

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 18, 2018

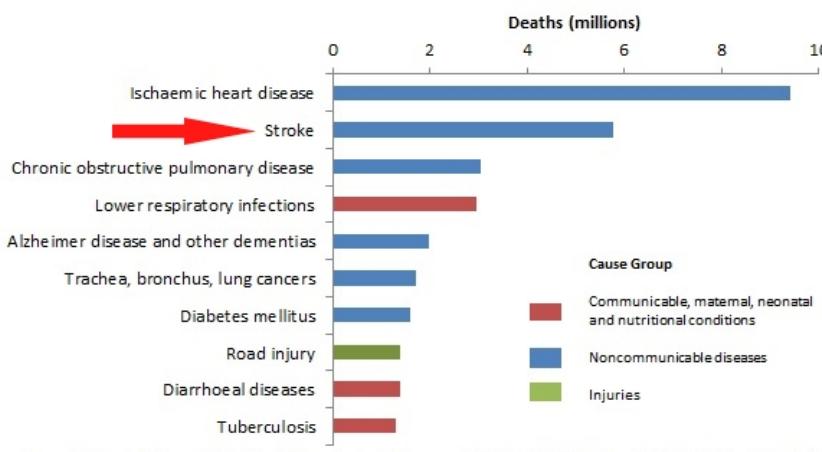
Marthe Böting, Robin Bruneel en Toon Ingelaere

### Inleiding

Beroertes zijn in onze westerse samenleving de derde grootste doodsoorzaak na hartinfarcten en kanker. Zoals we zien op Figuur 1 kapen ze op wereldvlak zelfs de tweede plaats weg [1]. In België komen er gemiddeld 25 000 beroertes per jaar voor. In 15% van deze gevallen overlijdt de patiënt. De overige 85% heeft na een beroerte vaak last van blijvende functiebeperkingen zoals cognitieve-, emotionele of gedragsproblemen. Een ischemische beroerte ontstaat doordat een bloedklonter emboliseert<sup>1</sup> en in één van de hersenbloedvaten vast komt te zitten. Deze bloedklonter moet verwijderd worden opdat de patiënt geen hersenschade oploopt. De huidige therapie focust daarom op het snel en efficiënt verwijderen van de bloedklonter uit het hersenbloedvat. In eerste instantie kan men proberen om met het geneesmiddel, weefsel plasminogeen activator, de klonter op te lossen. Dit geneesmiddel moet gegeven worden binnen de eerste 4,5 uur na het optreden van de symptomen. Van alle patiënten die het geneesmiddel toegediend krijgen, lost de klonter slechts in 1/3 van de gevallen op. Daardoor moet men gebruik maken van trombectomie om de klonter er manueel uit te halen.

Om huidige therapeutische opties te verbeteren, gebruikt men deze bloedklonters voor verder onderzoek. Hierbij wordt de samenstelling van de klonters geanalyseerd. Hiervoor worden afbeeldingen gemaakt van bloedklonter die voor welbepaalde componenten gekleurd zijn (bv. rode bloedcellen, witte bloedcellen en bloedplaatjes) en vervolgens geanalyseerd via kleur-gebaseerde segmentatie analyse. Deze analyses zijn echter erg tijdrovend. Er is ons dan ook gevraagd om een gebruiksvriendelijke app te ontwikkelen die de analyse van de afbeeldingen kan automatiseren. In dit verslag gaan we eerst in op wat de klant specifiek van ons verwacht en aan welke specificaties ons ontwerp moet voldoen. Daarna bespreken we ons design en lichten we het ook toe en ten slotte wordt er nog een blik geworpen naar de vakken uit eerste drie semesters die ons hierbij geholpen hebben.

### Top 10 global causes of deaths, 2016



Figuur 1: Statistieken van de *World Health Organization* van 2016 waarin te zien is dat wereldwijd beroertes (*strokes*) de tweede meest frequente doodsoorzaak is.

<sup>1</sup>Een bloedklonter emboliseert een bloedvat wanneer deze afgesloten wordt.



## Inhoudsopgave

<b>1 Klantenvereisten</b>	<b>4</b>
<b>2 Ontwerpspecificatie</b>	<b>4</b>
<b>3 Onze oplossing</b>	<b>5</b>
3.1 Achtergrondverwijdering . . . . .	5
3.1.1 Bepalen van de beste threshold . . . . .	5
3.1.2 Ruisfilter . . . . .	7
3.1.3 Uitzonderingen . . . . .	8
3.2 Lokalisatie van de indicator . . . . .	9
3.2.1 Het HSV kleurmodel . . . . .	10
3.2.2 Een algemene threshold . . . . .	11
3.2.3 Optimalisatie van de threshold . . . . .	12
3.2.4 Problemen . . . . .	14
3.3 De gebruiksvriendelijke applicatie . . . . .	14
<b>4 Verantwoordelijkheden en taakverdeling</b>	<b>15</b>
<b>5 Integratie van vakken</b>	<b>15</b>
<b>6 Besluit</b>	<b>15</b>

## 1 Klantenvereisten

De klant verwacht een gebruiksvriendelijke app die de afbeeldingen automatisch verwerkt. Een afbeelding moet automatisch ingeladen worden, bijgesneden worden en de achtergrond verwijderd worden. Daarnaast moet de samenstelling van de bloedklonter geanalyseerd worden aan de hand van de huidige indicator.

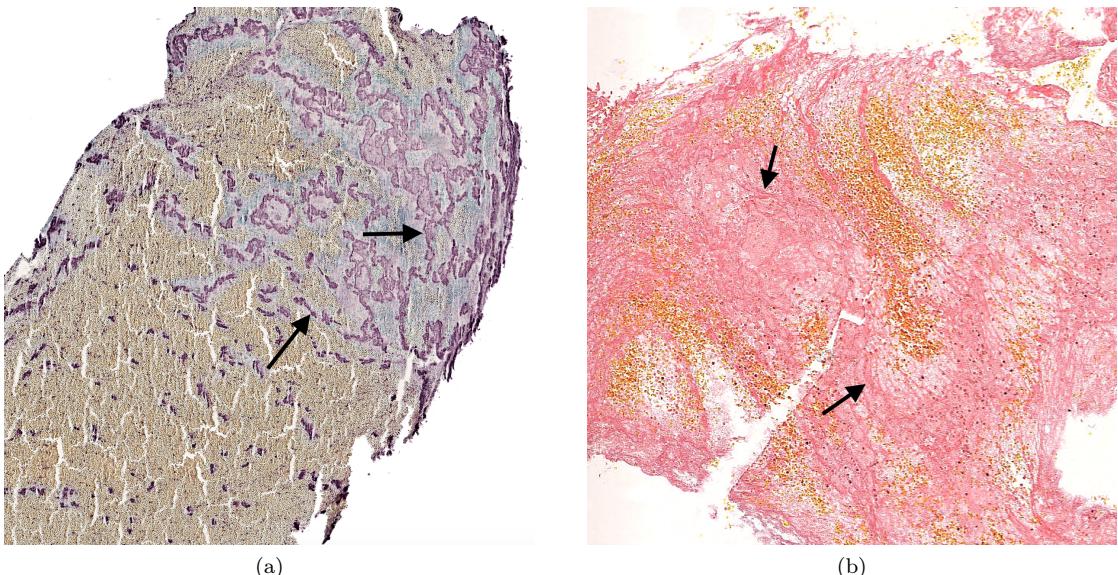
## 2 Ontwerpspecificatie

De gebruiker van de app wil dat een afbeelding van een bloedklonter automatisch bewerkt en geanalyseerd wordt.

Het bewerken van de afbeelding houdt twee dingen in. Eerst en vooral moet de afbeelding zodanig bijgesneden worden dat de volledige bloedklonter erop staat. Hierbij mogen we de randen echter niet te breed nemen, aangezien er dan nuttige geheugenruimte<sup>2</sup> verspild wordt. Naast het bijsnijden, moet ook de achtergrond verwijderd worden. Dit betekent dat alle pixels die niet tot de bloedklonter behoren wit gekleurd worden. Indien dit niet goed gebeurt, kunnen de resultaten van de kleurenanalyse namelijk vertekend zijn.

In de kleurenanalyse moet het percentage van de met indicator gekleurde pixels geteld worden. Voor dit project moeten we slechts twee soorten kleuringen analyseren. Een voorbeeld van deze is te zien in Figuur 2. In beide gevallen moet de app op een accurate manier onderscheid kunnen maken tussen de eiwitten die gedetecteerd moeten worden en de rest van de bloedklonter.

Dit alles moet verwerkt worden in een visuele en gebruiksvriendelijke app. Dit wil zeggen dat de app makkelijk te installeren en te gebruiken is. De gebruiker moet ook een overzicht van de verschillende afbeeldingen van de bloedklonter kunnen terugroepen. Dit overzicht bestaat uit de originele afbeelding, de afbeelding zonder achtergrond en de afbeelding waarbij de indicatorpixels zijn aangeduid. Zo kunnen mogelijke fouten snel gedetecteerd worden. Daarnaast moet deze app nog wat extra functionaliteiten uitvoeren zoals het opslaan van deze afbeeldingen en het manueel bewerken van het bekomen resultaat.



Figuur 2: 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).

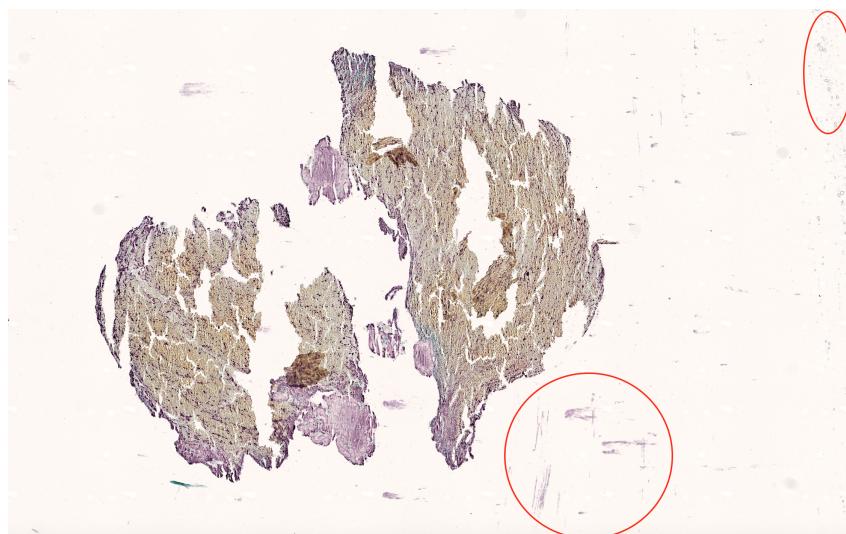
<sup>2</sup>We werken namelijk met foto's van de orde van 200MB, het sparen van pixels op ons resultaat is uiterst voordelig

### 3 Onze oplossing

In dit hoofdstuk trachten we een oplossing te vinden voor ons probleem. We willen namelijk het percentage indicatorpixels in een willekeurige afbeelding kunnen berekenen. Het hoofdstuk is opgedeeld in verschillende deelproblemen: het verwijderen van de achtergrond, het lokaliseren van de indicator en de gebruiksvriendelijke app. Alle resultaten zijn bekomen met behulp van de programmeertaal MATLAB.

#### 3.1 Achtergrondverwijdering

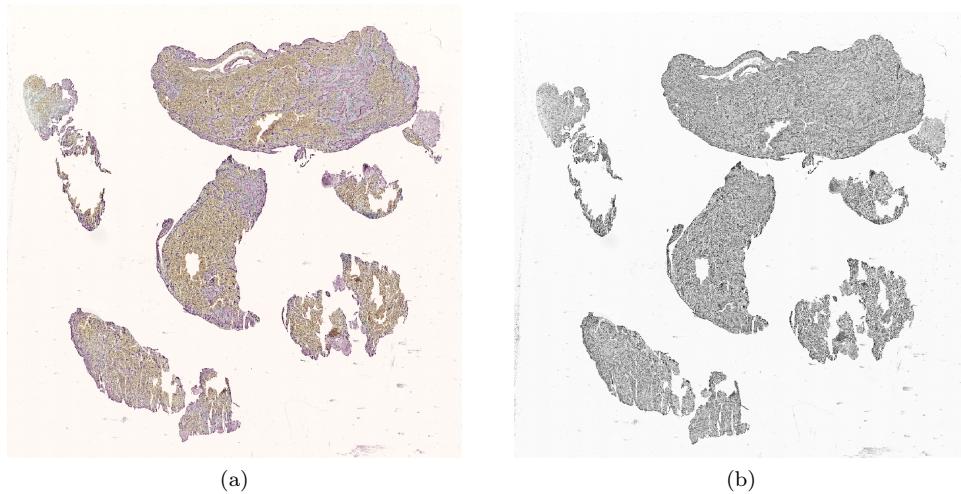
Het eerste deelprobleem is het verwijderen van de achtergrond. Op de afbeeldingen is er heel wat vuilheid te vinden. Voorbeelden hiervan zijn luchtbellen of kleine verkleuringen in de achtergrond zoals men ziet op de Figuur 3. Hieronder beschrijven we verschillende operaties om de grootste kloners te lokaliseren en alles wat geen kloner is uit de afbeelding te verwijderen. Dit leidt in feite tot een ruisvrije afbeelding.



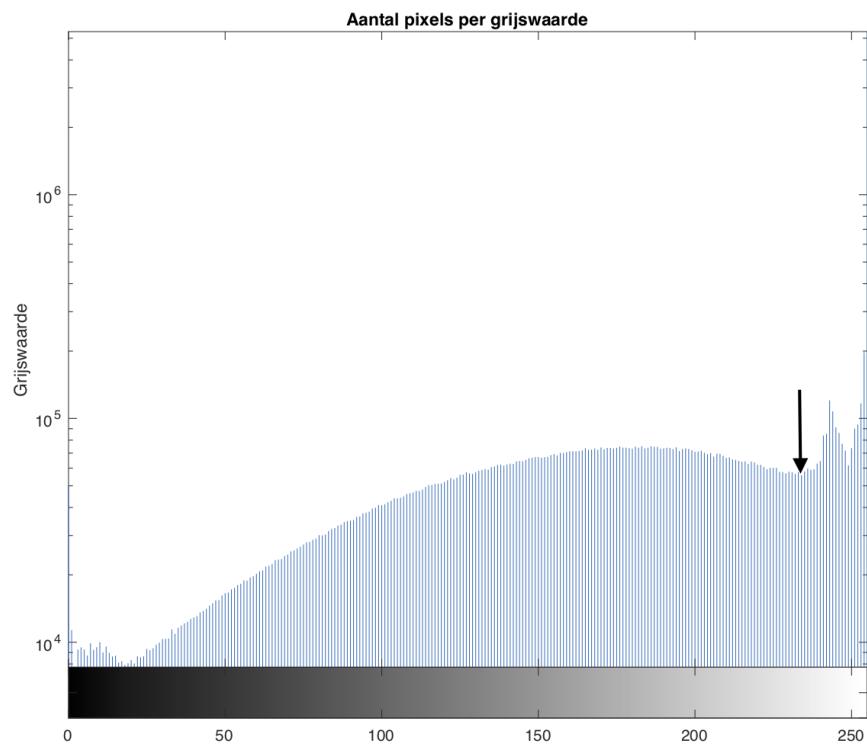
Figuur 3: In de rechterbovenhoek zien we luchtbellen en in de onderste cirkel zien we een verkleuring die geen deel uitmaakt van een bloedklont.

##### 3.1.1 Bepalen van de beste threshold

Het onderscheid tussen de achtergrond (eerder wit) en de bloedklont (eerder grijs) is makkelijker te zien op de grijswaarden van de afbeelding, zie Figuur 4. Eén waarde is nu voldoende om het onderscheid te maken. Een histogram van deze grijswaarden leert ons dat er een duidelijk dipje in de frequenties tussen het wit en het grijs te zien is, zie Figuur 5. 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. Wanneer we deze *threshold* toepassen, bekomen we de binaire Figuur 6. Dit lijkt de kloners in de afbeelding nagenoeg goed te detecteren.



Figuur 4: Illustratie van de originele foto (a) en deze omgezet in grijswaarden (b).



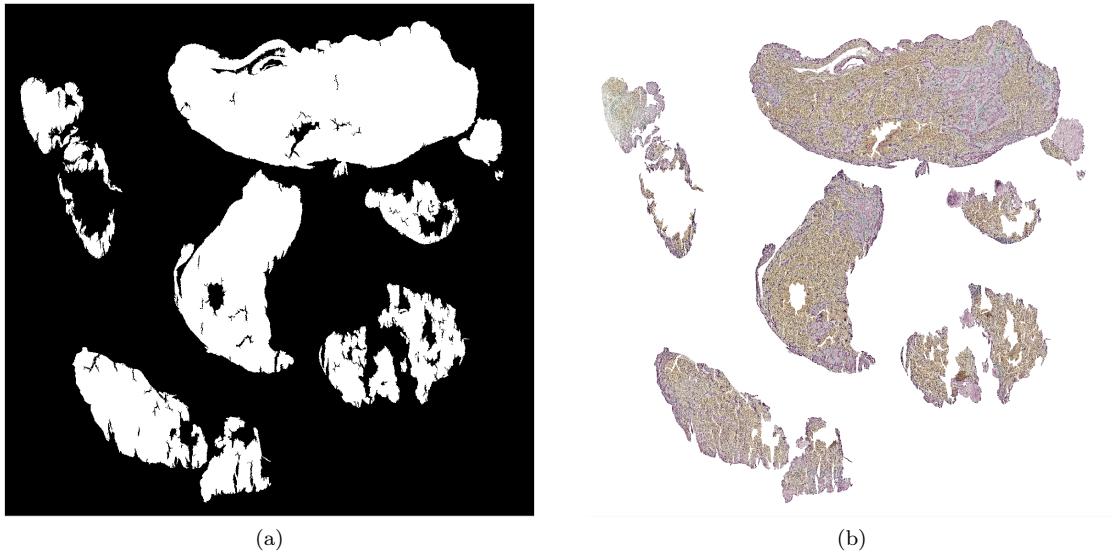
Figuur 5: Histogram van het aantal pixels gegroepeerd per grijswaarde. We zien duidelijk een lokaal minimum rond de waarde 230. Opmerking: we maken hier gebruik van een logaritmische y-as.



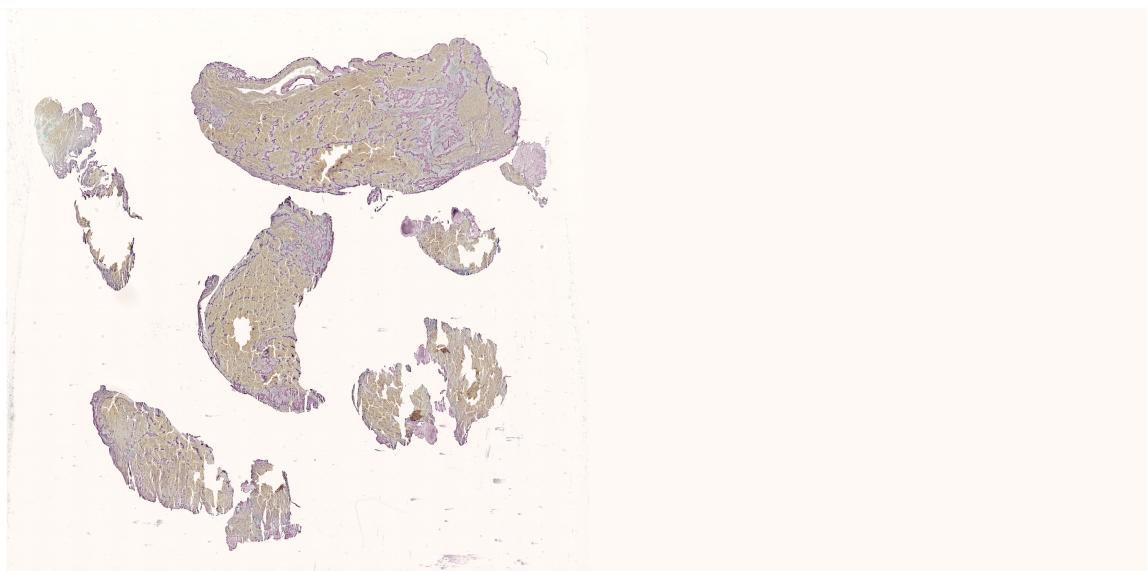
Figuur 6: Op deze binaire representatie zijn de bloedklonters duidelijk te zien.

### 3.1.2 Ruisfilter

Deze binaire representatie bevat heel wat holtes en ruis. In dit onderdeel proberen we dit probleem op te lossen. Hiervoor hebben we een simpele methode bedacht door enerzijds alle kleine groepen witte pixels te vervangen door zwarte pixels, zodat alleenstaande pixels (voornamelijk ruis) verdwijnen. Daarnaast vervangen we ook alle kleine groepen zwarte pixels door witte pixels, hierdoor worden daarenboven alle holtes in de bloedklonters opgevuld. Het resultaat van deze operatie is te zien op Figuur 7 (a). We bekomen nu een zogenaamde '*mask*' die toegepast kan worden op de originele afbeelding. Hierbij worden alle zwarte pixels op de *mask* vervangen door witte pixels op de originele afbeelding. Alle witte pixels op de *mask* behouden hun kleur op de originele afbeelding. Daarnaast wordt deze afbeelding nog bijgesneden om kostbare geheugenruimte te sparen, de afbeelding is te zien in Figuur 7 (b). In dit deelprobleem werd echter met een reeds bijgesneden bloedklonter gewerkt om de afbeeldingen duidelijk te houden. Een volledig onbewerkte afbeelding is echter te zien in Figuur 8



Figuur 7: Alle ruis is verwijderd, dit is een foto die verder verwerkt kan worden.



Figuur 8: Dit is een volledig onbewerkte afbeelding die we in 7 (b) hebben bijgesneden en waarvan de achtergrond verwijderd is.

### 3.1.3 Uitzonderingen

Naast de gewone bloedklonters zijn er ook nog enkele uitzonderingen. Sommige klonters hebben namelijk geen rode bloedcellen waardoor ze zeer wit zijn. Dit zien we ook op Figuur 9. Het probleem is dat we hier geen optimale *threshold* kunnen berekenen zoals eerder uitgelegd. Er is namelijk een zeer kleine afstand tussen de kleur van de bloedklonter en de feitelijke achtergrond. Vandaar dat we in de app een slider verwerkt hebben om desnoeds zelf die *threshold* aan te aanduiden. Wanneer de optimalisaties van de simpele binaire *mask* daarna op deze *threshold* toegepasten, bekomen we Figuur 10. Het is duidelijk te zien dat dit algoritme zeer krachtig is in het detecteren van bloedklonters.



Figuur 9: Een afbeelding van een bloedklonten zonder rode bloedcellen.



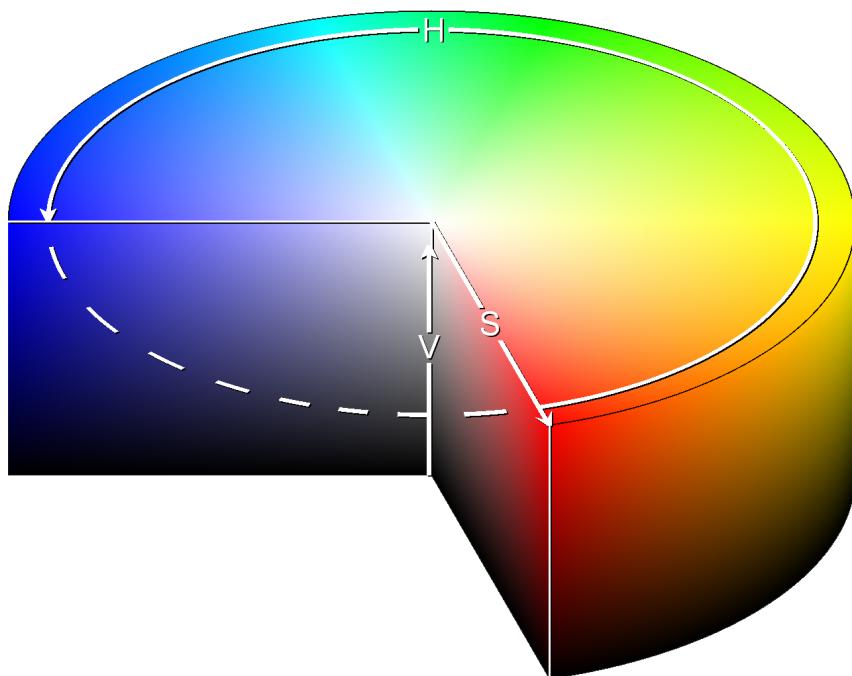
Figuur 10: Dit is de binaire *mask* die we bekomen na het toepassen van de *threshold* en de optimalisatie van deze. De klonters worden nu met zekere precisie aangeduid.

### 3.2 Lokalisatie van de indicator

Het tweede deel van ons project is de indicatoren lokaliseren en kwantificeren. Ons project moet in staat zijn twee specifieke soorten van indicatoren te onderscheiden. Van deze is een voorbeeld op Figuur 2 te zien. Omdat het kleurverschil tussen wat aangeduid worden (indicator) en wat niet (achtergrond) niet

altijd even groot is, vormen we het oorspronkelijke *RGB* kleurmodel<sup>3</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 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 11.

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 *CMYK* kleurmodel die gelijkaardige eigenschappen heeft en desnoods ook gebruikt kan worden. Eenmaal we een duidelijk onderscheid tussen de indicator en de achtergrond gemaakt hebben, berekenen we eenvoudig het percentage indicator als de verhouden van het aantal bloedklonter- en indicatorpixels. Ons stappenplan wordt in de volgende hoofdstukken besproken en wordt toegepast op de afbeelding uit Figuur 2 (a). Maar kan evenwel gebruikt worden op de andere kleuring met andere beginwaarden.



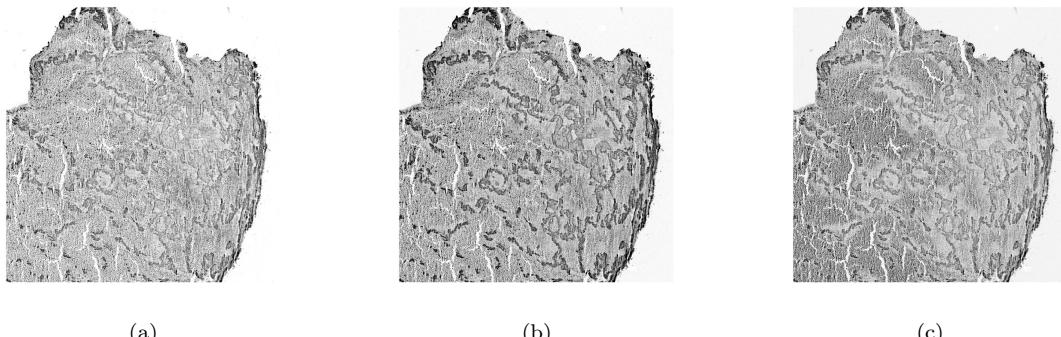
Figuur 11: Grafische voorstelling van het *HSV* (*Hue*, *Saturation*, *Value*) kleurmodel

### 3.2.1 Het *HSV* kleurmodel

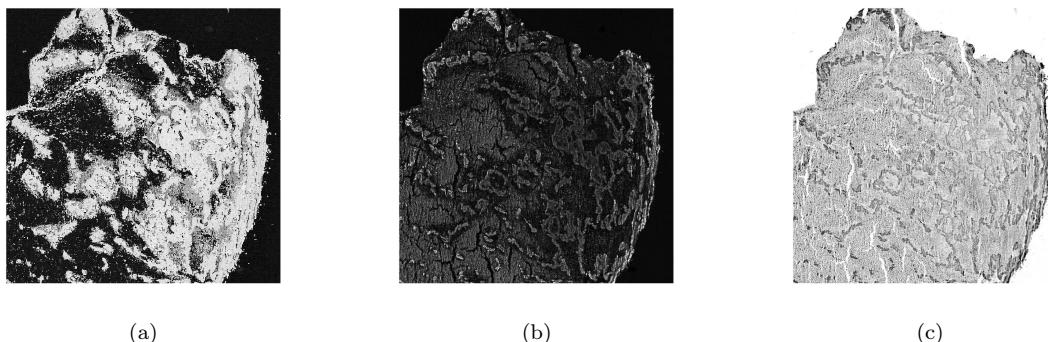
Zoals reeds vermeld, beginnen we met een transformatie naar het *HSV* kleurmodel. Bijgevolg kunnen we de twee voorstellingen vergelijken. We hebben het model telkens ontbonden in de drie kleurwaarden, waarbij zwart de laagste waarde voor die kleur voorstelt en wit de hoogste. De resultaten zijn te zien in de Figuren 12 en 13. Het verschil tussen de twee kleurmodellen is duidelijk te zien. Afbeelding (a) en (b) uit Figuur 13 lijken namelijk een iets agressiever onderscheid te maken.

---

<sup>3</sup>Het *RGB* kleurmodel is een voorstelling waarbij ieder kleur voorgesteld wordt door een waarde van de drie basiskleuren (rood, groen en blauw)



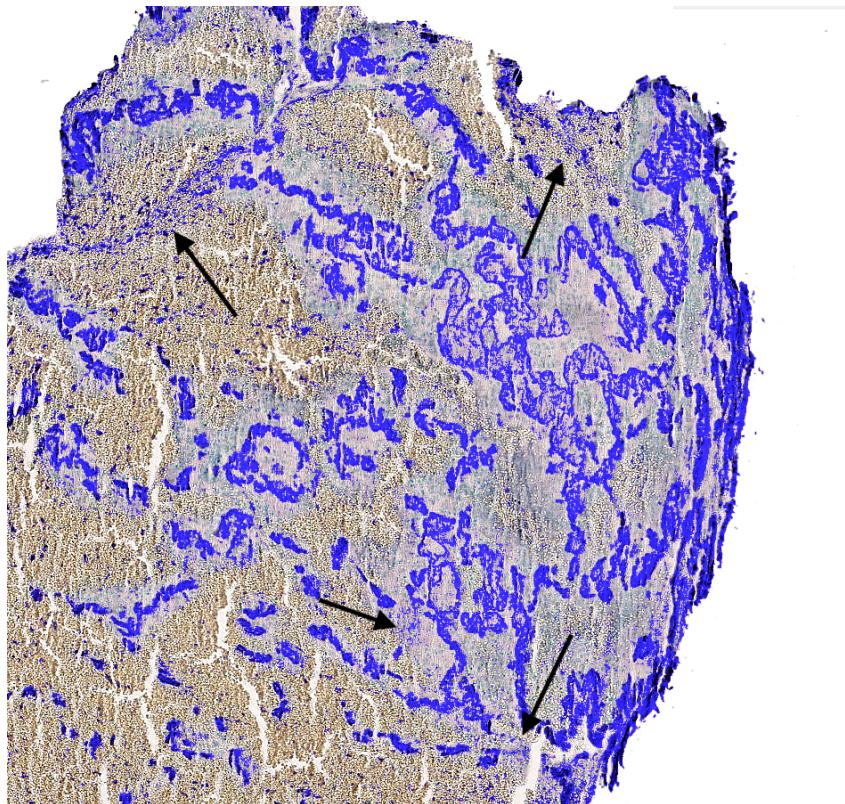
Figuur 12: Illustratie van respectievelijk de rode, groene en blauwe kleurwaarden (*RGB*). Hierbij komt wit overeen met de maximumwaarde en zwart met de minimumwaarde van die kleur.



Figuur 13: 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 een filter bepalen voor de indicatorpixels. Het probleem is echter dat een goede filter voor de ene afbeelding niet altijd een goede filter voor de andere is. Daarom hebben we voor iedere afbeelding afzonderlijk manueel een '*threshold*' bepaald en deze achteraf met elkaar vergeleken. Het resultaat is bijgevolg een vrij algemene *threshold* die alle indicatorpixels met zekerheid aanduidt, zoals te zien is op Figuur 14. Het enigste probleem is dat bepaalde pixels verkeerd aangeduid worden. Het aantal is weliswaar niet zo groot, maar aangezien ze in elke afbeelding voorkomen zullen we dit trachten te omzeilen.



Figuur 14: We hebben alle indicatoren met een blauwe kleur aangeduid. De algemene *threshold* selecteert alle indicatorpixels, maar jammer genoeg ook enkele verkeerde, deze zijn aangeduid met een pijl.

### 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 correcte *threshold* waardoor we een statistische analyse van wat onder de bijkomende *mask* ligt, kunnen uitvoeren. Hiervoor stellen we een histogram van de *HSV* kleurwaarden onder de *mask* op. Het levert ons een interessant resultaat dat te zien is in Figuur 15. We zien namelijk duidelijke verschillen in de frequenties van bepaalde kleurwaarden.

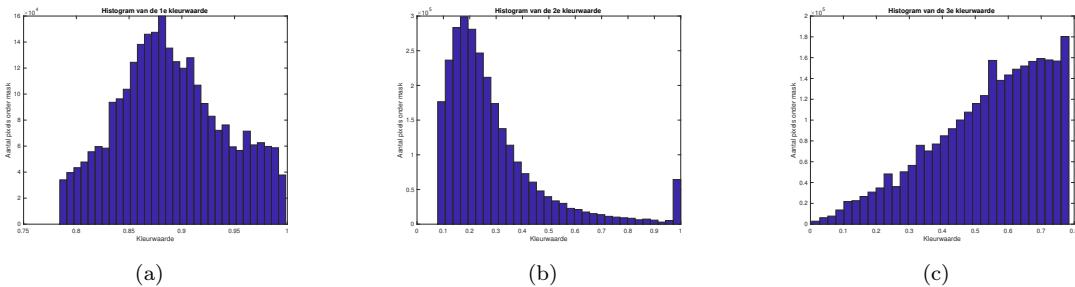
Het idee is om nu de verkeerde pixels via deze diagrammen er uit te filteren. Hiervoor doen we de aanname dat de verkeerde pixels essentieel in kleur verschillen van indicatorpixels. We weten ook uit Figuur 14 dat verkeerde pixels minder vaak voorkomen dan de juiste. De huidige *threshold* zou dus in feite manueel bijgestuurd kunnen worden, maar dit heeft geen zin. De grafieken van verschillende afbeelding hebben namelijk eenzelfde vorm, maar hun pieken liggen soms meer dan 5% verschoven. Daarom wordt op iedere afbeelding afzonderlijk een filter toegepast. In principe kunnen alle pixels met een frequentie onder een bepaalde grenswaarde geschrapt worden. Maar nee betere benadering is om het punt te zoeken, waar de frequentie van de pixels enorm begint toe te nemen. Indien de *threshold*-waarde naar de pieken toe geschoven wordt, worden eigenlijk grote ‘indicatoraders’ verwijderd. Dit ziet men op Figuur 16. Dit punt kan theoretisch benaderd worden als het maximum van de tweede afgeleide naar de kleurwaarde.

Wanneer deze filter toegepast wordt, krijgt men het resultaat dat te zien is op Figuur 17 (a). De zones die eerder verkeerd werden aangeduid zijn nu bijna volledig verdwenen.

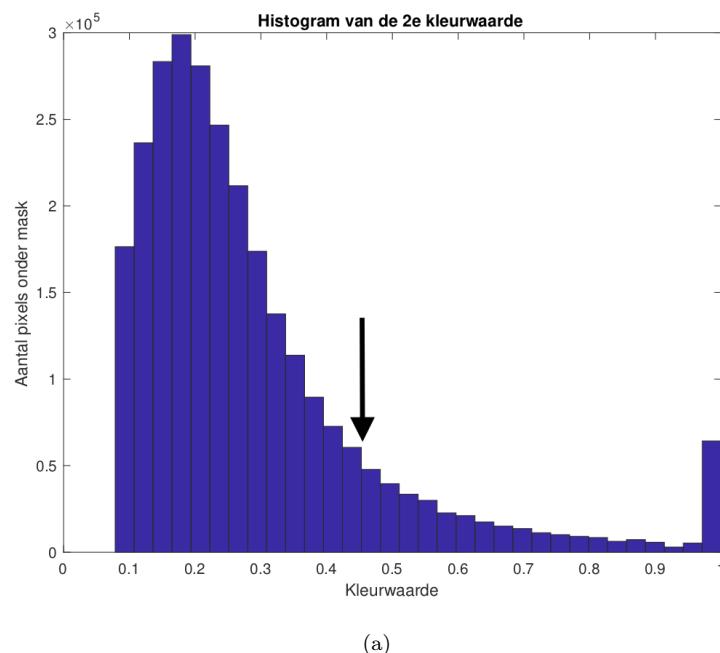
De zones die eerder verkeerd werden aangeduid zijn nu bijna volledig verdwenen.

Een extra stap die we tenslotte nog toepassen is het morfologisch sluiten van de pixels. Dit doen we door indicatorpixels die niet ver van elkaar gelegen zijn met elkaar te verbinden. Het is namelijk zo dat wanneer een pixel omringt is door indicatorpixels, de kans hoog is dat deze pixel ook een indicatorpixel is. Het resultaat is te zien op 17 (b).

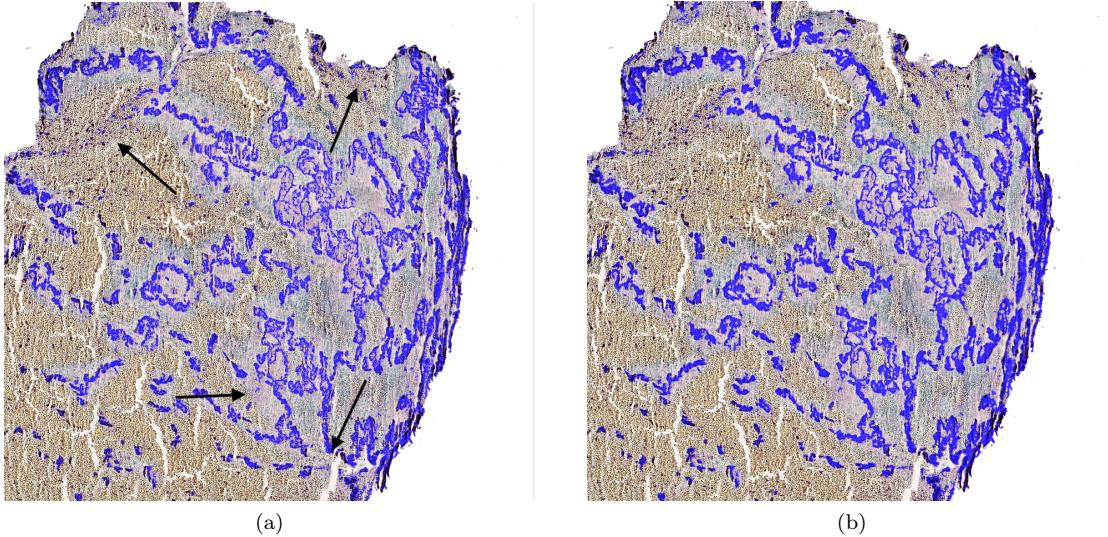
De laatste stap die nu nog rest is het optellen van alle aangeduide pixels en de bloedklonterpixels die gekend zijn na het verwijderen van de achtergrond.. Wanneer we dan de verhouding nemen, bekomen we het percentage indicator, wat net gevraagd werd.



Figuur 15: Histogrammen van de HSV kleurwaarden. We zien duidelijk het verschil in frequenties.



Figuur 16: Op de figuur is hebben we een punt op de curve aangeduid. Wanneer we dit punt naar links verschuiven, verwijderen we telkens grotere stukken indicator van onze afbeelding. Dit willen we net vermijden.



Figuur 17: Illustratie waarbij we in (a) een algoritmische filter hebben toegepast om mogelijke ruis te verwijderen. De pijltjes zijn de zones die weggefilterd zijn op Figuur 14. In afbeelding (b) hebben we deze mask nog eens morfologisch gesloten, waardoor we een duidelijke aanduiding van de indicatorpixels hebben.

### 3.2.4 Problemen

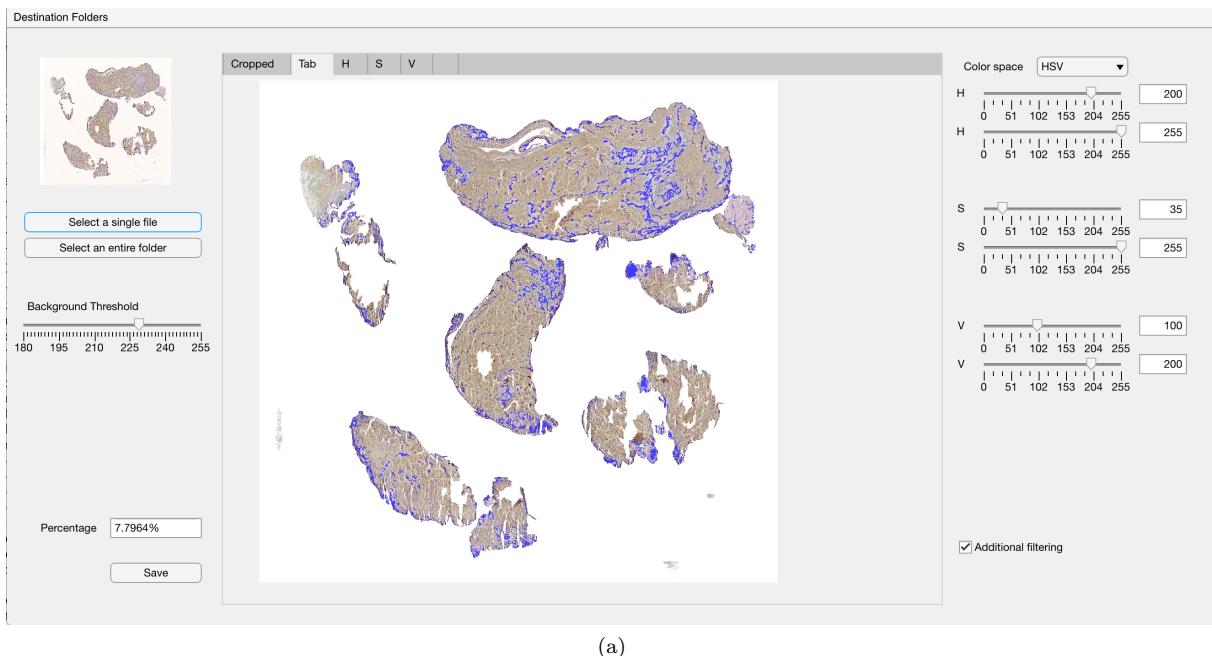
We kunnen nu wel met redelijke precisie de indicatoren in de afbeelding aanduiden, maar dit is niet even simpel voor elke afbeelding. Het probleem met deze kleuringen is namelijk dat er heel wat variatie in de afbeeldingen zit. De effectieve kleuring in het labo is afhankelijk van verschillende factoren zoals bijvoorbeeld temperatuur, druk, luchtvuchtigheid... Hierdoor kan deze kleuring van dag tot dag in intensiteit verschillen. Daarenboven zijn wij als student ingenieurswetenschappen ook niet getraind om rode bloedcellen te detecteren. Bijgevolg hebben we in onze app ervoor gezorgd dat onze filter met behulp van verschillende sliders altijd bijgestuurd kan worden door de gebruiker en dus toch het gewenste resultaat bekomen kan worden.

## 3.3 De gebruiksvriendelijke applicatie

Naast het verwerken van de kloners is het ook de bedoeling dat de afbeelding op een makkelijke manier verwerkt kan worden. Vandaar dat we een gebruiksvriendelijke app in MATLAB gemaakt hebben. Een screenshot van de grafische interface is te zien op Figuur 18. In dit hoofdstuk zullen we kort de functionaliteiten van deze app beschrijven.

De app kan op twee verschillende manieren gebruikt worden. In eerste instantie kan men een afbeelding ingeven die dan uiteindelijk het volledige verwerkingsproces van hoofdstuk 3.1 en 3.2 doorloopt. Men begint met een afbeelding en een folder om de tussenresultaten op te slaan te kiezen. In de eerste fase van het verwerkingsproces wordt de afbeelding bijgesneden zodat enkel de bloedklonten nog zichtbaar is. Deze bijgesneden afbeelding wordt dan getoond. Indien men niet blij is met de bekomen resultaten kan men manueel de *threshold* bijsturen. Vervolgens begint de kleurenanalyse, hier wordt de volledige afbeelding verwerkt om uiteindelijk een percentage indicator aan te duiden. Deze verwerkte afbeelding wordt dan ook getoond zoals te zien is op Figuur 18. Daarnaast wordt het percentage van de hoeveelheid aanwezige indicator weergegeven. Indien men niet tevreden is met het resultaat, kan men een transformatie naar een andere kleurruimte (HSV, CMYK of RGB) toepassen. Daarnaast kan men ook met verscheidene sliders een manuele selectie van de indicator doen. Men kan ten slotte nog opteren om de additionele filter uit hoofdstuk 3.2.3 uit te schakelen.

De app kan ook op een nog andere manier gebruikt worden. Men kan namelijk een volledige folder met afbeeldingen selecteren om de achtergrond van iedere afbeelding te verwijderen. Deze afbeeldingen worden bijgevolg opgeslagen in een eerder gedefinieerde folder.



(a)

Figuur 18: Afbeelding van de grafische interface van de app.

## 4 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 deze 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.

Helemaal achteraan dit verslag voegden we ook een Gantt-chart toe. Hierin staan enkele belangrijke deadlines vermeld en onze voorlopige planning. We hebben het project opgesplitst in vier verschillende taken die we elk ingepland hebben zoals afgebeeld. Ook de tijd die we rekenen voor het tussentijds - en het finaal verslag, zijn af te lezen in de chart.

## 5 Integratie van vakken

Om dit probleem op te lossen, hebben we gebruik gemaakt van MATLAB. Het is dan ook een meerwaarde dat we in het eerste semester het vak 'Beginselen van Programmeren' gehad hebben om sneller vertrouwd te raken met deze nieuwe programmeertaal. Daarnaast leren we ook in 'Nummerieke Wiskunde' werken met MATLAB. Om de verschillende thresholds te bepalen in ons programma, maken we ook gebruik van en statistische analyse van de pixelwaarden. Het vak 'Statistiek' draagt hier dus ook zijn steentje bij.

## 6 Besluit

We zijn er in geslaagd een app te ontwikkelen die in staat is om van een afbeelding van een bloedklontje de achtergrond te verwijderen zodanig dat deze ruisvrij wordt en die de foto automatisch kan bijsnijden. Daarnaast is er de mogelijkheid om het percentage van de hoeveelheid aanwezige indicator op te vragen. Zoals eerder vermeld, worden deze indicatoren in de afbeelding redelijk precies aangeduid maar is dit niet voor elke afbeelding even voor de hand liggend. Dit is te wijten aan de verschillen tussen de afbeelding



door andere andere de kleuring die van intensiteit kan variëren.

Daarom hebben we er ook voor gezorgd dat er in de app manueel kan worden ingegrepen door middel van verschillende sliders.

Verder hebben we er rekening mee gehouden dat onze app gebruikt kan worden door klanten die geen MATLAB hebben geïnstalleerd. Dit zorgt ervoor dat ons eindresultaat voor iedereen gebruiksvriendelijk is.



## Referenties

- [1] the top 10 causes of death, 24 may 2018. <http://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death>.

	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
<b>1</b>	Achtergrond verwijderen en afbeelding bijsnijden
<b>2</b>	Kleuren detecteren en kwantificeren
<b>3</b>	Ontwerpen van een gebruiksvriendelijke app
<b>4</b>	Optimalisatie van het algoritme

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