# Implementatieplan Grayscaling & Image Shell

Gemaakt door: Brandon Kroes en Maarten Wassenaar

Datum: 11 april 2019

# Doel

De bedoeling van deze implementatie is om het volgende mogelijk te maken binnen de HU Vision module:

* Grayscaling

Een manier om van kleur foto’s een conversie naar grijstinten mogelijk te maken.

* Een image shell

Een manier om binnen het HU-Vision pakket individuele pixels van een foto te bewerken.

# Methoden

## Grayscaling

Om grayscaling te verwerken zijn er een tal van manieren om het te doen. Zoals met vele dingen in de wetenschap zijn er discussies en is er eigenlijk geen enkele consensus om wat het *beste* is. Wij hebben gekeken naar de onderstaande algortimes.

ITU-R BT-601 (Rec. 601) en ITU-R BT-709 (Rec. 709)

Is een standaard dat is opgesteld voor standaard definitie TV’s door de Internationale Telecommunicatie-unie. Het standaard stamt uit 1982. De berekening is als volgt Rood \* 0.299 + Groen \* 0.587 + Blauw \* 0.114. De getallen die achter de waarde zijn worden ook wel de luma coëfficiënt genoemd [(zelfs de naamgeving brengt disucssie met zich mee).](http://poynton.com/PDFs/YUV_and_luminance_harmful.pdf)

In de nieuwe versie, Rec. 709 is de berekening aangepast naar Rood \* 0.2126 + Groen \* 0.7152 + Blauw \* 0.0722. Veel cameraregisseurs, editors en wetenschappers zijn het [niet eens](http://www.glennchan.info/articles/technical/hd-versus-sd-color-space/hd-versus-sd-color-space.htm) met de nieuwe luma coëfficiënt.

Het voordeel van een luma coefficient is dat het rekening houdt met wat mensen daadwerkelijk goed kunnen zien. Het nadeel kan echter zijn dat je fijnere operaties moet doen en afrondingen moet doen.

Beide algoritmes hebben wel een zwak punt als ze worden vergeleken met elkaar, ze zijn namelijk beide heel erg afhankelijk van de kwaliteit van het apparaat waarop het wordt bekeken. De gemiddelde TV voor een consument is erg kleur inaccuraat en hierdoor kan het verschil tussen beide [minuscuul](http://www.glennchan.info/articles/technical/rec709rec601/rec709rec601.html) worden voor de eindgebruiker.

Desaturatie

Desaturatie werkt doormiddel van het nemen van de maximale waarde, de minimale waarde en daar het gemiddelde van nemen. Desaturatie zorgt voor een egalere afbeelding doordat het verschil tussen de hoogste en laagste waarde wordt verkleind. Het nadeel wel is dat er steeds moet worden gezocht naar de minimale en maximale voordat de daadwerkelijke berekening wordt gedaan. De berekening ziet er als volgt uit: Grijs = (max (Rood, Groen, Blauw) + min(Rood, Groen, Blauw)) / 2.

Enkel kanaal

Een enkel kanaal gebruiken is de grayscale waarde direct koppelen aan een kanaal. Een enkel kanaal conversie vereist dus heel weinig computer kracht, echter is het risico dat de foto er heel anders uit ziet doordat het verschil tussen kleur kanalen groot was. Het klinkt als een hele rare manier van het doen van grayscaling, maar sommige fototoestellen gebruiken deze techniek om weinig calculaties te hoeven doen. De berekening is dus simpel Grijs = Rood | Groen | Blauw.

## Image shell

Voor de image shell hebben wij ervoor gekozen om niet een uitgebreid onderzoek te doen, omdat vergelijking binnen c++ containers al volmaakt beschikbaar zijn op het internet. We hebben het document [EECS 311: STL Containers](http://users.cs.northwestern.edu/~riesbeck/programming/c++/stl-summary.html) van de universiteit van North Western universiteit gepakt als referentie punt. Hieruit hebben wij de keuze voor een vector gemaakt doordat het een native container is binnen c++, de set en get tijden van orde 0(1) waren en het de functies had die we willen van een container. Het nadeel van het gebruiken van een vector is echter wel dat een apparaat gebruik moet kunnen maken van een heap om dynamische allocatie van gegevens te doen. Hierdoor is het niet meer mogelijk om het op embedded systemen te draaien.

# Keuze

## Grayscale

Aan de hand van onze meetrapporten raden wij [HIER] aan. We hebben de afweging gemaakt tussen dat wat het snelste is en wat het mooiste is en hier een goed balans in gevonden.

## Image shell

Zoals we al eerder hadden besproken, voor de image shell hebben wij gekozen voor een std::vector<> met daarin de pixel type (RGB of Intensity).

# Implementatie

De implementatie is zo gebouwd dat er enkel, zoals eerder besproken, een vector komt met daarin de pixel type afhankelijk van het type klas (RGB voor de RGBImageStudent en Intensity voor de IntensityImageStudent). De vector is te vinden in beide klasses onder de naam Pixels. De functie namen die al waren opgezet zijn ook gehanteerd. Het grayscaling algoritme is opgenomen in de *StudentPreProcessing.cpp*, zoals eerder besproken hebben we daarin het GIMP & Photoshop algoritme in verwerkt.

De implementatie van de image shell verwachten wij ongeveer 30 minuten mee bezig te zijn en de implementatie van de kleurconversie ongeveer 10 minuten. Beide situaties zijn echter geen rekening gehouden met het documenteren. We hebben geen documentatie eisen aangetroffen en kunnen daarom ook geen inschatting maken voor dat aspect.

# Evaluatie

De volgende twee experimenten zullen we doen en hebben we hier in hoofdlijnen omschreven. In het meetrapport worden deze nog verder toegelicht.

Snelheid

We zullen alle voorgestelde algoritmes testen op snelheid. Hiermee willen we in kaart brengen wat de snelheid is per algoritme en ze vergelijken met elkaar. De snelheid wordt gebaseerd op de hoeveelheid aan klokslagen (CPU) die de operaties erover doen. De klokslagen kunnen dus ook worden teruggerekend naar seconde met als doel het computer onafhankelijk te maken (Weliswaar chipset afhankelijk). Al met al verwachten hier een dag mee bezig te zijn door het herhalen en uitvoeren van een aantal duizend conversies.

Mooiheidfactor

We zullen een enquête doen onder studenten om te kijken welke uitkomst ze het mooist vinden. De enquête wordt verdeelt onder studenten waarin ze elk de keuze krijgen om te stemmen voor een bepaalde kleur conversie. Het opstellen van de enquête is niet zozeer een uitdaging, maar ervoor zorgen dat we een redelijke pool aan studenten krijgen vergt een aantal uur aan netwerken.

Aan de hand van deze twee waarden zullen wij ook onze aanbeveling doen.