**ImageShell & intensity**

# Namen en datum

Wilco Matthijssen

Daan Zimmerman van Woesik

21/02/2020

# Doel

Voor de faceRecognitionGUI van de cursus Vision willen wij onze eigen imageShell maken. Een imageshell is iets waar fotos kunnen worden opgeslagen, aangepast en uitgelezen worden. In dit geval wordt dat gedaan met RGB(kleur fotos) en intensity (zwart wit) ook moet er ook een functie zijn om de RGB fotos om te zetten naar intensity. De reden waarom wij een vernieuwde imageshell en conversie van RGB naar intensity gaan maken is omdat we denken dat het sneller kan dan de huidige oplossing.

Aannames

Er zijn meerdere manieren om fotos op te slaan en om te zetten van RGB naar intensity.

Om de volgende keuzes wat makkelijker te maken zijn er de volgende aannames:

* Een foto moet tijdens runtime van grote kunnen veranderen ook nadat het object wordt aangemaakt.
* Intensity is het gemiddelde van de RGB(rood, groen en blauw) waardes.
* Intensity is een 8 bit waarde die van 0 tot 255 gaat en van donker naar licht is.
* Foto’s worden per pixel opgeslagen

# Methodes

### Opslaan

Om pixels op te slaan van foto’s en het aantal pixels aan te kunnen passen is een vector en new c-array de meest voor de hand liggende opties.

Een vector heeft als voordeel dat het veel gebruikt is en makkelijk te gebruiken is. Wat wel een nadeel is dat het even snel is of trager vergeleken met een C-array

Het opslaan van een tweedimensionaal iets kan opgeslagen worden door 1 lange series aan waardes of een twee dimensionale series aan rijen. Het opslaan in 1 lange series is makkelijker in aanmaken en aanpassen. Een voordeel van een twee dimensionaal opslag is dat het een stuk gebruiksvriendelijker is om uit te lezen.

Het opslaan van waardes is een optie om de pointer naar de waardes opslaan maar dan moet je het voordeel hebben dat het item dat je opslaat groter is dan de pointer (32bit / 64bit). Wat ook een voorwaarde moet zijn om efficiënter te zijn met opslag is dat de waardes die opslaat vaak dezelfde zijn.

### Omzetten van RGB naar intensity

#### Omzetten naar intensity

Voor het omzetten van RGB zijn er verschillende methodes. Deze methodes zijn:

* Average
* Lightness
* Luminosity

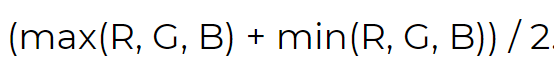
##### Average

Dit is een simpele en snelle methode om een RGB waarde om te zetten naar een intensity waarde. Echter zijn de waardes die uit deze berekening komen geen goede representatie voor wat je met je eigen ogen ziet dus wordt dit erg afgeraden.



##### Lightness

Deze methode zorgt ervoor dat de hoge waardes en lage waardes naar voren komen in de foto. Dit zorgt ervoor dat de foto zowaar wordt gesmootht. Hierdoor gaat veel van de contrast verloren en lijkt de foto wat meer overbelicht. Daarom wordt dit voor onze applicatie ook afgeraden, echter kan dit voor sommige toepassingen voordelen hebben.



##### Luminosity

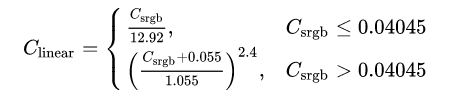
De methode luminosity is een geavanceerdere versie van de methode Average. Het is ook een gemiddelde van waarden maar er wordt meer nadruk gelegd op de menselijke perceptie. We zijn bijvoorbeeld gevoeliger voor de kleur groen dus die wordt het zwaarst gewogen in de formule.



#### Gamma correction

Onze ogen zien de wereld om ons heen heel anders dan een camera. Daardoor zijn fotos die worden gemaakt niet helemaal zoals jij dat ziet. Dit kan worden aangepast met een gamma correction en daarvoor zijn er 2 verschillende manieren om het te berekenen.

Een van de manieren, dit is ook wel de beste manier, is gedaan met deze formule. Deze formule zorgt voor een bijna perfecte nabootsing van het menselijk oog, hierbij komt echter ook een grotere computing power waardoor de correction veel langer kan duren.



Om de computing power te verkleinen is er een linear approximation gemaakt waardoor een deel van de accuracy verloren gaat maar wel een stuk sneller. 

#### Omzet process versnellen

Tijdens het omzetten van een RGB / intensity image naar de andere format kunnen we gebruik maken van memoisation. Echter zal dit niet veel sneller gaan lopen omdat het zoeken door een tabel van deze waardes dan langer gaat duren dan het uiteindelijk omzetten van RGB naar intensity. Toch kan memoisation sneller zijn als er een achtergrond wordt gebruikt met dezelfde kleur. Daarom kan je misschien een kleine buffer gebruiken die alleen de vorige pixel onthoud zodat je minder tijd kwijt bent doordat je aan het zoeken bent in een array.

# 

# Keuze

Als keuze is gekozen voor een vector over een c-array vanwege de ingebouwde veiligheid van een vector en omdat het [hier](https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md#Rsl-arrays.) wordt afgeraden. Het opslaan is ook gekozen voor een 1d array omdat het makkelijker is om van grootte aanpassen en [de kans dat het iets sneller is](https://stackoverflow.com/questions/17259877/1d-or-2d-array-whats-faster).

Voor de afweging pointer of waarde is er gekozen voor de waarde zelf omdat die kleiner of gelijk is aan een 32 bits pointer.

Voor de conversie van RGB naar intensity is gekozen voor de snelle gamma correction, dit wegens het feit dat dit sneller is dan de uitgebreide formule voor maar een beetje verlies in accuraatheid. Snelheid in deze context vonden wij belangrijker dus vandaar onze keuze hiervoor. Daarna gebruiken wij een algoritme om de RGB waardes om te zetten naar intensity en hiervoor is gekozen het algoritme luminosity. Dit is gekozen omdat het over het algemeen de beste resultaten weergeeft. Ten slotte kiezen we ook nog voor een memoisation. Dit zodat als er een foto wordt omgezet er veel performance voordelen behaald kan worden als de achtergrond dezelfde kleur heeft.

# Implementatie

Voor het implementeren van de imageshell zullen de bestaande klassen genaamd RGBImageStudent en IntensityImageStudent worden aangepast zodat het overeenkomt met de keuzes die er gemaakt zijn. Het omzetten van een RGB image naar intensity image zal de bestaande functie stepToIntensityImage in de klasse StudentPreprocessing voor worden aangepast met de methode die gekozen is.

# Evaluatie

Om te kunnen testen hoe snel een methode is hebben we een timer klasse van de volgende video gebruikt:[BENCHMARKING in C++ (how to measure performance)](https://www.youtube.com/watch?v=YG4jexlSAjc.).

Hoe en wat er gaat getest worden is overkopieren van een foto. Dit wordt gedaan door een foto in te laden in een RGBImage(Student) en dan te timen hoe snel het is om die klasse als argument te gebruiken om een nieuw object aan te maken duizend keer. Het testen van de conversie van en RGBImage(Student) naar een IntensityImage(Student) zal als volgt worden gedaan. Er zal een foto worden ingeladen in de RGBImage(Student) klasse. De klasse zal dan 1000 keer in functie stepToIntensityImage worden gebruikt en getimed hoe snel dat is.

# Bronnenlijst

BENCHMARKING in C++ (how to measure performance)

<https://www.youtube.com/watch?v=YG4jexlSAjc>

SL.con.1: Prefer using STL array or vector instead of a C array

<https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md#Rsl-arrays>

Converting RGB to intensity

<https://www.kdnuggets.com/2019/12/convert-rgb-image-grayscale.html>

<https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/>

<https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm>

Gamma correction

<https://helpx.adobe.com/nl/after-effects/using/color-basics.html>

<https://www.cambridgeincolour.com/tutorials/gamma-correction.htm>