1. Edge detection snelheid

1.1. Namen en datum

Alexander Freeman
Sander Kolman
3/6/2016

1.2. Doel

In dit meetrapport willen we de snelheid van ons edge detection algoritme testen. Daarnaast willen we die vergelijken met het standaard algoritme.

1.3. Hypothese

Ik verwacht dat ons algoritme langzamer is dan de default implementatie, mede dankzij het feit dat opencv beter geoptimaliseerd is dan ons programma. Daarnaast is onze methode van convolutie niet optimaal geïmplementeerd.

1.4. Werkwijze

We meten honderd keer de tijd dat beide algoritmes doen over hetzelfde plaatje. Hiervan rekenen we het het maximum, de gemiddelde tijd en de verhouding tussen deze waardes voor beide algoritmen uit. Hieruit kunnen we conclusies trekken over de efficiëntie en de algoritmes vergelijken.

1.5. Resultaten

	Default (50 keer)	Onze (50 keer)	Default (100 keer)	Onze (100 keer)	Default (150 keer)	Onze (150 keer)
Gemiddeld (s)	0.01064	0.29302	0.01067	0.29	0.01038	0.286513
Maximum (s)	0.014	0.324	0.014	0.335	0.014	0.33
Totaal (s)	0.532	14.651	1.067	29.462	1.557	27.6024
Verhouding totale tijd	27.5395		27.3703		27.1423	

1.6. Conclusie

Ons algoritme is een stuk trager dan het algoritme wat gebruikt word door de default methode. Dit komt waarschijnlijk door het feit dat de convolutie bij ons minder geoptimaliseerd is en doordat onze blurring meerdere keren wordt toegepast.

De verhouding tussen de tijd van het default algoritme en die van ons is echter een lineaire verhouding. Ons algoritme heeft dus dezelfde Big-O notatie. Door bijvoorbeeld kernels samen te voegen, zou de verhouding nog kunnen worden verminderd.

Daarnaast kan de convolutie code nog worden geoptimaliseerd door bijvoorbeeld de verticale delen en de horizontale delen apart uit te rekenen. Hierdoor hoeft niet elke keer de hele kernel herberekend hoeven te worden, alleen het nieuwe stuk van het plaatje. Dit verminderd de verhouding ook nog.

1.7. Evaluatie

Ons algoritme werkt zoals verwacht trager dan de implementatie van opencv. Wel is gelukkig de Big-O notatie van dezelfde klasse, dit betekent dat de oplossingsstrategie goed is en er alleen nog berekeningen gedaan worden die overbodig zijn en weg geoptimaliseerd kunnen worden. Een strategie die nog het meeste snelheidsverschil zou kunnen maken is de berekeningen in de gpu uit te voeren, als dit echt nodig zou zijn.