IMPLEMENTATIEPLAN

ImageShell & Intensity

Abstract

In dit verslag zal er besproken worden wat ons implementatieplan is voor de practicumopdracht: Image Shell & Intensity. Deze opdracht bestaat uit twee delen namelijk:
Het maken van een Image Shell voor RGB en voor Intensity images
Het maken van een code voor de conversie van RGB naar Intensity images

Christiaan van den Berg & Mike Hilhorst
1660475 & 1676029

Inhoud

Doel

Wij willen de huidige implementatie Image Shell en Intensity verbeteren, dit willen wij doen zonder het gebruik van de bibliotheek genaamd openCV.
Om dit practicum een succes te maken moeten wij twee meet rapporten maken. In deze meet rapporten moeten wij gaan meten en vastleggen wat het verschil is tussen de originele implementatie en onze implementatie. Deze implementaties mogen getest worden op onder andere:

- Snelheid
- Memory efficiency
- Robuustheid
- Volledigheid
- Extra functionaliteiten

Wij hebben gekozen voor Snelheid en Memory efficiency

Methoden

Er zijn meerder methode om greyscale te implementeren. Wij hebben gekeken naar zes verschillende technieken, namelijk:

Averaging

Deze techniek is een hele simpele manier om een afbeelding met kleur om te zetten in grijswaardes. De techniek is te beschrijven als volgt: Pakt de drie kleur kanalen van een afbeelding, tel deze bij elkaar op en deel deze resulterende waarde door drie. Hier een klein voorbeeld: pixel#1 = (R=124, G=200, B=75) -> new = (124+200+75)/3 = 399/3 -> Greyscale waarde is 133.

Voordeel: Simpel te implementeren
Nadeel: Geeft grijze afbeeldingen die niet goed overeenkomen met menselijke helderheidsperceptie

Luminance

Deze techniek maakt gebruik van het feit dat het menselijke oog meer gevoelig is voor groen, dan voor rood en blauw. De techniek is te beschrijven als volgt: Pakt de kleuren kanalen van een afbeelding, vermenigvuldig deze waardes een “magic number”, de resulterende waarde hiervan is je nieuwe greyscale waarde. Bijvoorbeeld: pixel#1 = (R=124, G=200, B=75) -> new = (124*0.299, 200*0.587 ,75*0.114) -> Greyscale waarde is ongeveer 163.

Voordeel: Maakt correctie waardoor de afbeelding voor het menselijk oog meer detail bevat
Nadeel: Door de vermenigvuldigingen vraag het meer van de computer dan bij “averaging”.

Desaturation

Deze techniek pakt het gemiddelde van de minimale waarde en de maximale waarde. De techniek is te beschrijven als volgt: Pakt de kleuren kanalen van een afbeelding, zoek de minimale waarde van een pixel en zoek de maximale. Deze waardes worden bij elkaar opgeteld en gedeeld door twee. De resulterende waarde is de greyscale waarde. Bijvoorbeeld: pixel#1 = (R=124, G=200, B=75) -> new = (max (124,200,75) +min (124,200,75))/2 = (200+75)/2 = 275/2 -> Nieuwe greyscale waarde is ongeveer 138.

Voordeel: Best simpel in implementatie, de foto wordt niet snel over/onder belicht
Nadeel: Deze techniek doet “veel” berekeningen (met min en max functies).

Maximum Decomposition

Deze techniek is best simpel, het pakt de maximale (of minimale) waarde als greyscale waarde. De techniek (maximum Decompostion) is te beschrijven als volgt: Pakt de kleuren kanalen van een afbeelding, zoek de maximale waarde van een pixel en gebruik deze als greyscale waarde. Voorbeeld: pixel#1 = (R=124, G=200, B=75) -> new = max (124,200,75) -> greyscale waarde is 200.

Voordeel: De simpelheid van dit is een groot voordeel.
Nadeel: De afbeeldingen kunnen makkelijk over/onderbelicht raken.

Minimum Decomposition

De techniek (minimum Decompostion) is te beschrijven als volgt: Pakt de kleuren kanalen van een afbeelding, zoek de minimale waarde van een pixel en gebruik deze als greyscale waarde. Voorbeeld
pixel#1 = (R=124, G=200, B=75) -> new = min (124,200,75) -> greyscale waarde is 75.

Voordeel: De simpelheid van dit is een groot voordeel.
Nadeel: De afbeeldingen kunnen makkelijk over/onderbelicht raken.

Single color channel

Deze techniek is snel. Deze techniek pakt een van de ”color channels” als nieuwe greyscale waarde. De moeilijkheid ligt in het bepalen van welk “color channel” je moet gebruiken. Hier een klein voorbeeld van de werking van deze techniek. Pixel#1 = (R=124, G=200, B=75) -> new = G - > Greyscale waarde is 200. Pixel#1 = (R=124, G=200, B=75) -> new = R - > Greyscale waarde is 124

Voordeel: (Heel)Snel en geheugen efficiënt.
Nadeel: Per afbeelding verschilt de optimale channel en dit is soms moeilijk te bepalen.

Keuze

Wij hebben gekeken naar de voor en nadelen van deze verschillende methodes, deze naast onze doelen gelegd. Wij willen graag de snelheid en memory efficiency verbeteren tegenover de standaard implementatie.

Verder willen wij graag ook mooie afbeeldingen hebben, dit is meer een ‘should’ een geen ‘must’. Na deze vergelijkingen tussen de verschillende methodes zijn wij op twee kandidaten uit gekomen namelijk: Luminance en Singel color channel.

Luminance heeft als voordeel dat het de waardes corrigeert naar een mooiere afbeelding, maar dit gaat ten koste van de snelheid. Singel color channel is snel, maar geeft soms niet de mooiste resultaten.

Na een paar kleine testen zij we tot de conclusie gekomen dat Singel color channel de beste methode is voor ons, omdat deze het snelst en geheugen efficiënt is. Dit zijn de punten die wij willen verbeteren tegen over de standaard implantatie, daarom werd het na de testen vrij makkelijk om te kiezen voor singel color channel.

Implementatie

Voor de uitwerking van single color hebben wij gekozen om de groenwaarde van een pixel te gebruiken als grijswaarde. Zoals eerder bij Luminance beschreven werd, het menselijk oog is extra gevoelig voor de kleur groen. Daarom hebben wij de keuze gemaakt om de groene channel te gebruiken voor de implementatie van de methode. Om deze methode als formule uit te drukken zal dit als volgt eruit komen te zien: Greyscale = G.

Voor het implementeren van singel color channel zijn de volgende stappen nodig:

- Van de ingevoerde afbeelding moeten de pixel waardes kunnen worden opgevraagd,
- Uit deze waardes moet het groene kanaal gehaald worden.
- Met deze waardes wordt de greyscale representatie opgebouwd

De nieuwe greyscale afbeelding wordt opgebouwd uit deze pixels, hiermee is de grijs conversie compleet.

Hier mee zullen wij niet al te lang bezig mee zijn, we schatten er rond 5 uur mee bezig te zijn. Dit is inclusief de documentatie en debuggen.

Evaluatie

Wij zullen de huidige implementatie bestuderen, om een beeld te krijgen van hoe de huidige werking functioneert en welke data nodig is om onze eigen implementatie te laten werken. Wij zullen online onderzoek doen naar wat de beste methodes zijn en voor ook waarom deze methodes de beste zijn (voor ons probleem). Verder zullen wij ook vast leggen wat bevindingen zijn tijdens het implementatie proces.

Als wij een werkbare versie hebben dan gaan wij deze testen met veel data. Dit gaan wij doen om een duidelijke en betrouwbare informatie te kunnen verzamelen over onze implantatie. De data die wij willen gaan gebruiken is:

Ongeveer honderd afbeeldingen per resolutie van:

- 128 bij 128
- 256 bij 256
- 512 bij 512
- 1024 bij 1024

Tien afbeeldingen per resolutie met veel gebruikte computerscherm resoluties:

- 640 bij 480
- 1280 bij 720
- 1920 bij 1080
- 2160 bij 1440

De tijd die het programma nodig heeft om de greyscale conversie uit te voeren op te slaan tegen over die van de standaard implementatie, kan gebruikt worden om aan te tonen hoeveel sneller de nieuwe implementatie is. Het geheugengebruik van de conversie kan ook gemeten worden tijdens de omzettingen van kleur naar grijs afbeldingen. Dus we gaan de tijd en geheugen gebruik vast leggen in dit experiment.

Het verzamelen van deze afbeeldingen gaat veel tijd in zitten. In totaal zullen we ongeveer 450 afbeeldingen vinden. Wij schatten hier ongeveer 3-4 uur mee bezig te zijn.

Omdat we plannen om het test proces te automatiseren zal dit proces snel gaan.
We zullen per folder kunnen gaan testen. Dus wij schatten dat het testen zelf 2-3 uur zal duren.

Dit experiment is best veel werk, maar we simpel om te doen. Wij denken hierom dat de uitvoerbaarheid hoog is van dit experiment.

Aan de hand van deze gegevens die uit dit experiment kan een conclusie worden getrokken welke implementatie het snelst geconverteerd kan worden. Met dit experiment kunnen wij dus verschillende implementatie testen op hun snelheid en geheugen gebruikt.

Bronnen

Helland, T. (2011, Oktober 1). Seven grayscale conversion algorithms . Opgehaald van tannerhelland: http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/

N.B. (2017, Februari 2). Grayscale . Opgehaald van Wikipedia: https://en.wikipedia.org/wiki/Grayscale