1. Meetrapport titel

1.1. Namen en datum

Jessy Visch
28-05-2018

1.2. Doel

Het doel van het expiriment is bewijzen dat het door mij geimplementeerde edge detection algoritme correct werkt. Daarnaast wordt de eigen implementatie vergeleken met de huidige implementatie op het gebied van snelheid.

1.3. Hypothese

De verwachting is dat de eigen implementatie langzamer is dan de huidige (openCV) implementatie. Dit omdat OpenCV door experts is geoptimaliseerd en mijn implementatie kan ongetwijfeld beter en efficienter.

1.4. Werkwijze

Op iedere afbeelding in de testset wordt de eigen geschreven edge detectie methode uitgevoerd. De output wordt vergeleken met de output van de OpenCV implementatie. Daarnaast wordt de tijd die het kost om de edges te detecteren gemeten en vergeleken. Dit wordt 100 keer herhaald om eventuele extreme resultaten te normaliseren.

1.5. Resultaten

	Bestaande implementatie X100	Eigen implementatie X100	Bestaande implementatie X500	Eigen implementatie X500	Bestaande implementatie X1000	Eigen implementatie X1000
Totaal	113MS	2213MS	523MS	11061MS	1021MS	22738
Gemiddeld	1,13MS	22MS	1,046MS	22MS	1,021MS	22MS
Factor x sneller	19x	-	21x	-	21x	-

ID	Bestaande implementatie	Eigen implementatie	Overige stappen slagen
Female-1.PNG			Ja
Female-2.PNG			Nee
Child-1.PNG			Ja
Female-3.PNG			Ja

Male-1.PNG			Ja
Male-2.PNG			Ja
Male-3.PNG			Ja

1.6. Verwerking

Er zijn een aantal berekeningen gebruikt om tot de cijfers in de bovenstaande tabel te komen:

Berekenen van verstreken tijd

Er wordt een timer gestart. Int start = get_time::now(). Vervolgens wordt de edge detection taak uitgevoerd. Dan wordt de eindtijd opgeslagen: int end = get_time::now(). De verstreken tijd wordt berekend door de starttijd van de endtijd af te trekken. Dus: int verstrekenTijd = end – start. De verstrekenTijd wordt opgeteld bij het totaal. Dit totaal wordt vervolgens gedeeld door het aantal keren dat de edge detection tijd is uitgevoerd, hier komt het gemiddelde uit.

Berekenen van factor x sneller in tabel

Om te bepalen hoeveel sneller de bestaande implementatie is de volgende berekening gehanteerd:GemiddeldeTijdVanHuidigeImplementatie/GemiddeldeTijdVanEigenImplementatie = X.

1.7. Conclusie

Uit de experimenten blijkt dat huidige implementatie bij meer dan 500 iteraties zo'n 21 keer sneller is dan de eigen implementatie. Van alle afbeeldingen in de testset slagen de vervolgstappen bij 1 van de afbeeldingen niet. Dit komt neer op een slagingspercentage van 85%. De afbeeldingen van de eigen implementatie zien er "schoner" uit. Deze afbeeldingen bevatten minder ruis(witte spikkels). Er is getracht om de falende afbeelding (Female-2.png) alsnog werkend te krijgen echter is dit niet gelukt. Een reden hiervoor heb ik niet kunnen ontdekken.

1.8. Evaluatie

In de hypothese werd gesteld dat de huidige openCV implementatie waarschijnlijk sneller zou zijn dan de eigen implementatie. Deze hypothese is waar gebleken. Dit is te wijden aan het feit dat openCV vele malen beter geoptimaliseerd is dan de eigen implementatie. De eigen implementatie kan nog op meerdere punten verbeterd worden, maar de verwachting is dat het nooit sneller gaat zijn dan openCV. Van de testset werkte alle afbeeldingen op 1 na. Waarom deze afbeelding niet werkte is niet duidelijk.