1. Meetrapport titel

1.1. Namen en datum

Nick Goris en Roxanne van der Pol, 09-04-2018

1.2. Doel

Het meten of deze nieuwe methode wel tot een bruikbare afbeelding leidt, specifiek voor de geïmplementeerde Facial Recognition.

1.3. Hypothese

Onze hypothese is dat deze nieuwe methode wel een bruikbare afbeelding zal leveren, die tevens sneller is kleinere code oplevert dan de eerder geïmplementeerde methodes.

1.4. Werkwijze

De timer is geïmplementeerd in onze code: https://github.com/arnokamphuis/vision-timer.

Deze timer wordt gestart aan het begin van de main(); en eindigt als de main() klaar is met zijn functionaliteit. De tijd in microseconde wordt dan naar de console geprint. De main wordt in een loop gestopt die 100 maal uitgevoerd wordt.

Voor de grootte van de code wordt gekeken naar de grootte van de resulterende object files van elke implementatie.

1.5. Resultaten

Resultaten van de timer:

	Studenten implementatie1	Studenten implementatie2	Standaard implementatie	Studenten implementatie3
Tijd verstreken (us) test 1	148151	112138	149157	150792
Tijd verstreken (us) test 2	154375	123166	142615	142431
Tijd verstreken (us) test 3	146209	111797	146815	164746
Tijd verstreken (us) test 4	146482	118792	147926	147731
Tijd verstreken (us) test 5	154409	115283	150880	150809
.. Tests 6-100	-	-	-	-
Gemiddelde verstreken tijd (us)	**153430**	**112161**	**151909**	**150958**

Bij bovenstaande tabel moet wel vernoemd worden dat de tweede implementatie heel vaak faalde. De exacte reden hiervoor is onbekend, maar de metingen voor Studenten implementatie2 zijn dus niet relevant.

Grootte van de code:

	Studenten implementatie 1	Studenten implementatie 2	Standaard implementatie	Studenten implementatie3
RGBImage (student or not)	21245 bytes	21245 bytes	41885 bytes	21245 bytes
IntensityImage (student or not)	17549 bytes	17549 bytes	29452 bytes	17549 bytes
PreProcessing (student or not)	7732 bytes	7760 bytes	103895 bytes	8290 bytes

1.6. Verwerking

Voor de tijd: alle resultaten voor iedere implementatie zijn bij elkaar opgeteld, waar een gemiddelde uit berekend is. Hieruit blijkt dat onze eerste implementatie langzamer is dan de standaardimplementatie. De resultaten van onze tweede implementatie zijn onbetrouwbaar.

Voor grootte: onze implementaties zijn op alle fronten kleiner dan de standaardimplementatie. Studenten implementatie3 is in PreProcessing groter uitgevallen dan verwacht. De grootte van de RGBImage en IntensityImage voor de studentenimplementaties zijn gelijk, aangezien in deze code niets verandert is tussen de twee implementaties.
Er moet hierbij wel gezegd worden dat de standaardimplementatie met opzet versluierd is, wat voor grotere code kan zorgen.

1.7. Conclusie

Uit deze resultaten kan geconcludeerd worden dat onze nieuwe implementatie sneller is dan de standaard en de vorige implementatie(s), maar wel grotere code oplevert dan bij de vorige implementaties. De code is nog steeds kleiner dan de standaardimplementatie.

Het is met de gekozen implementaties dus nog steeds niet mogelijk om een manier te kiezen die zowel kleiner als sneller is dan de standaard implementatie, en of de standaardimplementatie of onze implementatie gekozen wordt hangt dus af van het doel van de toepassing.

1.8. Evaluatie

Het doel van dit onderzoek is behaald. De nieuwe implementatie werkt naar behoren en is tegenover de standaardimplementatie en de vorige studenten implementaties gezet in een snelheidsvergelijking en in een vergelijking van grootte van de betreffende object files. De hypothese is echter fout – de nieuwe implementatie bleek sneller dan vorige resultaten, maar onze code was wel groter.