# Product Documentatie

Testopstelling –Soil Analyzer

Opdrachtgever: Royal IHC

Opleverdatum: **3 november 2014**

*Jelle Spijker*

*Datum 2 november 2014*
*Revisie 20141102*


**Contact gegevens:**

Jelle Spijker (495653) – 06-43272644 – Spijker.Jelle@gmail.com

# SUMMARY

Since the seventeen century Royal IHC has been at the forefront of dredging development and innovation. Its core business is the building and development of dredging ships and their related tools. A shipping company needs to asses which tools to use for a given job. Deployment and utilisation of these tools are mainly determined by the properties of the soil that needs to be dredged. Therefore these soil samples need to be analysed. This is usually done in land based laboratories. Because of long traveling times, from dredging sites to these laboratories, it's deemed wise to develop a small portable soil analyser which can determine certain key properties by visually inspecting the soil.

Royal IHC has asked Mr. Spijker to develop a prototype which can serve as a basis for a customer ready device. In order for the final prototype to be successful, investigation into the pits and potential risk are needed. This is done with a test setup, where the algorithms are performed on a computer running Matlab r2014b. This test setup is developed, by first determining its main function, namely: *Giving the user information quantifiable information on the soil properties of colour, texture and structure*. With this in mind the specifications are determined, all of these specifications are quantifiable and exact and serve as the basis for the test.

Interaction between the Soil Analyser and the user takes place in Matlab, where the analyser is presented as a Matlab app, with a Graphical User Interface. This app gets its data from the acquisition that takes place with an USB microscope. The user places a dried soil samples under this microscope and brings it in to focus. Afterwards the takes an snapshot. A preliminary user manual describes the actions the user must do in order for Matlab to analyse the soil.

The acquired image is transformed and analysed using five vision steps. These are acquisition, enhancement, segmentation, feature extraction and classification. Acquisition is the process of placing the sample correctly under a microscope. Making sure the individual particles don't overlap and bringing them into focus. At this point the user takes a snapshot which is gathered by Matlab in the form of a matrix in the RGB colorspace. This matrix is transform to an intensity Matrix, Both matrices are used throughout the project. After this enhancement a segmentation take place. During these processes, the image is enhanced in order for to segment of each particle from the background and each other. Hereafter the different features are extracted, because there're multiple analysis that need to be performed, there will also need to be multiple features extracted. There is a colour conversion from RGB to CIE La*b* and RI. Secondary the area for each blob is counted and finally the Fourier Descriptors for each particle contour is determined. These features are then transformed to visual data that the user can interpret. The Fourier Descriptor serve as the basis for particle shape classification. It was original devised to use a Neural Network for this classification. But the attempts made thus far are unsuccessful.

Since this project is a stepping stone for the prototype it is wise to take heed to the challenges that this test setup presented. Further effort should be made on the segmentation algorithms, which failed to detect every particle and also took a relatively long time to execute. It is prudent to look at different classification methods and algorithms for shape classification.

# VOORWOORD

Dit project zit vanaf het begin in een stroomversnelling. Vanuit mijn vorige opleiding ben ik gewend om mijzelf eerst te oriënteren op een opdracht, voordat ik een plan van aanpak en een ontwerp opzet. Veelal gebeurd dit door eerst de literatuur in te duiken en een grove opzet voor een literatuurstudie op te zetten.

Bij deze minor werd er in de eerste week al een plan van aanpak verwacht. Met een plan van aanpak, commit je jezelf aan een opdracht en een opdrachtgever. Deze verwacht dat deze opdracht uitgevoerd wordt zoals omschreven is in het plan van aanpak.

Ik heb in dit plan van aanpak eerst een opdracht en doelstelling met bijbehorende taken en producten geformuleerd. Na het schrijven van mijn literatuurstudie, bleken deze doelstellingen behoorlijk ambitieus te zijn.

Vooralsnog zit ik op schema en zoals het er nu uit ziet moet het eindproduct van volgend blok ook te halen zijn, maar de lange dagen hakken er in.

Om dit document te professionaliseren als school opdracht, is er gekozen voor een fictieve opdrachtgever, namelijk Royal IHC. De in deze minor gehanteerde opdracht is fictief en is niet uitgevaardigd door IHC. Op dit moment ben ik werkzaam bij IHC als engineer machinery and piping. Werkzaamheden welke uitgevoerd worden tijdens deze minor hebben geen parallellen met mijn dagelijkse werkzaamheden. Ik voer deze opdracht op eigen initiatief en in mijn eigen tijd uit.

Het is mijn beoogde doel om het ontwerp van deze Soil Analyzer vrij te geven onder een MIT opensource licentie. Hierdoor kan iedereen vrij profiteren van dit project. Dit doe ik omdat het analyseren van grond ook belangrijk kan zijn bij het verbouwen van gewassen. Wellicht kan dit project helpen bij het verbouwen van gewassen in ontwikkelingslanden.

Vanuit deze licenties staat het IHC vrij om dit concept verder uit te werken en ik zal ze hierbij alle steun geven.

*Jelle Spijker*

# INHOUDSOPGAVE

# 1 Inleiding

ACHTERGROND

Royal IHC houdt zich al sinds de zeventiende bezig met het baggeren. Zij leveren baggerboten en bijbehorende gereedschappen en staan rederijen bij met advies en hulp. Zij willen deze rederijen verder faciliteren, door een portable grond monster analyzer (Soil Analyzer) aan te bieden aan geïnteresseerde klanten. De grondsoort bepaalt namelijk in belangrijke mate welke gereedschap gebruikt moet worden.

Deze analyse wordt meestal aan land uitgevoerd. Een grond monster zal daarom eerst naar een laboratoria getransporteerd moeten worden. Hierdoor duurt het erg lang tot dat de resultaten van deze analyse beschikbaar zijn op locatie. Royal IHC onderzoekt de mogelijkheid om met behulp van computer vision een draagbaar laboratoria aan haar klanten aan te kunnen bieden.

Royal IHC heeft dhr. Spijker gevraagd om een prototype te ontwikkelen. Dit prototype is een embedded device. Om kwaliteit van het prototype te borgen zal eerst een testopstelling voor een personal computer ontwikkeld worden.

BELANG

Omdat analyseren van grond, met behulp van computer vision diverse uitdaging biedt, zal er eerst aangetoond worden dat correcte informatie geëxtrapoleerd kan worden op een personal computer. Door dit proof-of-concept hier op uit te werken, zullen risico, welke bestaan bij het ontwikkelen van een embedded computer vision device, inzichtelijk en/of geëlimineerd worden.

Dit product documentatie geeft inzicht in het te ontwikkelen testopstelling. Door functies en gestelde eisen van een testopstelling te omschrijven en uit te werken in een vision ontwerp en realisatie, kan in een vroeg stadia inzichtelijk gemaakt worden welke uitdagingen het prototype zal bieden.

PROBLEEM

Grond monsters kunnen visueel geanalyseerd worden op drie aspecten: kleur, textuur en structuur. Alle drie deze aspecten brengen hun eigen uitdagingen. Kleur zal gepresenteerd worden in een histogram in het CIE LAB en RI kleuren model; Dit vereist een efficiënte conversie vanuit RGB. De textuur van een monster wordt gepresenteerd met een Particle Size Distributie. Hierbij worden individuele korrels geteld en ingedeeld in bins gebaseerd op hun grootte. Doordat korrels elkaar raken of overlappen, zal er een verstoring van de werkelijke populatie zijn; Deze verstoring moet binnen acceptabele grenzen liggen. Structuur onderzoekt de vorm van de korrels en classificeert deze. Deze classificatie zal snel en efficiënt plaats moeten vinden.

WERKWIJZE

Deze product documentatie geeft inzicht in het functioneren van een testopstelling voor de ontwikkeling van een Soil Analyzer. Als eerst wordt er een globaal Input-Proces-Output opgezet. Hieruit kan de functionele specificatie geschreven worden. Hierin wordt onderscheidt gemaakt tussen functionele en technische eisen. Om de werking verder te illustreren wordt een User Interface uitgezet. Deze wordt bijgelicht met een korte gebruikershandleiding; Geschreven voor een eindgebruiker en een beheerder. Vanuit bovenstaande hoofdstukken kan een gedetailleerde Input-Proces-Output schema uitgewerkt worden. Hierna wordt een voortgangstest omschreven. Uit deze test kan getoetst worden of het systeem, functioneert en wat er verbeterd moet worden voor het prototype.

Vanuit deze hoofdstukken wordt een vision ontwerp opgesteld. Dit ontwerp toont de gewenste input en output van iedere vision stap. Vanuit dit ontwerp kan de realisatie uitgewerkt worden. Deze realisatie wordt uitgevoerd en hierna kan het product getest worden, met eerder opgestelde test.

RANDVOORWAARDE

Deze product documentatie wordt geschreven voor de ontwikkeling van een testopstelling waar vanuit een prototype ontwikkeld kan worden. Vanwege het preliminaire karakter van dit systeem is de gebruikelijke acceptatietest vervangen voor een voortgangstest. Deze test dient enkel als indicatie.

STRUCTUUROMSCHRIJVING

Dit document heeft de volgende structuur. In hoofdstuk twee wordt een functioneel ontwerp neergezet, hierna wordt in hoofdstuk drie een vision ontwerp uitgewerkt, welke als basis dient voor het vision realisatie in hoofdstuk vier. De testopstelling wordt in hoofdstuk vijf omschreven, hierna worden de testen en behaalde resultaten in hoofdstuk vijf uitgezet. In hoofdstuk zeven is een conclusie en aanbevelingen te vinden zijn.

# 2 Functioneel Ontwerp

In dit hoofdstuk wordt het ontwerp op het hoogste niveau omschreven. Dit wordt inzichtelijk gemaakt door eerst een Input-Proces-Output (IPO) schema op te stellen. Hierna kunnen de functionele specificatie opgesteld worden. Waarna een User Interface (UI) omschreven wordt. Gebruik van deze userinterface wordt doormiddel van een handleiding geïllustreerd. Uit bovengenoemde paragraven kan een gedetailleerde IPO opgesteld wordt. Als laatste wordt omschreven hoe de opstelling getest kan worden tegen de functionele specificaties.

## 2.1 Globaal Input-Proces-Output schema

In hoofdstuk twee wordt de belangrijkste functie van de testopstelling weergeven, hier de hoofdfunctie omschreven. Dit wordt zowel tekstueel als met een IPO-schema gedaan. IPO staat voor Input-Proces-Output, dit is een zogenaamde black box benadering van het systeem. Hier wordt input aan een proces aangereikt, welke deze input aanpast en een output geeft.

SYSTEEM OMSCHRIJVING

De testopstelling voor een grond monster analyzer (SA). Presenteert een analyse van eigenschappen van grond aan de eindgebruiker. Deze informatie wordt gevormd door de visuele karakteristieken van dit monster te onderzoeken. De gebruiker positioneert een grond monster onder een microscoop, waarna deze geanalyseerd wordt op kleur, structuur en textuur. Deze informatie wordt aan de gebruiker gepresenteerd in een Matlab 2014a Graphical User Interface (GUI).

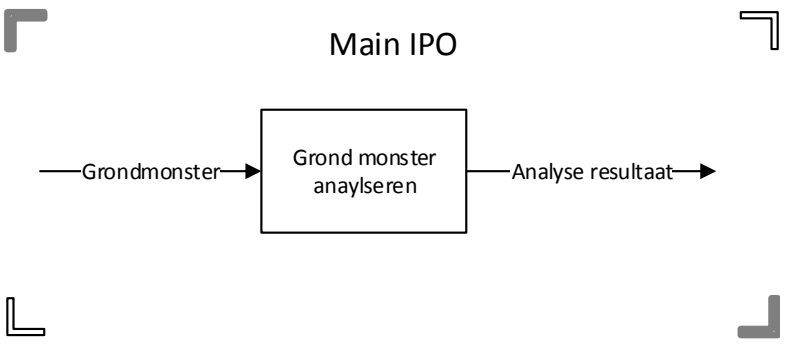
***TECHNISCH SYSTEEM:***

*Testopstelling een grond monster analyzer.*

***HOOFDFUNCTIE:***

*Een grondmonster analyseren en geëxtrapoleerde informatie over kleur, textuur en structuur begrijpelijke en bruikbaar aan een gebruiker terugkoppelen.*

MAIN-IPO




*Figuur 1 Globale IPO van de testopstelling*

## 2.2 Functionele specificaties

In dit hoofdstuk worden de functionele eisen voor een testopstelling van een grond monster analyzer uitgezet. Hierin wordt onderscheid gemaakt tussen functionele en technische eisen. Functionele eisen worden vereiste door een eindgebruiker terwijl technische eisen gedicteerd worden door een opdrachtgever. Onderbouwing van deze eisen wordt gegeven in het literatuuronderzoek: (Spijker, 2014), welke als bijlage is bijgevoegd.

FUNCTIONELE & TECHNISCHE EISEN

### 2.2.1 Functionele eisen

| ID | Omschrijving | Type | Bron |
|---|---|---|---|
| **F1** | **Kwantificeren van kleur** | | |
| F1.1 | Kleur bepalen van alle zichtbare grondkorrels | Vast | |
| F1.2 | Scatterplot van kleuren in CIE La*b* kleurenmodel alleen (a* en b*) | Vast | Lit. |
| F1.3 | Histogram van kleuren in RI (Redness Index) | Vast | Lit. |
| **F2** | **Kwantificeren van textuur** | | |
| F2.1 | Korrel grootte analyseren in het bereik $4[\mu m] \leq P_{size} \leq 2[mm]$ | Vast | PvA |
| F2.2 | Alleen "volledig" zichtbare korrels meten | Vast | Lit. |
| F2.3 | Betrouwbaarheidsinterval van 90% hanteren | Vast | Lit. |
| F2.4 | Het resultaat moet binnen 10% nauwkeurigheid vallen | Vast | Lit. |
| F2.5 | PSD in volume metrische bins categoriseren | Vast | Lit. |
| **F3** | **Kwantificeren van structuur** | | |
| F3.1 | Korrel grootte analyseren in het bereik van $63[\mu m] \leq P_{size} \leq 2[mm]$ | Vast | Lit. |
| F3.2 | Onderscheid maken tussen rond- en hoekigheid in 6 gradaties | Vast | Lit. |
| F3.3 | Onderscheid maken tussen bolvorming in 3 gradaties | Vast | Lit. |
| F3.4 | Alleen "volledig" zichtbare korrels analyseren | Vast | Lit. |
| F3.5 | Voorspelde t.o.v. werkelijk waarde $R \geq 0.9$ voor enkele korrel (lineair regressie) | Vast | Lit. |
| F3.6 | Betrouwbaarheidsinterval voor volledige monster van 90% | Vast | Lit. |
| F3.7 | Het resultaat moet binnen 10% nauwkeurigheid vallen | Vast | Lit. |
| **F4** | **User interface Software** | | |
| F4.1 | Alle data getoond op 1 scherm (1366 x 768) | | |
| F4.2 | Scatterplot van CIE LAB (in a*b* ruimte tonen) | Vast | Lit. |
| F4.3 | Histogram van RI (Alleen bins tonen) | Vast | Lit. |
| F4.4 | Histogram van PSD (Bins tonen aangevuld met tekstuele omschrijving van median, mean, standard deviation, skewness, kurtosis) | Vast | Lit. |

| | | | |
|---|---|---|---|
| F4.5 | RGB foto tonen van grond monster | Vast | |
| F4.6 | Preview modus om camera scherp te stellen | Vast | Lit |
| **F5** | **User interface hardware** | | |
| F5.1 | Gebruiker plaats zelf monster onder camera | Vast | |
| F5.2 | Gebruiker stelt zelf scherp | Vast | |

### 2.2.2 Technische eisen

| ID | Omschrijving | Type |
|---|---|---|
| T1 | Software omgeving | |
| T1.1 | Programma runt in Matlab 2014a | Vast |
| T1.2 | GUI is dockable | Vast |
| T1.3 | Code is vrij gegeven on MIT licentie | Vast |
| T1.4 | Code is in Engels gedocumenteerd | Vast |
| T2 | Hardware omgeving | |
| | geen | |

## 2.3 User Interface

DOEL VAN EEN UI

Het belangrijkste doel van een SA is het overbrengen van informatie naar een gebruiker. In (Spijker, 2014) wordt omschreven welke informatie uit een grondmonster gehaald kan worden. Dit gebeurd aan de hand van haar optische eigenschappen. Hier wordt verder omschreven hoe deze informatie overgebracht kan worden naar de gebruiker. Dit hoofdstuk omschrijft de interactie tussen gebruiker en testopstelling.

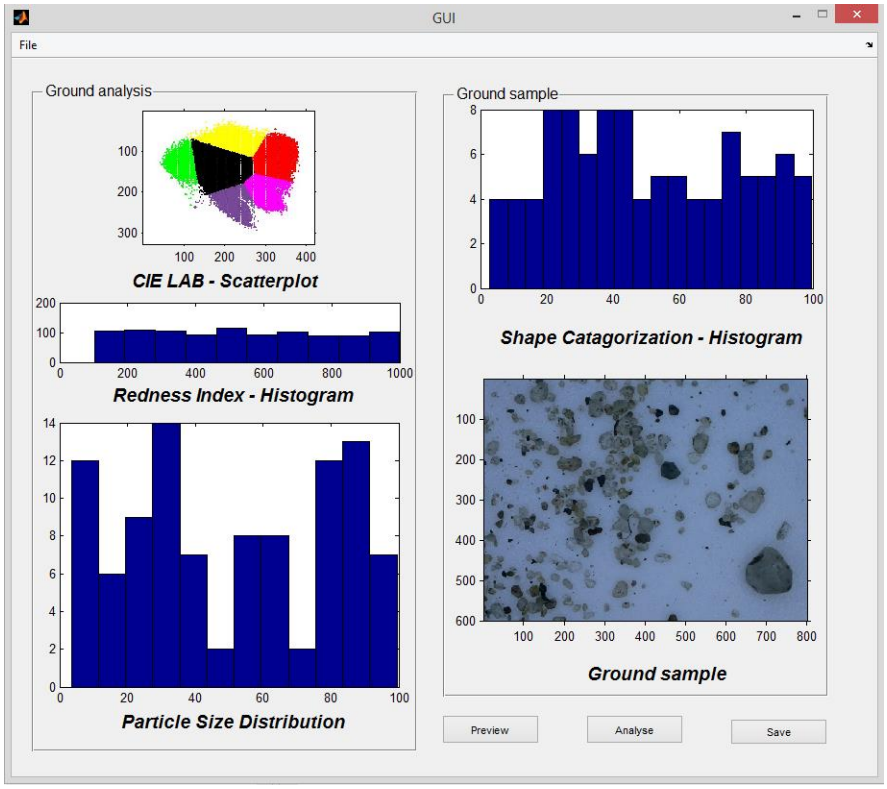
WAAROM MATLAB?

Er wordt bij MTI en IHC, veel gebruik gemaakt van Matlab, om de werking van het concept te illustreren, wordt deze testopstelling uitgevoerd in Matlab 2014a. De geschreven app toont de verkregen informatie in een Matlab GUI. Dit wordt verder uitgezet in paragraaf 4.1.

Voordat de computer het grondmonster kan analyseren, zal deze eerst onder de CMOS sensor geplaatst moeten worden. Verder interactie tussen testopstelling en gebruiker vind plaats bij de hardware interface, welke in paragraaf 4.2 wordt uitgezet.

### 2.3.1 Graphical User Interface

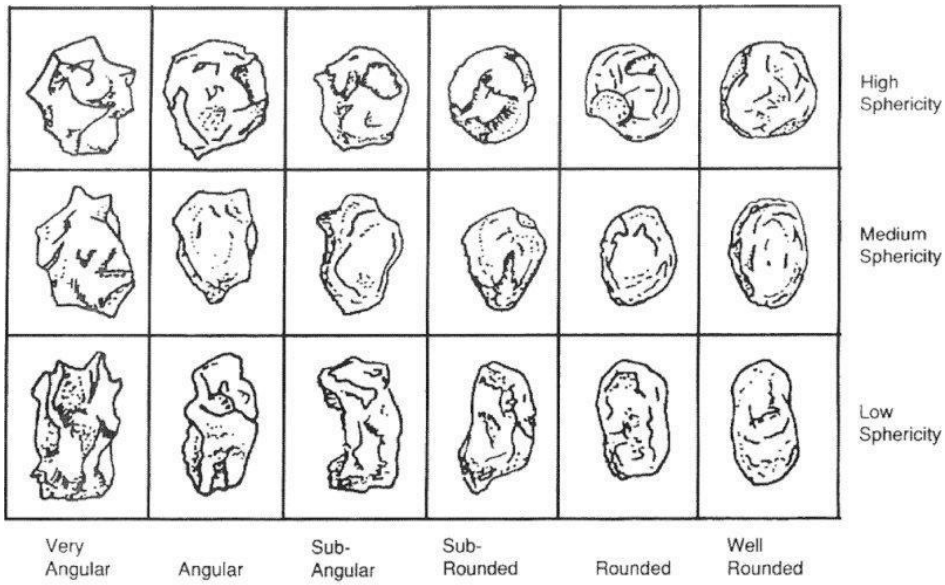
Een belangrijk deel van de interface tussen gebruiker en testopstelling is een GUI. Via deze GUI krijgt een gebruiker relevante informatie over de eigenschappen van een grondmonster. Hieronder is een eerste opzet gemaakt. Precieze weergave en indeling kan nog aan veranderingen onderhevig zijn.




*Figuur 2 Concept GUI van de testopstelling*

In deze GUI wordt minstens het volgende getoond:

- RGB Image van de geanalyseerde grondmonster. Een indicatie van de korrel grootte wordt verkregen met een schaalverdeling. IMAGE
- CIE La*b* scatterplot, hierin worden een individuele pixel, welke behoort tot een grondkorrel, weergegeven op een scatterplot welke a* ten opzichte van b* illustreert. Hierbij is a* de chromatische as rood-groen en b* de chromatische as blauw-geel. In (Spijker, 2014) wordt de keuze voor dit kleurmodel gemotiveerd. KLEUR
- RI Histogram, hierin worden individuele pixels, welke behoren tot een grondkorrel, geteld en gecategoriseerd volgens een Redness-Index. In (Spijker, 2014) wordt de keuze voor dit kleurmodel gemotiveerd.
- PSD, hierin wordt de onderlinge volume metrische verhoudingen van korrels weergegeven in de histogram, Dit wordt aangevuld met een met tekstuele omschrijving van de median, mean, standard deviation, skewness, kurtosis van deze verdeling. TEXTUUR
- Particle shape catagorization, hierin wordt de distributie van vormen in de sample weergegeven. Deze vormen zijn in 18 categorie verdeeld welke veelal op vorm vergelijkingskaarten zijn terug te vinden. STRUCTUUR



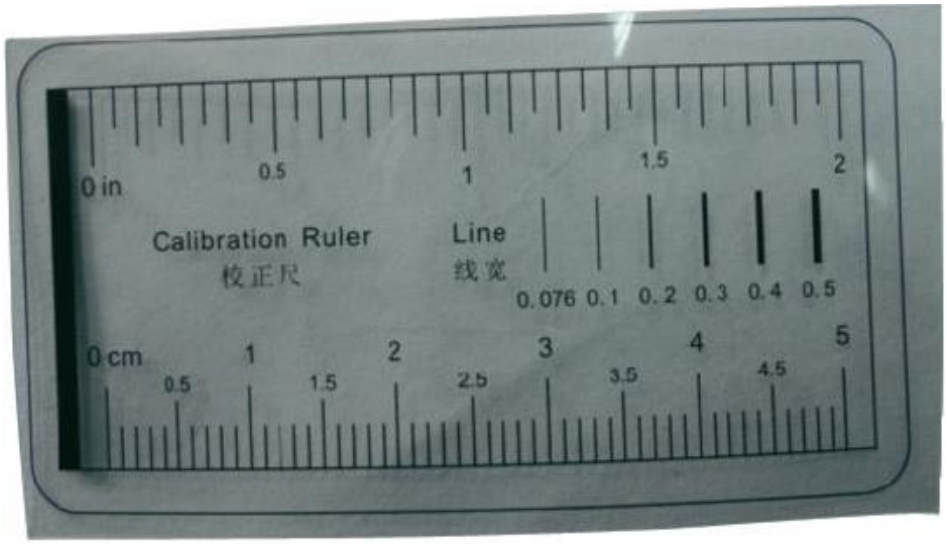

*Figuur 3 Kaart om korrel vormen te vergelijken en classificeren bron:(Miller & Henderson, 2010)*

### 2.3.2 Hardware User Interface

Grondmonster worden voor dit project bewaard in Petri schaaltjes. Deze monster worden gepositioneerd onder een USB microscoop, zie het figuur hieronder. De individuele korrels zullen zo veel als mogelijk van elkaar gescheiden en enkel laags gepositioneerd worden. Hierna zal een gebruiker de vergrotingsfactor en het scherpstellen handmatig uit voeren. Als laatst zal de camera gekalibreerd worden met een kaliber, zie het onderste figuur.

POSITIONEREN VAN EEN GRONDMONSTER

*Figuur 4 USB microscoop bron: (Conrad, 2014)*




*Figuur 5 Kalibratieregel voor de USB microscoop bron: (Conrad, 2014)*

## 2.4 Handleidingen

Verdere interactie van gebruiker en testopstelling wordt uitgezet in een handleiding. In dit hoofdstuk worden twee handleidingen aangereikt. In paragraaf 5.1 is een gebruikershandleiding geschreven, waarin het normale gebruiksprotocol omschreven wordt. Met deze handleiding kan een eind gebruiker, een grondmonster analyseren. In paragraaf 5.2 wordt een handleiding voor een beheerder beschreven.

### 2.4.1 Gebruikershandleiding

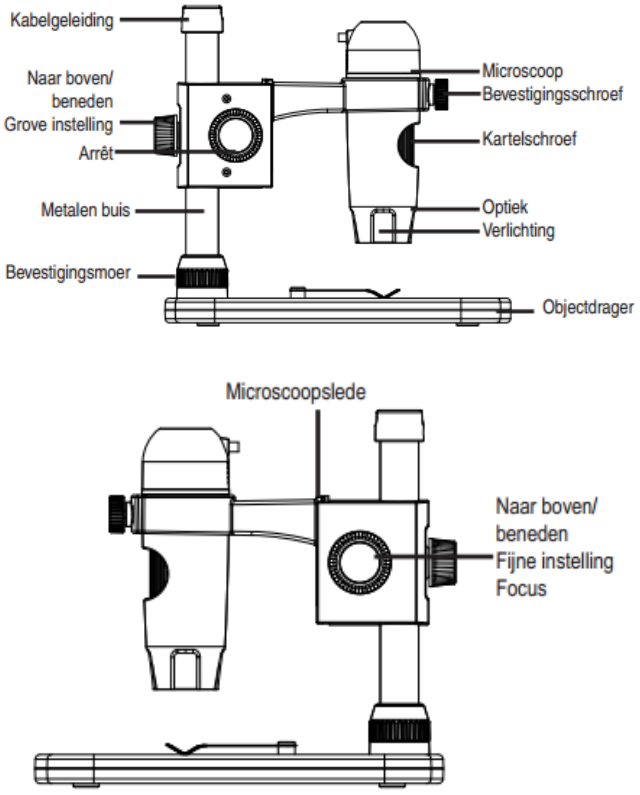
OPSTARTEN

Een gebruiker start de Matlab app vanuit de command window in Matlab, met het volgende commando: >> SA. Dit start de GUI, waarna een gebruiker op de preview knop kan drukken om de stream van de USB microscoop te zien. Onder deze microscoop wordt een willekeurige gekozen steekproef geplaatst. De minimale grootte van deze steekproef wordt op een later tijdstip bepaald. De gebruiker draagt zorgt om deze korrels enkel laags en gescheiden onder de microscoop neer te leggen.

MONSTER PLAATSEN

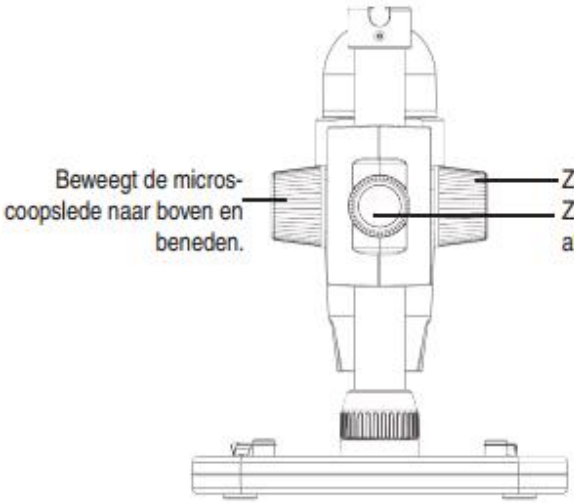
FOCUS

Hierna wordt de microscoop boven het monster gepositioneerd en wordt de minimaal benodigde vergroting handmatig ingesteld. Dit eerst wordt ingesteld met de draai knop: “grove instelling” waarna, met de “kartelschroef” de korrels in focus gebracht worden. Daarna kan het finetunen plaats vinden met de “fijne instelling focus” knop.

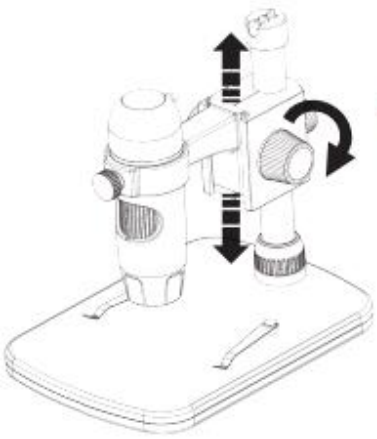



*Figuur 6 Interface van de microscoop bron:(Conrad, 2014)*

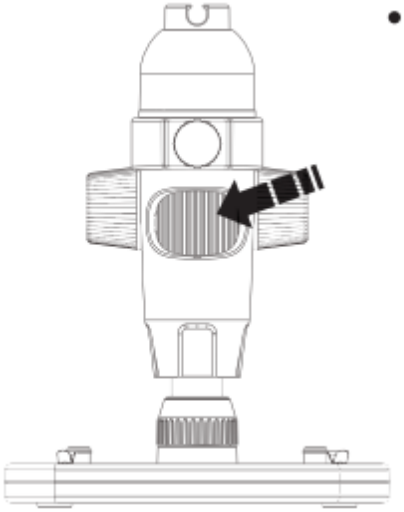
**Achteraanzicht**




- Stel grof in aan de schroef op de achterzijde van het statief. Maak de schroef los en breng dan de houder op een geschikte hoogte naar het object.



- Beweeg nu de slede naar een geschikte hoogte naar het object. U kunt daarmee grof focusseren.



- Voer vervolgens met de kartelschroef een fijne focus-sering resp. vergroting door.

Zodra het monster scherp gesteld onder de microscoop, drukt de gebruiker op de analyse knop. Deze knop neemt een snapshot van het huidige frame en voert de analyses uit. Indien de analyse langer dan 10 seconde zal duren, krijgt de gebruiker een voortgangsdialoog. Deze verdwijnt zodra de analyse klaar is.

KALIBRATIE

Op dit moment wordt een popup getoond waarin de gebruiker gevraagd wordt om de kalibratiemeetlat onder de lens te schuiven. Hierbij moet niet gelet worden dat de camera in dezelfde setting blijft. Door op de kalibratie knop te drukken wordt de werkelijke Euclidische afstanden van de pixels verkregen.

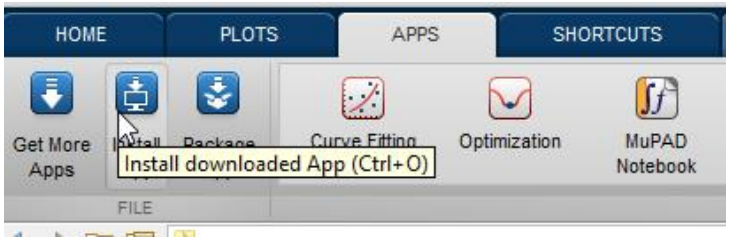
RESULTAAT

De uiteindelijke analyse wordt getoond in de GUI. Hierna heeft de gebruiker de mogelijkheid om de verkregen analyse data op te slaan als Matlab variabel door op de save knop te drukken. De app kan afgesloten worden door op het x-je te drukken.

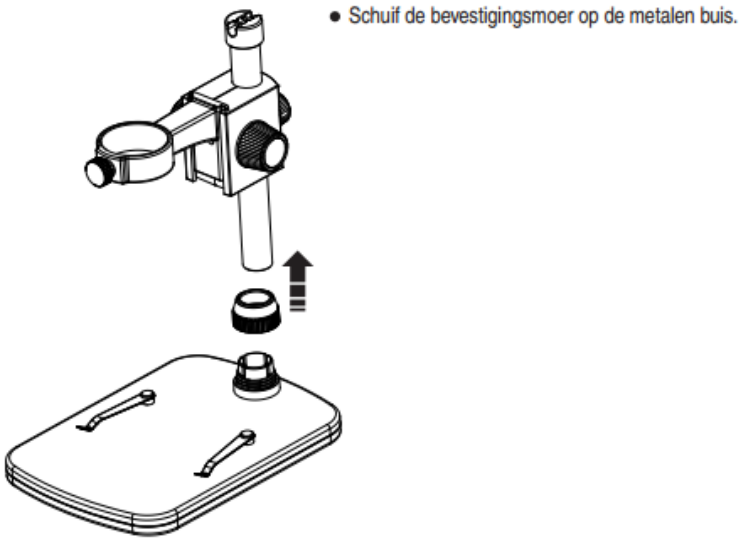
### 2.4.2 Beheerdershandleiding

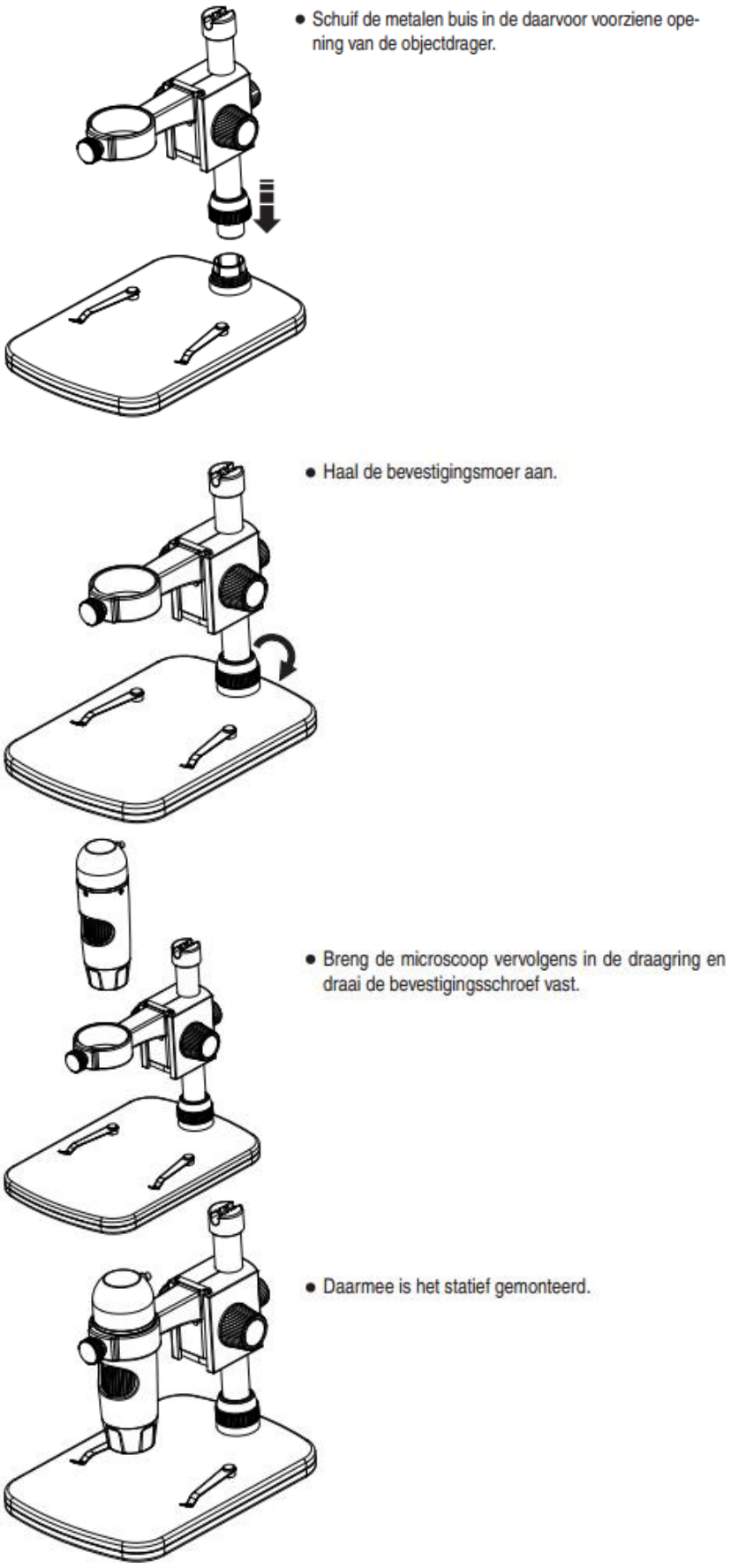
INSTALLATIE

De Matlab app kan gedownload worden vanuit de Matlab file server (http://www.mathworks.nl/discovery/matlab-apps.html ) Door op de "Install app" knob te drukken van de Matlab ribbon (zie figuur hieronder) kan de app geïnstalleerd worden.




De eerste ingebruikstelling van de hardware gaat als volgt:



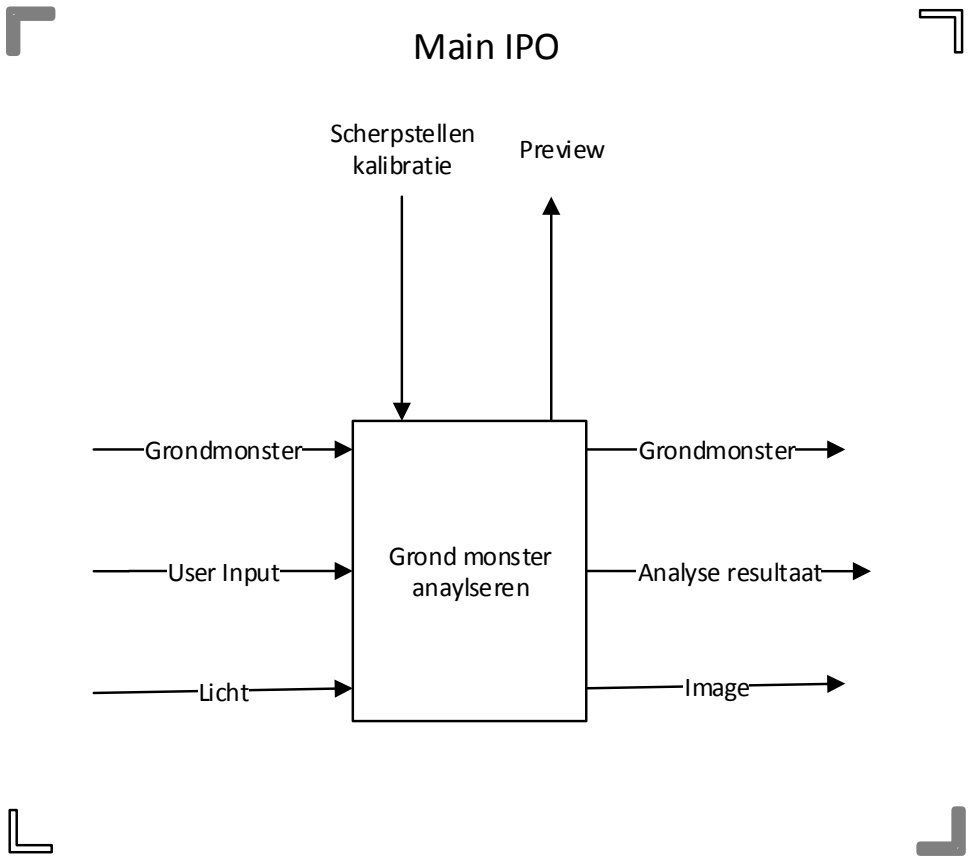



*Figuur 7 Eerste ingebruikstelling bron:(Conrad, 2014)*

## 2.5 Gedetailleerd Input-Proces-Output schema's

In hoofdstuk 2 is een voorlopig IPO schema geschetst, in die versie zijn alle stromen nog niet in balans. Vanuit de opvolgende hoofdstukken zijn deze stromingen verder uitgewerkt en weergeven in onderstaande IPO. In dit hoofdstuk wordt er verder ingegaan op deze stromingen.

GEDETAILLEERDE IPO




*Figuur 8 Uitgewerkte IPO van het hoofdproces*

STROMEN

Zoals duidelijk wordt uit het bovenstaande schema, rust de Matlab testopstelling op de volgende stromen. Als eerst wordt er een grondmonster aangeleverd. Deze wordt gedurende het proces niet verbruikt en wordt aan het eind weer weggenomen door de gebruiker. De testopstelling runt in een Matlab omgeving en maakt gebruik van deze resources. Het resultaat van de analyse wordt in een Matlab GUI getoond of weg geschreven als Matlab variabelen. Ook zal er licht op het monster geschenen worden, weerkaatsing van dit licht wordt opgevangen door een CMOS sensor waarna er een digitale image als resultaat uit komt.

## 2.6 Voortgangstest

In dit hoofdstuk wordt een voortgangstest aangereikt. Deze test kan gebruikt worden om de voortgang en prestaties van de testopstelling in kaart te brengen. Zoals omschreven in de randvoorwaarde welke te vinden zijn in de inleiding heeft deze testopstelling een preliminair karakter en wordt deze enkel gebruikt ontwikkelingsrisico's van het prototype te beperken. Ook dient deze om een gedetailleerd beeld van de uiteindelijke werking aan de opdrachtgever te schetsen. Om de voortgang, ontwikkeling en prestaties van de testopstelling in kaart te brengen kan gebruikt worden gemaakt van onderstaande voortgangstest. Deze test dient enkel als indicatie.

WAAROM VOORTGANGSTEST

Deze test maakt gebruik van de functionele en technische eisen. Welke onderverdeelt zijn in vaste en variabele. Vaste eisen moeten gehaald worden omdat deze de essentie van de testopstelling omschrijven. Deze eisen zijn ook geschreven dat ze alleen met een ja/nee beantwoord kunnen worden. Variabele eisen hebben betrekking op de prestaties van het systeem. Deze kunnen kwantitatief weergegeven worden, veelal worden deze gemeten of berekend.

In de test is omschreven welke eis getest wordt en of deze vast of variabel is. Vaste eisen worden afgevinkt terwijl variabele eisen gekwantificeerd weergeven worden. Onder de eis is de testmethoden omschreven. Deze test is in bijlage I te vinden.

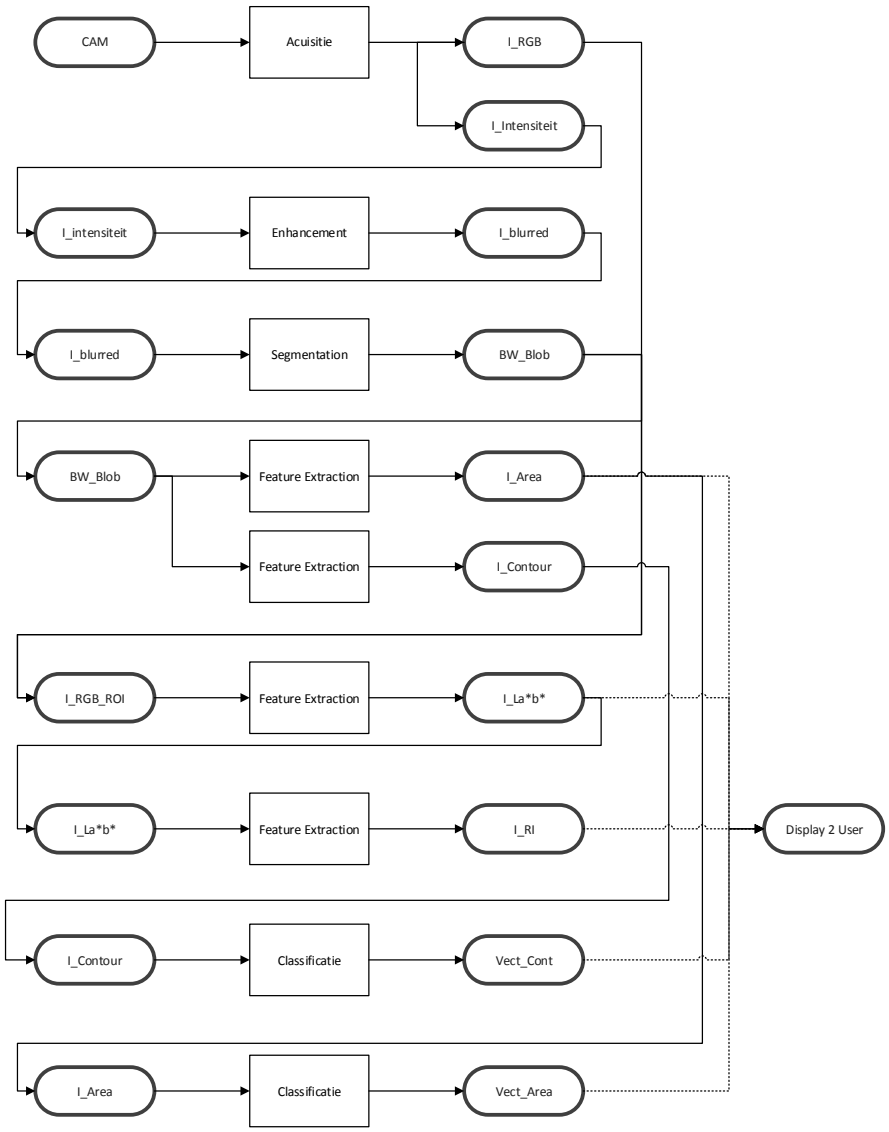
# 3 Vision Ontwerp

In onderstaande schema worden input en output van de benodigde Vision stappen omschreven Dit proces zal vijf stappen omvatten. Acquisition – het verkrijgen van het beeld; Enhancement – het verbeteren van het beeld; Segmentatie – het scheiden van de blobs ook wel de interessante objecten; Feature extraction – extraheren van karakteristieken en classificatie - indelen van het object in classes. In bijlage XIII.

INPUT EN OUTPUT VAN IEDERE VISION STAP

In onderstaande diagram zijn input en output van ieder proces weergegeven het kan zijn dat een enkele proces, meerdere in- als outputs kan hebben. Hierbij worden de volgende naam conventies gehanteerd:

- I = matrix met een 1- of 3-tuple vector bestaande uit discrete waardes tussen 0 en 255
- BW = Binaire matrix met discrete waardes van 0 of 1
- Vect = Vector met reële waardes



# 4 Vision Realisatie

In het volgende hoofdstuk wordt omschreven welke stappen er uitgevoerd worden om zand monsters te analyseren. Dit proces zal vijf stappen omvatten. Acquisition – het verkrijgen van het beeld; Enhancement – het verbeteren van dit beeld; Segmentatie – het scheiden van blobs; Feature extraction – extraheren van karakteristieken en classificatie uit het beeld - indelen van het object in classes. Voor iedere stap worden gebruikte methodes omschreven en gemotiveerd. In bijlage XIV is een flowchart van deze vision stappen weergegeven.

## 4.1 Acquisition

USB MICROSCOOP

Een van de eerste stappen is het verkrijgen van het te analyseren beeld. Zoals in hoofdstuk twee omschreven is, wordt een zandmonsters onder een USB microscoop met CMOS sensor geplaatst. Het correct plaatsen, scherpstellen en belichten wordt door de gebruiker uitgevoerd.

GESCHEIDEN KORRELS

Een gebruiker dient zorg te dragen dat alle korrels zoveel als mogelijk gescheiden liggen. Dit is belangrijk omdat korrels welke overlappen en/of raken, niet goed gedifferentieerd kunnen worden. Deze worden waargenomen als één enkele korrel. Deze waarnemingen verstoren de Particle Size Distributie (PSD). Zie bijlage II voor een uitgebreide motivatie.

Een gebruiker zal alle korrels zelf in focus moeten brengen. De afstand van lens tot bodemplaat zal ongeveer drie centimeter bedragen. Op deze afstand kan er gebruik gemaakt worden van twee brandpunt afstanden. Het is belangrijk dat de gebruiker kiest voor de afstand welke de vorm en afmeting van individuele korrels recht doet en tevens een korrel populatie van minimaal 68 korrels toont. Deze populatie moet willekeurig geplaatst zijn. Deze is theoretisch bepaald met onderstaande formule.

MINIMALE MONSTER POPULATIE

$$n_{th} \geq \frac{-p(p-1)z^2}{\alpha^2} \geq 68.06$$
$$z = 1.65$$
$$p = 0.5$$
$$\alpha = 0.1$$

VERLICHTING & BEAGLEBONE BLACK

Voorlopige testen hebben aangetoond dat belichting van een monster uniform en reproduceerbaar uitgevoerd moet worden. In het onderzoeksrapport (bijlage III) wordt aangetoond dat gewoon daglicht varieert gedurende de dag. Hierom wordt de microscoop geplaatst in een witte box met afmetingen van $300[mm] \cdot 300[mm] \cdot 300[mm]$ en wordt het monster door drie witte LED's beschenen. Deze LED's staan rondom het monster op intervallen van 120° onder een hoek van 45° . Aan en uit

switchen wordt uitgevoerd door een Beaglebone black. In bijlage VIII, IX, X en XV zijn respectievelijk de bijbehorende datasheets en een elektronica schema te vinden. Van deze opstelling.

ACQUISITION

Zodra een monster geplaatst is en alle korrels gescheiden zijn. Kan de microscoop scherp gesteld worden, zie paragraaf 2.4.1. Hierna kan een snapshot gemaakt worden. Deze image wordt binnengehaald door een manuele gebruikers actie, nl. het indrukken van een knop in de interface. Deze image komt binnen als een Matlab matrix $I$ waarbij de kleur van iedere pixel vertegenwoordigd wordt door een discrete triple welke een waarde kan aannemen tussen 0 en 255. Deze omschrijft een vector omschrijft en het RGB kleurenmodel.

$$I_{RGB} = \begin{pmatrix} \vec{I}_{11} & \cdots & \vec{I}_{1n} \\ \vdots & \ddots & \vdots \\ \vec{I}_{m1} & \cdots & \vec{I}_{mn} \end{pmatrix} : \left\{ \vec{I}_{n,m} \in \mathbb{Z} : 0 \le \vec{I}_{n,m} < 256 \right\}$$

CONVERSIE

Deze wordt geconverteerd naar een intensiteit matrix, in grijswaardes. Dit gebeurd met de Matlab step functie uit de vision toolbox, deze is beduidend sneller dan rgb2hsv (zie bijlage XI). Beide matrixen $I_{RGB}$ en $I_i$ worden het gehele proces gebruikt. De RGB matrix zal gebruikt worden voor kleur analyses. Vanuit de intensiteit matrix zullen de blobs gegeneerd worden.

$$I_i = \begin{pmatrix} I_{11} & \cdots & I_{1n} \\ \vdots & \ddots & \vdots \\ I_{m1} & \cdots & I_{mn} \end{pmatrix} : \left\{ I_{n,m} \in \mathbb{Z} : 0 \le I_{n,m} < 256 \right\}$$

## 4.2 Enhancement

Als eerst worden de intensiteit matrix geconverteerd naar een binaire matrix, dit gebeurd met een threshold. Deze waarde is het gemiddelde van alle intensiteiten, hierbij wordt nogmaals de helft van de standaard afwijking afgetrokken.

$$BW < \bar{I}_i - st.dev(I)^{-2} \left\{ BW \in \mathbb{Z} : \mathrm{BW} = 0 \vee 1 \right\}$$

MORFOLOGISCH

Doormiddel van een morfologische closing operations, uitgevoerd met een disk, welke een radius van 20 heeft, worden gaten en spleten tussen de korrels dichtgesmeerd. Deze binaire matrix wordt met behulp van 8-connected labelling, gelabeled. Ieder van deze blobs is een Region Of Interest (ROI), deze wordt vergroot met een offset van -5. Vanuit deze intensiteit matrix wordt deze ROI verder onderzocht. Hierover heen worden een Gaussian blur morfologische operatie uitgevoerd met een kernel van 5x5 en een $\sigma = 0.6$

ROI

$$K(\sigma)=\begin{pmatrix} a_{11} & \cdots & a_{11} \\ \vdots & \ddots & \vdots \\ a_{51} & \cdots & a_{55} \end{pmatrix}$$

### 4.3 Segmentation

SEGMENTATIE

Voor ieder eerdere bepaalde ROI wordt een nieuwe conversie uitgevoerd naar een binaire matrix. Dit gebeurd met behulp van een threshold met een waarde het gemiddelde van alle intensiteiten in de desbetreffende ROI, hierbij wordt nogmaals één derde van de standaard afwijking opgeteld.

$$BW_{ROI} < \overline{I}_{iROI} + st.dev_{iROI}(I)^{-3}\left\{BW_{ROI} \in \mathbb{Z} : \mathrm{BW}_{ROI} = 0 \vee 1\right\}$$

Doormiddel van het eroderen van iedere blob met een disk van $r_{disk} = 0.025\sqrt{\overline{A}}$, ontstaan nieuwe accuratere blobs. Deze blobs worden in een nieuwe binaire matrix geplaatst. Deze matrix correspondeert qua dimensies met de oorspronkelijke binaire matrix. Hierna worden blobs welke zich aan een rand van een matrix bevinden verwijderd.

### 4.4 Feature extraction

Er worden vier verschillende features onderzocht, twee gerelateerd aan kleur, één aan oppervlakte en de andere aan omtrek. Kleuren features worden alleen uitgevoerd op korrels. Het oorspronkelijke image wordt daarom vermenigvuldigd met binaire matrix. Achtergrond in de $I_{RGB}$ wordt op NaN gezet. Deze resulterende matrix wordt geconverteerd naar het CIE La*b* kleurenmodel waarna chromatische a* en b* tegenelkaar worden afgezet. Vanuit het CIE La*b* kleurenmodel kan een Redness Index berekend worden voor iedere pixel welke een waarde heeft. Zie bijlage II Voor de gehanteerde algoritmes.

CIE LA*B* & RI

Van ieder blob wordt het oppervlakte berekend, deze wordt omgezet naar een geschatte korrel diameter. Deze korreldiameter wordt voorlopig uitgedrukt in pixels, in plaats van de gebruikelijk SI-eenheid meters.

PARTICLE SIZE DISTRIBUTION

Zoals omschreven in (Gonzalez & Woods, 2011) wordt de contour van iedere korrel omschreven als een n-aantal Fourier Descriptoren. De coördinaten van iedere pixel, welke behoort tot het contour wordt omgezet naar een complex nummer:

FOURIER DESCRIPTOREN

$$s(k) = x(k) + jy(k)$$

Hierbij is k een afzonderlijke pixel. De discrete Fourier transformatie van de 1-D reeks $s(k)$ kan als volgt geschreven worden:

$$a(u) = \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K}$$

*Figuur 9 De omtrek van een korrel weergegeven als een progressieve reeks Fourier Descriptoren*

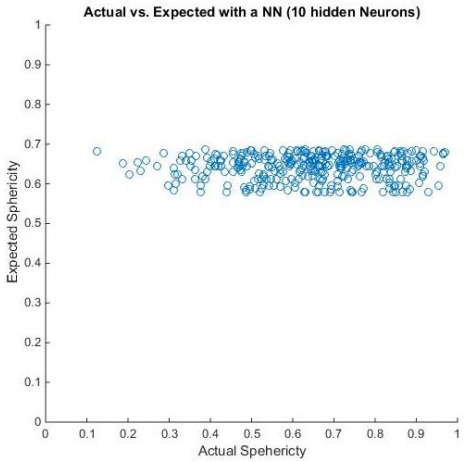
## 4.5 Classification

NEURAL NETWORKS

Classificatie wordt uitgevoerd door eerder verkregen Fourier Descriptoren als input te gebruiken in een Feedforward Neural Netwerk (NN). Er zal gebruikt gemaakt worden van twee NN's. Omdat deze NN's niet met complexe getallen kunnen rekenen zullen deze ingevoerd worden als een Phase en een Magnitude. Om de classificatie naar bolvormigheid te maken zal gebruikt worden gemaakt van de eerste zes descriptoren. De afronding en hoekigheid van de korrels wordt gebruikt door de eerste tien descriptoren te gebruiken.

SLECHTE PERFORMANCE

De huidige ontwikkeling van deze functie is gestagneerd. Het Neural Net is ingeleerd met 500 korrels, van iedere korrel is de sphericity bepaald. Het Neural Net constateert geen clausule verband.




*Figuur 10 Performance of a Neural Net*

# 5 De Matlab test opstelling

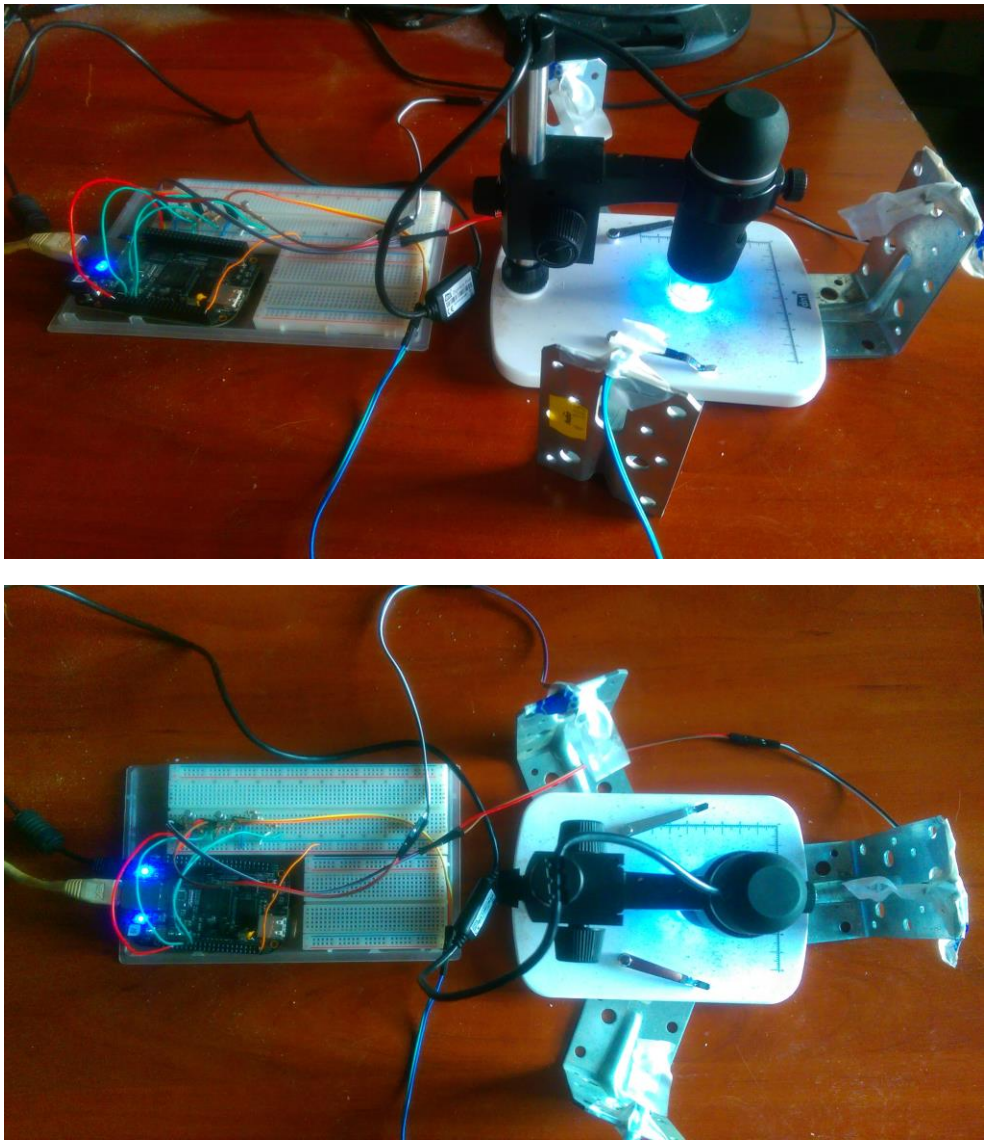
MATLAB CODE

Hierboven uitgewerkte stappen en procedures, ten samen met de flowchart (bijlage XIII) zijn voor uitgewerkt in Matlab code welke in bijlage V te vinden is. In deze bijlage is de GUI en helper functies omschreven. Verdere zijn er in bijlage VI Matlab functies uitgewerkt welke data handeling tijdens de ontwikkeling voor hun rekening nemen. Deze functies zijn voornamelijk ontwikkeld voor het verwerken van Neural Net datasets en Fourier Descriptoren.

Op dit moment 2-11-2014 is de ontwikkeling van de app nog bezig. De functies getoond bijlage V en VI zijn daarom nog aan verandering onderhevig. Voor de presentatie van op 6-11-2014 wordt de app afgerond.

Hieronder is de voorlopige test opstelling uitgezet.

TESTOPSTELLING HARDWARE



*Figuur 11 Test opstelling van de hardware*

# 6 Testen en resultaten

Omdat de Soil Analyzer app nog onder ontwikkeling is, is deze niet in zijn gehele hoedanigheid getest. Er is wel een benchmark uitgevoerd op individuele parts en functies. Deze test is in bijlage XII te vinden.

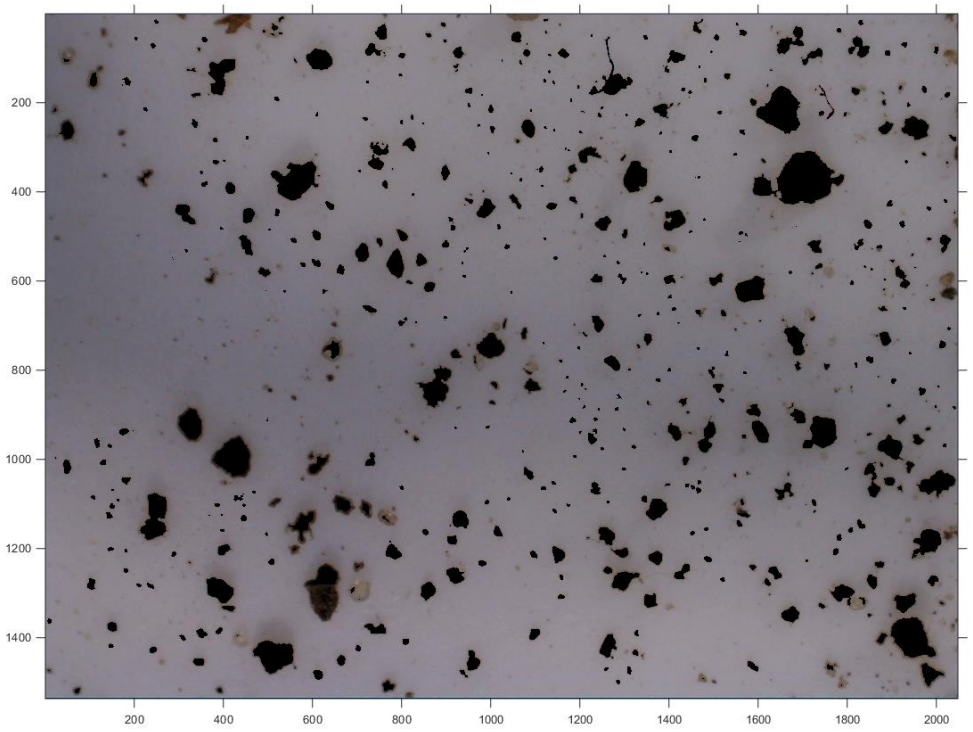
LANGZAAM OPSTARTEN

Wat opvalt uit de hier getoonde resultaten is dat het initiëren van de microscoop en het opzetten van communicatie met de Beaglebone Black relatief veel tijd kost. Ook is het binnen halen van de image over USB relatief veel tijd. Dit komt omdat de camera eerst tijd moeten krijgen om de automatische wit balans te regelen.

Overige analyserende functies zijn allemaal binnen $0.01[s]-2[s]$ uitgevoerd. Dit is inclusief de opbouw van de figuren, welke de uiteindelijke data aan de gebruiker tonen.

BLOBS BEPALEN KAN BETER

Grote uitschieters zijn conversie van een RGB matrix naar een binaire matrix welke alleen de blobs toont. Deze duurde $19[s]$. Ook komt het voor dat korrels niet altijd evengoed herkent worden. Zie het figuur hieronder. Hierin zijn de zwarte vlekken grondkorrels welke als dusdanig herkent zijn. Het figuur hieronder toont aan dat de algoritme last heeft met transparante korrels.




*Figuur 12 Niet herkende grondkorrels*

# 7 Conclusie en aanbevelingen

ACHTERGROND

Royal IHC heeft dhr. Spijker gevraagd om een prototype te ontwikkelen welke grondmonster, doormiddel van hun visuele aspecten, kan analyseren en kwantificeren. Hiervoor wordt eerst een testopstelling opgezet welke in Matlab op een personal computer uitgewerkt wordt. Deze product documentatie legt vast hoe deze testopstelling zal gaan functioneren, welke eisen er gesteld worden en hoe de interactie met zijn omgeving en gebruiker is.

De testopstelling zal geschreven worden als een Matlab app, deze software is uitermate geschikt voor imageprocessing en data analyses. Ook wordt er bij IHC en MTI veel gewerkt met dit programma. Dit zorgt voor een verduidelijkende werking van het uiteindelijke prototype voor de opdrachtgever.

De testopstelling bestaat uit Graphical User Interface en een 5MP USB microscoop, welke tot 300x kan vergroten. Een gebruiker plaats een monster onder deze microscoop en neemt een snapshot. Deze snapshot wordt met Matlab algoritmes geanalyseerd en de gebruiker krijgt informatie getoond over kleur, structuur en textuur van het grondmonster.

KLEUR

De kleuren informatie is visueel weergeven in de CIE La*b* kleurmodel en RI (Redness Index). Deze twee kleurmodellen zijn geschikt om uitspraken te kunnen doen over organische carbon en ijzer gehaltes van grond.

STRUCTUUR

Structuur van de grond wordt verkregen door de vorm van korrels te analyseren met behulp van Fourier Descriptoren en deze input in een Neural Netwerk te voeren welke de grondmonster classificeert in 6 gradaties rond-/hoekigheid en drie gradaties bolvormigeheid. Dit wordt met een histogram en een statistische omschrijving getoond aan de gebruiker zodat er een algemeen beeld van het gehele monster gevormd kan worden.

TEXTUUR

De textuur van de grond wordt statistisch gemeten door naar de afmetingen van de afzonderlijke korrels te kijken en dit in de vorm van een PSD (Particle Size Distribution) weer te geven.

WERKING

Werking van de testopstelling en voorlopige gebruikers protocollen zijn in dit document vastgelegd in een vorm van gebruikers- en beheerders-handleiding. Prestaties van het systeem en eventuele verbeterpunten voor het prototype kunnen gemeten worden met een voortgangstoets. Deze toetst in hoeverre gestelde eisen gehaald zijn.

CONCLUSIE

Deze product documentatie zet een basisontwerp neer waaruit een testopstelling van een Soil Analyzer in Matlab opgesteld kan worden. Een gedegen testopstelling borgt de kwaliteit van het ontwikkelingsproces voor

het prototype. Hierdoor worden ontwikkelingsrisico's verminderd en wordt de kwaliteit geborgd. Dit ontwerp en de realisatie hiervan is gebaseerd op vijf vision stappen: acquisition, enhancement, segmentation, feature extraction en classification.

AANBEVELINGEN

Voor het uitvoeren van bovengenoemde stappen zijn verschillende functies geschreven. Het testen van deze functies toont aan dat segmentation algoritme nog verbetering behoeft. Dit algoritme duur relatief lang, namelijk $19[s]$ en heeft moeite het herkennen van transparante grondkorrels. Voor het prototype zal dit algoritme geoptimaliseerd moeten worden. Hierbij rekening houden dat de kleur van transparante korrels niet mee genomen wordt bij de kleuren analyses.

Ook is het niet gelukt om voorspellende uitspraken over de vorm van een korrel te doen, met behulp van een Neural Netwerk. De Fourier Descriptoren lijken een korrelomtrek goed te ontleden. Hierin zijn de basisvormen te herkennen, maar het Neurale Net blijkt geen correlatie tussen vorm classificatie en Fourier Descriptor te herkennen. Voor het prototype wordt het dan ook aangeraden om classificatie via andere algoritmes uit te voeren, Hierbij kan nog wel gebruik gemaakt worden van de Fourier Descriptoren.

# 8 Literatuurlijst

Conrad. (2014). USB microscoop 20 tot 300x 5 Mpix. Retrieved September 28, 2014, from http://www.conrad.nl/nl/usb-microscoop-20-tot-300x-5-mpix-dnt-digimicro-profi-191393.html

Gonzalez, R. C., & Woods, R. E. (2011). *Digital Image Processing*. Pearson Education.

Miller, N. A., & Henderson, J. J. (2010). Quantifying Sand Particle Shape Complexity using a Dynamic, Digital Imaging Technique. *Agronomy Journal*, *102*(5), 1407. doi:10.2134/agronj2010.0097

Spijker, J. (2014). *Literatuuronderzoek: Optische kenmerken van grond, gebruikt bij computer vision*. Utrecht: Hogeschool van Arnhem en Nijmegen.

# Bijlage I. Plan van Aanpak

# Bijlage II. Literatuuronderzoek: Optische kenmerken van grond, gebruikt bij computer vision

# Bijlage III. Onderzoeksrapport: Effect van licht op kleurwaarneming

## Bijlage IV. Voortgangstest testopstelling

| ID | Omschrijving | Type | X |
|---|---|---|---|
| **F1** | **Kwantificeren van kleur** | | |
| F1.1 | Kleur bepalen van alle zichtbare grondkorrels<br>*Totaal aantal pixel, welke als grondkorrel gedefinieerd zijn, vergelijken met totaal van de RI histogram en de Scatterplot CIE La*b* 10x uitvoeren met 3 verschillende monsters.* | Vast | |
| F1.2 | Scatterplot van kleuren in CIE La*b* kleurenmodel alleen (a* en b*)<br>*Een uniforme kleurmonster met een gedefinieerde kleur meten en onderzoeken of de plot aan de verwachtingen voldoet* | Vast | |
| F1.3 | Histogram van kleuren in RI (Redness Index)<br>*Een uniforme kleurmonster met een gedefinieerde kleur meten en onderzoeken of de plot aan de verwachtingen voldoet* | Vast | |
| **F2** | **Kwantificeren van textuur** | | |
| F2.1 | Korrel grootte analyseren in het bereik $4[\mu m] \le P_{size} \le 2[mm]$<br>*Bins van de PSD histogram in kaart brengen, terwijl er bekende monsters getoetst zijn. 10x uitvoeren met 3 verschillende monsters.* | Vast | |
| F2.2 | Alleen "volledig" zichtbare korrels meten<br>*Visuele inspectie van de gemeten blobs ten opzichte van het oorspronkelijke. 10x uitvoeren met 3 verschillende monsters.* | Vast | |
| F2.3 | Betrouwbaarheidsinterval van 90% hanteren<br>*Is bij alle statistische berekeningen gewerkt met een* $z = 1.65$ | Vast | |
| F2.4 | Het resultaat moet binnen 10% nauwkeurigheid vallen<br>*Is bij alle statistische berekeningen gewerkt met een* $\alpha = 0.1$ | Vast | |
| F2.5 | PSD in volume metrische bins categoriseren<br>*Uitgevoerd?* | Vast | |
| **F3** | **Kwantificeren van structuur** | | |
| F3.1 | Korrel grootte analyseren in het bereik van $63[\mu m] \le P_{size} \le 2[mm]$<br>*Oppervlakte van de geanalyseerde korrels in kaart brengen. 10x uitvoeren met 3 verschillende monsters* | Vast | |
| F3.2 | Onderscheid maken tussen rond- en hoekigheid in 6 gradaties<br>*Is het classificatie systeem getest met 6 gradaties?* | Vast | |
| F3.3 | Onderscheid maken tussen bolvorming in 3 gradaties<br>*Is het classificatie systeem getest met 3 gradaties?* | Vast | |
| F3.4 | Alleen "volledig" zichtbare korrels analyseren<br>*Optische inspectie van de geanalyseerde blobs* | Vast | |
| F3.5 | Voorspelde t.o.v. werkelijk waarde $R \ge 0.9$ voor enkele korrel (lineair regressie)<br>*Lineair regressie analyse uitvoeren op de voorspelde waarde en de werkelijke waarde van de test data. Hierbij moet* $R \ge 0.9$ *zijn.* | Vast | |

| | | | |
|---|---|---|---|
| F3.6 | Betrouwbaarheidsinterval voor volledige monster van 90%<br>*Is bij alle statistische berekeningen gewerkt met een* $z = 1.65$ | Vast | |
| F3.7 | Het resultaat moet binnen 10% nauwkeurigheid vallen<br>*Is bij alle statistische berekeningen gewerkt met een* $\alpha = 0.1$ | Vast | |
| **F4** | **User interface Software** | | |
| F4.1 | Alle data getoond op 1 scherm (1366 x 768)<br>*Visuele inspectie.* | Vast | |
| F4.2 | Scatterplot van CIE LAB (in a*b* ruimte tonen)<br>*Visuele inspectie.* | Vast | |
| F4.3 | Histogram van RI (Alleen bins tonen)<br>*Visuele inspectie.* | Vast | |
| F4.4 | Histogram van PSD (Bins tonen aangevuld met textuele omschrijving van median, mean, standard deviation, skewness, kurtosis)<br>*Visuele inspectie.* | Vast | |
| F4.5 | RGB foto tonen van grond monster<br>*Visuele inspectie.* | Vast | |
| F4.6 | Preview modus om camera scherp te stellen<br>*Visuele inspectie.* | Vast | |
| **F5** | **User interface hardware** | | |
| F5.1 | Gebruiker plaats zelf monster onder camera<br>*Visuele inspectie.* | Vast | |
| F5.2 | Gebruiker stelt zelf scherp<br>*Visuele inspectie.* | Vast | |
| F5.3 | Gebruiker kan verlichting regelen<br>*Visuele inspectie.* | Vast | |
| T1 | Software omgeving | | |
| T1.1 | Programma runt in Matlab 2014a<br>*Visuele inspectie.* | Vast | |
| T1.2 | GUI is dockable<br>*Visuele inspectie.* | Vast | |
| T1.3 | Code is vrij gegeven on MIT licentie<br>*Visuele inspectie.* | Vast | |
| T1.4 | Code is in Engels gedocumenteerd<br>*Visuele inspectie.* | Vast | |
| T2 | Hardware omgeving | | |
| | geen | | |

# Bijlage V. Source code – Matlab app

```
function varargout = SoilAnalyzer(varargin)
% SOILANALYZER MATLAB code for SoilAnalyzer.fig
%      SOILANALYZER, by itself, creates a new SOILANALYZER or raises the existing
%      singleton*.
%
%      H = SOILANALYZER returns the handle to a new SOILANALYZER or the handle to
%      the existing singleton*.
%
%      SOILANALYZER('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SOILANALYZER.M with the given input arguments.
%
%      SOILANALYZER('Property','Value',...) creates a new SOILANALYZER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before SoilAnalyzer_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to SoilAnalyzer_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SoilAnalyzer

% Last Modified by GUIDE v2.5 15-Oct-2014 21:36:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SoilAnalyzer_OpeningFcn, ...
                   'gui_OutputFcn',  @SoilAnalyzer_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

% --- Executes just before SoilAnalyzer is made visible.
function SoilAnalyzer_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to SoilAnalyzer (see VARARGIN)

% Choose default command line output for SoilAnalyzer
handles.output = hObject;

% Init Beaglebone connection
global bbb;
bbb = beaglebone_black('192.168.178.20', 'root', 'open');
system(bbb, './pwm_vision_matlab');

% Init the CAM
border = 0;
global cam src;
cam = videoinput('winvideo', 1, 'MJPG_2048x1536');
cam.FramesPerTrigger = 1;
hImage = image(zeros(1536, 2048, 3), 'Parent', handles.axes_Image);
preview(cam, hImage);
cam.ROIPosition = [ border border 2048-border 1536-border ];
%cam.PowerLineFrequency = 50;



% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SoilAnalyzer wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = SoilAnalyzer_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbuttonSnap.
function pushbuttonSnap_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonSnap (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global bbb cam I L Dis labI;
system(bbb, 'echo 1 > /sys/class/gpio/gpio51/value && echo 1 > /sys/class/gpio/gpio50/value &&
echo 1 > /sys/class/gpio/gpio60/value');
start(cam)
I = getdata(cam);
system(bbb, 'echo 0 > /sys/class/gpio/gpio51/value && echo 0 > /sys/class/gpio/gpio50/value &&
echo 0 > /sys/class/gpio/gpio60/value');
L = GetLabeledImage(I);
[Dis labI] = CIELabSoilColorDistribution(L, I);

axes(handles.axes_Image);
aImage = I;
image(aImage);

% Update handles structure
```

```
guidata(hObject, handles);

% --- Executes on button press in pushbuttonAS.
function pushbuttonAS_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonAS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global L dis I P;
% Get CIElab scatterplot
L = GetLabeledImage(I);
dis = CIELabSoilColorDistribution(L,I);
axes(handles.axes_CIE);
ma = mean(dis(:,1));
mb = mean(dis(:,2));
handels.axes_CIE = scatter(dis(:,1),dis(:,2),'.');
hold on
plot([ma, ma],ylim,'r--','LineWidth',1);
plot(xlim, [mb, mb],'r--','LineWidth',1);
hold off

% Get RI Histogram
RI = RISoilColorDistribution(L, I);
axes(handles.axes_RI);
mri = mean(RI);
handels.axes_RI = hist(RI);
hold on
plot([mri, mri],ylim,'r--','LineWidth',1);
hold off

% Get PSD CDF plot
P = PSD(L);
axes(handles.axes_PSD);
handels.axes_PSD = cdfplot(P);

% Get the textuur histogram


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

*Published with MATLAB® R2014b*

```
function [ L n ] = GetLabeledImage( I )
%GETLABELEDIMAGE ( I) Convert the color image to a black and white logical image
%
%   Copyright (c) 2014 Jelle Spijker

hcsc = vision.ColorSpaceConverter('Conversion', 'RGB to intensity');
Int = step(hcsc, I);
[m, n] = size(Int);

L = zeros(m, n);
H = fspecial('gaussian',[5 5],0.6);

% calculate the treshold and convert to BW
Tr = uint8(mean(Int(:)) - (1/2)*std(double(Int(:))));
BW = zeros(size(Int));
BW(Int < Tr) = 1;

% Close the image (remove small holes and openings)
SE = strel('disk', 20,0);
BWROI = imclose(BW, SE);

% Determine Tresshold more closely for each region and build new picture
hblob = vision.BlobAnalysis;
hblob.AreaOutputPort = true;
hblob.BoundingBoxOutputPort = true;
hblob.MaximumCount = 1000;

[Area, Centroid, Blobs] = step(hblob, logical(BWROI));
[noblobs, ~] = size(Blobs);

for i = 1:noblobs
    % Extract ROI + border from source intensity picture an determine new thresshold
    boxROI = [Blobs(i,2) Blobs(i,2)+Blobs(i,4)-1 Blobs(i,1) Blobs(i,1)+Blobs(i,3)-1];
    expansionBorder = 10;
    if boxROI(1) <= expansionBorder
        boxROI(1) = 1;
    else
        boxROI(1) = boxROI(1) - expansionBorder;
    end
    if boxROI(2) > (m - expansionBorder)
        boxROI(2) = m;
    else
        boxROI(2) = boxROI(2) + expansionBorder;
    end
    if boxROI(3) <= expansionBorder
        boxROI(3) = 1;
    else
        boxROI(3) = boxROI(3) - expansionBorder;
    end
    if boxROI(4) >= (n - expansionBorder)
        boxROI(4) = n;
    else
        boxROI(4) = boxROI(4) + expansionBorder;
    end
    ROI = Int(boxROI(1):boxROI(2),boxROI(3):boxROI(4));

    ROI = imfilter(ROI,H,'symetric');
```

```
    level = graythresh(ROI) + ((1/3)*std(double(ROI(:))))/255;
    BWROI = im2bw(ROI,level);
    BWROI = ~BWROI;

    % Erode the new region of interrest with a disk of 2.5%
    total = sqrt(double(bwarea(BWROI)));
    se = strel('disk',round(total*0.025));
    erodedI = imerode(BWROI,se);

    BWROI = imfill(erodedI, 'holes');
    BWROI = RemoveBorderBlobs(BWROI);
    L(boxROI(1):boxROI(2),boxROI(3):boxROI(4)) = BWROI;
end

L = logical(L);

end
```

*Published with MATLAB® R2014b*

```
function [ L ] = RemoveBorderBlobs( BW )
%RemoveBorderBlobs Remove blobs on the border of an image
%
%   Copyright (c) 2014 Jelle Spijker

[r, c] = size(BW);
BW(1, :) = 1;
BW(:, 1) = 1;
BW(r, :) = 1;
BW(:, c) = 1;
L = bwlabel(BW,8);
L(L == L(1,1)) = 0;
L(L > 0) = 1;

L = logical(L);

end
```

*Published with MATLAB® R2014b*

```
function [ Ic ] = RemoveBackgroundVectorImage( BW, I )
%RemoveBackgroundVectorImage  Remove the background from the color image
%   BW is the logical image matrix, I is the three vector matrix
%
%   Copyright (c) 2014 Jelle Spijker

Ic = zeros(size(I));
[~, ~, n] = size(I);
for i = 1:n
    Ic(:,:,i) = I(:,:,i).*uint8(BW);
end
Ic = uint8(Ic);

end
```

*Published with MATLAB® R2014b*

```
function [ Dis, LAB] = CIELabSoilColorDistribution( L, I )
%CIELabSoilColorDistribution Summary of this function goes here

% Remove the background from the color image
[m, n, ~] = size(I);
BW = logical(L);

Ic = zeros(m, n, 3);
Ic(:,:,1) = I(:,:,1).*uint8(BW);
Ic(:,:,2) = I(:,:,2).*uint8(BW);
Ic(:,:,3) = I(:,:,3).*uint8(BW);
Ic = uint8(Ic);

%Convert the RGB matrix to CIE La*b* colorspace
hcscLAB = vision.ColorSpaceConverter;
hcscLAB.Conversion = 'sRGB to L*a*b*';
LAB = step(hcscLAB, double(Ic));

%Extract seperate values
Lu = LAB(:,:,1);
a = LAB(:,:,2);
b = LAB(:,:,3);

% Returns the distribution
[totSample, ~] = size(a(Lu ~= 0));
Dis = zeros(totSample, 2);
Dis(:, 1) = a(Lu ~= 0);
Dis(:, 2) = b(Lu ~= 0);
end
```

*Published with MATLAB® R2014b*

```
function [ P ] = PSD( L )
%PSD Summary of this function goes here

%Determine the Area for the blobs containing soil samples
hblob = vision.BlobAnalysis;
Area = step(hblob, logical(L));

blobs = size(Area);
P = zeros(sum(Area(:,1)),1);
j = 1;
% Loop through the blobs
for i = 1:blobs
    if i == 1
        P(1:Area(i)) = Area(i);
        j = j + Area(i);
    else
        P(j:(j + Area(i))) = Area(i);
        j = j + Area(i);
    end
end

P = Area2Volume(double(P));
end
```

*Published with MATLAB® R2014b*

```
function [ Ic ] = RemoveBackgroundVectorImage( BW, I )
%RemoveBackgroundVectorImage  Remove the background from the color image
%   BW is the logical image matrix, I is the three vector matrix

Ic = zeros(size(I));
[~, ~, n] = size(I);
for i = 1:n
    Ic(:,:,i) = I(:,:,i).*uint8(BW);
end
Ic = uint8(Ic);

end
```

*Published with MATLAB® R2014b*

```
function [ RI ] = RISoilColorDistribution( L, I )
%UNTITLED Summary of this function goes here

% Remove the background from the color image
[m n ~] = size(I);
BW = logical(L);

Ic(m, n, 3)=0;
Ic(:,:,1) = I(:,:,1).*uint8(BW);
Ic(:,:,2) = I(:,:,2).*uint8(BW);
Ic(:,:,3) = I(:,:,3).*uint8(BW);
Ic = uint8(Ic);

%Convert the RGB matrix to CIE La*b* colorspace
hcscLAB = vision.ColorSpaceConverter;
hcscLAB.Conversion = 'sRGB to L*a*b*';
LAB = step(hcscLAB, double(Ic));

%Extract seperate values
Lu = LAB(:,:,1);
a = LAB(:,:,2);
b = LAB(:,:,3);

RI(m, n) = 0;
for i = 1:m
     for j = 1:n
         RI(i,j) = (((Lu(i,j)*(a(i,j)^2 + b(i,j)^2)^(1/2))*10^(8.2)))/((b(i,j)*L(i,j)^6));
     end
end

RI = RI(L ~= 0);
end
```

*Published with MATLAB® R2014b*

```
function [ V ] = Area2Volume( A )
%Calculate the theoretical volume of a given Area
%   See Literatuur studie
    V = power(A,(3/2))/(6*sqrt(pi));
end
```

*Published with MATLAB® R2014b*

```
function [ z ] = frdescp( particle )
%FRDESCP Compute Fourier descriptors
%   Source: Gonzalez. Digital Image Processing Using MATLAB. McGraw-Hill Education (India) Pvt
Limited, 2013.

b = bwboundaries(particle, 'noholes');
s = b{1};

[np, nc] = size(s);
if nc ~= 2
    error('S must be of size np-by-2.');
end
if np/2 ~= round(np/2);
    s(end + 1, :) = s(end, :);
    np = np + 1;
end

% Create an alternating sequence of 1s and -1s for use in centering the
% transform.
x = 0:(np -1);
m = ((-1) .^x)';

% Multiply the input sequence by alternating 1s and -1s to center the
% transform
s(:, 1) = m .* s(:, 1);
s(:, 2) = m .* s(:, 2);

% Convert coordinates to complex numbers.
s = s(:, 1) + i*s(:,2);

% Compute the descriptors
z = fft(s);

% Opschalen????
end
```

*Published with MATLAB® R2014b*

```
function [ s ] = ifrdescp( z, nd )
%IFRDESCP Computes inverse Fourier descriptors
%   Source: Gonzalez. Digital Image Processing Using MATLAB. McGraw-Hill Education (India) Pvt
Limited, 2013.
% Some changes made by Jelle Spijker

% Preliminaries.
np = length(z);

% Check inputs.
if nargin == 1
    nd = np;
end
if mod(np,2) ~= 0
    error('length(z) must be an even integer.')
```

```
elseif mod(nd,2) ~= 0
    error('nd must be an even integer.')
end
% Create an alternating sequence of 1s and -1s for us in centering the
% transform
x = 0:(np - 1);
m = ((-1) .^ x)';

%use only nd descriptors in the inverse. Because the descriptors are
%centered, (np - nd)/2 terms from each end of the sequence are set to 0.

d = (np - nd)/2;
z(1:d) = 0;
z(np - d + 1:np) = 0;

% Compute the inverse and convert back to coordinates.
zz = ifft(z);
s(:,1) = real(zz);
s(:,2) = imag(zz);

% Multiply by alternating 1 and -s to undo the earlier centering.
s(:,1) = round(m .* s(:,1));
s(:,2) = round(m .* s(:,2));

s = uint16(s);
end
```

*Published with MATLAB® R2014b*

```
function [ image ] = bound2im( b, M, N )
% IMAGE = BOUND2IM(b) converts b, a np-by-2 array containing the integer
% coordinates of a boundary, into a binary image with 1s in the locations of
% the cooordinates in b an 0's elsewhere.
% Source: Gonzalez. Digital Image Processing Using MATLAB. McGraw-Hill Education (India) Pvt
Limited, 2013.
% Some alterations by Jelle Spijker

% Check input.
if size(b, 2) ~= 2
    error('The boundary must be of size np-by-2');
end

% Make sure the coordinates are integers.
if isinteger(b) ~= 1
    error('Matrix should consist of Integers');
elseif nargin == 1
    Mmin = min(b(:,1)) - 1;
    Nmin = min(b(:,2)) - 1;
    H = max(b(:,1)) - Mmin + 2;
    W = max(b(:,2)) - Nmin + 2;
    M = H + Mmin;
    N = W + Nmin;
end

% Make sure there aren't any 0 indexes
if size(b(b == 0)) > [0 0]
```

```
    b = b + 1;
end

% Create the image.
image = zeros(M, N);
linearIndex = sub2ind([M N], b(:,1), b(:, 2));
image(linearIndex) = 1;

end
```

*Published with MATLAB® R2014b*

## Bijlage VI. Source code – supporting functions

```
% Soil samples GetSamples from CAM
cam = videoinput('winvideo', 1, 'MJPG_2048x1536');
cam.FramesPerTrigger = 1;
cam.ROIPosition = [ 0 0 2048 1536 ];
src = getselectedsource(cam);
src.BacklightCompensation = 'off';
src.Sharpness = 2;
src.Contrast = 40;
%bbb = beaglebone_black('192.168.178.20', 'root', 'open');
%system(bbb, './pwm_vision_matlab');
addpath 'D:/OneDrive/Opleiding/HTS HAN/Minor/MatlabSoilAnalyzer'

DB = struct;
n = 1;

for i = 1:1
    preview(cam);
    disp('Focus sample and press any key when ready');
    pause;
    closepreview;

    start(cam)
    I = getdata(cam);
    I = I(5:end-5,5:end-5,:);
    L = logical(GetLabeledImage(I));

    hcsc = vision.ColorSpaceConverter('Conversion', 'RGB to intensity');
    Int = step(hcsc, I);

    hblob = vision.BlobAnalysis;
    hblob.MaximumCount = 1000;
    [~, ~, BBox] = step(hblob, L);
    [noBlob, ~] = size(BBox);

    for j = 1:noBlob
        DB(n).Ic = I(BBox(j,2):BBox(j,2)+BBox(j,4),BBox(j,1):BBox(j,1)+BBox(j,3),:);
        DB(n).I = logical(L(BBox(j,2):BBox(j,2)+BBox(j,4),BBox(j,1):BBox(j,1)+BBox(j,3)));
        DB(n).z = frdescp(DB(n).I);
        DB(n).Sphericity = -1;
        DB(n).Angularity = -1;
        n = n + 1;
    end

end
% system(bbb, 'echo 0 > /sys/class/gpio/gpio51/value && echo 0 > /sys/class/gpio/gpio50/value
&& echo 0 > /sys/class/gpio/gpio60/value');

[~, noS] = size(DB);
n = 1;

for i = 1:noS
    [xs ys] = (size(DB(i).I));
```

```
    if xs <= 25 && ys <= 25
        elRem(n) = i;
        n = n + 1;
    end
end

DB(elRem) = [];

rmpath 'D:/OneDrive/Opleiding/HTS HAN/Minor/MatlabSoilAnalyzer'
```

*Published with MATLAB® R2014b*

```
% Determine Particle Shape
clear;
load('matlab2.mat');
[~, n] = size(DB);
warning('off', 'images:imshow:magnificationMustBeFitForDockedFigure')
for i = 1:n
    subplot(3,3,1);
    imshow(DB(i).Ic);
    for j = 1:8
        subplot(3,3,j+1);
        imshow(bound2im(uint16(ifrdescp(DB(i).z,2*j))));
    end
    DB(i).Sphericity = input('Enter Sphericity: ');
end
warning('on', 'images:imshow:magnificationMustBeFitForDockedFigure')
%save('matlab.mat', DB);
```

*Published with MATLAB® R2014b*

```
clear;
C = cell(83+115+142+106, 5);
T = zeros(83+115+142+106, 1);

counter = 1;
for i = 1:4
   load(sprintf('matlab%d.mat', i));
   [~, s] = size(DB);
   for j = 1:s
       C{counter, 1} = DB(j).z;
       C{counter, 2} = ifrdescp(C{counter,1},6);
       C{counter, 3} = bound2im(C{counter,2});
       if (size(C{counter,1}) >= [6 1])
            C{counter, 4} = C{counter,1}(1:6);
       end
       s = regionprops(C{counter, 3}, 'Eccentricity');
       C{counter, 5} = s.Eccentricity;
       T(counter) = s.Eccentricity;
       counter = counter + 1;
   end
end

C(193,:) = [];
```

```
I = zeros(445,12);
O = zeros(445,1);
for i = 1:445
    I(i,:) = [phase(C{i,4}); angle(C{i,4})];
    O(i) = C{i,5};
end
```

*Published with MATLAB® R2014b*

```
clear;
C = cell(83+115+142+106, 6);
T = zeros(83+115+142+106, 1);

counter = 1;
for i = 1:4
   load(sprintf('matlab%d.mat', i));
   [~, s] = size(DB);
   for j = 1:s
       C{counter, 1} = DB(j).z;
       C{counter, 2} = ifrdescp(C{counter,1},200);
       C{counter, 3} = bound2im(C{counter,2});
       s = regionprops(C{counter, 3}, 'ConvexHull','Perimeter');
       C{counter, 4} = s.ConvexHull;
       [sc, ~] = size(s.ConvexHull);
       difCH = zeros(size(s.ConvexHull));
       difCH(1,:) = s.ConvexHull(2,:) - s.ConvexHull(1,:);
       for k = 2:sc
           difCH(k,:) = s.ConvexHull(k - 1,:) - s.ConvexHull(k,:);
       end
       difCH = difCH.^2;
       d = difCH(:,1)+difCH(:,2);
       d = d.^(1/2);
       C{counter, 5} = sum(d);
       C{counter, 6} = s.Perimeter;
       C{counter, 7} = C{counter, 5}/s.Perimeter;
       ratio(counter) = C{counter, 7};
       counter = counter + 1;
   end
end

C(193,:) = [];

for i = 1:445

end
```

*Published with MATLAB® R2014b*

## Bijlage VII. Datasheet USB Microscoop



### Technische informatie

| | |
|---|---|
| Beeldsensor | 2 megapixel |
| Videoresolutie | 2592x1944, 2048x1536, 1600x1200, 1280x960 pixel |
| Resolutie afzonderlijk beeld | 2592x1944, 2048x1536, 1600x1200, 1280x960 pixel |
| Kleur | 24-bit-RGB |
| Optiek | duale lens 27-voudig + 100-voudig |
| Focussering | handmatig, 10 mm tot 300 mm |
| Flikkerfrequentie | 50/60 Hz |
| Beeldherhalingsfrequentie | max. 30 beelden/s |
| Vergroting | ca. 20 tot 200 keer, 300 keer |
| Sluittijd | 1 sec. tot 1/1000 sec. |
| Beeldformaat | JPG |
| Videoformaat | AVI |
| Witbalans | Automatisch |
| Verlichting | Automatisch |
| Werkveldverlichting | 8 LED's, wit, traploos aan het apparaat instelbaar |
| PC-interface | USB 2.0 |
| Bedrijfssysteem | Windows XP SP2/Vista, Windows 7, Windows8 and MAC OS 10.6 of hoger |
| Spanningsvoeding | 5 V (USP-poort) |
| Microscoop | 110 x 35 mm |
| Standvoet | 175 x 140 x 110 mm |



### Conformiteitsverklaring

Hiermee verklaart dnt GmbH dat het apparaat DigiMicro Profi in overeenstemming is met de fundamentele eisen en de overige desbetreffende bepalingen van de richtlijn 2004/108/EG. De CE-verklaring is verkrijgbaar onder www.dnt.de/conformiteit.

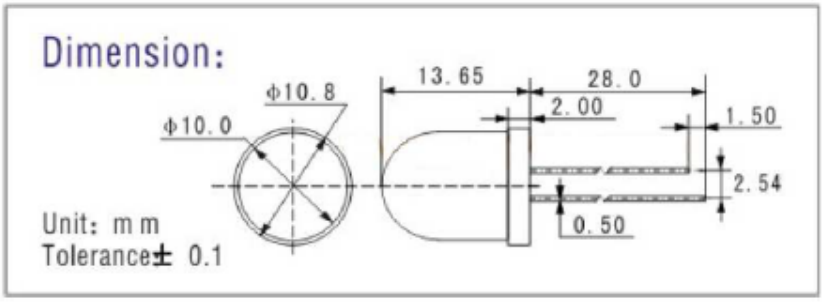
### Milieuvriendelijke afvoer

Dit apparaat is gemarkeerd overeenkomstig de Europese richtlijn 2002/96/EG betreffende afgedankte elektrische en elektronische apparatuur. Maak gebruik van de door uw gemeente ingerichte verzamelplaats voor het teruggeven en recycleren van afgedankte elektrische en elektronische apparaten.

## Bijlage VIII. Datasheet – Beaglebone Black – rev. C

| | BeagleBone Black $55 |
|---|---|
| Processor | AM3358BZCZ100, 1GHZ |
| Video Out | HDMI |
| DRAM | 512MB DDR3L 800MHZ |
| Flash | 4GB eMMC, uSD |
| Onboard JTAG | Optional |
| Serial | Header |
| PWR Exp Header | No |
| Power | 210-460 mA@5V |

| | Feature | |
|---|---|---|
| Processor | Sitara AM3358BZCZ100<br>1GHz, 2000 MIPS | |
| Graphics Engine | SGX530 3D, 20M Polygons/S | |
| SDRAM Memory | 512MB DDR3L 800MHZ | |
| Onboard Flash | 4GB, 8bit Embedded MMC | |
| PMIC | TPS65217C PMIC regulator and one additional LDO. | |
| Debug Support | Optional Onboard 20-pin CTI JTAG, Serial Header | |
| Power Source | miniUSB USB or DC Jack | 5VDC External Via Expansion Header |
| PCB | 3.4" x 2.1" | 6 layers |
| Indicators | 1-Power, 2-Ethernet, 4-User Controllable LEDs | |
| HS USB 2.0 Client Port | Access to USB0, Client mode via miniUSB | |
| HS USB 2.0 Host Port | Access to USB1, Type A Socket, 500mA LS/FS/HS | |
| Serial Port | UART0 access via 6 pin 3.3V TTL Header. Header is populated | |
| Ethernet | 10/100, RJ45 | |
| SD/MMC Connector | microSD , 3.3V | |
| User Input | Reset Button<br>Boot Button<br>Power Button | |
| Video Out | 16b HDMI, 1280x1024 (MAX)<br>1024x768,1280x720,1440x900 ,1920x1080@24Hz<br>w/EDID Support | |
| Audio | Via HDMI Interface, Stereo | |
| Expansion Connectors | Power 5V, 3.3V , VDD_ADC(1.8V)<br>3.3V I/O on all signals<br>McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 4 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID<br>(Up to 4 can be stacked) | |
| Weight | 1.4 oz (39.68 grams) | |
| Power | Refer to Section 6.1.7 | |




Notes:

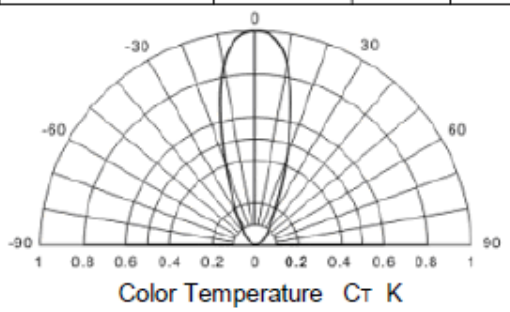
- 1000ucd = 1mcd, 1000mcd = 1cd
- All dimensions are in millimenter
- Clean only in isopropanol, ethanol, Freon TF (or equivalent)
- If forming is required, it must be done before soldering . Form pin leads by securing under 5mm from body and bedding wth radio pliers or the equivalent to avoid pressure on resin. When the LED is mounted into a P.C. board, pitch spacing should be aligned to prevent any stress to the resin. Any unsuitable stress applied to resin may break bonding wire in LED, which will cause failure.
- Protruded resin under flange is 1.5mm Max.
- Specifications are subjet to change without nice.

**Absolute Maximum Rating**

| Parameter | Maximum Rating | Unit |
|---|---|---|
| DC forward current | 30 | mA |
| Peak forward current<br>Pulse width Max. 10ms duty ratio Max 1/10 | 100 | mA |
| Reverse Voltage | 5 | V |
| Power dissipation | 100 | mW |
| Operating Temperature Range | - 40 °C to +85°C | |
| Storage Temperature Range | -40°C to +100°C | |
| Lead Soldering Temperature<br>[4mm From Body] | 260℃ for 5 seconds | |

**Electro-Optical Characteristics ( Ta=25℃ )**

| Parameter Radiant | Test Condition | Symbol | Min. | Typ | Max. | Unit |
|---|---|---|---|---|---|---|
| Forward Voltage | If=20mA | $V_{DD}$ | 2.8 | 3.6 | 4.0 | V |
| Color Temperature | If=20mA | $C_T$ | | | 10000 | K |
| Luminous Intensity | If=20mA | Iv | 80 | | 110 | cd |
| Reverse Current | Vr=5V | Ir | | | 100 | μA |
| Viewing Angle | If=20mA | 2θ1/2 | | 20 | | deg |



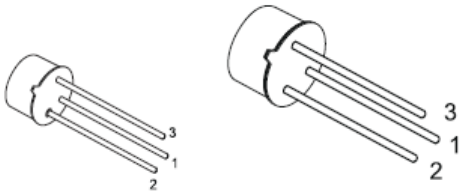
# 2N2219A
# 2N2222A

## HIGH SPEED SWITCHES

PRELIMINARY DATA

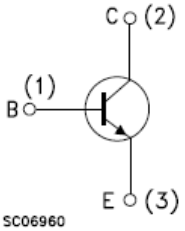
### DESCRIPTION

The 2N2219A and 2N2222A are silicon Planar Epitaxial NPN transistors in Jedec TO-39 (for 2N2219A) and in Jedec TO-18 (for 2N2222A) metal case. They are designed for high speed switching application at collector current up to 500mA, and feature useful current gain over a wide range of collector current, low leakage currents and low saturation voltage.




TO-18 TO-39

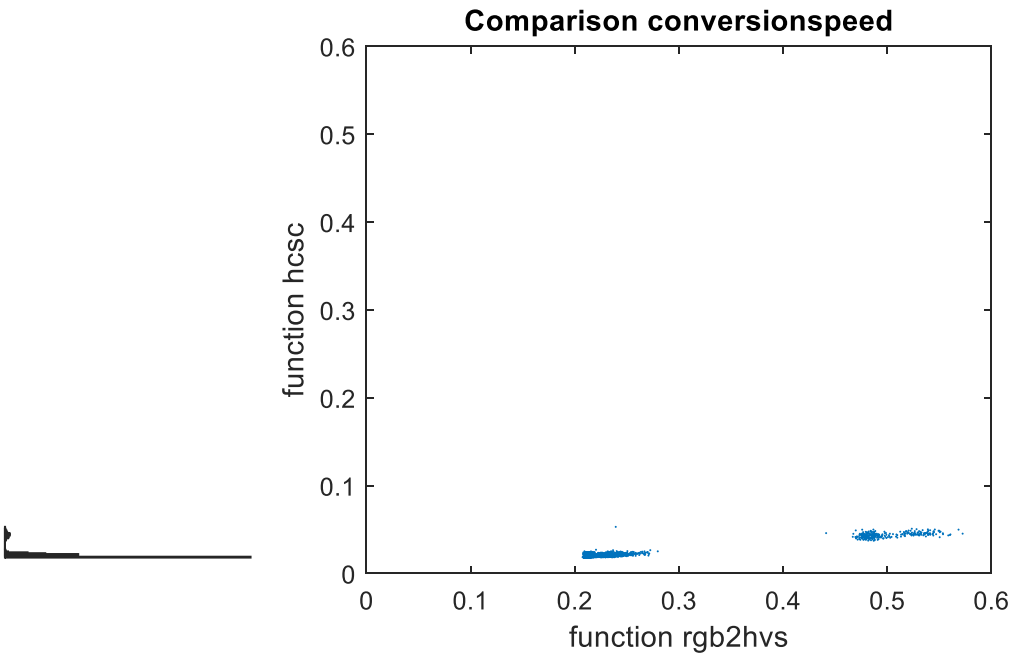
### INTERNAL SCHEMATIC DIAGRAM




### ABSOLUTE MAXIMUM RATINGS

| Symbol | Parameter | Value | Unit |
|---|---|---|---|
| $V_{CBO}$ | Collector-Base Voltage ($I_E$ = 0) | 75 | V |
| $V_{CEO}$ | Collector-Emitter Voltage ($I_B$ = 0) | 40 | V |
| $V_{EBO}$ | Emitter-Base Voltage ($I_C$ = 0) | 6 | V |
| $I_C$ | Collector Current | 0.6 | A |
| $I_{CM}$ | Collector Peak Current ($t_p$ < 5 ms) | 0.8 | A |
| $P_{tot}$ | Total Dissipation at $T_{amb} \leq 25\ ^{\circ}C$<br>for **2N2219A**<br>for **2N2222A**<br>at $T_C \leq 25\ ^{\circ}C$<br>for **2N2219A**<br>for **2N2222A** | <br>0.8<br>0.5<br><br>3<br>1.8 | <br>W<br>W<br><br>W<br>W |
| $T_{stg}$ | Storage Temperature | -65 to 175 | °C |
| $T_j$ | Max. Operating Junction Temperature | 175 | °C |

## Bijlage XI. Test resultaten verschillende Matlab conversies



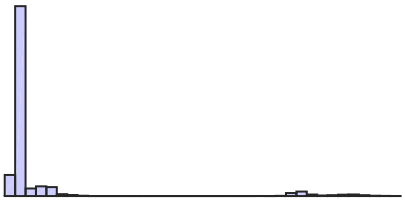



**hcsc**

mean: 0.0213 [s]
st.dev: 0.0682 [s]
range: 0.0347 [s]

**rgb2hvs**

mean: 0.2322 [s]
st.dev: 0.0682 [s]
range: 0.3648 [s]



# Bijlage XII. Performance test Matlab functies

### *INIT BEAGLEBONE CONNECTION*

```
tic;
clear;

bbb = beaglebone_black('192.168.178.20', 'root', 'open');
system(bbb, './pwm_vision_matlab');

formatSpec = 'Setting up de Beaglebone %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

```
Setting up de Beaglebone 4.106146e+00 [sec].
```

### *INIT THE CAM*

```
tic;

cam = videoinput('winvideo', 1, 'MJPG_2048x1536');
cam.FramesPerTrigger = 20;
border = 0;
cam.ROIPosition = [ border border 2048-border 1536-border ];

src = getselectedsource(cam);
src.BacklightCompensation = 'off';
src.Sharpness = 5;
src.Contrast = 40;

formatSpec = 'Setting up de cam %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

```
Setting up de cam 2.263421e+00 [sec].
```

### *GET IMAGES*

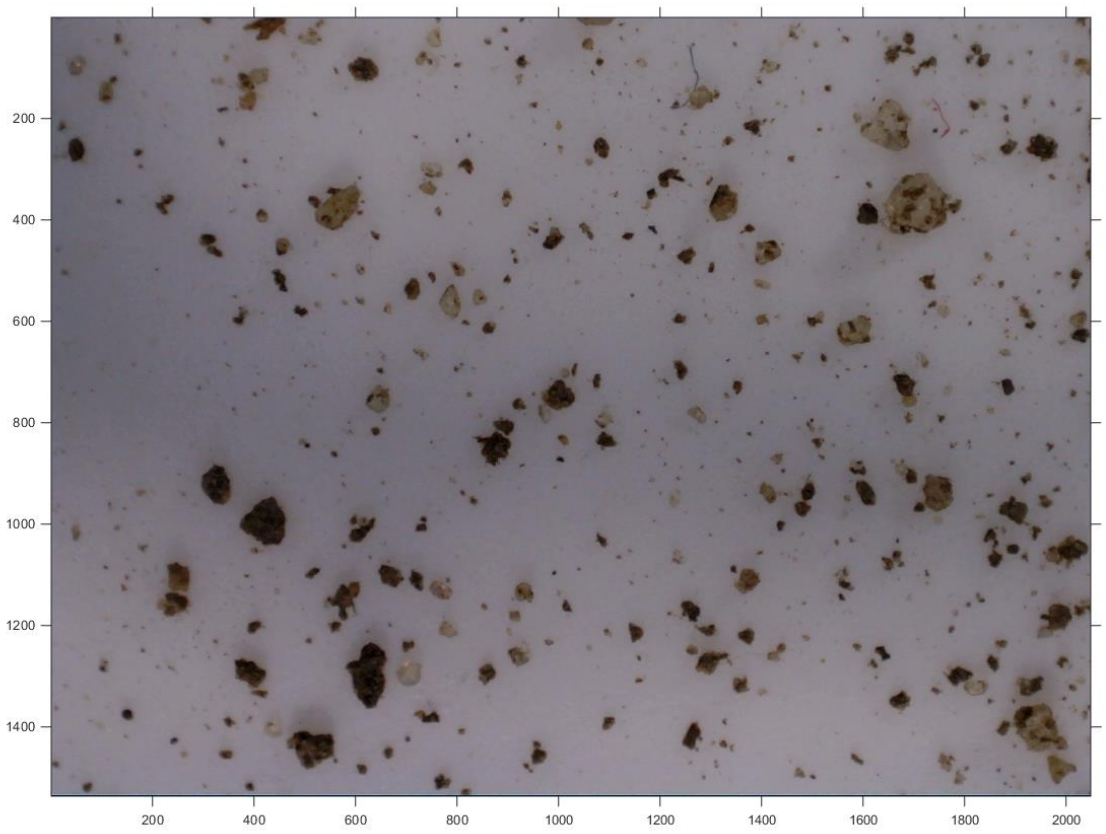
```
tic;

system(bbb, 'echo 1 > /sys/class/gpio/gpio51/value && echo 1 > /sys/class/gpio/gpio50/value &&
echo 1 > /sys/class/gpio/gpio60/value');
start(cam)
M = getdata(cam);
I = M(:,:,:,20);
system(bbb, 'echo 0 > /sys/class/gpio/gpio51/value && echo 0 > /sys/class/gpio/gpio50/value &&
echo 0 > /sys/class/gpio/gpio60/value');

imshow(I)

formatSpec = 'Getting a image %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

```
Warning: Image is too big to fit on screen; displaying at 50%
Getting a image 9.155018e+00 [sec].
```



### *CONVERT RGB TO INTENSITY*

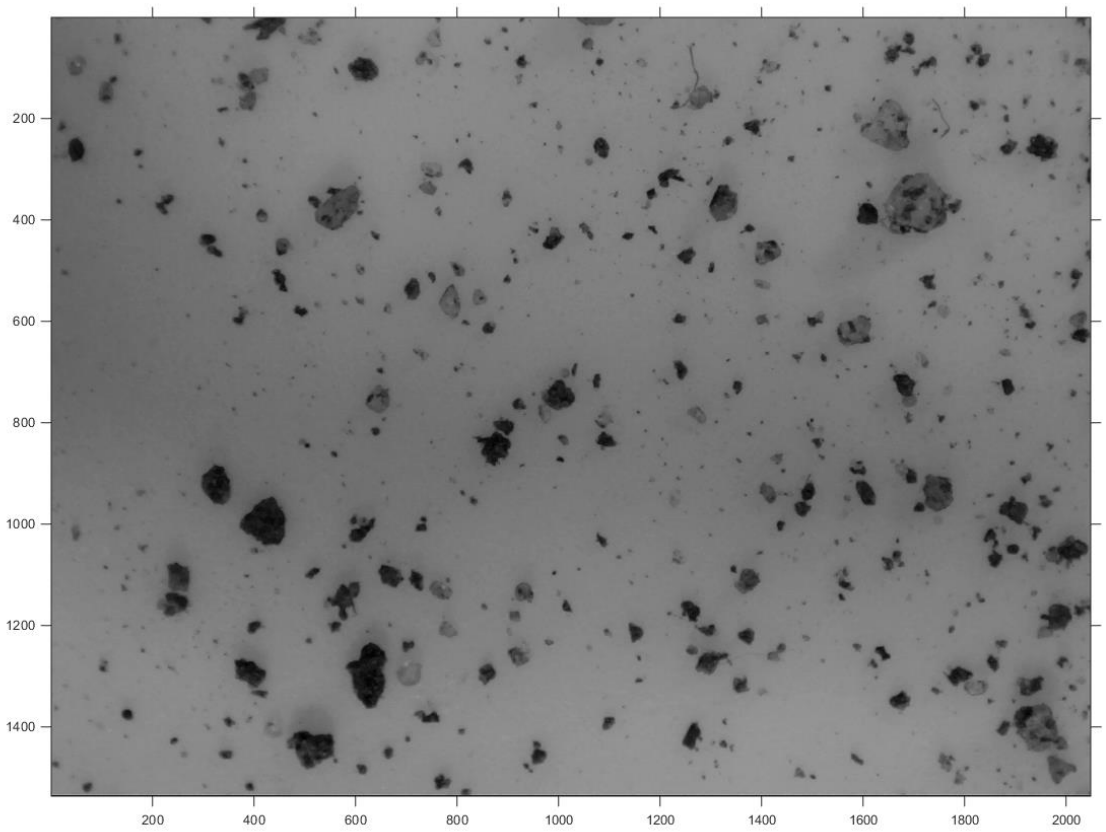
```
tic;

hcsc = vision.ColorSpaceConverter('Conversion', 'RGB to intensity');
Int = step(hcsc, I);

imshow(Int)

formatSpec = 'Convert RGB to Intensity %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

```
Warning: Image is too big to fit on screen; displaying at 50%
Convert RGB to Intensity 1.961357e-01 [sec].
```



### *CONVERT TO BW*

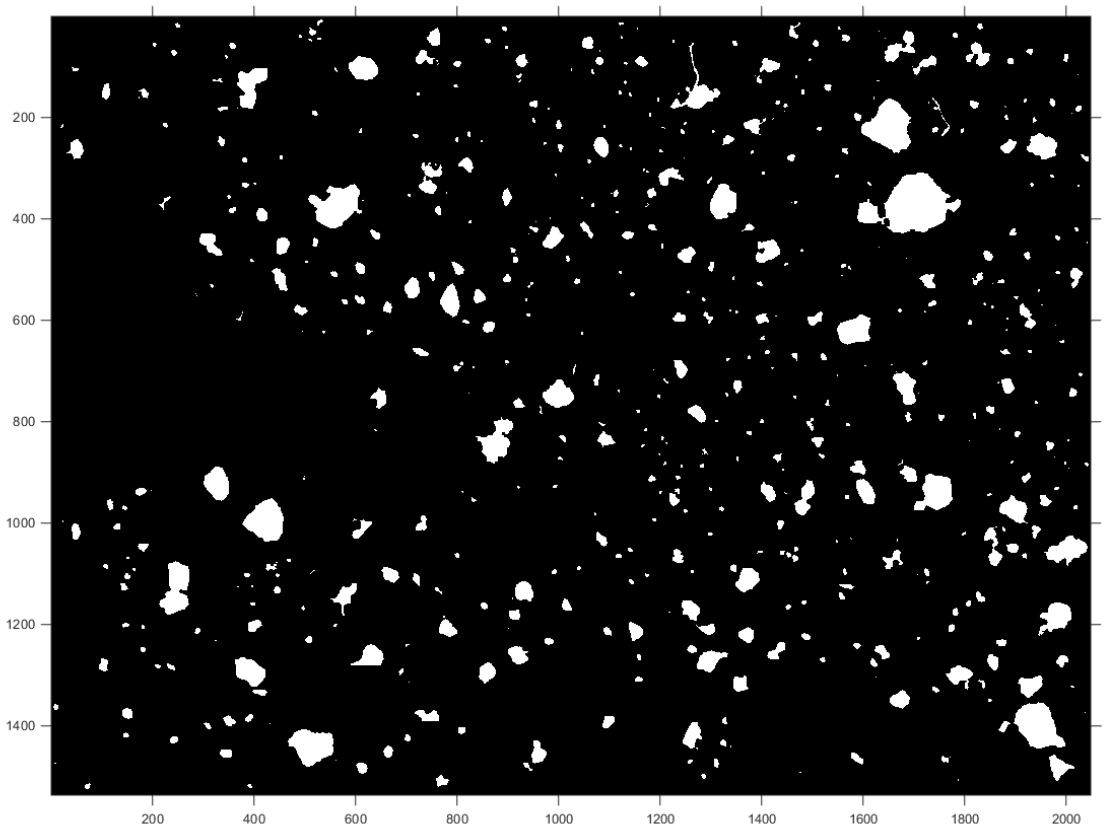
```
tic;

BW = GetLabeledImage(I);

imshow(BW)

formatSpec = 'Convert to BW %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

```
Warning: Image is too big to fit on screen; displaying at 50%
Convert to BW 1.968725e+01 [sec].
```



*REMOVE BORDER BLOBS*

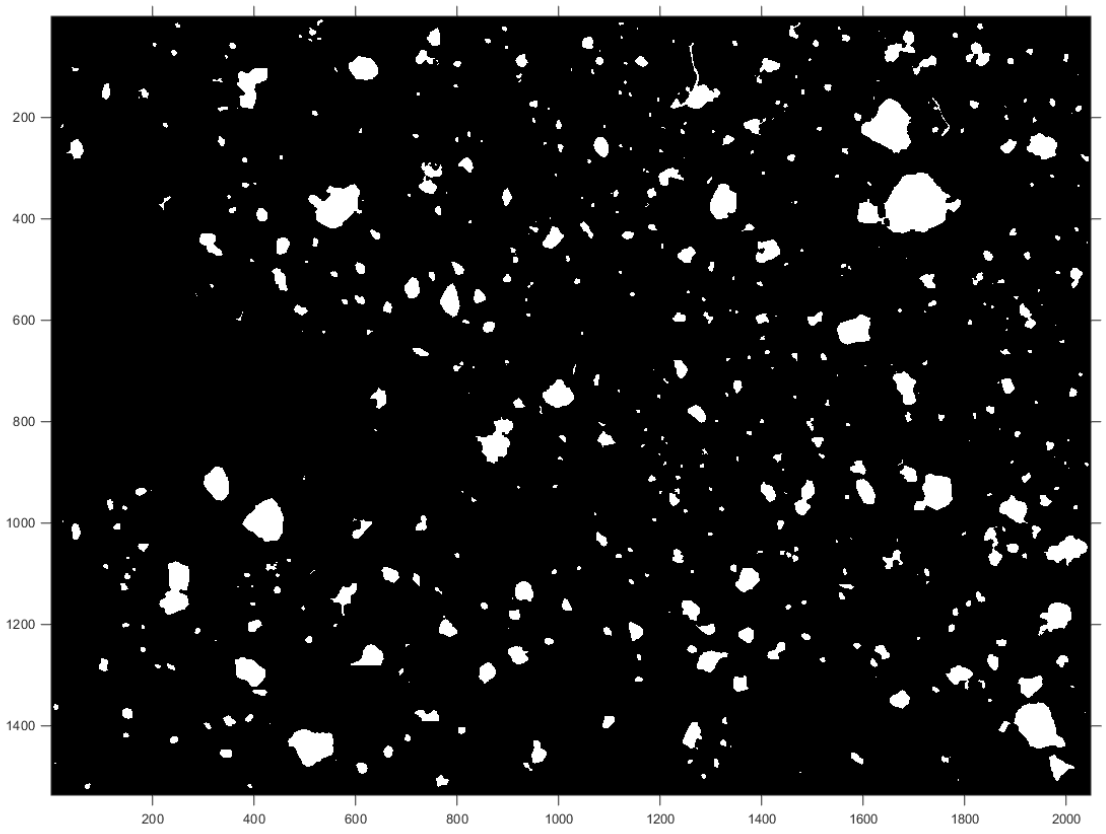
```
tic;

BWl = RemoveBorderBlobs(BW);

imshow(BWl)

formatSpec = 'Remove border blobs %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

```
Warning: Image is too big to fit on screen; displaying at 50%
Remove border blobs 1.425609e-01 [sec].
```



*CALULATE AREA*

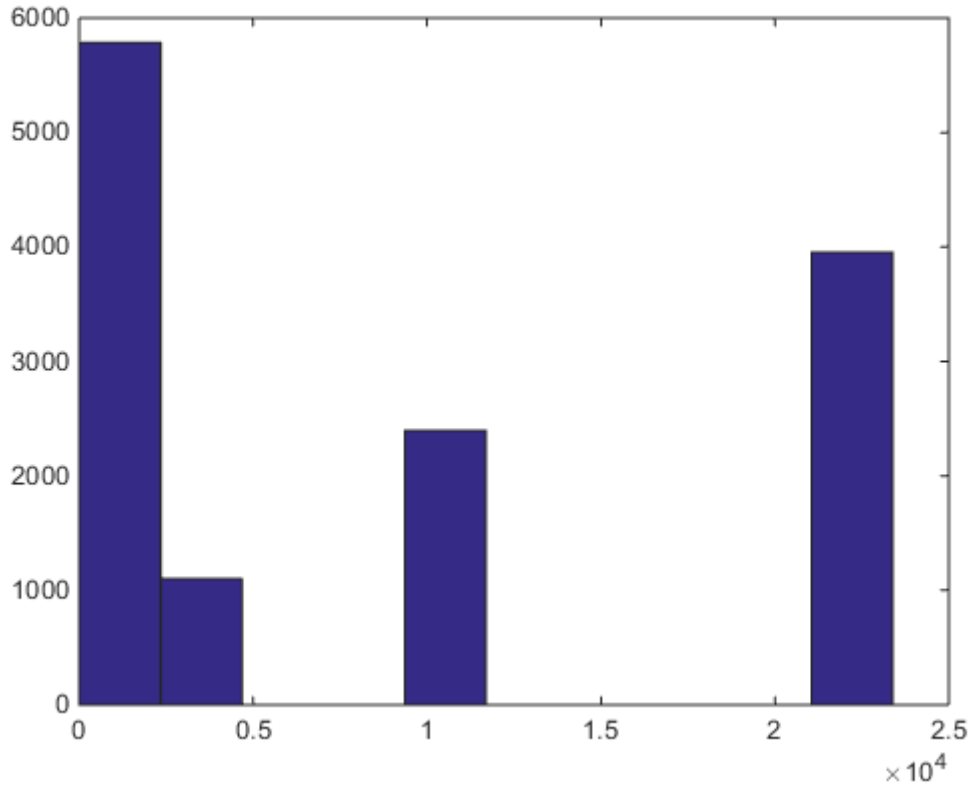
```
tic;

area = PSD(BWl);

figure; hist(area);

formatSpec = 'Calulate Area %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

```
Calulate Area 9.614259e-02 [sec].
```



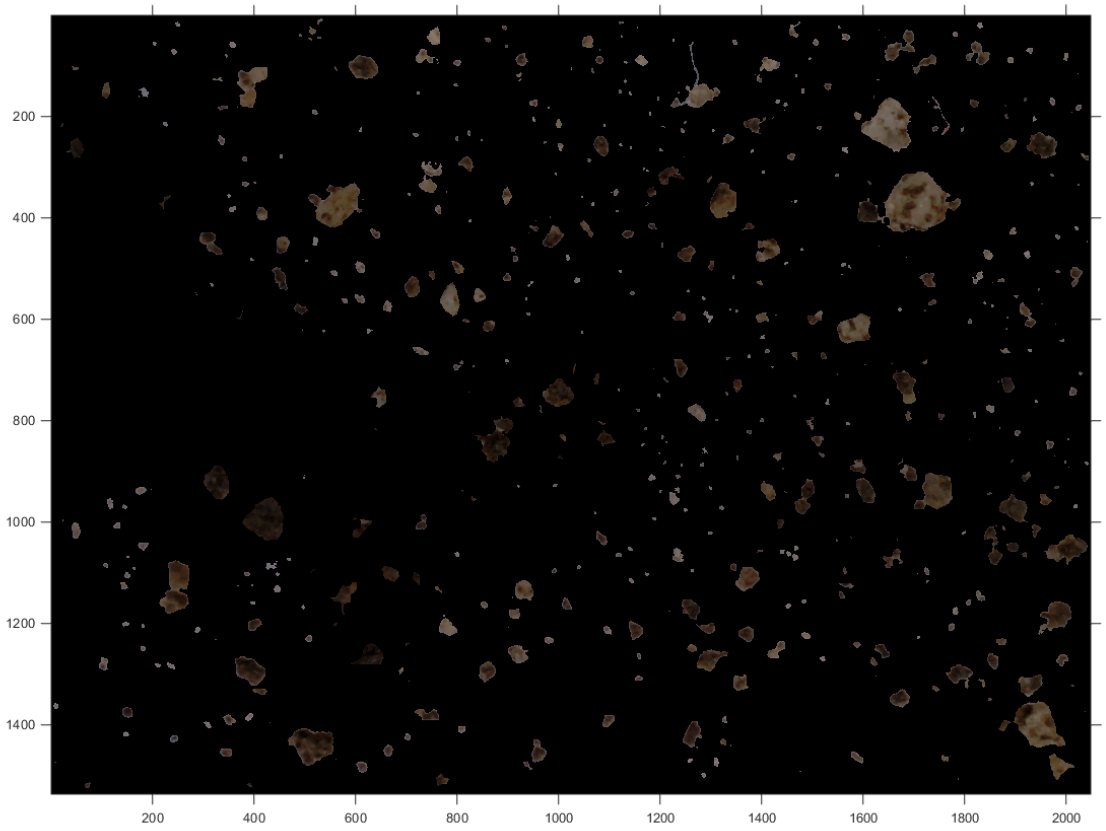
### *GET COLOR MATRIX OF FOREGROUND*

```
tic;

Ic = RemoveBackgroundVectorImage(BW, I);
imshow(Ic);

formatSpec = 'Get Color Matrix %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

```
Warning: Image is too big to fit on screen; displaying at 50%
Get Color Matrix 2.739707e-01 [sec].
```



### *CALC CIE LA*B**

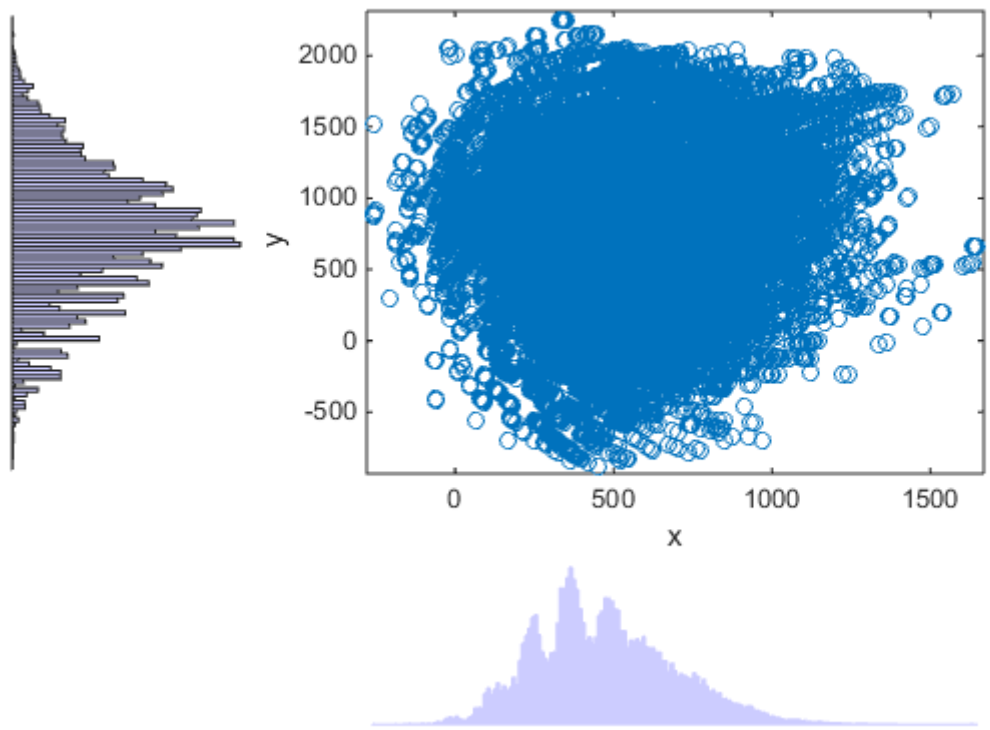
```
tic;

dis = CIELabSoilColorDistribution(BW, I);

figure; scatterhist(dis(:,1),dis(:,2));

formatSpec = 'Calc CIE La*b* alternative 2 %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

```
Calc CIE La*b* alternative 2 1.463346e+00 [sec].
```



*CALC RI*

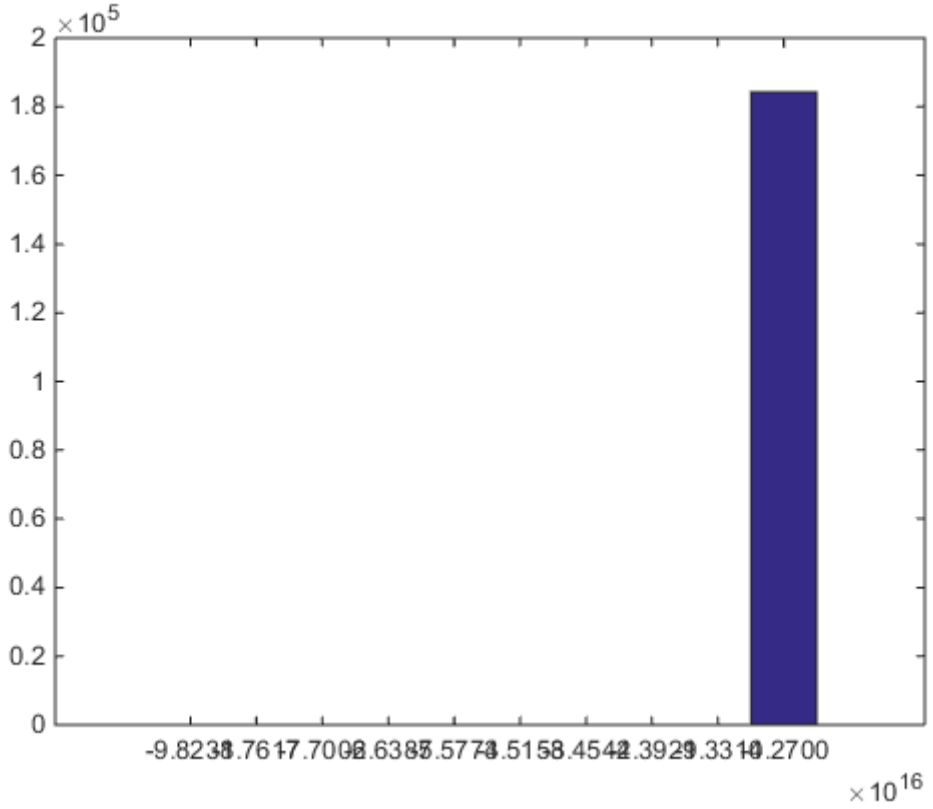
```
tic;

RI = RISoilColorDistribution(BW, I);

figure; hist(RI);

formatSpec = 'Calc RI %d [sec].';
str = sprintf(formatSpec,toc);
disp(str);
```

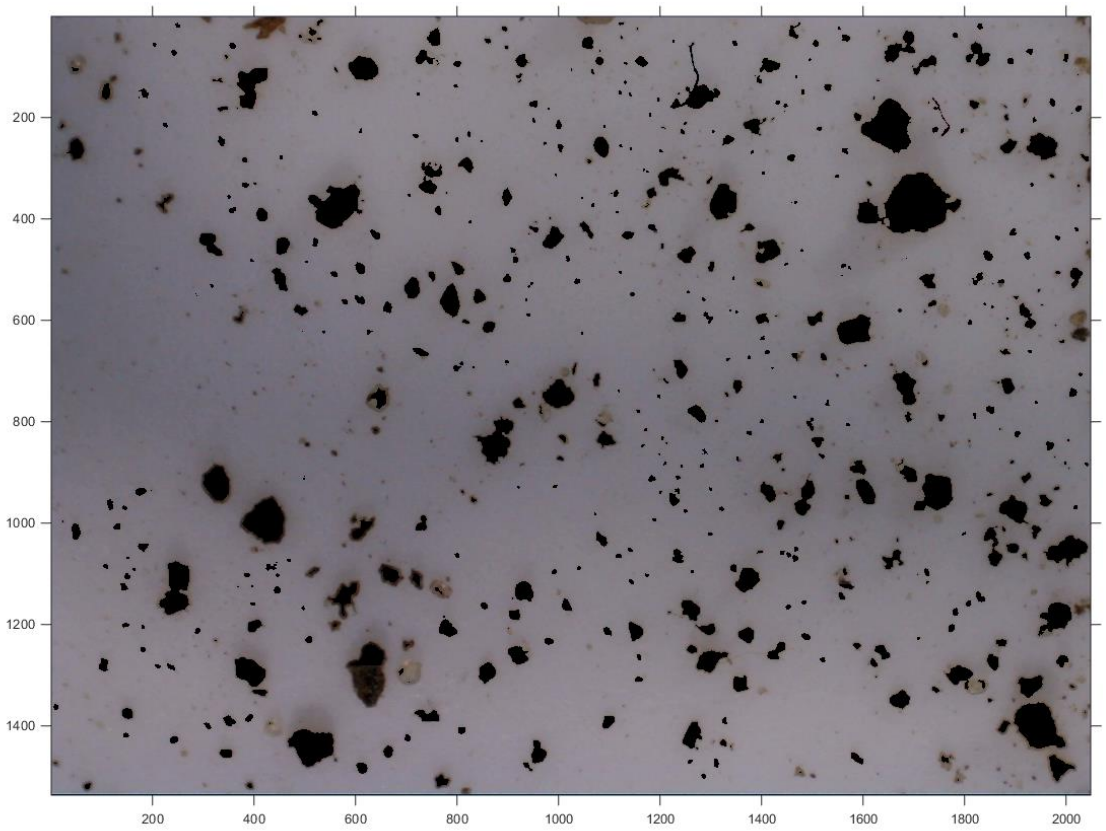
```
Calc RI 2.119925e+00 [sec].
```



### *LEFT BEHIND PARTICLES*

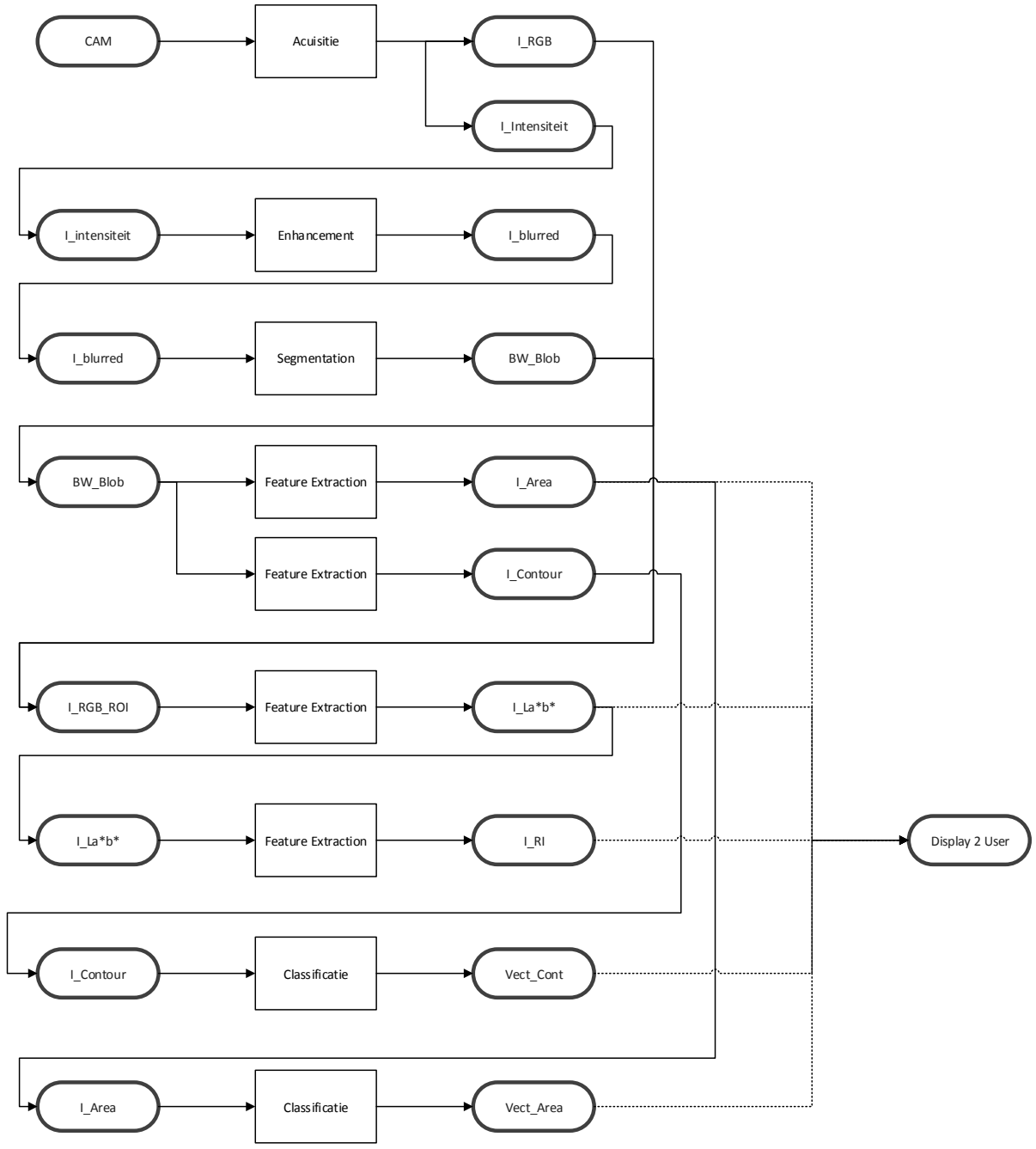
```
Inot = RemoveBackgroundVectorImage(~BW, I);
imshow(Inot);
```

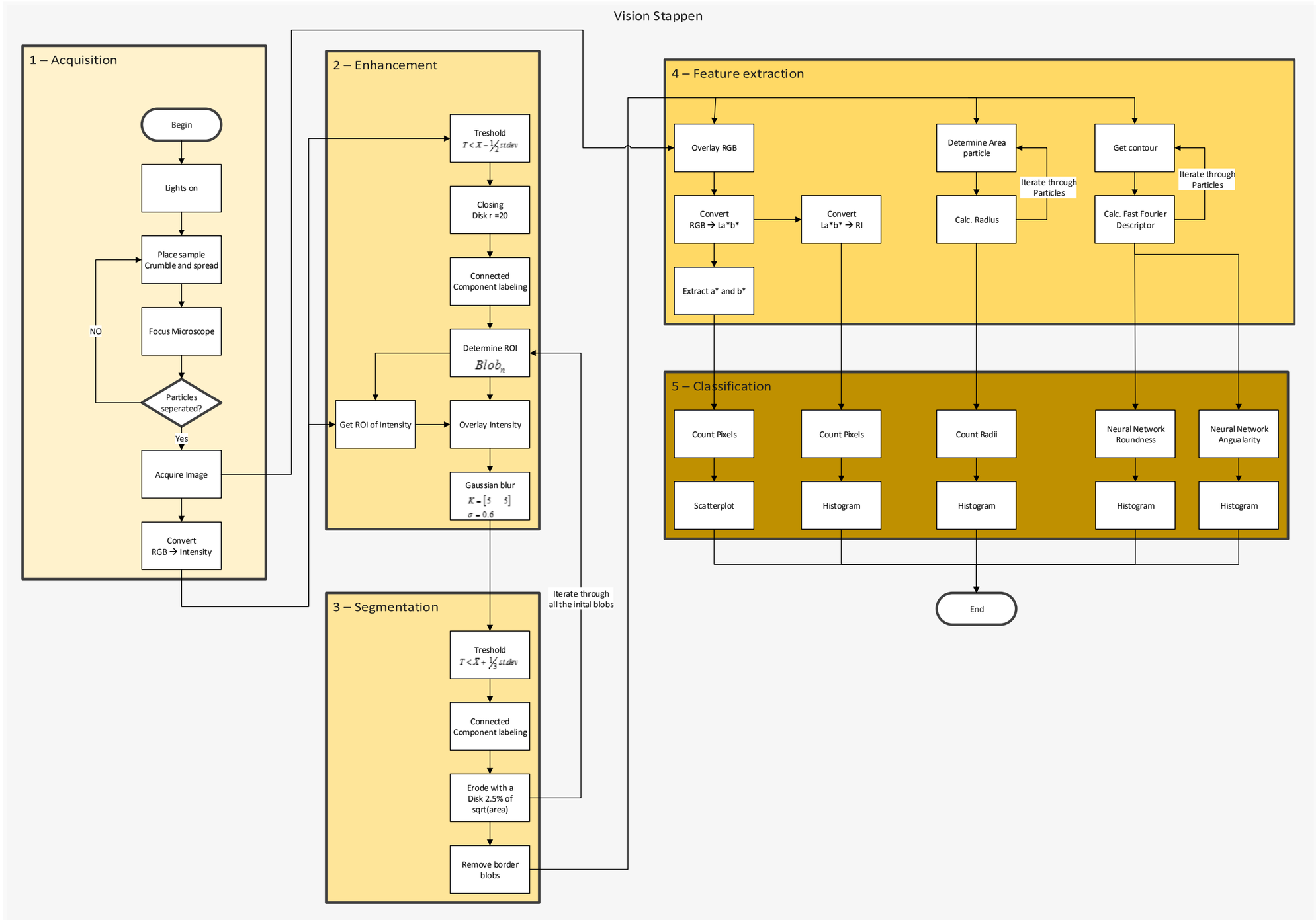
```
Warning: Image is too big to fit on screen; displaying at 50%
```

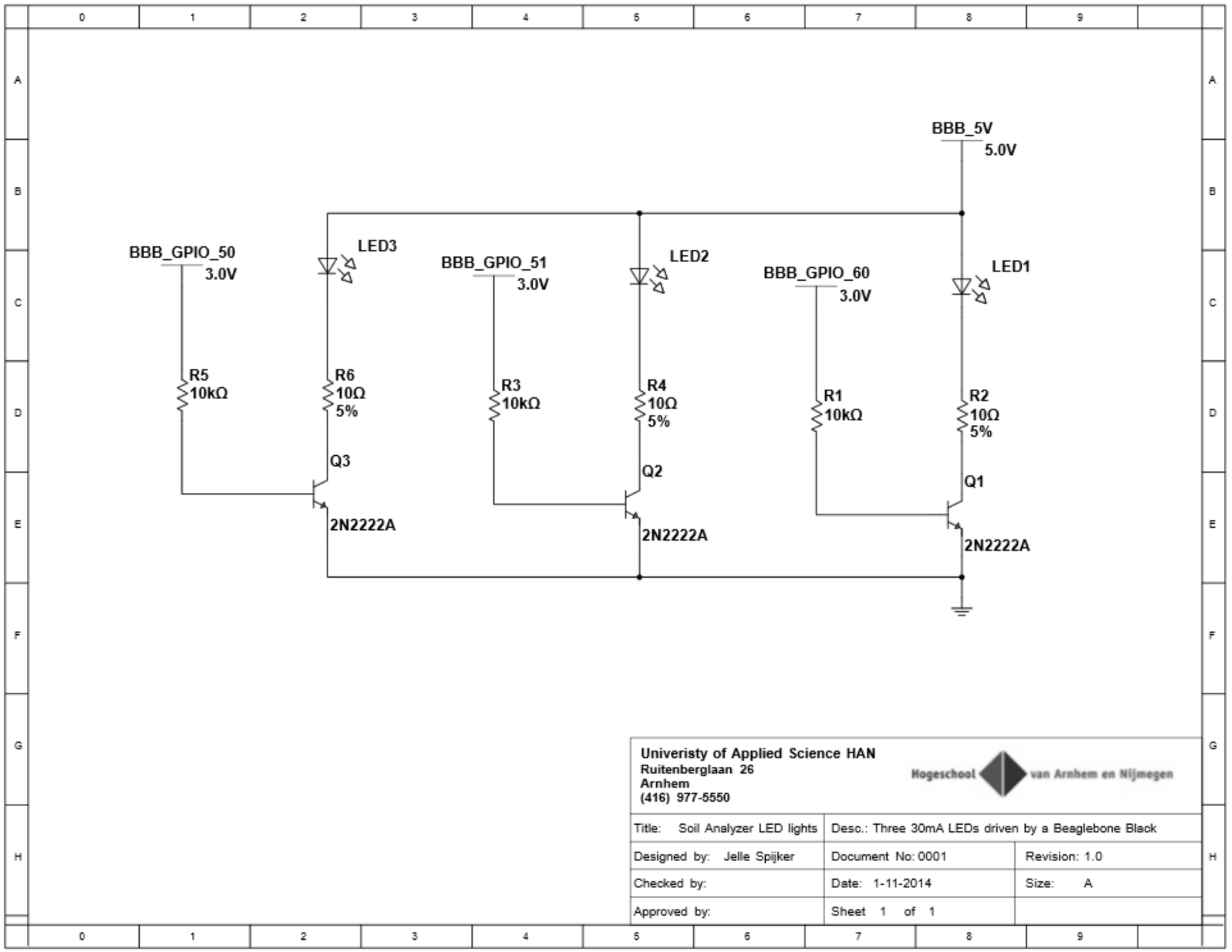



*Published with MATLAB® R2014b*

# Bijlage XIII. Vision Ontwerp




Vision Stappen
1 – Acquisition
Begin
Lights on
Place sample Crumble and spread
Focus Microscope
NO
Particles seperated?
Yes
Acquire Image
Convert RGB → Intensity
2 – Enhancement
Treshold
Closing Disk r =20
Connected Component labeling
Determine ROI
Get ROI of Intensity
Overlay Intensity
Gaussian blur
3 – Segmentation
Treshold
Connected Component labeling
Erode with a Disk 2.5% of sqrt(area)
Remove border blobs
Iterate through all the inital blobs
4 – Feature extraction
Overlay RGB
Convert RGB → La*b*
Extract a* and b*
Convert La*b* → RI
Determine Area particle
Calc. Radius
Iterate through Particles
Get contour
Calc. Fast Fourier Descriptor
Iterate through Particles
5 – Classification
Count Pixels
Count Pixels
Count Radii
Neural Network Roundness
Neural Network Angualarity
Scatterplot
Histogram
Histogram
Histogram
Histogram
End


BBB_5V
5.0V
BBB_GPIO_50
3.0V
LED3
R5
10kΩ
R6
10Ω
5%
Q3
2N2222A
BBB_GPIO_51
3.0V
LED2
R3
10kΩ
R4
10Ω
5%
Q2
2N2222A
BBB_GPIO_60
3.0V
LED1
R1
10kΩ
R2
10Ω
5%
Q1
2N2222A
Univeristy of Applied Science HAN
Ruitenberglaan 26
Arnhem
(416) 977-5550
Hogeschool van Arnhem en Nijmegen
Title: Soil Analyzer LED lights
Desc.: Three 30mA LEDs driven by a Beaglebone Black
Designed by: Jelle Spijker
Document No: 0001
Revision: 1.0
Checked by:
Date: 1-11-2014
Size: A
Approved by:
Sheet 1 of 1