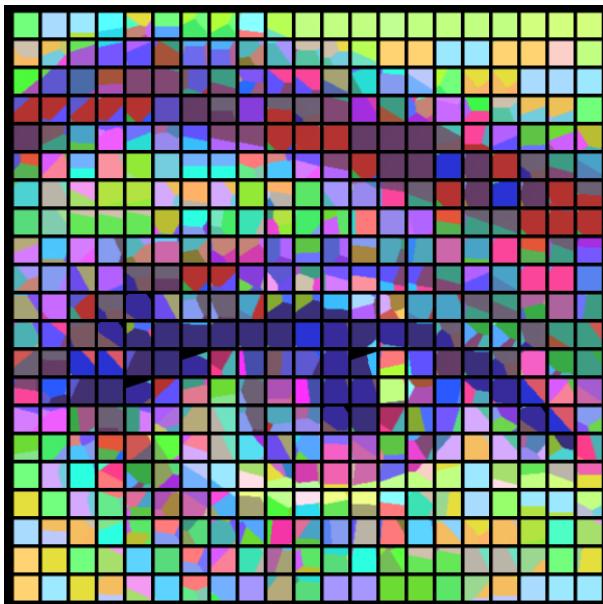


# GridTiles

Stijloverdracht met Mondriaantegels



Promotor: Prof. dr. ir. Philip Dutré  
Begeleider: Prof. dr. ir. Philip Dutré  
Lezer: Ir. Thierry Deruyttere  
Lezer: Dr. ir. Koen Yskout

Proefschrift ingediend tot het  
 behalen van de graad van  
 Master of Science in de Toegepaste Informatica  
 optie Multimedia

Academiejaar 2021-2022

# Voorwoord

Ik wil mijn professor, promotor en begeleider prof. dr. ir. Philip Dutré bedanken voor zijn wekelijkse inzet, voor zijn interessante en motiverende lessen, en voor zijn geduld tijdens de drukkere periodes van het jaar. Daarnaast wil ook Arno Coomans bedanken voor zijn hulp en ideeën in het begin van de thesis en tijdens de presentaties. Ten slotte wil ik de jury bedanken voor het lezen van de tekst.

*Matthias Vandersanden*

# Inhoudstafel

<b>Voorwoord</b>	<b>i</b>
<b>Samenvatting</b>	<b>v</b>
<b>Lijst van figuren</b>	<b>vi</b>
<b>Lijst van tabellen</b>	<b>viii</b>
<b>Lijst van afkortingen en symbolen</b>	<b>ix</b>
<b>1 Inleiding</b>	<b>1</b>
1.1 Achtergrond en motivatie . . . . .	1
1.2 Onderzoeksvragen . . . . .	1
1.3 Contributies . . . . .	2
1.4 Structuur . . . . .	2
<b>2 Achtergrond</b>	<b>3</b>
2.1 Beteigeling . . . . .	3
2.1.1 Mozaïek en parket . . . . .	3
2.1.2 Marqueterie . . . . .	3
2.2 Mondriaan . . . . .	4
2.3 Convolutie filters . . . . .	4
2.3.1 Box filter . . . . .	4
2.3.2 Gaussische blur . . . . .	4
2.3.3 Rolling guidance . . . . .	5
2.3.4 Template matching . . . . .	5
2.4 Randdetectie . . . . .	6
2.4.1 Sobel filter . . . . .	6
2.4.2 Canny filter . . . . .	7
2.5 Histogram egalisatie . . . . .	8
2.6 Saliency detectie . . . . .	9
<b>3 Literatuur</b>	<b>10</b>
3.1 Woodpixel . . . . .	10
3.2 Stijloverdracht . . . . .	10
3.3 Mondriaan . . . . .	12

<b>4 Overzicht</b>	<b>13</b>
4.1 Doel . . . . .	13
4.2 Contributies . . . . .	13
4.3 Input . . . . .	15
4.3.1 Afbeeldingen . . . . .	15
4.3.2 Parameters . . . . .	15
4.4 Preprocessing . . . . .	16
4.4.1 Schalen . . . . .	16
4.4.2 Features . . . . .	16
4.4.3 Masks . . . . .	17
4.5 Generatie . . . . .	17
4.5.1 Seedpoints . . . . .	17
4.5.2 Recursieve gridgeneratie . . . . .	17
4.5.3 Adaptieve gridgeneratie . . . . .	18
4.6 Matching . . . . .	18
4.6.1 Rotaties . . . . .	18
4.6.2 Mask . . . . .	18
<b>5 Preprocessing</b>	<b>20</b>
5.1 Schalering . . . . .	20
5.2 Features . . . . .	22
5.3 Rotaties . . . . .	22
5.4 Histogram egalisatie . . . . .	23
5.5 Hiërarchische randdetectie . . . . .	24
<b>6 Generatie</b>	<b>26</b>
6.1 Periodische tesselatie . . . . .	26
6.2 Seedpoints . . . . .	26
6.2.1 Target seedpoints . . . . .	27
6.2.2 Source seedpoints . . . . .	28
6.2.3 Mutaties . . . . .	28
6.2.4 Constraints . . . . .	29
6.2.5 Conclusie . . . . .	29
6.3 Recursieve gridgeneratie . . . . .	29
6.3.1 Keuzeheuristiek . . . . .	30
6.3.2 Splitsingsas-heuristiek . . . . .	32
6.3.3 Fractie-heuristiek . . . . .	33
6.3.4 Adaptieve splitsing . . . . .	34
6.3.5 Stop criterium . . . . .	37
6.3.6 Postprocessing . . . . .	37
<b>7 Matching</b>	<b>38</b>
7.1 Methode . . . . .	38
7.1.1 Template matching . . . . .	38
7.1.2 Template matching . . . . .	38
7.1.3 Keuze van rotatie index . . . . .	39
7.1.4 Parallelisatie . . . . .	40

7.2	Fabriceerbare matching . . . . .	40
7.2.1	Prioriteit . . . . .	40
7.2.2	Mask . . . . .	41
<b>8</b>	<b>Resultaten</b>	<b>43</b>
8.1	Invloed van feature gewichten . . . . .	44
8.2	Invloed van histogram egalisatie . . . . .	44
8.3	Non adaptief vs adaptief . . . . .	45
8.4	Kleur vs intensiteit . . . . .	45
8.5	Invloed van adaptieve gridgeneratie via gedilateerde randdetectie . . . . .	46
8.6	Verschillende source materialen . . . . .	47
8.7	Invloed van aantal rotaties . . . . .	47
8.8	Beperkingen . . . . .	49
<b>9</b>	<b>Conclusie</b>	<b>51</b>

# Samenvatting

# Lijst van figuren

2.1	Parket	3
2.2	Marqueterie	3
2.3	Nieuwe beelding	4
2.4	Voorbeeld van template matching	6
2.5	Verschillende filters voor image processing	7
2.6	Een voorbeeld van histogram egalisatie	8
3.1	Een voorbeeld van neurale stijloverdracht door Gatys et al. (2016)	11
3.2	Een mondriaan verdeling in drie dimensies	12
4.1	Een overzicht van de verschillende stappen in de pipeline	14
5.1	Voorstelling van de gebruikte schalering	21
5.2	Template matching met enkel intensiteit als feature	22
5.3	Geroteerde $I_S$ , features en mask voor $n_r = 3$	23
5.4	$I_R$ , berekend met histogram egalisatie en een niet-monochrome $I_S$	24
5.5	Hiërarchische randdetectie met $n = 10$ , $F_i$ (bovenaan) en $R_i$ (onderaan)	25
6.1	Een matching via een zeer fijne reguliere gridverdeling	26
6.2	Een deadlock tijdens seedpoint grid generatie	28
6.3	Tableau no. 1 door Piet Mondriaan	29
6.4	De gebruikte kansverdeling voor het kiezen van patches en zijn geïnverteerde $cdf$	30
6.5	Een overzicht van verschillende keuze-karakteristieken	30
6.6	Een overzicht van verschillende functies binnen de feature karakteristiek	31
6.7	Een overzicht van verschillende splitsings-heuristieken	32
6.8	Een overzicht van verschillende fractie-heuristieken	33
6.9	De werking van adaptieve splitsing	34
6.10	De werking van gedilateerde hiërarchische randdetectie	36
6.11	Een voorbeeld van een slechte matching door adaptieve gridverdeling	36
6.12	Het verschil tussen hiërarchische randdetectie en zijn gedilateerde variant	37
7.1	Een overzicht van verschillende sorteer-heuristieken	40
7.2	Extra maskering na template matching om overlap te voorkomen	41
7.3	Een overzicht van de verschillende stappen om de source mask te transformeren	42
8.1	Het standaard source materiaal, afkomstig uit (Iseringhausen et al., 2020)	43
8.2	De invloed van $w_{egalisatie}$ en $w_{intensiteit}$ op de resulterende afbeelding	44

8.3	Een rendering van twee aangezichten en een sneeuwhond met $w_{intensiteit} = 0.75$ en $w_{rand} = 0.25$ . De equalisatie factor $w_{equalisatie}$ varieert tussen 0% en 100%. . . . .	45
8.4	Het verschil tussen het gebruiken van kleur en intensiteit als feature. . . . .	46
8.5	Een afbeelding van Beethoven, genomen uit (Iseringhausen et al., 2020). . . . .	46
8.6	Een vergelijking van hiërarchische randdetectie en gedilateerde randdetectie. . . . .	47
8.7	Resultaten met verschillende source materialen. . . . .	48
8.8	Een aantal resultaten met een verschillend aantal rotaties, $I_T$ is weergegeven in figuur 6.11a en is afkomstig van (Iseringhausen et al., 2020). . . . .	49
8.9	Een matching waar zwarte patches foutief gematcht zijn. . . . .	50

# Lijst van tabellen

8.1 De standaard instellingen voor de parameters. . . . .	43
---	----

# Lijst van afkortingen en symbolen

## Afkortingen

CDF Cumulatieve distributiefunctie

## Symbolen

$\phi$	De gulden snede
$I_S$	De source- of bron-afbeelding
$I_T$	De target- of doel-afbeelding
$I_R$	De resulterende afbeelding, verkregen door stukken uit $I_T$ te vervangen met stukken uit $I_S$
$P_T$	Een target patch, een stuk genomen uit $I_T$
$P_S$	Een source patch, een stuk genomen uit $I_S$ , gevonden door $I_S$ te vergelijken met $P_T$

# Hoofdstuk 1

## Inleiding

### 1.1 Achtergrond en motivatie

Mozaïek is een kunstvorm waarbij een “target” afbeelding wordt afgebeeld door ze onder te verdelen in kleine tegels die van kleur, textuur of vorm verschillen. Vroeger verdeelde men de afbeelding vaak in reguliere tegels, zoals bij een regulier grid of een hexagonaal grid. Deze thesis onderzoekt de mogelijkheid om de target afbeelding onder te verdelen in een grid dat zich aanpast aan de vorm van deze afbeelding en de stijl van de schilder Piet Mondriaan benaderd. De resulterende mozaïek wordt gemaakt door de inhoud van de tegels –ook wel patches genoemd– te vergelijken met een “source” afbeelding. Elke patch uit de target afbeelding wordt dan vervangen door een patch uit de source afbeelding waarvan de inhoud gelijkaardig is. Door gebruik te maken van verschillende features van de target en de source afbeelding, is het mogelijk om de textuur die in de source afbeelding aanwezig is, te gebruiken om details van de target afbeelding weer te geven. Dit wordt matching genoemd. Om de matching betere resultaten te laten leveren, wordt adaptieve gridgeneratie toegepast. Het doel van adaptieve gridgeneratie is dat de het grid in gedetailleerde regio’s van de target afbeelding meer onderverdeeld is dan in uniforme regio’s. Op deze manier kan de invulling van het grid met source patches een beter resultaat geven.

Het creëren van een grid in de stijl van Mondriaan en het garanderen van een accuraat resultaat zijn twee doelen die niet hand in hand gaan. Een Mondriaan grid vereist dat alle tegels rechthoekig zijn, dat het grid niet te fijn verdeeld is en dat er een verschil is tussen de afmetingen van verschillende tegels. Daarentegen zorgen meer en kleinere tegels voor een accuratere mozaïek, omdat kleinere tegels beter gematcht kunnen worden met de source afbeelding. Het is belangrijk om een balans te vinden tussen deze twee doelen.

### 1.2 Onderzoeksvragen

In deze thesis worden twee onderzoeksvragen behandeld.

- Hoe kan een vlak computationeel onderverdeeld worden in een grid dat de stijl van Piet Mondriaan benadert?
- Welke heuristieken kunnen gebruikt worden om het resulterende grid aan te passen zodat het betere resultaten produceert?

Het doel is om een balans te vinden tussen de kwaliteit van het mozaïek en de kwaliteit van het grid.

## 1.3 Contributies

Deze thesis maakt de volgende contributies:

- **Een algoritme dat grids kan genereren die de stijl van Piet Mondriaan benaderen.**

Aan de hand van een aantal parameters en heuristieken kan het algoritme verschillende soorten grids genereren. Dankzij een iteratief algoritme kan de generatie tijdens elke iteratie stopgezet worden of kunnen parameters naar wens aangepast worden.

- **Een hiërarchische randdetectie filter  $F_{HR}$  die de randen van een afbeelding sorteert volgens hun belang.**

Een randdetectie filter dat de scherpe randen van een Canny filter bevat, maar wel rekening houdt met de belangrijkheid van de randen. Meer algemene contouren zijn helderder dan randen die veel detail bevatten.

- **Een adaptieve gridgenerator dat een Mondriaan grid kan genereren en zich aanpast aan de vorm van de target afbeelding.**

Aan de hand van hiërarchische randdetectie gaat het grid adaptief gegenereerd worden zodat het de vorm van de target afbeelding benadert. Op deze manier helpen we het algoritme om betere resultaten te creëren in de matching fase.

## 1.4 Structuur

Het volgende hoofdstuk beschrijft concepten dat gebruikt worden doorheen de thesis, zoals convolutie filters, randdetectie, template matching enzovoort. Deze concepten zijn geen contributies van deze thesis. Vervolgens bespreekt hoofdstuk 3 de relevante literatuur voor deze thesis. Hoofdstuk 4 geeft een overzicht van de volledige pipeline, zonder dieper in te gaan op specifieke details of implementatie. De volgende hoofdstukken doen dit wel en bespreken elke stap van de pipeline in detail. Hoofdstuk 5 bespreekt de preprocessing, hoofdstuk 6 gaat over gridgeneratie en 7 bespreekt de matching. Ten slotte sluit de thesis af met resultaten in hoofdstuk 8.

# Hoofdstuk 2

## Achtergrond

### 2.1 Betegeling

Betegeling, tessellatie of tiling is het volledig opvullen van een vlak met tegels of tiles, waarbij de tegels niet mogen overlappen met elkaar. De vorm en kleur van de tegels zijn dus vrije variabelen. Dit concept kan ook gebruikt worden in hogere dimensies, maar in deze thesis worden enkel tweedimensionale betegelingen beschouwd (Wikipedia, 2022a).

#### 2.1.1 Mozaïek en parket

De meest voorkomende betegelingen zijn periodisch van vorm, zoals een regulier grid. Dit soort betegeling wordt vaak gebruikt in mozaïek, waar men de kleur van de tegel varieert om een bepaald patroon of een bepaalde afbeelding weer te geven. Een meer specifieke vorm van mozaïek is parket, waarbij houten planken worden gebruikt als tegel en vaak in een geometrisch patroon gelegd worden. De kleur van het hout wordt dan gebruikt om patronen of figuren te vormen, terwijl de structuur en het detail van het hout eerder als decoratief effect wordt gebruikt (Iseringhausen et al., 2020) (zie figuur 2.1).

#### 2.1.2 Marqueterie

Een alternatieve manier om figuren weer te geven met tegels is marqueterie. Marqueterie is gelijkaardig aan parket in de zin dat het voornamelijk houten tegels gebruikt om het vlak op te vullen. In tegenstelling tot parket zijn de tegels vaak verschillend van vorm en worden ze verkregen door homogene regio's in het patroon of de afbeelding toe te kennen aan één tegel. De betegeling in marqueterie is meestal niet geometrisch van vorm en de tegels zijn vaak aangepast aan de vorm van het te betegelen patroon (zie figuur 2.2).



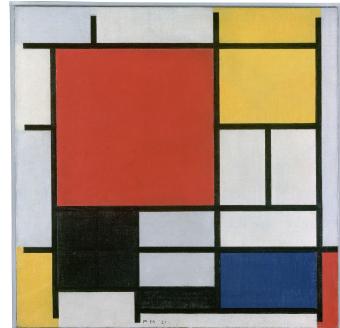
Figuur 2.1: Parket



Figuur 2.2: Marqueterie

## 2.2 Mondriaan

Piet Cornelis Mondriaan (Wikipedia, 2020) was een Nederlandse kunstschilder die leefde van 1872 tot 1944. In zijn vroege jaren waren zijn schilderijen eerder impressionistisch (Wikipedia, 2022c), zoals hij dat tijdens zijn kunstopleiding geleerd had. Tijdens een korte periode in Frankrijk volgde hij een expressionistisch kubistische stroming, zoals andere grote kunstschilders van die tijd (e.g. Picasso, Braque). Pas tijdens het begin van de Eerste Wereldoorlog begon hij in Nederland met zijn abstracte werken waarvoor hij zo bekend is. Samen met Theo van Doesburg ontwikkelde hij een stijl die ze “nieuwe beelding” of “neoplasticisme” noemden (zie figuur 2.3). Dit is de stijl dat deze thesis probeert te benaderen tijdens het genereren van een grid.



Figuur 2.3: Nieuwe beelding

## 2.3 Convolutie filters

Een convolutiefilter of kernel  $K$  is een matrix dat geconvoluteerd wordt over een afbeelding  $I$  en zo effecten kan toepassen zoals blurren en verscherpen. Kernels worden ook gebruikt om kenmerken van afbeeldingen te berekenen zoals afgeleiden en randen. Het toepassen van een convolutiefilter kan geschreven worden als  $O = K * I$  en de intensiteit van elke pixel in resultaat  $O$  is een gewogen gemiddelde van de omliggende pixels in  $I$ . Het exacte effect van de kernel wordt bepaald door de gewichten van de matrix.

### 2.3.1 Box filter

Een box filter is een simpele convolutiefilter waarvan de kernel de onderstaande vorm heeft:

$$K = \alpha \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix} \quad (2.1)$$

waarbij  $\alpha$  een eventuele normaliserende factor is. Box filters hebben een blur-effect op afbeeldingen en worden vaak gebruikt om een Gaussische blur te benaderen (Wikipedia, 2022b).

### 2.3.2 Gaussische blur

Een veelgebruikte convolutiefilter is een Gaussische blur filter, waarbij de gewichten van de kernel gekozen zijn volgens een tweedimensionale Gauss curve (zie formule 2.2). In theorie bereikt deze functie nooit de waarde 0, waardoor elke pixel in  $I$  invloed zou hebben op  $O$ . In praktijk is de grootte van de kernel vaak gelimiteerd tot  $3 \times 3$ ,  $5 \times 5$  of  $7 \times 7$  en limiteert dit dus ook de grootte van het receptief veld.

$$G_{gauss}(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.2)$$

### 2.3.3 Rolling guidance

Een populaire toepassing in beeldverwerking is het verwijderen van ruis en kleine details. Hiervoor kan een Gaussische blur operatie gebruikt worden, maar het nadeel van deze oplossing is dat deze filter geen onderscheid maakt tussen “sterke” en “zwakke” randen. Sterke randen worden gekenmerkt aan grote verandering in intensiteit en bevatten vaak belangrijke informatie. Zwakke randen bevatten daarentegen slechts een klein contrastverschil. Een optimaal algoritme behoudt sterke randen en verwijdert zwakke randen en graduele kleurovergangen. Dit is het doel van rolling guidance door Zhang et al. (2014). Het algoritme start met een Gaussische filter met een standaarddeviatie  $\sigma_s$ . Dit garandeert dat alle details die kleiner zijn dan  $\sqrt{\sigma_s}$ , verwijderd zijn in de resulterende afbeelding, zoals vermeld door Lindeberg (1994). Vervolgens worden de sterke randen die verwijderd waren tijdens de Gaussische filter, hersteld. Dit gebeurt iteratief via een joint bilaterale filtering, zoals beschreven in formule 2.3.

$$J^{t+1}(p) = \frac{1}{K_p} \sum_{q \in N(p)} \exp \left( -\frac{\|p - q\|^2}{2\sigma_s^2} - \frac{\|J^t(p) - J^t(q)\|^2}{2\sigma_r^2} \right) I(q) \quad (2.3)$$

met

$$K_p = \sum_{q \in N(p)} \exp \left( -\frac{\|p - q\|^2}{2\sigma_s^2} - \frac{\|J^t(p) - J^t(q)\|^2}{2\sigma_r^2} \right)$$

In deze formule zijn er drie parameters die gekozen moeten worden. Het aantal iteraties,  $\sigma_s$  en  $\sigma_r$ .  $\sigma_s$  is de standaarddeviatie van de Gaussische filter, en dient als gewicht voor het spatiale deel van de bilaterale filter. Deze parameter stelt de schaal in waarmee details verwijderd worden. Een hoger aantal iteraties verscherpt de sterke randen in de afbeelding. Een voorbeeld van rolling guidance is gegeven in figuur 2.5g.

### 2.3.4 Template matching

Template matching laat toe om de overeenkomst te berekenen tussen een afbeelding  $I$  met dimensie  $(W \times H)$  en kleinere template afbeelding  $T$  met dimensie  $(w \times h)$ . Elke pixel  $(p_x, p_y)$  van resultaat  $O$   $(W - w + 1 \times H - h + 1)$  bevat een waarde die weergeeft hoe goed  $T$  overeenkomt met de oppervlakte  $(p_x \rightarrow p_x + w, p_y \rightarrow p_y + h)$  in  $I$ , aan de hand van een afstandsmetriek  $\delta(p_I, p_T)$ . Deze metriek kan vrij gekozen worden, maar in deze thesis gebruiken we enkel het gekwadrateerde verschil

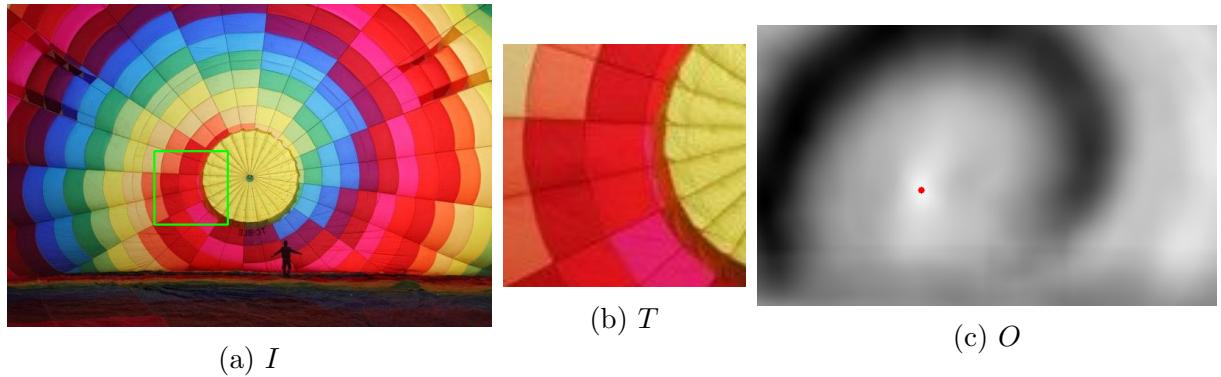
$$O_{sqdiff}(x, y) = \sum_{\substack{x'=0..w-1 \\ y'=0..h-1}} (T(x', y') - I(x + x', y + y'))^2 \quad (2.4)$$

of de correlatie

$$O_{corr}(x, y) = \sum_{\substack{x'=0..w-1 \\ y'=0..h-1}} T(x', y') * I(x + x', y + y') \quad (2.5)$$

als afstandsmetriek, of hun genormaliseerde versie zoals beschreven in OpenCV (2022b). Template matching kan gezien worden als een convolutie van  $T$  over  $I$  met een aangepaste operator om pixelwaarden te combineren.

Deze methode kan op verschillende manieren gebruikt worden:



Figuur 2.4: Voorbeeld van template matching

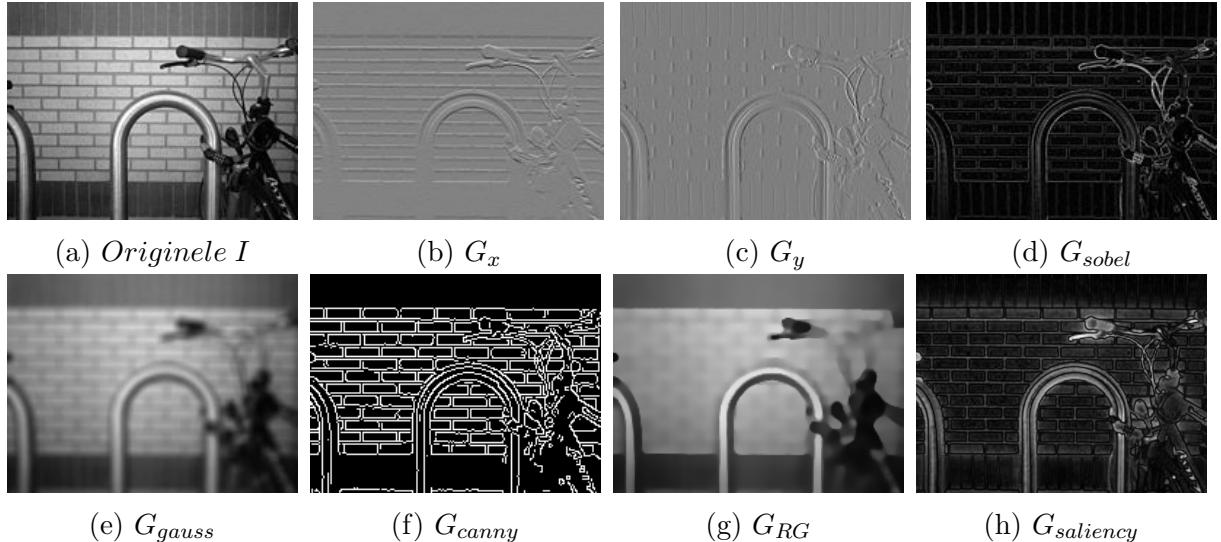
- Door een stuk uit  $I$  te knippen en dat te gebruiken als template, kan template matching detecteren waar  $T$  zich in  $I$  bevindt. Hiervoor wordt best het gekwadraatde verschil gebruikt als afstandsfunctie. De eerste waarde in  $O$  die dan 0 is, is de locatie waar  $T$  uit  $I$  geknipt is.
- Indien  $T$  geen exacte match heeft in  $I$ , kan de beste overeenkomst gevonden worden door de minimale of maximale waarde in  $O$  te zoeken, afhankelijk van de gekozen afstandsfunctie. Dit wordt weergegeven in figuur 2.4.
- Door een aangepaste filter te kiezen als  $T$ , kan er gezocht worden naar plaatsen die een sterke responsie geven op deze filter. Indien de filter bijvoorbeeld een witte rechthoek is van  $(1 \times x)$  pixels, zal de maximale (of minimale) waarde de plaats aanduiden die het best overeenkomt met deze rechthoek. Indien  $I$  dan een afbeelding is waar randdetectie op is uitgevoerd, kan template matching op deze manier gebruikt worden om horizontale randen van lengte  $x$  te lokaliseren.

## 2.4 Randdetectie

Edge filters of randdetectie wijst op een aantal methodes om randen te vinden in een afbeelding  $I$ , waar er snelle veranderingen van intensiteit of discontinuïteiten aanwezig zijn. Voor veranderingen in intensiteit te meten gebruikt men voornamelijk eerste en tweede orde afgeleiden via convolutiefilters. Deze methodes houden geen rekening met de werkelijke inhoud van de afbeelding, dus valse randen, niet-geconnecteerde randen en niet gedetecteerde randen zijn veelvoorkomende problemen.

### 2.4.1 Sobel filter

De meest bekende edge filter is de Sobel-Feldman operator (Sobel, 2014). Deze filter maakt gebruik van twee kernels, die respectievelijk de horizontale en de verticale afgeleide van de afbeelding berekenen (zie formule 2.6). Deze kernels kunnen geschreven worden als een product van een binomiaal-filter en een differentiatie-filter (e.g.  $G_x = [1 \ 2 \ 1]^T \cdot [-1 \ 0 \ 1]$ ), wat overeenkomt met een blur-operatie vooraleer de afgeleide wordt berekend. Dit reduceert de ruis in de resulterende afbeelding. De twee kernels  $G_x$



Figuur 2.5: Verschillende filters voor image processing

en  $G_y$  worden gecombineerd om de magnitude  $G$  te bepalen (zie formule 2.7), welke uiteindelijk de randen van  $I$  bevat. Ook de gradiënt van de intensiteit kan berekend worden via formule 2.8. Een nadeel van de Sobel-filter is dat de resulterende randen verschillende intensiteiten kunnen hebben, gevoelig zijn aan ruis, en vaak meerdere pixels breed zijn. Dit zorgt ervoor dat een Sobel-filter niet optimaal is om exacte locaties van randen te berekenen, maar wel voldoende is voor template matching.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \quad (2.6)$$

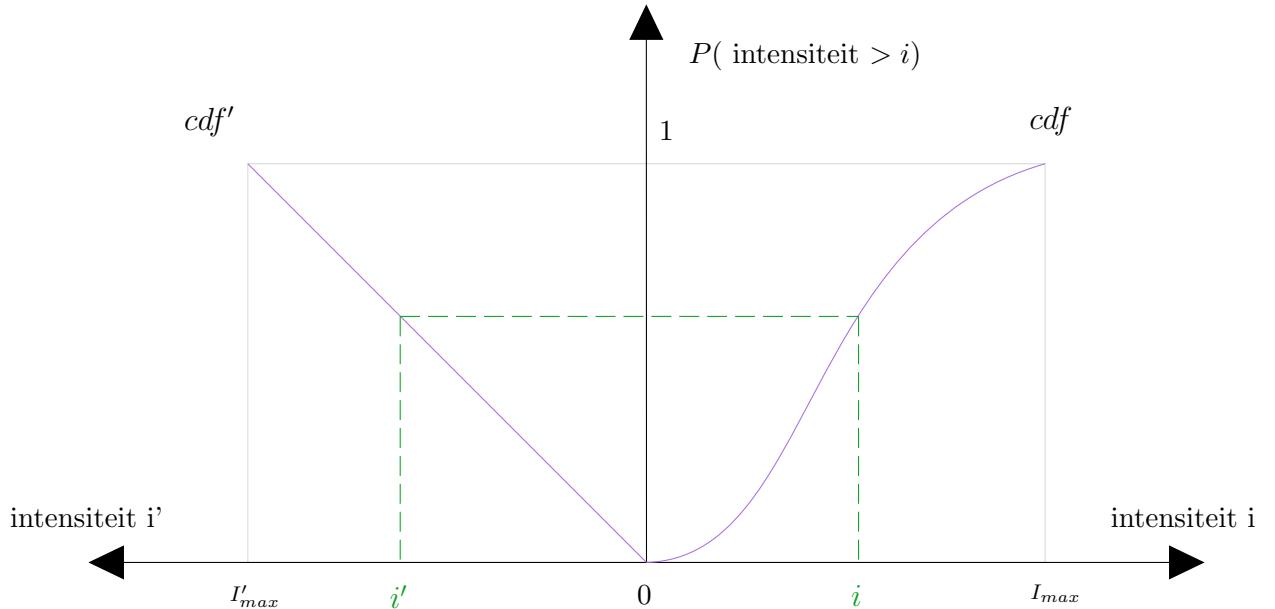
$$G_{sobel} = \sqrt{G_x^2 + G_y^2} \quad (2.7)$$

$$\Theta_{sobel} = \text{atan}2(G_y, G_x) \quad (2.8)$$

### 2.4.2 Canny filter

De Canny filter (Canny, 1986) is een alternatieve manier om randen te detecteren in afbeeldingen. Het produceert unieke randen die accuraat gelokaliseerd zijn met een lage fout. De filter maakt gebruik van vijf stappen:

1. De afbeelding wordt geconverteerd naar grijswaarden en een Gaussische blur wordt toegepast om ruis te verminderen.
2. De gradiënt van de intensiteit wordt berekend. Dit gebeurt op dezelfde manier als beschreven in formule 2.8. In tegenstelling tot de Sobel-filter gebruikt een Canny-filter ook diagonale kernels om diagonale randen beter te detecteren.
3. Lower bound cut-off suppressie wordt toegepast om randen dunner te maken.



Figuur 2.6: Een voorbeeld van histogram egalisatie

4. Double thresholding wordt gebruikt om sterke en zwakken randen te detecteren en te zwakke randen te markeren als ruis.
5. Edge tracking via hysteresis kijkt naar de zwakke randen en onderscheidt echte van valse randen door te kijken of er sterke randen in de nabije omgeving liggen. Indien dit niet zo is, wordt de zwakke rand als ruis beschouwd.

## 2.5 Histogram egalisatie

Histogram egalisatie is een methode om het contrast van een afbeelding te vergroten door intensiteitswaarden beter te verspreiden over het histogram van de afbeelding. Het histogram van een afbeelding bevat een weergave van hoe vaak elke intensiteit voorkomt. Een afbeelding met laag contrast gebruikt slecht een klein deel van het intensiteitspectrum, en dat is weergegeven in een histogram als een smalle strook aan intensiteiten. Het doel van histogram egalisatie is om deze strook uit te spreiden over het hele spectrum, om zoveel mogelijke intensiteiten te gebruiken, maar hun relatieve sortering te bewaren. In het beste geval is elke intensiteit evenveel aanwezig in de afbeelding, en hebben we een vlak histogram.

Om dit te bereiken maken we gebruik van de cumulatieve distributiefunctie  $cdf : \mathcal{R} \rightarrow \mathcal{R}$ . Deze functie toont het percentage  $x$  aan pixels die ten minste een intensiteit van  $cdf(x)$  hebben en kan berekend worden door het histogram te integreren vanaf de linkerkant. Om een vlak histogram te verkrijgen, moet de  $cdf$  een rechte zijn. Dit is zichtbaar in figuur 2.6. Intensiteit  $i$  wordt getransformeerd naar  $i'$  via  $cdf$  en  $cdf'$ , welke respectievelijk de  $cdf$  van de afbeelding en een rechte zijn.

Tot nu toe is er gesproken van histogram egalisatie die de intensiteitswaarden van een

afbeelding  $I$  transformeert om deze zo goed mogelijk te verspreiden. Anderzijds is het ook mogelijk om een aangepaste functie te kiezen voor  $cdf'$ . Indien  $cdf'$  gekozen wordt als de cdf van een andere afbeelding  $J$ , kunnen de intensiteitswaarden van  $I$  zo getransformeerd worden dat  $cdf_I = cdf_J$ . In dat geval gebruiken  $I$  en  $J$  hetzelfde bereik aan intensiteiten.

## 2.6 Saliency detectie

Saliency detectie is een methode om interessante en opvallende regio's of objecten te detecteren in een afbeelding. De originele methode was bedacht door Montabone and Soto (2010) om mensen te detecteren in statische beelden, maar is gegeneraliseerd om meerdere vormen van saliency te herkennen. Figuur 2.5h geeft een voorbeeld van een saliency map.

# Hoofdstuk 3

## Literatuur

Dit hoofdstuk geeft een overzicht van literatuur dat relevant is voor deze thesis.

### 3.1 Woodpixel

De paper van Iseringhausen et al. (2020) (WoodPixel) is de meest relevante literatuur voor deze thesis aangezien deze thesis voortbouwt op de ideeën die in de paper worden aangehaald. De paper beschrijft een manier om aan stijloverdracht te doen waarbij het source materiaal uitsluitend uit verschillende soorten hout bestaat. De focus van de paper ligt ook meer op het fabriceren van de uiteindelijke puzzel. Deze thesis beschrijft de nodige aanpassingen en methodes om een fabriceerbare puzzel te maken, maar focust meer op de generatie van het grid.

Net zoals deze thesis maakt Iseringhausen et al. (2020) gebruik van voornamelijk twee features om aan template matching te doen, een intensiteitsfeature en een randfeature. Aangezien het source materiaal uitsluitend monochroom is, worden de grijswaarden als feature gebruikt en wordt de target afbeelding geëgaliseerd zodat beide afbeeldingen hetzelfde bereik aan grijswaarden gebruikt. De tweede feature is een Sobel filtering van de grijswaarden. Deze feature zorgt ervoor dat de structuur van het hout gebruikt wordt om details in de target afbeelding weer te geven. Deze thesis gebruikt gelijkaardige features, maar beperkt zich niet tot hout als source materiaal. Dit is mede omdat de fabriceerbaarheid gewenst is, maar niet essentieel.

De generatie van het grid in WoodPixel verschilt fundamenteel van de methode in deze thesis. In de paper starten ze vanaf een regulier grid en worden de hoekpunten en randen van elke cel gemuteerd om een betere aansluiting te vinden met de target afbeelding. In deze thesis heeft het grid een Mondriaan stijl, wat een beperking legt op de vorm, de verdeling en de uitlijning van de cellen.

### 3.2 Stijloverdracht

Het gebruik van verschillende materialen om afbeeldingen na te maken bestaat al zeer lang. Een bekend voorbeeld hiervan is mozaïek en dateert terug tot de Romeinen en de Grieken. Sindsdien zijn er veel alternatieve vormen ontstaan zoals parket, marqueterie en

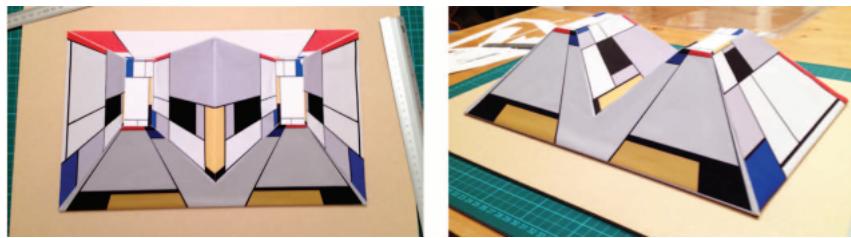


Figuur 3.1: Een voorbeeld van neurale stijloverdracht door Gatys et al. (2016).

worden materialen gebruikt zoals ivoor, keramiek, hout, parelmoer en bot.

Tegenwoordig bestaat er een groot bereik aan manieren om computationeel aan stijloverdracht te doen. Kyprianidis et al. (2013) beschrijft de taxonomie van de huidige methodes. Deze methodes maken gebruik van verschillende primitieven zoals lijnen (Hertzmann, 2003), stippen (Kim et al., 2009; Martín et al., 2010), patches (Iseringhausen et al., 2020; Orchard and Kaplan, 2008), en nog veel meer. Sommige methodes zijn example-based, en proberen de stijl van een source afbeelding te detecteren en over te brengen naar een target afbeelding. Dit is verschillend dan in deze thesis, waar een bepaalde stijl wordt verkregen door de keuze van het grid. De source afbeelding bepaalt enkel de kleur en het materiaal van de target afbeelding.

Deze laatste soort stijloverdracht is zeer populair en wordt tegenwoordig gerealiseerd via neurale netwerken (Gatys et al., 2016). Een convolutief neuraal netwerk wordt getraind om de afstand tussen een target afbeelding en zijn gestijlde output te minimaliseren. Daarnaast probeert het netwerk ook de afstand tussen de stijl van de output en de gewenste stijl te minimaliseren. Een voorbeeld van neurale stijloverdracht is zichtbaar in figuur 3.1.



Figuur 3.2: Een mondriaan verdeling in drie dimensies.

### 3.3 Mondriaan

Over stijloverdracht in de stijl van Mondriaan zijn weinig papers geschreven. Onderzoek door Stevanov and Zanker (2017) neemt een artistieke benadering en probeert een Mondriaan stijl te bereiken in drie dimensies, zoals weergegeven in figuur 3.2. Mcmanus and Gesiak (2014) probeert de stijl eerder wiskundig te verklaren. Deze thesis benadert de stijl op een eerder simpele manier, door een voorkeur te hebben voor uitgelijnde patches van verschillende groottes. Hoofdstuk 6 beschrijft onze methode.

# Hoofdstuk 4

## Overzicht

Dit hoofdstuk geeft een overzicht en een korte beschrijving van de verschillende stappen van het algoritme. De stappen zijn elk verder uitgelegd in de volgende hoofdstukken.

### 4.1 Doel

Het doel van deze thesis is om een “target” afbeelding  $I_T$  na te maken, gebruik makend van stukken of patches uit een “source” afbeelding  $I_S$ . Hierbij wordt  $I_T$  onderverdeeld in een grid dat voldoet aan een aantal stijlvereisten. Deze thesis onderzoekt een aantal veelvoorkomende grids, maar het uiteindelijke doel is om het resulterende grid de stijl van Mondriaan na te laten bootsen. Na de generatie van het grid worden alle tegels van het grid gematcht met source patches uit  $I_S$ . Ten slotte worden de target patches  $P_T$  vervangen met de source patches  $P_S$  om de resulterende afbeelding  $I_R$  te creëren. Daarnaast wordt er ook onderzocht hoe het algoritme resultaten kan produceren die fabriceerbaar zijn met echte materialen. In dit geval kan men de resulterende afbeelding zien als een puzzel, gemaakt met puzzelstukken uit  $I_S$ .

### 4.2 Contributies

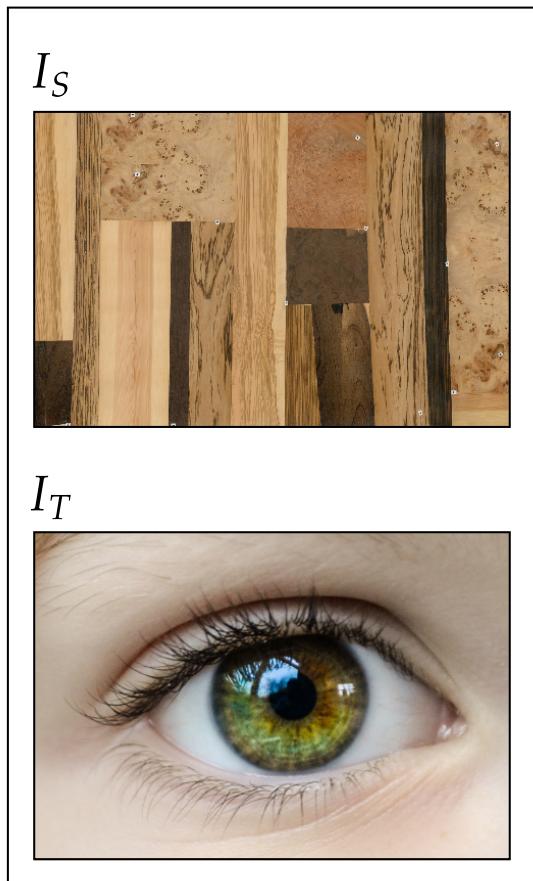
Deze thesis bouwt verder op het werk van Iseringhausen et al. (2020) en gebruikt een aantal ideeën van deze paper. De volgende ideeën zijn afgeleid:

- **De keuze van features**

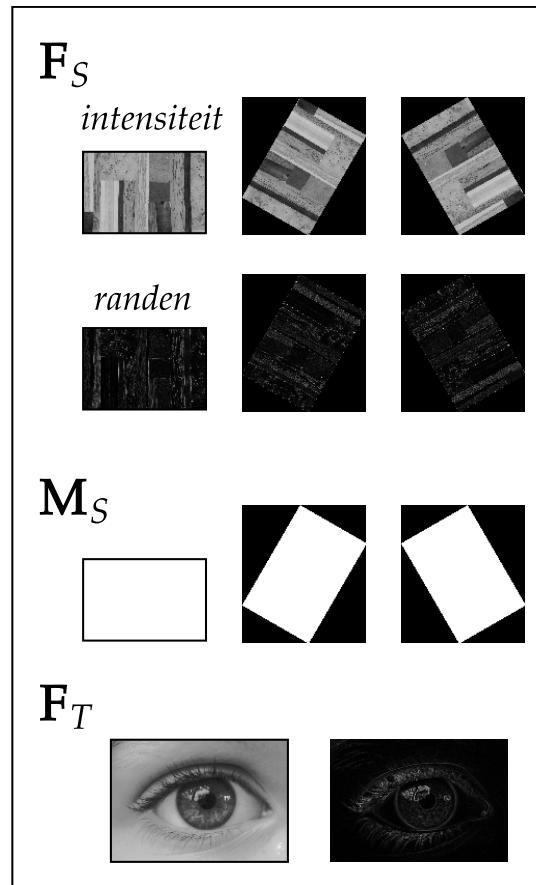
Deze thesis heeft een aantal features onderzocht om betere resultaten te krijgen tijdens het matchen. Uiteindelijk bleek dat de features uit (Iseringhausen et al., 2020) de beste resultaten gaven, al gebruikte deze thesis andere parameters.

- **Gebruik van saliency en rolling guidance**

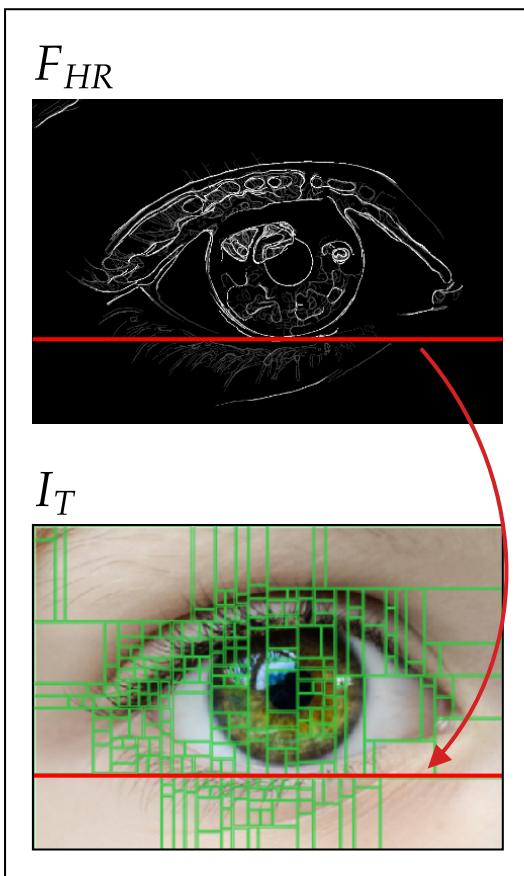
Saliency en rolling guidance worden door Iseringhausen et al. (2020) gebruikt voor het sorteren van patches en het muteren van het grid. Deze thesis experimenteerde eveneens met deze filters en gebruikte ze in verschillende delen van het algoritme.



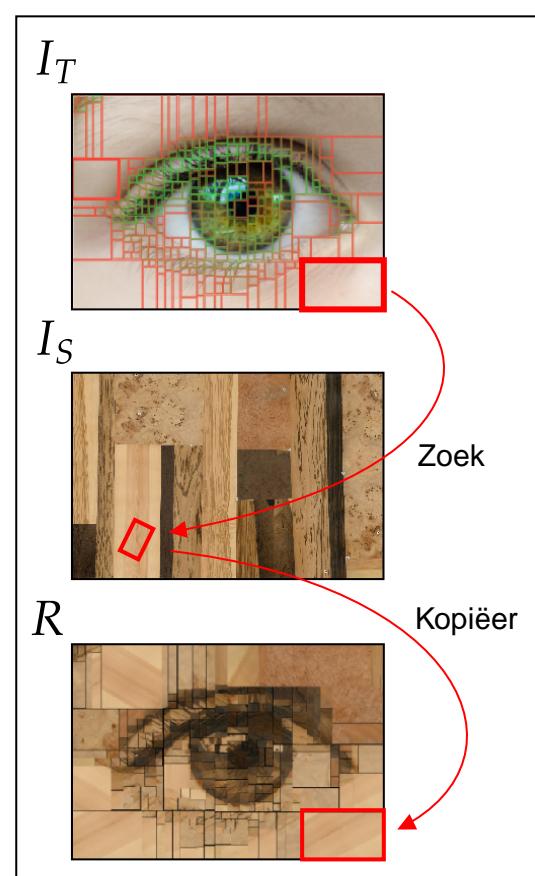
(a) Stap 1: Input



(b) Stap 2: Preprocessing



(c) Stap 3: Generatie



(d) Stap 4: Matching

Figuur 4.1: Een overzicht van de verschillende stappen in de pipeline.

De eigen contributies van deze thesis staan vermeld in de inleiding maar worden hier kort herhaald.

- **Een algoritme dat grids kan genereren die de stijl van Piet Mondriaan benaderen.**
- **Een hiërarchische randdetectie filter  $F_{HR}$  die de randen van een afbeelding sorteert volgens hun belang.**
- **Een adaptieve gridgenerator dat een Mondriaan grid kan genereren en zich aanpast aan de vorm van de target afbeelding.**

Daarnaast is de volledige implementatie van het algoritme origineel en niet afgeleid van de implementatie door Iseringhausen et al. (2020).

## 4.3 Input

De input stap bevat het kiezen van het source materiaal, de target afbeelding en een aantal parameters. Een voorbeeld van de input stap is weergegeven in figuur 4.1a. In deze figuur is het source materiaal  $I_S$  een reeks houten planken en de target afbeelding  $I_T$  een afbeelding van een oog.

### 4.3.1 Afbeeldingen

De eerste stap in het algoritme is het voorzien van de target afbeelding  $I_T$ . Deze afbeelding gaat nagemaakt worden van stukken materiaal uit de source afbeelding  $I_S$ . Afhankelijk van het soort materiaal dat gebruikt wordt voor  $I_S$ , gaan andere features gekozen worden tijdens de matching stap. Daarnaast is de grootte van de afbeeldingen ook van belang. Target afbeeldingen met een lage resolutie kunnen te weinig detail bevatten, terwijl te grote afbeeldingen geen verbetering bieden en het algoritme vertragen. Bij voorkeur is de source afbeelding van hoge kwaliteit, zodat veel details aanwezig zijn.

### 4.3.2 Parameters

Voor elke stap in het algoritme zijn er parameters die bepalen hoe het resulterende grid er uit zal zien. Deze parameters worden in detail uitgelegd in de volgende hoofdstukken maar de onderstaande parameters zijn het belangrijkst.

- **Het aantal geroteerde source afbeeldingen**

Zoals eerder vermeld gaan alle target patches vergeleken worden met patches uit de source afbeelding. Het kan mogelijk zijn dat een geroteerde versie van de source patch beter match met de target patch. Het aantal rotaties bepaalt hoeveel geroteerde duplicaten er worden gemaakt van  $I_S$  en zijn features  $F_S$ . Deze parameter verhoogt de kwaliteit van de matching maar vertraagt het algoritme.

- **De minimum afmetingen van een patch**

Om fabriceerbaarheid van patches te garanderen, moet een minimum afmeting ingesteld worden. Dit is de kleinste patch die uitgesneden kan worden en zorgt ook voor een stop-criterium tijdens de gridgeneratie.

- **Fabriceerbaarheid**

Deze instelling heeft een groot effect op het uiteindelijke resultaat. Indien de resulterende afbeelding fabriceerbaar moet zijn, mag geen elke source patch overlappen met een andere source patch. Hierdoor gaan de kwaliteit van de matching steeds minder optimaal zijn, naarmate er meer patches zijn gematcht.

## 4.4 Preprocessing

De preprocessing stap is de tweede stap van het algoritme en bevat alle handelingen die slechts één keer uitgevoerd moeten worden. Dit houdt het berekenen van features en masks in en het schalen en roteren van de features en de afbeeldingen. De preprocessing stap is gevisualiseerd in figuur 4.1b en wordt verder uitgelegd in hoofdstuk 5.

### 4.4.1 Schalen

Met de fabriceerbaarheid in gedachten, is het nodig dat de relatieve verhoudingen van  $I_S$  en  $I_T$  dezelfde blijven. Het zou bijvoorbeeld kunnen zijn dat het werkelijke source materiaal en de werkelijke puzzel een gelijkaardige grootte hebben, maar dat de resoluties van de target afbeelding en de source afbeelding zeer verschillend zijn. Hierdoor zou het kunnen dat twee gematchte patches die dezelfde grootte in pixels hebben, niet dezelfde grootte in millimeters hebben. Om dit tegen te gaan moeten  $I_S$  en  $I_T$  herschaald worden.

### 4.4.2 Features

Tijdens de matching fase wordt er gebruik gemaakt van features om de afstand te bepalen tussen een target patch en een source patch. De keuze van deze features komt overeen met de keuze van Iseringhausen et al. (2020). Deze features zijn ook zichtbaar in figuur 4.1b. Hierbij is  $\mathbf{F}_S$  een lijst van source features, één voor elke rotatie.  $\mathbf{F}_T$  is een lijst met de target features.

- **Intensiteit  $F_{intensiteit}$**

Deze feature bevat de intensiteit van de source of target afbeelding. Het kan zijn dat de target afbeelding minder contrast heeft dan de source afbeelding. In dat geval kan het zijn dat de matching stap geen gebruik maakt van de variëteit aan intensiteiten in het source materiaal. Om dat op te lossen wordt de intensiteit feature van de target afbeelding geëgaliseerd zodat de source en target intensiteit hetzelfde bereik aan intensiteiten hebben.

- **Randen  $F_{rand}$**

Deze feature bevat de randen van de source of target afbeelding en wordt gerealiseerd met een Sobel filter (Sobel, 2014). Deze feature helpt tijdens de matching stap om details uit het source materiaal te halen en deze te gebruiken om details van de target afbeelding weer te geven.

Daarnaast worden ook features zoals saliency (Montabone and Soto, 2010), rolling guidance (Zhang et al., 2014), Canny (Canny, 1986) en hiërarchische randdetectie berekend.

### 4.4.3 Masks

Ten slotte wordt er voor elke rotatie ook een source mask  $\mathbf{M}_S$  bijgehouden. Deze bakent de bruikbare oppervlakte in de geroteerde afbeeldingen af en speelt een rol in het garanderen van de fabriceerbaarheid.

## 4.5 Generatie

De derde en de belangrijkste stap van het algoritme is de generatie stap. Deze stap verdeelt de target afbeelding in tegels die later gematcht moeten worden tijdens de matching fase. Het doel van de generatie is om een grid te bekomen dat eventueel aangepast is aan de vorm van  $I_T$  en de stijl van Mondriaan benaderd. Deze stap is weergegeven in figuur 4.1d en wordt gedetailleerd uitgelegd in hoofdstuk 6.

### 4.5.1 Seedpoints

De eerste poging voor het creëren van een adaptief grid was aan de hand van seedpoints. De target afbeelding wordt afgezocht naar interessante plaatsen (seedpoints) en een patch met minimale afmetingen wordt op deze locatie geplaatst. Deze patches dienen als zaadjes waaruit het grid groeit. Vervolgens worden de patches iteratief gemuteerd door ze te verplaatsen, roteren en schaleren. Deze mutaties zijn random maar worden enkel geaccepteerd als de patches een betere match hebben dan voor de mutatie. Uiteindelijk bereiken de patches een toestand waar ze niet meer kunnen groeien omdat overlap niet toegelaten is. Indien er niet opgevulde plekken zijn in het grid, worden nieuwe seedpoints geplaatst die op hun beurt groeien. Deze methode was theoretisch veelbelovend maar bleek uiteindelijk onpraktisch. Het garanderen van een volledige opvulling van het vlak en het benaderen van een Mondriaan grid bleek onhandig via deze methode.

### 4.5.2 Recursieve gridgeneratie

De tweede poging begon vanaf een patch dat de hele target afbeelding innam. Vervolgens wordt deze patch in twee geknipt volgens de horizontale of de verticale as. Dit proces herhaalt zich tot een stop criterium is bereikt of de gebruiker tevreden is met het grid. Elke stap van dit algoritme kan aangepast worden via een aantal heuristieken.

- **Keuze-heuristiek**

De keuze-heuristiek kiest welke patches in de huidige iteratie gekozen worden om te splitsen. Dit gebeurt aan de hand van een bepaalde score die toegekend wordt aan elke patch. Deze score kan op verschillende manieren berekend worden en elke manier heeft een effect op de vorm van het grid. De keuze van patches op basis van deze score kan greedy verlopen, maar ook via een kansdichtheidsfunctie. Op deze manier kan het algoritme uit een lokaal optimum geraken.

- **Splits-heuristiek**

De splits-heuristiek bepaalt volgens welke as de patch gesplitst zal worden. Aangezien enkel rechthoekige patches toegelaten zijn, is er enkel keuze tussen de x-as en de y-as. Deze heuristiek kan gebruik maken van de eerder berekende score, of de keuze uitstellen tot een latere stap, waar de beste splitsingsas gekozen kan worden.

- **Fractie-heuristiek**

Nadat een patch en de splitsingsas gekozen zijn, moet er nog gekozen worden waar de splitsing precies geplaatst wordt. Dit gebeurt aan de hand van de fractie-heuristiek. Het aanpassen van deze heuristiek resulteert in een groot aantal verschillende grids. Door bijvoorbeeld een constante factor te nemen als fractie, is het mogelijk om reguliere grids te genereren. Indien deze dan gecombineerd wordt met een bepaalde keuze-heuristiek, zou het bijvoorbeeld mogelijk zijn om grids te genereren die de vorm van een quadtree hebben.

### 4.5.3 Adaptieve gridgeneratie

Adaptieve gridgeneratie is één van de contributies van deze thesis. Het is recursieve gridgeneratie waarbij de verschillende heuristieken specifiek gekozen zijn zodat het grid zich aanpast aan de vorm van de target afbeelding. Dit is weergegeven in figuur 4.1c. De hiërarchische randdetectie wordt gebruikt om randen te vinden die een sterk horizontaal of verticaal karakter vertonen. Dit gebeurt door een horizontale of verticale witte lijn te matchen met  $F_{HR}$ . De beste locatie wordt dan gebruikt om de fractie-heuristiek in te stellen en de beste splitsingsas wordt gebruikt om de split-heuristiek in te stellen. Aangezien de algemene contouren van  $I_T$  helderder zijn in  $F_{HR}$  dan de meer gedetailleerde randen, gaat het grid tijdens de eerste iteraties hier een voorkeur voor hebben en zich aanpassen aan de algemene vorm van de target afbeelding. Pas tijdens latere iteraties gaat het grid zich uitlijnen met de gedetailleerde randen van  $I_T$ . Een diepere uitleg over adaptieve gridgeneratie is gegeven in sectie 6.3.4.

## 4.6 Matching

De vierde en laatste stap van het algoritme is de matching. Deze stap neemt alle target patches uit  $I_T$  en zoekt in  $I_S$  naar een source patch dat de kleinste afstand heeft volgens de gekozen features. Nadat de source patch gevonden is, wordt deze gekopieerd naar de resulterende afbeelding  $R$ . Dit is visueel weergegeven in figuur 4.1d en wordt verder uitgelegd in hoofdstuk 7.

### 4.6.1 Rotaties

De matching fase is de stap waar de geroteerde afbeeldingen en features een rol spelen. De zoektocht naar de beste source patch voor een bepaalde target patch gebeurt parallel voor alle rotaties tegelijk. De afstand tussen twee patches wordt bepaald door de features van de target patch te template matchen met de features van  $I_S$ , en vervolgens de beste waarde te zoeken in een gewogen combinatie van de responsen. De rotatie en locatie waar de beste match gevonden werd, bepaalt de locatie en rotatie van de source patch.

### 4.6.2 Mask

Om de bruikbare oppervlakte in de geroteerde afbeeldingen aan te duiden, worden masks gebruikt. Deze dienen ook om de locaties te maskeren die al gereserveerd zijn door eerder gematchte patches. Hierdoor wordt voorkomen dat patches met elkaar overlappen. Het

uitvoeren van deze extra maskering is enkel nodig indien het resultaat fabriceerbaar moet zijn.

# Hoofdstuk 5

## Preprocessing

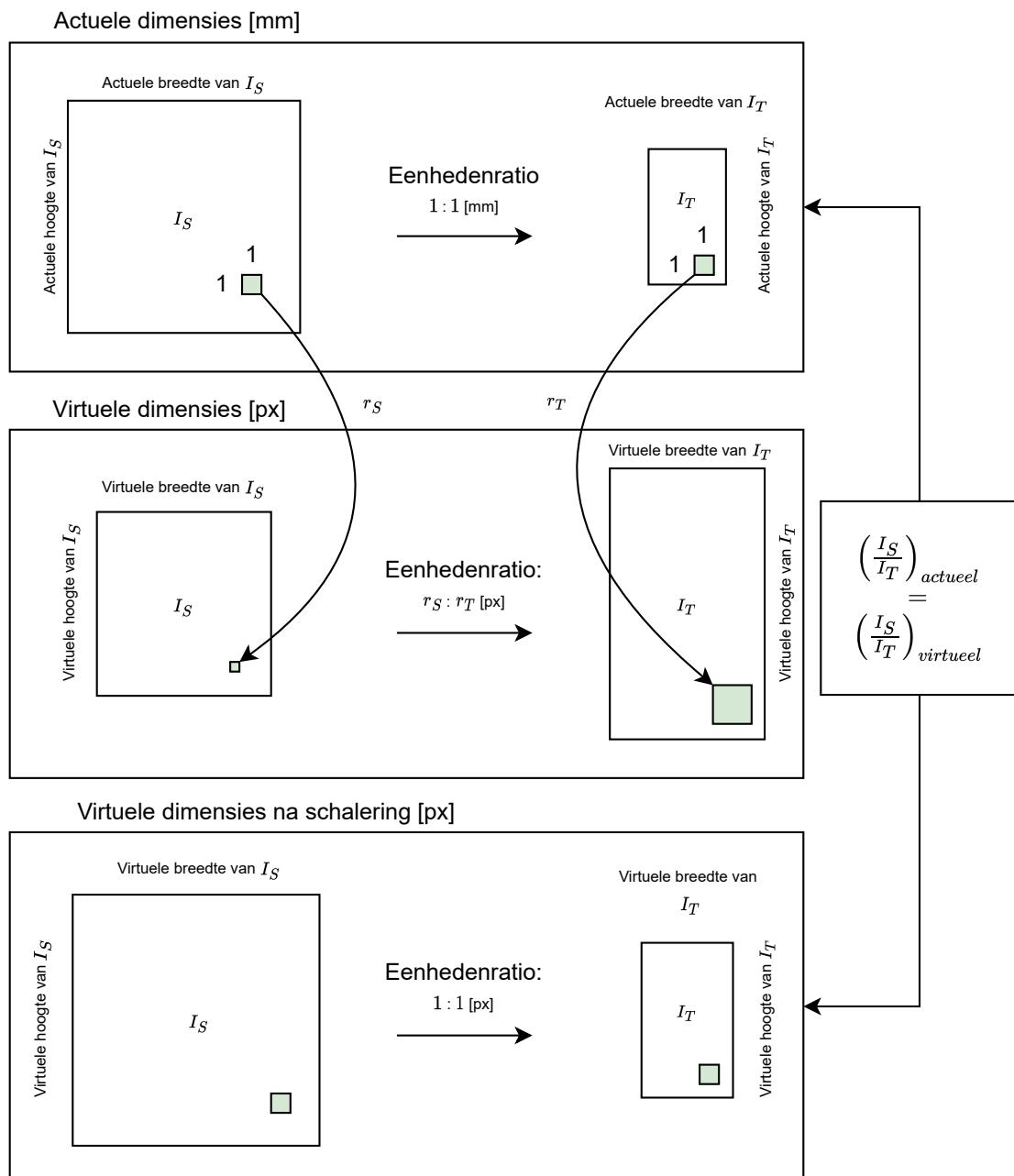
Dit hoofdstuk beschrijft het eerste deel van de pipeline, het preprocessen. Dit deel behandelt alle acties die slechts één keer moeten gebeuren voor een gegeven source en target afbeelding.

### 5.1 Schaling

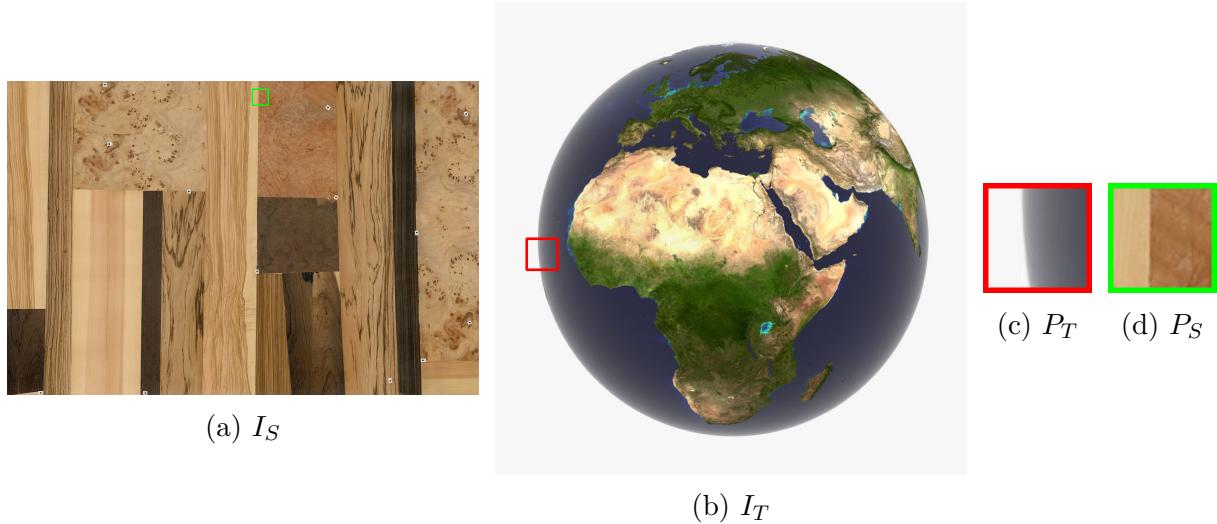
Eén van de doelen van deze thesis is om het resultaat van het algoritme ook fysisch realiseerbaar te maken. Dit legt een aantal beperkingen op de keuze van de schaal van de afbeeldingen. In de computationele pijplijn komen de afbeeldingen  $I_S$  en  $I_T$  binnen als afbeeldingen met virtuele dimensies, uitgedrukt in pixels. Daarentegen zijn de actuele dimensies van het fysisch materiaal en de fysische puzzel verschillend van de virtuele dimensies en zijn ze uitgedrukt in millimeters. Het is bijvoorbeeld mogelijk dat de actuele dimensie van  $I_S$  groter is dan  $I_T$ , maar de virtuele dimensie van  $I_S$  kleiner is dan  $I_T$ . Indien we dan twee stukken van  $(x \times x)$  pixels uit  $I_S$  en  $I_T$  met elkaar zouden vergelijken, zouden de actuele dimensies van die stukken verschillend zijn en is de vergelijking betekenisloos.

Om deze mismatch te voorkomen moeten  $I_S$  en  $I_T$  herschaald worden zodat de verhouding van het aantal pixels per millimeter in  $I_S$  (i.e.  $r_S$ ) gelijk is aan de verhouding van het aantal pixels per millimeter in  $I_T$  (i.e.  $r_T$ ). Hierdoor is het gegarandeerd dat twee stukken van dezelfde virtuele grootte, ook dezelfde actuele grootte zullen hebben. Het resultaat van deze herschaling is dat de verhouding van de afmetingen van  $I_S$  en  $I_T$  dezelfde zal zijn in de actuele dimensie en de virtuele dimensie. Deze uitleg is visueel weergegeven in figuur 5.1.

De schalingsfactor is dus een enkele variabele en is niet verschillend voor de breedte of de hoogte van  $I_T$  en  $I_S$ , dit komt door de veronderstelling dat de aspect ratio gelijk is in de actuele en de virtuele dimensie, zowel voor  $I_S$  als voor  $I_T$ . De enige keuze die nog overblijft is welke afbeelding herschaald moet worden. Een slechte keuze van afbeelding kan ervoor zorgen dat een van de afbeeldingen sterk omhoog of omlaag geschaald zal worden. Stel dat in de realiteit  $I_S$  en  $I_T$  even groot zijn, maar virtueel is  $I_S$  veel groter dan  $I_T$ .  $I_S$  is dus een afbeelding met een veel hogere resolutie. In dat geval zal  $I_T$  sterk omhoog geschaald moeten worden, wat niet optimaal is. De beste keuze zou zijn om  $I_S$  omlaag te schalen en  $I_T$  omhoog te schalen, zodat er niet teveel verlies is aan informatie in  $I_S$  en  $I_T$  niet te veel omhoog geschaald moet worden.



Figuur 5.1: Voorstelling van de gebruikte schalering



Figuur 5.2: Template matching met enkel intensiteit als feature

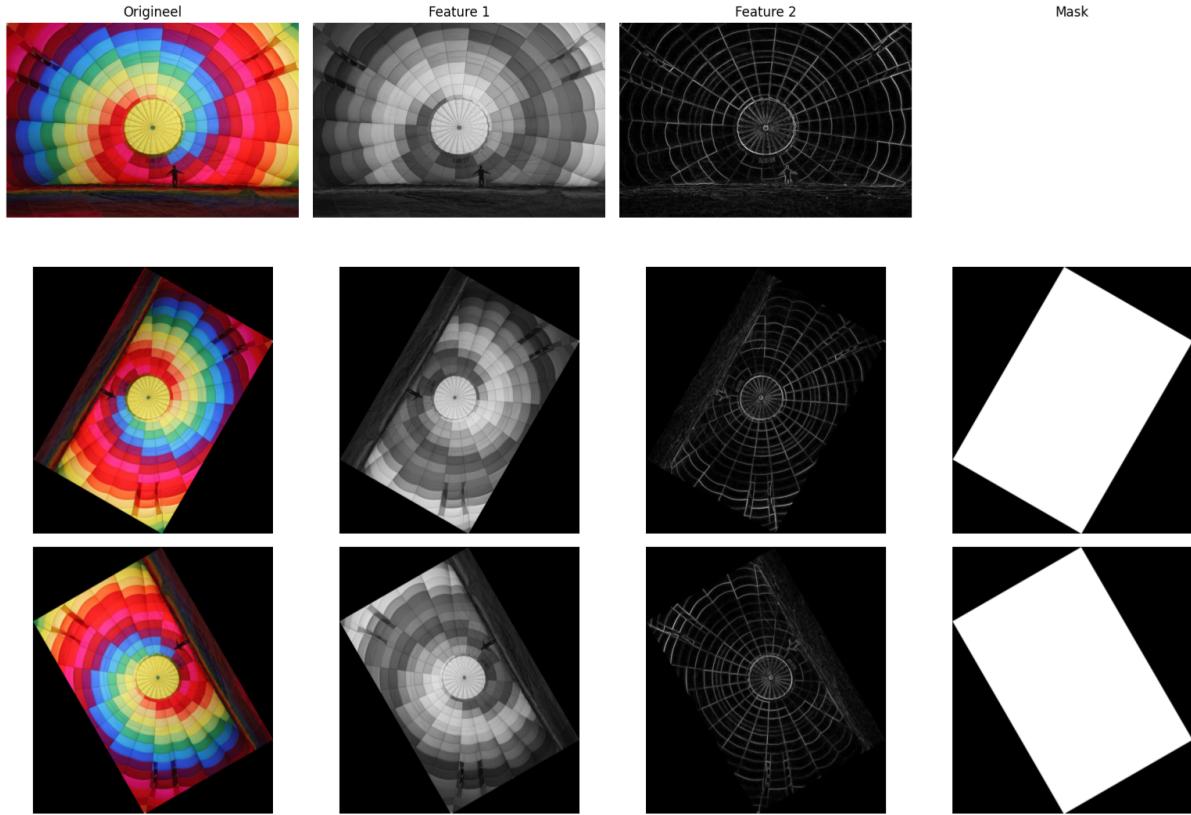
## 5.2 Features

Tijdens het matchen van patches met  $I_S$  wordt er gebruik gemaakt van gewogen features om de afstand te bepalen tussen de target patch en de source patch. De keuze van deze features heeft veel invloed op het resultaat en is ook afhankelijk van het bronmateriaal. Indien het bronmateriaal bijvoorbeeld een eentonige kleur heeft zoals hout, is het interessant om de grijswaarden of intensiteit  $F_{intensiteit}$  te gebruiken als feature. We zoeken namelijk naar stukken uit  $I_S$  die dezelfde intensiteit hebben als de target patch. Daarnaast willen we dat de textuur van  $I_S$  gebruikt wordt om randen en details van  $I_T$  weer te geven. Dit gebeurt via een edge feature  $F_{rand}$ . Deze feature is het resultaat van een Gaussische blur, gevolgd door een Sobel-filter. Deze features zijn te zien in figuur 5.3.

Merk op dat de  $F_{intensiteit}$  zelf al in staat is om textuur uit  $I_S$  te halen. Dit komt door de manier waarop de afstand berekend wordt tussen twee patches. Zoals vermeld in hoofdstuk 2, gaat er een afstandsfunctie uitgevoerd worden voor elke  $i^{de}$  pixel van  $P_S$  en  $P_T$ . Vanwege deze pixel-per-pixel vergelijking doen niet alleen de kleuren van de pixels er toe, maar ook de plaats van de pixels. Dit is weergegeven in figuur 5.2. Er wordt een match  $P_S$  gezocht voor patch  $P_T$  uit  $I_T$ , door te zoeken in  $I_S$ . De resulterende match bevat dezelfde verticale rand, ook al is enkel de intensiteit gebruikt als feature.

## 5.3 Rotaties

Tijdens het zoeken naar patches in de matching fase bekijken we niet enkel de beste locatie, maar ook de beste rotatie om een patch uit  $I_S$  te snijden. Aangezien het zoeken naar optimale patches computationeel zwaar is, staat er een limiet op het aantal mogelijke rotaties  $n_r$ . Er worden dan op voorhand  $n_r$  rotaties van  $I_S$  opgeslagen in een lijst. Hetzelfde gebeurt voor alle  $n_f$  features die gebruikt worden tijdens het matchen. Ten slotte wordt er voor elke rotatie een mask bijgehouden. Dit is nodig omdat er tijdens het roteren extra zwarte ruimte toegevoegd wordt aan de afbeelding, die geen informatie bevat. De mask geeft weer welk deel van de geroteerde afbeelding gesampled mag worden. In totaal



Figuur 5.3: Geroteerde  $I_S$ , features en mask voor  $n_r = 3$

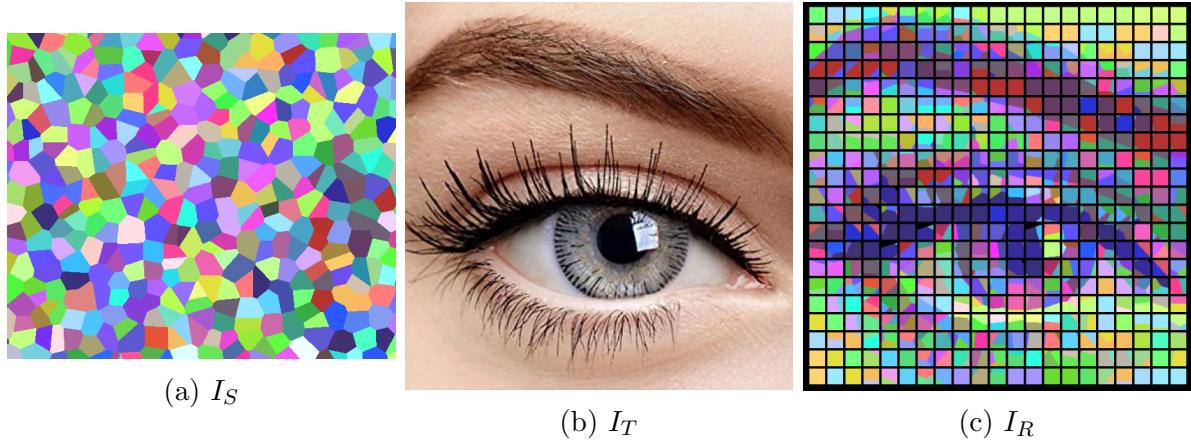
worden er dus  $(1_{(I_S)} + n_f + 1_{(mask)}) * n_r$  afbeeldingen opgeslagen. Dit is weergegeven in figuur 5.3. Tijdens het matchen houdt elke source patch een rotatie index  $i_{rot}$  bij die aangeeft welke rotatie die patch heeft. De hoek van de patch is dan gegeven door  $360^\circ * \frac{i_{rot}}{n_r}$ .

Merk op dat deze transformaties geen pure rotaties zijn, aangezien er dan stukken van de afbeelding weggeknippt zouden worden. De transformatie roteert de afbeelding rond zijn middelpunt, vergroot de afmetingen van de bounding box en transleert de geroteerde afbeelding vervolgens zodat de volledige afbeelding binnen de bounding box past. Deze operatie zal invers toegepast moeten worden indien men coördinaten van een geroteerde afbeelding wil omzetten naar coördinaten in de originele afbeelding.

## 5.4 Histogram egalisatie

Omdat  $I_S$  en  $I_T$  verschillende afbeeldingen zijn, genomen tijdens verschillende omstandigheden, is het mogelijk dat de distributie van grijswaarden verschillend is voor  $I_S$  en  $I_T$ . Een fel belichte  $I_T$  zou dan enkel gebruik maken van de lichtste kleuren hout, in plaats van het volledige spectrum aan kleuren te gebruiken dat in  $I_S$  aanwezig is. Om dit op te lossen wordt CDF-gebaseerde histogram egalisatie toegepast (zie hoofdstuk 2).

Meestal heeft een eentonig materiaal een klein bereik aan intensiteiten en is histogram egalisatie essentieel. In tegenstelling gaat de egalisatie in target afbeeldingen met een laag contrast minder goede resultaten opleveren. Dit wordt opgelost door te interpol-



Figuur 5.4:  $I_R$ , berekend met histogram egalisatie en een niet-monochroom  $I_S$

eren tussen de originele grijswaarden en de geëqualiseerde grijswaarden, gelijkaardig als beschreven door Iseringhausen et al. (2020) (zie forumle 5.1).

$$F'_{intensiteit} = (1 - w_{egalisatie}) * F_{intensiteit}(I_T) + w_{egalisatie} * F_{egalisatie}(I_S, I_T) \quad (5.1)$$

Merk op dat histogram egalisatie enkel nuttig is indien er met grijswaarden wordt gewerkt. In het geval van RGB waarden kan de egalisatie ervoor zorgen dat de originele kleuren in  $I_T$  veranderen, wat een ongewenst effect is. Op dezelfde manier is het niet nuttig om grijswaarden te gebruiken als feature wanneer  $I_S$  niet monochroom is. Twee dezelfde kleuren kunnen namelijk mappen op dezelfde grijswaarde, waardoor er ongewenste kleurverschillen optreden in  $I_T$ . Dit is te zien in figuur 5.4. Langs de andere kant kan dit geïnterpreteerd worden als een artistiek effect.

## 5.5 Hiërarchische randdetectie

Randdetectie algoritmes zoals Sobel of Canny werken goed om randen te detecteren maar geven geen indicatie over de belangrijkheid van de randen. Om dit op te lossen maakt deze thesis gebruik van een hiërarchische randdetectie, gebaseerd op de eerder vermelde rolling-guidance (zie hoofdstuk 2). De reden waarom de belangrijkheid van randen gekend moet zijn, wordt verder toegelicht in sectie 6.3.4. De methode start door de input-afbeelding  $I$ ,  $n$  keer te filteren met de rolling-guidance filter, waarbij de parameter  $\sigma_s$  steeds kleiner wordt. We beschikken dan over  $n$  features  $F_i$ , waarbij elke feature meer details bevat dan de vorige, aangezien  $\sigma_s$  de schaal regelt waarmee details verwijderd worden. Vervolgens passen we Canny randdetectie toe op elke  $F_i$  om de randen  $R_i$  te verkrijgen. De bedoeling is dat –aangezien  $F_1$  het minste details bevat–  $R_1$  een algemene contour bevat van de inhoud van  $I$ . Naarmate  $i$  groter wordt, gaan de randen steeds meer detail bevatten. Ten slotte combineren we alle randen door een gewogen max-operatie toe te passen. Elke randfeature  $R_i$  krijgt een gewicht, omgekeerd evenredig met  $i$ . Randen met meer algemene contouren krijgen dus een hoger gewicht dan randen die meer details bevatten. De resulterende afbeelding is verkregen door voor elke pixel het maximum in de gewogen randen te vinden:  $I_R(x, y) = \max_i\{w_i * R_i(x, y)\}$ . De methode is beschreven in algoritme 1 en weergegeven in figuur 5.5.

---

**Algorithm 1:** Hiërarchische randdetectie.

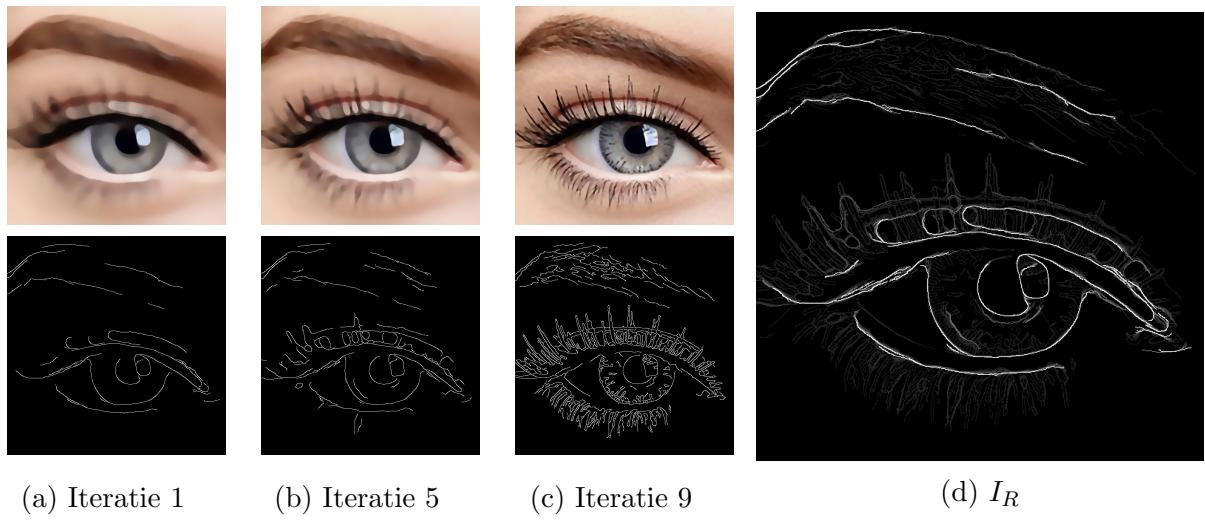
**Input:**  $I$ : de input-afbeelding  
**Output:**  $I_R$ : de hiërarchische randdetectie

```

1:  $I_R \leftarrow$  zwarte afbeelding
2: for  $i$  in  $1..n$  do
3:    $\sigma_s \leftarrow 11 - 10 \frac{i}{n}$ 
4:    $F_i \leftarrow RollingGuidance(\sigma_s, \sigma_r = 10, iterations = 4)$ 
5:    $R_i \leftarrow Canny(F_i, threshold_1 = 50, threshold_2 = 150)$ 
6:    $w_i \leftarrow \frac{1}{i+1}$ 
7:    $I_R \leftarrow \max(I_R, w_i * R_i)$ 
8: end for

```

---



Figuur 5.5: Hiërarchische randdetectie met  $n = 10$ ,  $F_i$  (bovenaan) en  $R_i$  (onderaan)

# Hoofdstuk 6

## Generatie

Dit hoofdstuk bespreekt het tweede deel van de pipeline, de manier waarop het algoritme een grid genereert. In deze thesis zijn drie methodes onderzocht: periodische tesselatie, tesselatie via seedpoints en recursieve gridverdeling.

### 6.1 Periodische tesselatie

De meest voor de hand liggende manier om een grid te genereren is via een periodische tesselatie, waarbij elke tile dezelfde vorm en afmetingen heeft. Een voorbeeld hiervan is een regulier grid, zoals weergegeven in figuur 6.1 of figuur 5.4c. Dit soort grid geeft goede resultaten en is makkelijk te genereren maar houdt geen rekening met de inhoud van de doelafbeelding. Daarnaast voldoet het ook niet aan de stijlvereisten in deze thesis, het grid moet de stijl van Mondriaan benaderen.



Figuur 6.1: Een matching via een zeer fijne reguliere gridverdeling.

### 6.2 Seedpoints

De seedpoint methode is een manier om een willekeurig grid te genereren op basis van mutaties. De methode start met het zoeken van een aantal interessante punten in  $I_T$ . Op deze plaatsen worden target seedpoints geplaatst, dat zijn patches met minimale afmetingen en vormen de eerste tiles van het grid. Deze patches hebben dus de grootte van het kleinst mogelijk fabriceerbare puzzelstuk en kunnen gezien worden als de zaadjes van het grid. Vervolgens wordt er voor elke target seedpoint een overeenkomstige source seedpoint geplaatst in  $I_S$ , door de best mogelijke match te zoeken. Ten slotte gaat het algoritme de target patches iteratief muteren door ze te verplaatsen, schalen of roteren, en worden de overeenkomstige source patches mee geüpdatet. De keuze van mutatie, de keuze van seedpoint, de stapgrootte en het aantal mutaties hangt af van willekeurige kansen. De mutaties zelf kunnen onderhevig zijn aan beperkingen, om de generatie van het grid te leiden in een bepaalde richting. Tijdens de iteratieve mutatie is het mogelijk dat nieuwe seedpoints toegevoegd worden in nog niet opgevulde delen van het grid.

### 6.2.1 Target seedpoints

Target seedpoints zijn de plaatsen waar de eerste tiles gezet worden en groeien met behulp van mutaties. Tijdens het spawnen van target seedpoints moet er rekening gehouden worden met de onderlinge afstand van seedpoints. Indien twee seedpoints te dicht bij elkaar liggen, kunnen de patches niet goed groeien, wat resulteert in een minder optimale verdeling. Daarnaast willen we dat de dichtheid aan patches groter is in regio's met veel detail. In eerder vlakke regio's verkiezen we minder en grotere patches. Voor performantie redenen zijn de target seedpoints in een regulier grid datastructuur geplaatst.

#### Initiële plaatsing

De initiële locatie van de target seedpoints kan op verschillende manieren gekozen worden, maar de methodes verdeeld in twee categorieën:

- **Greedy**

Een greedy algoritme neemt steeds de best mogelijke optie, maar kan vast blijven steken in een lokaal optimum. Deze methode werkt op basis van saliency, zoals besproken in hoofdstuk 2. De maximum waarde van de saliency map geeft de locatie aan van de eerste seedpoint. Daarna wordt de regio rond de maximum waarde in de saliency map gemaskeerd met een cirkel met straal  $r$ , zodat de volgende maximum waarde niet te dicht bij de vorige ligt. Deze stappen worden herhaald tot het gewenste aantal seedpoints is bereikt of tot dat er geen seedpoints meer geplaatst kunnen worden. Deze methode garandeert dat alle seedpoints een afstand  $r$  van elkaar liggen en dat ze zich bevinden op plaatsen met opvallende eigenschappen.

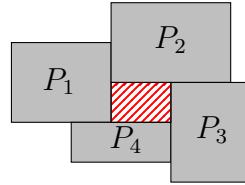
- **Regulier**

Deze methode start met de saliency map te verdelen in cellen met een regulier grid. Vervolgens wordt er een seedpoint geplaatst in elke cel van het regulier grid, op voorwaarde dat de cel interessant genoeg is. Dit wordt bepaald door de saliency waarden in de cel op te tellen en na te kijken of deze waarde een bepaalde drempel overschrijdt. Indien de cel de controle passeert, wordt de seedpoint geplaatst op de locatie van de maximale saliency waarde in de cel. Deze methode is snel aangezien ze parallel uitgevoerd kan worden, maar garandeert geen minimale onderlinge afstand. Alternatief kan de seedpoint ook steeds in het centrum van de cel geplaatst worden.

#### Dynamische plaatsing

Tijdens het muteren kan het zijn dat de patches in een soort van deadlock komen, zoals voorgesteld in figuur 6.2. De vier patches  $P_1..P_4$  mogen onderling niet overlappen en belemmeren elkaar om de rood gearceerde oppervlakte op te vullen. Om dit op te lossen zijn er twee mogelijke oplossingen.

1. De eerste oplossing is dat twee patches –zoals  $P_2$  en  $P_3$ – samen muteren om de lege oppervlakte op te vullen. In dat geval zou  $P_3$  moeten krimpen en  $P_2$  moeten uitzetten. Deze oplossing is niet voor de hand liggend om te implementeren, aangezien er gedetecteerd zou moeten worden dat er deadlocks bestaan, welke patches de reden zijn voor de deadlock en welke het probleem kunnen oplossen.



Figuur 6.2: Een deadlock tijdens seedpoint grid generatie.

2. De tweede methode houdt voor elke cel van de reguliere grid datastructuur bij wat de dekking is van die cel. Dit is de oppervlakte van de cel die opgevuld is door patches, genormaliseerd door de oppervlakte van de cel. Indien de dekking niet 100% bereikt heeft en niet meer stijgt na een aantal iteraties, is er vermoedelijk een deadlock. Deze wordt opgelost door een patch toe te voegen in de lege regio. Deze patch zal dan muteren tot te lege oppervlakte is opgevuld.

### 6.2.2 Source seedpoints

Source seedpoints zijn de alter ego's van de target seedpoints, maar worden geplaatst in  $I_S$  in plaats van  $I_T$ . Source seedpoints muteren mee met de target seedpoints. Indien een target patch groeit, zal de overeenkomstige source patch mee groeien. De initiële source seedpoints worden geplaatst door de patch van de target seedpoint te matchen met  $I_S$ . Vanaf dan worden mutaties van de source patches bepaald door mutaties van hun overeenkomstige target patches.

Indien een mutatie van een target patch zorgt voor een overlap van zijn overeenkomstige source patch met een andere source patch, kan de mutatie verworpen worden, of wordt de source patch verplaatst door de target patch opnieuw te matchen met  $I_S$ .

### 6.2.3 Mutaties

Tijdens elke iteratie  $i$  worden er  $n_p$  patches gekozen om te muteren. Voor elke patch  $p_i$  worden er dan  $n_m$  verschillende mutaties uitgeprobeerd. Voor elke mutatie wordt bekeken of de mutatie verworpen moet worden, als bijvoorbeeld de mutatie voor overlap zorgt of buiten de randen van  $I_T$  valt. Indien de mutatie geaccepteerd is, wordt de afstand berekend tussen de nieuwe target patch  $P_T$  en de nieuwe source patch  $P_S$ . Van al de geaccepteerde mutaties wordt diegene met de beste match tussen  $P_T$  en  $P_S$  uitgevoerd.

De mogelijke mutaties worden hieronder opgesomd. Elke mutatie heeft ook een stapgrootte  $\delta$ .

- **Translatie** (horizontaal / verticaal)

Deze mutatie verplaatst de target en source patches. Het teken van de stapgrootte bepaalt de richting van de translatie (links of rechts, boven of onder).

- **Schalering** (boven / onder / links / rechts)

Deze mutatie verplaatst een van de randen van de target en source patch en zorgt zo voor een schalering. Het teken van de stapgrootte bepaalt of de schalering een

uitzetting of inkrimping is. Omdat het grid uiteindelijk volledig bedekt moet zijn met patches, is de kans op uitzetting groter dan op inkrimping. Daarnaast is de kans op schalering groter dan op translatie of rotatie.

#### • Rotatie

Deze mutatie werkt enkel op de source patch. Zoals vermeld in hoofdstuk 5 zijn er  $n_r$  geroteerde versies van  $I_S$  berekend en houdt elke source patch een rotatie index bij die aangeeft welke geroteerde  $I_S$  de patch gevonden kan worden. Deze mutatie verhoogt of verlaagt deze index met 1, afhankelijk van het teken van de stapgrootte.

### 6.2.4 Constraints

Tot nu toe muteren alle patches onder het motto “elke patch voor zich”. Dit werkt redelijk maar resulteert in drukke grids zonder enige symmetrie, uitlijning of orde. De stijl van het grid kan vergeleken worden met het werk van Mondriaan in figuur 6.3, maar deze thesis streeft eerder naar ordelijk uitgelijnde grids, zoals in figuur 2.3. Om dit te bereiken moeten er dus een aantal constraints opgelegd worden tijdens het muteren van target patches. Een voorbeeld van zo een constraint zou de randen van twee nabijgelegen patches vergelijken. Indien twee randen tegen elkaar liggen of zo goed als uitgelijnd zijn, kan er een constraint op deze randen gelegd worden. Deze constraint oefent een magneetkracht uit die beide randen uitgelijnd probeert te houden maar –indien te stapgrootte te groot is– nog steeds de mogelijkheid biedt om aan de constraint te ontsnappen. Deze methode zorgt ervoor dat de resulterende grid geneigd is om uitgelijnde patches te vertonen.



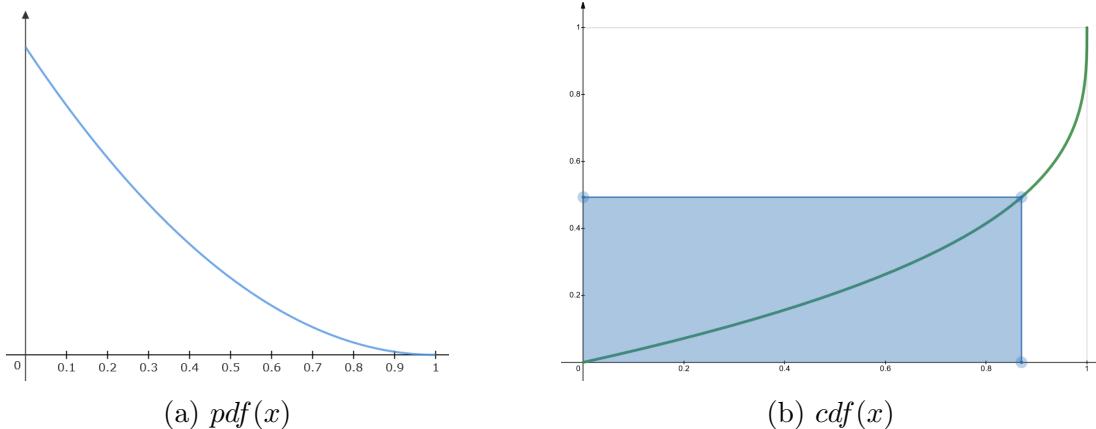
Figuur 6.3:  
Tableau no. 1  
door Piet Mondriaan

### 6.2.5 Conclusie

Deze methode is lang onderzocht geweest maar bleek uiteindelijk niet optimaal te zijn voor het genereren van Mondriaan-achtige grids. Het vullen van lege ruimtes is een moeilijke taak, omdat de tiles enkel rechthoekig mogen zijn en de lege ruimte soms kleiner is dan de minimum dimensies van een patch. Daarnaast is het muteren een computatief intensieve taak en werken de constraints niet optimaal. Dit leidde tot een volledig alternatieve manier van gridgeneratie: recursieve gridgeneratie.

## 6.3 Recursieve gridgeneratie

Waar de seedpoint methode begon met een aantal kleine patches en deze liet groeien, gaat recursieve gridgeneratie beginnen met een grote patch die het volledige vlak opvult. Deze patch wordt vervolgens verdeeld in twee kleinere patches, door een splitsingsvlak te kiezen volgens de x-as of de y-as. Deze stap wordt recursief herhaald, tot een stop-criterium is bereikt. Het algoritme is een top-down algoritme en heeft het voordeel dat het vlak ten allen tijde volledig bedekt is. Vanwege de recursieve opdeling kunnen de patches in een boomstructuur geplaatst worden waar de bladeren van de boom de target patches zijn en de knopen de geschiedenis van de opdeling bevatten. Het resulterende grid is afhankelijk



Figuur 6.4: De gebruikte kansverdeling voor het kiezen van patches en zijn geïnverteerde  $cdf$ .

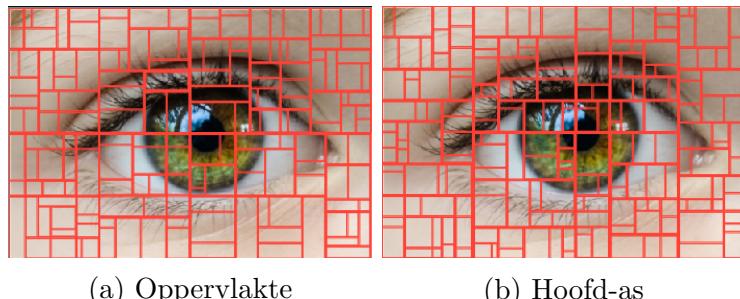
van de splitsingsas ( $x$  of  $y$ ), de plaats van de splitsing (de fractieheuristiek), de patch die gesplitst wordt tijdens de huidige iteratie (de keuzeheuristiek) en het stop-criterium.

### 6.3.1 Keuzeheuristiek

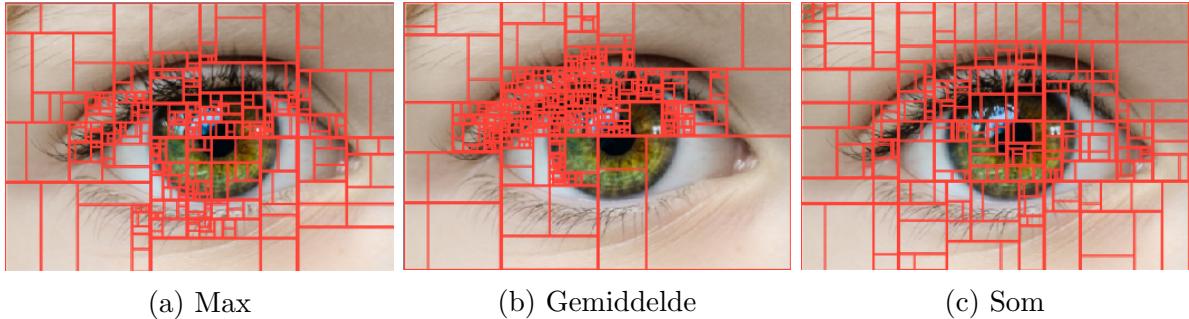
De eerste keuze die gemaakt moet worden is welke patch opgesplitst wordt. Dit wordt behandeld door een keuze-heuristiek. Deze heuristiek heeft een grote invloed op de plaatsing van de patches en de gedetailleerdheid van het grid. De heuristiek start met alle patches te sorteren volgens een bepaalde karakteristiek. Vervolgens wordt de lijst van patches bemonsterd met een kansdichtheidsfunctie  $pdf$  die een voorkeur heeft voor patches die meer vooraan in de lijst liggen en dus betere karakteristieken hebben. De kansdichtheidsfunctie kiest  $n_{splits}$  unieke patches uit de gesorteerde lijst van patches.

#### Kansdichtheidsfunctie

De gebruikte kansdichtheidsfunctie is  $pdf(x) = 3(x-1)^2$  en is te zien op figuur 6.4a. Zoals voor elke kansdichtheidsfunctie geldt dat  $\int_0^1 pdf(x)dx = 1$ . Bemonstering volgens een specifieke kansdichtheidsfunctie wordt gedaan door de inverse cumulatieve kansdichtheidsfunctie  $cdf^{-1} : x = \int_0^y pdf(t)dt$  te bemonsteren met een uniform verdeeld willekeurig getal tussen 0 en 1. Deze is te zien op figuur 6.4b. Hierop is te zien dat een groot bereik aan x-waarden mapt op een klein bereik aan y-waarden. Het bemonsteren van deze func-



Figuur 6.5: Een overzicht van verschillende keuze-karakteristieken.



Figuur 6.6: Een overzicht van verschillende functies binnen de feature karakteristiek.

tie heeft dus voorkeur voor lagere  $y$ -waarden, of elementen vooraan in een gesorteerde lijst, nadat de  $y$ -waarde omgezet wordt in een index. Alternatief kan ook een lineaire ( $pdf(x) = 2(1 - x)$ ) of een uniforme ( $pdf(x) = x$ ) kansdichtheidsfunctie genomen worden.

De keuze van de karakteristiek en de kansdichtheidsfunctie bepaalt de resulterende vorm van het grid. De onderzochte karakteristieken zijn de volgende:

- **Oppervlakte**

De oppervlakte karakteristiek sorteert de patches op basis van hun oppervlakte. Dit zorgt ervoor dat elke patch ongeveer even groot is. Dit plaatst geen beperking op de vorm van de patch (vierkant of rechthoekig), aangezien enkel de oppervlakte meetelt. Af en toe zitten er uitschieters tussen, aangezien de kansdichtheidsfunctie nog steeds willekeurig is, en bij toeval een kleinere patch kan selecteren. Een voorbeeld van een grid gemaakt via deze karakteristiek is te zien in figuur 6.5a.

- **Hoofdas**

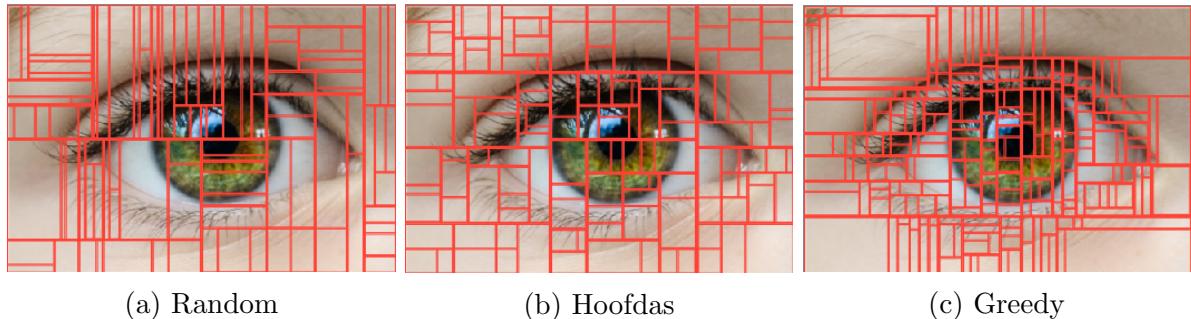
Deze karakteristiek sorteert de patches op basis van de lengte van hun langste as. Visueel lijkt deze karakteristiek fel op de oppervlakte karakteristiek, maar de resulterende patches zijn minder verschillend van vorm. Bij de oppervlakte karakteristiek zouden bijvoorbeeld een vierkantige en een smalle, langwerpige patch dezelfde oppervlakte hebben en dus gelijkaardig zijn. Bij deze karakteristiek zou de langwerpige patch een van de beste kandidaten zijn om opgesplitst te worden. Een figuur van de hoofdas karakteristiek is te vinden in figuur 6.5b.

- **Feature**

De vorige karakteristieken verdelen het volledige vlak vrij uniform op in patches. Dit heeft weg van de stijl van Mondriaan, maar meer variatie in de grootte van patches zou welkom zijn. Daarnaast willen we dat grote patches eerder vlakke regio's van  $I_T$  bedekken, terwijl kleine patches plaatsen bedekken met meer detail. Hiervoor wordt de feature karakteristiek gebruikt. Deze karakteristiek neemt een bepaalde feature en snijdt de patch uit deze feature map. Vervolgens wordt de inhoud van de feature patch bekeken om de waarde van de karakteristiek te bepalen. De functie om de waarde te bepalen is een van de volgende:

- **Maximum**

Het maximum neemt de maximale waarde in de feature patch en kiest deze als waarde van de karakteristiek. Zoals te zien op figuur 6.6a geeft dit redelijke



Figuur 6.7: Een overzicht van verschillende splitsings-heuristieken.

resultaten maar is er een neiging om clusters te vormen waar het grid sterk onderverdeeld is.

- **Som**

De som telt alle waarden in de feature patch op en kiest deze als waarden van de karakteristiek. Merk op dat grotere patches meer waarden hebben om op te tellen en dus vaak een grotere som zullen hebben. Dit heeft een voordeel en een nadeel, aan de ene kant voorkomt het dat er een te grote focus is op plaatsen waar er hoge feature-waarden liggen, maar aan de andere kant heeft het meer de neiging om grote patches te verdelen. Een voorbeeld is zichtbaar in figuur 6.6c.

- **Gemiddelde**

Het gemiddelde is een genormaliseerde vorm van de somfunctie. Deze versie onafhankelijk van de grootte van de patch. Deze methode legt een zeer grote focus op plaatsen waar de feature een hoge waarde heeft. Aangezien grote patches minder kans hebben om relatief evenveel hoge waarden te hebben als de kleine patches, gaat het grid sterk geconcentreerd zijn in clusters zoals in figuur 6.6b.

De gekozen feature geeft ook verschillende resultaten. Bij voorkeur geeft de feature een sterke responsie op plaatsen waar detail aanwezig is. Voorbeelden zijn saliency, Sobel randdetectie of hiërarchische randdetectie. In figuur 6.6 is saliency gebruikt als feature.

### 6.3.2 Splitsingsas-heuristiek

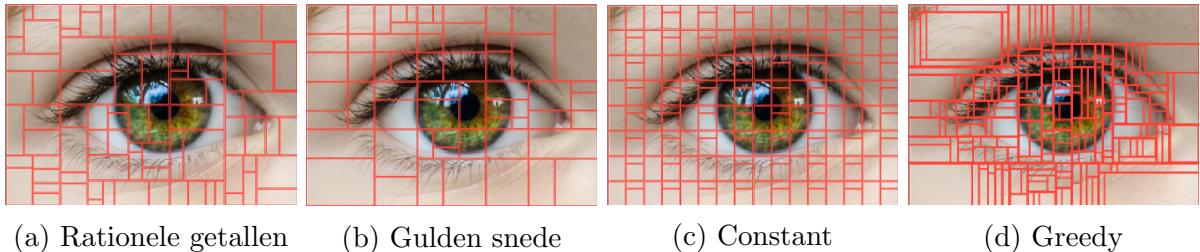
Na de keuze van patches moet er voor elke patch beslist worden of de patch horizontaal of verticaal gesplitst wordt. Er zijn drie mogelijkheden:

- **Random**

De as wordt willekeurig gekozen. Eventueel wordt er aan een as een hogere kans toegekend voor artistieke doeleinden. Dit leidt algemeen gezien tot slechte verdelingen met zeer langwerpige patches.

- **Hoofdas**

De patch wordt steeds gesplitst volgens de langste as, zodat de resulterende patches niet te langwerpig worden. Dit geeft betere resultaten met minder langwerpige patches.



(a) Rationale getallen      (b) Gulden snede      (c) Constant      (d) Greedy

Figuur 6.8: Een overzicht van verschillende fractie-heuristieken.

- **Greedy**

De patch wordt zowel horizontaal als verticaal gesplitst in patch  $P_a$  en patch  $P_b$ . Van deze patches wordt een match gezocht in  $I_S$  en de afstanden tussen  $P_a$ ,  $P_b$  en hun matches wordt samengegeteld. De splitsingsas met de beste gecombineerde afstand wordt gekozen als splitsingsas.

Algemeen gezien zou de greedy methode de beste resultaten moeten geven, maar dit hangt ook af van de fractie heuristiek. De kwaliteit van de matching is namelijk sterk afhankelijk van de plaats waar de splitsing uitgevoerd wordt. Een slechte plaatsing heeft dus invloed op het resultaat van de greedy methode. Voor een uniform verdeeld grid zal de hoofdas methode steeds goede resultaten geven.

### 6.3.3 Fractie-heuristiek

Nu de patches en de splitsingsas gekozen zijn, moet er enkel nog beslist worden waar de patch gesplitst wordt. Dit gebeurt via een fractie-heuristiek. Deze heuristiek heeft een grote invloed op de vorm van de patches. De fractie  $f$  is een getal tussen 0 en 1 en zet het splitsingsvlak op  $f * 100\%$  van de te splitsen zijde van de patch. De volgende opties zijn getest:

- **Rationale getallen**

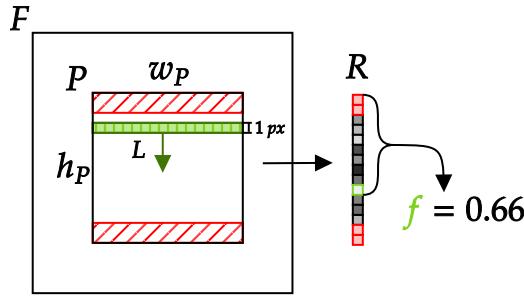
Bij deze methode wordt een willekeurig rationeel getal gekozen uit  $\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}\}$ . Dit geeft vaak mooi verdeelde grids zonder repetitie, en die meerdere uitgelijnde patches hebben, zoals in figuur 6.8a.

- **Gulden snede**

De gulden snede  $\phi \approx 1.618$  is volgens de geschiedenis een verhouding dat een esthetische meerwaarde biedt. In deze methode word een patch onderverdeeld in patches  $P_a$  en  $P_b$  waarvoor geldt dat  $\frac{a+b}{a} = \frac{a}{b}$  of  $\frac{a+b}{b} = \frac{b}{a}$ , waarbij  $a$  en  $b$  de lengtes van de zijden dwars op de splitsingsas van respectievelijk  $P_a$  en  $P_b$  zijn. Het resulterende grid bevat zeer veel uitlijningen, omdat er slechts twee fracties mogelijk zijn (zie figuur 6.8b).

- **Constante**

Indien een constante gebruikt wordt is het resulterende grid zeer simpel. Merk op dat indien  $f = 0.5$  wordt gekozen als fractie, het grid een regulier grid wordt. Indien er dynamisch met deze constante wordt gespeeld, kan men speciale grids verkrijgen. Zie bijvoorbeeld figuur 6.8c, hier verkregen we een regulier grid door  $f = 0.5$  te



Figuur 6.9: De werking van adaptieve splitsing.

kiezen en vervolgens zijn een aantal patches opgesplitst met  $f = 0.3$ . Deze soort grids zijn zeer repetitief en lijken minder op een Mondriaan verdeling.

- **Adaptief**

De adaptieve methode is de meest complexe. Hier wordt template matching gebruikt om de meest interessante locatie te vinden om de patch te splitsen. Afhankelijk van de splits-heuristiek zal deze methode anders reageren. Indien een greedy splits-heuristiek gekozen is, zal deze methode de patch zowel horizontaal als verticaal splitsen en de beste splitsings-as kiezen. Aangezien deze methode een grote contributie is in deze thesis, wordt ze apart besproken in sectie 6.3.4.

### 6.3.4 Adaptieve splitsing

De simpelste manier om  $I_T$  op te vullen met patches uit  $I_S$  is door het grid onafhankelijk te generen van de gekozen  $I_T$  en pas later  $I_T$  te gebruiken tijdens het matchen. Hierdoor zijn alle patches vaak uniform verdeeld over het vlak, zonder een al te grote variatie in de afmetingen van de patches. Aan de andere kant kan dit zorgen voor een groot aantal patches in homogene regio's van  $I_T$ , of weinig patches in gedetailleerde regio's van  $I_T$ , wat het moeilijker maakt om een geschikte patch te vinden in  $I_S$ .

Adaptieve splitsing probeert  $I_T$  in rekening te brengen tijdens de generatie door patches te splitsen op interessante plaatsen, zoals op randen. Hiervoor kan een simpele Sobel filter gebruikt worden, maar zoals eerder vermeld, bevat deze feature geen informatie over de belang van randen. Net om deze reden is hiërarchische randdetectie bedacht (zie sectie 5.5). De bedoeling is dat de meest algemene contouren van  $I_T$  eerst gemodelleerd worden in het grid, door de splitsingsvlakken van de eerste patches uit te lijnen met de randen van de algemene contour. Vervolgens worden deze patches verder onderverdeeld volgens minder algemene contouren. Het resultaat is een grid waarvan de meeste patches uitgelijnd zijn volgens de algemene vorm in  $I_T$ , homogene vlakken bedekt worden door grote patches en details in  $I_T$  bedekt zijn door kleine patches.

#### Methode

De manier waarop de fractie  $f$  bepaald wordt is weergegeven in figuur 6.9. De volgende uitleg is voor een splitsing volgens de x-as, maar is gelijkaardig voor een splitsing volgens de y-as. Stel dat we een patch  $P$  ( $w_P \times h_P$ ) hebben uit  $I_T$ , en we de hiërarchische randdetectie gebruiken als feature  $F$ . We convolueren dan een witte lijn  $L$  ( $w_P \times 1 \text{ px}$ ) over  $P$

zoals uitgelegd in sectie 2.3.4 over template matching. Het resultaat van deze matching is een vector  $R$  ( $1 \text{ px} \times h_P$ ) waarbij  $R_i$  aangeeft hoe goed de  $i^{de}$  rij van  $P$  matcht met een horizontale lijn. Aangezien  $P$  uit  $F$  komt, en  $F$  de randen van  $I_T$  bevat, gaat een hoge waarde  $R_i$  aangeven dat er een horizontale rand aanwezig is op de  $i^{de}$  rij van  $P$ . De rij  $i$  met de maximale waarde wordt dan gekozen en de fractie  $f$  kan dan berekend worden volgens  $f = \frac{i}{h_P}$ .

Zoals eerder vermeld gaan in hiërarchische randdetectie de randen van meer algemene contouren een hoger gewicht hebben, en dus helderder zijn. Hierdoor gaan deze randen een sterkere responsie hebben tijdens het template matchen van de witte lijn  $L$ , en dus verkozen worden boven meer gedetailleerde randen (die een lager gewicht hebben). Dit zorgt ervoor dat in het begin van de generatie het grid zich gaat uitlijnen op de algemene contour, en pas later zich gaat uitlijnen op de details in  $I_T$ .

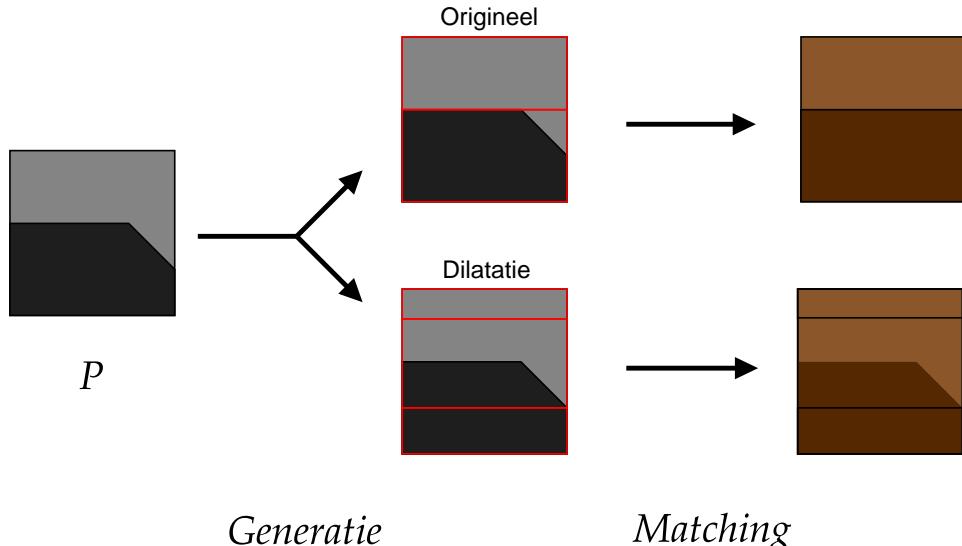
## Marges

Aangezien patches fabriceerbaar moeten zijn, kunnen niet alle fracties gekozen worden. Te kleine of te grote fracties resulteren in te kleine patches en moeten dan verworpen worden, wat verloren werk is. Hiervoor worden er marges geplaatst in  $P$ , zoals aangeduid in figuur 6.9 met de rood gearceerde oppervlaktes. Indien de beste match met  $L$  in deze regio's gevonden zou worden, zou dit resulteren in een te kleine patch. Hierdoor negeren we deze regio's tijdens een template matchen, wat een kleine verbetering is qua performantie, te kleine patches voorkomt, en bovenop er voor zorgt dat dezelfde horizontale rand in  $I_T$  geen twee keer gebruikt wordt.

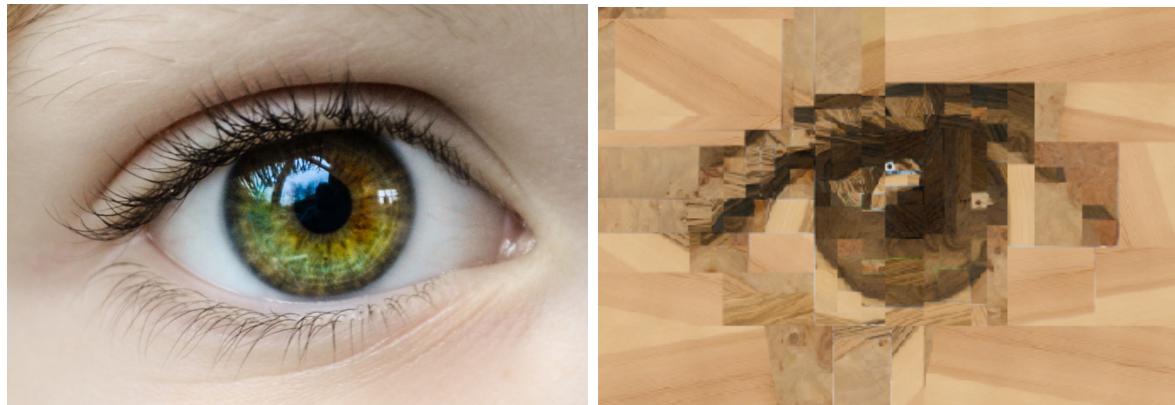
## Dilatatie

Het plaatsen van splitsingsvlakken op randen in de target afbeelding zorgt er voor dat tijdens de matching fase zoveel mogelijk patches een uniforme inhoud hebben. Op deze manier kunnen patches beter gematcht worden. Tijdens het testen van de adaptieve gridverdeling op basis van hiërarchische randdetectie, viel het op dat de patches inderdaad vaak een redelijk uniforme inhoud hebben, maar aan de randen van de patch er zich vaak details bevinden. Omdat deze details zeer klein zijn in vergelijking met de resterende uniforme inhoud van de patch, worden de details niet goed gematcht. Dit resulteert in scherpe, blokvormige randen zoals zichtbaar in figuur 6.11b. Deze uitleg is ook visueel getoond in figuur 6.10. Indien we een patch  $P$  moeten opsplitsen, gaat het splitsingsvlak waarschijnlijk gekozen worden op de plaats waar het lichtgrijze stuk overgaat op het donkergrijze stuk. Het resultaat is dat er nu een lichtgrijze patch is, en een donkergrijze patch met aan de randen een klein lichtgrijs stukje. Als we deze verdeling gaan matchen, kan het zijn dat het lichtgrijs stukje verwaarloosd wordt door de overvloed aan donkergrijze pixels. Het resultaat is dat het lichtgrijs stukje niet zichtbaar is in de resulterende afbeelding.

Een alternatieve methode gaat een extra bewerking uitvoeren op de hiërarchische randdetectie. Deze bewerking gaat de aanwezige randen in  $F_{HR}$  op een speciale manier dilateren en is zichtbaar in figuur 6.12. Dilatatie is origineel een manier om pixels toe te voegen aan de randen van een afbeelding. Hierdoor worden de randen dikker. De aangepaste



Figuur 6.10: De werking van gedilateerde hiërarchische randdetectie.



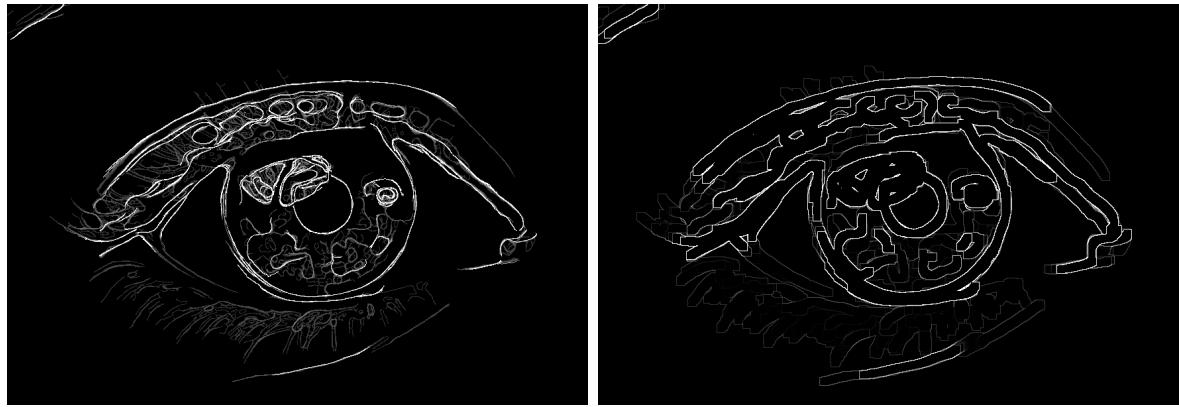
(a) De target afbeelding.

(b) De resulterende afbeelding.

Figuur 6.11: Een voorbeeld van een slechte matching door adaptieve gridverdeling.

dilatatie in deze thesis gaat deze operatie twee keer apart uitvoeren, met een verschillende dilatatie-sterkte. Het resultaat is twee afbeeldingen met dikkere randen, waarvan de randen van de ene afbeelding net iets dikker zijn dan de andere. Indien we de afbeelding met de smallere randen dan aftrekken van de afbeelding met de dikkere randen verkrijgen we figuur 6.12b. Deze afbeelding kan gezien worden als een bewerking op figuur 6.12a waarbij de randen omgezet worden in holle buisjes en waarbij de originele randen de middellijn van de buisjes volgen.

De bedoeling van deze bewerking is weergegeven in figuur 6.10. Door de dilatatie gaan de patch  $P$  niet meer gesplitst worden op de overgang tussen het lichtgrijze en het donkergrizige deel, maar er net iets boven en er niet iets onder. Aangezien het aantal donkergrizige pixels en het aantal lichtgrijze pixel nu even groot is, gaan beide stukken een even groot aandeel hebben tijdens het matchen en gaat het kleine lichtgrijze stukje niet genegeerd worden. De resultaten van dilatatie zijn te vinden in hoofdstuk 8.



(a) De hiërarchische randdetectie feature.

(b) De gedilateerde hiërarchische randdetectie feature.

Figuur 6.12: Het verschil tussen hiërarchische randdetectie en zijn gedilateerde variant.

### 6.3.5 Stop criterium

Het stop criterium hangt vooral af van de gebruiker. Er kan bijvoorbeeld een limiet gesteld worden op het aantal patches. Puzzels komen namelijk vaak in vaste hoeveelheden aan puzzelstukken. Daarnaast kan de generatie van een grid op elk moment stopgezet worden, aangezien de recursieve gridverdeling garandeert dat het vlak ten allen tijden volledig opgevuld is met patches. Merk op dat er wel een bovengrens staat op het aantal patches, die bepaald is door de minimale afmetingen van een patch en de werkelijke afmetingen van  $I_T$ .

### 6.3.6 Postprocessing

Na (maar ook tijdens) de automatische generatie van het grid kunnen er manuele aanpassingen gemaakt worden. Patches kunnen terug samengevoegd worden indien ze te diep gesplitst zijn. Dit komt doordat de geschiedenis van het splitsen bijgehouden wordt in een boomstructuur. Patches kunnen ook verplaatst worden, maar dit breekt de garantie van een volledig opgevuld vlak.

# Hoofdstuk 7

## Matching

Dit hoofdstuk bespreekt de manier waarop patches uit  $I_T$  gematcht worden met stukken uit  $I_S$ . De methode en de gebruikte features zijn grotendeels afgeleid van Iseringhausen et al. (2020) omdat deze thesis zich vooral focust op de gridgeneratie.

### 7.1 Methode

Het doel van matching is het vinden van een patch  $P_S$ , afkomstig uit  $I_S$ , door een target patch  $P_T$  uit  $I_T$  te template matchen met  $I_S$ . De pseudocode van het algoritme is te vinden in algoritme 2.

#### 7.1.1 Template matching

Het algoritme start met de target patch  $P_T$  te template matchen met elke geroteerde source afbeelding. Dit gebeurt niet via de originele  $I_S$  en  $I_T$ , maar maakt gebruik van speciaal gekozen features  $\mathbf{F}_T$  en  $\mathbf{F}_S$ , zoals vermeld in sectie 5.2.  $\mathbf{F}_T$  is een vector van  $n_f$  features. In deze thesis bestaat  $\mathbf{F}_T$  uit een intensiteit feature  $F_{T,intensiteit}$  en een feature  $F_{T,rand}$  dat de randen van  $I_T$  bevat.  $\mathbf{F}_S$  is gelijkaardig aan  $\mathbf{F}_T$ , maar is een matrix in plaats van een vector, aangezien de intensiteit feature  $F_{S,intensiteit}$  en de rand feature  $F_{S,rand}$  berekend is voor  $n_r$  rotaties. Daarnaast heeft elk van de  $n_f$  features een gewicht dat opgeslagen is in een gewichtvector  $\mathbf{W}_f$ .

#### 7.1.2 Template matching

Het algoritme gaat voor elke rotatie  $i_r$  de features uit  $\mathbf{F}_T$  template matchen met de overeenkomstige features uit  $\mathbf{F}_S[i_r]$ , volgens een afstandsfunctie  $\delta$ . Merk op dat de features uit  $\mathbf{F}_T$  dienen als de template in de template matching en dus bijgesneden worden zodat enkel het stuk waar  $P_T$  mee overlapt nog overblijft, zoals in lijn 7 van algoritme 2. Vervolgens worden de responsen van de template matching gewogen volgens  $\mathbf{M}_S$ , samengegeteld en ten slotte opgeslagen in  $\mathbf{R}[i_r]$ .  $\mathbf{R}$  gaat dus  $n_r$  responsen bevatten, waarbij elke respons een gewogen sommatie is van de template matching van  $\mathbf{F}_T$  met  $\mathbf{F}_S[i_r]$ . Dit is neergeschreven in lijnen 1 tot en met 10 in algoritme 2.

### 7.1.3 Keuze van rotatie index

Nadat de gewogen responsen berekend en opgeslagen zijn in  $\mathbf{R}$ , gaan we voor elke respons de beste locatie voor  $P_S$  zoeken. Dit komt overeen met het zoeken van de beste locatie van  $P_S$  in alle geroteerde varianten van  $I_S$ . Tijdens het zoeken naar een locatie voor  $P_S$  in  $\mathbf{R}[i_r]$  moet eventueel rekening gehouden worden met eerder gevonden source patches. Hiervoor wordt er gebruik gemaakt van de  $n_r$  masks in  $\mathbf{M}_S$ . Dit wordt verder uitgelegd in sectie 7.2.2. Vervolgens wordt de beste locatie en de waarde van de respons op die locatie berekend voor elke respons in  $\mathbf{R}$ . Afhankelijk van de gekozen afstandsfunctie  $\delta$  gebeurt door de minimum- of de maximumwaarde te zoeken in  $\mathbf{R}[i]$ . Ten slotte wordt locatie met de beste waarde gekozen als locatie voor de source patch  $P_S$ . De afmetingen van de source patch zijn dezelfde als die van  $P_T$ .

---

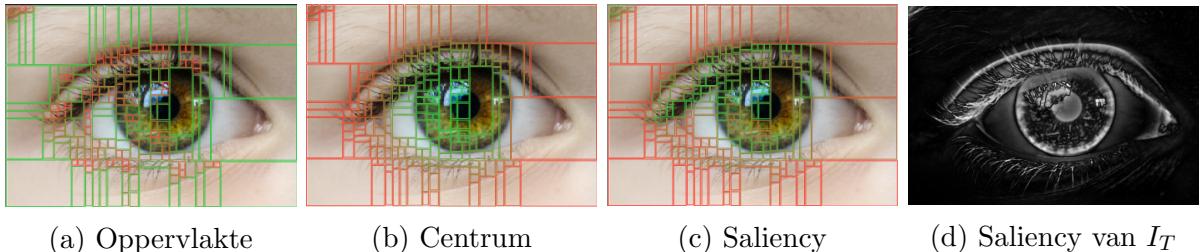
#### Algorithm 2: Patch matching.

---

**Input:**  $P_T = (x, y, w, h)$ : de target patch  
**Input:**  $\mathbf{M}_S$ : een vector van  $n_r$  source masks, één voor elke rotatie  
**Input:**  $\mathbf{F}_S = \{F_{S,intensiteit}, F_{S,rand}\}_1, \dots, \{F_{S,intensiteit}, F_{S,rand}\}_{n_r}\}$ : een vector van  $n_f$  source features  $F_S$  voor  $n_r$  rotaties  
**Input:**  $\mathbf{F}_T = \{F_{T,intensiteit}, F_{T,rand}\}$ : een vector van  $n_f$  target features  $F_T$   
**Input:**  $\mathbf{W}_f = \{w_{egalisatie}, w_{rand}\}$ : een vector van  $n_f$  gewichten, één voor elke feature  
**Input:**  $\delta$ : de afstandsfunctie, in dit geval het kwadratisch verschil  
**Output:**  $P_S = (x, y, w, h, r)$ : de source patch

- 1:  $\mathbf{R} \leftarrow \{\}$ : een vector van  $n_r$  responsen, één voor elke rotatie
- 2: **for**  $i_r$  in  $1..n_r$  **do**
- 3:   **for**  $i_f$  in  $1..n_f$  **do**
- 4:      $F_S \leftarrow \mathbf{F}_S[i_r][i_f]$
- 5:      $F_T \leftarrow \mathbf{F}_T[i_f]$
- 6:      $w_f \leftarrow \mathbf{W}_f[i_f]$
- 7:      $F_{P_T} \leftarrow F_T(P_T)$  ; // Snij de patch uit de huidige feature
- 8:      $\mathbf{R}[i_r] \leftarrow \mathbf{R}[i_r] + w_f * matchTemplate(F_S, F_{P_T}, \delta)$  ; // Tel de gewogen responsen op
- 9:   **end for**
- 10: **end for**
- 11:  $B = (x, P, r) \leftarrow (\infty, (0, 0), 1)$  ; // Een tuple met de huidige beste waarde, locatie en rotatie index
- 12: **for**  $i_r$  in  $1..n_r$  **do**
- 13:    $M_S \leftarrow \mathbf{M}_S[i_r]$
- 14:    $M \leftarrow thresholdBoxFilter(M_S)$
- 15:    $(x, P) \leftarrow minLoc(\mathbf{R}[i_r], M)$
- 16:   **if**  $x < B_x$  **then**
- 17:      $B \leftarrow (x, P, i_r)$
- 18:   **end if**
- 19: **end for**
- 20: **return**  $(B_{P_x}, B_{P_y}, P_{T_w}, P_{T_h}, B_r)$

---



(a) Oppervlakte      (b) Centrum      (c) Saliency      (d) Saliency van  $I_T$

Figuur 7.1: Een overzicht van verschillende sorteer-heuristieken.

### 7.1.4 Parallelisatie

De matching fase is computationeel zonder twijfel de zwaarste fase van het algoritme. Hierdoor is het belangrijk om zo veel mogelijk stappen parallel uit te voeren. De eerste twee lussen in algoritme 2 kunnen allebei geparallelliseerd worden, maar het updaten van  $\mathbf{R}[i_r]$  is een kritische sectie voor de binnenste lus, en mag dus niet door twee threads tegelijk uitgevoerd worden voor eenzelfde rotatie index  $i_r$ . De zoektocht naar de beste rotatie index (het tweede deel van algoritme 2) kan niet triviaal geparallelliseerd worden.

## 7.2 Fabriceerbare matching

Om een fabriceerbare puzzel te maken, is het nodig dat source patches niet overlappen. Hiervoor moet er tijdens het matchen rekening gehouden worden met eerder gematchte patches. Dit legt ook beperkingen op aan het source materiaal. Indien de werkelijke afmetingen van  $I_S$  niet groot genoeg zijn, is het mogelijk dat de matching niet voltooid kan worden. De enige manier om dit tegen te gaan is door er voor te zorgen dat  $I_S$  altijd genoeg materiaal bevat, of door de werkelijke afmetingen van  $I_T$  te verkleinen.

### 7.2.1 Prioriteit

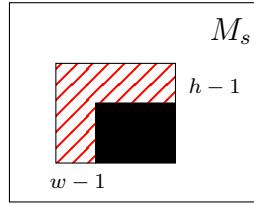
De kwaliteit van de matching gaat zeer afhankelijk zijn van de volgorde in dewelke patches gematcht worden. Naarmate er meer patches gematcht zijn, zal er minder materiaal aanwezig zijn in  $I_S$  en is de kans op een goede match kleiner. Het kan dus interessant zijn om het grid volgens een sorteer-heuristiek te sorteren, en die sortering als volgorde gebruiken tijdens het matchen. Figuur 7.1 toont een aantal mogelijke sorteer-heuristieken.

#### • Oppervlakte

Een voor de hand liggende keuze is de oppervlakte van de patch. Grottere patches bevatten vaak –afhankelijk van de heuristieken tijden de generatie-fase– nauwelijks details of een grote feature zoals de rechtse patches van figuur 7.1a. Indien kleine patches eerst worden gematcht, kunnen ze een goede plaatsing van grotere patches dwarsbomen.

#### • Centrum

De interessante locaties in een afbeelding bevinden zich vaak in het centrum. Door de afstand tussen elke patch en het centrum van  $I_T$  te berekenen kunnen we de patches zo sorteren dat de binnenste patches voorrang krijgen op de buitenste



Figuur 7.2: Extra maskering na template matching om overlap te voorkomen.

patches. Deze methode is weergegeven in figuur 7.1b wordt ook beschreven door Iseringhausen et al. (2020).

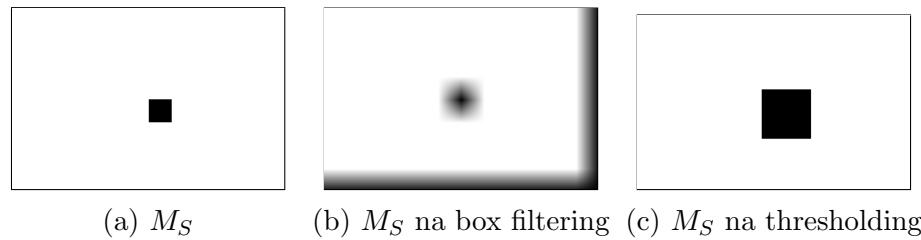
- **Feature**

De beste keuze is vaak om de sortering te doen op basis van een bepaalde feature. Indien het grid bijvoorbeeld gegenereerd is volgens de saliency van  $I_T$ , kunnen de patches ook gesorteerd worden op basis van saliency. Aangezien de adaptieve gridverdeling ervoor zorgt dat locaties met veel saliency meer onderverdeeld worden, gaan de kleinste patches van het grid vaak eerst gematcht worden. Figuur 7.1c toont een sortering op basis van saliency (zie figuur 7.1d).

### 7.2.2 Mask

Het voorkomen van overlappende source patches gebeurt op basis van masks, gelijkaardig als beschreven door Iseringhausen et al. (2020). De template matching in deze thesis gebruikt een ingebouwde functie van OpenCV (2022a), en ondersteunt het gebruik van een mask niet tijdens het template matchen. Hierdoor worden de masks pas gebruikt tijdens het zoeken naar de beste locatie van de source patch. Zoals op lijnen 13 tot en met 15 in algoritme 2, nemen we eerst de correct geroteerde mask  $M_S$  uit de lijst van masks  $\mathbf{M}_S$ . Vervolgens passen we een boxfilter toe op  $M_S$ , thresholden we het resultaat en slagen dit op in  $M$ . Dit is nodig omwille van de volgende reden. Zoals beschreven in sectie 2.3.4, bevat het resultaat van de template matching  $\mathbf{R}[i_r]$  op elke pixel  $(x, y)$  een waarde die aanduid hoe goed een target patch  $P_T$  met afmetingen  $(w \times h)$  matcht met  $I_S$  op de oppervlakte  $(x \rightarrow x + w, y \rightarrow y + h)$ . Aangezien elke pixel in  $\mathbf{R}[i_r]$  representatief is voor een oppervlakte in  $I_S$ , moet de mask  $M_S$  hier aan aangepast worden. Dit is weergegeven in figuur 7.2. Stel dat een deel van  $M$  gemaskeerd is (de zwarte oppervlakte), dan betekent dat dat een patch die oppervlakte in  $I_S$  heeft gereserveerd. Om een nieuwe patch met afmetingen  $(w \times h)$  te plaatsen, moet er een extra ruimte gemaskeerd worden, weergegeven door de rood gearceerde oppervlakte. Plaatsing van een patch binnen deze oppervlakte zou er namelijk voor zorgen dat er overlap optreedt. Enkel als de patch minstens  $w$  pixels links en/of  $h$  pixels boven een gemaskeerd deel in  $M_S$  geplaatst wordt, kan er gegarandeerd worden dat er geen overlap optreedt.

Deze extra maskering gebeurt via een box filter over  $\mathbf{R}[i_r]$  met een grootte van  $(w - 1 \times h - 1)$  en het ankerpunt in de linkerbovenhoek. Dit betekent dat voor pixel  $(x, y)$  de kernel geconvoluteerd wordt over de oppervlakte  $(x \rightarrow x + w - 1, y \rightarrow y + h - 1)$  in  $\mathbf{R}[i_r]$ . Indien een enkele pixel in deze oppervlakte gemaskeerd is (en dus zwart is), zal het resultaat van de filtering niet perfect wit zijn. Na een thresholding operatie worden alle pixels die niet perfect wit zijn, zwart gemaakt en verkrijgen we de gewenste extra



Figuur 7.3: Een overzicht van de verschillende stappen om de source mask te transformeren.

maskering. Deze stappen zijn gevisualiseerd in figuur 7.3. Merk op dat de rechter- en onderkant van figuur 7.3b ook donkere delen bevatten. Dit vormt geen probleem, deze stukken worden van de figuur afgeknipt om ervoor te zorgen dat  $M_S$  en  $\mathbf{R}[i_r]$  dezelfde afmetingen hebben.

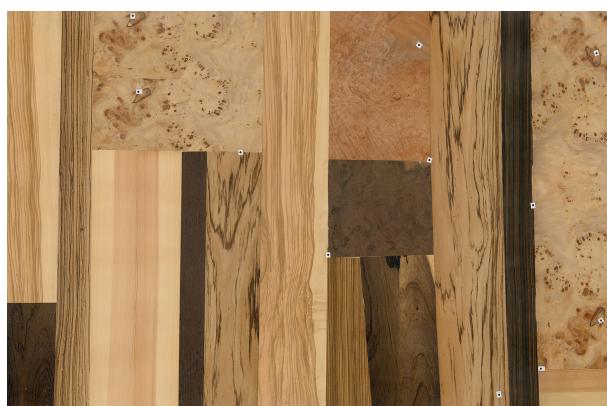
# Hoofdstuk 8

## Resultaten

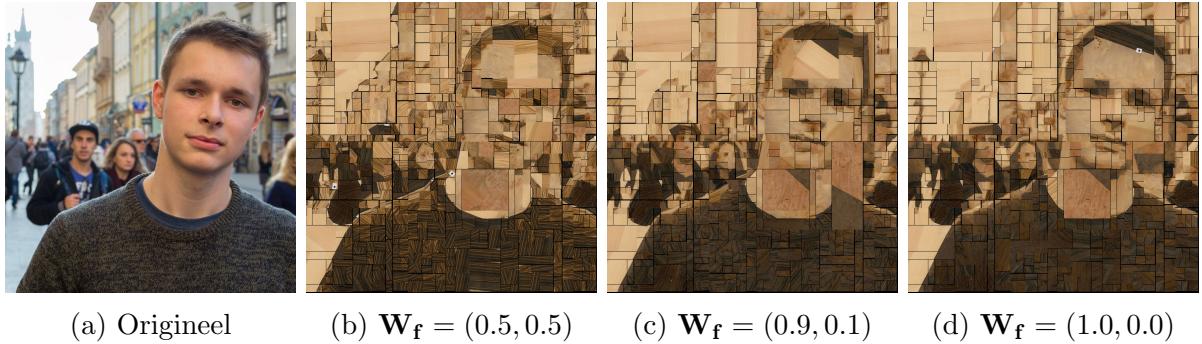
Dit hoofdstuk bespreekt de resultaten van deze thesis. De resultaten onderzoeken het effect van verschillende parameters, features en heuristieken. Tenzij anders vermeld, zijn de parameters ingesteld volgens tabel 8.1. Het standaard source materiaal is weergegeven in figuur 8.1 en komt uit de paper van Iseringhausen et al. (2020). Deze afbeelding is gekozen vanwege zijn rijke textuur met verschillende soorten hout. Daarnaast is deze afbeelding redelijk monochroom, waardoor de intensiteitsfeature gebruikt kan worden. Indien de source afbeelding meer kleur bevat, moeten andere features gebruikt worden. Dit is onderzocht in sectie 8.4.

Parameter	waarde
$w_{egalisatie}$	0.5 (hangt af van $I_T$ )
$w_{intensiteit}$	0.90
$w_{rand}$	0.10
$n_r$	13

Table 8.1: De standaard instellingen voor de parameters.



Figuur 8.1: Het standaard source materiaal, afkomstig uit (Iseringhausen et al., 2020).



Figuur 8.2: De invloed van  $w_{egalisatie}$  en  $w_{intensiteit}$  op de resulterende afbeelding.

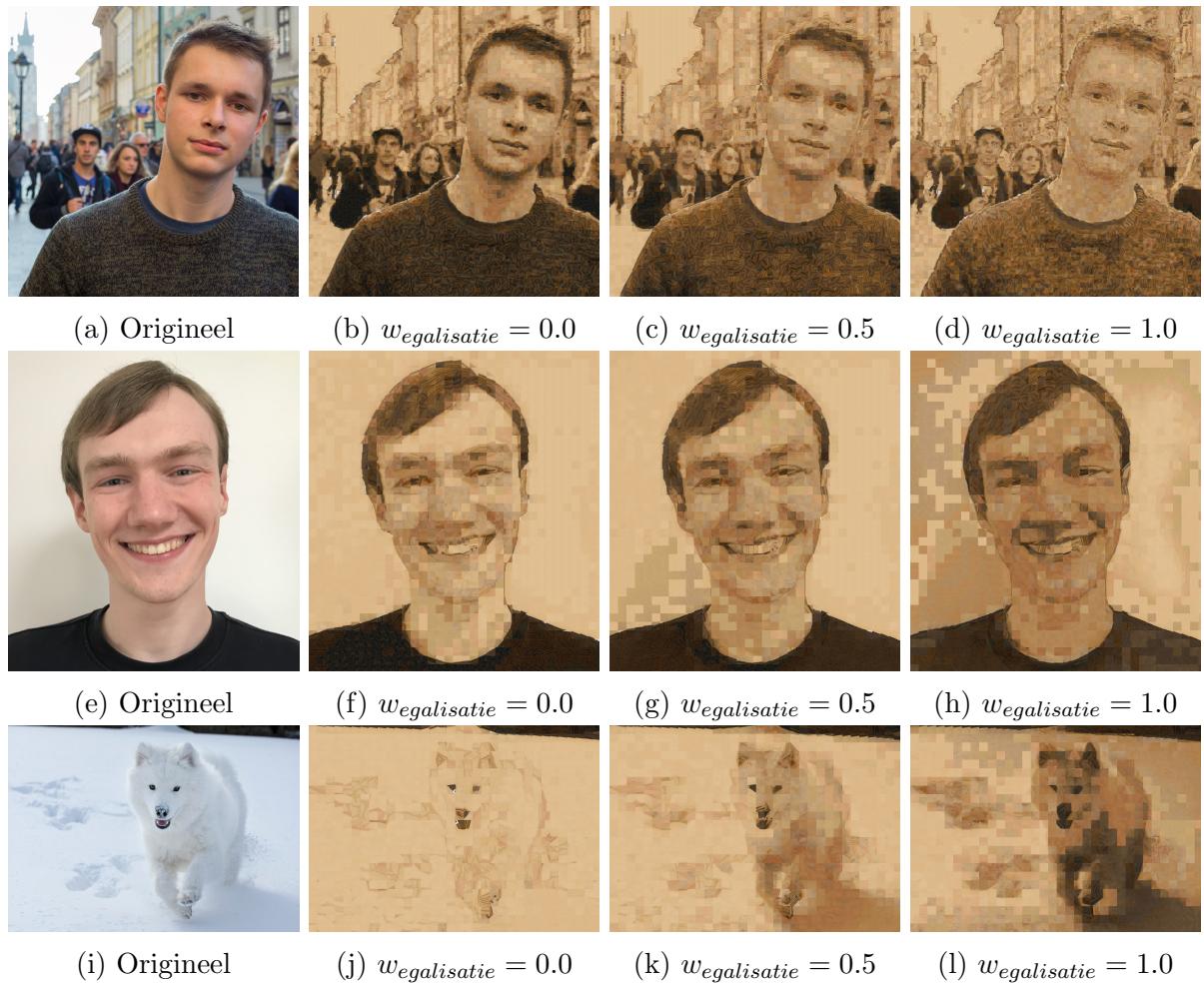
## 8.1 Invloed van feature gewichten

De invloed van  $w_{egalisatie}$  en  $w_{intensiteit}$  op de resulterende afbeelding is uitvoerig besproken door Iseringhausen et al. (2020) en wordt hier dus maar beperkt besproken. Figuur 8.2 toont drie verschillende instellingen voor  $\mathbf{W}_f$ . Indien beide gewichten even belangrijk gekozen worden (figuur 8.2b), is er duidelijk meer aandacht besteed aan de details in de kleren en de gebouwen in de achtergrond. Dit gaat wel ten koste van de kwaliteit van de kleur. Figuur 8.2d besteedt enkel aandacht aan de kleur en laat de edge feature links liggen. De kleuren zijn beter gematcht in deze afbeelding en het algoritme is nog steeds capabel om randen en detail weer te geven. Dit wijkt af van de resultaten in (Iseringhausen et al., 2020). Zij kozen  $\mathbf{W}_f = (0.5, 0.5)$  als optimale waarden voor de feature gewichten, maar onze resultaten geven voorkeur aan  $\mathbf{W}_f = (0.9, 0.1)$  of zelfs  $\mathbf{W}_f = (1.0, 0.0)$ .

## 8.2 Invloed van histogram egalisatie

De invloed van histogram egalisatie is het duidelijkst wanneer de tegels zeer fijn verdeeld zijn. Hierdoor zijn de resultaten in figuur 8.3 gerenderd met een reguliere gridverdeling. De egalisatie factor  $w_{egalisatie}$  varieert tussen 0% en 100%. Afhankelijk van de afbeelding gaat deze egalisatie een ander effect hebben. Bij de eerste twee rijen van figuur 8.3 heeft de originele target image een groot bereik aan kleuren en intensiteiten. Beiden personen hebben donkere kleren, een belichte huid en een felle achtergrond. Hierdoor is het bereik aan intensiteiten van de target afbeelding groter dan die van de source afbeelding in figuur 8.1. Indien dit het geval is zal histogram egalisatie het contrast van de target afbeelding verkleinen, wat vaak resulteert in mindere resultaten. Voor de eerste twee rijen is  $w_{egalisatie} = 0$  duidelijk de beste keuze. In de tweede rij zorgt histogram egalisatie zelfs voor ruis in de achtergrond, die origineel uniform oogt.

Voor target afbeeldingen met een laag contrast en dus een klein bereik aan intensiteiten kan histogram egalisatie wel betere resultaten geven. De derde rij van figuur 8.3 toont een sneeuwhond waarvan de volledige afbeelding zeer helder is. Zonder egalisatie worden enkel lichte stukken hout genomen en zijn de contouren van de hond slecht zichtbaar. Indien de afbeelding geëgaliseerd wordt, verhoogt het contrast en worden de contouren van de hond beter zichtbaar.



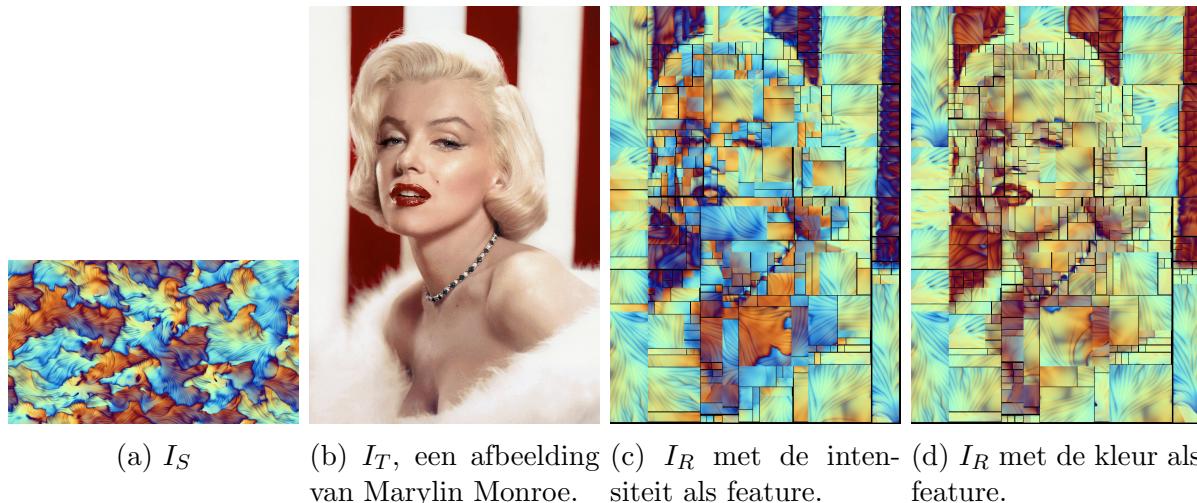
Figuur 8.3: Een rendering van twee aangezichten en een sneeuwhond met  $w_{intensiteit} = 0.75$  en  $w_{rand} = 0.25$ . De equalisatie factor  $w_{egalisatie}$  varieert tussen 0% en 100%.

Uit deze resultaten kan geconcludeerd worden dat histogram equalisatie zeer nuttig kan zijn indien het zorgt voor een hoger contrast in de target afbeelding.

### 8.3 Non adaptief vs adaptief

### 8.4 Kleur vs intensiteit

Voor source images die zeer kleurrijk zijn kan het interessanter zijn om de kleur van  $I_S$  te gebruiken als feature in plaats van de intensiteit. Het kan bijvoorbeeld voorkomen dat twee kleuren dezelfde intensiteitswaarde hebben, waardoor het algoritme er geen verschil tussen kan zien. Figuur 8.4 toont dit aan. De source afbeelding is een abstracte afbeelding met verschillende kleuren. Figuur 8.4c is gerenderd met de intensiteit als feature. De huid van de vrouw heeft een intensiteit die overeenkomt met meerdere kleuren in  $I_S$ , en het algoritme kies dus een willekeurige kleur. Figuur 8.4d is gerenderd met de kleur van  $I_S$  als feature. Niet alleen trekt de foto veel meer op  $I_T$ , ook de huid is nu correct gekleurd en het eerder genoemde probleem is opgelost.



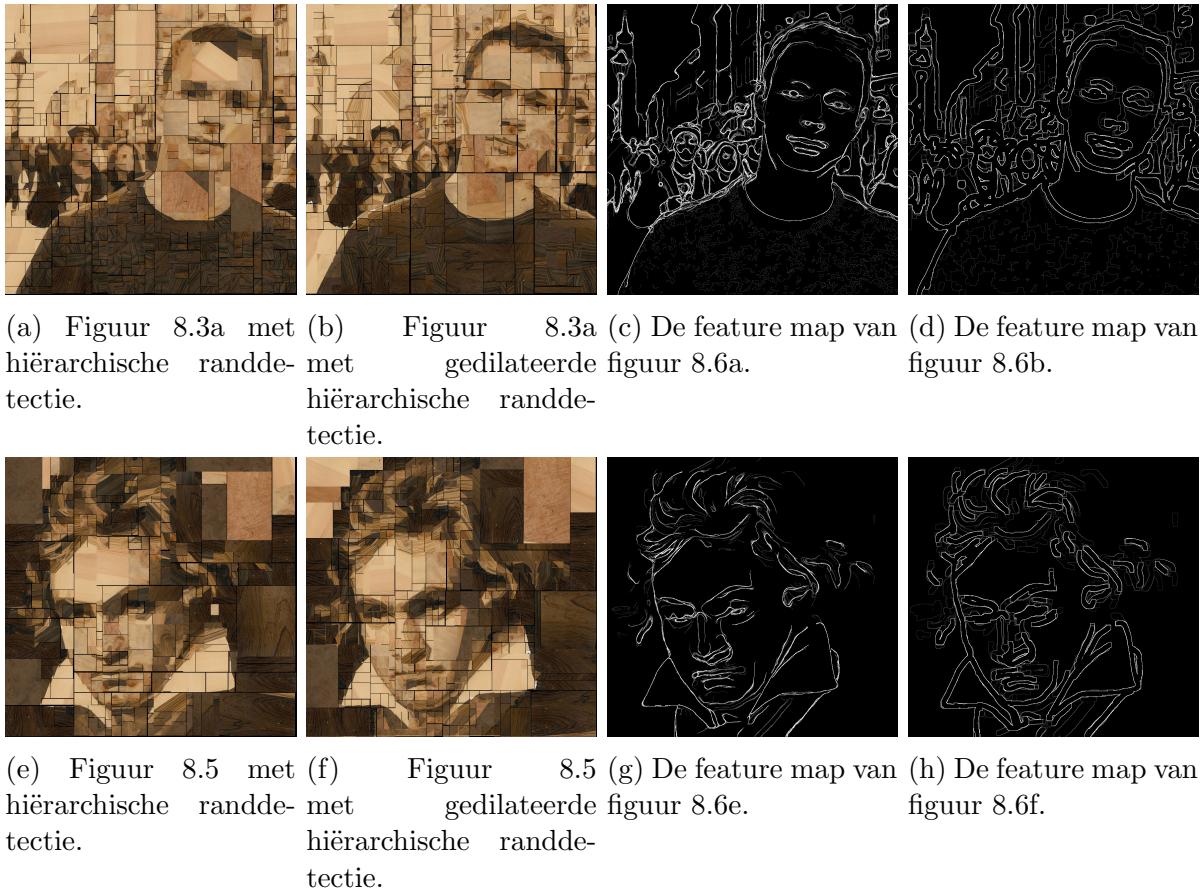
Figuur 8.4: Het verschil tussen het gebruiken van kleur en intensiteit als feature.

## 8.5 Invloed van adaptieve gridgeneratie via gedilateerde randdetectie

Het gebruik van gedilateerde randdetectie heeft een groot effect op de resulterende matching. Zoals beschreven in sectie 6.3.4 zorgt de dilatatie er voor dat kleine details niet wegwerkt worden door splitsingsvlakken recht op randen te plaatsen. In de plaats hiervan worden splitsingsvlakken iets boven of onder randen geplaatst, om zo een betere matching te krijgen. Figuur 8.6b en figuur 8.6f tonen dit aan. De details in de gezichten en het haar zijn hier veel beter weergegeven dan in figuren 8.6a en 8.6e. De lichter gekleurde patches in de bovenhoeken van figuren 8.6f en 8.6e zijn foutief gematcht en worden besproken in sectie 8.8. Dankzij deze resultaten kan geconcludeerd worden dat gedilateerde randdetectie betere resultaten biedt.



Figuur 8.5: Een afbeelding van Beethoven, genomen uit (Iseringhausen et al., 2020).



Figuur 8.6: Een vergelijking van hiërarchische randdetectie en gedilateerde randdetectie.

## 8.6 Verschillende source materialen

Het gebruiken van andere source materialen zorgt er voor dat de resulterende afbeelding een verschillende stijl krijgt. Deze sectie test een aantal source materialen uit die verschillen van het standaard materiaal in figuur 8.1. Een vereiste voor het source materiaal is dat het veel detail bevat. Veel schilderwerken in de impressionistische stijl voldoen hier aan. De aanwezigheid van stroken in het schilderij zorgt duidelijk voor goede resultaten, net zoals de nerven in het hout van het standaard materiaal. De resultaten zijn zichtbaar in figuur 8.7.

## 8.7 Invloed van aantal rotaties

Het aantal geroteerde source afbeeldingen  $n_r$  heeft minder invloed op het resultaat dan verwacht. Zelfs met  $n_r = 3$  rotaties kunnen zeer mooie resultaten behaald worden, zoals weergegeven in figuur 8.8. Het aantal rotaties wordt meestal ingesteld op een oneven getal, op die manier voorkomen we dat twee geroteerde afbeeldingen symmetrieën zijn van elkaar. Dit effect zie je duidelijk in figuur 8.8b waar het aantal rotaties er voor zorgt dat de randen in  $I_S$  enkel horizontaal of verticaal staan.

De kwaliteit van de resulterende afbeelding neemt toe met het aantal rotaties, maar



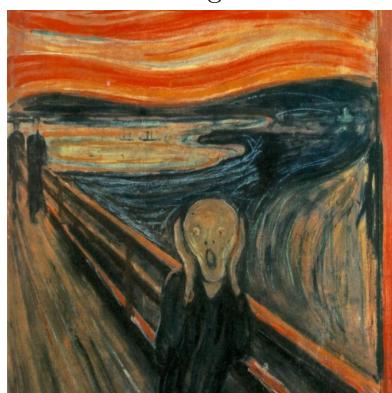
(a)  $I_S$ , De sterrennacht door Vincent van Gogh.



(b)  $I_T$ , De grote golf van Kanagawa door Hokusai



(c)  $I_R$



(d)  $I_S$ , De Schreeuw door Edvard Munch



(e)  $I_T$ , De sterrennacht door Vincent van Gogh.



(f)  $I_R$



(g)  $I_S$ , Bridge over a Pond of Water Lilies door Claude Monet.

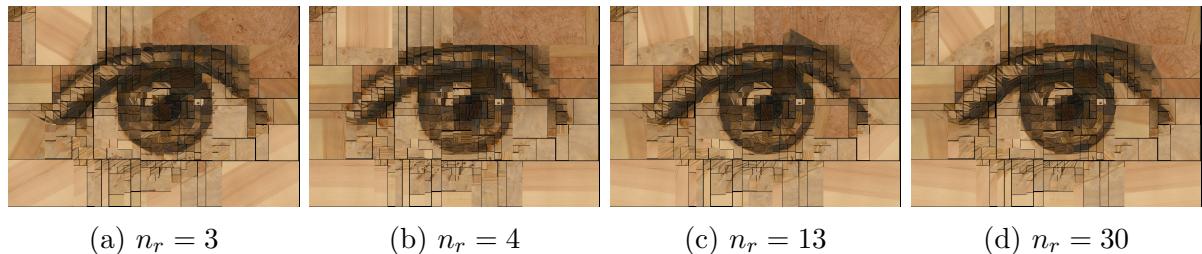


(h)  $I_T$ , San Giorgio Maggiore door Claude Monet.



(i)  $I_R$

Figuur 8.7: Resultaten met verschillende source materialen.



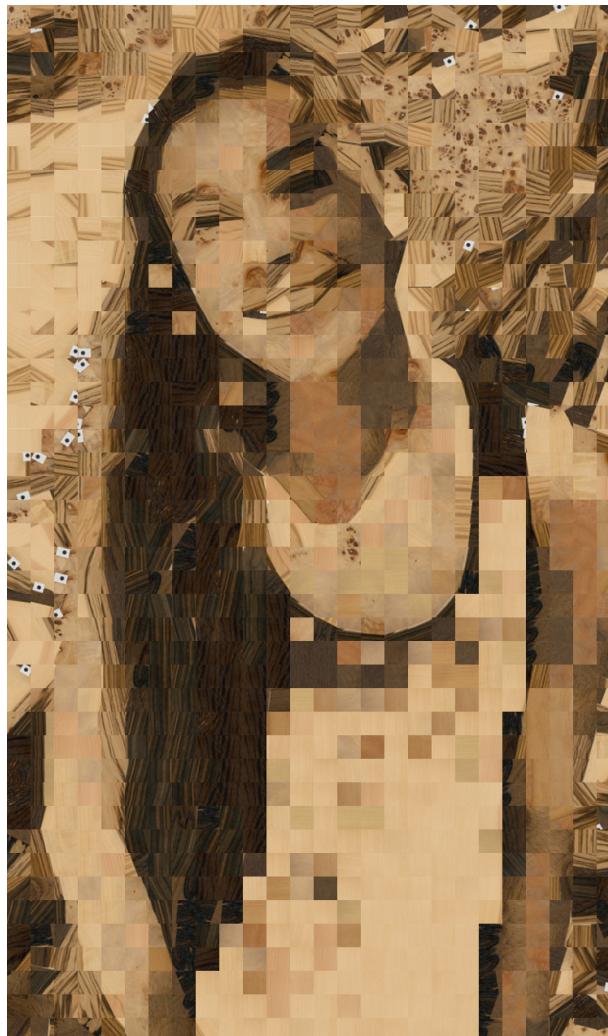
Figuur 8.8: Een aantal resultaten met een verschillend aantal rotaties,  $I_T$  is weergegeven in figuur 6.11a en is afkomstig van (Iseringhausen et al., 2020).

niet sterk. Het verschil tussen  $n_r = 13$  en  $n_r = 30$  is zichtbaar maar niet groot. De extra tijd die nodig is voor  $n_r = 30$ , is niet in proportie met de toename in kwaliteit.

## 8.8 Beperkingen

Het algoritme heeft ook een aantal beperkingen waar toekomstig werk voor verbetering kan zorgen.

- **Donkere regio's worden gematcht met lichte patches.**  
Zoals zichtbaar in figuur 8.6e en figuur 8.6f heeft het algoritme moeilijkheden met het matchen van volledig zwarte patches in de target afbeelding. Een extreem geval hiervan is weergeven in figuur 8.9. Dit probleem zou theoretisch gezien niet mogen voorkomen, en we vermoeden dat het dus een probleem in de implementatie is. Door tijdsgebrek is deze fout niet opgelost geraakt.
- **Vergelijking met resultaten door (Iseringhausen et al., 2020).**  
Een beperking van onze methode is dat de matching nooit dezelfde kwaliteit als die van Iseringhausen et al. (2020) kan hebben. Dit komt doordat patches in deze thesis enkel rechthoekig mogen zijn. Daarnaast is het belangrijk om het grid niet te fijn onder te verdelen, omdat de indruk van een Mondriaan verdeling dan verloren gaat. Iseringhausen et al. (2020) maakt gebruik van Bézier curves om de vorm van de patches aan te passen aan de target afbeelding en kunnen zo meer detail verwerken in hun grid.



Figuur 8.9: Een matching waar zwarte patches foutief gematcht zijn.

# Hoofdstuk 9

## Conclusie

1

---

<sup>1</sup>**TODO: SCHRIJF CONCLUSIE**

# Referenties

- Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698.
- Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Hertzmann, A. (2003). A survey of stroke-based rendering. *IEEE Computer Graphics and Applications*, 23(4):70–81.
- Iseringhausen, J., Weinmann, M., Huang, W., and Hullin, M. B. (2020). Computational parquetry: Fabricated style transfer with wood pixels. *ACM Trans. Graph.*, 39(2).
- Kim, S. Y., Maciejewski, R., Isenberg, T., Andrews, W. M., Chen, W., Sousa, M. C., and Ebert, D. S. (2009). Stippling by example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR ’09, page 41–50, New York, NY, USA. Association for Computing Machinery.
- Kyprianidis, J. E., Collomosse, J., Wang, T., and Isenberg, T. (2013). State of the ‘art’: A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):866–885.
- Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21:224–270.
- Martín, D., Arroyo, G., Luzon, M., and Isenberg, T. (2010). Example-based stippling using a scale-dependent grayscale process.
- Mcmanus, I. and Gesiak, P. (2014). Experimenting with mondrian: Comparing the method of production with the method of choice.
- Montabone, S. and Soto, A. (2010). Human detection using a mobile platform and novel features derived from a visual saliency mechanism. *Image and Vision Computing*, 28:391–402.
- OpenCV (2022a). Opencv. <https://opencv.org/>. Accessed: 2022-08-02.
- OpenCV (2022b). Template match modes. [https://docs.opencv.org/3.4/df/dfb/group\\_\\_imgproc\\_\\_object.html#](https://docs.opencv.org/3.4/df/dfb/group__imgproc__object.html#). Accessed: 2022-07-16.

- Orchard, J. and Kaplan, C. S. (2008). Cut-out image mosaics. In *Proceedings of the 6th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '08, page 79–87, New York, NY, USA. Association for Computing Machinery.
- Sobel, I. (2014). An isotropic 3x3 image gradient operator. *Presentation at Stanford A.I. Project 1968*.
- Stevanov, J. and Zanker, J. (2017). Exploring mondrian compositions in three-dimensional space. *Leonardo*, 53:1–11.
- Wikipedia (2020). Piet mondriaan. [https://nl.m.wikipedia.org/wiki/Piet\\_Mondriaan](https://nl.m.wikipedia.org/wiki/Piet_Mondriaan). Accessed: 2022-07-12.
- Wikipedia (2022a). Betegeling. <https://nl.m.wikipedia.org/wiki/Betegeling>. Accessed: 2022-07-13.
- Wikipedia (2022b). Box blur. [https://en.wikipedia.org/wiki/Box\\_blur](https://en.wikipedia.org/wiki/Box_blur). Accessed: 2022-08-03.
- Wikipedia (2022c). Lijst van werken van van piet mondriaan. [https://nl.m.wikipedia.org/wiki/Lijst\\_van\\_werken\\_van\\_Piet\\_Mondriaan](https://nl.m.wikipedia.org/wiki/Lijst_van_werken_van_Piet_Mondriaan). Accessed: 2022-07-12.
- Zhang, Q., Shen, X., Xu, L., and Jia, J. (2014). Rolling guidance filter. In *ECCV*.

**AFDELING**  
Straat nr bus 0000  
3000 LEUVEN, BELGIE  
tel. + 32 16 00 00 00  
fax + 32 16 00 00 00  
[www.kuleuven.be](http://www.kuleuven.be)

