IMPLEMENTATIEPLAN RGBIMAGESTUDENT & INTENSITYIMAGESTUDENT

NAMEN EN DATUM

Nicky van Steensel van der Aa

DOEL

Mijn doel is om de zelfgeschreven image klassen te gebruiken en in staat te zijn om ExternalDLL te draaien en alle overige stappen te laten verlopen via de meegeleverde implementaties.

METHODEN

Er zijn voor de implementatie weinig keuzes die gemaakt kunnen worden. Een van de variabelen die er is is de opslagmethode van de pixel structs. Dit kan het makkelijkst met één of twéé dimensionale, arrays of vectoren.

Voor de implementatie van RGBImageStudent zijn er weinig keuzes die gemaakt kunnen worden: de enige keuze is de methode waarmee de structs opgeslagen worden. Enkele opties zijn in een een- of tweedimensionaal array of in een een- of tweedimensionale vector.

Voor de implementatie van IntensityImageStudent moet ook gekozen worden op wat voor manier de kleuren omgezet worden in grijswaarden. Gevonden mogelijkheden hiervoor zijn o.a.:

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
```

De laatste formule die ik hier vermeld, vond ik interresant omdat de berekening ruimte geeft voor het invullen van het aantal resulterende grijskleuren, echter is dit voor het Vision project niet interessant omdat je hierdoor edges gaat krijgen die er niet zijn, wat navolgende stappen hindert.

KEUZE

Ik heb voor zowel RGBImageStudent als IntensityImageStudent het type gekozen waar mij het makkelijkst mee leek te werken; een eendimensionale vector.

Ik heb besloten de Luminance methode te gebruiken, dit omdat ik deze code grotendeels af heb van voorgaande jaren, maar ook omdat ik deze (toen al) de beste methode vind.

EVALUATIE

Voor de evaluatie kijk ik of mijn implementatie werkende en vergelijkbare resultaten geeft net als de implementatie van Arno.