Vision image shell en intensity

Door: Niels Risseeuw & Joost van Bussum
06-03-2020

Doel:

Het doel van de opdracht is om een nieuwe implementatie te schrijven van de intensity/grayscale image shell. De nieuwe implementatie zou preciezer moeten werken terwijl deze een vergelijkbaar resultaat levert qua snelheid als de originele implementatie. Ook hopen we hierbij in het proces efficiënter gebruik te maken van geheugen.

Methoden:

1. **RGB Channel average**

   RGB channel average is de makkelijkste methode. Je pakt de RGB waarden en deelt die door 3 en dat wordt de greyscale waarde.
   Gray = (Red + Green + Blue) / 3

2. **Luminosity Method**

   De luminosity methode in theorie hetzelfde als de RGB channel average methode met een kleine twist. Het verschil is dat de voorheen genoemde methode niet de menselijke interpretatie van het beeld mee neemt in de vergelijking. Omdat mensen rood, groen en blauw anders waarnemen dan computers neemt de luminosity methode hier de corresponderende factoren in mee. Veel gebruikte factoren zijn:
   Gray = (Red * 0.3 + Green * 0.59 + Blue * 0.11)
   Gray = (Red * 0.2126 + Green * 0.7152 + Blue * 0.0722)
   Gray = (Red * 0.299 + Green * 0.587 + Blue * 0.114)

3. **Desaturation Method**

   Desaturation werkt door een RGB triplet om te zetten in een HSL triplet. En vervolgens de saturation op 0 te zetten.
   Bij desaturation pak je de hoogste RGB waarde en de laagste, die twee tel je bij elkaar op en deel je door 2. Dat wordt je nieuwe greyscale waarde.
   Gray = ((Max(Red, Green, Blue) + Min(Red, Green, Blue)) / 2

4. **Single Channel Method**

   Single channel is bij verreweg de simpelste methode. Deze methode neemt een enkel kleur kanaal en gebruikt deze waardes direct als grijs waardes. Hierdoor kan je drie compleet verschillende resultaten krijgen die met verschillend succes werken. Voorbeeld algorithme:
   Gray = Red of Gray = Green of Gray = Blue

5. **Decomposition Method**

   Decomposition kan je zien als een simpele vorm van desaturation.
   Bij decomposition pak je de hoogste of laagste waarde van R G of B en dat is je greyscale waarde. Een maximum decomposition geeft een lichtere image terwijl een minimum decomposition een donkere image geeft.
   Gray = Max(Red, Green, Blue)
   Gray = Min(Red, Green, Blue)

Keuze:

Onderzoeksvraag: *"Is er een marginaal verschil te detecteren in snelheid, precisie en memory efficiëntie tussen de traditionele methode van gray scaling in de aangeleverde code en de eigen implementaties op basis van desaturatie en luminosity."*

Onderbouwing:
Wij hebben gekozen voor de desaturation en luminosity methode. Deze hebben wij gekozen omdat wij een goede combinatie wilde testen van snelheid en precisie. De desaturation methode gebruikt wat meer calculaties dan de andere methoden maar eindigt op een simpeler eindresultaat met minder onscherpe kleurovergangen. Hierdoor hopen wij scherpere edges te maken om hier dan vervolgens een betere detectie uit te krijgen en als gevolg een hogere precisie in gezichtsherkenning.

Implementatie:

Wij willen deze methode uitwerken door de formule uit te werken in de code. Deze formule willen wij gaan loslaten op een eventuele 1 dimensionele array om de snelheid te verhogen ten opzichte van een 2 dimensionale array. Ook willen wij kijken of na het desaturaten van de originele afbeelding, er door meer lightness te introduceren meer contrast komt tussen de edges en dit dus nog meer de edge detectie zou bevorderen.

Voor de luminosity willen wij een soortgelijk stappenplan volgen door te werken en vergelijken met een 1 dimensionele array.

Evaluatie:

Om onze code te testen hebben wij ervoor gekozen om op drie punten te testen:

1. Precisie
2. Snelheid
3. Memory Efficiëntie

Over alle drie deze punten zullen wij tests uitvoeren en meetrapporten maken over hoe goed de code heeft gepresteerd binnen deze categorie. Deze kunnen wij vergelijken met dezelfde tests die wij gaan uitvoeren op de aangeleverde code. Hieruit kunnen wij een dataset maken met eventuele veranderingen in de drie voorgenoemde categorieën om hier een procentueel verschil uit te halen.

De precisie van de twee methodes gaan wij testen door het percentage hits (correct gedetecteerde gezichten) over *N* aantal bijgeleverde foto's te berekenen per methode.

De snelheid testen wij door built-in tools van visual studio te gebruiken en te kijken naar hoe snel de twee methodes door de foto's heen gaan en alle bewerkingen hebben uitgevoerd van start tot einde.

Voor memory efficiëntie geldt hetzelfde, we gebruiken de built-in tools van visual studio om te kijken en te vergelijken hoeveel memory de twee methodes gebruiken ten opzichte van de andere methodes.

Het uiteindelijke doel is om al deze informatie en testresultaten op te nemen in de testrapporten en hier een definitieve uitslag te krijgen op onze onderzoeksvraag.