

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

### Inhoudsopgave

1 ~~Kantoorvereisten~~ 27.01.18

---

<b>2 Ontwerpspecificatie</b>	<b>3</b>
<b>3 Onze oplossing</b>	<b>3</b>
3.1 Achtergrondverwijdering . . . . .	3
3.1.1 Bepalen van de beste threshold . . . . .	3
3.1.2 Lokaliseren van de bloedklonters . . . . .	3
3.1.3 Ruisfilter . . . . .	4
3.2 Lokalisatie van de indicator . . . . .	4
3.2.1 het HSV kleurmodel . . . . .	4
3.2.2 Een algemene threshold . . . . .	4
3.2.3 Optimalisatie van de threshold . . . . .	4
3.3 De gebruiksvriendelijke applicatie . . . . .	5
<b>4 Voorlopige resultaten</b>	<b>5</b>
<b>5 Verantwoordelijkheden en taakverdeling</b>	<b>5</b>
<b>6 Integratie met vakken uit eerste 3 semesters</b>	<b>5</b>
<b>7 Besluit</b>	<b>5</b>

# Inleiding

Als gevolg van een bloedklonter in een hersenbloedvat ontstaat een ischemische beroerte. Deze klonter moet men zo snel mogelijk verwijderen dit kan met behulp van een geneesmiddel die de klonter oplost of door middel van mechanische verwijdering. Door deze mechanische verwijdering kan men de bloedklonter verder analyseren in het labo om een duidelijker beeld te krijgen.

Vandaag de dag verliest men zeer veel tijd aan het verwerken van de bloedklonters. Er is ons dan ook gevraagd om een app te ontwikkelen die de analyse van de foto's volledig automatisch kan uitvoeren.

In dit verslag gaan we eerst in op wat de klant specifiek van ons verwacht en aan welke specificaties ons ontwerp moet voldoen. Hierna gaan we ons design bespreken en toelichten. Verder bespreken we ook onze voorlopige resultaten. Ten slotte wordt er nog een blik geworpen naar de vakken uit eerste 3 semesters die ons hierbij geholpen hebben.

## 1 Klantenvereisten

De klant verwacht een gebruiksvriendelijke app waarbij hij een foto kan ingeven en dat deze automatisch bewerkt wordt. De foto moet worden bijgesneden en de achtergrond moet verwijderd worden. Daarnaast is het de bedoeling om de bloedklonter te analyseren met andere woorden het berekenen van het percentage van een eiwit. Dit gebeurt door een hoeveelheid kleur te quantificeren in de bloedklonter.

## 2 Ontwerpspecificatie

De klant wil een app die elke foto kan bijwerken en ook een percentage van de eiwitten weergeven. Er zijn geen verdere voorwaarden verbonden aan de grootte van de foto. Een absolute voorwaarde om de foto bij te snijden is dat de bloedklonter er volledig opstaat. Desondanks mag de rand niet te groot zijn omdat dit anders te veel geheugen in beslag neemt. De achtergrond moet volledig wit gemaakt worden zodat er geen fouten worden gemaakt bij het berekenen van een hoeveelheid kleur. De app moet voor iedereen bruikbaar zijn. Daarnaast moeten de verschillende foto's over de verschillende fasen getoond worden.

## 3 Onze oplossing

### 3.1 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 kloners te lokaliseren en alles dat geen klonter is te verwijderen. Dit leidt tot een ruisvrije afbeelding.

#### 3.1.1 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.

#### 3.1.2 Lokaliseren van de bloedklonters

Eenmaal deze binaire representatie gevonden is, lokaliseren we de grootste kloners om het ruis te verwijderen. Hiervoor zullen we eerst alle holtes in de kloners opvullen door alle ingesloten pixels<sup>1</sup> wit te maken. Het resultaat is te zien in figuur 5. De kloners 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

### 3.1.3 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.

## 3.2 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).

### 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 iets agressiever een onderscheid tussen de kleuren te maken.

### 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 zullen we trachten te omzeilen in het volgende hoofdstuk. Een voorbeeld van deze algemene threshold is te zien op de volgende figuur 11

### 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 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 standaardvoorbeeld van MATLAB in figuur 13 weer.

<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)

### 3.3 De gebruiksvriendelijke applicatie

Momenteel zijn dit nog afzonderlijke programma's waarbij we steeds zelf de te aan te passen foto erin inzetten. Naar de klant toe is dit geen manier van werken en wordt dit in een app samengebracht. De klant zal dan zelf kunnen ingeven welke foto hij wil laten verwerken. Eénmaal de foto gekozen is, zal in de eerste fase de achtergrond van de foto wit worden gemaakt en zal de foto afgesneden worden zodanig dat enkel de bloedklonter zichtbaar is. Als dit gebeurd is, zal er al een eerste resultaat getoond worden. Hierdoor heeft de klant een beeld waarop de verdere analyse zal gebeuren en kan die eventueel ingrijpen bij fouten. Vervolgens zal de kleurenanalyse gebeuren. Ook hiervoor zal de foto getoond worden waarop de klant kan zien welke delen de indicator bevatten en zal het percentage gegeven worden. De basis van de app hebben we al zoals te zien is in figuur 14

## 4 Voorlopige resultaten

Op dit moment kunnen we al effectief de foto gebruiksklaar maken voor de analyse ervan. Dit houdt in dat we alle pixels die niet tot de bloedklonter behoren wit kunnen maken en de afbeelding zodanig kunnen bijsnijden dat er geen overbodig geheugen ingenomen wordt door de afbeelding. In Figuur 15 wordt dit geïllustreerd.

Momenteel zijn we bezig met de implementatie van de kleurendetectie. We hebben hier al veel vooruitgang geboekt, maar de huid staat nog niet volledig op punt. Ook hebben we al even geëxperimenteerd met de app designer van Matlab. Hier willen we namelijk onze gebruiksvriendelijke applicatie in maken. Op basis van wat we nu al bereikt hebben, denken we dat dit project een succes kan worden en dat men het effectief zal kunnen gebruiken in het onderzoek naar de samenstelling van bloedklonters.

## 5 Verantwoordelijkheden en taakverdeling

In onze groep hebben we ervoor gekozen om Robin aan te stellen als projectleider. Hij verdeelt elke week de taken en heeft het laatste woord bij discussies. Voor de verslagen en de eindpresentatie is Marthe verantwoordelijk. Zij is de eindredactrice van alle verslagen. We schrijven de verslagen uiteraard samen, maar het is haar taak ervoor te zorgen dat deze volledig en gestructureerd zijn en dat ze voor de deadline ingediend worden.

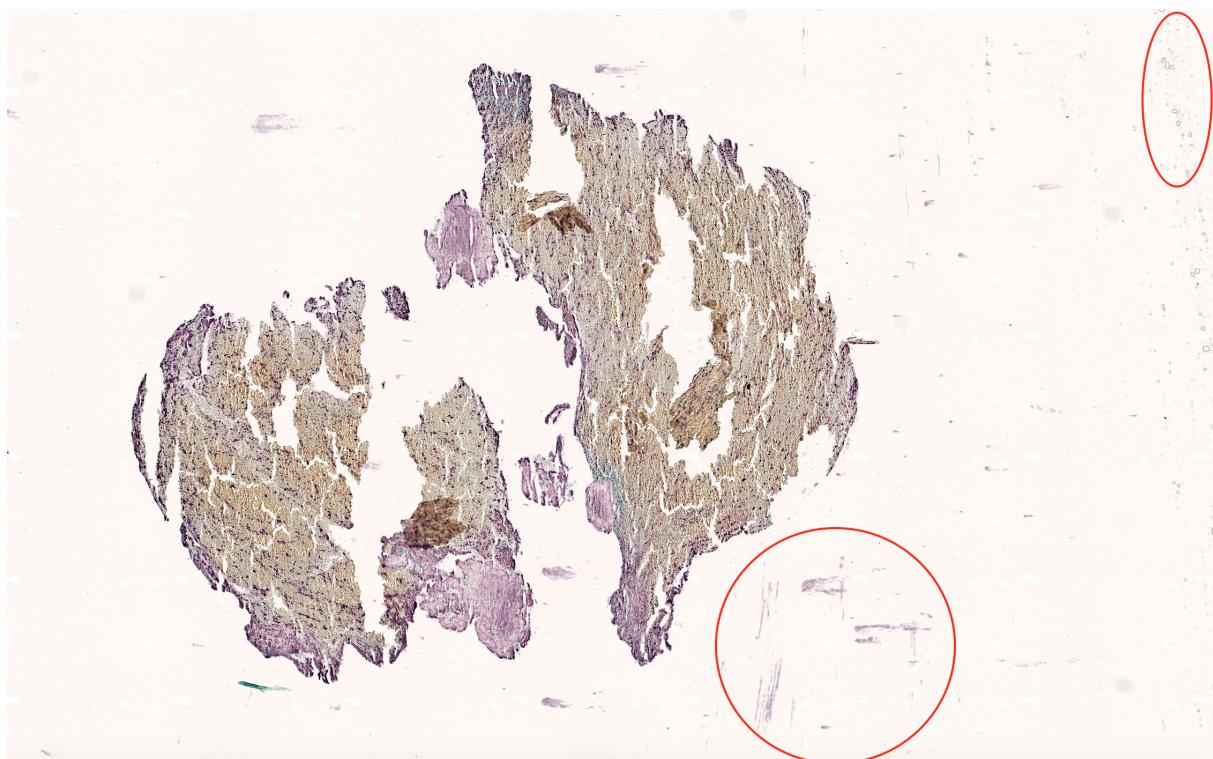
Voor de implementatie van de code zijn we elk verantwoordelijk voor ons eigen deel. Zo wordt het verwijderen van de ruis en het bijsnijden van de afbeelding door Toon geleid, de kleurdetectie door Robin en de implementatie van de app door Marthe.

## 6 Integratie met vakken uit eerste 3 semesters

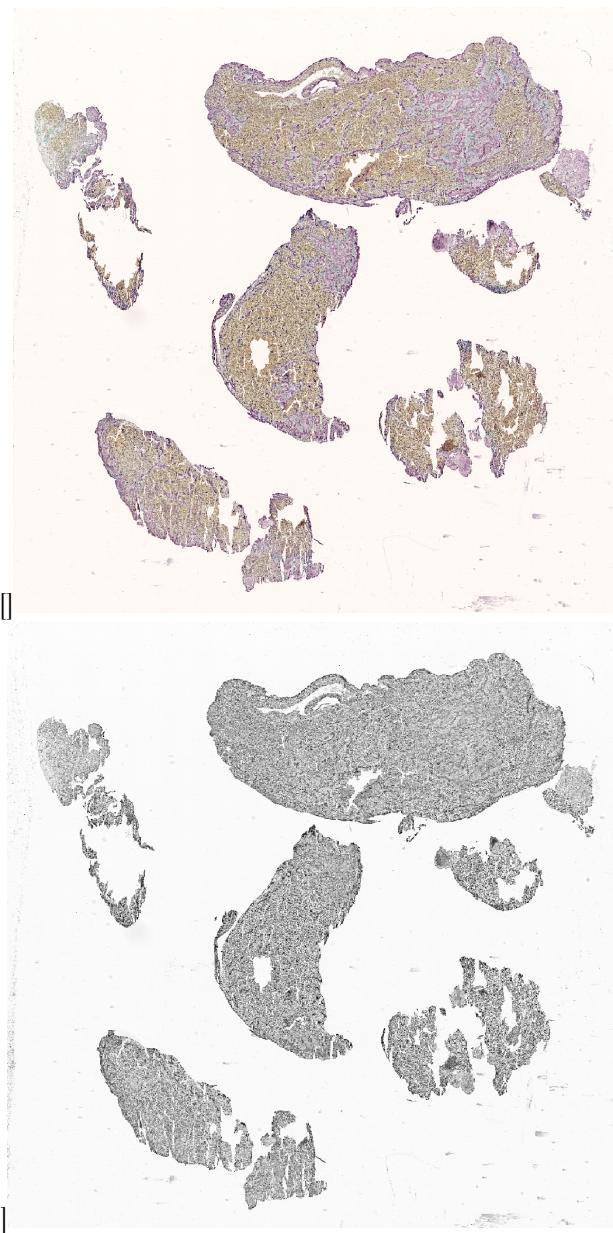
Om dit probleem op te lossen hebben we gebruik gemaakt van matlab. Het is dan ook een meerwaarde dat we in het eerste semester 'beginselen van programmeren' gehad hebben waardoor we sneller vertrouwd geraken met deze programmeertaal. Daarnaast leren we in 'nummerieke wiskunde' ook werken met matlab. Statistische analyseren we voor het analyseren van de verschillende kleuren.

## 7 Besluit

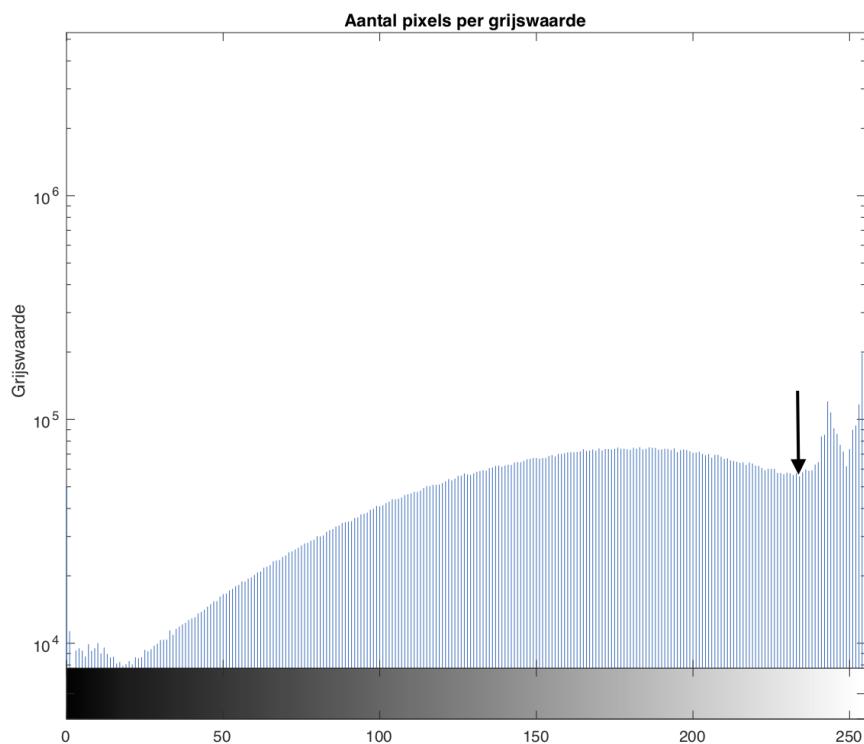
Momenteel kunnen we besluiten dat we goed aan het vorderen zijn. Zoals eerder vermeld kunnen we de foto automatisch bijsnijden en de achtergrond ruisvrij maken. Onze focus ligt nu voornamelijk bij de implementatie van de kleurendetectie. Eénmaal deze ook op punt staat kunnen we beide programma's in de app steken en deze optimaliseren met de nadruk op gebruiksvriendelijkheid naar de klant toe. Verder houden we de klant op de hoogte over onze vooruitgang via onder andere dit verslag.



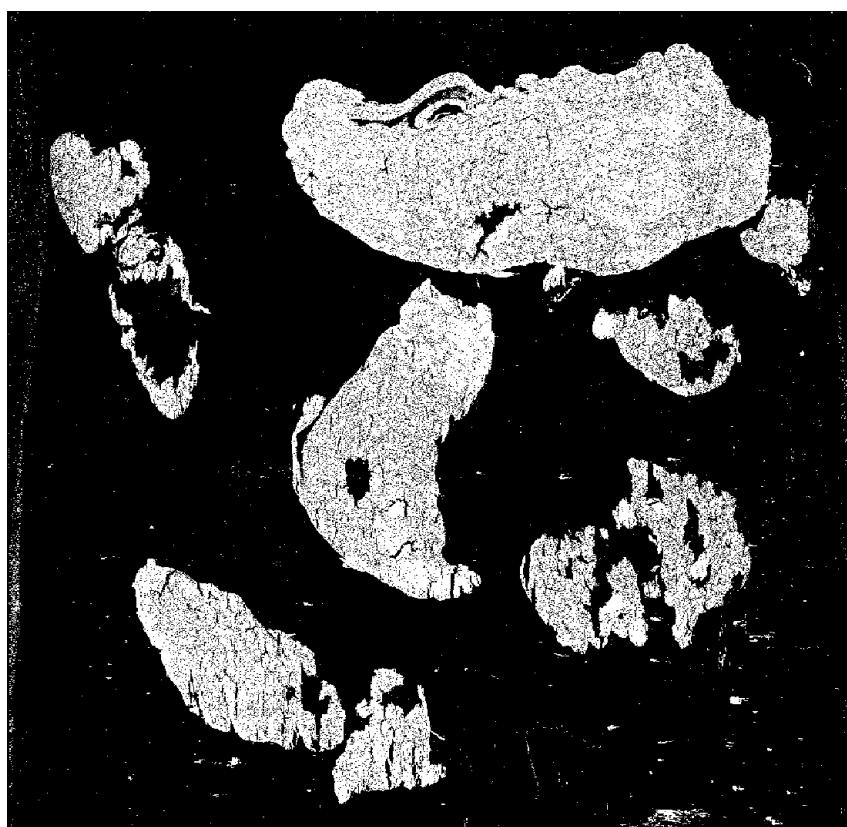
Figuur 1: In de rechterbovencirkel zien we luchtbellen en in de onderste cirkel zien we een verkleuring die geen deel uitmaakt van een bloedklonter.



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



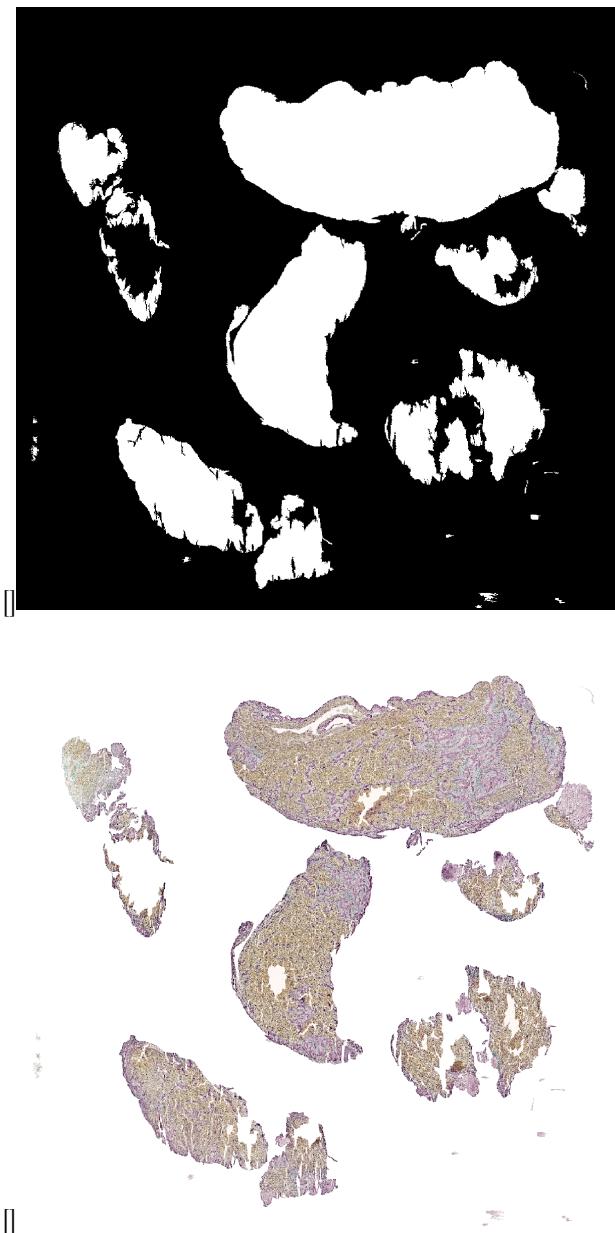
Figuur 3: Histogram van het aantal pixels gegroepeerd per grijswaarde. We zien duidelijk een lokaal minimum rond de waarde 230. Opmerking: we maken hier gebruik van een logaritmische y-as.



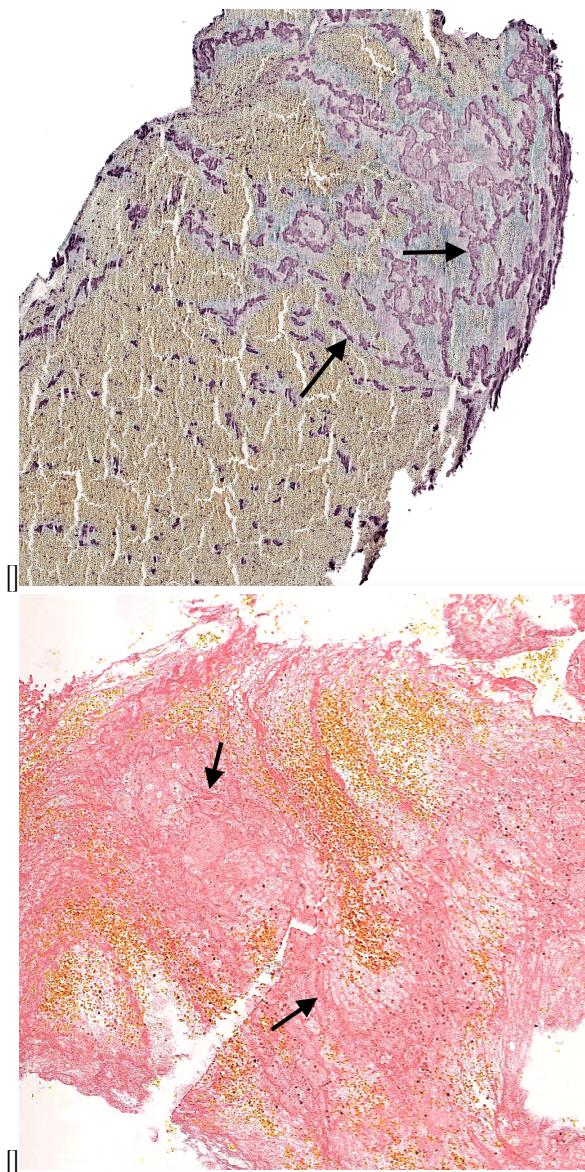
Figuur 4: Op deze binaire representatie zijn de bloedklonters duidelijk te zien



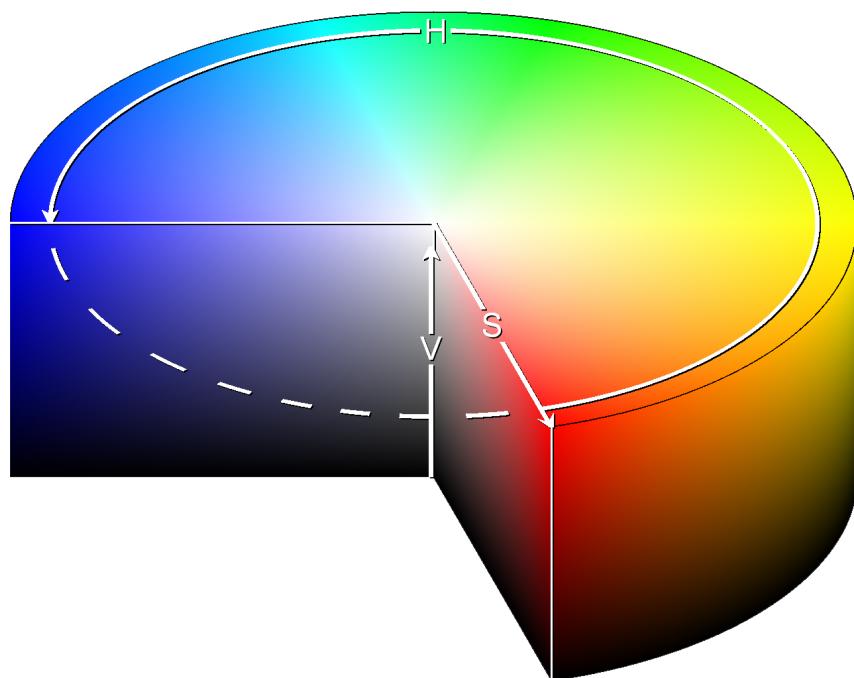
Figuur 5: Alle gaatjes zijn nu opgevuld, waardoor we de grootte van de bloedklonters kunnen berekenen



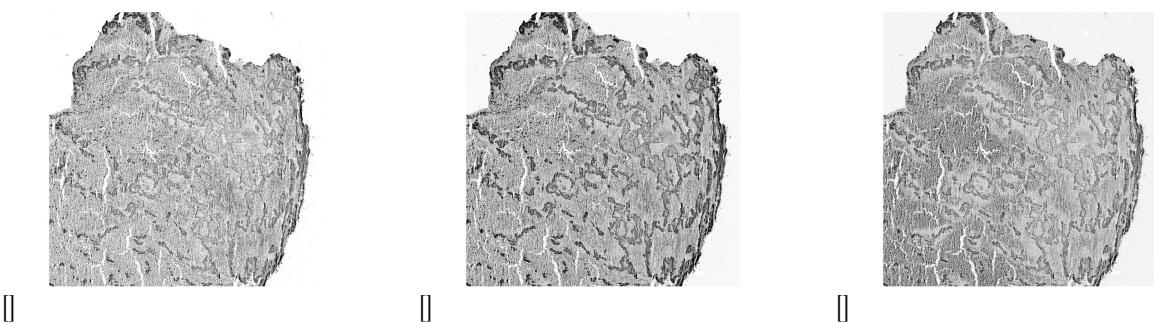
Figuur 6: Alle ruis is verwijderd, dit is een foto waarmee we kunnen werken



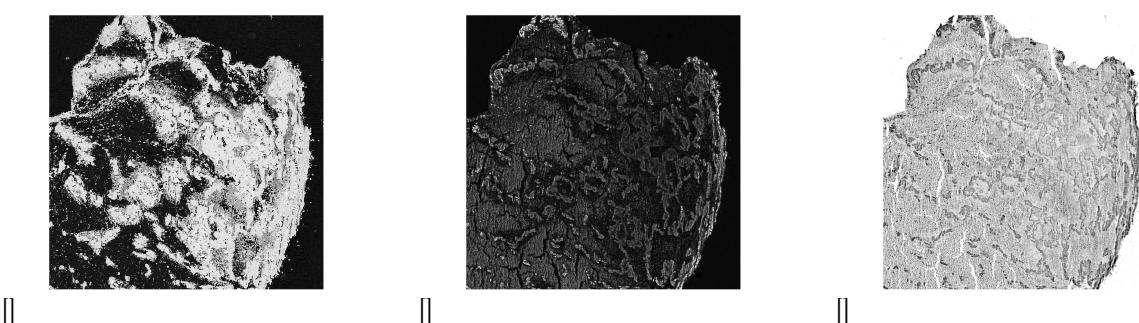
Figuur 7: Illustratie van de twee soorten indicator (respectievelijk paars en donkerroze) die gedetecteerd moeten worden. We zien duidelijk dat het detecteren op afbeelding (a) eenvoudiger zal zijn dan op afbeelding (b)



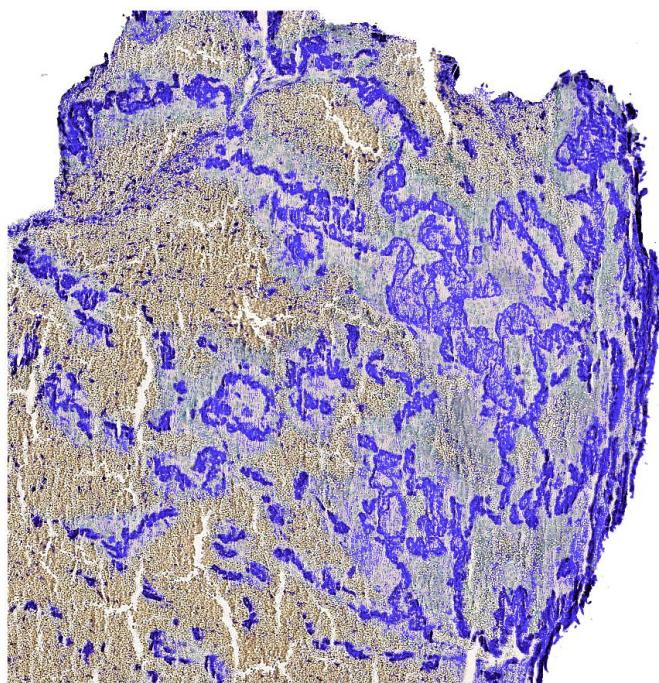
Figuur 8: Grafische voorstelling van het HSV (Hue, Saturation, Value) kleurmodel



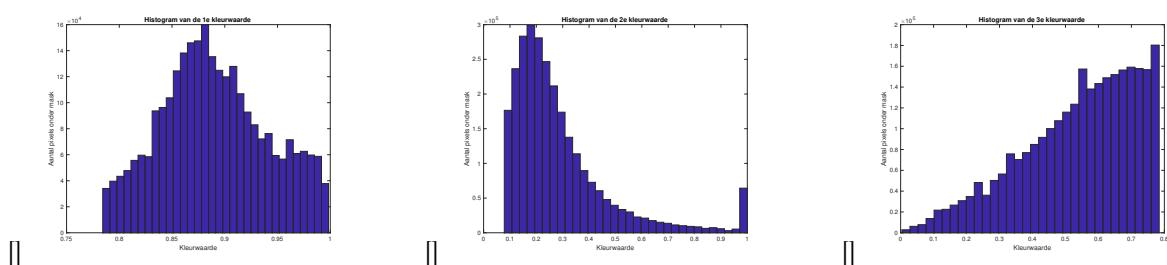
Figuur 9: Illustratie van respectievelijk de rode, groene en blauwe kleurwaarden (RGB). Hierbij komt wit overeen met de maximumwaarde en zwart met de minimumwaarde van die kleur.



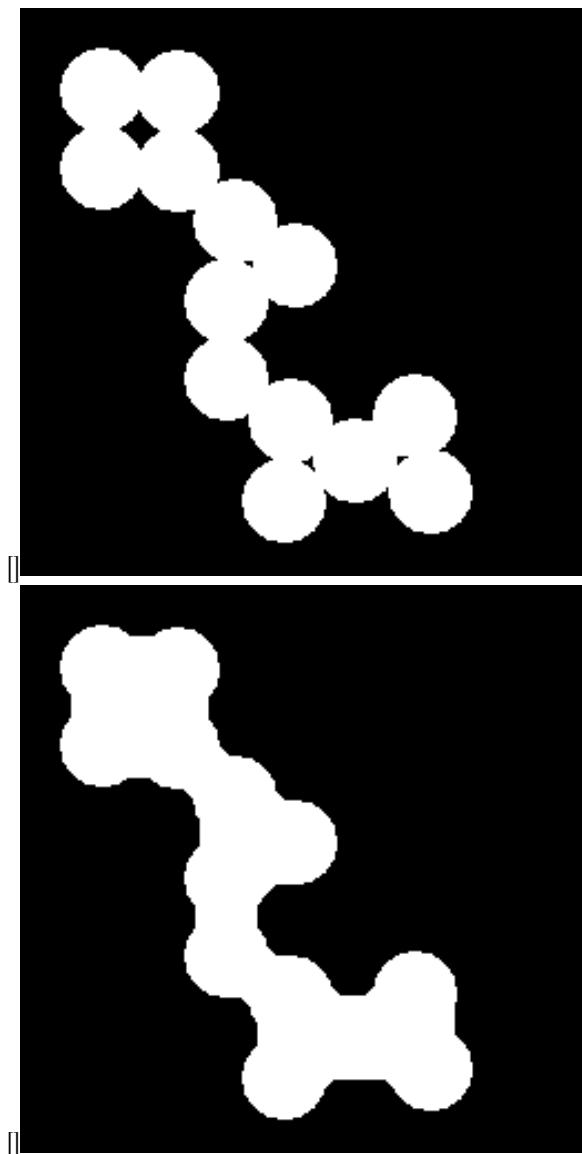
Figuur 10: Illustratie van respectievelijk de hue, saturation en value kleurwaarden (HSV). Hierbij komt wit overeen met de maximumwaarde en zwart met de minimumwaarde van die kleurwaarde.



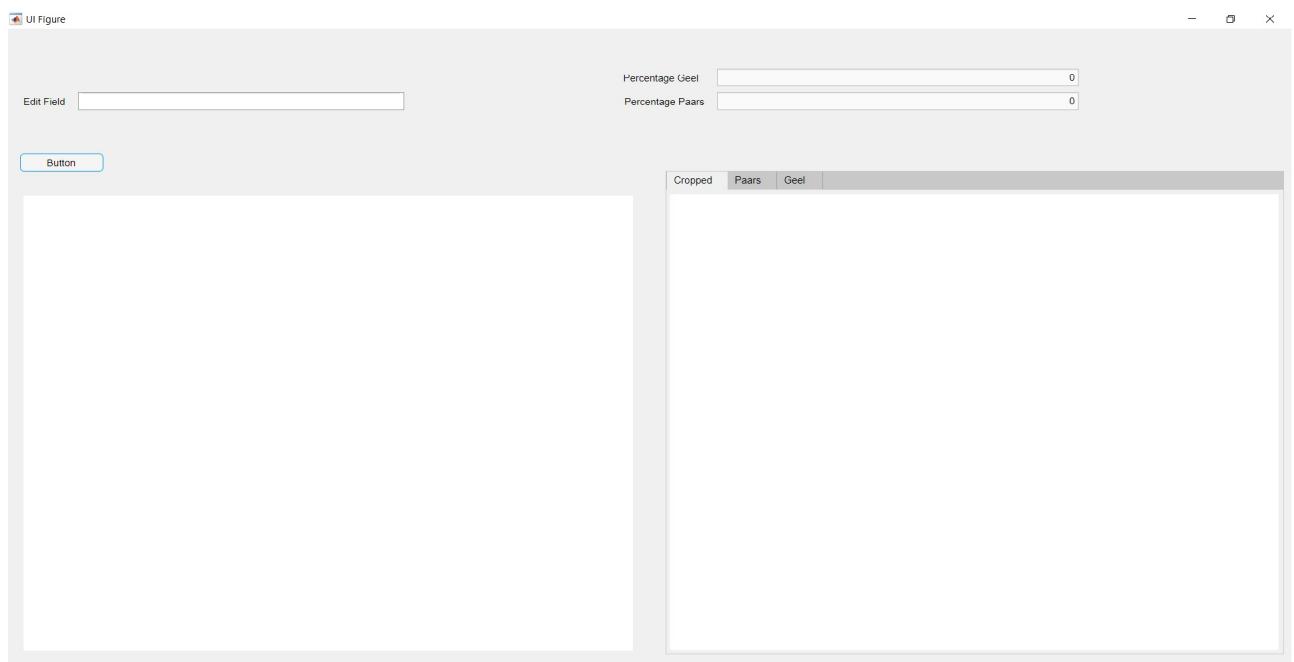
Figuur 11: We hebben alle indicatoren met een blauwachtige kleur aangeduid. De algemene threshold selecteert alle indicatorpixels, maar jammer genoeg ook enkele verkeerde (voornamelijk rechtsonder).



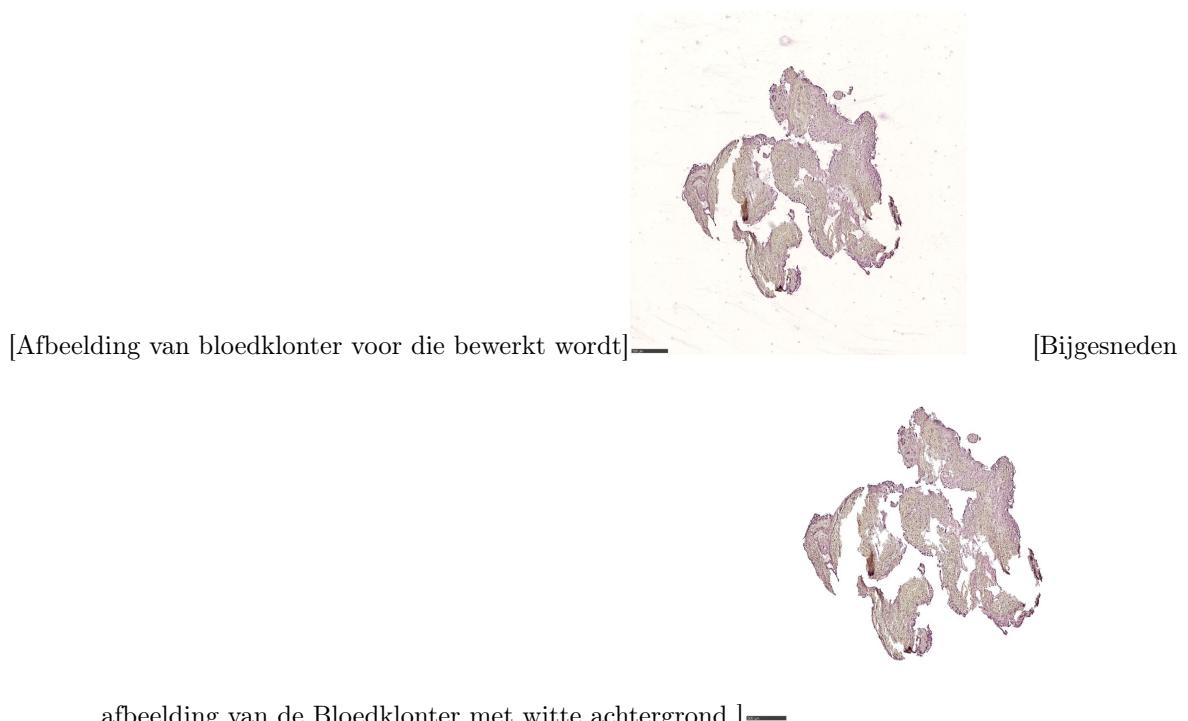
Figuur 12: Histogrammen van de HSV kleurwaarden. We zien duidelijk het verschil in frequenties



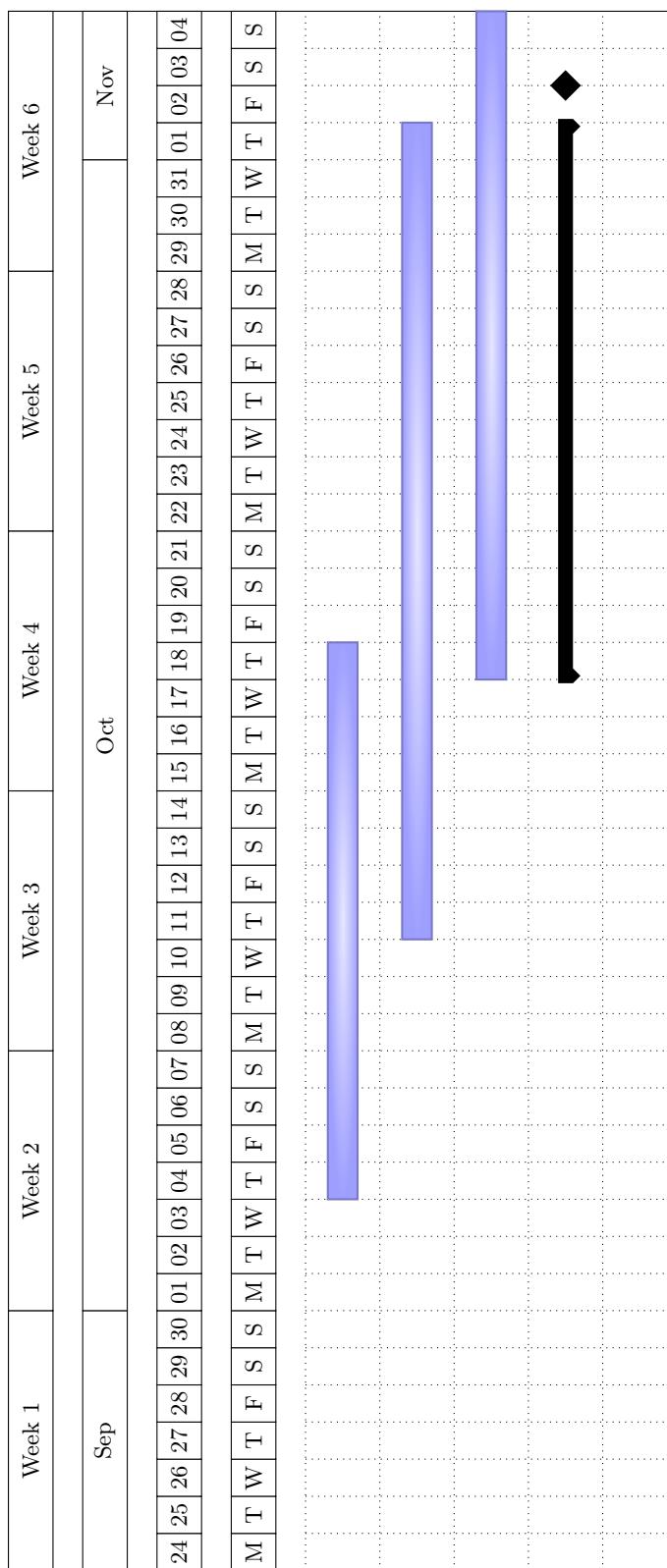
Figuur 13: Illustratie waarbij de witte cirkels uit afbeelding (a) morfologisch gesloten worden in afbeelding (b)



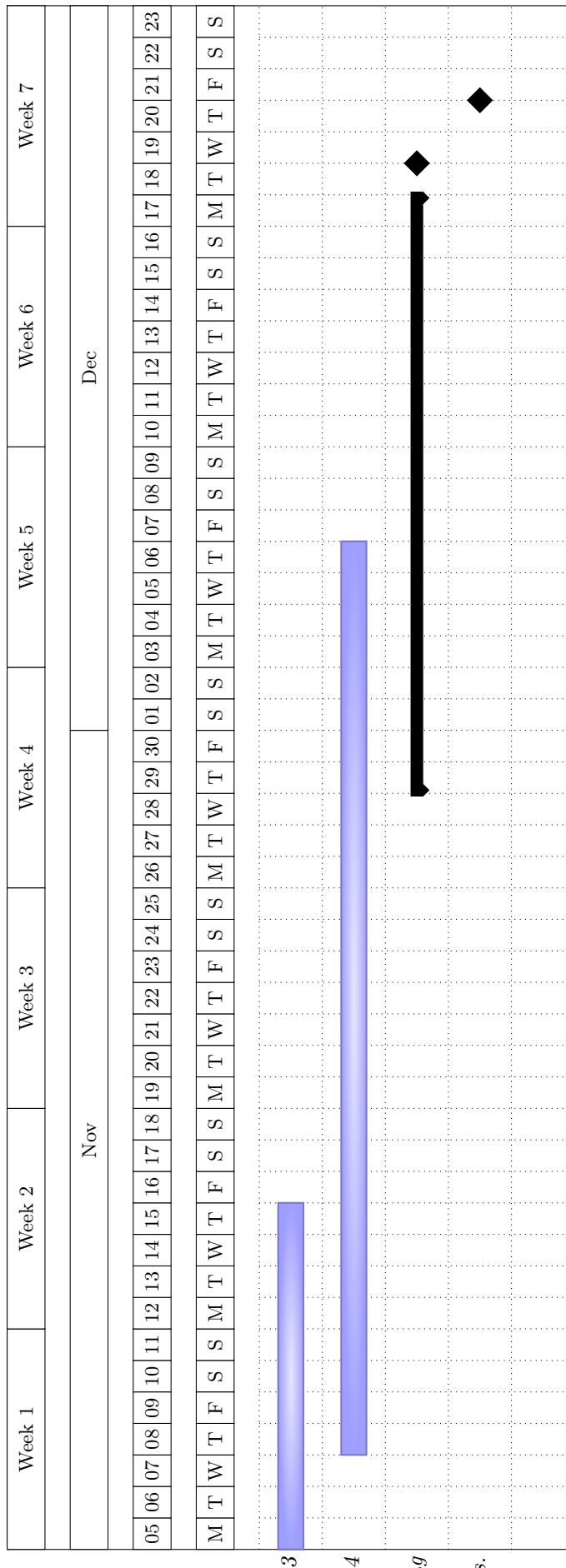
Figuur 14: Voorstelling van de app



Figuur 15: Illustratie van hoe we de afbeelding gebruiksbaar maken voor de analyse ervan



*T.T. verslag*



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

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