Meetrapport: conversie snelheid

Stoeltie, Ferdi – 1665045
Agterberg, Ole – 1651981

18-03-2020

Image: http://bubble.ro/How_to_convert_an_image_to_grayscale_using_PHP.html

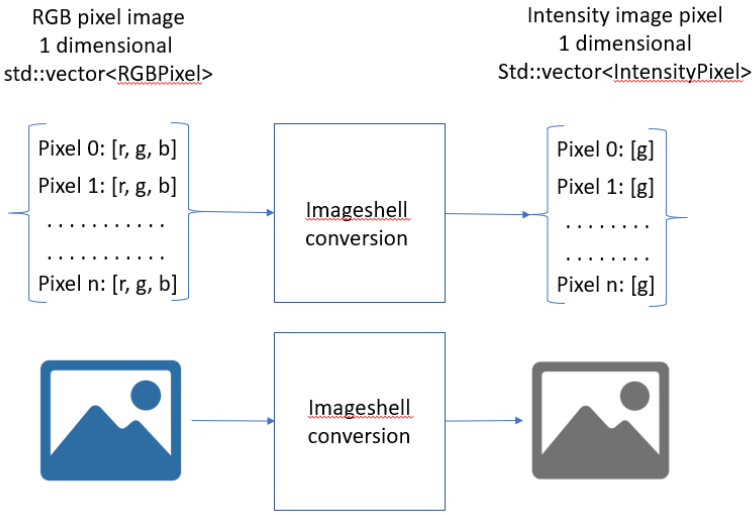
1. Doel

Het doel van het meetrapport is om te kijken of snelheid van de conversie van RGB naar intensiteit is verbeterd ten op zichten van de bestaande conversie. De snelheid wordt beïnvloed door het wegschreven van pixels naar de intensity klassen, naar het verkrijgen van de RGB pixels uit de RGB klassen en natuurlijk de conversie van RGB naar grayscale.

2. Hypothese

Wij verwachten dat de student implementatie minder snel zal zijn, omdat we niet de snelste conversie methode hebben gekozen. Daarnaast hebben we gekozen om de data op te slaan in een vector. Vector is voor het toevoegen van elementen niet snel. We denken dat de default implementatie gebruik maakt van een array, die erg snel elementen kan toevoegen.

Wel is er gekozen voor een vector waarvan de grootte, eenmalig gedefinieerd wordt door de hoeveelheid pixels N van de image. Hierdoor heeft de vector een constante grootte N en hoeft niet te worden verplaatst wanneer de pointers naar de pixels worden gezet. Schalen van een vector kost veel tijd en dat hoeft niet met deze aanpak.



3. Werkwijze

Voor elke test zal de functie executePreProcessingStep1 meerdere malen aangeroepen worden, het aantal samples. Bij elke afbeelding is het aantal samples 100, behalve bij de eekhoorn daar is het aantal 10. De tijd die nodig was om deze test uit te voeren wordt bijgehouden. Vervolgens wordt de test vijf keer herhaald voor de student klassen en vijf keer herhaald voor de default klassen. Hierna zal het verschil in tijd in percentage berekend worden door de volgende formule:

$$Snelheids\ verbetering = \left(\frac{Default_{time}}{Student_{time}} - 1\right) * 100\%$$

Het programma kan op vier verschillende manieren gecompileerd worden;

- O2S -> O2 (optimized for speed), single-threaded;
- O2M -> O2 (optimized for speed), multi-threaded;
- DM -> Default (not optimized), multi-threaded;
- DS -> Default (not optimized), single threaded.

Variable / Category	O2	Multi-threaded
O2S	✓	X
O2M	✓	✓
DM	X	✓
DS	X	X

Voor dit meetrapport is gekozen om de testen uit te voeren met de instellingen O2S en DS.

4. Resultaten

De resultaten van de metingen zijn in de volgende tabellen terug te vinden. De resultaten zijn in 100.000.000 van een seconde opgeschreven; 100000000 = 1 seconde.

Gebruikte afbeelding:

Afbeelding	**Pixels (h*b)**
Child-1	57375
Female-1	50310
Male-4	174592
Male-5	778680
Animal-1	2359296

Tabel met de gemaakte metingen. Dikgedrukte cijfers zijn de gemiddelde van de testen.

	Student			**Default**	
	O2S	**DS**		**O2S**	**DS**
Child-1	180864700	757285600	Child-1	140393900	666056000
	162445400	640912900		138415400	615432400
	161765500	517842800		153405800	592689300
	166638000	512369900		147653200	616219800
	168876000	461254900		133169400	599927200
	168117920	**577933220**		**142607540**	**618064940**
Female-1	167878700	838774000	Female-1	190057500	596530900
	149529500	665389600		154147500	588577100
	137392500	586412900		210465100	605766700
	135789700	508044700		181499500	597299200
	137529700	490337700		163502900	611728300
	145624020	**617791780**		**179934500**	**599980440**
Male-4	554010500	2110608100	Male-4	1010417300	2150367500
	525276400	2690071400		694622800	2131887600
	521435600	1668088900		608576600	2168401800
	652585400	1555005400		617556300	2130157900

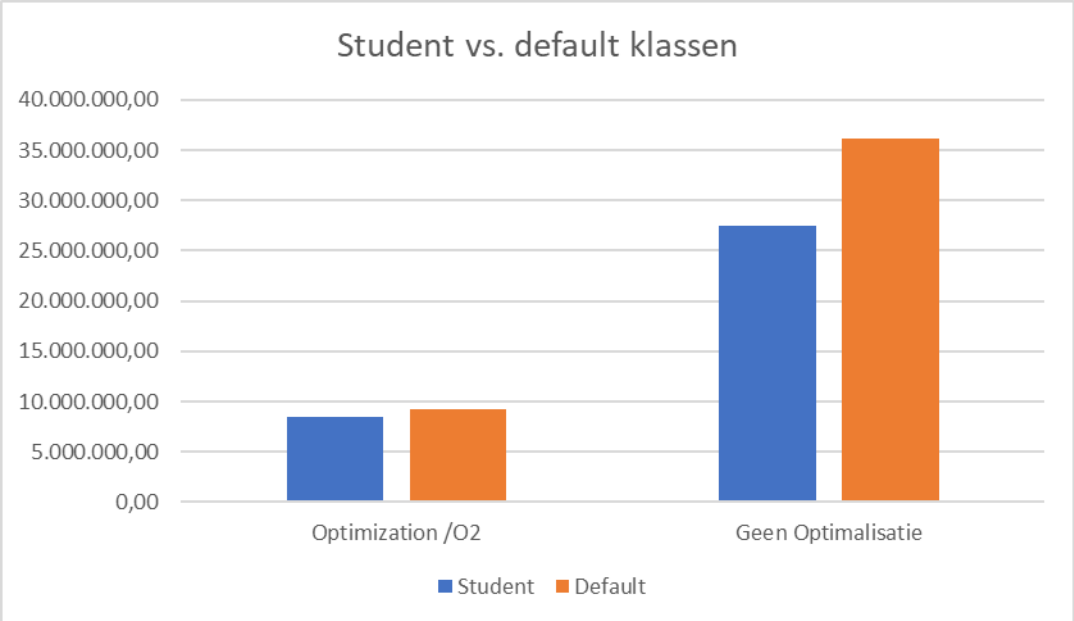
	964585500	1572208800		685718600	2128895500
	643578680	**1919196520**		**723378320**	**2141942060**
Male-5	3210855600	7843665100	Male-5	2661443100	11317604700
	2655215000	7286609500		2609242900	10910300900
	2072083100	7370217900		2586306600	11172444900
	2053688000	8442621900		2598502300	11316889500
	2046781100	8443485600		2671319100	10819146800
	2407724560	**7877320000**		**2625362800**	**11107277360**
	1895250800	4759066600		2254591300	5506063300
	1732424600	3879787000		2235737500	5652737500
Animal-1	1775494200	4881481400	Animal-1	2259621900	5784283300
	1735468700	3569592900		2255840300	5447923700
	1734618300	3836421500		2428611900	5414206700
	1774651320	**4185269880**		**2286880580**	**5561042900**

5. Verwerking

De verwerkte resultaten zijn allemaal door het aantal samples gedeeld. Dus 100 bij child-1, female-1, male-4, male-5 en 10 bij animal-1.

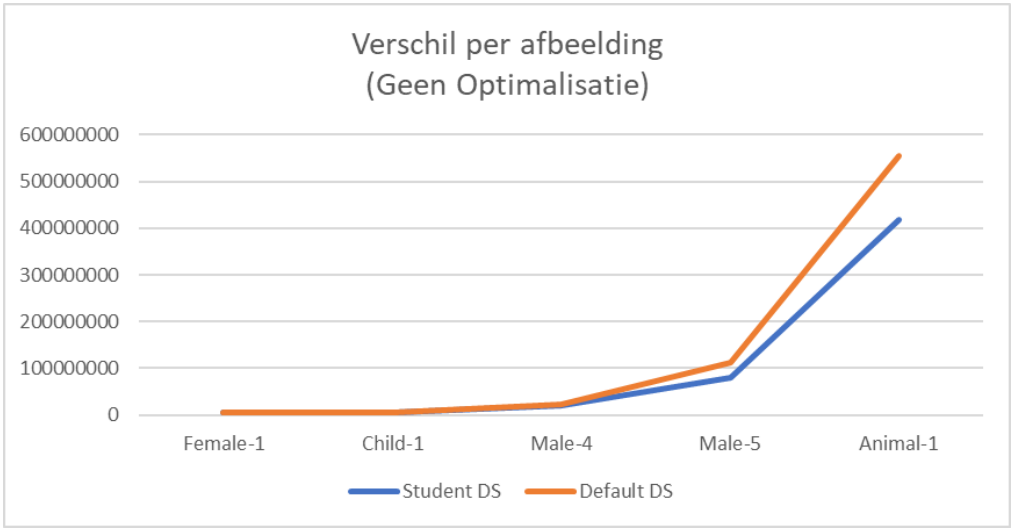
De volgende tabel en grafiek laten zien wat het verschil is tussen de student- en de default-klassen, zowel in de geoptimaliseerde als in de niet geoptimaliseerde code. De resultaten laten zien dat de student klassen sneller is dan de default klassen. De Imageshell is zonder dat de code op snelheid is geoptimaliseerd 32% sneller. Als de code wel geoptimaliseerd is dit 9%.

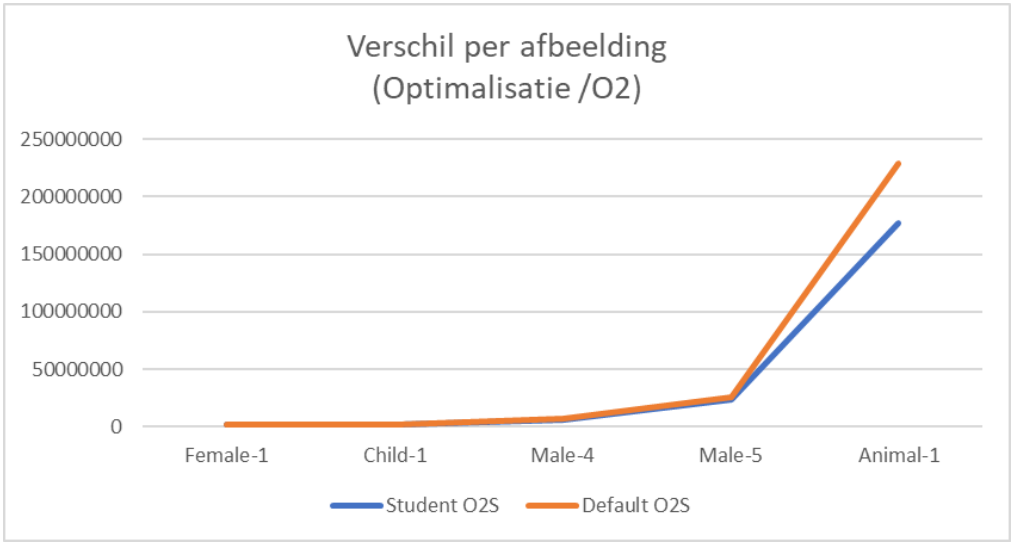
	Optimalisatie /O2	**Geen Optimalisatie**
Student	8412613	27480604
Default	9178208	36168162
Verbetering in percentage	**9%**	**32%**



De volgende twee tabellen en grafieken laten zien hoe de grootte van een afbeelding het verschil tussen de student en de default klassen kan beïnvloeden. Ze zijn namelijk op volghorde neergezet, female-1 is het kleinste en animal-1 is het grootste. Hierin is duidelijk te zien dat hoe groter de afbeeldingen hoe hoger de verbetering is.

	Student DS	Default DS	Student O2S	Default O2S	Verbetering
Female-1	6177918	5999804	1456240	1799345	2%
Child-1	5779332	6180649	1681179	1426075	2%
Male-4	19191965	21419421	6435787	7233783	12%
Male-5	78773200	111072774	24077246	26253628	34%
Animal-1	418526988	556104290	177465132	228688058	32%





6. Conclusie

We hadden verwacht dat het minder snel zou zijn. Dit is niet het geval, uit de resultaten is namelijk af te leiden dat de student implementatie sneller is dan de default implementatie. De gemeten verbetering was maximaal 32% bij de grootste afbeelding.