1. Meetrapport Edge detection

1.1. Namen en datum

Stein Bout
Nick Swaerdens
Laurens van der Sluis

14-04-2018

1.2. Doel

Edge detectie wordt gebruikt voor het detecteren van objecten in afbeeldingen. Er zijn een groot aantal berekeningen nodig om dit goed te kunnen doen. Het doel van dit experiment is het implementeren van een snel algoritme dat een optimaal resultaat geeft.

Met dit document willen we aantonen dat het Deriche algoritme een sterk alternatief is van het veelgebruikte Canny algoritme.

Het meetrapport analyseert de snelheid van onze implementatie van het Deriche algoritme. Dit resultaat moet de volgende lokalisatie stappen kunnen uitvoeren met een correct resultaat.

De onderzoeksvraag van dit meetrapport luidt:
Hoe vergelijkt het Deriche algoritme in snelheid met andere implementaties van andere algoritmes?

1.3. Hypothese

De hypothese is dat zowel smoothing als de gradiënt magnitude en direction sneller werkt dan de standaard implementatie. We verwachten dat de OpenCV implementatie van het Canny algoritme beter geoptimaliseerd is en de laatste stappen sneller zal uitvoeren.

Wij denken dat onze implementatie sneller is omdat wij verwachten dat het Deriche algoritme minder berekeningen hoeft te maken per pixel. Maar we verwachten niet een heel groot verschil met de de standaard implementatie omdat deze implementatie geen edge thinning en edge directions gebruikt dus minder berekeningen hoeft te maken.

1.4. Werkwijze

Wij hebben twee afbeeldingen van verschillende groottes gebruikt voor het testen van de algoritmes. Het algoritme wordt met verschillende kernel groottes uitgevoerd om de snelheid met verschillende kernels te testen. Deze groottes zijn 3×3, 5×5 en 9×9. De grote van de kernel heeft geen effect op Deriche omdat het Deriche algoritme met een formule over de afbeelding heen itereert.

Elk algoritme wordt 5 keer per kernel uitgevoerd. Hier wordt vervolgens het gemiddelde van genomen om grote verschillen te onderdrukken. Daarnaast wordt elk algoritme ook getest op de duratie van de volledige executie. Al deze resultaten zullen vervolgens in een tabel worden gezet.

De verstreken tijd zal worden berekend met de chrono library om te kijken hoe lang de implementatie er over doet. We controleren ook of het resultaat van de implementaties kan worden gebruikt in de lokalisatie stappen.

Stappenplan:

Stap 1: Start met het bijhouden van de tijd vlak voordat de functie van het algoritme wordt aangeroepen.
Stap 2: Voer het algoritme uit.
Stap 3: Meet de verstreken tijd.
Stap 4: Herhaal voor andere input afbeeldingen en kernel groottes.

1.5. Resultaten

Algoritme	Kernel 3×3	Kernel 5×5	Kernel 9×9
Standaard	0,1636	x	0,5806
Laplacian	0,215	0,6792	0,893
Sobel	0,1902	0,3626	0,5262
Deriche	1,342	1,342	1,342

Volledig Algoritme	Time(ms)
Standaard	0,5806
Canny	0,982
Deriche	10,5174

Testresultaten van afbeeldingen van 255px × 255px.

Algoritme	Kernel 3×3	Kernel 5×5	Kernel 9×9
Standaard	5,4688	x	32,9256
Laplacian	7,5822	31,6872	47,741
Sobel	5,4604	16,148	32,43
Deriche	116,3626	116,3626	116,3626

Volledig Algoritme	Time(ms)
Standaard	32,9256
Canny	44,7656
Deriche	959,0508

Testresultaten van afbeeldingen van 1960px × 1960px.

Zoals te lezen is uit de tabellen is het Deriche algoritme het langzaamst. Wat het meest opvallend is, is het verschil in tijd bij grote afbeeldingen wanneer het hele algoritme wordt uitgevoerd.

Wat ook opvallend is, is de sterke groei van tijd dat het Deriche algoritme meemaakt bij grotere afbeeldingen.

1.6. Verwerking

De hypothese is niet correct. De huidige implementatie van het Deriche algoritme blijkt niet geoptimaliseerd genoeg om de prestaties van de OpenCV implementaties te evenaren. Dit is duidelijk en zonder uitzondering te zien in de resultaten.

Bij de volledige uitvoeren van het standaard algoritme worden een aantal stappen overgeslagen. Deze stappen zijn de gradiënt directions en de edge thinning technieken. Dit geeft het standaard algoritme een groot voordeel.

1.7. Conclusie

Zoals te zien is in de verwerking van de resultaten is canny algoritme 10 keer sneller dan onze implementatie van Deriche. Onze implementatie is ook veel langzamer dan de standaard implementatie maar zoals beschreven staat in de verwerking is dit niet geheel betrouwbaar.

Het Deriche algoritme is het langzaamst van alle testonderdelen.

1.8. Evaluatie

Tijdens het maken van het implementatieplan werd het duidelijk welke methoden er zijn en wat hun voordelen en nadelen waren. Hierdoor was het makkelijk om een goede keuze te maken welke methoden er gebruikt gingen worden voor de edge detection.

De implementatie geeft niet al te snel resultaat dit zou verbeterd kunnen worden door meer optimalisaties toe te passen en meer de beschikbare hardware te benutten. Een belangrijke optimalisatie zou kunnen zijn om alle vectoren te vervangen door een array. Dit zorgt ervoor dat de image niet meer op de heap staat wat een significante performance increase zou moeten geven.

Het Deriche algoritme biedt de mogelijkheid voor parallellisatie. Deze parallellisatie kan worden toegepast over de derivatives op beide assen. Ook kan de formule parallel worden uitgevoerd want de rijen en kolommen zijn onafhankelijk van elkaar dus hier kan parallellisatie worden toegepast.

Het zou leuk zijn om deze algoritmes op langzamere hardware te testen om te zien wat de verschillen zijn wanneer er weinig rekenkracht beschikbaar is.