

Subfaculteit wetenschappen



Probleemoplossen en ontwerpen 3

Biologische Data Analyse App

**Marthe Böting
Robin Bruneel
Toon Ingelaere**

Titularis : Koen Van Den Abeele

Begeleider : Senna Staessens

Academiejaar 2018 - 2019



BDA App

December 13, 2018

Marthe Böting, Robin Bruneel en Toon Ingelaere

Inleiding

Inhoudsopgave

1 Klantenvereisten	2
2 Ontwerpspecificatie	2
3 Onze oplossing	2
3.1 Achtergrondverwijdering	2
3.1.1 Bepalen van de beste threshold	3
3.1.2 Ruisfilter	5
3.2 Lokalisatie van de indicator	6
3.2.1 Het HSV kleurmodel	8
3.2.2 Een algemene threshold	8
3.2.3 Optimalisatie van de threshold	9

1 Klantenvereisten

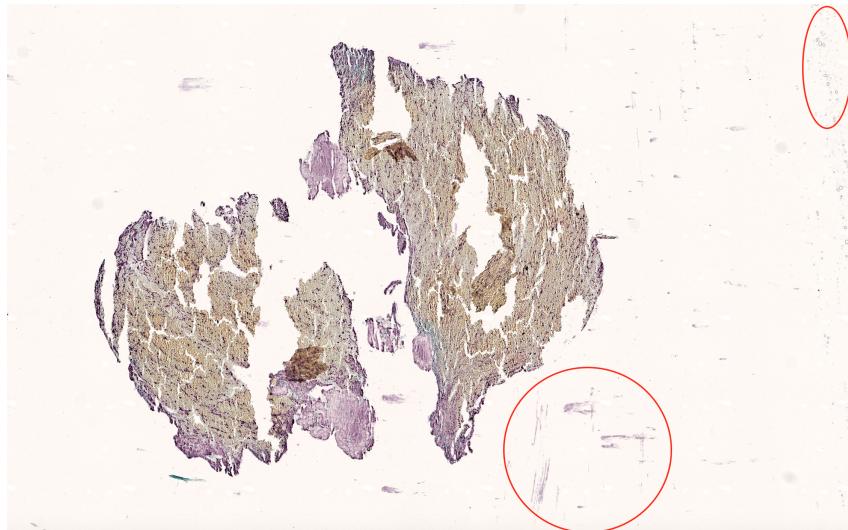
2 Ontwerpspecificatie

3 Onze oplossing

Dit hoofdstuk is opgedeeld in verschillende deelproblemen: het verwijderen van de achtergrond, het lokaliseren van de indicator en de gebruiksvriendelijke app. Dit alles is gerealiseerd met behulp van de programmeertaal MATLAB.

3.1 Achtergrondverwijdering

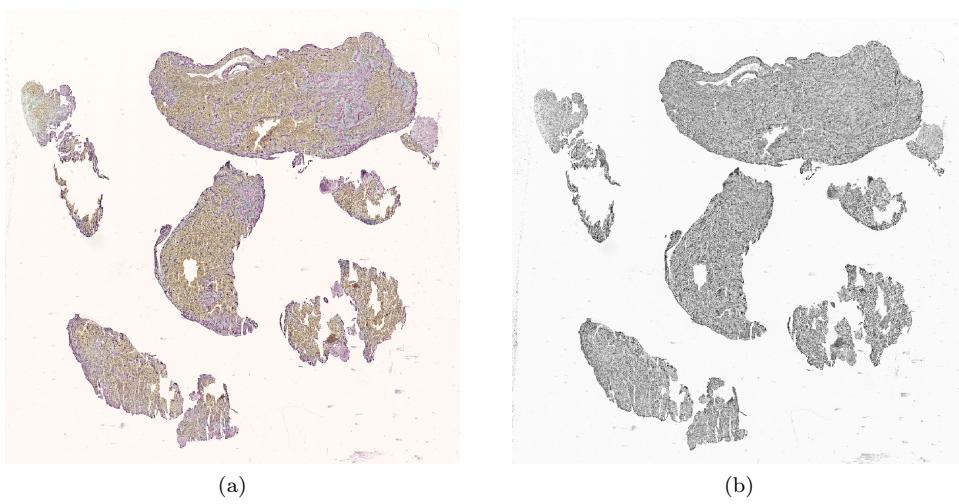
Het eerste deelprobleem van ons project is het verwijderen van de achtergrond. Op de afbeeldingen is er heel wat vuilheid te vinden. Voorbeelden hiervan zijn luchtbellen of kleine verkleuringen in de achtergrond zoals men ziet op de Figuur 1. Hieronder beschrijven we verschillende operaties om de grootste kloners te lokaliseren en alles wat geen klonter is te verwijderen. Dit leidt tot een ruisvrije afbeelding.



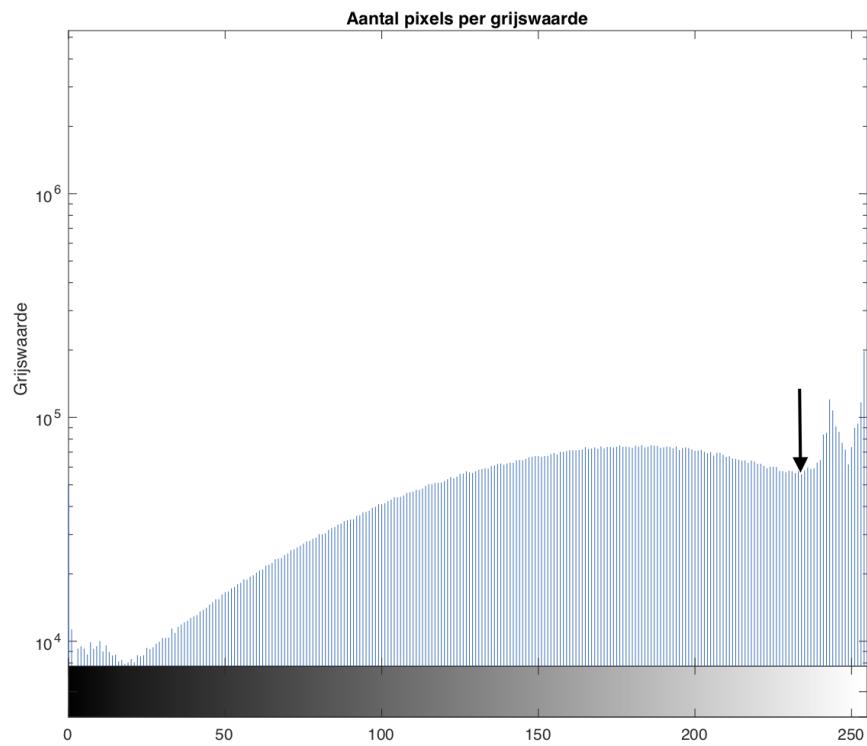
Figuur 1: In de rechterbovenhoek zien we luchtbellen en in de onderste cirkel zien we een verkleuring die geen deel uitmaakt van een bloedklonter.

3.1.1 Bepalen van de beste threshold

Het onderscheid tussen de achtergrond (eerder wit) en de bloedklonter (eerder grijs) is makkelijker te zien op de grijswaarden van de afbeelding, zie Figuur 2.. Daarnaast is één grijswaarde voldoende om het onderscheid te maken. Wanneer een histogram opgesteld wordt van deze waarden is een duidelijk dipje tussen het wit en het grijs te zien, zie Figuur 3. Dit is dan ook de theoretisch optimale threshold, de grijswaarde om een bloedklonterpixel van een achtergrondpixel te onderscheiden. Deze bepalen we simpelweg als het minimum in de tweede helft van de histogram. Na het toepassen van de threshold komt de binaire Figuur 4 tevoorschijn. Dit lijkt de kloners nagenoeg goed te detecteren.



Figuur 2: Illustratie van de originele foto (a) en deze omgezet in grijswaarden (b).



Figuur 3: Histogram van het aantal pixels gegroepeerd per grijswaarde. We zien duidelijk een lokaal minimum rond de waarde 230. Opmerking: we maken hier gebruik van een logaritmische y-as.



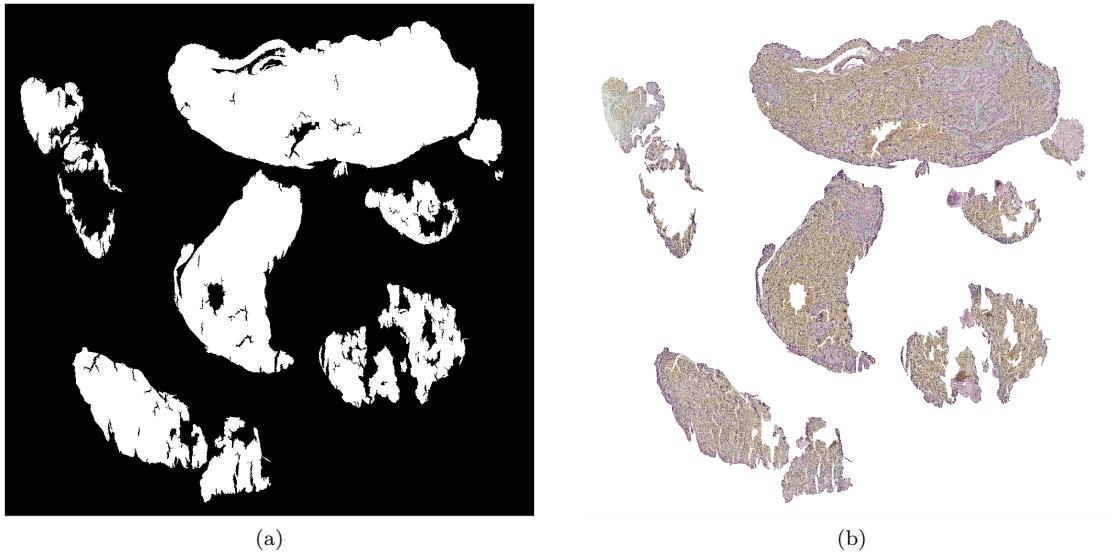
Figuur 4: Op deze binaire representatie zijn de bloedklonters duidelijk te zien.

3.1.2 Ruisfilter

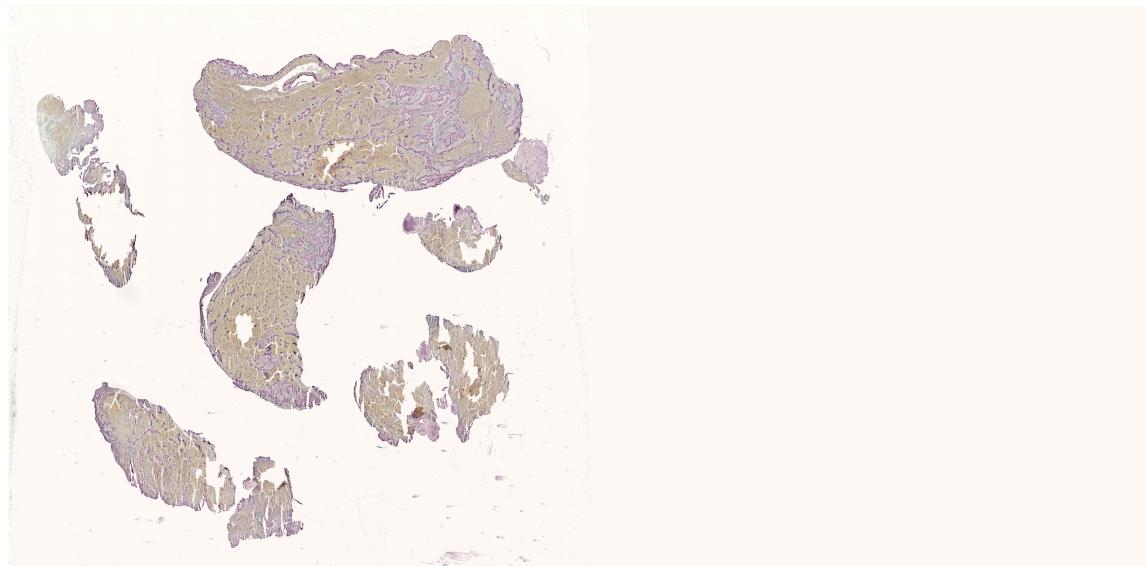
Eenmaal de binaire representatie is opgesteld, kunnen de bloedklonters nauwkeurig gelokaliseerd worden door al het overbodige ruis te verwijderen. De ruis kan makkelijk verwijderd worden door alle kleine groepen witte/zwarre pixels respectievelijk te vervangen door zwarte/witte pixels. Het resultaat is te zien op Figuur 5 (a).

Deze zogenaamde 'mask'¹ kan toegepast worden op de originele afbeelding om de ruisvrije afbeelding te vormen. Daarnaast wordt deze afbeelding bijgesneden om kostbare geheugenruimte te sparen, zoals te zien is in Figuur 5 (b). In dit deelprobleem werd echter met een reeds bijgesneden bloedklonten gewerkt om de afbeeldingen duidelijk te houden. Een volledig onbewerkte afbeelding is echter te zien in Figuur 6

¹Bij het toepassen van een mask op een afbeelding, worden alle pixels in de afbeelding die zwart zijn in de mask, wit gemaakt. De andere pixels behouden hun kleur.



Figuur 5: Alle ruis is verwijderd, dit is een foto waarmee we kunnen werken.



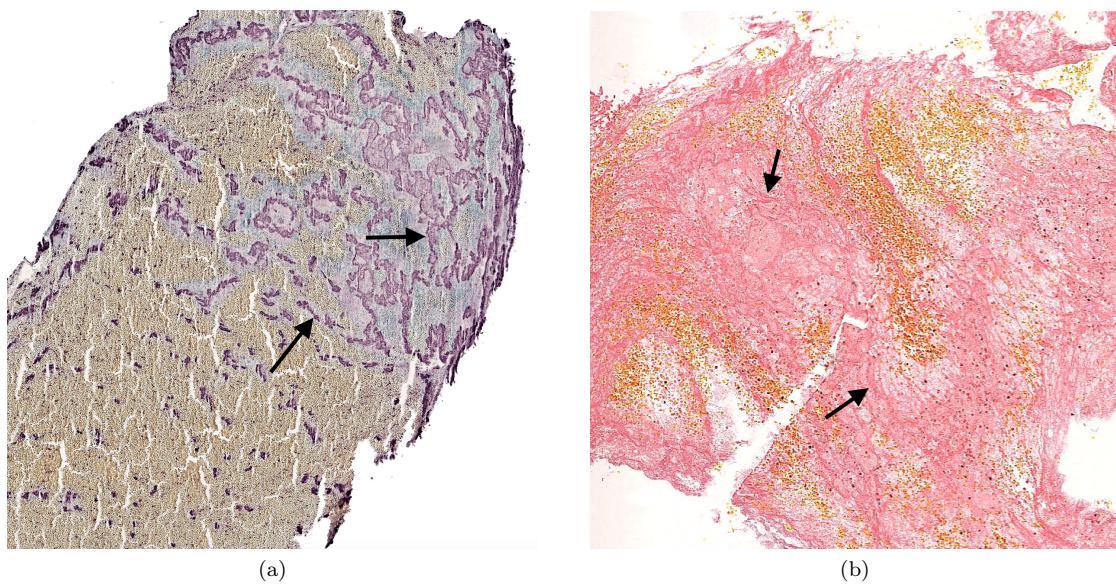
Figuur 6: Dit is een volledig onbewerkte afbeelding die we in 5 (b) hebben bijgesneden en waarvan de achtergrond verwijderd is.

3.2 Lokalisatie van de indicator

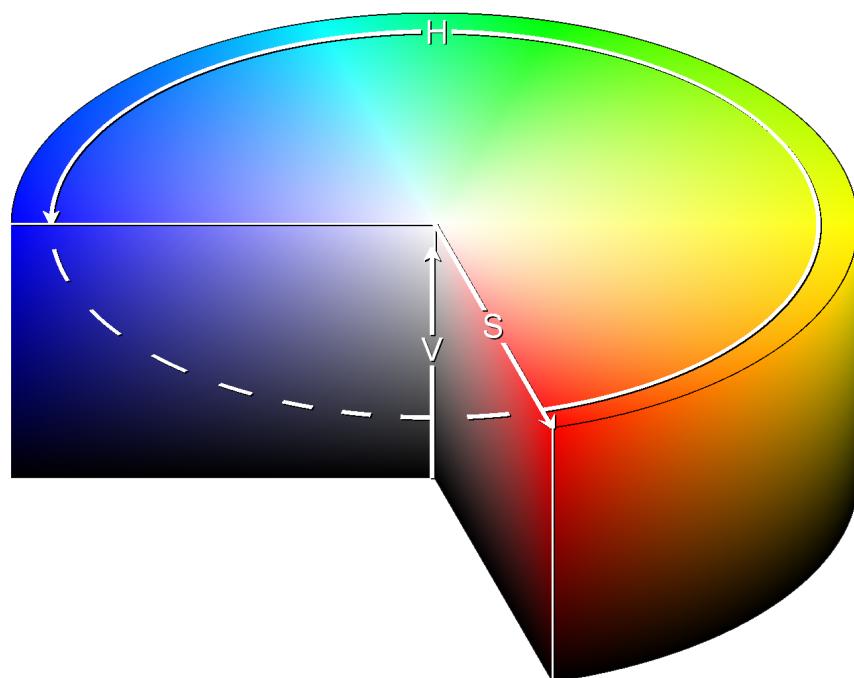
Het tweede deel van ons project is de indicators lokaliseren en kwantificeren. We hebben de opdracht gekregen twee soorten te kunnen onderscheiden. Van deze is een voorbeeld op Figuur 7 te zien. Omdat het kleurverschil tussen indicator en achtergrond niet altijd even groot is, vormen we het oorspronkelijke *RGB* kleurmodel² om naar het zogenaamde *HSV* kleurmodel. Dit model is een alternatieve voorstelling waarbij men alle kleuren op een cirkel voorstelt, de hoek die dit kleur dan maakt, noemt men de *Hue*. Naast deze waarde heeft *HSV* nog twee andere parameters namelijk *Saturation* en *Value*. *Saturation* kan simpel beschouwd worden als een aanduiding van de hoeveelheid witte kleur en *Value* een aanduiding van de zwarte kleur. Een grafische voorstelling is te zien op Figuur 8.

²Het *RGB* kleurmodel is een voorstelling waarbij ieder kleur voorgesteld wordt door een waarde van de drie basiskleuren (rood, groen en blauw)

Het voordeel van deze transformatie is dat het heel wat eenvoudiger is om een onderscheid tussen dichtbijgelegen kleuren te vinden. Een andere mogelijke transformatie is die naar het *LAB* of het *CYMK* kleurmodel die gelijkaardige eigenschappen heeft en desnoods ook gebruikt kan worden. Eenmaal we een duidelijk onderscheid tussen de indicator en de achtergrond maken, kunnen we eenvoudig het percentage indicator berekenen door het aantal bloedklonter- en indicatorpixels op te tellen. Ons stappenplan wordt in volgende hoofdstukken besproken en wordt toegepast op de afbeelding uit Figuur 7 (a).



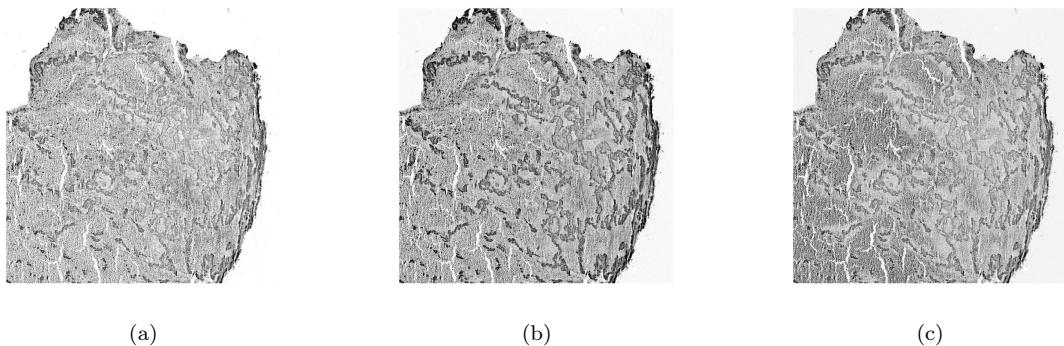
Figuur 7: Illustratie van de twee soorten indicator (respectievelijk paars en donkerroze) die gedetecteerd moeten worden. We zien duidelijk dat het detecteren op afbeelding (a) eenvoudiger zal zijn dan op afbeelding (b).



Figuur 8: Grafische voorstelling van het *HSV* (*Hue, Saturation, Value*) kleurmodel

3.2.1 Het HSV kleurmodel

Zoals reeds vermeld, beginnen we met een transformatie naar het *HSV* kleurmodel. Bijgevolg kunnen we de twee voorstellingen vergelijken. We hebben het model telkens ontbonden in de drie kleurwaarden, waarbij zwart de laagste waarde voor dat kleur is en wit de hoogste. De resultaten zijn te zien in de Figuren 9 en 10. Het verschil tussen de twee kleurmodellen is duidelijk te zien. Afbeelding (a) en (b) uit Figuur 10 lijken namelijk een iets agressiever onderscheid te maken.

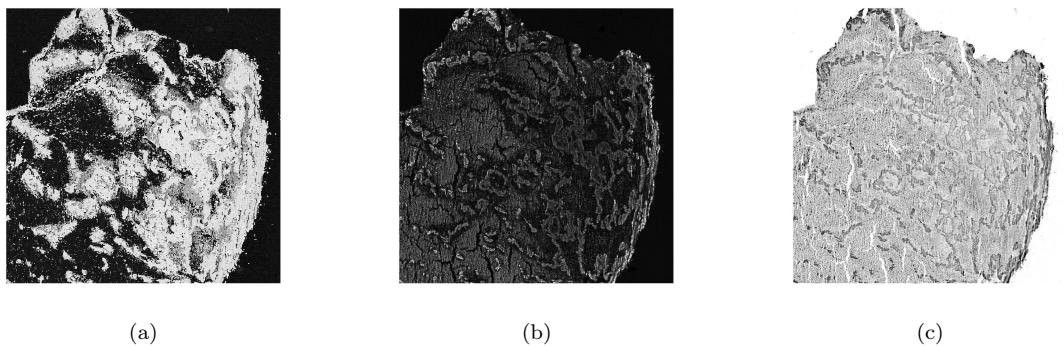


(a)

(b)

(c)

Figuur 9: Illustratie van respectievelijk de rode, groene en blauwe kleurwaarden (*RGB*). Hierbij komt wit overeen met de maximumwaarde en zwart met de minimumwaarde van die kleur.



(a)

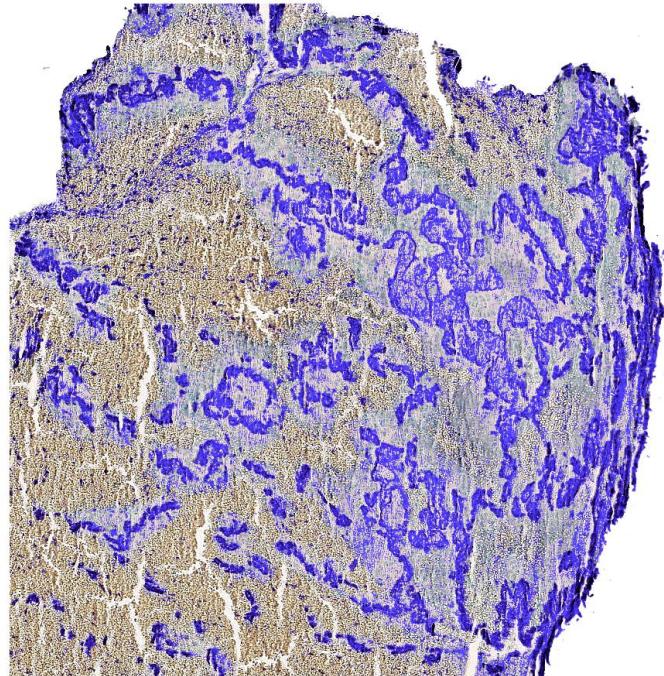
(b)

(c)

Figuur 10: Illustratie van respectievelijk de *hue*, *saturation* en *value* kleurwaarden (*HSV*). Hierbij komt wit overeen met de maximumwaarde en zwart met de minimumwaarde van die kleurwaarde.

3.2.2 Een algemene threshold

De volgende stap is nu een filter bepalen voor alle indicatorpixels. Het probleem is echter dat een goede filter voor de ene foto niet altijd een goede filter voor de andere foto is. Daarom hebben we voor iedere foto manueel een 'threshold' bepaald en deze achteraf met elkaar vergeleken. Het resultaat is een vrij algemene threshold die alle indicatorpixels met zekerheid aanduidt. Af en toe worden jammer genoeg verkeerde pixels aangeduid. Het aantal is weliswaar niet zo groot, maar dit zullen we trachten te omzeilen in het volgende hoofdstuk. Een voorbeeld van deze algemene threshold is te zien op de Figuur 11.

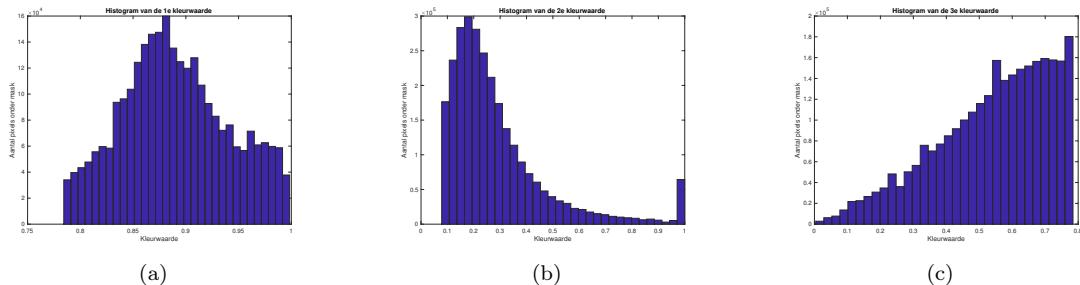


Figuur 11: We hebben alle indicatoren met een blauwe kleur aangeduid. De algemene threshold selecteert alle indicatorpixels, maar jammer genoeg ook enkele verkeerde (voornamelijk rechtsonder).

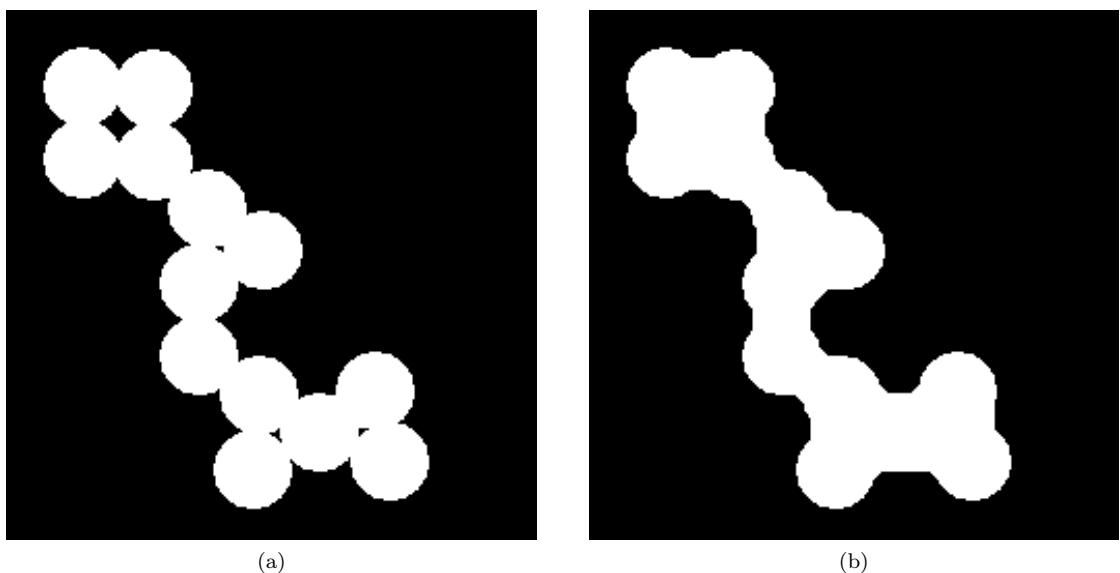
3.2.3 Optimalisatie van de threshold

In dit hoofdstuk willen we het aantal verkeerde pixels verminderen zonder de juiste te beïnvloeden. We hebben echter al een vrij goede threshold waardoor we een statistische analyse van wat onder deze mask ligt, kunnen uitvoeren. We stellen hiervoor een histogram van de *HSV* kleurwaarden onder de mask op. Het levert ons namelijk een interessant resultaat dat te zien is in Figuur 12. We zien namelijk duidelijke verschillen in de frequenties van de kleurwaarden. De grafieken tussen de verschillende afbeeldingen zijn nagenoeg gelijk van vorm, maar de pieken liggen soms meer dan 5% verschoven.

Het idee is om nu de verkeerde pixels via deze diagrammen eruit te filteren. Indien we dat de verkeerde pixels essentieel in kleur verschillen van indicatorpixels en dat ze niet vaak voorkomen. Dan kunnen we in principe alle pixels met een frequentie onder een bepaalde grenswaarde schrappen. Een betere benadering is misschien om het punt te vinden, waar de frequentie van de pixels enorm begint toe te nemen of we met andere woorden grote 'indicatoraders' aan het verwijderen zijn. Dit punt kan theoretisch benaderd worden als het maximum van de tweede afgeleide naar de kleurwaarde. Hierdoor worden nog iets minder verkeerde pixels aangeduid zoals verwacht. Het nadeel is dat ook alle pixels met een waarde rond de 1 op Figuur 12 natuurlijk negeren. Waardoor deze weer manueel toegevoegd moeten worden. Een extra stap is tenslotte nog de indicatorpixels morfologisch te sluiten met een algoritme dat reeds in MATLAB verwerkt is. Dit is zeer nuttig omdat de indicatorpixels meestal met elkaar verbonden zijn en daardoor deze weer onderling kunnen verbinden.



Figuur 12: Histogrammen van de HSV kleurwaarden. We zien duidelijk het verschil in frequenties.



Figuur 13: Illustratie waarbij de witte cirkels uit afbeelding (a) morfologisch gesloten worden in afbeelding (b)

	Week 1							Week 2							Week 3							Week 4							Week 5							Week 6						
	Sep							Oct							Nov																											
	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	S	S	M	T	W	F	S	S	M	T	W	T	F	S	S	S									
24	25	26	27	28	29	30	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	01	02	03	04	

1 2 3

T.T. verslag

Week 1		Week 2		Week 3		Week 4		Week 5		Week 6		Week 7	
		Nov						Dec					
05	06	07	08	09	10	11	12	13	14	15	16	17	18
M	T	W	T	F	S	S	M	T	W	F	S	S	M

3

4

Endvertrag

Endpres.

Taaknummer	Taakomschrijving
1	Achtergrond verwijderen en afbeelding bijsnijden
2	Kleuren detecteren en kwantificeren
3	Ontwerpen van een gebruiksvriendelijke app
4	Optimalisatie van het algoritme

Table 1: Omschrijvingen van de taken in de gantt-chart