Meetrapport snelste algoritme

1.1. Namen en datum

Brandon Kroes en Maarten Wassenaar - 2 April 2019

1.2. Doel

Constateren wat de snelheid van de conversie algoritmes is en welke het snelst is.

1.3. Hypothese

Wij denken dat een van de enkel kanaal conversie het snelste is. Een enkel kanaal conversie heeft namelijk geen calculatie. Aangezien er een aantal duizend (of zelfs miljoen) pixels zijn die elk een calculatie nodig hebben zullen de algoritmes binnen de categorie het langst duren en algoritmes die geen calculaties doen aanzienlijk sneller zijn. De vraag welk van de enkel kanaal conversies het snelste is van ze is nog moeilijk te antwoorden in deze fasen.

1.4. Werkwijze

We zullen voor een enkele test set een algoritme 100 keer een gray scaling conversie doen op de foto's uit de test foto's die waren aangeleverd. Elke foto in de testset zal door al onze gebruikte algoritmes worden verwerkt, waarna we het gemiddelde nemen. Alle tests worden uitgevoerd op dezelfde computer. Op de algoritmes wordt dieper in gegaan in het implementatie plan.

In visueel studio hebben een aanpassing gedaan in de main.cpp zodat we door alle foto's heen lopen. Echter hebben we het werkende algoritme handmatig per test aangepast in de StudentPreProccesing.cpp, omdat we puur de functie willen hebben met verwerking voor ons onderzoek. We hebben overwogen om in de functie een switch te bouwen voor het gebruikte algoritme, toch hebben we dit niet gedaan, omdat dat processing tijd scheelt in de functie.

1.5. Resultaten

Foto\algoritme:	Single channel:	ITU-R_BT-709:	ITU-R_BT-601:	GIMP & Photoshop:	Desaturation:
child-1.png	28	29	29	29	162
female-1.png	26	27	27	27	160
female-2.png	8	9	9	9	53
female-3.png	27	27	27	27	160
male-1.png	27	28	27	27	162
male-2.png	26	27	27	27	162
male-3.png	27	27	27	27	161
Average:	24,14	24,86	24,71	24,71	145,71

Hierboven zijn de resultaten in tabel afgebeeld in milliseconden voor 100x de berekening.

1.6. Verwerking

We hebben de test per algoritme gerund en verwerkt in de bovenstaande tabel. Daarbij viel ons op dat er 1 afbeelding zeer uit de maat liep. Dit bleek te liggen aan de bestandsgrootte van “female-2.png”, die ongeveer in grootte een 3de omvatte ten opzichte van de andere afbeeldingen.

We hebben alle berekeningen gemeten aan het begin van de functie tot net na de functie. Al deze waardes hebben we opgeteld en daar een gemiddelde uitgenomen.

1.7. Conclusie

We zien kleine verschillen bij de eerste 4 algoritmes, de desaturation daarin tegen een stuk langzamer. Dit komt waarschijnlijk doordat bij dit algoritme, er 2 keer een sorteer algoritme moet optreden om de minimale en maximale verzadiging te vinden.

Single channel is het snelst. Dan komen we bij ITU-R_BT-601 en GIMP & Photoshop, Hier zien we een verrassende identieke uitkomst. Ondanks dat het ITU-R_BT-601 algoritme met 3 decimalen achter de komma vermenigvuldigt en GIMP & Photoshop maar met 2 decimalen achter de komma. Echter is bij ITU-R_BT-709 is er wel een verschil te zien. Dit werkt dan ook met 4 decimalen achter de komma.

Het snelste algoritme is dus single channel maar als kwaliteit ook belangrijk is bij de uitkomst dan zijn ITU-R_BT-601, ITU-R_BT-709 en GIMP & Photoshop ook een geschikte kandidaat.

1.8. Evaluatie

In onze hypothese stelde we dat single channel het snelste zou functioneren, dit is ook zo maar is relatief gezien niet heel veel sneller. De volgende 3 algoritmes volgen algauw de snelste op qua snelheid.

Onze proef is gedaan op 1 computer. Dit zal altijd verschillen per computer omdat dit afhankelijk is aan de snelheid van de computer. Dus eigenlijk zijn de verhoudingen tussen de algoritmes alleen noemenswaardig. Dan hebben we nog de functie zelf waar variabelen in worden geïnitialiseerd, de waardes van de oude kleuren afbeelding worden ingeladen en na het algoritme wordt het weer weggeschreven. Dit is natuurlijk niet perse de snelheid van het algoritme maar wel nodig om dit algoritme te testen. Echter omdat ieder meet resultaat dit bevat kun je ze wel tegen elkaar vergelijken.