Meetrapport Imageshell speed

1. Namen en datum

Stefan van der Ham & Bas van Rossem, 6 april 2019.

2. Doel

We willen met dit experiment kijken welke van de twee implementaties sneller is. De default implementatie van de imageshell tegenover onze implementatie van de imageshell.

3. Hypothese

Wij verwachten tot onze implementatie 50% sneller is, omdat het een vrij simpele oplossing is. Wij verwachten dat wij ongeveer een half uur bezig zijn met het verzamelen van de testresultaten, dit komt vooral omdat alles automatisch gegenereerd wordt. Hoewel 1 iteratie van het programma heel snel is, gaat het toch best lang duren omdat wij het programma zo vaak laten draaien.

4. Werkwijze

4.1. Algemeen

Om de snelheid van de imageshell te testen wordt er gebruik gemaakt van de vision timer gemaakt door Arno Kamphuis (https://github.com/arnokamphuis/vision-timer). Hoe deze wordt geïmplementeerd wordt uitgelegd in 4.2 code uitleg.

De test wordt gedaan met afbeelding “child-1.png”, en wordt gedraaid in de debug mode van visual studio 2017.

4.2. Code uitleg

De imageshell wordt getest door de main, die de images inleest en de facerecognition uitvoert, meerde malen uit te voeren en die gegevens automatisch op te slaan in een .csv bestand.

We doen 5 tests achter elkaar waarbij het aantal keer dat de aangepaste main wordt uitgevoerd met een factor van wordt 5 vermenigvuldigd.

De aangepaste main voert zichzelf 10 keer uit. De aanpassing die aan de main is gemaakt is het volgende: zowel de default implementatie en de studentimplementatie worden baseN * mainiteratie maal uitgevoerd. Hierbij is baseN 5 * op welke iteratie van de test we zitten (1 t/m 5) en de mainiteratie is hoe vaak de main is uitgevoerd (1 t/m 10). De tijd die het programma erover doet om die vermenigvuldiging van die iteraties uit te voeren wordt opgeslagen het eerder besproken .csv bestand. Zo hebben wij volledig geautomatiseerde tests.

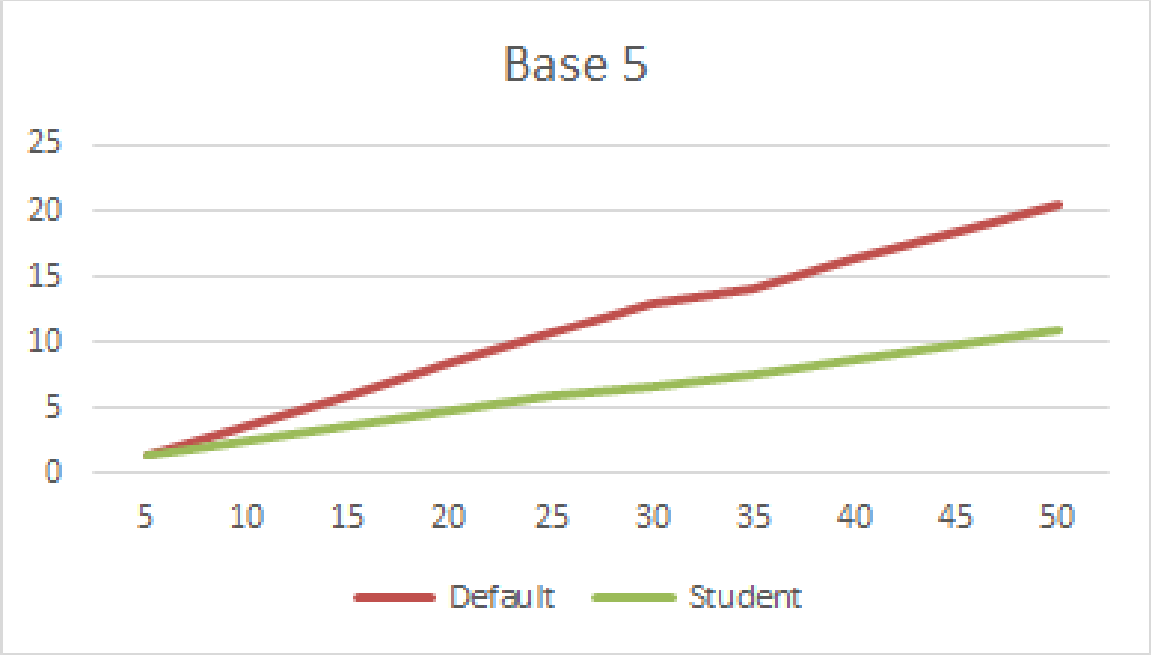
4.3. Opbouw code

Onze code heeft dus de volgende opbouw:

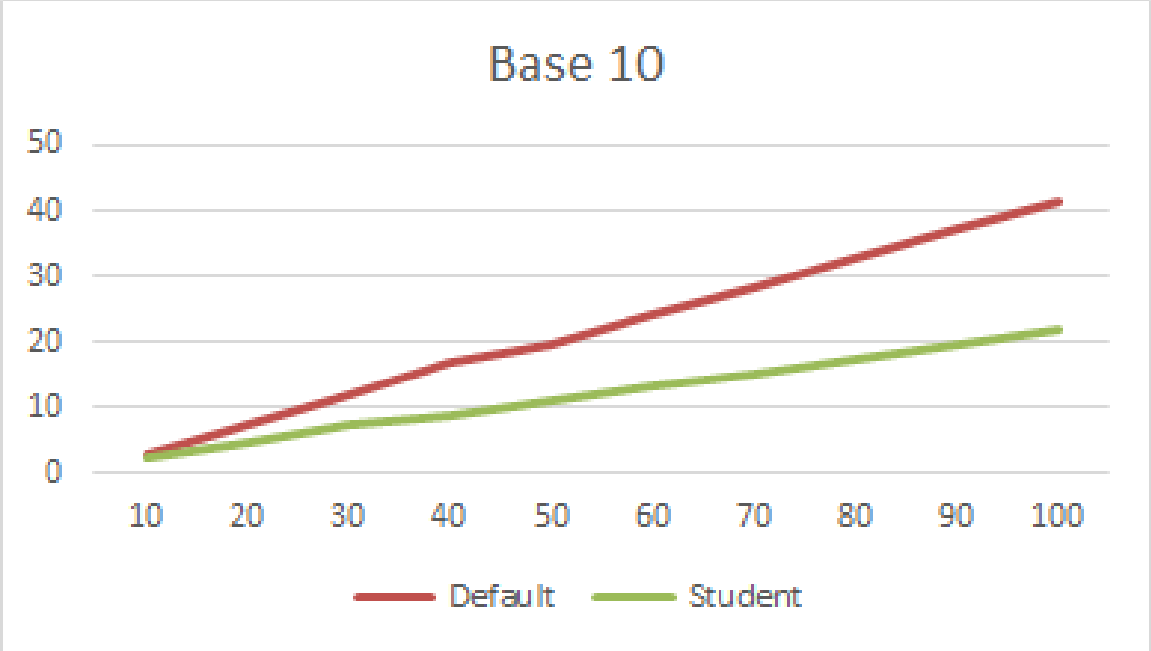
```
int main(int argc, char * argv[]) {
	for (int mainIteration = 1; mainIteration <= 5; mainIteration++) {
		baseN = 5 * mainIteration
		myBaseTimer = basetimer

		for (int iteratorCount = startingI; iteratorCount <= 10; iteratorCount++) {
			ImageFactory::setImplementation(ImageFactory::DEFAULT);

			myBaseTimer->start();
			for (int i = 0; i < baseN * iteratorCount; i++) {
				//Execute default imageshell implementation
			}

			myBaseTimer->stop();

			resultFile.write(results)

				myBaseTimer->reset();

			ImageFactory::setImplementation(ImageFactory::STUDENT);

			myBaseTimer->start();

			for (int i = 0; i < baseN * iteratorCount; i++) {
				//Execute student imageshell implementation
			}

			myBaseTimer->stop();

			resultFile.write(results)

		}
	}
}
```

5. Resultaten

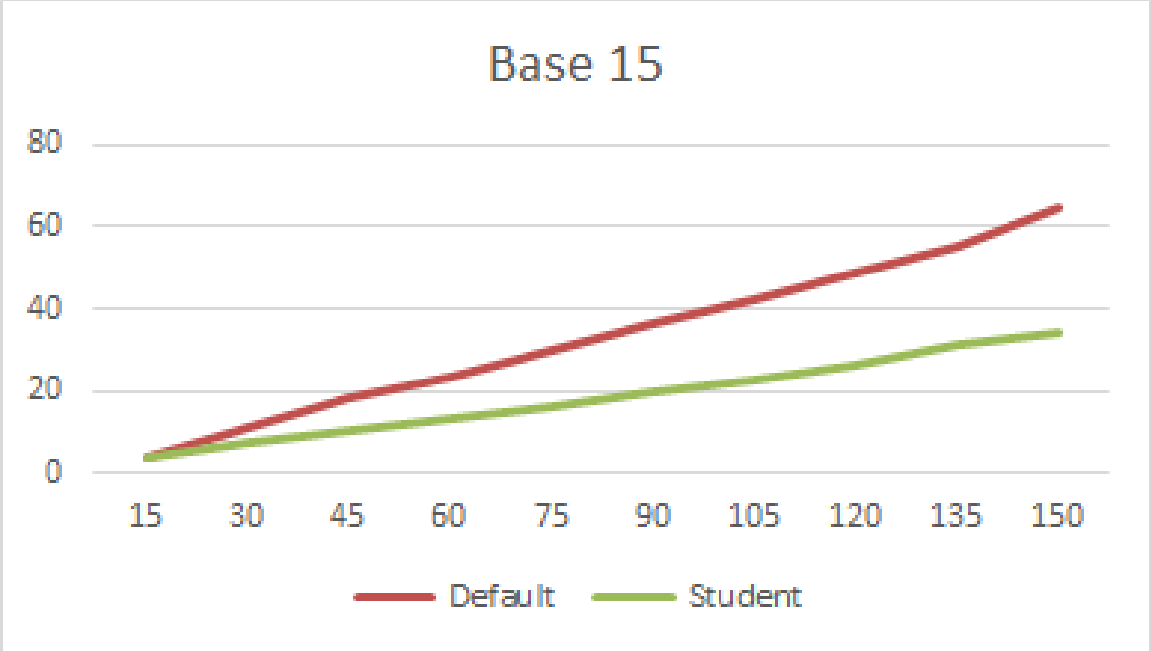
Base 5	***Cycles***	***Default (in seconden)***	***Student (in seconden)***	***Verschil (in seconden)***	***Gewogen verschil (in seconden per cycle)***
	5	1.43322	1.28974	0.14348	0.028696
	10	3.74853	2.40293	1.3456	0.13456
	15	5.97744	3.58336	2.39408	0.159605
	20	8.36134	4.75181	3.60953	0.180477
	25	10.7435	5.96316	4.78034	0.191214
	30	12.9383	6.50001	6.43829	0.21461
	35	14.0908	7.56266	6.52814	0.186518
	40	16.2728	8.61289	7.65991	0.191498
	45	18.3604	9.73837	8.62203	0.191601
	50	20.4859	10.8112	9.6747	0.193494
Total	**275**	**112.41223**	**61.21613**	**51.1961**	**0.167227**
Student % sneller	**45.543176**				



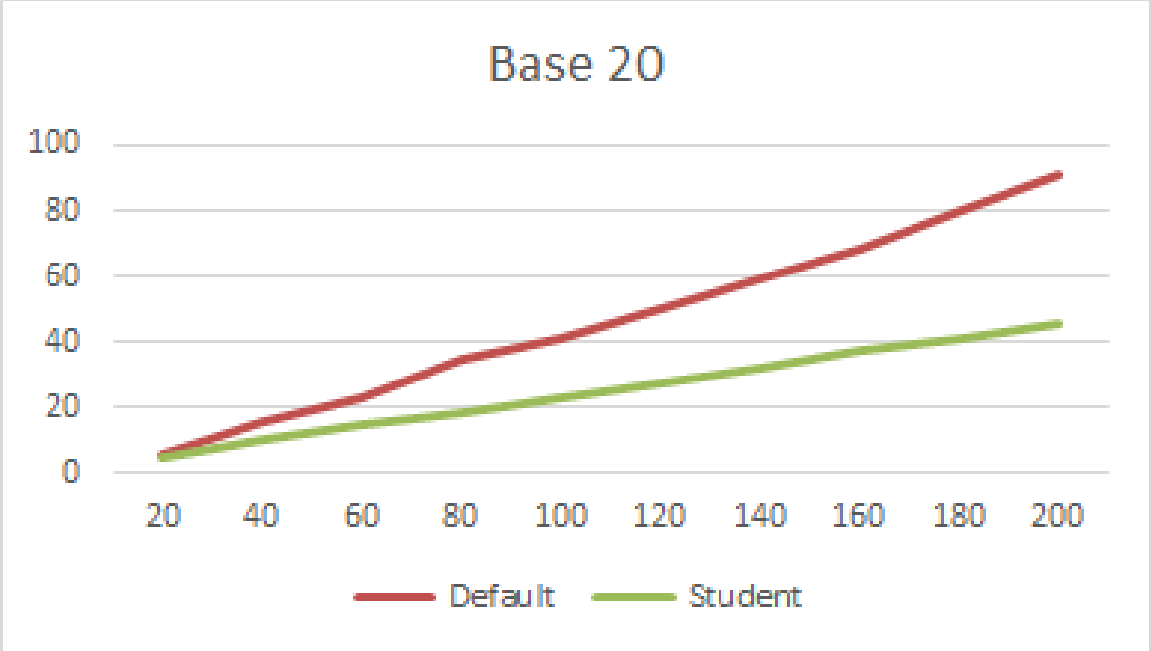
Base 10	***Cycles***	***Default (in seconden)***	***Student (in seconden)***	***Verschil (in seconden)***	***Gewogen verschil (in seconden per cycle)***
	10	2.57258	2.38643	0.18615	0.018615
	20	7.18996	4.77385	2.41611	0.120806
	30	11.9202	7.17161	4.74859	0.158286
	40	16.7386	8.85176	7.88684	0.197171
	50	19.756	10.8701	8.8859	0.177718
	60	23.9371	13.0249	10.9122	0.18187
	70	28.3192	15.23	13.0892	0.186989
	80	32.6713	17.4044	15.2669	0.190836
	90	37.0472	19.6006	17.4466	0.193851
	100	41.3547	21.8105	19.5442	0.195442
Total	**550**	**221.50684**	**121.12415**	**100.38269**	**0.162158**
Student % sneller	**45.318099**				



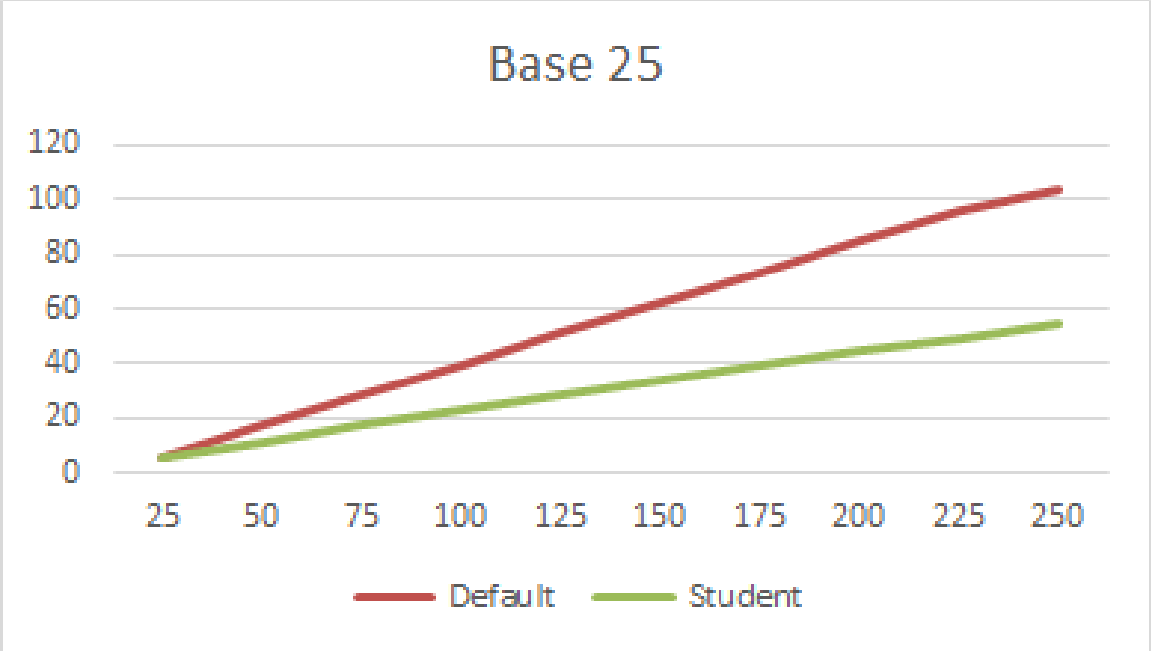
Base 15	***Cycles***	***Default (in seconden)***	***Student (in seconden)***	***Verschil (in seconden)***	***Gewogen verschil (in seconden per cycle)***
	15	3.84335	3.59333	0.25002	0.016668
	30	10.8149	7.21848	3.59642	0.119881
	45	18.0076	10.5595	7.4481	0.165513
	60	23.6423	13.1194	10.5229	0.175382
	75	29.5427	16.3159	13.2268	0.176357
	90	36.0185	19.6095	16.409	0.182322
	105	42.5333	22.7995	19.7338	0.187941
	120	48.978	26.1777	22.8003	0.190003
	135	55.4227	31.2018	24.2209	0.179414
	150	65.0045	33.8508	31.1537	0.207691
Total	**825**	**333.80785**	**184.44591**	**149.36194**	**0.160117**
Student % sneller	**44.744885**				



Base 20	***Cycles***	***Default (in seconden)***	***Student (in seconden)***	***Verschil (in seconden)***	***Gewogen verschil (in seconden per cycle)***
	20	5.21178	5.03205	0.17973	0.008987
	40	15.1774	9.89142	5.28598	0.13215
	60	22.3704	14.9629	7.4075	0.123458
	80	34.2377	18.1686	16.0691	0.200864
	100	40.9765	22.6737	18.3028	0.183028
	120	49.8665	27.2289	22.6376	0.188647
	140	59.0522	31.9003	27.1519	0.193942
	160	68.3714	37.2653	31.1061	0.194413
	180	79.7631	40.63	39.1331	0.217406
	200	91.1548	45.1539	46.0009	0.230005
Total	**1100**	**466.18178**	**252.90707**	**213.27471**	**0.16729**
Student % sneller	**45.749259**				



Base 25	***Cycles***	***Default (in seconden)***	***Student (in seconden)***	***Verschil (in seconden)***	***Gewogen verschil (in seconden per cycle)***
	25	5.68612	5.69274	-0.00662	-0.00026
	50	17.0211	11.3878	5.6333	0.112666
	75	28.3328	16.9763	11.3565	0.15142
	100	39.5432	22.5409	17.0023	0.170023
	125	50.7934	28.2563	22.5371	0.180297
	150	62.2665	33.8184	28.4481	0.189654
	175	73.4179	39.4691	33.9488	0.193993
	200	84.8043	45.286	39.5183	0.197592
	225	96.1907	48.8059	47.3848	0.210599
	250	103.166	54.1173	49.0487	0.196195
Total	**1375**	**561.22202**	**306.35074**	**254.87128**	**0.160217**
Student % sneller	**45.413628**				



6. Verwerking

Hier zijn alle totaalresultaten bij elkaar. Het totaal bij gewogen verschil is het gemiddelde van het verschil.

	Cycles	***Default (in seconden)***	***Student (in seconden)***	***Verschil (in seconden)***	***Gewogen verschil (in seconden per cycle)***
Base 5	275	112.41223	61.21613	51.1961	0.16722718
Base 10	550	221.50684	121.12415	100.38269	0.162158377
Base 15	825	333.80785	184.44591	149.36194	0.160117208
Base 20	1100	466.18178	252.90707	213.27471	0.167289863
Base 25	1375	561.22202	306.35074	254.87128	0.160217355
Total	**4125**	**1695.1307**	**926.044**	**769.08672**	**0.163401997**
Student % sneller	**45.37035**				

Student % sneller is berekend door het totale verschil te delen door de totale default tijd.

7. Conclusie

De imageshell implementatie van de student is 45.37% sneller dan de default implementatie. We kunnen dus concluderen dat de imageshell die wij hebben gemaakt sneller is dan de default implementatie van de imageshell.

8. Evaluatie

Het doel van dit experiment was om te kijken hoeveel sneller onze implementatie van de image shell was in vergelijking tot de default implementatie. Onze hypothese was bijna goed, we zaten er maar een paar procentpunten naast. We hebben heel wat metingen gedaan, maar de code crashte heel vaak als het preprocessen van het rechteroog werd uitgevoerd. Ook was er ergens een memory-leak. We weten dit waar die vandaan komt en op welke manier deze invloed hadden op de meetresultaten. Door die crashes zijn we iets langer bezig geweest dan een half uurtje, maar dat scheelde niet veel.