectangle1(ho_corner4, hv_Rmax.TupleSub(New HTuple(100)),
hv_Cmax.TupleSub( _
        New HTuple(100)), hv_Rmax.TupleAdd(New HTuple(20)), hv_Cmax.TupleAdd(New
HTuple(20)))
        HOperatorSet.AreaCenter(ho_corner4, hv_AreaC4, hv_CenterRowCorner4,
hv_CenterColumnCorner4)
        HOperatorSet.ReduceDomain(ho_Baksteenstrip2, ho_corner4, ho_cornerZoom4)
        HOperatorSet.SmallestRectangle2(ho_cornerZoom4, hv_x4, hv_y4, hv_phi4, hv_W4,
hv_H4)
        HOperatorSet.GenRectangle2(ho_co4, hv_x4, hv_y4, hv_phi4, hv_W4, hv_H4)
        HOperatorSet.Rectangularity(ho_cornerZoom4, hv_rect4)
        hv_tupleRec =
        (((hv_rect1.TupleConcat(hv_rect2))).TupleConcat(hv_rect3)).TupleConcat(hv_rect4)

        'Volgende code geeft weer of alle hoeken voldoen aan de eisen.

        hv_cornerCorrect = New HTuple(1)

        For hv_i = (New HTuple(0)) To ((New
HTuple(hv_tupleRec.TupleLength()))).TupleSub(New HTuple(1)) Step (New HTuple(1))
            If New HTuple(hv_cornerCorrect.TupleEqual(New HTuple(1))).I() Then

```

```

        If New
HTuple(((hv_tupleRec.TupleSelect(hv_i))).TupleLess(Instellingen_Scherm.RandeisWaarde /
100)).I() Then

            hv_cornerCorrect = New HTuple(0)

            Rand_Invoer.ForeColor = Color.Red
            Rand_Invoer.Text = "Afgekeurd"

        ElseIf New
HTuple(((hv_tupleRec.TupleSelect(hv_i))).TupleGreaterEqual(Instellingen_Scherm.Randeis
Waarde / 100)).I() Then

            hv_cornerCorrect = New HTuple(1)

            Rand_Invoer.ForeColor = Color.Green
            Rand_Invoer.Text = "Goedgekeurd"

        End If
    End If
Next

hv_secondTrigger = New HTuple(0)
hv_thirdTrigger = New HTuple(0)
hv_cyclus = New HTuple(0)

HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0), hv_Left)
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0), hv_Right)
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0), hv_GemHoogtes)
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0), hv_GroupLinks)
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0), hv_GroupRechts)
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0), hv_GroupsLinksMean)
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0), hv_GroupsRechtsMean)

Do While New HTuple(hv_secondTrigger.TupleEqual(New HTuple(0))).I()

    HOperatorSet.GrabImageAsync(ho_ImageLine, hv_AcqHandle, New HTuple(-1))
    HOperatorSet.Decompose3(ho_ImageLine, ho_R2, ho_G2, ho_B2)
    HOperatorSet.Threshold(ho_R2, ho_lijn, New HTuple(215), New HTuple(255))
    HOperatorSet.DilationCircle(ho_lijn, ho_lijnverdikking, New HTuple(20))
    HOperatorSet.Connection(ho_lijnverdikking, ho_dikkelijnen)
    HOperatorSet.ErosionCircle(ho_dikkelijnen, ho_lijnen, New HTuple(20))
    HOperatorSet.FillUp(ho_lijnen, ho_vollelijnen)
    HOperatorSet.SelectShape(ho_vollelijnen, ho_lines, New HTuple("area"), New
HTuple("and"), New HTuple(500), New HTuple(104105))

    HOperatorSet.CountObj(ho_lines, hv_Number)

    If (((New HTuple(hv_Number.TupleEqual(New HTuple(1))))).TupleOr(New
HTuple(hv_Number.TupleEqual( _
        New HTuple(2))))).TupleOr(New HTuple(hv_thirdTrigger.TupleEqual(New
HTuple(1))))).I() Then
        HOperatorSet.TupleLength(hv_Left, hv_LeftLength)
        HOperatorSet.TupleLength(hv_Right, hv_RightLength)
        HOperatorSet.SelectObj(ho_lines, ho_midden, New HTuple(1))
        HOperatorSet.GetRegionPoints(ho_midden, hv_RijMidden, hv_KolMidden)

        If ((New HTuple(hv_LeftLength.TupleGreater(New
HTuple(0))))).TupleAnd(New HTuple( _
            hv_RightLength.TupleGreater(New HTuple(0))))).I() Then

            For hv_i = (New HTuple(1)) To (New HTuple(5)) Step (New HTuple(1))

```

```

        For hv_y = ( _
                    (((hv_LeftLength.TupleSub(New
HTuple(1))))).TupleMult(hv_i.TupleSub( _
                    New HTuple(1))))).TupleDiv(New HTuple(5)) To
                    (((hv_LeftLength.TupleSub( _
                    New HTuple(1))))).TupleMult(hv_i)).TupleDiv(New HTuple(5))
Step (New HTuple(1))
                    HOperatorSet.TupleReplace(hv_GroupLinks,
hv_y.TupleSub((((hv_LeftLength.TupleSub( _
                    New HTuple(1))))).TupleMult(hv_i.TupleSub(New
HTuple(1))))).TupleDiv( _
                    New HTuple(5))), hv_Left.TupleSelect(hv_y),
hv_GroupLinks)

        Next
        HOperatorSet.TupleMean(hv_GroupLinks, hv_LinksMean)
        HOperatorSet.TupleInsert(hv_GroupsLinksMean, hv_i.TupleSub(New
HTuple(1)), _
                    hv_LinksMean, hv_GroupsLinksMean)

        For hv_z = (((hv_RightLength.TupleSub( _
                    New HTuple(1))))).TupleMult(hv_i.TupleSub(New
HTuple(1))))).TupleDiv( _
                    New HTuple(5)) To (((hv_RightLength.TupleSub(New
HTuple(1))))).TupleMult( _
                    hv_i)).TupleDiv(New HTuple(5)) Step (New HTuple(1))
                    HOperatorSet.TupleReplace(hv_GroupRechts,
hv_z.TupleSub((((hv_RightLength.TupleSub( _
                    New HTuple(1))))).TupleMult(hv_i.TupleSub(New
HTuple(1))))).TupleDiv( _
                    New HTuple(5))), hv_Right.TupleSelect(hv_z),
hv_GroupRechts)

        Next
        HOperatorSet.TupleMean(hv_GroupRechts, hv_RechtsMean)
        HOperatorSet.TupleInsert(hv_GroupsRechtsMean,
hv_i.TupleSub(New HTuple(1)), _
                    hv_RechtsMean, hv_GroupsRechtsMean)

        Next

        HOperatorSet.TupleLength(hv_GroupsLinksMean, hv_lengthLgem)
        HOperatorSet.TupleLength(hv_GroupsRechtsMean, hv_lengthRgem)
        HOperatorSet.TupleMean(hv_Left, hv_LeftMean)
        HOperatorSet.TupleMean(hv_Right, hv_RightMean)

        Try
            If ((New HTuple(((hv_Left.TupleSelect(New
HTuple(0))))).TupleGreater(((hv_Left.TupleSelect( _
                    hv_LeftLength.TupleSub(New HTuple(1))))).TupleAdd(New
HTuple(4))))).TupleOr( _
                    New HTuple((((hv_Left.TupleSelect(New
HTuple(0))))).TupleAdd(New HTuple(4))))).TupleLess( _
                    hv_Left.TupleSelect(hv_LeftLength.TupleSub(New
HTuple(1)))))).I() Then

                    hv_GoedeStrip = New HTuple(0)

            ElseIf ((New HTuple(((hv_Right.TupleSelect(New
HTuple(0))))).TupleGreater((( _

```

```

        hv_Right.TupleSelect(hv_RightLength.TupleSub(New
HTuple(1))))).TupleAdd( _
        New HTuple(4))))).TupleOr(New
HTuple((((hv_Right.TupleSelect(New HTuple(0))))).TupleAdd( _
        New
HTuple(4))))).TupleLess(hv_Right.TupleSelect(hv_RightLength.TupleSub( _
        New HTuple(1))))).I() Then

        hv_GoedeStrip = New HTuple(0)

        ElseIf ((New
HTuple(hv_LeftMean.TupleLess(((hv_Left.TupleSelect(New HTuple(0))))).TupleSub( _
        New HTuple(4))))).TupleOr(New
HTuple(hv_RightMean.TupleLess(((hv_Right.TupleSelect( _
        New HTuple(0))))).TupleSub(New HTuple(4))))).I() Then

        hv_GoedeStrip = New HTuple(0)

        ElseIf ((((((New HTuple(((hv_GroupsLinksMean.TupleSelect(New
HTuple(0))))).TupleLess( _
        ((hv_GroupsLinksMean.TupleSelect(New
HTuple(1))))).TupleSub(New HTuple(1)))))).TupleAnd( _
        New HTuple(((hv_GroupsLinksMean.TupleSelect(New
HTuple(1))))).TupleLess( _
        ((hv_GroupsLinksMean.TupleSelect(New
HTuple(2))))).TupleSub(New HTuple(1)))))).TupleAnd( _
        New HTuple(((hv_GroupsLinksMean.TupleSelect(New
HTuple(2))))).TupleLess( _
        ((hv_GroupsLinksMean.TupleSelect(New
HTuple(3))))).TupleSub(New HTuple(1)))))).TupleAnd( _
        New HTuple(((hv_GroupsLinksMean.TupleSelect(New
HTuple(3))))).TupleLess( _
        ((hv_GroupsLinksMean.TupleSelect(New
HTuple(4))))).TupleSub(New HTuple(1))))).I() Then

        hv_GoedeStrip = New HTuple(0)

        ElseIf ((((((New HTuple(((hv_GroupsLinksMean.TupleSelect(New
HTuple(0))))).TupleGreater( _
        ((hv_GroupsLinksMean.TupleSelect(New
HTuple(1))))).TupleAdd(New HTuple(1)))))).TupleAnd( _
        New HTuple(((hv_GroupsLinksMean.TupleSelect(New
HTuple(1))))).TupleGreater( _
        ((hv_GroupsLinksMean.TupleSelect(New
HTuple(2))))).TupleAdd(New HTuple(1)))))).TupleAnd( _
        New HTuple(((hv_GroupsLinksMean.TupleSelect(New
HTuple(2))))).TupleGreater( _
        ((hv_GroupsLinksMean.TupleSelect(New
HTuple(3))))).TupleAdd(New HTuple(1)))))).TupleAnd( _
        New HTuple(((hv_GroupsLinksMean.TupleSelect(New
HTuple(3))))).TupleGreater( _
        ((hv_GroupsLinksMean.TupleSelect(New
HTuple(4))))).TupleAdd(New HTuple(1))))).I() Then

        hv_GoedeStrip = New HTuple(0)

        ElseIf ((((((New HTuple(((hv_GroupsRechtsMean.TupleSelect(New
HTuple(0))))).TupleGreater( _
        ((hv_GroupsRechtsMean.TupleSelect(New
HTuple(1))))).TupleAdd(New HTuple(1)))))).TupleAnd( _
        New HTuple(((hv_GroupsRechtsMean.TupleSelect(New
HTuple(1))))).TupleGreater( _

```

```

                ((hv_GroupsRechtsMean.TupleSelect(New
HTuple(2))))).TupleAdd(New HTuple(1)))))).TupleAnd( _
                New HTuple(((hv_GroupsRechtsMean.TupleSelect(New
HTuple(2))))).TupleGreater( _
                ((hv_GroupsRechtsMean.TupleSelect(New
HTuple(3))))).TupleAdd(New HTuple(1)))))).TupleAnd( _
                New HTuple(((hv_GroupsRechtsMean.TupleSelect(New
HTuple(3))))).TupleGreater( _
                ((hv_GroupsRechtsMean.TupleSelect(New
HTuple(4))))).TupleAdd(New HTuple(1)))))).I() Then

                hv_GoedeStrip = New HTuple(0)

                ElseIf ((((((New HTuple(((hv_GroupsRechtsMean.TupleSelect(New
HTuple(0))))).TupleGreater( _
                ((hv_GroupsRechtsMean.TupleSelect(New
HTuple(1))))).TupleAdd(New HTuple(1)))))).TupleAnd( _
                New HTuple(((hv_GroupsRechtsMean.TupleSelect(New
HTuple(1))))).TupleGreater( _
                ((hv_GroupsRechtsMean.TupleSelect(New
HTuple(2))))).TupleAdd(New HTuple(1)))))).TupleAnd( _
                New HTuple(((hv_GroupsRechtsMean.TupleSelect(New
HTuple(2))))).TupleGreater( _
                ((hv_GroupsRechtsMean.TupleSelect(New
HTuple(3))))).TupleAdd(New HTuple(1)))))).TupleAnd( _
                New HTuple(((hv_GroupsRechtsMean.TupleSelect(New
HTuple(3))))).TupleGreater( _
                ((hv_GroupsRechtsMean.TupleSelect(New
HTuple(4))))).TupleAdd(New HTuple(1)))))).I() Then

                hv_GoedeStrip = New HTuple(0)
            Else
                hv_GoedeStrip = New HTuple(1)
            End If

            Catch HDevExpDefaultException1 As HalconException
                HDevExpDefaultException1.ToHTuple(hv_Exception1)
            End Try

        End If
        HOperatorSet.TupleRemove(hv_Left, HTuple.TupleGenSequence(New
HTuple(0), hv_LeftLength.TupleSub(New HTuple(1)), New HTuple(1)), hv_Left)
        HOperatorSet.TupleRemove(hv_Right, HTuple.TupleGenSequence(New
HTuple(0), hv_RightLength.TupleSub(New HTuple(1)), New HTuple(1)), hv_Right)
        HOperatorSet.TupleLength(hv_Left, hv_LeftLength)
        HOperatorSet.TupleLength(hv_Right, hv_RightLength)

        hv_secondTrigger = New HTuple(1)
        hv_thirdTrigger = New HTuple(0)

    ElseIf New HTuple(hv_Number.TupleEqual(New HTuple(3))).I() Then

        HOperatorSet.SelectObj(ho_lines, ho_links, New HTuple(2))
        HOperatorSet.GetRegionPoints(ho_links, hv_rijLinks, hv_KolLinks)
        HOperatorSet.SelectObj(ho_lines, ho_rechts, New HTuple(3))
        HOperatorSet.GetRegionPoints(ho_rechts, hv_rijRechts, hv_KolRechts)
        HOperatorSet.SelectObj(ho_lines, ho_midden, New HTuple(1))
        HOperatorSet.GetRegionPoints(ho_midden, hv_RijMidden, hv_KolMidden)
        HOperatorSet.TupleSort(hv_KolLinks, hv_sortKL)
        HOperatorSet.TupleSort(hv_KolRechts, hv_sortKR)
        HOperatorSet.TupleMedian(hv_rijLinks, hv_MediaanLinksR)
        HOperatorSet.TupleMedian(hv_sortKL, hv_MediaanLinksK)

```

```

HOperatorSet.TupleMedian(hv_sortKR, hv_MediaanRechtsK)
HOperatorSet.TupleMedian(hv_rijRechts, hv_MediaanRechtsR)
HOperatorSet.TupleLength(hv_KolMidden, hv_ElementenMiddellijn)
HOperatorSet.TupleLength(hv_KolLinks, hv_ElementenLinks)
HOperatorSet.TupleLength(hv_KolRechts, hv_ElementenRechts)
HOperatorSet.TupleSelect(hv_sortKL, hv_ElementenLinks.TupleSub(New
HTuple(1)), hv_EindkolomLinks)
HOperatorSet.TupleSelect(hv_sortKR, New HTuple(0),
hv_BeginKolomRechts)
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0), hv_hoogtes)

hv_BeginKolomLinks = hv_MediaanLinksK.Clone()
hv_BeginrijLinks = hv_MediaanLinksR.Clone()
hv_EindrijLinks = hv_MediaanLinksR.Clone()
hv_EindKolomRechts = hv_MediaanRechtsK.Clone()
hv_EindrijRechts = hv_MediaanRechtsR.Clone()
hv_BeginrijRechts = hv_MediaanRechtsR.Clone()

HOperatorSet.GenRegionLine(ho_Verbindingslijn, hv_BeginrijLinks,
hv_BeginKolomLinks, hv_EindrijRechts, hv_EindKolomRechts)
HOperatorSet.GenRegionLine(ho_Linkerlijn, hv_BeginrijLinks,
hv_BeginKolomLinks, hv_EindrijLinks, hv_EindkolomLinks)
HOperatorSet.GenRegionLine(ho_Rechterlijn, hv_BeginrijRechts,
hv_BeginKolomRechts, hv_EindrijRechts, hv_EindKolomRechts)
HOperatorSet.TupleSort(hv_KolMidden, hv_KolMiddenSort)
HOperatorSet.TupleUniq(hv_RijMidden, hv_RijUniekSort)
HOperatorSet.TupleUniq(hv_KolMiddenSort, hv_KolUniekSort)
HOperatorSet.TupleLength(hv_KolUniekSort, hv_KU lengte)
HOperatorSet.TupleLength(hv_RijUniekSort, hv_RU lengte)
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0),
hv_RijMiddenNieuw)
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0),
hv_KolMiddenNieuw)

For hv_f = (New HTuple(0)) To ((hv_KU lengte.TupleSub(New
HTuple(1))))).TupleDiv(New HTuple(100)) Step (New HTuple(1))
HOperatorSet.TupleGenConst(New HTuple(0), New HTuple(0),
hv_rijengroep)

For hv_g = (New HTuple(0)) To ((hv_RU lengte.TupleSub(New
HTuple(1))))).TupleDiv(New HTuple(4)) Step (New HTuple(1))
HOperatorSet.TupleLength(hv_rijengroep, hv_rijengroep lengte)
hv_yCo = hv_RijUniekSort.TupleSelect(hv_g.TupleMult(New
HTuple(4)))
hv_xCo = hv_KolUniekSort.TupleSelect(hv_f.TupleMult(New
HTuple(100)))
HOperatorSet.TestRegionPoint(ho_midden, hv_yCo, hv_xCo,
hv_IsInside)

If New HTuple(hv_IsInside.TupleEqual(New HTuple(1))).I() Then
HOperatorSet.TupleReplace(hv_rijengroep,
hv_rijengroep lengte, hv_yCo, hv_rijengroep)
End If

Next

HOperatorSet.TupleLength(hv_RijMiddenNieuw,
hv_RijMiddenNieuw Lengte)
HOperatorSet.TupleLength(hv_KolMiddenNieuw,
hv_KolMiddenNieuw Lengte)

```

```

        If New HTuple(hv_rijengroepLengte.TupleNotEqual(New
HTuple(0))).I() Then
            HOperatorSet.TupleMean(hv_rijengroep, hv_rijengroepMean)
            HOperatorSet.TupleInsert(hv_RijMiddenNieuw,
hv_RijMiddenNieuwLengte, hv_rijengroepMean, hv_RijMiddenNieuw)
            HOperatorSet.TupleInsert(hv_KolMiddenNieuw,
hv_KolMiddenNieuwLengte, hv_xCo, hv_KolMiddenNieuw)
            End If

        Next

        HOperatorSet.TupleLength(hv_KolMiddenNieuw, hv_KolMiddenNieuwLengte)
        HOperatorSet.TupleLength(hv_RijMiddenNieuw, hv_RijMiddenNieuwLengte)

        If New HTuple(hv_KolMiddenNieuwLengte.TupleGreater(New HTuple(4))).I()
Then

            For hv_k = (New HTuple(0)) To hv_KolMiddenNieuwLengte.TupleSub(New
HTuple(1)) Step (New HTuple(1))
                HOperatorSet.TupleSelect(hv_RijMiddenNieuw, hv_k, hv_PuntY)
                HOperatorSet.TupleSelect(hv_KolMiddenNieuw, hv_k, hv_PuntX)
                HOperatorSet.DistancePl(hv_PuntY, hv_PuntX, hv_BeginrijLinks,
hv_BeginKolomLinks, _
                    hv_EindrijRechts, hv_EindKolomRechts, hv_afstand)
                hv_afstandCM = hv_afstand.TupleMult(hv_pixelToMm)
                hv_hoogtePunt = ((hv_hoek.TupleTan())).TupleMult(hv_afstandCM)
                HOperatorSet.TupleInsert(hv_hoogtes, hv_k, hv_hoogtePunt,
hv_hoogtes)

                Next
                HOperatorSet.TupleMean(hv_hoogtes, hv_MeanHoogtes)
                HOperatorSet.TupleInsert(hv_GemHoogtes, hv_cyclus, hv_MeanHoogtes,
hv_GemHoogtes)

                HOperatorSet.TupleLength(hv_hoogtes, hv_ElementenHoogtes)
                HOperatorSet.TupleSelect(hv_hoogtes, New HTuple(0),
hv_ReferentieLinks)
                HOperatorSet.TupleSelect(hv_hoogtes,
hv_ElementenHoogtes.TupleSub(New HTuple(1)), hv_ReferentieRechts)
                HOperatorSet.TupleInsert(hv_Left, hv_cyclus,
hv_hoogtes.TupleSelect(New HTuple(0)), hv_Left)
                HOperatorSet.TupleInsert(hv_Right, hv_cyclus,
hv_hoogtes.TupleSelect(hv_ElementenHoogtes.TupleSub(New HTuple(1))), hv_Right)
                hv_cyclus = hv_cyclus.TupleAdd(New HTuple(1))

                HOperatorSet.TupleGreaterEqualElem(hv_hoogtes, New HTuple(17),
hv_Greatereq)

                HOperatorSet.TupleMean(hv_Greatereq, hv_equationMean)
                If ((New
HTuple(hv_equationMean.TupleGreater(Instellingen_Scherm.SpievormigheidseisWaarde /
100))).TupleAnd(New HTuple(hv_equationMean.TupleLessEqual(New HTuple(1))).I() Then

                    hv_GoedeStrip = New HTuple(1)
                    Spievorming_Invoer.ForeColor = Color.Green
                    Spievorming_Invoer.Text = "Goedgekeurd"

                ElseIf New
HTuple((((hv_ReferentieLinks.TupleSub(hv_ReferentieRechts))).TupleAbs())).TupleLessEq
ual(New HTuple(Instellingen_Scherm.SpievormigheidseisHoogteWaarde)).I() Then

                    hv_GoedeStrip = New HTuple(1)

                    Spievorming_Invoer.ForeColor = Color.Green

```

```

        Spievorming_Invoer.Text = "Goedgekeurd"
    Else

        hv_GoedeStrip = New HTuple(0)

        Spievorming_Invoer.ForeColor = Color.Red
        Spievorming_Invoer.Text = "Afgekeurd"
    End If

    If ((New HTuple((New HTuple(-
5)).TupleLessEqual(hv_MeanHoogtes))))).TupleAnd(New
HTuple(hv_MeanHoogtes.TupleLessEqual(New HTuple(10))))).I() Then

        hv_thirdTrigger = New HTuple(1)

    ElseIf New HTuple(hv_GoedeStrip.TupleEqual(New HTuple(0))).I()
Then

        hv_secondTrigger = New HTuple(1)

        End If
    End If
End If

Try
    HOperatorSet.TupleMean(hv_GemHoogtes, hv_GemHoogteStrip)
' catch (Exception)
Catch HDevExpDefaultException1 As HalconException
    HDevExpDefaultException1.ToHTuple(hv_Exception)
End Try
Loop

If (hv_GemHoogteStrip > New HTuple(5)) Then
    striphoogte = hv_GemHoogteStrip
    HoogteStrip_Invoer.Text = striphoogte
End If

If (New HTuple(hv_GemHoogteStrip.TupleGreaterEqual(New
HTuple(10))).TupleAnd(hv_contourInkeping.TupleEqual(New
HTuple(1)).TupleAnd(hv_cornerCorrect.TupleEqual(New HTuple(1))))).I() Then
    sql.addAfmetingenSQL(striplengte, stripbreedte, striphoogte)
    AantalGoedeStrippen = AantalGoedeStrippen + 1
    AantalGoedeStrippenInvoer.Font = New Font("Verdana", 12,
FontStyle.Regular)
    AantalGoedeStrippenInvoer.Text = "Aantal goed strippen: " &
AantalGoedeStrippen
    End If
End Sub

Private Sub Button6_Click(sender As System.Object, e As System.EventArgs) Handles
InstellingenKnop.Click
    Instellingen_Scherm.Show()
    Me.Hide()
End Sub

Public Sub New()
    InitializeComponent()
    sql = New DatabaseConnectie()
End Sub
End Class

```



## Bijlage I: programmacode van form 3 in Visual Studio

**Public Class** Instellingen\_Scherm

```
Public ThresholdOndergrensWaarde As Integer
Public ThresholdBovengrensWaarde As Integer
Public AreaOndergrensWaarde As Integer
Public AreaBovengrensWaarde As Integer
Public AreaOndergrens As Integer
Public AreaBovengrens As Integer
Public LaserHoekWaarde As Integer
Public ContoureisWaarde As Integer
Public RandeisWaarde As Integer
Public SpievormigheidseisWaarde As Integer
Public vormEisOppWaarde As Integer
Public schaallengtepixel As Double
Public SpievormigheidseisHoogteWaarde As Integer

Public Sub NormaleThreshold_Knop_Click(sender As System.Object, e As
System.EventArgs) Handles NormaleThreshold_Knop.Click
    ThresholdOndergrens_Schuifbalk.Value = 0
    ThresholdBovengrens_Schuifbalk.Value = 50
    ThresholdOndergrensWaarde = ThresholdOndergrens_Schuifbalk.Value
    ThresholdBovengrensWaarde = ThresholdBovengrens_Schuifbalk.Value
    Threshold_OnderGrens.Text = ThresholdOndergrensWaarde
    Threshold_BovenGrens.Text = ThresholdBovengrensWaarde
End Sub

Public Sub Terug_Knop_Click(sender As System.Object, e As System.EventArgs)
Handles Terug_Knop.Click
    VisieControle_Scherm.Show()
    Me.Hide()
End Sub

Private Sub HScrollBar1_Scroll(sender As System.Object, e As
System.Windows.Forms.ScrollEventArgs) Handles ThresholdOndergrens_Schuifbalk.Scroll
    ThresholdOndergrensWaarde = ThresholdOndergrens_Schuifbalk.Value
    Threshold_OnderGrens.Text = ThresholdOndergrensWaarde
End Sub

Public Sub HScrollBar2_Scroll(sender As System.Object, e As
System.Windows.Forms.ScrollEventArgs) Handles ThresholdBovengrens_Schuifbalk.Scroll
    ThresholdBovengrensWaarde = ThresholdBovengrens_Schuifbalk.Value
    Threshold_BovenGrens.Text = ThresholdBovengrensWaarde
End Sub

Public Sub StandaardInstel_Knop_Click(sender As System.Object, e As
System.EventArgs) Handles StandaardInstel_Knop.Click
    NormaleThreshold_Knop.PerformClick()
    NormaleArea_Knop.PerformClick()
    DunneStrip_Knop.PerformClick()
    LaserHoek_Knop.PerformClick()
    Contoureis_Knop.PerformClick()
    Randeis_Knop.PerformClick()
    Spievormigheidseis_Knop.PerformClick()
End Sub

Public Sub DunneStrip_Knop_Click(sender As System.Object, e As System.EventArgs)
Handles DunneStrip_Knop.Click
    DunneStripIngesteld.ForeColor = Color.Black
```

```

        DunneStripIngesteld.Font = New Font("Microsoft Sans Serif", 12.25,
FontStyle.Regular)
        DunneStripIngesteld.Text = "Huidig ingesteld"
        DikkeStripIngesteld.ForeColor = Color.Black
        DikkeStripIngesteld.Font = New Font("Microsoft Sans Serif", 8.25,
FontStyle.Regular)
        DikkeStripIngesteld.Text = "Niet ingesteld"
        schaallengtepixel = 1056.0
    End Sub

    Public Sub DikkeStrip_Knop_Click(sender As System.Object, e As System.EventArgs)
Handles DikkeStrip_Knop.Click
        DikkeStripIngesteld.ForeColor = Color.Black
        DikkeStripIngesteld.Font = New Font("Microsoft Sans Serif", 12.25,
FontStyle.Regular)
        DikkeStripIngesteld.Text = "Huidig ingesteld"
        DunneStripIngesteld.ForeColor = Color.Black
        DunneStripIngesteld.Font = New Font("Microsoft Sans Serif", 8.25,
FontStyle.Regular)
        DunneStripIngesteld.Text = "Niet ingesteld"
        schaallengtepixel = 1075.262
    End Sub

    Private Sub Label6_Click(sender As System.Object, e As System.EventArgs) Handles
Threshold_BovenGrens.Click
    End Sub

    Private Sub HScrollBar3_Scroll(sender As System.Object, e As
System.Windows.Forms.ScrollEventArgs) Handles AreaOndergrens_Schuifbalk.Scroll
        AreaOndergrensWaarde = AreaOndergrens_Schuifbalk.Value
        Area_OnderGrens.Text = AreaOndergrensWaarde
    End Sub

    Private Sub HScrollBar4_Scroll(sender As System.Object, e As
System.Windows.Forms.ScrollEventArgs) Handles AreaBovengrens_Schuifbalk.Scroll
        AreaBovengrensWaarde = AreaBovengrens_Schuifbalk.Value
        Area_BovenGrens.Text = AreaBovengrensWaarde
    End Sub

    Private Sub NormaleArea_Knop_Click(sender As System.Object, e As System.EventArgs)
Handles NormaleArea_Knop.Click
        AreaOndergrens_Schuifbalk.Value = 300000
        AreaBovengrens_Schuifbalk.Value = 700000
        AreaOndergrensWaarde = AreaOndergrens_Schuifbalk.Value
        AreaBovengrensWaarde = AreaBovengrens_Schuifbalk.Value
        Area_OnderGrens.Text = AreaOndergrensWaarde
        Area_BovenGrens.Text = AreaBovengrensWaarde
    End Sub

    Private Sub LaserHoek_Schuifbalk_Scroll(sender As System.Object, e As
System.Windows.Forms.ScrollEventArgs) Handles LaserHoek_Schuifbalk.Scroll
        LaserHoekWaarde = LaserHoek_Schuifbalk.Value
        LaserHoek.Text = LaserHoekWaarde
    End Sub

    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles
LaserHoek_Knop.Click
        LaserHoek_Schuifbalk.Value = 59
        LaserHoekWaarde = LaserHoek_Schuifbalk.Value
        LaserHoek.Text = LaserHoekWaarde
    End Sub

```

```

Private Sub Contoureis_Schuifbalk_Scroll(sender As System.Object, e As
System.Windows.Forms.ScrollEventArgs) Handles Contoureis_Schuifbalk.Scroll
    ContoureisWaarde = Contoureis_Schuifbalk.Value
    Contoureis.Text = ContoureisWaarde & " %"
End Sub

Private Sub Contoureis_Knop_Click(sender As System.Object, e As System.EventArgs)
Handles Contoureis_Knop.Click
    Contoureis_Schuifbalk.Value = 94
    ContoureisWaarde = Contoureis_Schuifbalk.Value
    Contoureis.Text = ContoureisWaarde & " %"
    vormEisOpp_Schuifbalk.Value = 40000
    vormEisOppWaarde = vormEisOpp_Schuifbalk.Value
    vormEisOpp.Text = vormEisOppWaarde
End Sub

Private Sub Randeis_Knop_Click(sender As System.Object, e As System.EventArgs)
Handles Randeis_Knop.Click
    Randeis_Schuifbalk.Value = 65
    RandeisWaarde = Randeis_Schuifbalk.Value
    Randeis.Text = RandeisWaarde & " %"
End Sub

Private Sub Randeis_Schuifbalk_Scroll(sender As System.Object, e As
System.Windows.Forms.ScrollEventArgs) Handles Randeis_Schuifbalk.Scroll
    RandeisWaarde = Randeis_Schuifbalk.Value
    Randeis.Text = RandeisWaarde & " %"
End Sub

Private Sub Spievormigheidseis_Knop_Click(sender As System.Object, e As
System.EventArgs) Handles Spievormigheidseis_Knop.Click
    Spievormigheidseis_Schuifbalk.Value = 95
    SpievormigheidseisWaarde = Spievormigheidseis_Schuifbalk.Value
    Spievormigheidseis.Text = SpievormigheidseisWaarde & " %"
    SpievormigheidseisHoogte_Schuifbalk.Value = 4
    SpievormigheidseisHoogteWaarde = SpievormigheidseisHoogte_Schuifbalk.Value
    SpievormigheidseisHoogte.Text = SpievormigheidseisHoogteWaarde & " mm"
End Sub

Private Sub Spievormigheidseis_Schuifbalk_Scroll(sender As System.Object, e As
System.Windows.Forms.ScrollEventArgs) Handles Spievormigheidseis_Schuifbalk.Scroll
    SpievormigheidseisWaarde = Spievormigheidseis_Schuifbalk.Value
    Spievormigheidseis.Text = SpievormigheidseisWaarde & " %"
End Sub

Private Sub vormEisOpp_Schuifbalk_Scroll(sender As System.Object, e As
System.Windows.Forms.ScrollEventArgs) Handles vormEisOpp_Schuifbalk.Scroll
    vormEisOppWaarde = vormEisOpp_Schuifbalk.Value
    vormEisOpp.Text = vormEisOppWaarde
End Sub

Private Sub SpievormigheidseisHoogte_Schuifbalk_Scroll(sender As System.Object, e
As System.Windows.Forms.ScrollEventArgs) Handles
SpievormigheidseisHoogte_Schuifbalk.Scroll
    SpievormigheidseisHoogteWaarde = SpievormigheidseisHoogte_Schuifbalk.Value
    SpievormigheidseisHoogte.Text = SpievormigheidseisHoogteWaarde & " mm"
End Sub

Private Sub DunneStripIngesteld_Click(sender As System.Object, e As
System.EventArgs) Handles DunneStripIngesteld.Click
End Sub
End Class

```

## Bijlage J: datasheet uEye UI-1460LE-C



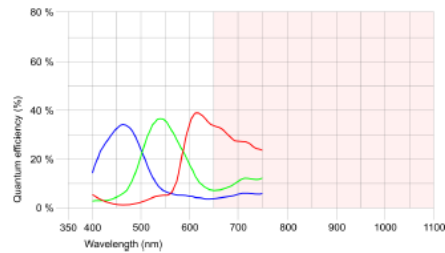
### UI-1460LE-C-HQ



### Specification

#### Sensor

Sensor Technology	CMOS Color
Manufacturer	Aptina
Resolution (h x v)	2048 x 1536
Color depth (sensor)	10 bit
Color depth (camera)	8 bit
Pixel Class	3 MP
Sensor Size	1/2"
Shutter	Rolling shutter
max. fps in Freerun Mode	11.2
Binning Modes	Color
Subsampling Modes	Color
Sensor Model	MT9T031
Pixel size	3.2 $\mu\text{m}$
Optical Size	6.554 mm x 4.915 mm



#### Design

Interface	USB 2
Lens Mount	CS- / C-Mount
I/O In	-
I/O Out	-
I/O RS-232	-
I/O GPIO	-
I/O I2C	-
IP code	IP30
Dimensions H/W/L	48.6 mm x 44.0 mm x 25.6 mm
Mass	41 g
Power supply	USB Cable

Subject to technical modifications

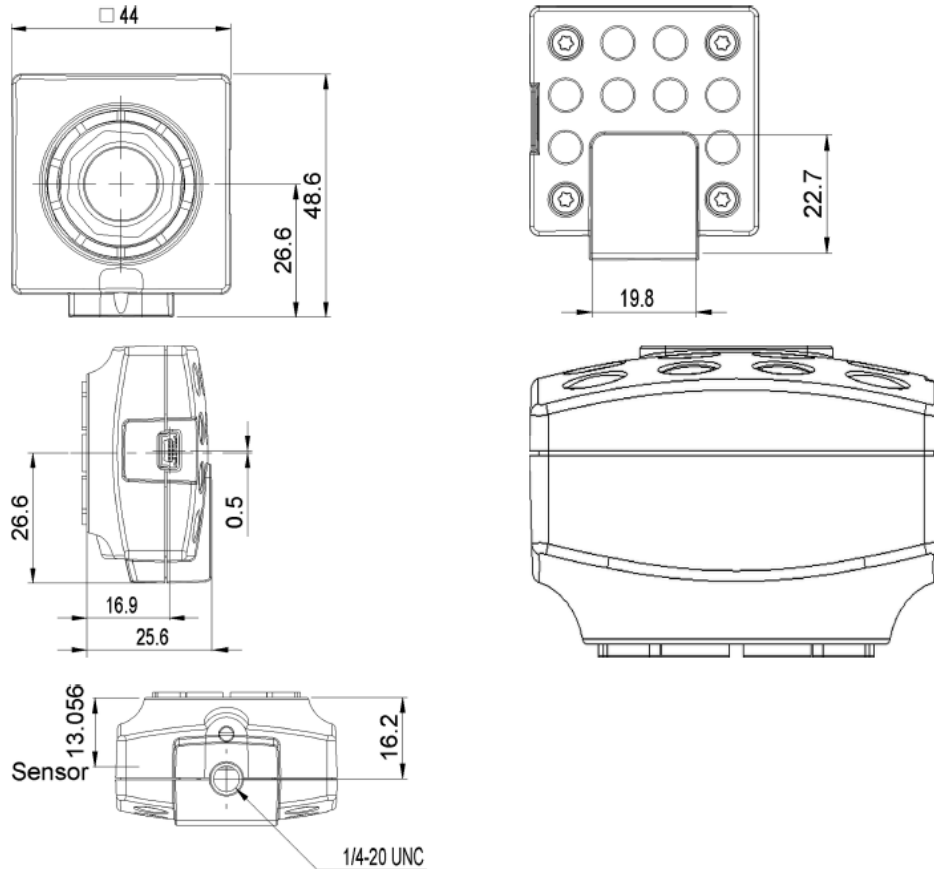
Page 1 of 2

<http://www.ids-imaging.com>

IDS Imaging Development Systems GmbH

Dimbacher Straße 6 · 8 · 74182 Obersulm · Phone +49 7134/96196-0 · Fax +49 7134/96196-99 · E-Mail [info@ids-imaging.de](mailto:info@ids-imaging.de)

## UI-1460LE-C-HQ



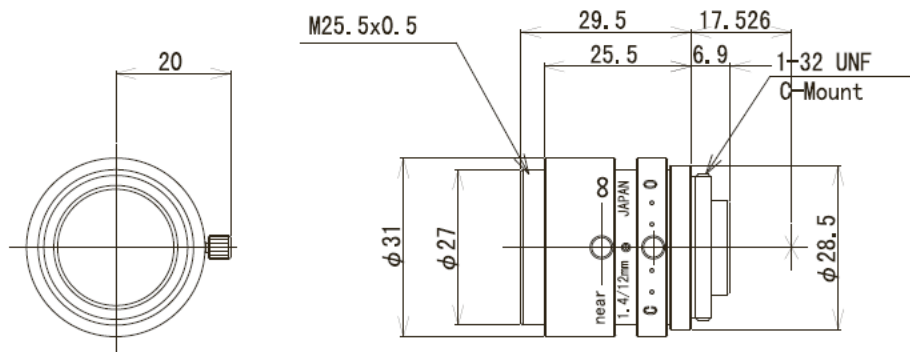
Subject to technical modifications

Page 2 of 2

<http://www.ids-imaging.com>

IDS Imaging Development Systems GmbH  
Dimbacher Straße 6 - 8 · 74182 Obersulm · Phone +49 7134/96196-0 · Fax +49 7134/96196-99 · E-Mail [info@ids-imaging.de](mailto:info@ids-imaging.de)

# Bijlage K: datasheet Kowa LM12NCL (lens)



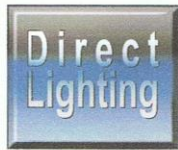
1/2" 1.4/12mm C		1/3"
Focal Length	f=12mm±5%	
Max. Diameter (Ratio)	F=1 : 1.4±5%	
Element	5 Group 6 Elements	
Picture Size	4.8x6.4mm	3.6x4.8
Angle	Ver.	23.0°
	Hol.	30.7°
	Dia.	38.5°
Distortion (TV)	-0.8%	
Shooting Range at M.O.D.	Ver.	123mm
	Hol.	167mm
	Dia.	213mm
Minimum Object Distance	0.3m	
Flange Back	17.526 mm in air	
Back Focus	11.12mm in air	
Filter Screw Size	M30.5x0.5	
Front/Rear Effective Dia.	F: φ12.3mm	R: φ11.7mm
Mount	C-MOUNT	
Exit Pupil	-38.9mm	
Temperature Range	-10~+45° C	
Strong Temperature Range	-40~+60° C	
Resolution	Center:100Line/mm	Corner: 40Line/mm
Stopper Strength	10kgf · cm	
Endurance Tests	Iris:5,000Times	Focus:5,000Times
Operating Torque	Iris:50~600gf · cm	Focus:100~1000gf · cm
Weight	70g±5g	

1/2" 1.4/12mm

LM12NCL

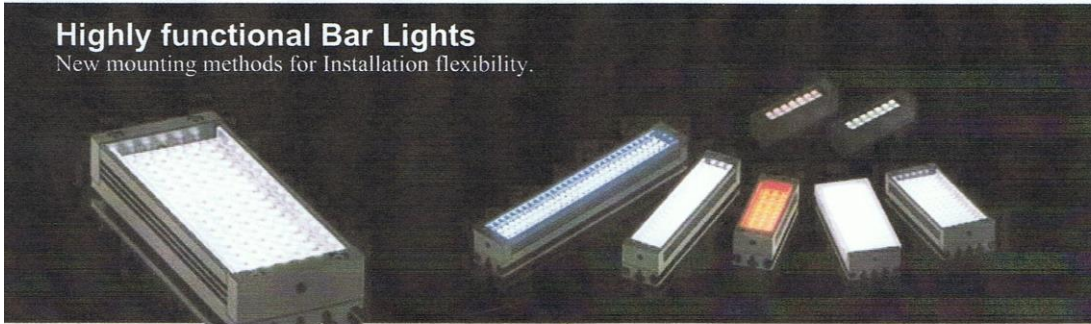
98.8.24

# Bijlage L: datasheet belichtingselementen (catalogo)



## Bar Lights LDL2 Series

**Highly functional Bar Lights**  
New mounting methods for Installation flexibility.



### A Broad Range of Applications

A freely adjustable light direction and angle allow these models to handle a wide variety of applications.

The two figures shown below demonstrate how the illumination's axis and angle can be changed to yield completely different images. This becomes an issue with workpieces with glossy surfaces or parallel grooves, for example. Bar Lights are adjustable allowing you to change the light direction and angle to obtain the optimal image.



Light Direction, Image A



Light Direction, Image B



Image A is washed out because the light reflects straight back from parallel metallic grooves. Image B shows the lettering clearly because the light reflects out of the field of view, leaving the background dark.

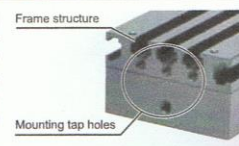
Mounting brackets are available for four different illumination directions. For further details, refer to page 105.



For the LDL2-33x8 Series For the LDL2 Series

### Flexibility of Mounting to Match Your Site Environment

Installation can be achieved with either frame mounting or traditional mounting with tap holes. You therefore have the freedom to select the installation method according to your site environment.



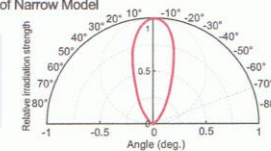
\*The LDL2-33x8 Series provides only tap holes for installation.

### Focus Angle Characteristics of Wide and Narrow Types

There are two directional pattern selections: narrow, which focuses the light into a beam, and wide (WD), which spreads the light out over a broad area. This selection is available over the entire lineup.

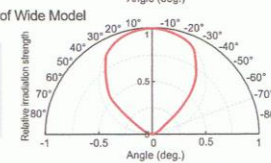
#### Directional Characteristics of Narrow Model

Larger emission surface



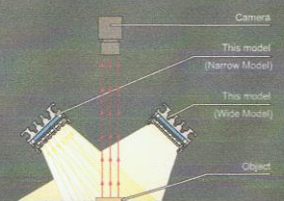
#### Directional Characteristics of Wide Model

Larger emission surface



### Illumination structure of LDL2-74x30

LEDs are arranged at high-density on a single flat circuit board and the work can be illuminated from any angle as desired.



### The LDL2-33x8, the smallest member of the Series, helps you save space

Lightweight, compact designs lend themselves to installation in tight equipment spaces.



\*Only the wide directional pattern is available.

#### Application Examples

Installation in tight equipment spaces is also possible.



Supplementing other lighting is an other possible application.



Direct Number : A direct number is a 7-digit number assigned to a CCS product. You can easily access the web page providing information on any desired product by simply entering the direct number in the space provided on the CCS website pages for machine vision. (Refer to the back cover of this brochure.)

### Product Lineup Table

Series	Direct Number	Model Name	Color	Power Consumption	Option	Dimension		
LDL2	1004646	LDL2-33x8RD	●	24V / 1.3W	D P P B	1		
	1004647	LDL2-33x8SW	○	24V / 0.8W				
	1004648	LDL2-33x8BL	●					
	1004649	LDL2-33x8GR	●	24V / 1.9W	D P C C B	2		
	1003702	LDL2-41x16RD	●					
	1003705	LDL2-41x16SW	○					
	1003704	LDL2-41x16BL	●					
	1003703	LDL2-41x16GR	●					
	1003706	LDL2-41x16RD-WD	●					
	1003709	LDL2-41x16SW-WD	○					
	1003708	LDL2-41x16BL-WD	●					
	1003707	LDL2-41x16GR-WD	●					
	1003710	LDL2-80x16RD	●				24V / 3.8W	D P C C B
	1003713	LDL2-80x16SW	○					
	1003712	LDL2-80x16BL	●					
	1003711	LDL2-80x16GR	●					
	1003714	LDL2-80x16RD-WD	●					
	1003717	LDL2-80x16SW-WD	○					
	1003716	LDL2-80x16BL-WD	●					
	1003715	LDL2-80x16GR-WD	●					
1003718	LDL2-119x16RD	●	24V / 5.7W	D P C C B	2			
1003721	LDL2-119x16SW	○						
1003720	LDL2-119x16BL	●						
1003719	LDL2-119x16GR	●						
1003722	LDL2-119x16RD-WD	●						
1003725	LDL2-119x16SW-WD	○						
1003724	LDL2-119x16BL-WD	●						
1003723	LDL2-119x16GR-WD	●						

\*The peak wavelength for Red lights is 635 nm. If a sharp-cut filter is required, use a R60 Filter (optional).

\*The LDL2-33x8 provides only the wide directional pattern.

\*The following letters indicate options.

D: Diffusion Plate, P: Polarizing Plate, C: Cover, B: Bracket

\*For further details on these options, refer to page 103 to 105.

Existing Bar Light LDL series was discontinued at the end of July, 2011. LDL2 series is recommended as replacement.

Series	Direct Number	Model Name	Color	Power Consumption	Option	Dimension
LDL2	1003726	LDL2-74x30RD	●	24V / 5.7W	D P P C B	3
	1003729	LDL2-74x30SW	○			
	1003728	LDL2-74x30BL	●			
	1003727	LDL2-74x30GR	●			
	1003730	LDL2-74x30RD-WD	●			
	1003733	LDL2-74x30SW-WD	○	24V / 12W	D P P C B	3
	1003732	LDL2-74x30BL-WD	●			
	1003731	LDL2-74x30GR-WD	●			
	1003734	LDL2-146x30RD	●			
	1003737	LDL2-146x30SW	○			
	1003736	LDL2-146x30BL	●	24V / 18W	D P P C B	3
	1003735	LDL2-146x30GR	●			
	1003738	LDL2-146x30RD-WD	●			
	1003741	LDL2-146x30SW-WD	○			
	1003740	LDL2-146x30BL-WD	●			
	1003739	LDL2-146x30GR-WD	●	24V / 21W	D P P C B	3
	1003742	LDL2-218x30RD	●			
	1003745	LDL2-218x30SW	○			
	1003744	LDL2-218x30BL	●			
	1003743	LDL2-218x30GR	●			
	1003746	LDL2-218x30RD-WD	●	24V / 21W	D P P C B	3
	1003749	LDL2-218x30SW-WD	○			
	1003748	LDL2-218x30BL-WD	●			
	1003747	LDL2-218x30GR-WD	●			
	1003750	LDL2-266x30RD	●			
	1003753	LDL2-266x30SW	○			
	1003752	LDL2-266x30BL	●			
	1003751	LDL2-266x30GR	●			
	1003754	LDL2-266x30RD-WD	●			
	1003757	LDL2-266x30SW-WD	○			
	1003756	LDL2-266x30BL-WD	●			
	1003755	LDL2-266x30GR-WD	●			

### Built-to-order models

Series	Model Name	Color	Emitter spacing	Power Consumption	Option	Dimension
LDL2	LDL2-158x16(-WD)	●	158x16mm	24V / 7.6W	D P C C B	2
	LDL2-197x16(-WD)	○	197x16mm	24V / 9.5W		
	LDL2-236x16(-WD)	●	236x16mm	24V / 12W		
	LDL2-275x16(-WD)	●	275x16mm	24V / 14W		
	LDL2-314x16(-WD)	●	314x16mm	24V / 16W		
	LDL2-353x16(-WD)	●	353x16mm	24V / 18W		
	LDL2-392x16(-WD)	●	392x16mm	24V / 19W		
	LDL2-431x16(-WD)	●	431x16mm	24V / 21W		
	LDL2-470x16(-WD)	●	470x16mm	24V / 23W		
	LDL2-509x16(-WD)	●	509x16mm	24V / 25W		

\*The peak wavelength for Red lights is 635 nm. If a sharp-cut filter is required, use a R60 Filter (optional).

\*The optional WD suffix indicates the wide directional pattern.

\*The following letters indicate options.

D: Diffusion Plate, P: Polarizing Plate, C: Cover, B: Bracket

\*For the availability of other options, ask your CCS representative.

Series	Model Name	Color	Emitter spacing	Power Consumption	Option	Dimension
LDL2	LDL2-26x30(-WD)	●	26x30mm	24V / 1.9W	B C P D	3
	LDL2-50x30(-WD)	○	50x30mm	24V / 3.8W		
	LDL2-98x30(-WD)	●	98x30mm	24V / 7.6W		
	LDL2-122x30(-WD)	●	122x30mm	24V / 9.5W		
	LDL2-170x30(-WD)	●	170x30mm	24V / 14W		
	LDL2-194x30(-WD)	●	194x30mm	24V / 16W		
	LDL2-242x30(-WD)	●	242x30mm	24V / 19W		
	LDL2-290x30(-WD)	●	290x30mm	24V / 23W		
	LDL2-314x30(-WD)	●	314x30mm	24V / 25W		
	LDL2-338x30(-WD)	●	338x30mm	24V / 27W		
	LDL2-362x30(-WD)	●	362x30mm	24V / 29W		
	LDL2-386x30(-WD)	●	386x30mm	24V / 31W		
	LDL2-410x30(-WD)	●	410x30mm	24V / 33W		
	LDL2-434x30(-WD)	●	434x30mm	24V / 35W		
	LDL2-458x30(-WD)	●	458x30mm	24V / 37W		
	LDL2-482x30(-WD)	●	482x30mm	24V / 38W		
	LDL2-506x30(-WD)	●	506x30mm	24V / 40W		

### Dimension Diagrams (Unit: mm)

**1.** (Emitting surface)

**2.** (Emitting surface)

**3.** (Emitting surface)

The wide versions (-WD) have the same sizes.

Model Name	A	B	Model Name	A	B
LDL2-41x16(RD/SW/BL/GR)	53	41	LDL2-81x16(RD/SW/BL/GR)	326	314
LDL2-89x16(RD/SW/BL/GR)	92	80	LDL2-353x16(RD/SW/BL/GR)	365	353
LDL2-119x16(RD/SW/BL/GR)	131	119	LDL2-392x16(RD/SW/BL/GR)	404	392
LDL2-158x16(RD/SW/BL/GR)	170	158	LDL2-431x16(RD/SW/BL/GR)	443	431
LDL2-197x16(RD/SW/BL/GR)	209	197	LDL2-470x16(RD/SW/BL/GR)	482	470
LDL2-236x16(RD/SW/BL/GR)	248	236	LDL2-509x16(RD/SW/BL/GR)	521	509
LDL2-275x16(RD/SW/BL/GR)	287	275			

The wide versions (-WD) have the same sizes.

Model Name	A	B	Model Name	A	B
LDL2-26x30(RD/SW/BL/GR)	38	26	LDL2-194x30(RD/SW/BL/GR)	206	194
LDL2-50x30(RD/SW/BL/GR)	62	50	LDL2-218x30(RD/SW/BL/GR)	230	218
LDL2-98x30(RD/SW/BL/GR)	86	74	LDL2-242x30(RD/SW/BL/GR)	254	242
LDL2-122x30(RD/SW/BL/GR)	110	98	LDL2-266x30(RD/SW/BL/GR)	278	266
LDL2-170x30(RD/SW/BL/GR)	134	122	LDL2-290x30(RD/SW/BL/GR)	302	290
LDL2-194x30(RD/SW/BL/GR)	158	146	LDL2-314x30(RD/SW/BL/GR)	326	314
LDL2-242x30(RD/SW/BL/GR)	182	170	LDL2-338x30(RD/SW/BL/GR)	350	338
			LDL2-362x30(RD/SW/BL/GR)	374	362
			LDL2-386x30(RD/SW/BL/GR)	398	386
			LDL2-410x30(RD/SW/BL/GR)	422	410
			LDL2-434x30(RD/SW/BL/GR)	446	434
			LDL2-458x30(RD/SW/BL/GR)	470	458
			LDL2-482x30(RD/SW/BL/GR)	494	482
			LDL2-506x30(RD/SW/BL/GR)	518	506

If you register as a CCS member, you can download all materials (such as PDF or DXF drawings and operation manuals) from our website. You can also request us to select the appropriate Light Unit, borrowing demonstration units and to quote the product's price. Please go ahead and register as a member today. (Refer to back cover of this brochure.)



## Bijlage M: code in Visual Studio voor connectie met database

```
Imports System.Data.SqlClient
Imports System.Globalization
Public Class DatabaseConnectie
    Dim connectieString As String
    Dim connectie As SqlConnection
    Dim aantal As Integer

    Public Sub New()
        connectieString = My.Settings.AfmetingenStripConnectieSQL
        connectie = New SqlConnection(connectieString)
    End Sub

    Public Sub addAfmetingenSQL(ByVal Lengte As Double, ByVal Breedte As Double, ByVal
Hoogte As Double)

        Dim query = "insert into StripAfmetingen(Lengte,Breedte,Hoogte) " & "
values(@Lengte,@Breedte,@Hoogte)"

        Debug.WriteLine(query)
        Dim commando = New SqlCommand(query, connectie)
        commando.Parameters.Add(New SqlParameter("@Lengte", Lengte))
        commando.Parameters.Add(New SqlParameter("@Breedte", Breedte))
        commando.Parameters.Add(New SqlParameter("@Hoogte", Hoogte))

        commando.Connection.Open()
        Dim aantal = commando.ExecuteNonQuery

        commando.Connection.Close()

    End Sub
End Class
```

## **Auteursrechtelijke overeenkomst**

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:  
**Visioncamera gebaseerde kwaliteitscontrole van strippen**

Richting: **master in de industriële wetenschappen: energie-automatisering**  
Jaar: **2014**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

**Scheelen, Frederik**

**Ferson, Wouter**

Datum: **4/06/2014**