

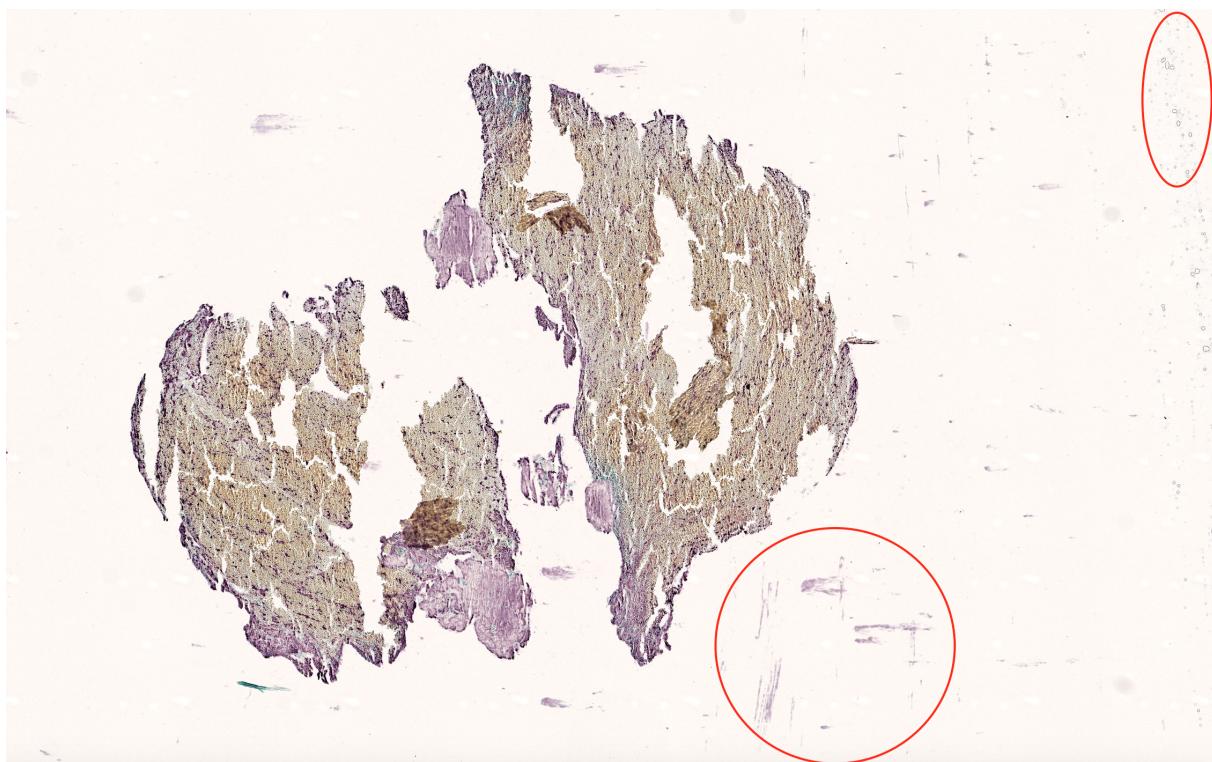
**Titel van het document**

Academiejaar 2017-2018

Naam van de auteurs

**Motivatie****1 Onze oplossing****1.1 Achtergrondverwijdering**

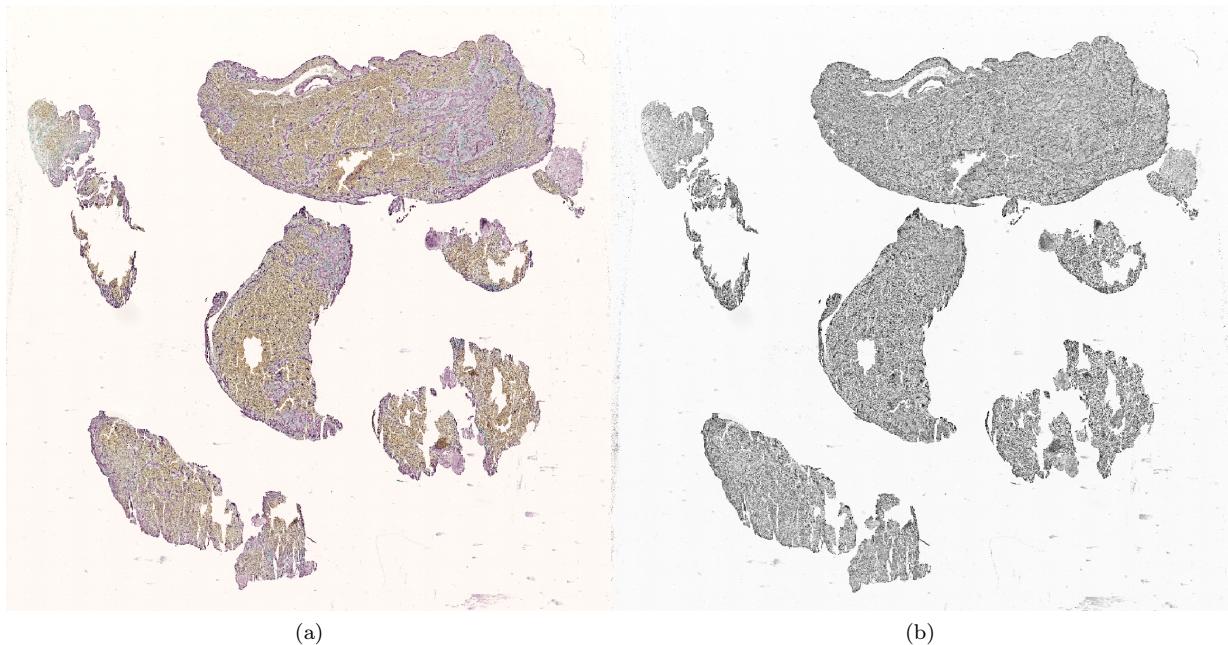
Het eerste deel van ons project is de achtergrond van de bloedklonters te verwijderen. Op de afbeeldingen is er heel wat vuilheid te vinden. Een voorbeeld hiervan zijn luchtbellen of kleine bloedklonters zoals men ziet op de volgende figuur 1. Hieronder zullen we verschillende operaties beschrijven om de grootste klonters te lokaliseren en alles dat geen klonter is te verwijderen. Dit leidt tot een ruisvrije afbeelding.



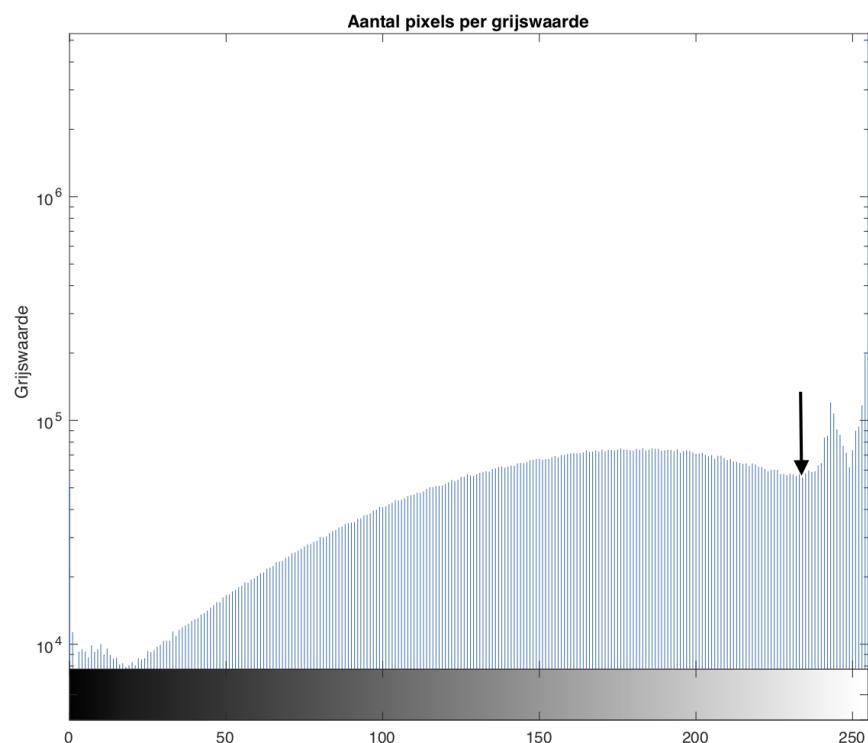
Figuur 1: In de rechterbovcircel zien we luchtbellen en in de onderste cirkel zien we een verkleuring die geen deel uitmaakt van een bloedklonter.

**1.1.1 Bepalen van de beste threshold**

Wanneer we de grijswaarden van de afbeelding berekenen, zien we een duidelijk verschil tussen de achtergrond (eerder wit) en de bloedklonter (eerder grijs) zoals we zien op figuur 2. Wanneer we een histogram van deze grijswaarden opstellen, vinden we ook dat er een lokaal minimum is tussen het grijs en het wit. Dit zien we duidelijk op het histogram 3. Wanneer we dit minimum berekenen vinden we de theoretisch optimale threshold, de waarde om een bloedklonterpixel van een achtergrondpixel te onderscheiden. Wanneer we die threshold toepassen en een binaire representatie zoals figuur 4 vormen, zien we duidelijk dat dit een goede threshold is.



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

### 1.1.2 Lokaliseren van de bloedklonters

Eenmaal deze binaire representatie gevonden is, lokaliseren we de grootste klonters om het ruis te verwijderen. Hiervoor zullen we eerst alle holtes in de klonters opvullen door alle ingesloten pixels<sup>1</sup> wit te maken. Het resultaat is te zien in figuur 5. De klonters zijn nu nog veel duidelijker zichtbaar, waardoor we kunnen overgaan naar de effectieve ruisfilter.

---

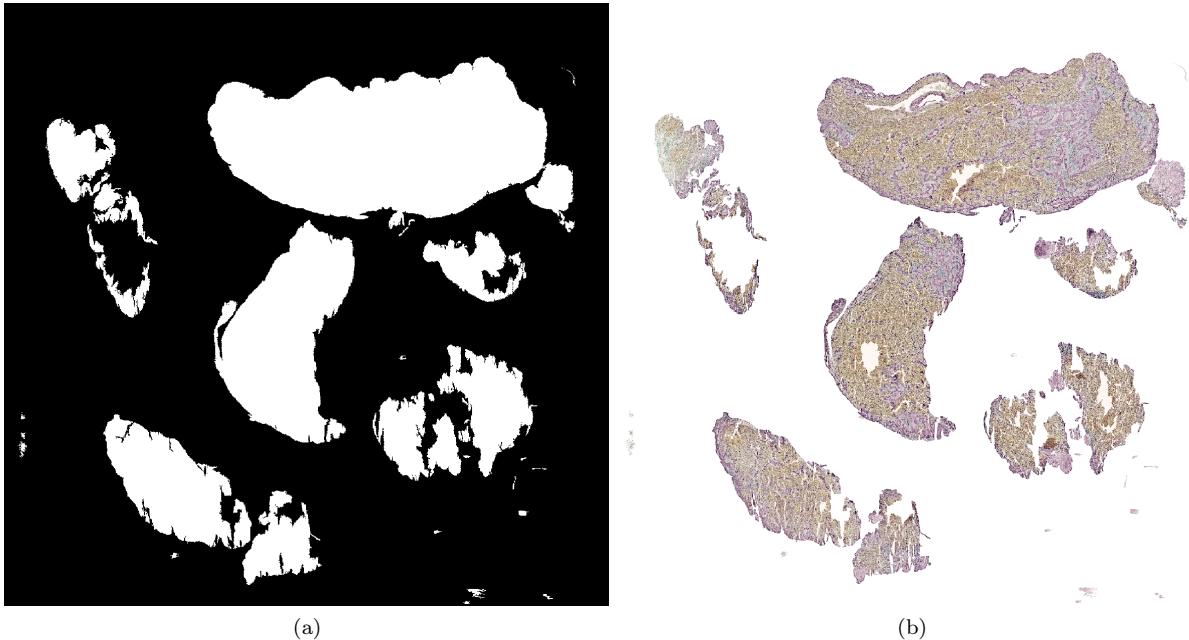
<sup>1</sup>Een ingesloten pixel is een zwarte pixel die onmogelijk de achtergrond van de afbeelding kunnen bereiken zonder over witte pixels te gaan



Figuur 5: Alle gaatjes zijn nu opgevuld, waardoor we de grootte van de bloedklonters kunnen berekenen

### 1.1.3 Ruisfilter

Voor de ruisfilter verwijderen we elke groep witte pixels met een omvang kleiner dan een constante waarde. Indien dit zo is, worden de witte pixels door zwarte vervangen. Dit simpel algoritme verwijdert alle ruis zoals te zien is in figuur 6. Deze zogenaamde 'mask' kunnen we toepassen op de originele afbeelding om officieel het ruis te verwijderen.



Figuur 6: Alle ruis is verwijderd, dit is een foto waarmee we kunnen werken

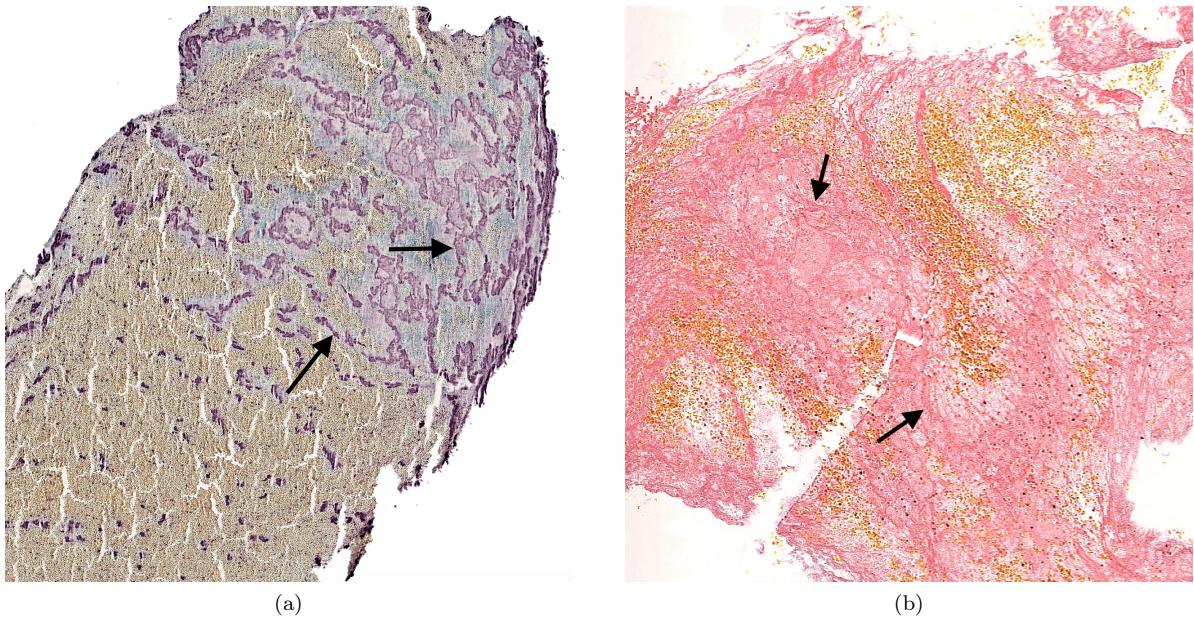
## 1.2 Lokalisatie van de indicator

Het tweede deel van ons project is de indicators lokaliseren. We hebben de opdracht gekregen twee soorten te kunnen onderscheiden. Een voorbeeld van deze twee is op de volgende figuren 7 te zien. Omdat het kleurverschil tussen indicator en achtergrond niet altijd even groot is, vormen we het oorspronkelijke *RGB* kleurmodel<sup>2</sup> 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 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 grafisch overzicht is te zien op figuur 8.

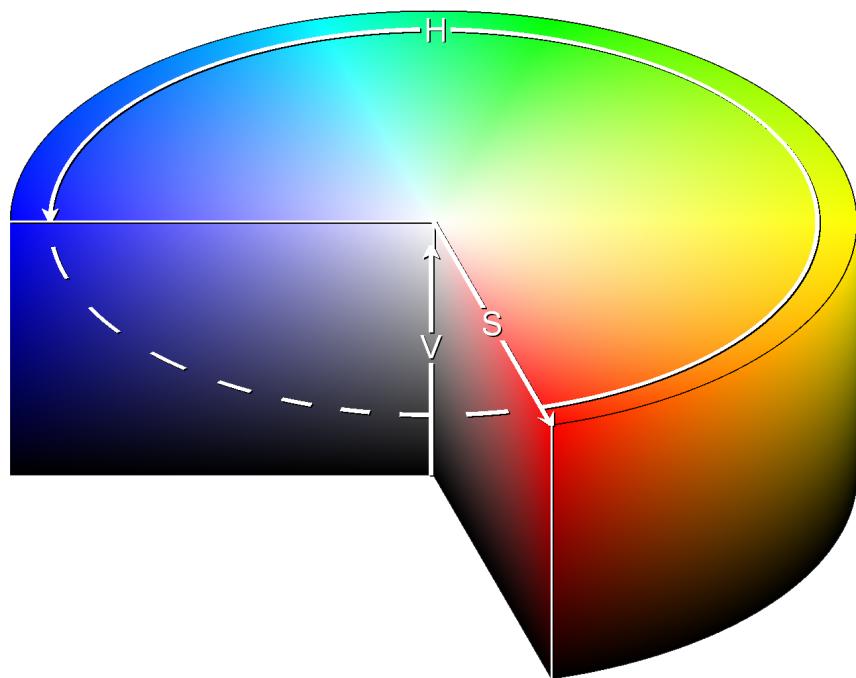
Het voordeel van deze transformatie is dat het heel wat eenvoudiger is om een onderscheid tussen dichtbijgelegen kleuren te vinden. Een andere mogelijke transformatie is die naar het *Lab* kleurmodel die gelijkaardige eigenschappen heeft en desnoods ook gebruikt kan worden. Eenmaal we een duidelijk onderscheid 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).

---

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



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)

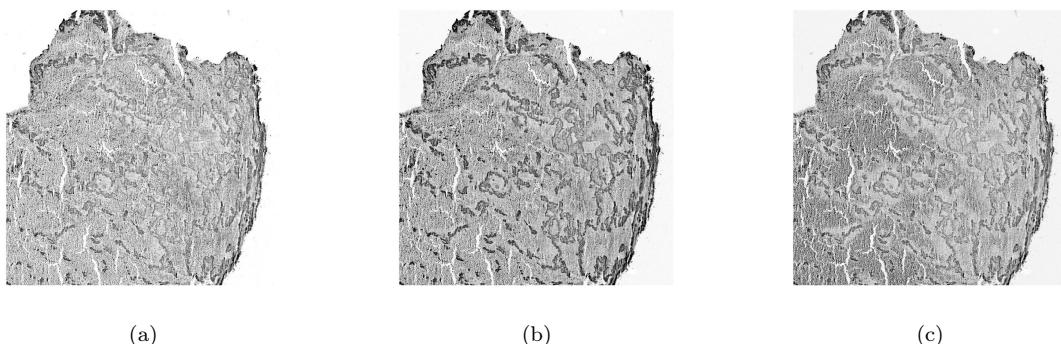


Figuur 8: Grafische voorstelling van het HSV (Hue, Saturation, Value) kleurmodel

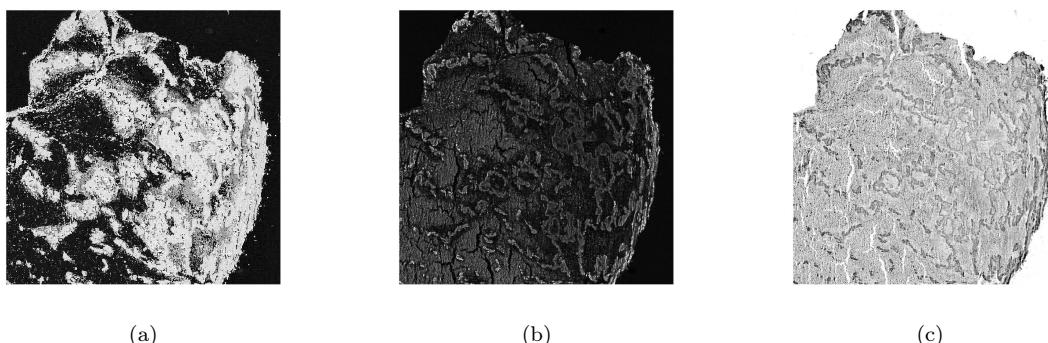
### 1.2.1 het HSV kleurmodel

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 iets agressiever een onderscheid tussen de kleuren te maken.



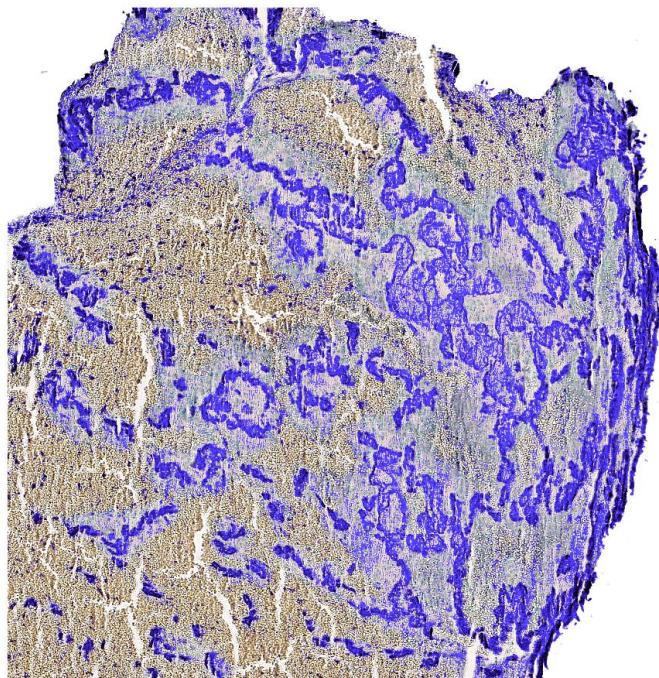
Figuur 9: Illustratie van respectievelijk de rode, groene en blauwe kleurwaarden (RGB). Hierbij komt wit overeen met de maximumwaarde en zwart met de minimumwaarde van die kleur.



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.

### 1.2.2 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 zullen we trachten te omzeilen in het volgende hoofdstuk. Een voorbeeld van deze algemene threshold is te zien op de volgende figuur:



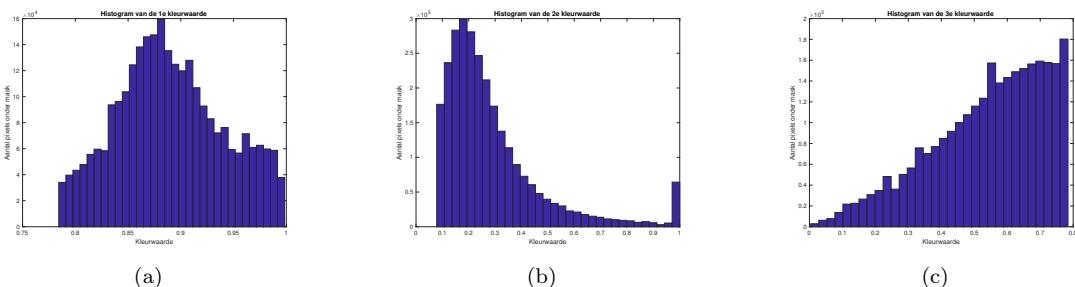
Figuur 11: We hebben alle indicatoren met een blauwachtige kleur aangeduid. De algemene threshold selecteert alle indicatorpixels, maar jammer genoeg ook enkele verkeerde (voornamelijk rechtsonder).

### 1.2.3 Optimalisatie van de threshold

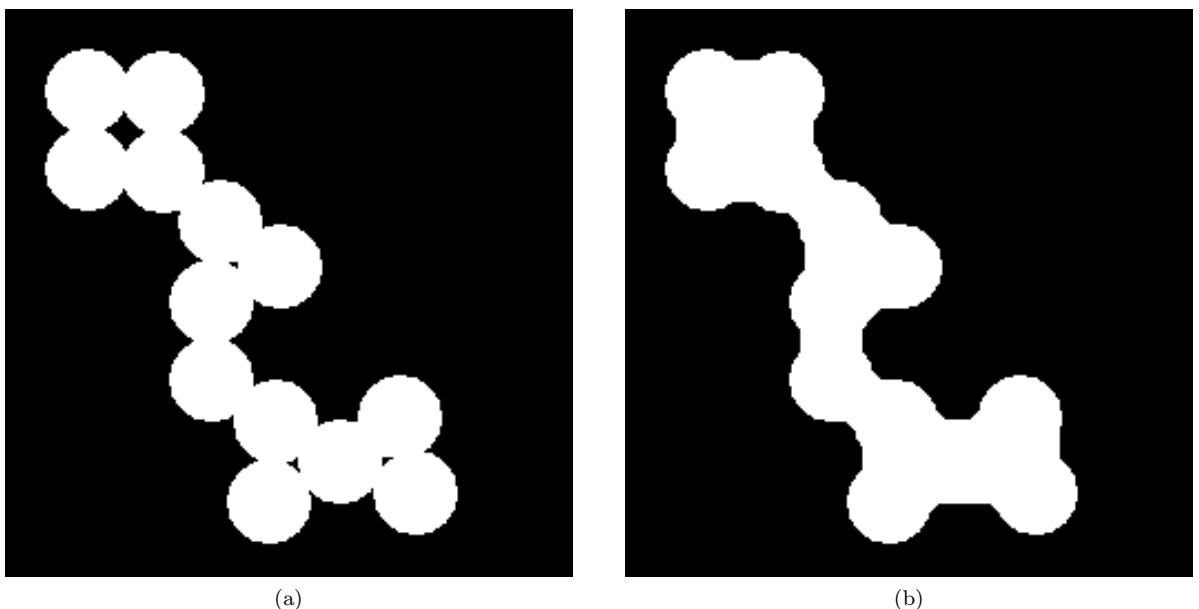
In dit hoofdstuk willen we het aantal verkeerde pixels verminderen zonder de juiste te beïnvloeden. We hebben echter al een vrij goede threshold waardoor we een statistische analyse van wat onder deze mask ligt, kunnen uitvoeren. We stellen hiervoor een frequentiediagram van de *HSV* kleurwaarden onder 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.

Het idee is om nu de verkeerde pixels via deze diagrammen eruit te filteren. Indien we veronderstellen dat de frequenties van deze specifieke pixels laag zijn en dat de verkeerde pixels in kleur verschillen met de indicatorpixels. 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. Jammer genoeg hebben we de tijd nog niet gehad om dit idee te testen.

Een mogelijk extra stap is om de indicatorpixels morfologisch te sluiten met een algoritme dat reeds in MATLAB verwerkt is. Indien we met onze filtering enkele holtes maken, kunnen deze dan zonder problemen worden opgevuld. Ter illustratie geven we het standaard voorbeeld van MATLAB in figuur 13 weer.



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)

### 1.3 De gebruiksvriendelijke applicatie

## Besluit

Afsluitende tekst