1. Implementatieplan titel

1.1. Namen en datum

- Twan Hoogveld
- David de Jong

17-04-2020

1.2. Doel

Het doel van de implementatie is om een zo nauwkeurig mogelijke edge detection toe te passen op een gezicht & dit te verbeteren met dynamic threshholding. Dit project zal redelijk trial and error zijn, omdat het een goed, leerzaam begin zal zijn aan edge detection.

Het probleem bij de huidige implementatie is dat deze problemen geeft bij de lokalisatie van bepaalde aspecten van het gezicht, het probleem hierachter is de edge detection verwachten wij. Als dit resultaat word verbeterd, dan wordt de overige processing ook een stuk duidelijker.

Niet alleen is het verbeteren van de accuratie belangrijk, maar willen wij ook letten op de snelheid. Mocht dit te lang duren, dan heeft het nog weinig nut. Er word in de te gebruiken methodiek een scheiding gemaakt tussen accuratie en snelheid.

1.3. Methoden

Wij gaan onderzoek doen naar de verschillende soorten edge detection methodes, hieruit gaan we een onderbouwen oordeel trekken om de beste manier van edge detection te kunnen toepassen. De verschillende soorten edge detection zullen wij hieronder behandelen.

Sobel

De Sobel operator is gemaakt in 1968, door Irwin Sobel & Gary Feldman.
De manier waarop de sobel operator werkt, is dat deze de intensiteit tussen pixels berekend om op deze manier een scheiding te kunnen maken tussen randen op een afbeelding. Het verschil is een berekening op de X-as en de Y-as, dit verschil maak je door een 3x3 matrix (Kernel) te gebruiken zoals deze:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Deze kernel haal je over alle pixels heen van de image, bij iedere iteratie meet je de verandering van het gradiënt van de pixels dat in deze kernel valt. Hoe groter de verandering van de pixel intensiteit, hoe significanter de edge. Dit algoritme is dus zeer afhankelijk van convolutie.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Bronnen ->

https://medium.com/@aryamansharda/how-image-edge-detection-works-b759baac01e2
https://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-3-greyscale-conversion/

https://en.wikipedia.org/wiki/Kernel_(image_processing)#Convolution
https://en.wikipedia.org/wiki/Sobel_operator
https://www.projectrhea.org/rhea/index.php/An_Implementation_of_Sobel_Edge_Detection
https://www.semanticscholar.org/paper/Sobel-Edge-Detection-Algorithm-Gupta-Mazumdar/6bcafdf33445585966ee6fb3371dd1ce15241a62
https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

Laplacian

0	-1	0
-1	4	-1
0	-1	0

The laplacian operator

-1	-1	-1
-1	8	-1
-1	-1	-1

The laplacian operator (include diagonals)

De Laplacian edge detection lijkt op de Sobel methode. Het grootste verschil is dat de Laplacian methode maar 1 kernel gebruikt. Een serieus probleem met deze filter is dat het heel erg gevoelig is voor noise. Dit zou je dus zoveel mogelijk moeten reduceren, door bijvoorbeeld het gebruik van een guassian blur. Maar op die manier verlies je dus ook kwaliteit en scherpte. Het grootste voordeel is dat het maar 1 kernel gebruikt, dit is computationeel veel sneller dan een methode die 2 kernels gebruikt. Net als Sobel, is dit uit te rekenen met convolutie. De input van de methode is grayscale image.

Bronnen ->

https://www.aishack.in/tutorials/sobel-laplacian-edge-detectors/
https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

Prewitt

Prewitt is een manier voor edge detection. De operator berekent het kleurverloop van de intensiteit bij elke pixel gegeven de richting van de grootste toename tussen licht en donker en hoe vaak die verandering plaatsvindt in die richting.

De operator gebruikt twee verschillende maskers voor het berekenen van edge detection, een masker in de verticale richting en een masker in de horizontale richting. Het past de maskers toe op de afbeelding en berekent de verschillende pixel intensiteiten bij delen waar randen zijn. Hiermee verandert de operator de linker en rechterkant van de rand waardoor de rand zelf intenser wordt. Hetzelfde geldt voor onder en boven. De input voor de methode is een grayscale beeld, of RGB beeld. Met een RGB input, gaat het algoritme over alle 3 de kleur kanalen heen & dat gaat dus wel ten kosten van rekenkracht & snelheid.

$$\mathbf{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$

Bij deze afbeelding is er eerst het verticaal masker gebruikt en vervolgens het horizontale masker.

Bronnen:

https://www.tutorialspoint.com/dip/prewitt_operator.htm

https://en.wikipedia.org/wiki/Prewitt_operator

https://www.ijser.org/researchpaper/Edge-Detection-by-Using-Canny-and-Prewitt.pdf

Robert's cross

Net als Sobel gebruikt Robert's cross 2 kernels om de intensiteit tussen pixels te vergelijken. Het werkt wel volledig anders. Dit komt omdat er 2 maal een 2x2 kernel wordt gebruikt. De kernels zijn gelijk aan elkaar, alleen 90 graden gedraaid tegenover de ander. Zoals de afbeelding hiernaast laat zien.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

Het voordeel van deze manier is dat het goed werkt op schuine randen, van ongeveer 45 graden. De input van de methode is een grayscale image.

Bronnen ->

http://homepages.inf.ed.ac.uk/rbf/HIPR2/roberts.htm

Laplacian of Guassian

De methode is een 2D isotrope meeting voor de 2^{e} ruimtelijke afgeleide van een afbeelding. De Laplacian van een afbeelding highlight regio's van snelle intensiteit veranderingen Dit is dus handig voor edge detection.

Het wordt toegepast op een afbeelding die eerst is gladgestreken met iets dat een guassian filter om de gevoeligheid voor ruis te verminderen De L(x,y) van een afbeelding met pixelintensiteitswaarde l(x,y) wordt gegevens door de volgende functie:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

De meest gebruikte kernels zijn als volgt:

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Het grootste voordeel van deze methode is het vinden van de juiste positie van de rand en hoe groot deze zijn, het is zeer accuraat tegenover andere methoden. Het nadeel is dat het zeer gevoelig is, dus je moet er een guassian filter overheen doen, dit heeft een groot (nadelig) effect op de uitkomst.

De input van de methode is een grayscale image, de meeste implementaties hebben ook een grayscale image nodig die al gefilterd is met een guassian filter.

Canny

Canny edge detection is een techniek die veel wordt gebruikt binnen computer vision. Het gebruikt een multi-staged algorithm die kan opgedeeld worden in 4 stappen volgens OpenCV:

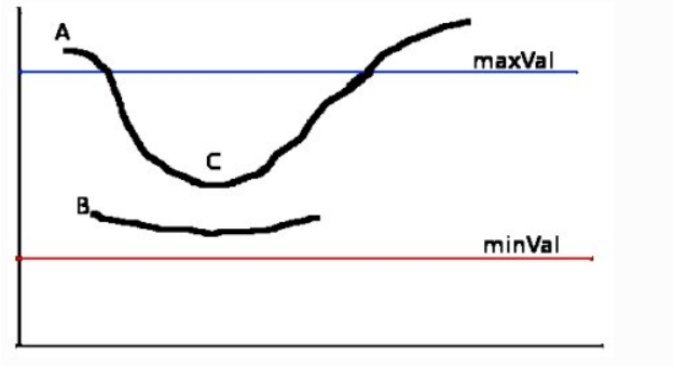
1. Toepassen van Gaussian filter, dit is voor het filteren van ruis vaak wordt er een 5x5 Gaussian filter gebruikt.

2. Het vinden van de intensiteit van het kleurverloop, hierdoor worden edges makkelijker gevonden door de intensiteit die rondom de randen wordt gefilterd. Met deze formules kan de rand gradient en de richting van elke pixel gevonden worden:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$
$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

3. Toepassen van Non-maximum supression, het is een algoritme dat de grootste randen zoekt, vervolgens worden de grote randen verdunt.

4. Hysteresis thresholding, om de laatste paar fouten en ruis te filteren wordt er gebruikt gemaakt van een doubel threshold. Een threshold voor lage waardes en een threshold voor hoge waardes. Ondanks dat C buiten het veld ligt wordt het alsnog beschouwt als edge aangezien het gelinkt is aan het gedeelte van A. Hierin zie je dat B niet gelinkt is aan waardes boven de MaxVal dus wordt B ook weggefilterd.



Voor- en nadelen

Robert's Cross ->
Voordeel: Makkelijk & Snel.
Nadeel: Door voordelen is deze redelijk inaccuraat.

Sobel ->
Voordeel: Simpel, redelijk accuraat op randen en de bijhorende oriëntatie.
Nadeel: Gevoelig voor ruis, bij teveel ruis heel inaccuraat.

Prewit ->
Prewit komt heel erg overeen met Sobel, maar uit reeds verricht onderzoek kunnen wij concluderen dat Sobel beter werkt.

"Edge detection efficiency of Sobel operator is better than Prewitt and Robert operator. Edges detected by Sobel operator based edge detection system are thick. For application like road lane detection in smart cars where fast processing is required, Robert operator based edge detection system can be use. For system like bar code reader, Sobel operator based edge detection system can be use to get sharp edges."

Laplacian of Guassian ->
Voordeel: Smalle edges die gevonden worden.
Nadeel: Benodigde Guassian Filter zorgt voor veel impact op het resultaat.

Canny ->
Voordeel: Juiste randen kunnen worden gedetecteerd, Deze kan ook beter om gaan met ruis dan met andere methodes,
Nadeel: Heeft meer resources nodig, slomer dan andere methodes, *False Zero Crossing error*.

1.4. Keuze

Onze keuze is gevallen op de Canny edge methode. Wij kiezen voor deze methode, omdat wij een zo accuraat mogelijke edge detection willen creëren met zo min mogelijk fouten. Voor ons systeem is de snelheid en het geheugengebruik van belang, maar niet essentieel

Het implementeren van Canny edge lijkt ons uitdagend, maar vooral leerzaam. Daarnaast is deze methode makkelijk op te splitsen in verschillende onderdelen, waardoor er efficiënter gewerkt kan worden. Denk stappen zoals een filter toevoegen, intensiteit van de gradiënt, thresholding etc.

1.5. Implementatie

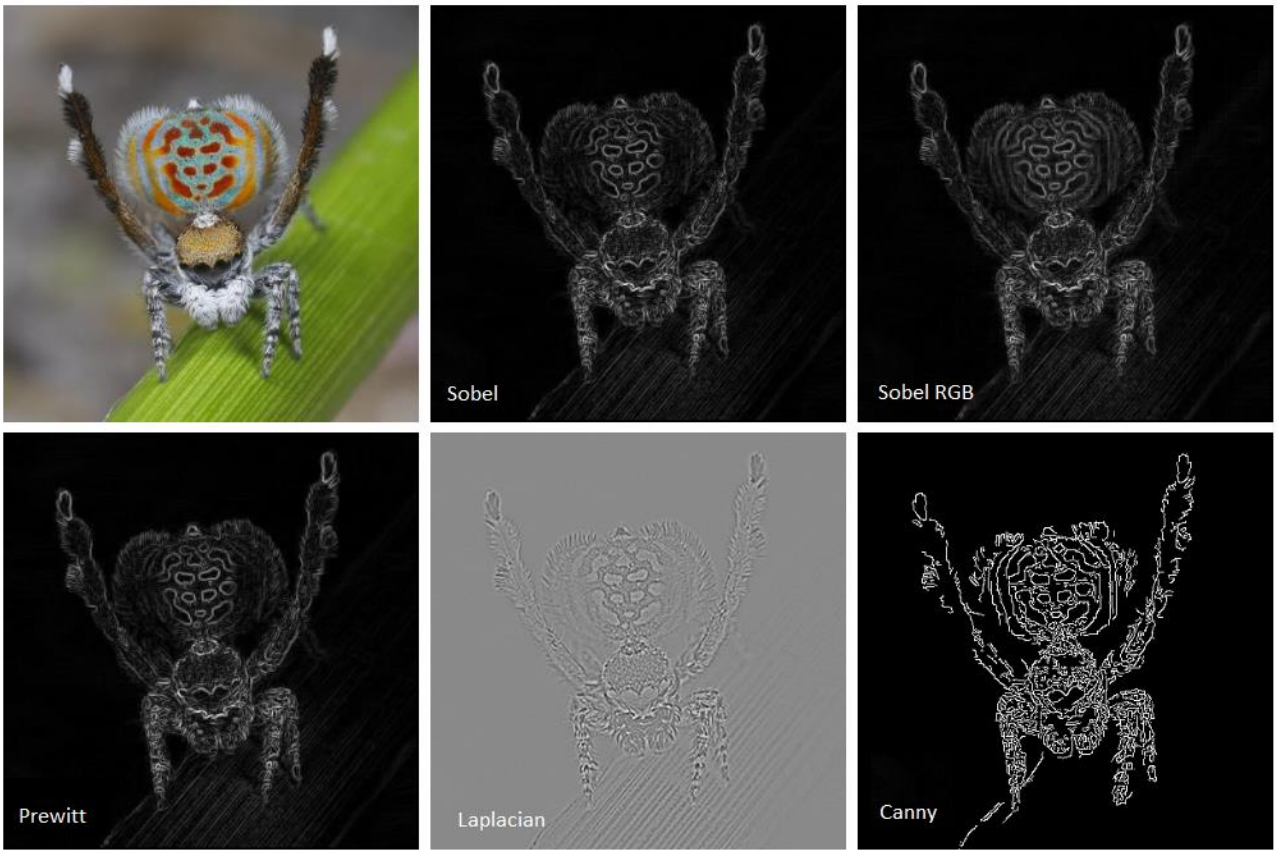
Je geeft aan hoe deze keuze is geimplementeerd in de code

1.6. Evaluatie

Wij willen aantonen dat Canny edge beter werkt dan het huidige edge detection systeem.

Dit is relevant voor ons doel, omdat je duidelijk aspecten van het gezicht wilt herkennen, zoals een neus of mond. Wij gaan dit aantonen door beide algoritmes op een groot aantal verschillende gezichten te testen en te kijken welk algoritme het meest accuraat is.

Wij gaan vooral letten op hoe goed de algoritmes de belangrijke delen van het gezicht laten zien (ogen, mond, neus). Wij verwachten dat het door ons gekozen algoritme deze delen vaker en duidelijker laat zien dan het huidige algoritme. Ook wordt er gelet op hoe accuraat de algoritmes zijn in situaties met weinig licht. Hier verwachten wij dat het door ons gekozen algoritme accurater is dan het huidige algoritme.


Sobel
Sobel RGB
Prewitt
Laplacian
Canny