Meetrapport snelheid van de student implementatie en de standaard opencv implementatie

Jullian van Doorn en Jari van Dam 10-4-2018

Doel

Het doel van dit experiment is bepalen welke implementatie sneller is, de eigen implementatie of de standaard opencv implementatie.

Hypothese

De verwachting is dat de student implementatie sneller is dan de standaard opencv implementatie. De opencv implementatie maakt namelijk ook gebruik van de sobel operator maar voert ook nog extra berekeningen uit. Omdat in de eigen implementatie al deze extra berekeningen weglaten zijn zou deze implementatie sneller moeten zijn.

Werkwijze

De uitvoertijd van de student implementatie en de standaard opencv implementatie zullen afzonderlijk gemeten worden voor iedere afbeelding uit de testset. Er zal gebruik gemaakt worden van de vision-timer gemaakt door Arno Kamphuis (https://github.com/arnokamphuis/vision-timer).

Als eerst dient de code van de student implementatie geschikt gemaakt te worden om de uitvoertijd te bepalen. Hiervoor dient code toegevoegd te worden om de timer te starten, te stoppen en het resultaat naar het scherm schrijven. Als eerst moet de code om de timer te starten geplaatst worden. Dit gebeurt door in de functie StudentPreProcessing::stepEdgeDetection de volgende code bovenaan neer te zetten.

```
BaseTimer* bt = new BaseTimer();
bt->start();
```

Hierna blijft de code voor de uitvoer van de eigen implementatie staan. Aan het einde van deze functie wordt de code gezet om de timer te stoppen en de waarde naar het scherm te schrijven.

```
bt->stop();
std::cout << bt->elapsedMicroSeconds() << std::endl;
```

Vervolgens worden ongeveer dezelfde stappen doorlopen voor standaard opencv implementatie. Ook hierin zal als eerst de code om de te timer te starten toegevoegd worden. Dit gebeurt door dezelfde code als bij de eigen implementatie bovenaan in de functie DefaultPreProcessing::stepEdgeDetection te zetten.

```
BaseTimer* bt = new BaseTimer();
bt->start();
```

Hierna staat dan de code die de standaard opencv implementatie uitvoert. Hierna zal weer een stuk code volgen om de timer te stoppen en die waarde naar het scherm te schrijven.

```
bt->stop();
std::cout << bt->elapsedMicroSeconds() << std::endl;
```

Zodra deze code is toegevoegd kan voor elke afbeelding bepaald worden hoe snel deze herkend kan worden met de eigen implementatie en met de student implementatie. Eerst wordt van iedere afbeelding uit de testset bepaald wat de uitvoertijd is bij de standaard opencv implementatie. Vervolgens wordt er overgeschakeld naar de eigen implementatie en wordt opnieuw voor iedere afbeelding uit de testset bepaald wat de uitvoertijd is.

Resultaten

Hieronder zijn alle resultaten weergegeven in twee tabellen. Elke afbeelding is 5 keer gemeten om zo te proberen een gemiddelde waarde te krijgen.

Student implementatie uitvoertijd in µs					
Afbeelding/run	1	2	3	4	5
child-1.png	53607	60283	51915	52336	59461
female-1.png	65480	71974	68457	62009	74159
female-2.png	19779	17969	18573	20259	24496
female-3.png	65607	57444	62972	61251	58706
male-1.png	64509	58362	68116	59360	67884
male-2.png	74122	69387	60342	63890	62569
male-3.png	74688	69035	64646	60238	69386

Standaard implementatie uitvoertijd in µs					
Afbeelding/run	1	2	3	4	5
child-1.png	13419	9092	8482	8679	8396
female-1.png	8247	9900	11158	8479	8508
female-2.png	4345	2774	4104	2756	3530
female-3.png	13295	8241	8077	11172	10325
male-1.png	11625	12381	12500	8389	12293
male-2.png	13694	11995	8611	9399	8408
male-3.png	8368	8427	10690	10034	8434

Verwerking

Hieronder zijn van de gemeten tijden per afbeelding en implementatie gemiddelden berekend. Hierbij zijn de hoogste en laagste waarde weggelaten omdat deze meestal een grote invloed hebben op het gemiddelde terwijl het in dit experiment vooral gaat om wat de uitvoertijd meestal zal zijn.

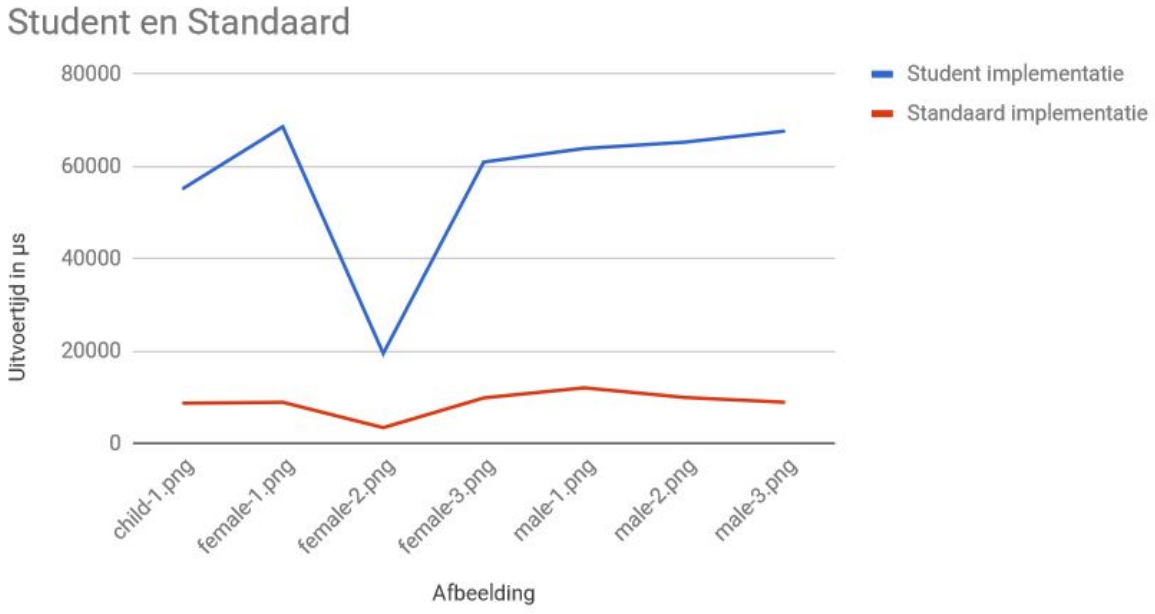
Student implementatie uitvoertijd in µs						
Afbeelding/run	1	2	3	4	5	Gemiddelde zonder hoogste en laagste waardes in hele µs
child-1.png	53607	60283	51915	52336	59461	55135
female-1.png	65480	71974	68457	62009	74159	68637
female-2.png	19779	17969	18573	20259	24496	19537
female-3.png	65607	57444	62972	61251	58706	60976
male-1.png	64509	58362	68116	59360	67884	63918
male-2.png	74122	69387	60342	63890	62569	65282
male-3.png	74688	69035	64646	60238	69386	67689

Standaard implementatie uitvoertijd in µs						
Afbeelding/run	1	2	3	4	5	Gemiddelde zonder hoogste en laagste waardes in hele µs
child-1.png	13419	9092	8482	8679	8396	8751
female-1.png	8247	9900	11158	8479	8508	8962
female-2.png	4345	2774	4104	2756	3530	3469
female-3.png	13295	8241	8077	11172	10325	9913
male-1.png	11625	12381	12500	8389	12293	12100
male-2.png	13694	11995	8611	9399	8408	10002
male-3.png	8368	8427	10690	10034	8434	8965

In onderstaande tabel zijn per afbeelding de gemiddelden van de student implementatie en de standaard opencv implementatie naast elkaar gezet. Hierdoor wordt het zichtbaar dat de verschillen tussen de beide implementaties heel groot zijn. Om te kijken of dit verschil overal even groot is zijn de percentages berekend. Hieruit blijkt dat hoeveel sneller een implementatie is sterk afhangt van naar welke afbeelding gekeken wordt.

Gemiddelde uitvoertijd in µs zonder hoogste en laagste waarde vergeleken per afbeelding			
Afbeelding	**Student implementatie**	**Standaard implementatie**	**student/standaard * 100 (%)**
child-1.png	55135	8751	630
female-1.png	68637	8962	766
female-2.png	19537	3469	563
female-3.png	60976	9913	615
male-1.png	63918	12100	528
male-2.png	65282	10002	653
male-3.png	67689	8965	755
Gemiddeld	57311	8880	644

Uit de grafiek kan geconcludeerd worden dat uitvoertijd binnen de implementaties redelijk constant zijn onafhankelijk van de afbeelding. De dip bij female-2.png is te verklaren doordat deze afbeelding minder pixels bevat dan de andere afbeeldingen.



Conclusie

Uit de bovenstaande resultaten kan geconcludeerd worden dat de standaard opencv implementatie gemiddeld genomen ongeveer 644% sneller is dan de eigen geschreven implementatie. Hoe groot het verschil precies is hangt af van de afbeelding. Bij sommig afbeeldingen is het verschil grooter dan bij andere afbeeldingen. Echter is de uitvoertijd van van dezelfde implementatie elke keer ongeveer gelijk. Er is alleen geen relatie tussen de uitvoertijd van de eigen implementatie en de standaard opencv implementatie.

Evaluatie

Het uitvoeren van het experiment zoals dit in het implementatieplan was bedacht ging goed. Er waren geen noemenswaardige problemen bij het integreren van de vision-timer. Ook het uitvoeren van de meting heeft niet tot problemen geleid.

De eigen implementatie voor edge detection is niet sneller dan de standaard implementatie. De eigengemaakte implementatie is zelfs een stuk langzamer dan de opencv implementatie. Dit zou kunnen komen doordat de omgeving waarin de edge detection wordt uitgevoerd is geoptimaliseerd is voor de standaard implementatie. Het kan zo zijn dat er bijvoorbeeld meer debug code uitgevoerd worden voor de student implementatie dan voor de standaard implementatie. Hierdoor zou de uitvoering van de standaard implementatie veel sneller kunnen zijn. Ook kan het dat er voor de opencv implementatie gebruik wordt gemaakt van low-level code, bijvoorbeeld assembly.

Als wij een zelfde snelheid als opencv willen bereiken zullen ook wij een deel van de code in bijvoorbeeld assembly moeten schrijven om op die manier de code nog efficiënter te maken. Om de kwaliteit van de eigen implementatie nog hoger te maken zal er ook gekeken moeten worden hoe er meer gezichten herkend kunnen worden. Momenteel kan de eigen implementatie maar een beperkt aantal gezichten uit de testset herkennen.