Meetrapport titel

1.1. Namen en datum

Mike Hoogendoorn & Richard Janssen - 18-3-2020

1.2. Doel

Het doel van het experiment is om er achter te komen hoeveel de calculatietijd verschilt tussen de bestaande en nieuwe implementaties.

1.3.Hypothese

Ik verwacht dat het verschil in calculatie tijd tussen de standaard en nieuwe implementaties verwaarloosbaar is.

1.4. Werkwijze

We gaan per implementatie testen met de afbeeldingen child-1, Easter, 4K2, greenblob, purpleblob, en office. Per Implementatie gaan we 10 keer de afbeelding converteren naar een Intensiteits afbeelding en bijhouden hoeveel microseconden de computer daarmee bezig is..

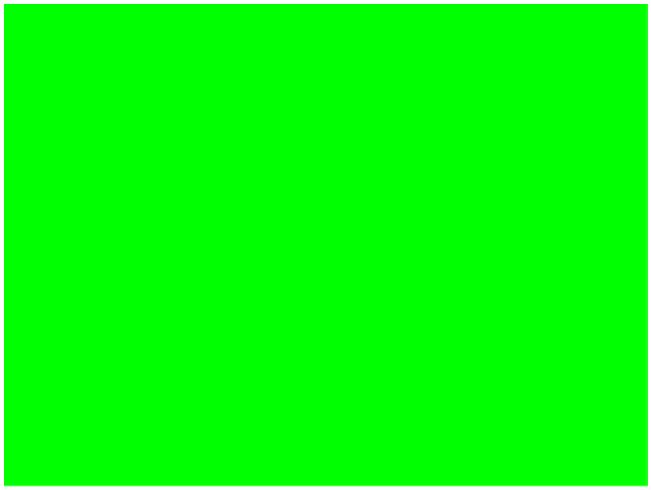
Aan het einde hebben wij dan per afbeelding per implementatie 10 keer de calculatie tijden die nodig waren om de afbeeldingen te converteren.

We gaan dan per afbeelding kijken welke implementatie het snelste is en welke implementatie over alle afbeeldingen het snelste is.

Child-1.png 255x255

Easter.png 1500x1600

4K2.png 4032x3024



greenblob.png 4032x3024

office.png 1500x1102

1.5. Resultaten

Geef de meetresultaten overzichtelijk weer in de vorm van een tabel en/of diagram.

child-1				
in µs	**Default**	**(R+G+B)/3**	**R/3 + G/2 + B/6**	**R * 30% + G * 59%, B * 0.11%**
1	11864	7642	7831	7570
2	11839	7653	7933	7556
3	11763	7639	7745	7551
4	11744	7610	7742	7548
5	11776	7643	7847	7531
6	11857	7602	7739	7547
7	11894	7601	7785	7532
8	11774	7653	7730	7571
9	11856	7604	7728	7570
10	11803	7630	7735	7557
Gemiddelde	**11817 µs**	**7627.7 µs**	**7781.5 µs**	**7553.3 µs**
Standaarddeviatie	**51,33116879**	**21,34400775**	**68,3784404**	**14,59109317**

easter				
in µs	**Default**	**(R+G+B)/3**	**R/3 + G/2 + B/6**	**R * 30% + G * 59%, B * 0.11%**
1	579855	361003	366352	360150
2	586550	366932	366066	360970
3	558360	363865	368747	364525
4	556112	362769	368891	364260
5	552236	362469	368529	361719
6	569777	361869	364964	355775
7	554356	358957	366819	355737
8	553778	362124	365641	354501
9	552933	359232	367699	357341
10	551678	359721	367414	357679
Gemiddelde	**561563,5 µs**	**361894,1 µs**	**367112,2 µs**	**359265,7 µs**
Standaarddeviatie	**12640,85073**	**2395,460589**	**1366,952718**	**3588,093553**

4K2				
in µs	**Default**	**(R+G+B)/3**	**R/3 + G/2 + B/6**	**R * 30% + G * 59%, B * 0.11%**
1	3545084	1816991	1856316	1965722
2	3559176	1834210	1862991	1967866
3	3566981	1822160	1857358	1967876
4	3562339	1822646	1853557	1961657
5	3551941	1830227	1868099	1969890
6	3560910	1827724	1863782	1959232
7	3556985	1849647	1862880	1960789
8	3557309	1838270	1853653	1971189
9	3570002	1826396	1853653	1918596
10	3542259	1831248	1856420	1969867
Gemiddelde	**3557298,6 µs**	**1829951,9 µs**	**1858870,9 µs**	**1961268,4 µs**
Standaarddeviatie	**8827,268776**	**9289,525134**	**5162,241383**	**15560,53038**

office				
in µs	**Default**	**(R+G+B)/3**	**R/3 + G/2 + B/6**	**R * 30% + G * 59%, B * 0.11%**
1	406496	283797	285869	280300
2	383181	268452	279164	268613
3	391175	268209	280320	266804
4	386375	269236	282138	268301
5	391532	268195	276459	265764
6	379756	268469	276432	266231
7	382104	270508	274630	280167
8	387592	267906	281190	270436
9	418525	279313	275545	272842
10	416591	273205	277688	278184
Gemiddelde	**394332,7 µs**	**271729 µs**	**278943,5 µs**	**271764,2 µs**
Standaarddeviatie	**14304,63748**	**5516,635045**	**3486,007785**	**5780,740356**

greenblob				
in µs	**Default**	**(R+G+B)/3**	**R/3 + G/2 + B/6**	**R * 30% + G * 59%, B * 0.11%**
1	3503322	1962042	2069705	1973239
2	3577233	1974521	1955219	1985101
3	3605057	1964951	2005065	1977989
4	3567148	1927510	2000686	1973321
5	3564515	1988957	2016187	1908211
6	3555595	1967655	2005408	1956762
7	3613657	1960559	2007840	1980317
8	3553407	1955410	1982416	1964735
9	3555296	1902177	2004588	1970488
10	3590658	1965139	2000474	1970079
Gemiddelde	**3568588,8 µs**	**1956892,1 µs**	**2004758,8 µs**	**1966024,2 µs**
Standaarddeviatie	**31229,0737**	**24691,31063**	**28571,52923**	**21814,48268**

1.6. Verwerking

Zoals in de resultaten te zien is hebben we de gemiddelde calculatietijd per implementatie al verwerkt. Nu gaan we berekenen hoeveel procent sneller onze implementaties zijn tegenover de default implementatie.

procentueel sneller tegenover de standaard implementatie			
	Default vs (R + G + B) / 3	Default vs R/3 + G/2 + B/6	Default vs R * 30% + G * 59% + B * 11%
child -1	$\frac{7627.7 - 11817}{11817} * 100 = -35.5\%$	$\frac{7781.5 - 11817}{11817} * 100 = -34.1\%$	$\frac{7553.3 - 11817}{11817} * 100 = -36.1\%$
easter	-35.6%	-34.6%	-36%
4K2	-48,6%	-47,7%	-44,9%
office	-31,1%	-29,3%	-31,1%
greenblob	-45,2%	-43,8%	-44,9%
Gemiddelde	**-41,6%**	**-40.3%**	**-39%**
Standaard deviatie	**9,279188183**	**9,695531617**	**6,851459698**

1.7. Conclusie

De (R + G + B) / 3 Intensity Image implementatie is gemiddeld 41,6% sneller dan de default implementatie.

De R/3+G/2+B/6 implementatie is gemiddeld 40,3% sneller dan de default implementatie

De R * 30% + G * 59% + B * 11% implementatie is gemiddeld 39% sneller dan de default implementatie

Dus al onze implementatie zijn ongeveer 40% sneller dan de standaard implementatie.

1.8. Evaluatie

Leg een verband tussen de getrokken conclusie en het doel van het experiment (en de hypothese). Ga daarbij ook in op bijvoorbeeld de meetonzekerheid als gevolg van de gebruikte meetmethoden of eventuele meetfouten.

Zoals in de resultaten en verwerking te zien is is de Default implementatie bij elke afbeelding duidelijk het langzaamst. Elke andere implementatie is gemiddeld ongeveer 40% sneller. Helaas is hier wel sprake van een grote standaarddeviatie. Die is namelijk voor 2 van de 3 implementaties rond de 9,5. verrassend is dat de de standaarddeviatie voor de implementatie R*30%+G*59%+B*11% lager ligt dan voor de andere implementaties. Waarom dit zo is kan ik niet verklaren. Waarschijnlijk ligt dit aan de testomgeving. Het is namelijk getest in een normale desktop windows 10 Home omgeving waar alle andere applicaties zoveel mogelijk uit zijn gezet. Dit is geen perfecte testomgeving om de calculatie tijd te testen omdat er nog steeds veel verschillende processen op de achtergrond lopen om windows 10 draaiend te houden.

Het is ook interessant om te zien dat bij 4K resolutie afbeelding meer winst wordt gepakt dan bij de kleinere afbeeldingen. Terwijl tussen de 255x255 afbeelding en de easter afbeelding geen duidelijk verschil zit. En office zelfs sneller is dan de twee. Terwijl het kwa grootte tussen de twee ligt.

Onze hypothese was dan ook onjuist. Er is wel degelijk een winst in calculatietijd te merken tussen de standaard en onze zelf gecodeerde implementaties.