1. Implementatieplan Edge detection

1.1. Namen en datum

Stein Bout
Nick Swaerdens
Laurens van der Sluis

09-03-2018

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.

1.3. Methoden

Laplacian Edge Detection

Laplacian edge-detection maakt gebruik van maar één kernel. Het berekent de tweede orde afgeleiden in een single pass. Dit heeft als gevolg dat laplacian erg snel werkt. Hieronder zijn er twee kernels te zien voor een Laplacian edge-detection:

De Laplaciaan		
0	-1	0
-1	4	-1
0	-1	0

De Laplaciaan (Ook diagonaal)		
-1	-1	-1
-1	8	-1
-1	-1	-1

Dit zijn twee voorbeelden van kernels die te gebruiken zijn bij laplacian. Als er een grotere benadering gewenst is dan is het eventueel mogelijk om een 5x5 kernel te maken. Een 5x5 kernel is heel eenvoudig, 24 in het midden en de rest is weer -1. Laplacian's voordeel heeft echter wel een groot nadeel. Het is extreem gevoelig voor ruis. Dit komt omdat laplacian werkt met afgeleiden voor de tweede orde. Hierdoor is het verstandig om eerst de ruis te verminderen. Dit kan bijvoorbeeld door middel van een Gaussian blur. Laplacian geeft soms uitmuntende resultaten. Helaas werkt het door zijn gevoeligheid voor ruis niet altijd goed en is het in sommige gevallen alleen maar lastig.

Canny Edge Detection
Het algoritme van Canny kan worden opgesplitst in 5 verschillende stappen. Alleen de eerste drie stappen zullen worden besproken. De laatste twee zijn thresholding en vallen buiten de scope van dit document.

1. Smoothing: Gaussian-blur toepassen om ruis te verwijderen.
2. Gradiënten vinden: Markeer de locaties waar de gradiënt magnitudes hoog zijn.
3. Non-maximum suppression: Alleen de lokale maxima moeten als edges worden gemarkeerd.

Smoothing:

Zoals alle andere edge detection systemen is Canny ook gevoelig voor ruis. Daarom is het aangeraden om de ruis zoveel mogelijk te onderdrukken. Om dat te doen wordt er een Gaussian filter overheen gehaald. Over het algemeen is een 5x5 Gaussian kernel voldoende. Het is van belang dat de kernel niet te groot is. Hoewel de ruis beter onderdrukt wordt met een grotere kernel, wordt de edge detectie onbetrouwbaar als de kernel te groot wordt. Bovendien gaat het filter dan ook veel meer resources nodig hebben.

Gradiënten vinden:

Na het smoothen zullen de gradiënt magnitudes en directions moeten worden gevonden. Dit wordt gedaan door een kernel voor de X en Y directie over de afbeelding te halen. Het resultaat hiervan wordt dan samengevoegd om de gradiënt magnitude te krijgen. Dit resultaat zal vervolgens worden gebruikt om de gradiënt directions te bepalen die nodig zijn voor non-maximum suppression.

Non-maximum suppression

Na de edge detectie moeten de gedetecteerde lijnen verdund worden. Dit kan door middel van Non-maximum suppression. Dit wordt gedaan door middel van de gradiënt directions die gemaakt zijn in de vorige stap. Voor elke pixel wordt naar de gradiënt gekeken, als de pixel kleiner is dan de pixel waar de gradiënt naar wijst en de pixel diagonaal daar tegenover dan wordt deze verwijderd.

Deriche edge detector (Canny-Deriche)
Derich's algoritme is gebaseerd op het werk van Canny. Het enige verschil zit in de eerste twee stappen. Deriche gebruikt een IIR filter i.p.v. Canny's FIR filter om de afbeelding te smoothen en de gradient magnitudes te verkrijgen. Voor het aanbrengen van een grote hoeveelheid smoothing zal Deriche een stuk sneller zijn dan Canny. Dit komt door Canny's FIR kernel die groter wordt afhankelijk van de hoeveelheid smoothing die aangebracht moet worden.

Het algoritme kan met een enkele sigma-waarde van sterkte worden veranderd. Hoe hoger de sigma hoe beter de lokalisatie van objecten in de afbeelding. Hoe meer de sigma de 0 benaderd, hoe sterker de smoothing wordt.

Tot slot kan Deriche's algoritme goed gebruik maken van parallellisatie.

$$f(x) = \frac{S}{\omega} e^{-\alpha|x|} sin\omega x$$

Deriche's IIR filter.

Sobel Edge Detection
Sobel gebruikt twee kernels. Een kernel voor de horizontale edges en een kernel voor de verticale edges en voor een betere detectie wordt een smooth gebruikt. De kernels kunnen er als volgt uit zien:

Horizontaal		
-1	0	1
-2	0	2
-1	0	1

Verticaal		
-1	-2	-1
0	0	0
1	2	1

1.4. Keuze
We hebben ervoor gekozen om het Deriche algoritme te implementeren. De reden hiervoor is de efficiëntie van een IIR filter en het feit dat er maar één enkele parameter meegegeven hoeft te worden. De parameter is afhankelijk van de afbeelding die gebruikt wordt. Door andere waardes te gebruiken kunnen betere resultaten behaald worden.

Het Deriche algoritme brengt echter ook risico's met zich mee. Zo is er weinig informatie beschikbaar online en wordt het algoritme niet vaak gebruikt omdat Canny de standaard is.

1.5. Implementatie
Het Deriche algoritme wordt volgens de formule in figuur 1 gebruikt. De aanpassingen van Deriche kunnen aan de hand van Deriche's coëfficiënten (te zien in figuur 2) en de onderstaande formules naar code omgezet worden.

$$y^1_{ij} = a_1 x_{ij} + a_2 x_{ij-1} + b_1 y^1_{ij-1} + b_2 y^1_{ij-2}$$
$$y^2_{ij} = a_3 x_{ij+1} + a_4 x_{ij+2} + b_1 y^2_{ij+1} + b_2 y^2_{ij+2}$$
$$\theta_{ij} = c_1 (y^1_{ij} + y^2_{ij})$$
$$y^1_{ij} = a_5 \theta_{ij} + a_6 \theta_{i-1j} + b_1 y^1_{i-1j} + b_2 y^1_{i-2j}$$
$$y^2_{ij} = a_7 \theta_{i+1j} + a_8 \theta_{i+2j} + b_1 y^2_{i+1j} + b_2 y^2_{i+2j}$$
$$\Theta_{ij} = c_2 (y^1_{ij} + y^2_{ij})$$

Figuur 1, Deriche's algoritme uitgedrukt in stappen.

	smoothing	x-derivative	y-derivative
k	$\frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}}$	$\frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}}$	$\frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}}$
a_1	k	0	k
a_2	$ke^{-\alpha}(\alpha-1)$	1	$ke^{-\alpha}(\alpha-1)$
a_3	$ke^{-\alpha}(\alpha+1)$	-1	$ke^{-\alpha}(\alpha+1)$
a_4	$-ke^{-2\alpha}$	0	$-ke^{-2\alpha}$
a_5	k	k	0
a_6	$ke^{-\alpha}(\alpha-1)$	$ke^{-\alpha}(\alpha-1)$	1
a_7	$ke^{-\alpha}(\alpha+1)$	$ke^{-\alpha}(\alpha+1)$	-1
a_8	$-ke^{-2\alpha}$	$-ke^{-2\alpha}$	0
b_1	$2e^{-\alpha}$	$2e^{-\alpha}$	$2e^{-\alpha}$
b_2	$-e^{-2\alpha}$	$-e^{-2\alpha}$	$-e^{-2\alpha}$
c_1	1	$-(1-e^{-\alpha})^2$	1
c_2	1	1	$-(1-e^{-\alpha})^2$

Figuur 2, Deriche's filter coëfficiënten.

Met de formules hierboven kunnen de smoothing en gradiënt magnitudes X en Y worden verkregen. Om deze gradiënt magnitudes vervolgens samen te voegen wordt de onderstaande formule gebruikt:

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

Figuur 3, samenvoegen van de gradiënt magnitudes X en Y.

Na het samenvoegen van de gradiënt magnitudes moeten de gradiënt directions worden gevonden. Dit wordt gedaan met de formule:

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

Figuur 4, gradiënt directions.

De gradiënt directions worden gebruikt voor non-maximum-suppression. Dit dund de edges uit voordat thresholding zal worden toegepast. Non-maximum-suppression gebruikt de gradiënt directions om te zien met welke pixels de huidige pixel vergeleken moet worden. Als de huidige pixel het grootst is zal de waarde worden bewaard, zo niet dan wordt deze verwijderd.

De output van dit proces is een afbeelding met dunne edges waar vervolgens thresholding op uitgevoerd kan worden om ruis en zwakke edges te verwijderen.

1.6. Evaluatie

Om te kijken hoe goed de implementatie het doel heeft behaald, wordt er een meetrapport gemaakt. Aan de hand van dit meetrapport zal worden gekeken hoe snel het algoritme de edge detection kan uitvoeren. Met als eis dat het resultaat nog steeds gebruikt kan worden in de lokalisatie stappen.

Het meten van de snelheid zullen we doen met chrono library. De snelheid zal worden vergeleken met de default implementatie. Om de metingen zo nauwkeurig mogelijk uit te voeren zullen we verschillende tests doen met verschillende parameterwaarden.

1.7 Bronvermelding

Algemeen
http://aishack.in/tutorials/sobel-laplacian-edge-detectors/
http://users.polytech.unice.fr/~lingrand/Ens/up/Lesson7and8-segmentation.pdf

Canny Edge Detection
http://kiwi.bridgeport.edu/cpeg585/CannyEdgeDetector.pdf

Deriche edge detector (Canny-Deriche)
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.476.5736&rep=rep1&type=pdf
https://en.wikipedia.org/wiki/Deriche_edge_detector
https://en.wikipedia.org/wiki/Infinite_impulse_response
https://www.sciencedirect.com/science/article/pii/104996609190006B