Eindrapport Intensity to RGB

*27 maart 2020*

Door Tobias van den Hoogen en Jippe Heijnen

Inhoud

[Inleiding 3](#_Toc36224702)

[Resultaten 4](#_Toc36224703)

[Conclusie 5](#_Toc36224704)

[Evaluatie 6](#_Toc36224705)

[Functionaliteit 6](#_Toc36224706)

[Snelheid en memorygebruik 6](#_Toc36224707)

[Samenvattend 7](#_Toc36224708)

# Inleiding

Tijdens deze cursus hebben wij de opdracht gekregen om een implementatie te schrijven voor het converteren van kleurenfoto’s naar grijswaarden. Hierbij hebben wij onderzocht of onze implementatie significant beter presteerde vergeleken met de default implementatie. Dit hebben wij op twee gebieden gedaan, namelijk op gebied van functionaliteit én op gebied van snelheid & memorygebruik.

Bij het onderzoek van functionaliteit hebben we o.a. gecontroleerd op false-positives en false-negatives. Bij het onderzoek van snelheid en memorygebruik hebben we per foto gekeken hoe snel het script uitvoerde en hoe efficiënt het omging met geheugen.

# Resultaten

Tabel : Onderzoeksresultaten van de functionaliteitsonderzoek

|  |  |  |
| --- | --- | --- |
| **Formule** | **Default** | **Onze implementatie** |
| Gemiddelde | 0,5714286 | 0,607142857 |
| Standaarddeviatie | 0,5039526 | 0,497347461 |
| Accuraatheidspercentage | 57% | 61% |

Hierboven zien we een tabel waarin de herkenning van foto’s staat aangegeven. Hoe dichter het kommagetal bij de 1 is, des te meer gezichten herkend worden. We kunnen uit de 61% opmaken dat onze implementatie nét iets meer gezichten herkent dan de standaard implementatie. Echter is dit verschil niet groot genoeg om ook daadwerkelijk hieruit te concluderen dat onze implementatie op dit gebied beter is, aangezien de standaarddeviatie niet significant verschilt van de default implementatie.

Tabel : Onderzoeksresultaten van de memorygebruik

|  |  |  |
| --- | --- | --- |
| **Formule** | **Default** | **Onze implementatie** |
| Gemiddelde | 6,035714mb | 21,82142857mb |
| Standaarddeviatie | 0,188982mb | 2,93198923mb |

Uit bovenstaande tabel kunnen we opmaken dat de default implementatie een minder intensief gebruik van geheugen heeft dan onze eigen implementatie. Daarnaast kunnen we ook zien dat onze code een hogere standaardafwijking heeft. Dit geeft o.a. aan dat onze code minder stabiel is dan de default qua verbruik.

Tabel : Onderzoeksresultaten van de snelheidsonderzoek

|  |  |  |
| --- | --- | --- |
| **Formule** | **Default** | **Onze implementatie** |
| Gemiddelde | 2,928571 | 4,321429 |
| Standaarddeviatie | 0,377964 | 1,156418 |

Aan deze bovenstaande tabel kunnen we zien dat onze implementatie trager is dan de default implementatie. Ook is onze code instabieler bij het uitvoeren omdat de standaardafwijking groter is dan bij de default implementatie.

# Conclusie

We kunnen uit het functionaliteitsonderzoek concluderen dat onze implementatie op gebied van functionaliteit nauwelijks verschilt. Op gebied van snelheid en geheugengebruik hebben we gemerkt dat onze implementatie zelfs slechter is dan de default. We kunnen als eindconclusie dus zeggen dat onze implementatie helaas niet significant beter is dan de default implementatie. De mogelijke oorzaken van onze resultaten zullen we bespreken bij de evaluatie.

# Evaluatie

## Functionaliteit

Na de resultaten van het functionaliteitsonderzoek besproken te hebben, zijn we tot een mogelijke reden gekomen waarom onze implementatie iets, maar niet significant beter is, vergeleken met de default implementatie. Waarschijnlijk is onze implementatie iets beter omdat wij in onze formule meer bezighouden met de kleurwaarden vergeleken met de default implementatie. Omdat de default implementatiecode onleesbaar is weten we zelf ook niet zeker voor welke formule is gekozen, maar waarschijnlijk heeft de default implementatie voor een simpelere formule gekozen voor het converteren naar de RGB waarden naar Intensity waarden.

De reden waarom onze implementatie niet significant beter is, is omdat de grayscale output van onze implementatie vergeleken met die van de default implementatie ongeveer hetzelfde is. En daarom zullen de verdere processen (edge detection, scaling enz.) ongeveer dezelfde uitkomsten hebben.

Wat we de volgende keer zouden kunnen doen is onderzoek doen naar hoe de kleurwaardes het best geconverteerd kunnen worden naar greyscale.

## Snelheid en memorygebruik

We waren erg verrast bij de uitkomst van het snelheids en memorygebruik onderzoek. Na de resultaat en conclusie onder elkaar te bespreken kwamen op een mogelijke reden uit waarom onze implementatie minder snel en meer geheugen gebruikte vergeleken met de default implementatie.

Wij denken dat het grotere memorygebruik en de lagere snelheid van onze implementatie komt omdat wij een std::pair gebruiken als key van een std::map, voor de opslag van de pixels. Wij hadden ervoor gekozen om een std::pair als een key te gebruiken omdat het hierdoor overzichtelijker werd voor het opslaan voor de pixels. Wij denken dat het gebruiken van een std::pair als key ervoor zorgt dat er meer memorygebruik en dat het programma met tragere snelheid runt. Dit zou komen doordat het een object is en als een object honderden keren wordt opgeslagen zal dat voor meer memorygebruik zorgen.

W denken dat het memorygebruik bij de default implementatie zo laag is,  omdat zij waarschijnlijk een map gebruikten met een integer als key. Dit zorgt wél voor minder overzichtelijkheid maar óók voor minder memorygebruik en hogere snelheid, maar dit kunnen wij niet bevestigen aangezien de default code onleesbaar is gemaakt.

Wat we de volgende keer zouden kunnen doen is onderzoek doen naar het overzichtelijk opslaan van pixels met zo min mogelijk geheugengebruik.

## Samenvattend

Om kort samen te vatten is het handig als we voor de volgende keer een onderzoek doen over hoe kleurwaardes het best geconverteerd kunnen worden en is het ook handig om onderzoek te doen naar het overzichtelijk opslaan van pixels met zo min mogelijk geheugengebruik.