Implementatieplan ToIntensityImage

Stefan van der Ham & Bas van Rossem
Vision
14-4-2019

1. Doel

Het doel van de opdracht is om een RGB afbeelding om te zetten in een intensity afbeelding. Het doel van deze implementatie is om ervoor te zorgen dat onze versie sneller is in het berekenen van de nieuwe waardes van de afbeelding en dat hij nog steeds gebruikt kan worden met betrekking tot de facial recognition.

2. Methoden

2.1. Gemiddelde

Een van de makkelijkste manieren om een RGB pixel om te zetten nar een intensity pixel om te zetten is om de gemiddelde waarde te nemen van de rood, groen en blauw waarde. Wat je dan krijgt is deze formule: Intensity = (R + G + B) / 3. Deze manier is heel snel omdat er geen moeilijke berekeningen aan te pas komen. Een probleem is dat sommige kleuren er te donker of te licht uit zien.

2.2. Helderheid gemiddelde [1]

Een probleem met het gemiddelde nemen van de 3 waardes is dat de intensiteit van het licht niet mee wordt genomen. Een formule die daar wel rekening mee houd is: Intensity = (0.3 * R) + (0.59 * G) + (0.11 * B). De afbeelding die door deze formule gegenereerd zou worden is lichter dan het normale gemiddelde en beter voor computer vision.

2.3. Luma methode [2]

De luma methode houd rekening met de manier waarop mensen licht zien, en dan vooral de intensiteit waarmee dat licht binnen komt. Hierbij komt de volgende formule aan bod: Intensity = (R * 0.2126) + (G * 0.7152) + (B * 0.0722). Deze methode is niet heel geschikt voor machine vision.

2.4. Single channel [3]

Een andere methode is om maar een enkel kanaal van licht te pakken. De formule hiervan is: Intensity = R of G of B. Dit is heel snel, maar je verliest veel detail omdat je maar een enkel kaneel bekijkt. Di tis niet geschikt voor computer vision.

3. Keuze

We kiezen om het gemiddelde van de 3 waardes te nemen, dit is een redelijke manier om detail in je afbeelding te krijgen en het is heel snel. Met die snelheid hopen we sneller te zijn dan de default implementatie.

4. Implementatie

Ons gekozen algortime gaan wij toepassen in de function stepToIntenityImage in StudentPreProcessing.cpp. Die functie zal een intensityImage aanmaken en vervolgens een for loop hebben die over alle pixels in de image gaat en de intensiteit waarde berekend. Deze intensiteit waarde wordt vervolgens die in de eerder aangemaakte intensityImage zet.

5. Evaluatie

De snelheid van de implementatie wordt getest door de functie heel vaak uit te testen van 5 tot 250 maal achter elkaar. Die timen we door middel van de vision timer van Arno Kamphuis, deze is te vinden op https://github.com/arnokamphuis/vision-timer.

Om er zeker van te zijn runnen wij eerst een maal de gui, zodat we weten hoe de afbeelding eruit ziet en of de face recognition werkt. Als dat werkt kunnen wij vervolgens een main.cpp runnen die automatisch tests maakt.

Om te testen of onze implementatie efficiënter is. Maken wij gebruik van de Visual studio memory snapshot functie. Daarin kunnen wij zien hoeveel allocations zijn gemaakt en hoeveel wij gebruik maken van de heap. We doen op deze manier 10 metingen.

Het implementeren van deze functie is niet veel werk, we verwachten dat we ongeveer 1,5 uur nodig hebben met het schrijven en testen. veel van onze tests kunnen automatisch, maar het uitlezen van de heap moeten we toch wel met de hand doen..

6. Bronnen

[1] Tutorialspoint.com. (z.d.). Grayscale to RGB Conversion. Geraadpleegd op 14 april 2019, van https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm

[2] Lemeden, R. (2019, 25 maart). A Closer Look at Color Lightness. Geraadpleegd op 14 april 2019, van https://thoughtbot.com/blog/closer-look-color-lightness

[3] Wikipedia contributors. (2019, 16 maart). Grayscale. Geraadpleegd op 14 april 2019, van https://en.wikipedia.org/wiki/Grayscale