RIJKSUNIVERSITEIT GRONINGEN

BACHELORVERSLAG

# Semi-automatische Jaarringdetectie in doorsnedefoto's van boomstammen

*Auteur:*
Herbert Kruitbosch

*Begeleiders:*
1$^e$: Dr. Arnold Meijster
2$^e$: Dr. Michael Wilkinson

13 juli 2012

# Inhoudsopgave


1 Inleiding . . . 3
2 ROI-selectie . . . 4
3 Bepaling hart . . . 8
3.1 Definitie hart . . . 9
3.2 Kandidaatharten bepalen . . . 9
3.3 Hart bepalen . . . 11
3.4 Hart nauwkeuriger bepalen . . . 12
4 Rechttrekken . . . 14
4.1 Ringen . . . 15
4.2 Rechttrekken . . . 15
4.3 Ringen bepalen . . . 17
5 Seamcarving . . . 25
5.1 Seamcarving-algoritme . . . 25
5.2 Energiebeeld . . . 25
5.3 Ringen bepalen . . . 26
5.4 Ring uitzoeken . . . 27
6 Gebruik . . . 30
6.1 Openen en opslaan . . . 30
6.2 ROI . . . 30
6.3 Hart . . . 30
6.4 Rechttrekken . . . 32
6.5 Ringen . . . 33
7 Conclusies . . . 34
7.1 Gevonden ROI . . . 34
7.2 Gevonden hart . . . 34
7.3 Gevonden jaarringen . . . 34
8 Toekomstige uitbreidingen . . . 36
8.1 Meerkanaalse invoer . . . 36
8.2 Convergentiegedrag hart . . . 36
8.3 Ruis verwijderen . . . 36
8.4 Alternatieve methode voor zoeken naar het hart . . . 37

8.5 Meerdere metingen . . . . . 37
8.6 Verloren jaarringen alsnog vinden . . . . . 37
8.7 Jaarringen tellen . . . . . 37
8.8 Usability test . . . . . 38

## Samenvatting

In dit verslag bespreek ik mijn bachelorproject, waarin ik me bezig heb gehouden met het automatisch detecteren van de jaarringen van een boom. Dit systeem heeft als invoer een digitale foto van een doorsnede van een boomstam. De doorsnedefoto bevat in het *centrum* een hart met daar omheen jaarringen. Een jaarring wordt aangegeven door een harde overgang, of rand, tussen donker en licht hout, zoals die in de stam van bomen typisch worden aangetroffen. De uitvoer van het systeem is een set van jaarringen. De jaarringen zijn elk op hun beurt een verzameling van poolcoördinaten die de ring aangeven ten opzichte van het *hart*. De poolcoördinaten zijn een numerieke benadering van de jaarringen. De gedetecteerde jaarringen liggen over het algemeen precies op de rand tussen een donkere en een lichte ring. Helaas worden niet alle ringen gevonden, en de gevonden ringen wijken ook zo nu en dan af van de werkelijke ringen. Bijvoorbeeld doordat een gevonden ring meerdere andere ringen volgt, door zo nu en dan te *verspringen*.

Tevens bespreek ik ook de gebruikersinterface, die ervoor probeert te zorgen dat een gebruiker de jaarringen geautomatiseerd kan detecteren en waar nodig fouten kan corrigeren of jaarringen handmatig kan toevoegen.

# 1 Inleiding

Het systeem dat ik in dit artikel bespreek probeert boomringen te detecteren. Het systeem begint met een foto van een stuk boomstam die ooit is doorgezaagd. De gedachte achter het systeem is vervolgens dat wanneer deze foto en het stuk stam van voldoende kwaliteit zijn, het de jaarringen in de foto kan detecteren.

Het systeem gaat ervan uit dat de digitale foto, die het als invoer krijgt, drie kleurkanalen heeft, namelijk rood, groen en bauw. Het berekent daarna in drie stadia de jaarringen die op die foto staan. De digitale foto bevat naast de *region of interest* (of *ROI*) in veel gevallen ook achtergrond. In sectie 2 bespreek ik hoe de ROI wordt bepaald.

In het tweede stadium bepaalt het proces waar het *hart* ligt. Aan de hand van de positie van dit hart kunnen alle later gebruikte digitale foto's of *beelden* getransformeerd worden naar een poolcoördinaten-representatie. Door gebruik te maken van een twee-dimensionale poolcoördinatenbeeld ontstaat een duidelijkere scheiding van de afstand die een ring tot het hart heeft en de positie op de ring. Hoe het hart wordt bepaald bespreek ik in sectie 3.

In het derde en laatste stadium wordt de positie van de ringen bepaald in de vorm van een verzameling van poolcoördinaten. Dit deel bespreek ik in secties 4 en 5.

Als de berekeningen van het systeem besproken zijn, dan zal ik in sectie 6 nog toelichten hoe deze afzonderlijke delen als een geheel tot een programma zijn verwerkt. Ik heb een interface gemaakt waarmee de gebruiker correcties kan uitvoeren tussen alle stappen. Op die manier kunnen bij een eventuele fout bij de berekening van bijvoorbeeld het hart of de ROI alsnog nuttige vervolgstappen worden gedaan. De interface is tevens een *wizard* die voor elke berekening uitleg geeft over hetgeen dat de gebruiker moet doen of controleren.

De uitvoer van het programma is een verzameling jaarringen die een boomexpert kan gebruiken om informatie te verzamelen over de situatie en het klimaat op de plaats waar de boom gedurende zijn leven stond.

## 2 ROI-selectie

In deze sectie bespreek ik hoe het systeem de ROI van de invoerfoto bepaalt. Zo'n invoerfoto bevat veel informatie, zoals de kleur van een bepaald gebied en de *lichtheid* of *luminatie*. De kleur van de ROI is typisch bruin, omdat hout bruin is. De achtergrond is typisch een andere kleur. Dat de achtergrond niet bruin is, is een aanname en geen feit. Immers, de foto kan gemaakt zijn van een stuk stam dat op een tafel van hetzelfde soort hout ligt.

Fig. 1: Voorbeeld van een invoerfoto.

De *hsv*-kleurruimte (*hue*, *saturation* en *value*, zie [8]) is een geschikte ruimte voor berekeningen die met zulke kleur-heuristieken werken. Stel we nemen de foto $F$ uit figuur 1, met $F_{hue}$ het hue-beeld van deze foto en $F_{hue}(x, y) \in [0..256) \subset \mathbb{Z}$. Omdat we voor het bepalen van de ROI geen details nodig hebben, verminderen we eerst ruis door te stellen dat $\hat{F}_{hue} = F_{hue} * G_{4.0}$ met $*$ de convolutie-operator en

$$G_\sigma(x, y) = \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}.$$

Vervolgens bepalen we een $a$ en $b$ en selecteren alle pixels op positie $(x, y)$ waarvoor

$$a \leq \hat{F}_{hue}(x, y) < b \tag{1}$$

geldt. Met andere woorden, we selecteren het gebied waar de *hue*-waarde in $[a..b)$ ligt. Voor een *goed* gekozen $a$ en $b$, in dit geval respectievelijk 3 en 24, onstaat dan figuur 2. Uit dit figuur blijkt dat (1) een redelijke manier is om de boomstam van *de rest* te onderscheiden. Er is nog niet persé een perfecte selectie, er is immers nog ruis, maar daar kom ik later op terug. Nu bespreek ik eerst een methode om een geschikte $a$ en $b$ te bepalen.

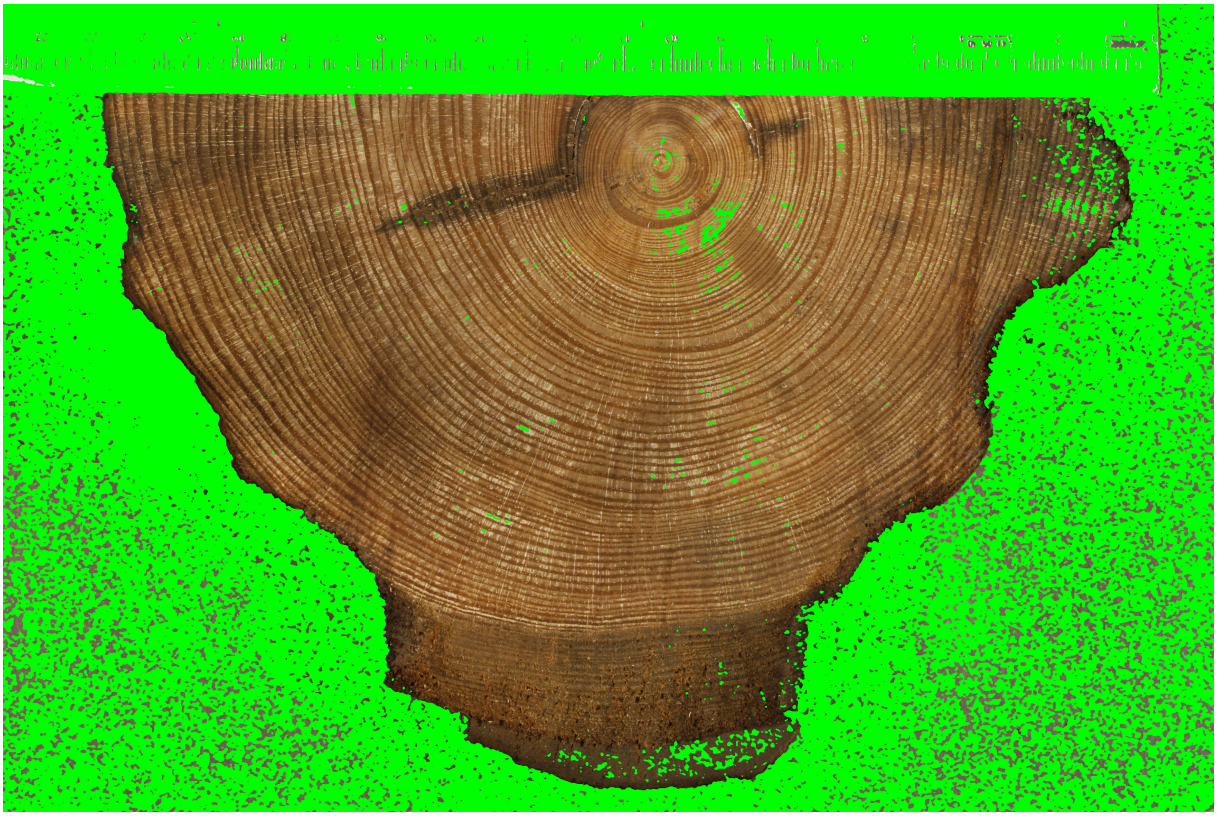
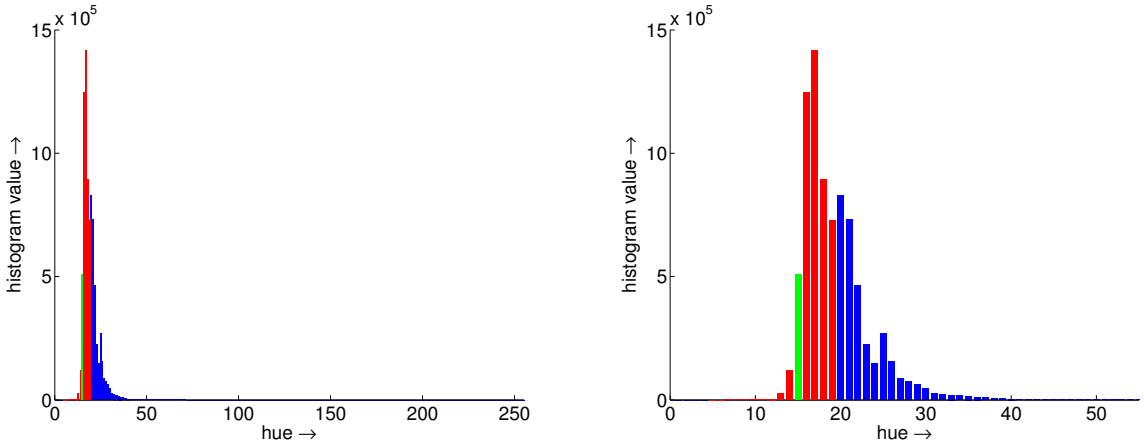


Fig. 2: Selectie volgens (1) met $a = 3$ en $b = 24$. Alle pixels die niet geselecteerd zijn, zijn fel groen.

Stel we benaderen (1) door alle pixels $(x, y)$ te selecteren zodat $\hat{F}_{hue}(x, y) = c$. Dan hoeven we nog maar 1 waarde te bepalen, namelijk $c$. Daarvoor kunnen we een constante bruine waarde uit het *hue*-bereik $[0..256)$ nemen, maar het nadeel daarvan is dat die methode erg gevoelig is voor kleurverschuivingen, zoals door andere belichting of door *bijzonder* hout. Sommige houtsoorten zijn bijvoorbeeld eerder rood of juist groen dan bruin.

Om een benadering van de kleur van het hout te krijgen, nemen we een *sample* uit de foto waarvan we sterk vermoeden dat daar alleen hout van de stam, de ROI, afgebeeld is. Zoals een vierkant van pixels $V = \{(x, y)|x \in [\dot{x} - N..\dot{y} + N] \text{ en } y \in [\dot{y} - N..\dot{y} + N]\}$, met $\dot{x}$ en $\dot{y}$ de helft van respectievelijk de breedte en hoogte van de foto in pixels. $V$ is een vierkant in het midden van de foto, waarvan het waarschijnlijk is dat daar alleen ROI wordt weergegeven.

De gemiddelde *hue*-waarde $c = \frac{\sum_{(x,y)\in V} \hat{F}_{hue}(x,y)}{\#V}$ in dat gebied is dan een aardige indicatie van de kleur van de stam. Deze $c$ noem ik de pijlwaarde. In principe hebben we nu een *hue*-bereik $[c, c+1)$ om mee te selecteren, desalnietemin werkt dit bereik slecht. De aanname dat $a = b+1 = c$ is dan ook onterecht, de stam zal immers typisch meer dan 1 *hue*-waarde in het histogram beslaan. Maar we kunnen deze $c$ wel gebruiken om een bereik $[a..b)$ te bepalen dat een beter resultaat geeft dan $[c..c + 1)$.




(a) $H_1$ volledig (b) $H_1$ met $0 \leq hue \leq 55$

Fig. 3: Histogram $H_1$ van $\hat{F}_{hue}$

Figuur 3a geeft een histogram $H_1$ weer van alle *hue*-waarden van $\hat{F}_{hue}$. Een hue-domein[1] tussen twee opvolgende lokale minima $a < b$, dat geen ander lokaal minimum bevat, noem ik een berg. Wanneer er maar weinig overlap bestaat tussen de hue-waarden in de ROI en de rest van de foto, dan bestaat er een berg $[a..b)$ die de ROI selecteert, omdat de ROI tevens de kleur $c$ bevat moet gelden $a \leq c < b$. Er geldt dan

$$\begin{aligned}\hat{H}_1(i) &= H_1((i+256) \mod 256)\\ a &= \max\{i | i \in [0..256), \hat{H}_1(i-1) > \hat{H}_1(i) \leq \hat{H}(i+1), i < c\} \mod 256\\ b &= \min\{i | i \in [0..256), \hat{H}_1(i-1) \leq \hat{H}_1(i) > \hat{H}(i+1), i > c\} \mod 256\end{aligned}$$

In figuur 3a is het rechter minimum nauwelijks zichtbaar omdat die in dit geval maar 1 balkje breed is. Dat komt doordat in ons invoerbeeld de achtergrond *zwartachtig* is. Omdat voor grijswaarden geen hue-waarden gedefinieerd zijn, is zwart niet persé goed te onderscheiden aan de hand van een hue-waarde. Door een blauwe achtergrond te gebruiken bij het nemen van de foto kan de ROI beter geselecteerd worden.

Ik heb desalniettemin toch gekozen voor een algoritme dat aan de hand van kleur de ROI bepaald, in plaats van bijvoorbeeld grijswaarde of luminatie. Een *threshold* op een waarde afhankelijk van luminatie is erg gevoelig voor de belichting van de foto. Daardoor is het lastig om een globaal selectiecriterium, zoals een thresholdwaarde, te vinden die daar rekening mee houdt. Een lokaal selectiecriterium negeert op zijn beurt het globale karakter van de ROI. De ROI is immers een aaneengesloten en eenvoudige, zonder enclaves, *blob*. Voor het midden van zo'n blob bestaat bijvoorbeeld geen goede manier om een lokale threshold te bepalen, omdat daar geen informatie over de achtergrond is opgeslagen.

Ondanks onze goede intentie om de ROI te selecteren met een globaal criterium, is het met $[a..b)$ geselecteerde gebied niet aaneengesloten (i) en ook niet eenvoudig (ii). In figuur 2 zien we dat dit wel *bijna* zo is. Door in eerste instantie het grootste aaneengesloten geselecteerde

[1] Omdat we het nu over een histogram hebben, is dit een domein en niet een bereik. Maar in essentie is dit hetzelfde als het *hue*-bereik van eerder.

gebied te nemen, en de rest te *deselecteren*, wordt in ieder geval aan (i) voldaan (figuur 4a). Aan deze selectie voegen we het minimale aantal pixels toe zodat deze ook eenvoudig is en aan (ii) wordt voldaan. Die selectie is de uiteindelijke ROI (figuur 4b).

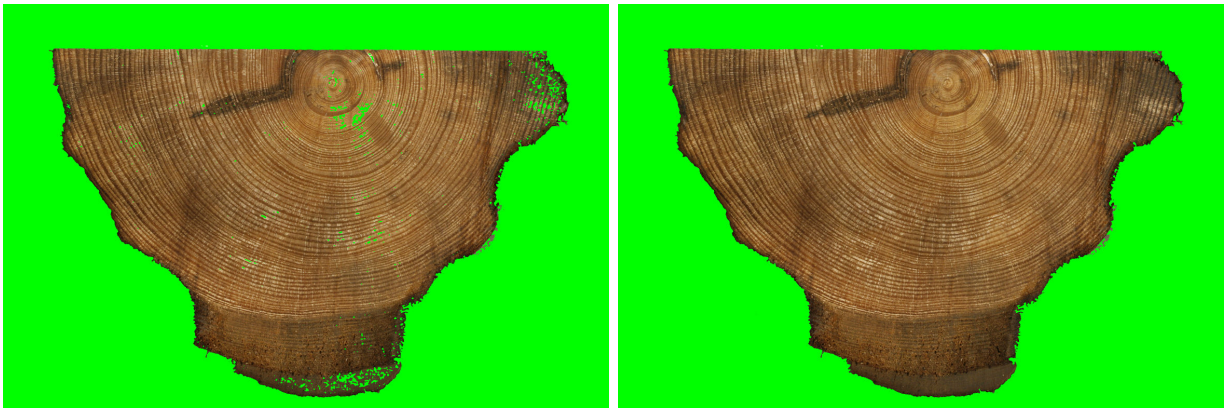


Fig. 4: In (a) het grootste aaneengesloten gebied uit figuur 2, in (b) het gebied uit (a) met de minimale set pixels toegevoegd zodat (b) eenvoudig is.

# 3 Bepaling hart

*Na het bepalen van de ROI gaat het systeem verder met 2 beelden. Een binair beeld $R$ dat enkel in de ROI* 1 *is en een grijswaardenbeeld $G$. Het beeld $G$ kan in feite iedere grijswaardenconversie zijn van de foto, zolang er maar een duidelijk onderscheid gemaakt kan worden tussen donkere en lichte ringen. Het systeem gebruikt de volgende:*

$$\begin{aligned} G(x,y) &= F_{saturation}(x,y) \cdot (1 - F_{value}(x,y)) \\ &= \min\{F_{red}(x,y), F_{green}(x,y), F_{blue}(x,y)\}. \end{aligned}$$

*Dit grijswaardenbeeld en de ROI kort ik af als het paar $(G, R)$.*

## 3.1 Definitie hart

In deze sectie bespreek ik hoe het systeem met $(G, R)$ het hart van de boomstam bepaalt. Dit hart kunnen we later gebruiken om $(G, R)$ naar poolcoördinaten te transformeren. Het hart van de boom is een positie $(x, y)$ in het domein van $G$ die binnen alle jaarringen ligt.

Zo'n hart moet voornamelijk aan één eis voldoen, namelijk dat het *binnen* elke jaarring ligt. De jaarringen waar het hart niet binnen ligt zullen na de transformatie in de volgende sectie een vreemde vorm krijgen en niet gedetecteerd worden. Stel dat we de verzameling jaarringen $r_1..r_n$ proberen te vinden waarbij voor $i < j$ geldt dat $r_i$ binnen $r_j$ ligt. Daarmee is $r_1..r_n$ dus gesorteerd van *klein* naar *groot*. Inherent aan deze sortering is dat als het hart in $r_i$ ligt, het ook in $r_j$ ligt met $j \geq i$.

Het hart dat het systeem wil vinden ligt dus binnen $r_1$. In principe zijn er meerdere punten die binnen $r_1$ liggen, maar omdat de binnenste ring erg klein is, kunnen we stellen dat er bij benadering dus precies één hart is.

## 3.2 Kandidaatharten bepalen

Eigenlijk zouden we voor het hart dus het zwaartepunt van $r_1$ moeten nemen. Maar om het hart te bepalen kunnen we helaas de jaarringen niet gebruiken. Daarmee komt het *kip en ei*-karakter van dit probleem naar voren. Het hart voldoet aan een eis gebaseerd op de geometrie van de ringen, en om de geometrie van de ringen te kunnen bepalen gebruiken we het hart.

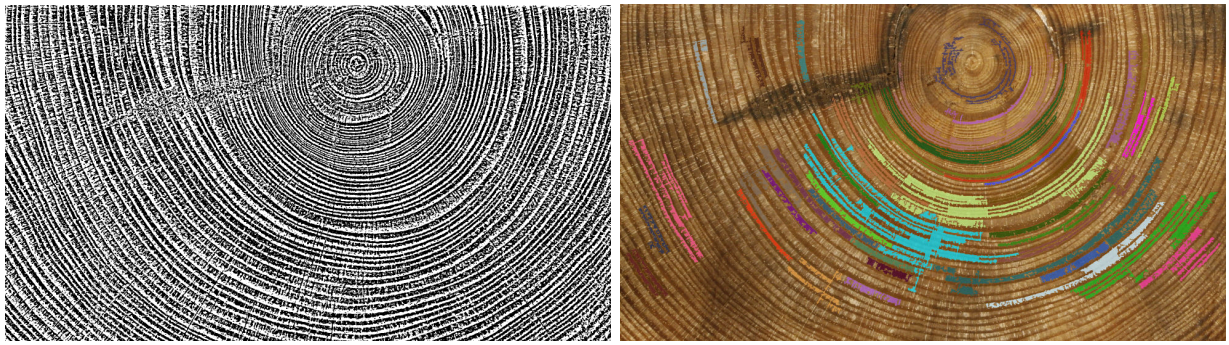
Daarom bepalen we eerst een verzameling ringstukken. Die ringstukken vormen een geometrisch beginsel voor het vinden van het hart. De ringstukken worden gezocht door in de afbeelding te *thresholden*. We gaan ervan uit dat ongeveer de helft van het oppervlak in $(G, R)$ een donkere ring is, en dat het overige deel een lichte ring is, daarvoor gebruiken we een *adaptive threshold*. Die bepaalt voor iedere punt $(x, y)$ een lokale threshold $T(x, y)$ aan de hand van een vierkant om dat punt, namelijk $V_N(x, y) = \{(x, y) | x \in [x-N..x+N], y \in [y-N..y+N]\}$. $T(x, y)$ is dan het gemiddelde van alle grijswaarden in $V_N(x, y)$. Dan geldt:

$$H(x,y) = \begin{cases} 0 \text{ if } G(x,y) \leq T(x,y) \text{ or } R(x,y) = 0 \\ 1 \text{ if } otherwise \end{cases} \qquad \text{waarbij}$$

$$T(x,y) = \frac{\sum_{p \in V_N(x,y)} G(p)}{\#V_N(x,y)}$$

Daarin geldt dat $N = 15$ en dat $H$ een binair beeld is, daarin zitten dus *connected components*. We nemen daarvan de $N_{cc} = 40$ grootste connected components, en stellen dat zo'n component een benadering voor een stuk van een ring is. In figuur 5 worden $H$ en deze connected components weergegeven.

Het idee is dat we loodlijnen op de connected components construren. Net de loodlijn op een cirkel door het midden van die cirkel gaat, zou hier de loodlijn op een connected component door het hart van de boom moeten gaan. Een loodlijn is echter nog niet gedefinieerd op een connected component.



(a) threshold $H$ (b) connected components

Fig. 5: In (a) staat $H$, in (b) staan de $N_{cc} = 40$ grootste connected components in verschillende kleuren afgedrukt.

Dit is vergelijkbaar met een loodlijn op een cirkel: die loodlijn gaat door het midden van de cirkel. Omdat een connected component geen curve is, maar een verzameling pixels, is het lastig een loodlijn op zo'n object te definiëren. Wel kunnen we een rechte lijn op een connected component *fitten*. Het systeem fit de lijn als eerst orde *least-square*-oplossing van alle pixelcoordinaren van een connected component.

De lijn die door het zwaartepunt van een connected component heengaat en loodrecht op de gefitte lijn staat, dat is de loodlijn van de connected component. Figuur 6 geeft dit weer.

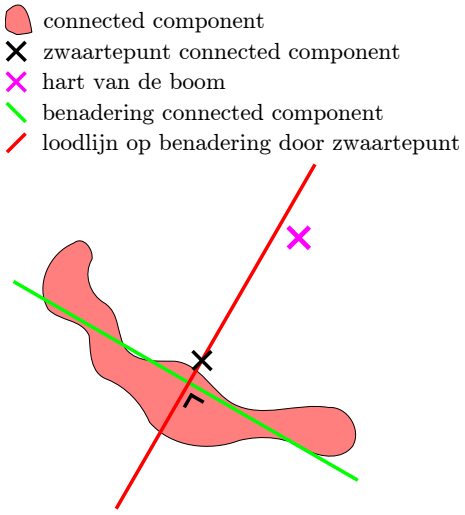



Fig. 6: Eerst wordt de groene lijn als benadering van het connected component bepaald, daarna de rode lijn.

Het systeem neemt aan dat zo'n loodlijn op de ring door het hart heen gaat. Twee loodlijnen zouden in feite een kandidaat-hart vormen: het snijpunt van beide. Omdat de ringen slechts bij benadering cirkels zijn en omdat de connected components niet precies een stuk jaaring definiëren, is zo'n kandidaathart niet persé een geschikt hart. In de meeste gevallen zal de afstand tussen het hart en zo'n loodlijn niet 0 zijn.

Wanneer we echter voor alle connected components zo'n loodlijn construeren, dan onstaat $N_{cc}$ loodlijnen, en daarmee $N_{cc} \cdot (N_{cc} - 1)$ snijpunten. Sommige loodlijnen zullen evenwijdig zijn, of bijna evenwijdig, waardoor een snijpunt lastig te bepalen is. Die snijpunten worden verworpen. De snijpunten die overblijven gebruiken we dan om het midden te bepalen. Dat kan bijvoorbeeld door het gemiddelde te nemen. Ik zal nu een subtielere, minder democratische methode bespreken, die stelt dat snijpunten in een *dichtbevolkt* gebied waarschijnlijk een betrouwbaarder kandidaatmiddelvormen dan anderen.

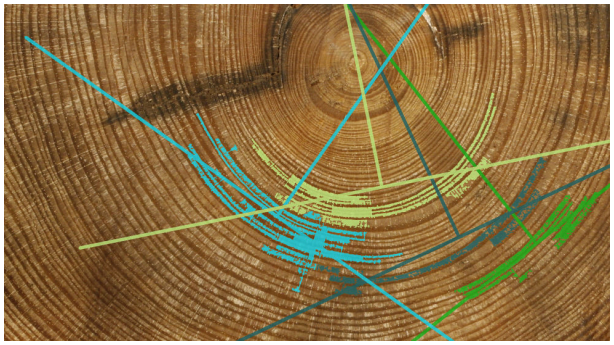


Fig. 7: De 4 grootste *connected components* uit 5b, inclusief gefitte lijnen en loodlijnen.

## 3.3 Hart bepalen

Zoals figuur 8a weergeeft, kunnen de snijpunten behoorlijk over het beeld verspreid liggen. We stellen dat snijpunten in een gebied met een hoge dichtheid van snijpunten een kleinere fout hebben dan snijpunten in een dun verspreid gebied. Dat volgt doordat de loodlijnen die een grotere fout hebben, en dus niet bij benadering één snijpunt hebben, hun snijpunten veel meer verspreiden dan loodlijnen die een kleinere fout hebben. Stel bijvoorbeeld dat de helft van de loodlijnen een grote fout heeft, en de andere helft niet. Dan liggen alle snijpunten tussen ten minste één *foute* loodlijn en een andere loodlijn niet bij elkaar in de buurt. Anderzijds liggen de snijpunten van twee *goede* loodlijnen wel bij elkaar in de buurt. Grofweg een kwart van de punten ligt dan bij elkaar in de buurt, en de rest niet. Een kwart omdat grofweg 1 deel van de snijpunten afkomstig is van twee goede loodlijnen, en grofweg 3 delen van ten minste 1 foute loodlijn.

In principe kunnen alle loodlijnen een grote fout hebben, en alsnog een aanzienlijke deelverzameling een gezamelijk snijpunt hebben. Dat snijpunt ligt dan ver van het hart af. Dat gedrag volgt echter niet uit de eigenschappen van de connected components die we hebben gevonden, immers benaderen die stukken ring, die op hun beurt cirkels om het hart benaderen.

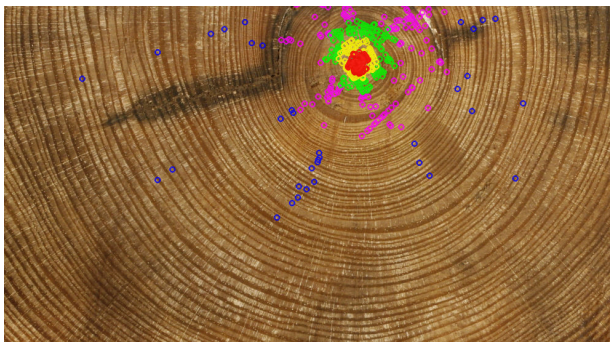
Om nu het *dichtbevolkte* gebied te vinden, bepalen we iteratief een hart aan de hand van het vorige gevonden hart. Het eerste hart is het gemiddelde van alle snijpunten. Om een volgend hart te bepalen wordt de helft van de snijpunten verworpen, namelijk dat deel dat het verst van het hart afligt. Het volgende hart is dan het gemiddelde van de overgebleven snijpunten. Stel dat $\mathcal{S}$ de verzameling snijpunten is, waarvan er nog geen is verworpen, stel tevens dat $h_i$ het $i$-de gevonden hart is en stel dat $\mathcal{S}_i$ de verzameling snijpunten is waarmee $h_i$ werd bepaalt. Om de te verwerpen helft van de punten in $\mathcal{S}_i$ wordt de mediaan van alle afstanden van alle punten in $\mathcal{S}_i$ tot $h_i$ gebruikt. Alle punten die een grotere afstand tot $h_i$ hebben dan de mediaan worden verworpen.

Afhankelijk van of de linker of rechter mediaan gerbuikt wordt of het bestaan van meerdere punten zijn die precies op $d_i$ afstand van $h_i$ liggen hoeft niet persé *precies* de helft van de snijpunten verworpen te worden. Over het algemeen zal het aantal snijpunten echter voldoende afnemen om een hart te bepalen. In het kunstmatige geval dat alle snijpunten op $d_i$ om $h_i$ liggen, en dus allemaal op een cirkel om $h_i$ liggen, neemt het aantal snijpunten bijvoorbeeld niet af. Dan is er hoe dan ook niets beters te doen dan aannemen dat $h_i$ het hart is.

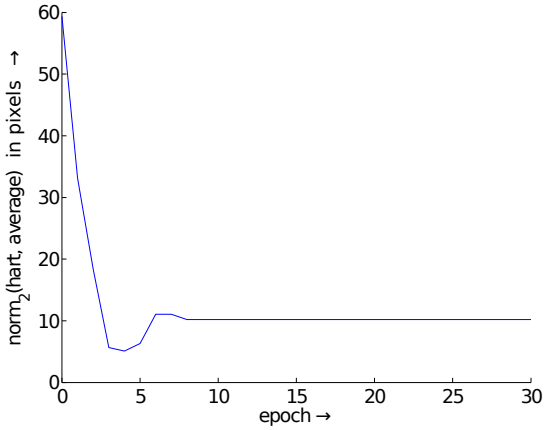
Nu geldt dus dat $h_i$ het gemiddelde van $\mathcal{S}_i$ is en $\mathcal{S}_1 = \mathcal{S}$. Dan geldtvoor $i = 2, 3, ...$ dat

$$
\begin{aligned}
h_{i-1} &= \frac{\sum_{h \in \mathcal{S}_{i-1}} h}{\#\mathcal{S}_{i-1}} && \text{gemiddelde van } \mathcal{S}_{i-1} \\
d_i &= med\{\|h - h_{i-1}\| : h \in \mathcal{S}_{i-1}\} && \text{mediaan van afstanden tot gemiddelde van } \mathcal{S}_{i-1} \\
\mathcal{S}_i &= \{h : h \in \mathcal{S}_{i-1}, \|h - h_{i-1}\| \leq d_i\} && \text{helft van de punten die het dichtst bij } h_{i-1} \text{ liggen} \\
h_i &= \frac{\sum_{h \in \mathcal{S}_i} h}{\#\mathcal{S}_i} && \text{gemiddelde van } \mathcal{S}_i
\end{aligned}
$$

In figuur 8b zien we dat het aantal snijpunten convergeert naar het daadwerkelijke hart. Dat wil zeggen, naar een afstand om het daadwerkelijke hart die ik handmatig heb opgegeven. Merk overigens op dat we beginnen met (ongeveer) $40 \times 39$ snijpunten, en niet met 40. Dat aantal komt na 10 keer halveren aardig in de buurt van 1. We zien na 10 epochs dan ook dat $h_i$ niet meer verandert.



(a) snijpunten




(b) fout

Fig. 8: Alle snijpunten in $\mathcal{S}$, de snijpunten die overblijven na een bepaalde epoch hebben een ander kleur. Blauw voor alle epochs, paars na de eerste, groen na de tweede, geel na de derde en rood na de vierde.

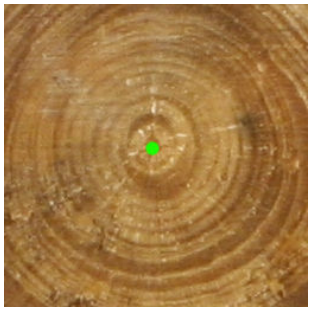
## 3.4 Hart nauwkeuriger bepalen

Het systeem vindt zo een hart. Helaas vindt het iteratieve beleid nog niet persé een nauwkeurig hart, typisch zal deze alsnog 10 tot 15 pixels afwijken. Immer, de gebruikte connected components zijn van zowel grotere als kleinere ringen afkomstig. Omdat de grotere ringen groeien op basis van de vorm van de kleinere ringen, tellen fouten zich op naarmate een boom ouder wordt en meer ringen krijgt. Daardoor hebben jongere ringen meer de neiging om af te wijken van cirkels dan oudere. Daarnaast gebruiken we de grotere connected compontents, en daarmee zijn meer loodlijnen ook waarschijnlijker afkomstig van de grotere ringen. De conclusie is dat het hart niet bepaald wordt met de meest nauwkeurige informatie die er is.

Daarom gebruikt het systeem het gevonden hart om opnieuw te beginnen met het zoeken naar een tweede hart, maar nu alleen met de connected components in de buurt van het gevonden hart. Daarom maakt het een uitsnede uit $(G, S)$ om het gevonden hart, namelijk

$U = \{(x, y)|x \in [h_x - M..h_x + M]$ en $y \in [h_y - M..h_y + M]\}$, met $h$ het gevonden hart. Door deze uitsnede te gebruiken om opnieuw het hart te bepalen, wordt informatie van de kleinere en nauwkeurigere ringen gebruikt. In de praktijk blijkt dat het systeem een fout van 10 pixels voor de uitsnede kan verkleinen tot 2 pixels, in figuur 9 staat een voorbeeld van het gevonden hart voor en na de uitsnede.

(a) eerste gevonden hart



(b) hart gevonden in uitsnede

Fig. 9: De harten, in (a) zoals gevonden in de originele afbeelding en in (b) zoals gevonden in de uitsnede $V$ uit (a) met $M = 128$.

# 4 Rechttrekken

*Nu het hart bekend is, kunnen we $(G, R)$ naar poolcoördinaten transformeren. De resolutie van zo'n poolcoördinatenbeels kunnen we in feite kiezen. Ik heb gekozen voor een resolutie van $8000_r \times 800_\phi$, dit blijk voldoende te zijn voor de bewerkingen die volgen. We introduceren de notatie $(\dot{G}, \dot{R})$ voor de beelden in poolcoördinaten. De transformatie wordt gegeven door:*

$$H = \{(0,0), (width, 0), (0, height), (width, height)\} \qquad \text{alle hoekpunten}$$

$$r_{max} = \max\{\|p - h\| \;\; |p \in H\} \qquad \text{maximale afstand tussen een pixel en het hart}$$

$$t_x(\phi, r) = \frac{8000}{r_{max}} r \cdot \cos(\frac{2\pi}{800}\phi) + h_x \qquad \text{transformatie } (\phi, r) \to x$$

$$t_y(\phi, r) = \frac{8000}{r_{max}} r \cdot \sin(\frac{2\pi}{800}\phi) + h_y \qquad \text{transformatie } (\phi, r) \to y$$

$$\dot{G}(\phi, r) = G(t_x(\phi, r), t_y(\phi, r)) \qquad \text{signaal, in poolcoördinaten}$$

$$\dot{R}(\phi, r) = R(t_x(\phi, r), t_y(\phi, r)) \qquad \text{region of interest.}$$

*Inherent aan deze transformatie is dat $(\dot{G}, \dot{R})$ $8000_r \times 800_\phi$ pixels van afmeting is, ongeacht de afmeting van $(G, R)$. Merk op dat in de context van dit document $\dot{G}$ en $\dot{R}$ geen afgeleiden naar de tijd zijn. Daarnaast zullen $t_x$ en $t_y$ in $\mathbb{R}$ zitten, waardoor $(t_x, t_y)$ in het (discrete) signaal $(G, R)$ niet goed gedefinieerd is. De interpolatie die in dit geval gebruikt wordt bespreek ik hier niet. Het systeem gebruikt de methode* remap *uit de opencv-bibliotheek (zie [1]) om de transformatie uit te voeren, die handelt het interpolatie-probleem af.*

(a) poolcoördinaten

Fig. 10: Figuur 1 in poolcoördinaten. Om de afbeelding op de pagina te laten passen, zonder details te verliezen, is van de $r$-as alleen het [0..3200)-deel van [0..8000) weergegeven.

*In figuur 1 en volgende figuren is de verticale as $\phi$ en de horizontale as $r$. Daarnaast tonen we in alle volgende figuren in poolcoördinaten steeds het gebied met $r < 3200$ vanwege layout-redenen.*

## 4.1 Ringen

Nu we een signaal $(\dot{G}, \dot{R})$ in poolcoördinaten hebben kunnen we op zoek gaan naar jaarringen. Een jaarring is in essentie een gesloten curve over het domein van $(\dot{G}, \dot{R})$. Merk op dat in figuur 10 een ring er niet gesloten uitziet, want hij begint in de bovenste en eindigt in de onderste scanline. Toch noem ik zo'n curve gesloten, omdat de onderste en bovenste scanline naburig zijn in het origineel. In feite kunnen we die scanlines aan elkaar plakken, zodat het beeld een torus wordt.

Zo'n gesloten curve kunnen we benaderen met een veelhoek: een lijst punten $(r_0, \phi_0), (r_1, \phi_1), ..., (r_{799}, \phi_{799})$ die op de curve liggen. Omdat de curve gesloten is geldt dan dat $(r_0, \phi_0)$ en $(r_{799}, \phi_{799})$ aangrenzend zijn.

Een eis die ik aan de gesloten curve heb opgelegd is dat er voor iedere hoek $\phi$ precies één punt $(r_\phi, \phi)$ bestaat dat op de curve ligt. Merk op dat een hoek hier niet een punt op de veelhoek is, maar een waarde op de verticale $\phi$-as van $(\tilde{G}, \tilde{R})$. Een *hoek* op de veelhoek noem ik een punt op de veelhoek. In het discrete signaal $(\dot{G}, \dot{R})$ betekent dit dat zo'n curve te benaderen is met een punt voor iedere hoek $\phi \in [0..800)$, namelijk: $(r_0, 0_\phi), (r_1, 1_\phi), ..., (r_{799}, 799_\phi)$. Zo'n verzameling punten noem ik vanaf nu een ring. Er is een subtiel verschil tussen een ring en een jaarring. Een ring is een berekend object en een jaarring is een verschijnsel in de natuur, namelijk in de stam van een boom.

Als $r_0 = r_1 = ... = r_{799}$ geldt, dan benaderen deze punten een perfecte cirkel. Immer, dan is de afstand tot het hart constant. Op een ring waar voor een $\phi$ geldt dat $r_\phi \neq r_{\phi+1}$, geldt dan dat $|r_\phi - r_{\phi+1}|$ een maat is voor de lokale afwijking van een perfecte cirkel om het hart op dat punt. Het algoritme dat de jaarringen bepaalt, dat wordt besproken in sectie 5, zoekt naar ringen die veel op perfecte cirkels lijken en die volgens $(\dot{G}, \dot{R})$ een jaarring lijken te zijn. Daardoor worden jaarringen in $(\dot{G}, \dot{R})$ die te veel van een perfecte cirkel afwijken niet goed gevonden.

Een voorbeeld van zo'n jaarring is te zien in figuur 12a, en in figuur 12b is een voorbeeld te zien waar deze jaarring *rechtgetrokken* is aan de hand van een transformatie die ik zometeen bespreek. Daardoor wijkt de jaarring nauwelijks af van een perfecte cirkel, en ook ringen in de buurt lijken meer op cirkels. Daardoor vindt het algoritme uit sectie 5 de jaarringen beter.

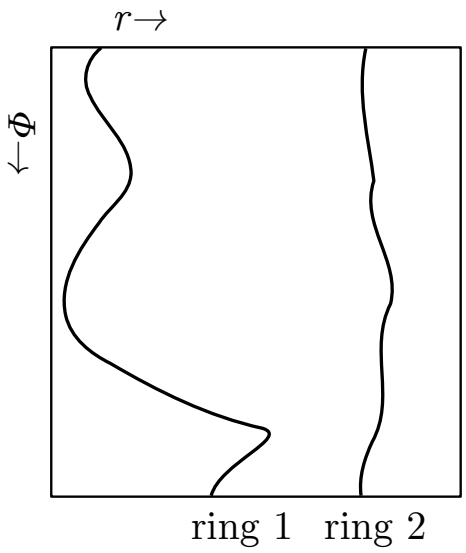



Fig. 11: Bij ring 1 is $|r_\phi - r_{\phi+1}|$ over het algemeen groter dan bij ring 2.

## 4.2 Rechttrekken

De intentie van dit algoritme is om gegeven een ring, $(\dot{G}, \dot{R})$ aan te passen zodanig dat $(\tilde{G}, \tilde{R})$ ontstaat, waarin die ring een perfecte cirkel is. Tevens dient de transformatie niet te veel

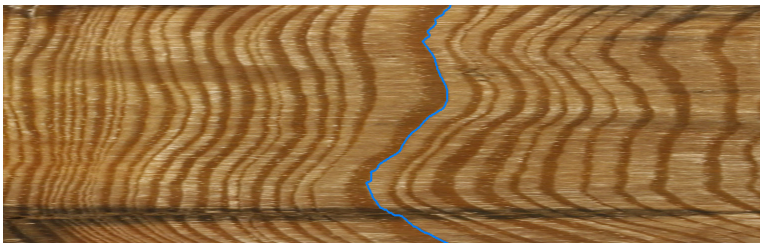
van de geometrie van $(\dot{G}, \dot{R})$ te veranderen. Wanneer we dan één jaarring $j$ gebruiken zal die na de transformatie een perfecte cirkel zijn. Omdat jaarringen in de buurt van $j$ over het algemeen niet veel afwijken van $j$, zullen die na de transformatie ook meer op perfecte cirkels gaan lijken. Op die manier hoeven we maar een deel van alle jaarringen op te geven om zo $(\dot{G}, \dot{R})$ geschikter te maken voor het bepalen van jaarringen.

Ik zal nu uitleggen hoe de transformatie precies werkt. Stel nu dat de ring die we rechttrekken $(r_0, 0), ..., (r_{799}, 799)$ is. In feite moeten we ervoor zorgen dat deze ring in $(\tilde{G}, \tilde{R})$ gelijk is aan $(\overline{r}, 0), ..., (\overline{r}, 799)$ voor een zekere $\overline{r}$. Dan is het immers een perfecte cirkel geworden. In principe kunnen we voor $\overline{r}$ elke waarde in $[0...8000)$ nemen, we willen echter dat de ring niet veel verplaatst. Daarom is het gemiddelde van $r_0, ..., r_{799}$ een betere keuze.

Nu we voor alle $\phi \in [0..800)$ een transformatie $(r_\phi, \phi) \to (\overline{r}, \phi)$ hebben, kunnen we ook voor alle overige punten in het domein van $(\dot{G}, \dot{R})$ een transformatie definiëren. Namelijk zo:

$$\tilde{G}(r, \phi) = \begin{cases} \dot{G}(\frac{r \cdot r_\phi}{\overline{r}}, \phi) & \text{als } r < \overline{r} \\ \dot{G}(r_\phi + \frac{(r-\overline{r}) \cdot (8000-\overline{r})}{8000-r_\phi}, \phi) & \text{anders.} \end{cases} \tag{2}$$

In feite schalen we hiermee in iedere scanline ($\phi$ =constant) het deel links en het deel rechts van $(r_\phi, \phi)$ lineair zodat $(r_\phi, \phi)$ op de positie $(\overline{r}, \phi)$ komt te liggen. Merk op dat het deel links van $(r_\phi, \phi)$ dus met een andere factor wordt geschaald dan het deel rechts, daardoor blijft de lengte van de scanline 8000.



(a) voor

(b) na

Fig. 12: Eén jaarring wordt rechtgetrokken, in (a) de afbeelding voor en in (b) na transformatie (2).

Een soortgelijke transformatie bestaat ook voor het rechttrekken van $n+2$ ringen: $c_0, .., c_{n+1}$. Stel nu dat ring $c_i$ bestaat uit de punten $(r^i_0, 0), ..., (r^i_{799}, 799)$ voor $i \in [1..n]$ en $\overline{r^i}$ het gemiddelde van $r^i_0, ..., r^i_{799}$ is.

Daarin zijn $c_0$ en $c_{n+1}$ twee kunstmatige ringen, namelijk $c_0 = (0_r, 0_\phi), (0_r, 1_\phi), .., 0_r, 799_\phi)$ en $c_{n+1} = (8000_r, 0_\phi), (8000_r, 1_\phi), .., 8000_r, 799_\phi)$. Daardoor heeft iedere pixel in $(\dot{G}, \dot{R})$ een

ring links en rechts liggen. Voor alle ringen geldt dat ze elkaar niet snijden. Alle ringen zijn tevens gesorteerd naar positie op de $r$-as. Dat kan omdat de ringen elkaar niet snijden (en continu zijn), waardoor een ring of geheel *links* of geheel *rechts* van een andere ring ligt. We transformeren nu $\dot{G}$ naar een $\tilde{G}$ waarin $c_0, ..., c_{n+1}$ perfecte cirkels zijn:

$$\tilde{G}(r,\phi) = \dot{G}(r^1_\phi + \frac{(r-\overline{r^1})\cdot(\overline{r^2}-\overline{r^1})}{r^2_\phi - r^1_\phi}, \phi) \text{ als } \overline{r^1} \leq r < \overline{r^2} \tag{3}$$

Nu weten we hoe we $\dot{G}$ kunnen transformeren, maar we willen ook $\dot{R}$ transformeren. Uiteraard kunnen we precies dezelfde transformatie gebruiken op $\dot{R}$, zodat $(\tilde{G}, \tilde{R})$ ontstaat.

In principe kan de gebruiker een aantal ringen invoeren, om vervolgens zo'n transformatie uit te voeren. Een gebruiker hoeft slechts een paar jaarringen te markeren, en niet alle. In dat laatste geval zou het uiteindelijk bepalen van de jaarringen een flauwe exercitie zijn.

## 4.3 Ringen bepalen

Het is echter ook mogelijk om deze ringen automatisch te berekenen. De methode daarvoor die ik ga bespreken zoekt in eerste instantie niet naar hele jaarringen, maar naar aangesloten delen van jaarringen. Daardoor vind het deelringen. Uiteindelijk worden die delen geëxtrapoleerd, zodat ze ringen vormen waarmee we $(\dot{G}, \dot{R})$ kunnen transformeren. De gevonden ringen benaderen niet altijd even goed een echte jaarring. Een gevonden ring kan bijvoorbeeld tussen twee jaarringen in liggen. Zo'n ring neemt dan wel de vorm van de jaarring aan. De methode bestaat uit een aantal stappen. Deze stappen zijn te zien in figuur 13.

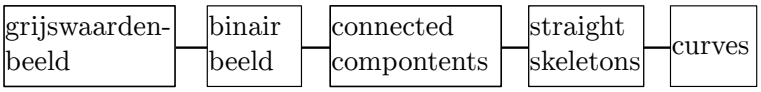



Fig. 13: De stappen voor het bepalen van de curves om $(\dot{G}, \dot{R})$ te bewerken.

In de eerste stap worden het grijswaarden-signaal omgezet naar een binair signaal. Het idee is dat op die manier connected components ontstaan. Wanneer de connected components een *slurfachtige* vorm hebben en jaarringen benaderen, kunnen we die gebruiken om op hun beurt curves te benaderen. Dit binaire signaal verkrijgen we door eerst de gebieden te markeren waar een lichte ring eindigt en een donkere ring begint. Dat kan aan de hand van een sobel-filter in de $r$-richting.

$$S = \begin{array}{c} \\ \phi \\ \downarrow \end{array} \begin{array}{c} \begin{array}{ccc} r & \rightarrow & \end{array} \\ \begin{pmatrix} -\frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & 0 & \frac{1}{4} \end{pmatrix} \end{array} \qquad \text{sobel matrix}$$

$$\dot{H} = |\dot{G} * S|$$

Het resultaat hiervan staat in figuur 14a. In deze figuur zien we op de posities van de ringen een dal of heuvel. Van zo'n dal of heuvel willen we een connected component maken, door het te verscherpen. Op die manier wordt de rand tussen het zwart van de achtergrond en het wit van de voorgrond scherper. De methode die het systeem hiervoor gebruikt wordt de *Laplace-methode* genoemd. Deze methode berekent

$$\begin{aligned} \dot{I} &= \frac{\partial^2 \dot{H}}{\partial \phi^2} + \frac{\partial^2 \dot{H}}{\partial r^2} && \text{met} \\ \frac{\partial^2 \dot{H}}{\partial r^2} &= \tilde{G} * S_{11,r} && \text{en} \\ \frac{\partial^2 \dot{H}}{\partial \phi^2} &= \tilde{G} * S_{11,\phi}, && \text{met} \end{aligned}$$

$$S_{11,r} = \begin{pmatrix} -1 & -8 & -27 & -48 & -42 & 0\ 42 & 48 & 27 & 8 & 1 \\ -10 & -80 & -270 & -480 & -420 & 0\ 420 & 480 & 270 & 80 & 10 \\ -45 & -360 & -1215 & -2160 & -1890 & 0\ 1890 & 2160 & 1215 & 360 & 45 \\ -120 & -960 & -3240 & -5760 & -5040 & 0\ 5040 & 5760 & 3240 & 960 & 120 \\ -210 & -1680 & -5670 & -10080 & -8820 & 0\ 8820 & 10080 & 5670 & 1680 & 210 \\ -252 & -2016 & -6804 & -12096 & -10584 & 0\ 10584 & 12096 & 6804 & 2016 & 252 \\ -210 & -1680 & -5670 & -10080 & -8820 & 0\ 8820 & 10080 & 5670 & 1680 & 210 \\ -120 & -960 & -3240 & -5760 & -5040 & 0\ 5040 & 5760 & 3240 & 960 & 120 \\ -45 & -360 & -1215 & -2160 & -1890 & 0\ 1890 & 2160 & 1215 & 360 & 45 \\ -10 & -80 & -270 & -480 & -420 & 0\ 420 & 480 & 270 & 80 & 10 \\ -1 & -8 & -27 & -48 & -42 & 0\ 42 & 48 & 27 & 8 & 1 \end{pmatrix}$$

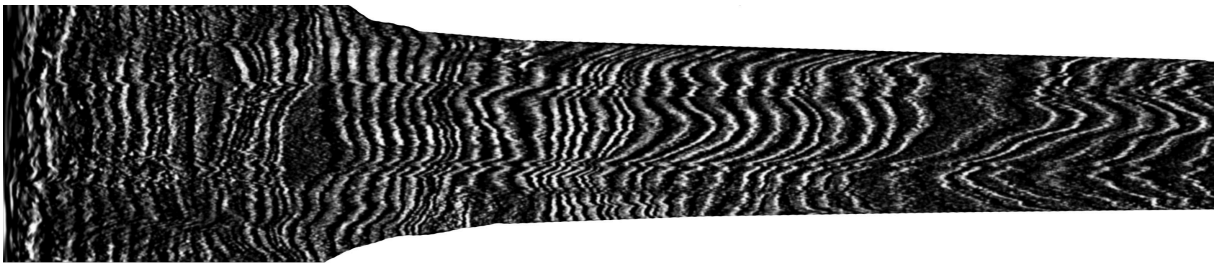
en $S_{11,\phi} = S_{11,r}^T$.

Het systeem berekent deze met de Laplacian methode van de opencv-bibliotheek (zie [1]). Het resultaat is te zien in figuur 14b.

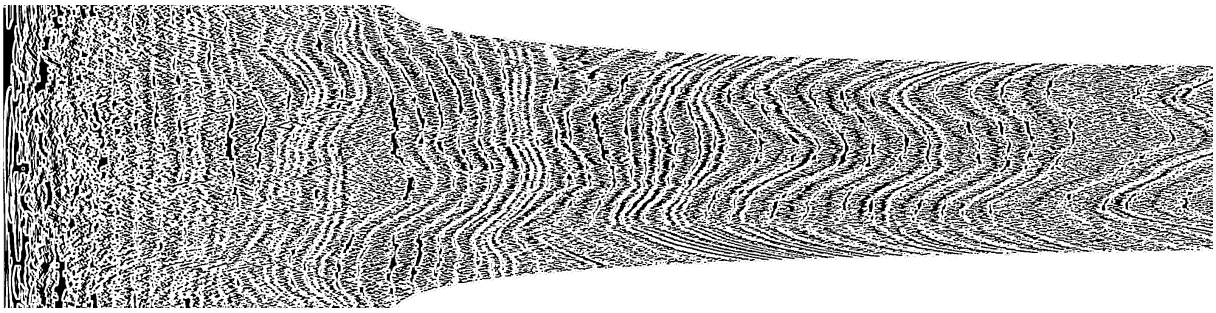
Wanneer we de laplace zodanig schalen dat $\dot{I}(r, \phi) \in [0, 1]$, dan is $\dot{I}(r, \phi)$ alsnog geen binair signaal, al blijkt dat de meeste waarden wel in de buurt van 0 of 1 liggen. Daarom nemen we een threshold van $\dot{I}$ op 0.5. Het resultaat, $\dot{J}$, is dan een binair signaal.

Nu we eindelijk een binair signaal hebben, kunnen we daarin connected components zoeken. In figuur 14c staan een aantal connected components met kleur gemarkeerd. Dit zijn de *grotere* connected components, en de essentie is dat ze de richting van de jaarringen aannemen. Het systeem gebruikt alleen de connected components die groter zijn dan 400 pixels. Ondanks dat op het oog goed te zien is dat de connected components een curve-achtige vorm hebben, zijn het geen curves waarmee we wat kunnen. De ringen, die gebruikt worden bij het rechttrekken, zijn immers lijsten van punten, en geen connected components.

Om zo'n connected component te benaderen met een curve, kunnen we een zogenaamd *straight skeleton* (zie [2]) gebruiken. De boost-bibliotheek bevat hier een methode voor, namelijk: create_interior_straight_skeleton_2 (zie [4]) om zo'n skelet voor een connected component te berekenen. Zo'n skelet, feitelijk een boom-datastructuur zonder expliciete *root*, is dan een topologisch skelet: iedere knoop in het skelet is een coördinaat. De knopen met orde 1 bepalen dan de hoekpunten van het connected component. In figuur 15 staat zo'n topologisch skelet.



(a) sobel



(b) laplace

(c) laplace (uitvergroot)

Fig. 14: In (a) sobel-convolutie van 10 (geïnverteerd), in (b) een laplace van (a) met een kernelgrootte van 11. In (c) staat (b) uitvergroot, met sommige connected components gemarkeerd.

Omdat het berekenen van een straight skeleton qua rekentijd afhankelijk is van het aantal hoekpunten van een connected component, vereenvoudigen we een connected component eerst tot een veelhoek met minder hoekpunten dan het connected component. Deze veelhoek verschilt weliswaar van de connected component, maar vooral in die zijn dat grillige uitsteeksels aan de rand benaderd worden met een rechte curve. In principe bevatten deze uitsteeksels geen informatie over de jaarring, en zijn ze als ruis te beschouwen.

Het algoritme dat het systeem voor het vereenvoudigen gebruikt is het algoritme uit [6]. Dat algoritme vereenvoudigd een *lineaire spline*, een veelhoek maar dan niet gesloten, door het aantal hoekpunten te verminderen. Dat doet het door de spline eerst te benaderen met een rechte van het beginpunt tot het eindpunt van de spline. Vervolgens zoekt het algoritme naar het punt op de spline dat het verst van de spline af ligt. Als dat punt verder dan een bepaalde tolerantie van de spline afligt, dan moet de benadering preciezer gemaakt worden.

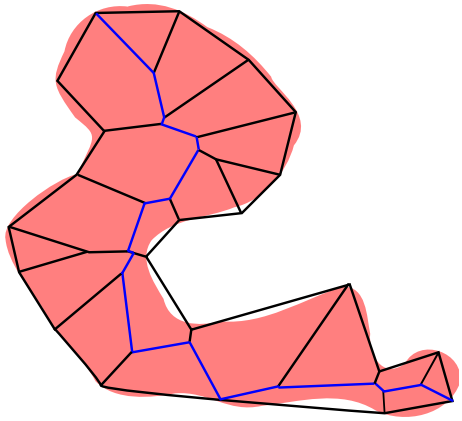


Fig. 15: Het rode gebied is een connected component die wordt benaderd met een veelhoek. In de veelhoek is een topologisch skelet getekend met daarin de langste curve, de *backbone*.

Dat doet het algoritme door de spline te benaderen met twee rechten, een van het beginpunt naar het *verste punt* en een van het verste punt naar het eindpunt. Daarna gebeurt dit recursief over de twee *deelbenaderingen* totdat de afstand van het verste punt wel kleiner is dan de tolerantie. Dit algoritme kan een veelhoek vereenvoudigen door het als lineaire spline te beschouwen met als begin- en eindpunt, twee aangrenzende hoekpunten. Daarmee wordt alleen het lijnstuk tussen die twee hoekpunten niet vereenvoudigd. Desalniettemin is de veelhoek eenvoudiger en kunnen we er dus sneller een straight skeleton voor berekenen.

De *tolerantie* die het systeem gebruik is 10.0. Zo'n topologisch skelet is alsnog geen curve, maar we kunnen wel zoeken naar een curve in zo'n skelet. Het idee daarvan is dat we het langste pad in het skelet gebruiken als curve, dat is de *backbone*. In figuur 15 staat een voorbeeld van een topologisch skelet met zo'n backbone. De backbone loopt niet per sé door het midden van de connected component, maar is desalniettemin toch een aardige benadering.

In figuur 16 staan de straight skeletons voor zulke connected components. Hieronder, algoritme 1, staat een algoritme om zo'n backbone te vinden. Dit algoritme verwijdert net zo lang externe randen van het skelet, tot een pad overblijft. Daarna zoekt hij aan beide uiteinden, met *depth first search* het langste pad naar de buitenkant. Deze twee paden voegt het algoritme weer toe, en het totaal vormt het langste pad in het skelet.

In principe hebben we, wanneer we voor alle connected components op deze manier een curve berekenen, een begin voor het bepalen van de ringen waarmee we kunnen rekken. De curves zijn in feite lijsten van punten, maar ze zijn nog niet geschikt om mee te rekken. Eén zo'n curve is bijvoorbeeld slechts voor een deel van $\phi \in [0..800)$ gedefinieerd, een connected component hoeft immers niet de hele $\phi$-as te beslaan. Figuur 16 laat ook zien dat geen van de rode curves helemaal van boven naar helemaal beneden gaat. Voor het deel van de $\phi$-as

**Algoritme 1 algoritme om het langste pad in een straight skeleton te vinden.**

```
proc longestPath (
        N : ℕ²,                          {* nodes of the tree *}
        E : ℕ² × ℕ²,                     {* edges of the tree *}
        w : E → ℝ,                       {* weight of each edge *}
) : list of ℕ²                           {* ordered set of coordinates of the longest path in the tree *}
        S:=[]
        {* Remove external edges from E, untill there are only 2 *}
        {* nodes with degree 1, and all others have degree 2. *}
        while #{n|n ∈ N, degree_E(n) ≠ 2} > 2
                for all n ∈ N with degree_E(n) = 1
                        T := edges in E adjacent to n
                        append T to S
                        remove T from E
                end
        end
        {* Since there are now only two nodes with degree 1, *}
        {* E defines a path. *}
        for all n ∈ N with degree_E(n) = 1
                p := according to w longest path from n using S {* depth first search *}
                remove p from S
                add p to E
                {* Adding these vertices to an external node does not change *}
                {* that exactly two nodes have degree 1, *}
        end
        {* Now none of the edges in S can be appended to E, and *}
        {* since exactly two nodes have degree_E 1, E defines *}
        {* a path. *}
        return E as a path
end
```

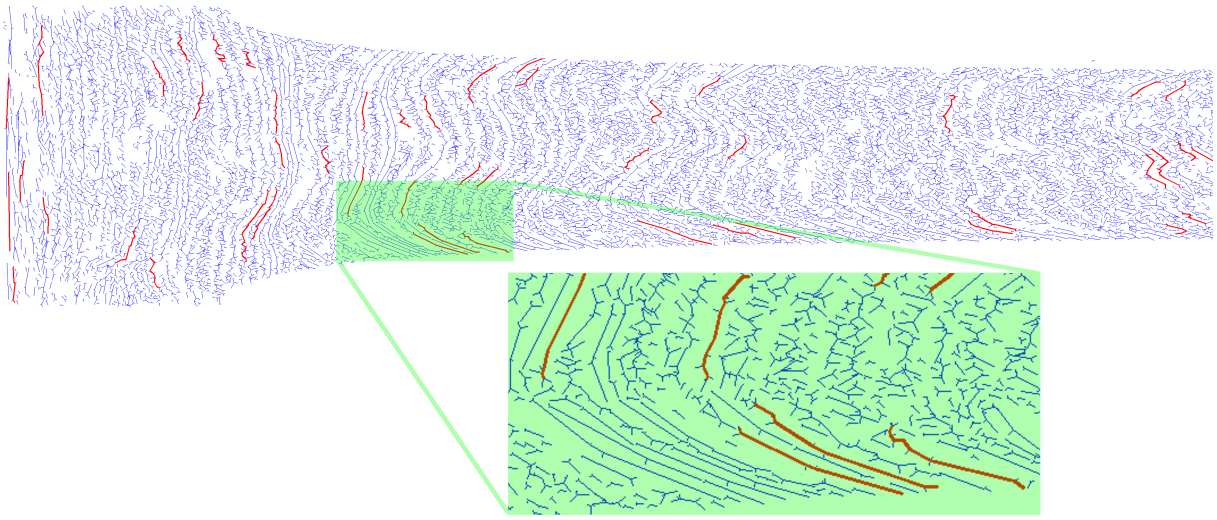


Fig. 16: Straight skeletons voor de connected components uit figuur 14. In sommige skeletten is de backbone rood gemarkeerd.

waar de curve wel bestaat hoeft hij ook niet precies één punt per $\phi \in [0..800)$ te hebben. In figuur 17 staat zo'n curve. In het eerste plaatje kunnen we bijvoorbeeld een lijn $\phi$ =constant bepalen die de curve 2 keer snijdt.

Dat laatste probleem is eenvoudig op te lossen door het langste deelpad, zodat zo'n lijn maximaal 1 keer snijdt. Figuur 17 ligt dit toe. Eerst wordt het langste deel van de curve genomen dat slechts één richting op gaat. Op die manier is er voor een $\phi$ maximaal één punt $(r, \phi)$ op de curve. Daarna wordt de curve geïnterpoleerd, zodat hij voor iedere $\phi$ precies één punt $(r, \phi)$ heeft dat op de curve ligt. Het resultaat is een deelring: een ring die voor een deel $\phi \in [a..b) \subset [0..800)$ bestaat.

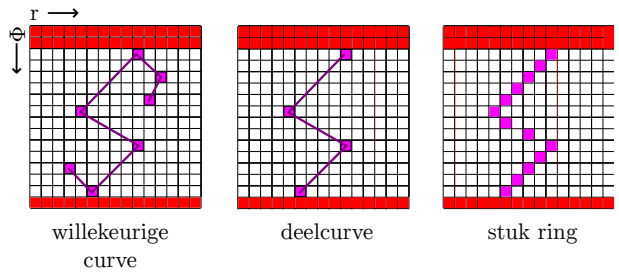



Fig. 17: Nadat een skelet is benaderd met een curve, wordt de curve aangepast om per $\phi$ maximaal één punt $(r, \phi)$ te hebben dat op de curve ligt.

Dat zo'n deelring voor iedere $\phi \in [0..800)$ één punt $(r, \phi)$ heeft is niet helemaal waar. Er is immers nog een gebied boven en onder de deelring, waarvoor we nog niets hebben. In figuur 17 zijn de scanlines waar de deelring niet komt rood gemarkeerd. Door de deelringen te extrapoleren wordt dit opgelost, dat bespreek ik later.

Allereerst wordt een deel van alle deelringen verworpen. Van twee curves die minder dan 100 pixels bij elkaar vandaan liggen wordt er één verworpen. Dat is degene met de kleinste deviatie van $\phi$. Op die manier blijven deelringen met de meeste deviatie van $\phi$ over die minstens 100 pixels van elkaar verwijderd zijn. Dit doet het systeem, omdat deelringen die

dicht bij elkaar liggen bij het rekken vreemd gedrag kunnen veroorzaken. Bijvoorbeeld dat lokaal erg sterk gerekt wordt.

Het laatste probleem is dan het extrapoleren. In principe kunnen we de ringen naar boven en naar beneden gewoon rechtdoor verder laten gaan. Dat is echter onwenselijk, daarmee stellen we dat op die plekken de jaarringen wel op perfecte cirkels lijken, en dat weten we in feite niet. Daarom bespreek ik hier een methode die alle deelringen extrapoleert zonder zulke informatie toe te voegen.

Stel dat we nu de deelringen $d_1, .., d_n$ hebben, om die te extrapoleren voegen we eerst twee ringen toe aan de verzameling van deelringen, namelijk $c_0$ en $c_{n+1}$. Dat zijn de twee kunstmatige ringen die we in (3) hebben gebruikt. Daardoor weten we dat voor iedere $(r, \phi) \in (0..8000-1) \times (0..800-1)$ er minstens één punt $(r_{links}, \phi)$ en één punt $(r_{rechts}, \phi)$ bestaan die beide op een deelring liggen en waarvoor geldt $r_{links} < r < r_{rechts}$. Met andere woorden, ieder punt heeft een deelcurve die er links of rechts van ligt.

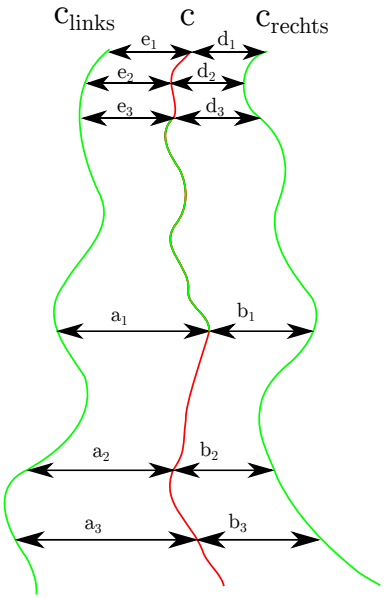



Fig. 18: Extrapolatie van de middelste groene curve, waardoor het rode deel ontstaat. Telkens geldt dat $\frac{c_i}{d_i} = \frac{c_j}{d_j}$ en $\frac{a_i}{b_i} = \frac{a_j}{b_j}$ voor $i, j \in [1..3]$.

De volgorde waarop ringstukken worden geëxtrapoleerd maakt, op numerieke fouten na, niet uit. Tevens kan een curve niet zomaar een grote verschuiving op de $r$-as maken doordat de linker of rechter curve die gebruikt wordt om te extrapoleren een andere is dan de linker of rechter curve die gebruikt werd voor de vorige geëxtrapoleerde pixel. Met 'verschuiving' bedoel ik hier het absolute verschil tussen $r_\phi$ en $r_{\phi+1}$, dit is een lokale eigenschap van een curve.

Dat de verschuiving niet onverwacht heel groot is komt door de eis voor $c_{links}$ en $c_{rechts}$ dat ze bestaan in twee scanlines, namelijk: die van de laatst bekende pixel van $c$ en die waar we een volgende pixel willen extrapoleren (zie figuur 19). Dan volgt dat de grootte van de verschuiving van $r_{new}$ ten opzichte van $r_{last}$ afhankelijk is van de verschuiving van $r^1_{links}$ t.o.v. $r^2_{links}$ én van de verschuiving van $r^1_{rechts}$ t.o.v. $r^2_{rechts}$. Met andere woorden, de verschuiving in de geëxtrapoleerde curve is afhankelijk van de (lokale) verschuiving in de linker en rechter curve, en niet van het plotseling gebruiken van andere curves voor de extrapolatie.

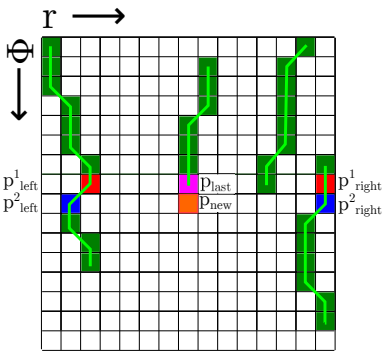
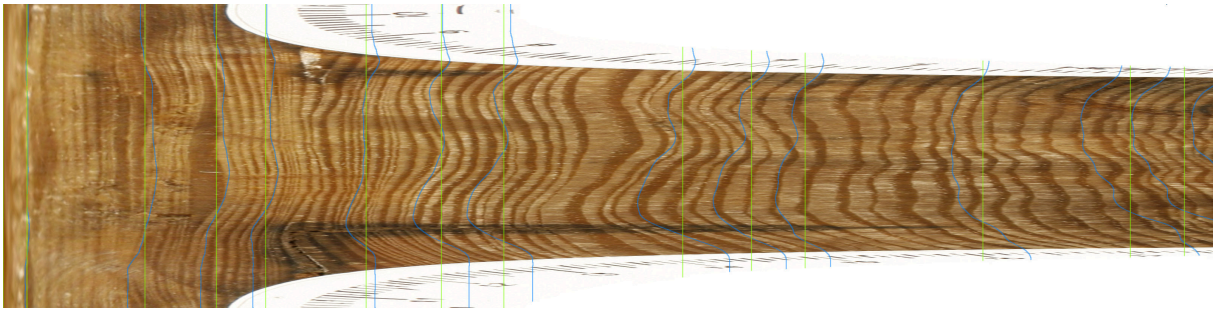



Fig. 19: Er geldt: $\frac{r^1_{rechts} - r^1_{links}}{r^2_{rechts} - r^2_{links}} = \frac{r_{last} - r_{links}}{r_{new} - r^2_{links}}$



(a) niet rechttrekken: $(\dot{G}, \dot{R})$

(b) wel rechtgetrokken: $(\tilde{G}, \tilde{R})$

Fig. 20: De blauwe curves in (a) geven de richting van de ringen aan die worden rechtgetrokken, de groene curves in (a) zijn parallel aan de $\phi$-as. De blauwe curves worden rechtgetrokken naar de groene curves, het resultaat daarvan wordt in (b) weergegeven.

In figuur 18 staat zo'n deelring $c$, aan de hand van $c_{rechts}$ en $c_{links}$ zijn de rode delen geëxtrapoleerd. De essentie is dat de verhoudingen tussen $a_i$'s en $b_i$'s hetzelfde zijn, dus: $\frac{a_i}{b_i} = \frac{a_j}{b_j}$. Hetzelfde geldt voor $d_i$'s en $e_i$'s. In principe kunnen voor verschillende pixels op de rode curves verschillende $c_{links}$ en $c_{rechts}$ gebruikt worden, bijvoorbeeld omdat een $c_{links}$ of $c_{rechts}$ ook niet over het hele $\phi$-domein bestaat.

Na de extrapolatie zijn ringen ontstaan die we voor het rekken kunnen gebruiken. In figuur 20a staat een voorbeeld van zulke ringen, gevonden voor $(\dot{G}, \dot{R})$, in figuur 20b staat het resultaat van het rekken met die curves.

# 5 Seamcarving

In deze sectie zal ik de *essentie* van het proces bespreken, namelijk het berekenen van de ringen. Dit gebeurt door een *enorme* verzameling van paden te bepalen die van de bovenkant van de invoerafbeelding naar beneden gaan. In de vorige sectie werd dit al besproken. De conclusie was dat een ring een geordende verzameling poolcoördinaten is: $\{(r_0, 0), (r_1, 1), \ldots, (r_{799}, 799)\}$. Daarvoor geldt tevens $|\frac{\Delta r}{\Delta \phi}| \leq N_\Delta$. Nu zal ik bespreken op welke manier het systeem $r_0, r_1, \ldots, r_{799}$ bepaalt. Dat gebeurt met het *seamcarving*-algoritme.

## 5.1 Seamcarving-algoritme

Het seamcarving-algoritme (zie [3]) zoekt *seams* in een grijswaarden beeld. Zo'n seam is dan een curve van de linker- naar de rechterzijde, van de boven- naar de benedenzijde of precies andersom. Zo'n seam die van boven naar beneden gaat is dan precies wat ik eerder een ring noemde. Omdat zo'n seam dan dus voor iedere scanline één punt heeft kun je door al die punten te verwijderen het beeld een pixel minder breed maken. Wanneer het seamcarving-algoritme dan zo is ingesteld dat de seam objecten in het beeld ontwijkt, kan op die manier een afbeelding in de breedte geschaald worden. Immers kan een willekeurig aantal seams verwijderd worden. Dit heet *content aware image resizing*: de verwijderde pixels liggen in gebieden waarvan verwacht wordt dat ze geen *belangrijke informatie* bevatten. Dat is waar seamcarving veelal voor gebruikt wordt.

In ons geval stellen we het algoritme zo in dat een seam een jaarring benaderd en verwijderen we de pixels van een seam niet. Ik zal eerst uitleggen hoe seamcarving werkt en daarna uitleggen hoe we het instellen om jaarringen te vinden. We gaan ervan uit dat het algoritme een ring van boven naar beneden probeert te vinden. Dan gaat dit algoritme van een energiebeeld $E$ uit, in feite is dat een beeld met grijswaarden. Dit beeld kan van alles zijn, bijvoorbeeld de gradiënt van een beeld.

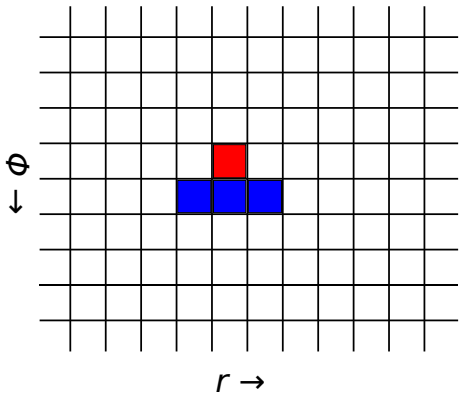



Fig. 21: De blauwe pixels zijn de mogelijke pixels die volgen op de rode pixel met $W = 1$.

## 5.2 Energiebeeld

In dit energiebeeld probeert het algoritme dan een ring te bepalen zodat de integraal van die ring over $E$ minimaal is. Voor een gevonden ring $r$ wil dat zeggen dat $\sum_{(x,y)\in r} E(x, y)$ minimaal is. In principe bevat die ring uit iedere scanline het punt met de minimale waarde voor $E$. Dan is het bijvoorbeeld mogelijk dat een ring gelijk is aan $..(r_\phi, \phi), (r_{\phi+1}, \phi + 1)..$ met $|r_\phi - r_{\phi+1}|$ erg groot, bijvoorbeeld 100. Dat is ongewenst, want in een jaarring komt zoiets niet voor. Daarom legt het algoritme een beperking $|r_\phi - r_{\phi+1}| < W$ op, met in ons geval $W = 1$. Omdat de afbeelding in de vorige sectie is rechtgetrokken, moet dit voldoende

*beweegruimte* zijn voor een (rechtgetrokken) jaarring. In figuur 21 staan de pixels die kunnen volgen op een willekeurige pixel in zo'n ring.

Om dan de *goedkoopste* ring te vinden moeten we iets meer moeite doen. We bewerken eerst $E$ dusdanig dat we een beeld $M$ krijgen waar op ieder punt de energie van het minimale pad tussen de onderste scanline en dat punt staat. De onderste scanline van $M$ is dan gelijk aan de onderste scanline van $E$.

Voor alle andere scanlines betekent dat:

$$M(x,y) = E(x,y) + \min\{kernel(n) \cdot E(x+n, y-1) : n \in [-W..W]\}$$

Daarin schaalt $kernel(n)$ het signaal afhankelijk van wat de daadwerkelijke waarde van $|r_\phi - r_{\phi+1}|$. Bijvoorbeeld met $kernel(n) = \sqrt{n^2+1}$ wordt de integraal van het pad tevens bepaald door de afstand tussen twee opvolgende punten (pixels) op de ring. In principe kan zo'n kernel alles zijn, echter zorgt deze keuze voor een voorkeur voor een recht pad, waar $r_\phi - r_{\phi+1} = 0$.

Aan de hand van $M$ kan vervolgens een ring worden bepaald met een minimale integraal, namelijk door te beginnen met de pixel in de bovenste scanline die in $M$ de kleinste waarde heeft. Die pixel is dan $(r_0, 0)$ op de ring. Vervolgens is de pixels die volgt op $(r_\phi, \phi)$ $(\phi < 799)$ een van de volgende: $(r_\phi - 1, \phi + 1)$, $(r_\phi, \phi + 1)$ en $(r_\phi + 1, \phi + 1)$ (zie figuur 21). Namelijk die met de laagste waarde in $M$. Op die manier kan recursief een pad bepaald worden. In principe kan voor iedere pixel in de eerste scanline zo'n pad bepaald worden, in plaats van enkel op het minimum. Dat gebruiken we om de jaarringen te vinden. We hebben immers niet voldoende aan 1 jaarring, de boom heeft er meer.

## 5.3 Ringen bepalen

Het algoritme dat ik net beschreef werkt prima om één of meerdere kolommen van een beeld te verwijderen, maar dat is niet het probleem dat we aan het oplossen zijn. Daarnaast moeten we een energie-signaal bepalen die ervoor zorgt dat het jaarringen vind, en niet willekeurige ringen. Dat bespreek ik nu.

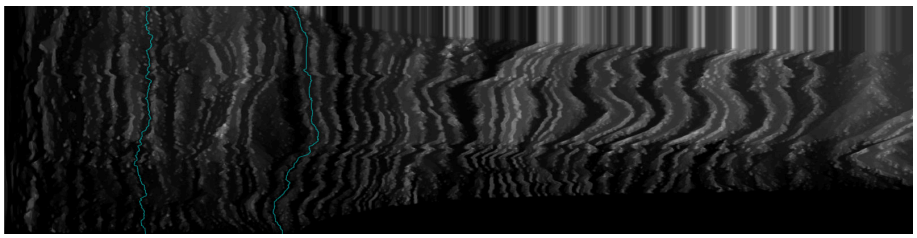


Fig. 22: Het energiesignaal $\tilde{M}$. De blauwe curves die een minimale integraal hebben.

De jaarringen, zoals we ze ook kunnen zien in figuur 20b, zijn dan wel donker, dan wel licht. Het algoritme moet een ring vinden die op een rand tussen een donkere en lichte ring gaat. Daarom voeren we over dit uitgerekte signaal een sobel uit:

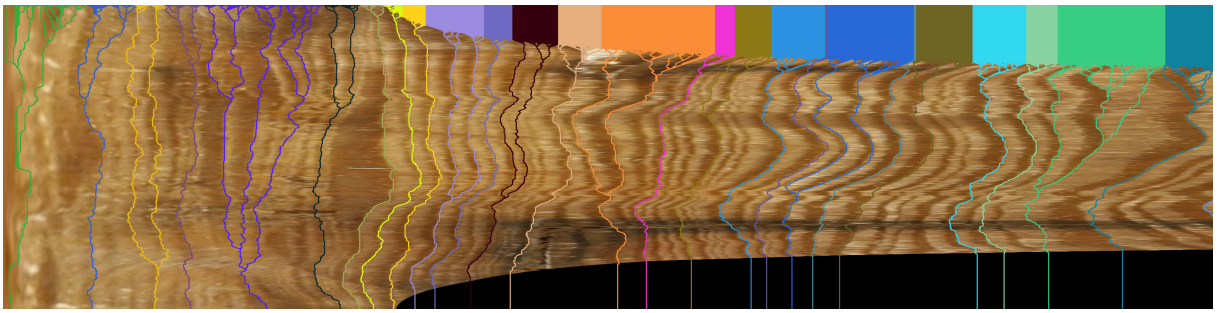
$$S = \begin{array}{c} \\ \phi \\ \downarrow \end{array} \begin{array}{c} \begin{array}{ccc} r & & \rightarrow \end{array} \\ \begin{pmatrix} -\frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & 0 & \frac{1}{4} \end{pmatrix} \end{array}$$

$$\tilde{E} = |\tilde{G} * S|$$

Op die manier kan het systeem werken met $(\tilde{E}, \tilde{R})$. Deze beelden hebben op de randen tussen donkere en lichte ringen een hoge of juist lage (negatieve) waarde. Alle andere gebieden, die een constantere waarde hebben, zijn in $\tilde{E}$ in de buurt van 0. Op een rand hebben die aan de linker kant donker is en aan de rechter kant licht (type 2), is $\tilde{E}$ groter. Omgekeerd is $\tilde{E}$ kleiner (negatiever) op een rand die aan de linker kant licht is en de rechter donker (type 1). Daardoor vind het algoritme met deze $\tilde{E}$ dus de jaarringen van type 1 en met $-\tilde{E}$ van type 2. In figuur 22 staat vervolgens de bewerking van $\tilde{E}$, namelijk $\tilde{M}$.

## 5.4 Ring uitzoeken

Het signaal $\tilde{E}$ is makkelijk te berekenen, daarmee kunnen we meteen voor een willekeurig punt $(r_0, 0)$ in de eerste scanline een ring bepalen. Wanneer we dat voor ieder punt in de eerste scanline doen hebben we daarmee 8000 ringen. Daarmee hebben we over de gehele $r$-as van onze boomstam paden bepaald die proberen de jaarringen te volgen. Echter heeft een boom waarschijnlijk veel minder dan 8000 jaarringen. Ik zal nu bespreken hoe we bepalen hoeveel dat er dan wel zijn en welke dat zijn.



(a) poolcoördinaten

Fig. 23: Iedere *trechter* is in een kleur getekend.

Stel dat twee ringen een gezamenlijk punt $(r, \phi)$ hebben, dan volgt dat alle punten $(r', \phi')$ met $\phi' > \phi$ op de ene ring ook op de andere ring liggen. Voor beide ringen is de minimale ring van $(r, \phi)$ naar de laatste scanline immers hetzelfde. Stel dat er nog een derde ring tussen deze twee ringen begon, dan volgt dat die ring voor $(r, \phi)$ één van beide ringen moet snijden om bij de laatste scanline te komen, hij wordt immers door beide ringen ingesloten.

Hieruit volgt het *trechtergedrag* van deze ringen, in figuur 23 kunnen we die trechters zien. Een trechter is dan een verzameling ringen die in dezelfde pixel eindigen. In eerste instantie is de kans dat zo'n ring precies op een jaarring ligt vrij klein, immers begint lang niet bij iedere pixel in de eerste scanline een jaarring. Door de keuze van $\tilde{E}$ probeert iedere ring

echter een jaarring te volgen als die er één tegenkomt, daarmee is de kans vrij groot dat zo'n ring uiteindelijk wel een jaarring volgt.

Ergens in zo'n trechter ligt daarom de jaarring die we zoeken. Het systeem bepaalt dan ook voor iedere trechter precies één jaarring. Daardoor is het alsnog mogelijk dat een jaarring niet wordt gevonden. In figuur 24 staat een voorbeeld daarvan. In eerste instantie volgt de oranje trechter twee jaarringen, maar op een gegeven moment (in de rode rechthoek) verspringt het linker deel naar de rechter trechter.

Fig. 24: Twee jaarringen die in 1 trechter terecht gekomen zijn.

Dit gedrag treedt typisch daar op waar een jaarring lokaal te veel afwijkt van een perfecte cirkel of waar de kwaliteit van het beeld afneemt. Bijvoorbeeld door ruis van de camera of doordat het hout daar van slechte kwaliteit is. Het is lastig om een goede manier te vinden die twee jaarringen uit een trechter haalt, omdat beide jaarringen elkaar niet mogen snijden, dat gebeurt in een boom immers ook niet. Alle ringen in een trechter snijden elkaar per definitie. Tevens is in het voorbeeld de informatie over het laatste deel van de ring niet beschikbaar. Dit soort problemen moeten door de gebruiker opgelost worden door bijvoorbeeld handmatig een ring toe te voegen of door van te voren de afbeelding op zulke plaatsen goed recht te trekken (zie vorige sectie).

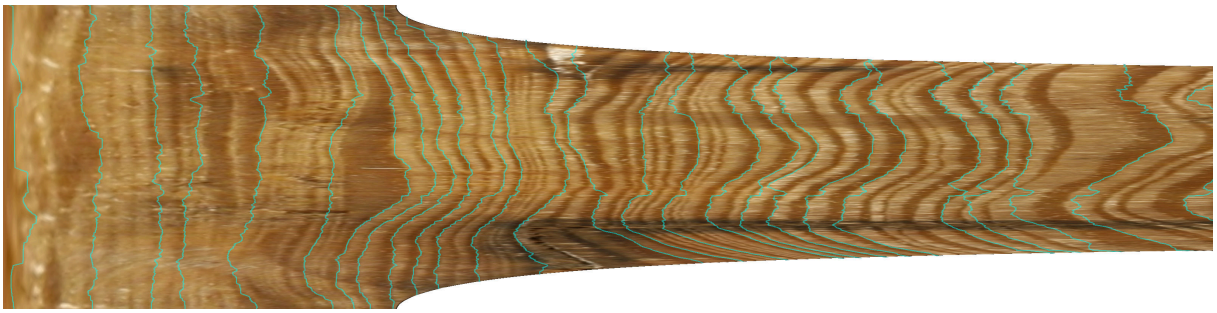
Zoals gezegd selecteert het proces van zo'n groep maximaal één ring, de anderen moet hij verwerpen. Een andere, subtielere mogelijkheid, is dat hij de ringen gebruikt om één ring te bepalen, die niet persé overeenkomt met één van de ringen uit de groep. Mogelijke opties zijn

- de ring met de kleinste integraal,
- de ring waar het absolute verschil van $r$ in de onderste en bovenste scanline minimaal is,
- de ring die op iedere scanline voor $r$ de gemiddelde waarde van alle andere ringen heeft, of
- de ring die begint op het gemiddelde begin van alle andere ringen.

De tweede optie probeert het begin en einde van een ring zoveel mogelijk te laten overeenkomen, zodat een echte ring ontstaat. In ons origineel is de laatste scanline immers een hoek $\phi$ die dicht in de buurt van de hoek van de eerste scanline ligt. Dit is een logische keuze voor het deel van de ringen die in het gebied van de foto liggen waar alle hoeken in de ROI gedefinieerd zijn. Wanneer de doorsnedefoto echter rechthoekig is en een deel van de boomstam daardoor weglaat, dan is niet iedere jaarring voor alle hoeken gedefinieerd. In dat geval zijn de andere opties logischer.

In het deel waar de jaarringen volledig op de foto staan gebruikt het systeem de tweede optie, om zo die ringen goed op elkaar te laten aansluiten. In het deel waar de jaarringen

worden ondebroken wordt de eerste optie gebruikt, omdat de minimale integraal in principe een goede maat moet zijn voor het vinden van een jaarring.



(a) poolcoördinaten

(b) carthesisch

Fig. 25: Oorspronkelijke afbeelding met daarin de gevonden ringen getekend.

Wanneer het proces op deze manier voor iedere trechter één ring bepaalt, ontstaat alsnog een verzameling van ringen. Iedere ring wordt gezien als een numerieke benadering van een jaarring. In figuur 25 staat de oorspronkelijke afbeelding met daarin de gevonden ringen getekend.

# 6 Gebruik

Ik heb voor het gebruik van het systeem een *user interface* gemaakt die een gebruiker in staat stelt om aanpassingen te doen aan de resultaten van berekeningen. Dit is nodig als bijvoorbeeld het hart niet goed wordt bepaald of als niet alle ringen goed worden herkend. Dit proces bestaat naast het openen en opslaan van de in- en uitvoer, uit 4 stappen. Die worden in deze sectie besproken.

## 6.1 Openen en opslaan

Voor de volledigheid laat ik deze stappen ook zien. In figuur 26 staan de schermen voor het selecteren van de in- en uitvoer. De uitvoer heeft de mogelijkheid om de afbeelding op te slaan in poolcoördinaten of carthesische coördinaten.

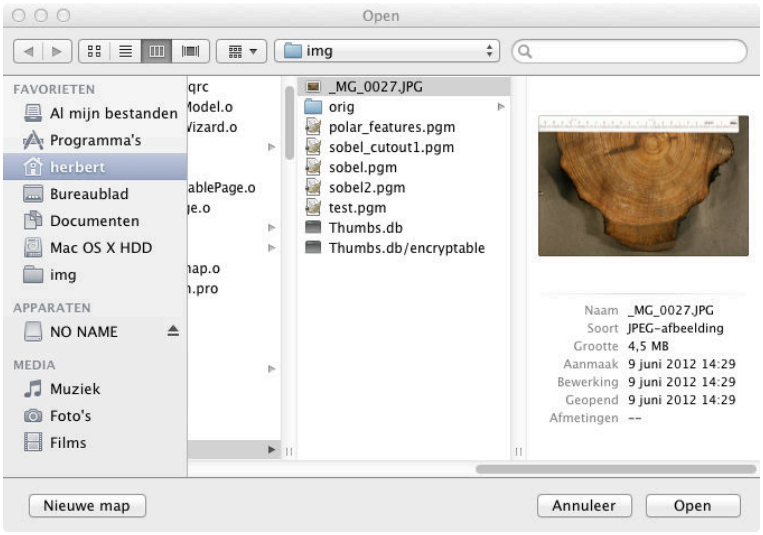



Fig. 26: Het openen van een doorsnedefoto

## 6.2 ROI

Een gebruiker kan na het openen de gevonden *region of interest* aanpassen met een *brush*. Zoals te zien is in figuur 27 kan hij of zij de grootte van een *brush* aanpassen om daarmee vervolgens delen aan de *region of interest* toe te voegen of te verwijderen.

## 6.3 Hart

Als tweede stap kan de gebruiker het hart aangeven. In principe heeft het proces deze berekent, maar de gebruiker kan hem alsnog veranderen. Dat staat in figuur 28

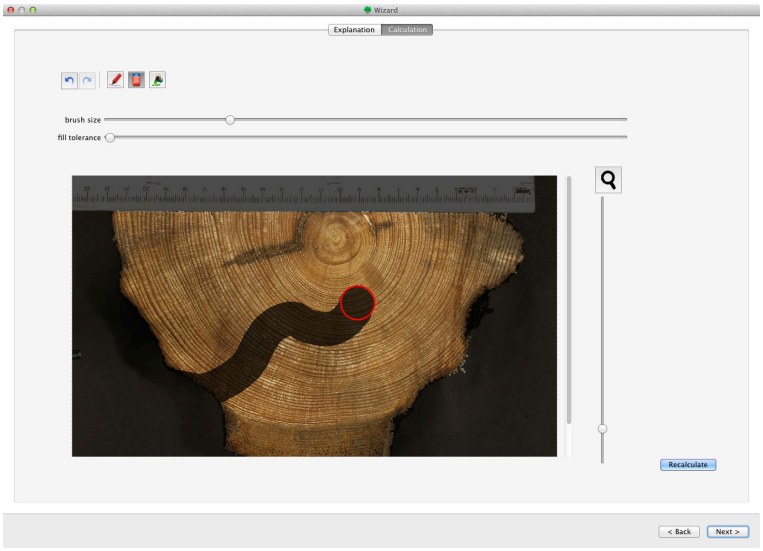



Fig. 27: Het aanpassen van de *region of interest*. Het rode gebied is de gevonden *region of interest*, de rode cirkel is de *brush*. Er is tevens een *fill*-optie.

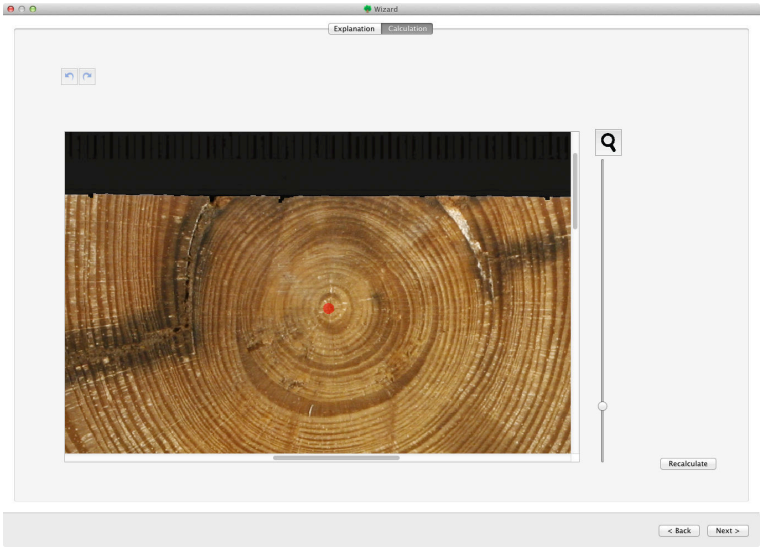
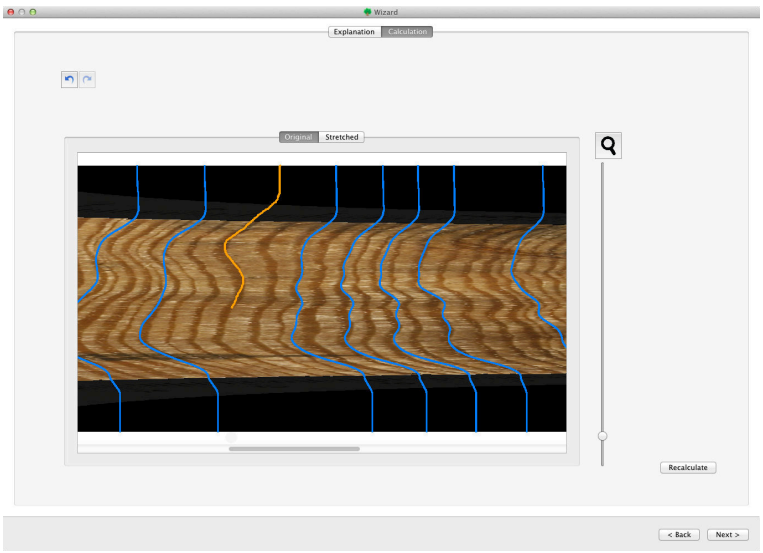



Fig. 28: De gebruiker kan het *hart* handmatig selecteren. De rode stip is het hart die gevonden is door het proces, die kan de gebruiker aanpassen door ergens anders op het beeld te klikken

## 6.4 Rechttrekken

In dit scherm kan de gebruiker twee afbeeldingen bekijken, namelijk die in poolcoördinaten en die tevens is rechtgetrokken. Op deze manier kan hij goed zien of de ringen wel daadwerkelijk rechter worden. Eventueel kan hij of zij daarna ringen verwijderen of toevoegen door ze te tekenen.

Vanaf nu krijgt de gebruiker telkens afbeeldingen in poolcoördinaten te zien. Op deze manier zijn curves die in de richting van de ringen lopen makkelijker te tekenen. Het is immers lastig om een cirkel te tekenen, maar makkelijk om een *min of meer rechte* curve te tekenen.




(a) rechttrekken

Fig. 29: De gebruiker kan curves verwijderen of toevoegen, en tevens het resultaat van het rechttrekken bekijken. De oranje curve werd door de gebruiker getekend.

## 6.5 Ringen

Tenslotte kan de gebruiker ringen toevoegen of verwijderen, als hij of zij het niet (voldoende) eens is met de automatisch gevonden ringen. Ook kan de gebruiker van een ring alleen het onderste of bovenste deel van een ring aanpassen. En een gebruiker kan een ring toevoegen aan de hand van twee andere ringen, dan wel als lineaire combinatie van beide, dan wel als ring die naar boven en naar beneden de minimale integraal volgt vanuit het geselecteerde punt.

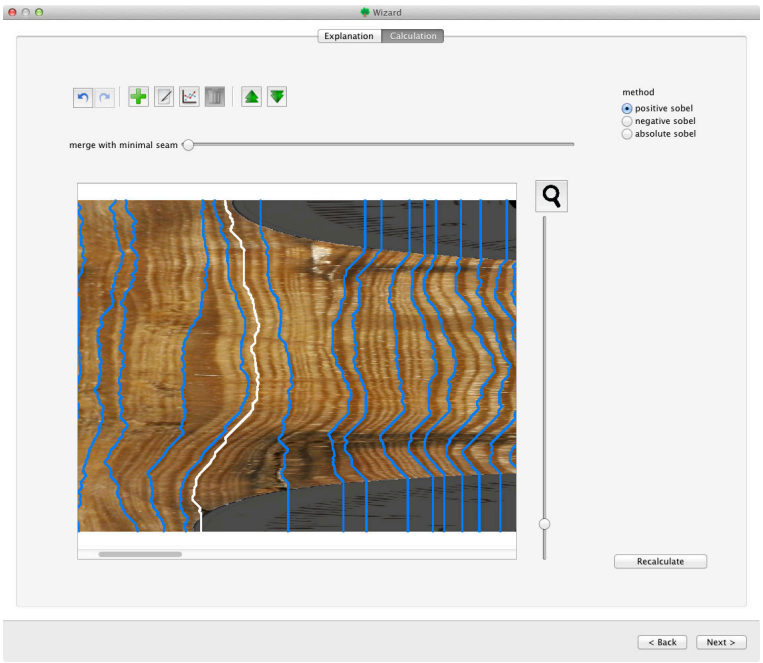



Fig. 30: De oranje ring wordt toegevoegd als lineaire combinatie van de twee buurringen.

# 7 Conclusies

In deze sectie wil ik graag toelichten hoe de verschillende delen van het systeem presteren: het vinden van de ROI, het bepalen van het midden en het vinden van de jaarringen aan de hand van het rechtgetrokken beeld. De resultaten zijn gebasseerd op die van 5 foto's van dezelfde doorsnede.

## 7.1 Gevonden ROI

Deze gevonden ROI hoeft niet per sé voor alle invoerfoto's goed te werken. Bijvoorbeeld doordat de boomstam op een houten tafel ligt. Omdat het bepalen van de ROI gebeurt aan de hand van de kleur van het hout, is het een goed idee om de achtergrond een andere kleur te laten hebben. Het liefst een kleur die niet voorkomt in de stam, bijvoorbeeld blauw. In de gevallen dat de achtergrond niet goed gedetecteerd wordt, moet de gebruiker dat corrigeren.

## 7.2 Gevonden hart

De methode waarmee het aantal snijpunten wordt verminderd om het hart een betrouwbaardere positie te geven is essentieel voor de fout van het gevonden hart ten opzichte van het daadwerkelijke hart. Ten eerste dient de methode om evidente reden te garanderen dat er iedere vermindering minstens 1 snijpunt overblijft. Daarnaast is het ook gevaarlijk als het aantal snijpunten te snel afneemt. In dat geval kan een klein gebied met een hoge dichtheid snel de overgebleven verzameling zijn, terwijl het daadwerkelijke hart elders ligt. Daarom heb ik voor een methode gekozen die het aantal halveert, of net niet halveert. De conclusie is dat het algoritme het hart over het algemeen vrij nauwkeurig bepaald, namelijk met een afwijking van rond de 10 pixels of minder. Op de resolutie van de foto's die ik heb gebruikt is dat verschil net zichtbaar. In alle 5 foto's werd het hart nauwkeurig gevonden.

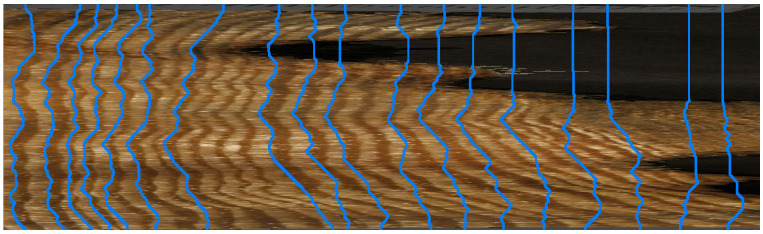
## 7.3 Gevonden jaarringen

Daarnaast worden sommige jaarringen niet gevonden, bijvoorbeeld omdat ze te weinig contrast hebben of omdat er trechters uit sectie 5 onterecht zijn samengevallen. In die gevallen stelt de gebruikersinterface de gebruiker in staat jaarringen toe te voegen.

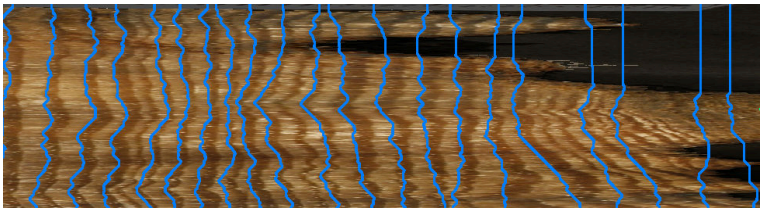
In de doorsnedefoto die ik steeds gebruikt heb, zie figuur 1, worden van de eerste 15 jaarringen er maar 6 gevonden, zowel na handmatig rechttrekken als onrechtgetrokken. In totaal bevat de foto 91 jaarringen, daarvan worden er onrechtgetrokken 45 ringen gevonden, echter zijn 6 van die ringen bijna foutloos, en slechts van 33 van die ringen geldt dat het langste deel van de ring één jaarring volgt.

Wanneer er wel wordt rechtgetrokken worden er 61 ringen van de 91 jaarringen gevonden. Alle 61 ringen volgen voor meer dan de helft een jaarring, en 49 daarvan zelfs geheel.

Of een gevonden jaarring een ring nauwkeurig benaderd is sterk afhankelijk van de *cirkelvormigheid* van zo'n ring. Daarom kan het rechttrekken van het beeld een enorm verschil maken. De curves die het systeem zelf bepaald zijn in zo'n geval al een verbetering, maar als de gebruiker een exactere benadering maakt van ongeveer 10% van de jaarringen kan het



(a) niet rechtgetrokken



(b) met de hand rechtgetrokken

Fig. 31: Het systeem vind met de hand rechtgetrokken ringen aanzienlijk beter, maar mist er alsnog een aantal.

systeem ze veel beter vinden. Figuur 31 laat twee voorbeelden zien, één waar de afbeelding vrij nauwkeurig is rechtgetrokken door de gebruiker en één waar dat niet is gedaan. In het tweede geval verspringen de jaarringen veel minder dan in het eerste geval. Bij het rechttrekken moeten echter vooral lokaal heftige afwijkingen (van cirkels) gevolgd worden, anders is het risico vrij groot dat het systeem daar van ring verspringt.

# 8 Toekomstige uitbreidingen

Tot slot wil ik nog een aantal zaken opsommen die mogelijke verbeteringen zijn van van hetgeen ik in dit verslag verdedig.

## 8.1 Meerkanaalse invoer

Het systeem gaat telkens uit van een beeld dat één kanaal heeft, maar de invoer zal typisch uit 3 of 4 kanalen bestaan. Het is goed mogelijk het contrast tussen donkere en lichte ringen te vergroten door verschillende methoden om meerdere kanalen, typisch kleurkanalen, te combineren te onderzoeken.

## 8.2 Convergentiegedrag hart

De methode om het hart te bepalen is afhankelijk van een verzameling snijpunten die op hun beurt afhankelijk is van een verzameling loodlijnen, die op hun beurt afhankelijk zijn van een verzameling ringstukken. Dat betekent dat we aan de kansverdeling van de afwijking van een ringstuk of loodlijn een kansverdeling zouden kunnen bepalen van de straal tussen zo'n snijpunt en het daadwerkelijke hart de afwijking van een ringstuk is dan de afwijking van die ring van een perfecte cirkel om het daadwerkelijke hart, en de afwijking van een loodlijn is dan de afstand tussen die lijn en het hart.

Daarnaast convergeert het *tijdelijke hart* over de verschillende *epochs* uit sectie 3, dat gedrag is afhankelijk van de spreiding van de snijpunten. Het is dan interessant om te onderzoeken wat de kansverdeling is van de fout tussen het hart waarheen geconvergeerd wordt en het daadwerkelijke hart.

Dit kunnen we zowel aan de hand van een realistische maar verzonnen kansverdeling van de loodlijnen als een daadwerkelijke dataset bekijken.

## 8.3 Ruis verwijderen

In de foto's van de boom die ik heb gezien bevatten veel van de ringen witte vlekjes, omdat de kleur van die vlekjes op zowel de donkere als de lichte ringen hetzelfde is, namelijk wit, is het daar lastig onderscheid maken tussen beide. Het seamcarving-algoritme kan in zo'n geval makkelijk van ring verspringen. Tevens blokkeert het thresholding-technieken om een gehele ring te detecteren. Bij de witte vlekjes wordt een ring dan onderbroken of juist gekoppeld aan andere jaarringen.

Dat is vervelend: een mogelijke oplossing kan bijvoorbeeld een erosie zijn met een langwerpige-kernel over de $\phi$-as. Daarmee wordt een witte vlek teruggebracht naar het minimum met zijn onder- en bovenburen. In een beeld in poolcoördinaten kun je dan stellen dat dat min of meer de kleur is van de ring waar de witte pixel bij hoort. Het nadeel is dat deze methode in het beeld ook harde randen verplaatst en vervaagt.

Een andere vorm van verstoring is simpelweg dat de boom op bepaalde plekken vernield is. In zo'n geval wil je dat kunnen detecteren, en als dat niet lukt, door de gebruiker laten

markeren. Dan worden gevonden ringen in dat gebied in feite onderbroken en is het een interessante vraag hoe je deze ringstukken weer aan elkaar koppelt.

## 8.4 Alternatieve methode voor zoeken naar het hart

Wanneer we de het invoerbeeld, zoals 1, bekijken zien we in om het hart een redelijk grote, circa 7 jaarringen groot, vast lichaam dat ongeveer dezelfde *bruinwaarde* heeft. Waneer we die proberen te vinden door in een *max tree* (zie [7]) te zoeken naar cirkelvormige objecten. Dat kan bijvoorbeeld aan de hand van *shape descriptors*, zoals de *compactness* of *circular variance* (zie [5]). Het midden van die cirkel is dan een kandidaathart.

## 8.5 Meerdere metingen

Wanneer een boom op meerdere punten doorgezaagd wordt kunnen we meerdere doorsnedefoto's maken, en hebben we dus informatie over verschillende doorsnedes van de boom. Daarmee kan het systeem een driedimensionale verzameling van jaaringen bepalen, of de informatie uit naburige doornedes gebruiken om een nauwkeuriger antwoord te vinden.

Voor het bepalen van een driedimensionale jaarring moeten algoritmes verzonnen worden om jaarringen uit de ene doornede te *matchen* met jaarringen uit de andere. Daarbij dienen we dan rekening te houden met jaarringen die in één doorsnede wel zijn gevonden, maar in een andere niet.

Het gebruik voor betere nauwkeurigheid is vrij abstract, dat kan door de doorsnedebeelden te combineren tot één beeld of door de resultaten te combineren.

## 8.6 Verloren jaarringen alsnog vinden

Sommige jaarringen vallen tussen wal en schip, bijvoorbeeld omdat twee trechters (sectie 5) onterecht zijn samengevoegd. Die ringen vindt het systeem niet. Een mogelijkheid om ze toch op te sporen is om de afbeelding aan de hand van de de twee naburige, gevonden ringen recht te trekken en vervolgens in dat rechtgetrokken gebied nogmaals te zoeken naar jaarringen. In principe kan dan weer het seamcarving-algoritme gebruikt worden, in de hoop dat doordat nu de jaarringen meer op cirkels lijken, ze wel gevonden worden.

## 8.7 Jaarringen tellen

Wanneer we jaarringen willen tellen, dan kunnen we gewoon de automatisch gevonden jaarringen tellen. Echter weten we dan vrijwel zeker dan we wel jaarringen missen, maar niet jaarringen te veel tellen. Op de aantal-ringen-as ligt de kansverdeling van het daadwerkelijke aantal jaarringen dus volledig *rechts* van het aantal getelde jaarringen. We kunnen een methode proberen te bepalen die het gemiddelde van die kansverdeling bepaald aan de hand van het gevonden aantal jaarringen, om zo een nauwkeuriger antwoord te bepalen.

## 8.8 Usability test

Ondanks dat er een gebruikersinterface is, is er geen duidelijkheid over de gebruikersvriendelijkheid daarvan. Een onderzoek waar een groep potentiële gebruikers wordt gevraagd bepaalde taken uit te voeren, kan duidelijkheid geven over de problemen waar zulke gebruikers tegenaan lopen.

## Referenties

[1] Website van de opencv-bibliotheek (http://opencv.org/), 2012.

[2] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.

[3] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. In *ACM Transactions on Graphics (TOG)*, volume 26, page 10. ACM, 2007.

[4] Fernando Cacciola. Cgal user and reference manual (http://www.cgal.org/, release 4.0.2), July 2012.

[5] M. Peura and J. Iivarinen. Efficiency of simple shape descriptors. In *Proceedings of the third international workshop on visual form*, pages 443–451. Citeseer, 1997.

[6] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.

[7] P. Salembier, A. Oliveras, and L. Garrido. Antiextensive connected operators for image and sequence processing. *Image Processing, IEEE Transactions on*, 7(4):555–570, 1998.

[8] A.R. Smith. Color gamut transform pairs. In *ACM Siggraph Computer Graphics*, volume 12, pages 12–19. ACM, 1978.