# Implementatieplan - ImageShell en Intensity

Nicky van Steensel van der Aa 20-3-18

## Disclaimer

Na een tijdje zoeken kwam ik op zeven verschillende methoden uit, dit zijn er 4 meer dan vorig jaar, omdat de uiteindelijk gekozen implementatie het zelfde is als vorig jaar heb ik echter het programmeerwerk opnieuw gedaan. Ik heb het implementatieplan kritisch bekeken en waar nodig aangepast, het leek mij echter niet nodig om een volledig nieuw plan te schrijven. Mocht dit toch een vereiste zijn dan hoor ik dat graag.

## Doel

Het maken van een ImageShell voor RGB & Intensity images en het schrijven van code voor de conversie van RGB naar Intensity.

## Methoden

Ik heb 7 methodes gekozen om nader te bekijken voor het omzetten van RGB naar grijswaarde/intensity  
Bron: http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/

### *Methode 1*

De Averaging formule, deze formule is de simpelste van de drie door ons gevonden methoden, bij deze methode krijgt elke pixel als grijswaarde de som van alle drie de kleur kanalen gedeeld door 3, dus vrij letterlijk ‘het gemiddelde’ van alle kleuren omgezet naar zwart-wit.

### *Methode 2*

De Luma / Luminance formule, deze formule maakt gebruik van een vaste factor voor elk kleur kanaal om RGB om te zetten naar grijswaardes.   
elke pixel krijgt als grijswaarde een som van de volgende waardes, deze wegingen zijn een veelgebruikte verhouding, maar er zijn toch veel verschillende waardes in gebruik.  
0.3 keer de waarde van het rood kanaal.  
0.59 keer de waarde van het groen kanaal.  
0.11 keer de waarde van het blauw kanaal.

### *Methode 3*

Desaturatie, deze methode werkt theoretisch door de RGB waarde om te zetten in een HSL waarde, en de saturatie naar nul te brengen. Wat je dan overhoud zijn de H en L van HSL (Hue, saturation, lightness) waarbij hue de originele kleur voorstelt en lightness de “witheid”, dit lijkt moeilijk te implementeren maar is gelukkig wiskundig makkelijk te doen door de afgeleide formule te gebruiken (grijs = (max(r,g,b)+min(r,g,b))/2)

### *Methode 4*

De-compositie, deze methode is simpel, maar niet al te elegant, hij werkt door simpelweg de hoogste (Maximum Decomposition) of de laagste (Minimum Decomposition) kleurwaarde te nemen   
Voorbeeld: RGB(0,150,255) word bij max decomposition 255, en bij min decomposition 0.

### *Methode 5*

Single color channel is de volgende en werkt door een kanaal te kiezen, en deze waardes letterlijk over te nemen voor de grijswaarde bijvoorbeeld RGB(5,10,20) > Intensity(10), aangezien er in gezichten nogal veel informatie in alle kanalen tegelijk zit is deze methode niet geprefereerd.

### *Methode 6*

ConversionFactor = 255 / (NumberOfShades - 1)

AverageValue = (Red + Green + Blue) / 3

Gray = Integer((AverageValue / ConversionFactor) + 0.5) \* ConversionFactor

Dit de laatste formule die ik hier vermeld, ik vond hem noemenswaardig omdat de berekening ruimte geeft voor het invullen van het aantal resulterende grijskleuren, echter is dit voor ons project niet interessant omdat we hierdoor edges gaan krijgen die er niet zijn, wat navolgende stappen hindert.

## Keuze

Ik heb alle verschillende formules ingevuld in de code, en hoewel sommige resultaten best mooi waren, is de enige die ik goed bruikbaar acht voor computer vision methode 2.

## Implementatie

De regel code hier onder bevat het gebruikte algoritme van de Luma formule/methode. Zoals je kunt zien worden alle drie de kleurkanalen vermenigvuldigd met een eigen factor. De drie waardes die daar uit komen worden bij elkaar opgeteld en opgeslagen als de nieuwe grijswaarde in ‘pixel’

pixel = ((0.3f \* rgb.r) + (0.59f \* rgb.g) + (0.11f \* rgb.b));

## Evaluatie

Het gemaakte algoritme zal getest worden op snelheid, . De resultaten van deze test zullen worden verwerkt in de meetrapporten.