Implementatieplan imageshell & intensity

*19 februari 2020*

Door Tobias van den Hoogen en Jippe Heijnen.

Inhoud

[Doel 3](#_Toc33010743)

[Methoden 4](#_Toc33010744)

[Keuze 4](#_Toc33010745)

[Implementatie 4](#_Toc33010746)

[Evaluatie 4](#_Toc33010747)

# Doel

Ons doel voor dit onderdeel is het converteren van een RGB foto naar een Grey-scale zodat de intensity van de foto beter afgemeten kan worden. Ook kan het worden gebruikt voor edge detection omdat hierdoor de vormen en kanten beter te zien zijn. Dit zal inhouden dat we twee klassen moeten implementeren. Deze klassen zijn RGBImage en IntensityImage. Ook moeten we een functie implementeren waarbij RGB waardes worden converteert naar Intensity waardes. Bij het maken van de implementatie zullen we leren hoe het converteren van een plaatje werkt en wat het inhoudt. Daarnaast gaan we onze klassen ook moeten testen door middel van een GUI.

# 

# Methoden

Er zijn drie verschillende manieren om een RGB waarde te converteren naar een Intensity waarde:

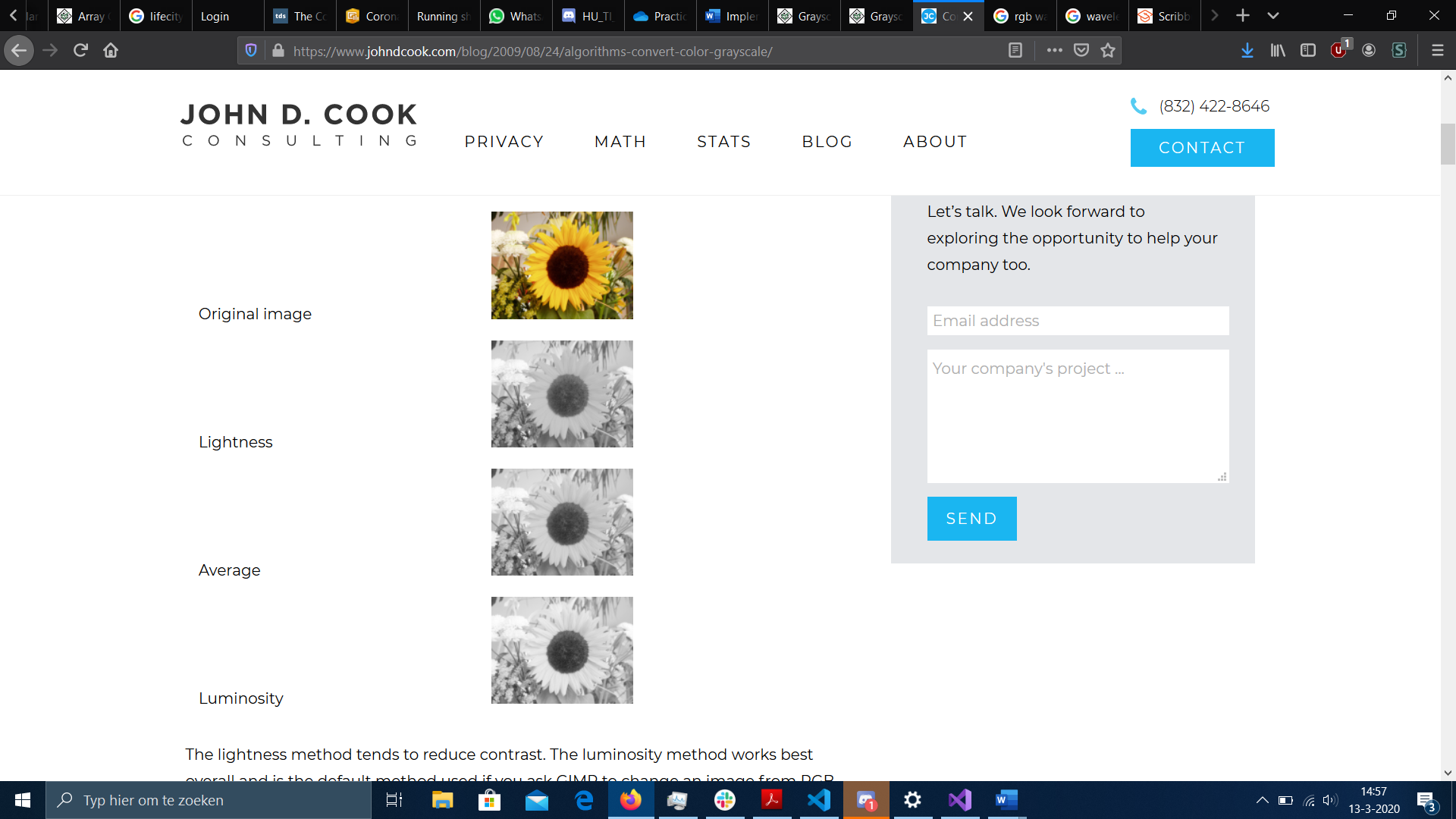
Ten eerste is er de lichtheid methode. Hierbij wordt er van de RGB waarde de maximum en minimum gekozen en wordt de som daarvan door twee gedeeld. Hierbij geldt de formule: *Intensity = (max(R, G, B) + min(R, G, B)) / 2.* (Cook, 2009)

Ten tweede is er de gemiddelde methode. Dit is de meest simpele methode waarbij de RGB waarde door drie wordt gedeeld. Hierbij geldt de formule: *Intensity = (R, G, B) / 3.* Een nadeel van deze methode is volgens (Tutorialpoint, z.d) dat er geen rekening wordt gehouden met de verschillende golflengtes van de kleurenspectrum. Hierdoor kan de helderheid van de afbeelding te licht/donker worden.

Als laatste is er ook de helderheid methode. Dit is een meer ingewikkelde versie van de gemiddelde methode. Het verschil vergeleken met de gemiddelde methode is dat bij deze methode rekening wordt gehouden met de menselijke perceptie. Omdat een mens volgens het kleurspectrum gevoeliger zijn voor groen dan voor rood en blauw heeft het ook een zwaardere weging . Hierbij geldt de formule: *Intensity = (0.21R + 0.72G + 0.07B).* (Cook, 2009)

Bij de afbeelding hieronder kunt u de verschillende uitkomsten van de formules zien:

Vergelijking 1: De verschillende uitkomsten van elke formule (Cook, 2009).



# Keuze

We kiezen voor de helderheidsmethode omdat het naar onze mening de meest accurate methode is voor de conversie. Ten eerste wordt het volgens beide bronnen aangeraden aangezien het rekening houdt met de menselijke perceptie van kleuren. Dit zorgt ervoor dat de afbeelding niet te helder/donker wordt en dat er details ook te zien zijn. Ook zijn wij ervan overtuigt dat volgens de afbeelding hierboven de helderheidsmethode het meeste detail vertoond.

# Implementatie

Eerst worden alle functies van de RGBstudent en Intensitystudent geïmplementeerd. Hierbij verwerken wij de todo’s die vermeld staan in de klassen zodat beide klassen gebruikt kunnen worden voor de conversie.

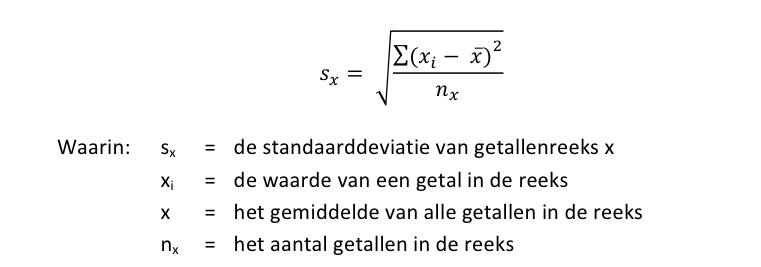
In de RGB klasse wordt dan de functie RGBtoIntensity gecreëerd. Deze functie zal geimplementeerd worden bij de file *DefaultPreProcessing.cpp*. In de functie zal er eerst per pixel de getPixel() functie worden aangeroepen. Daarna wordt de RGB waarde converteert naar Intensity met de helderheidsformule. Uiteindelijk wordt er de intensity klasse opgeroepen en wordt er bij die klasse de setPixel() functie per pixel opgeroepen om de geconverteerde waarde in te zetten. Uiteindelijk als alle pixelwaardes geconverteerd zijn zal er een Intensitystudent object worden gereturnd.

# Evaluatie

We zullen onderzoek doen naar de functionaliteit en efficiëntie van de helderheidsmethode.

Eerst zullen we een onderzoek doen naar de functionaliteit van de methode. Hierbij zullen we een dataset van 25 afbeeldingen maken. Hierbij wordt er het accuraatheidspercentage en de standaardafwijking per afbeelding getoond. Ook wordt er bijgenoteerd of het resultaat een false positive of negative is zodat er kan worden gekeken naar de betrouwbaarheid van de methode.

Daarna zullen we een onderzoek schrijven naar de efficiëntie van de methode waarbij we kijken naar de snelheid en memorygebruik van de methode vergeleken met de originele methode die in de testcode is verwerkt. Hierbij zullen we ook weer van 25 afbeeldingen een dataset maken. Met behulp van de tools en library’s die Visual studio biedt kunnen we per runtime proces van een afbeelding de memory-usage en snelheid van het proces meten. In de datasets wordt er de memory usage/snelheid percentage getoond met de standaardafwijking ervan.

[[1]](#footnote-2)

# Bronvermelding

Cook , J. D. (2009, 24 augustus). *Three algorithms for converting color to grayscale*. Geraadpleegd op 5 maart 2020, van <https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/>

*Grayscale to RGB Conversion*. (z.d.). Geraadpleegd op 10 maart 2020, van <https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm>

van der Zee, F. (2017). *Standaarddeviatie of spreiding*. Geraadpleegd op 13 maart 2020, van <https://hulpbijonderzoek.nl/online-woordenboek/standaarddeviatie/>

1. van der Zee, F. (2017). Deze afbeelding toont aan hoe de standaarddeviatie berekent moet worden. [↑](#footnote-ref-2)