Implementatie Plan *ImageShell & Intensity*

Intensity

Gemaakt door: Bob Thomas en Robbie Valkenburg

Technische informatica 2016 – 2017

Datum: 13 maart 2017

1.1. Doel

Het doel van dit project is een digitale afbeelding omzetten naar een intensity image.
Gebaseerd op de grijswaarden van het plaatje in plaats van de 3 kleur kanalen (RGB).
Dit zal het belast op het geheugen verminderen en de snelheid van calculaties gebaseerd op intensiteit verhogen.

1.2. Methoden

Wij hebben onderzoek gedaan naar verschillende algoritmes om een intensity image (afbeelding gebaseerd op grijstinten).
Via onze bron Kanan, C., & Cottrel;, G. W. (2012). Hebben we meerdere algoritmes ontdekt,
Al deze algoritmes worden per pixel toegepast.
Er zitten en paar interessante tussen die ons goeie resultaten kunnen geven er zitten ook een paar algoritmes tussen die zo simpel zijn dat ze het wel omzetten naar zwart of wit maar met te harde waardes. Een paar vormen van een algoritme wat te harde waren terug geeft zijn `value` en het `luster algoritme` deze twee algoritmes zijn makkelijk te programmeren maar niet erg accuraat.
Het 'Value' algoritme werkt als volgt hij kiest uit die 3 kleur kanalen (RGB) de max waarde en zet dat om in een grijs waarde (intensiteit).
Het `Luster` algoritme pakt in plaats van de max en min waarde van de kleur kanalen pakt die het gemiddelde van de 3 kanalen. Deze algoritmes zijn dus erg makkelijk om te implementeren maar waarschijnlijk niet accuraat genoeg. De volgende reeks algoritmes zijn een stuk accurater volgens onze bron Kanan, C., & Cottrel;, G. W. (2012). Het meest bekende algoritme genaamd `Intensity` neemt het gemiddelde van de 3 kanalen en dat word de intensiteit.

$$\boldsymbol{G}intensity = \frac{1}{3}(R + G + B)$$

Dit algoritme zal er al een stuk natuurlijker uitzien vergeleken `luster` of `value`.
Maar ik verwacht dat dit algoritme nog iets te intense resultaten zal leveren.
Gelukkig zijn we in de bron ook achter het algoritme `luminance` gekomen.
Dit algoritme wordt gebruikt in Opencv, Matlab, en Gimp de formule van dit algoritme is als volgt.

$$GLuminance = (0.3 * R + 0.59 * G + 0.11 * B)$$

We doen 1 van de RGB-kanalen keer een gespecifieerd gewicht gebaseerd op onderzoek van hoe wij als mens intensiteit in zien.
Aangezien Opencv dit algoritme ook gebruikt verwacht ik dat het framework wat we gebruiken voor onze testen en vergelijkingen dit algoritme ook toepast.
Het andere algoritme wat volgens de bron het beste uit de test komt is het algoritme genaamd `Gleam`. Dit algoritme werkt hetzelfde als het `Intensity` algoritme alleen hier zijn de rgb met gamma correctie bewerkt.
Uit onze tweede bron (Helland, n.d.) kwamen we de zelfde algoritmes tegen maar ook een hele simpele maar met goede resultaten namelijk het single color channel algoritme. Hierbij neem je uit de RGB kanalen een kanaal bijv. G en dat word gewoon de intensiteit waarden. Deze methode word vaak gebruikt in digitale camera's volgens de bron.

1.3. Keuze

Wij heben gekozen tussen3 methodes en die zijn Luminance, single color channel en Intensity.
Na dat ik de paper van Kanan, C., & Cottrel;, G. W. (2012) had gelezen kwam bij hun testen eruit dat Luminance de beste resultaten gaf. Zelf weten we er nog vrij weinig van af behalve de resultaten uit het artikel. Maar aangezien Matlab en GIMP het luminance algoritme gebruiken lijkt mij dit ook een sterke kandidaat en zeker omdat Matlab het gebruikt voor computer vision doeleinden.De methode de we gekozen hebben is Luminance en Intensity omdat we willen onderzoeken hoe snel het is en of de resultaten kunnen vergelijken tegen de implementatie van GIMP of Matlab wat ons wel een interessante test leek. Een het single color channel algoritme lijkt zo simpel dat het eigenlijk niet zou moeten werken dus die willen we er ook bij testen

1.4. Implementatie

Wij hebben er voor gekozen om te werken in Microsoft Visual studio 2015. Dit omdat het project grotendeels is ontwikkeld in Microsoft Visual studio 2015. Vanuit deze omgeving hebben wij de code geïmplementeerd. Het algoritme wordt geïmplementeerd in het bestand StudentPreProcessing.cpp. Hier zullen we over alle pixels gaan in het plaatje en het algoritme toepassen en daarbij een nieuwe Intensity Image genereren.

```
IntensityImage * StudentPreProcessing::stepToIntensityImage(const RGBImage &image) const {
    GrayscaleAlgorithm grayScaleAlgorithm;
    IntensityImageStudent *test = new IntensityImageStudent(image.getWidth(), image.getHeight());

    for (int i = 0; i < test->getWidth()*test->getHeight(); i++) {

        // Intensity  algoritm
        //test->setPixel(i, 0.3 * (image.getPixel(i).r + image.getPixel(i).g + image.getPixel(i).b));

        // Luminace algoritm
        //test->setPixel(i, ((float)image.getPixel(i).r*0.299) + ((float)image.getPixel(i).g * 0.587) + ((float)image.getPixel(i).b * 0.114));

        //value
        //test->setPixel(i, std::max({ image.getPixel(i).r, image.getPixel(i).g, image.getPixel(i).b}));

        //single-collor channel
        //test->setPixel(i, Intensity(image.getPixel(i).g));


    }
    return test;
}
```

1.5. Evaluatie

De metingen die we gaan doen bestaat uit.

- Snelheid van het proces vergeleken de bestaande implementatie
- Welke van 4 algoritmes de beste grijs waardes generen op basis van een enquête waar de deelnemer kan kiezen welk plaatje hij het mooiste vind

2. Bronnen

Kanan, C., & Cottrel;, G. W. (2012). *Color-to-Grayscale: Does the Method Matter in Image Recognition?*. Retrieved from http://tdlc.ucsd.edu/SV2013/Kanan_Cottrell_PLOS_Color_2012.pdf

(https://nl.mathworks.com/help/matlab/ref/rgb2gray.html)https://nl.mathworks.com/help/matlab/ref/rgb2gray.html

(https://www.gimp.org/)https://www.gimp.org/

Helland, T. (n.d.). Seven grayscale conversion algorithms (with pseudocode and VB6 source code). Retrieved from http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/