Implementatieplan edge detection

Rick van Mourik

Noah Titarsole

21-02-2020

# Doel

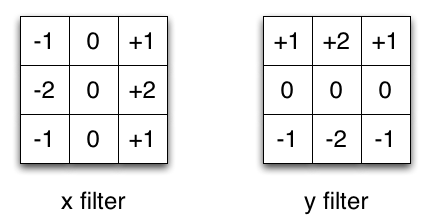
Wij willen ons huidige edge detection systeem verbeteren. Het edge detection systeem is een onderdeel van een beveiligingssysteem voor bedrijven met afgeschermde gebieden. Het doel van de edge detection is de gezichtskenmerken van een persoon duidelijk te maken, zodat deze vervolgens verwerkt kunnen worden. Als deze kenmerken zijn verwerkt is het einddoel om dezelfde persoon te herkennen, zodat deze persoon in de afgeschermde afdeling kan, mits deze persoon hiervoor de juiste machtigingen heeft. De gezichtskenmerken die we duidelijk willen maken door middel van edge detection zijn: de ogen, de mond en de neus. We gebruiken deze kenmerken, omdat deze niet snel zullen veranderen in tegenstelling tot het haar van een persoon en de wangen. Dit edge detection systeem moet zeer accuraat zijn om de veiligheid te waarborgen. De snelheid van het systeem is van minder belang. Ons huidig systeem is vrij snel, maar maakt veel fouten waardoor men meerdere malen de software moet gebruiken. Dit willen wij verhelpen door minder te focussen op de snelheid en geheugen intensiteit en in de plaats daarvan juist meer op de accuraatheid te focussen.

# Methoden

Wij gebruiken diverse andere studies om de verschillende edge detection methodes met elkaar te vergelijken. Wij vergelijken de verschillende methode op basis van snelheid, geheugen gebruik, robuustheid en volledigheid. De methodes die we gaan vergelijken zijn Sobel, Prewitt, Robert’s cross, Laplacian, laplacian of gaussian, Canny.

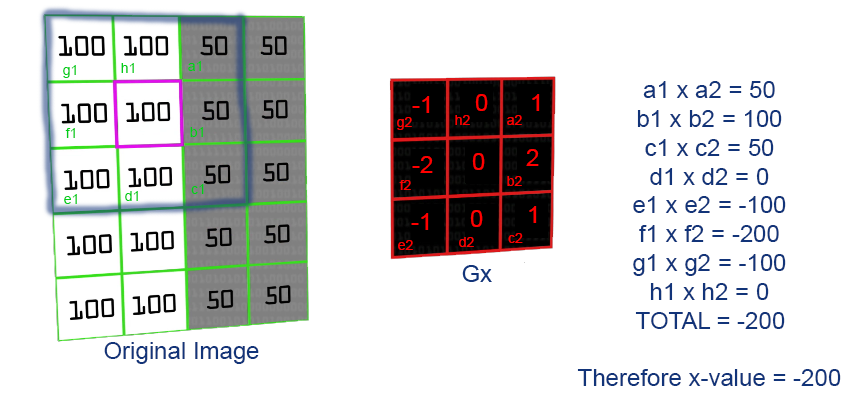
## Sobel

De sobel operator gebruikt het verschil in intensiteit tussen pixels om een edge te detecteren.

Het verschil in intensiteit wordt voor elke pixel berekent in zowel de x als y-as. Het verschil wordt berekend met behulp van de volgende filters:

bron: <https://blog.saush.com/2011/04/20/edge-detection-with-the-sobel-operator-in-ruby/>

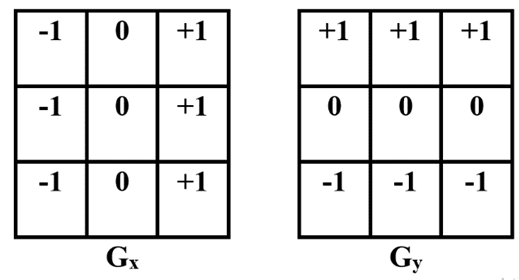
Met deze filters worden 2 nieuwe afbeeldingen die de kernel convolutie geven. Als deze convolutie niet 0 is, dan wordt de pixel gezien als een pixel van een edge. Hier een voorbeeld van zo’n berekening voor een pixel in de x-as:



bron: <https://www.projectrhea.org/rhea/index.php/An_Implementation_of_Sobel_Edge_Detection>

## Prewitt

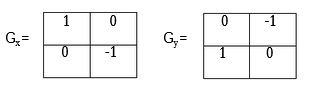
Prewitt gebruikt hetzelfde principe als Sobel, Prewitt maakt ook gebruik van een masker voor de x en y-as en berekent daarmee het verschil in intensiteit tussen de pixels. Wat deze operators verschillend maakt zijn de maskers. Sobel maakt een onderscheidt tussen pixels die direct op de te analyseren pixel sluiten en de pixels die diagonaal op de pixel aansluiten, Prewitt doet die niet. Dit leidt tot het volgende masker voor Prewitt:



bron: <https://www.researchgate.net/figure/Masks-for-the-Prewitt-gradient-edge-detector-The-Laplacian-operator-is-based-on-second_fig3_317754223>

## Robert’s cross

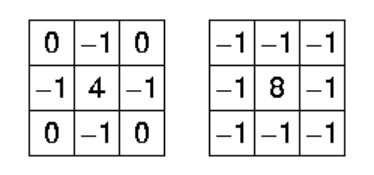
Robert’s cross gebruikt net als sobel het verschil in intensiteit tussen pixels, maar kijkt alleen diagonaal. Omdat er gebruik wordt gemaakt van 2x2 kernels en er dus geen middelpunt in deze kernels zit is de uitkomst voor een geïnterpoleerd punt. Het maakt gebruik van deze 2 kernels:



<https://www.researchgate.net/figure/Masks-used-by-Roberts-Operator_tbl1_303142762>

## Laplacian

Laplacian gebruikt, in tegenstelling tot Sobel, Prewitt en Robert’s cross, maar 1 kernel om zijn berekeningen te doen. Dit is omdat de tweede afgeleide in 1 keer wordt uitgerekend. 2 veel gebruikte kernels zijn:

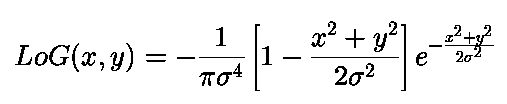


bron:

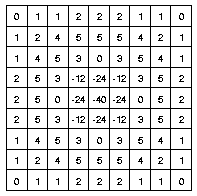
<https://medium.com/@nikatsanka/comparing-edge-detection-methods-638a2919476e>

## Laplacian of Gaussian

Laplacian of Gaussian is een samenvoeging van een Gaussian filter om ruis te verminderen en Laplacian. Deze kunnen samengevoegd worden tot 1 kernel om je algoritme sneller te maken. Om deze kernel te berekenen gebruik je deze functie:



De uitkomst bij 𝜎= 1,4:



bron:

<https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm>

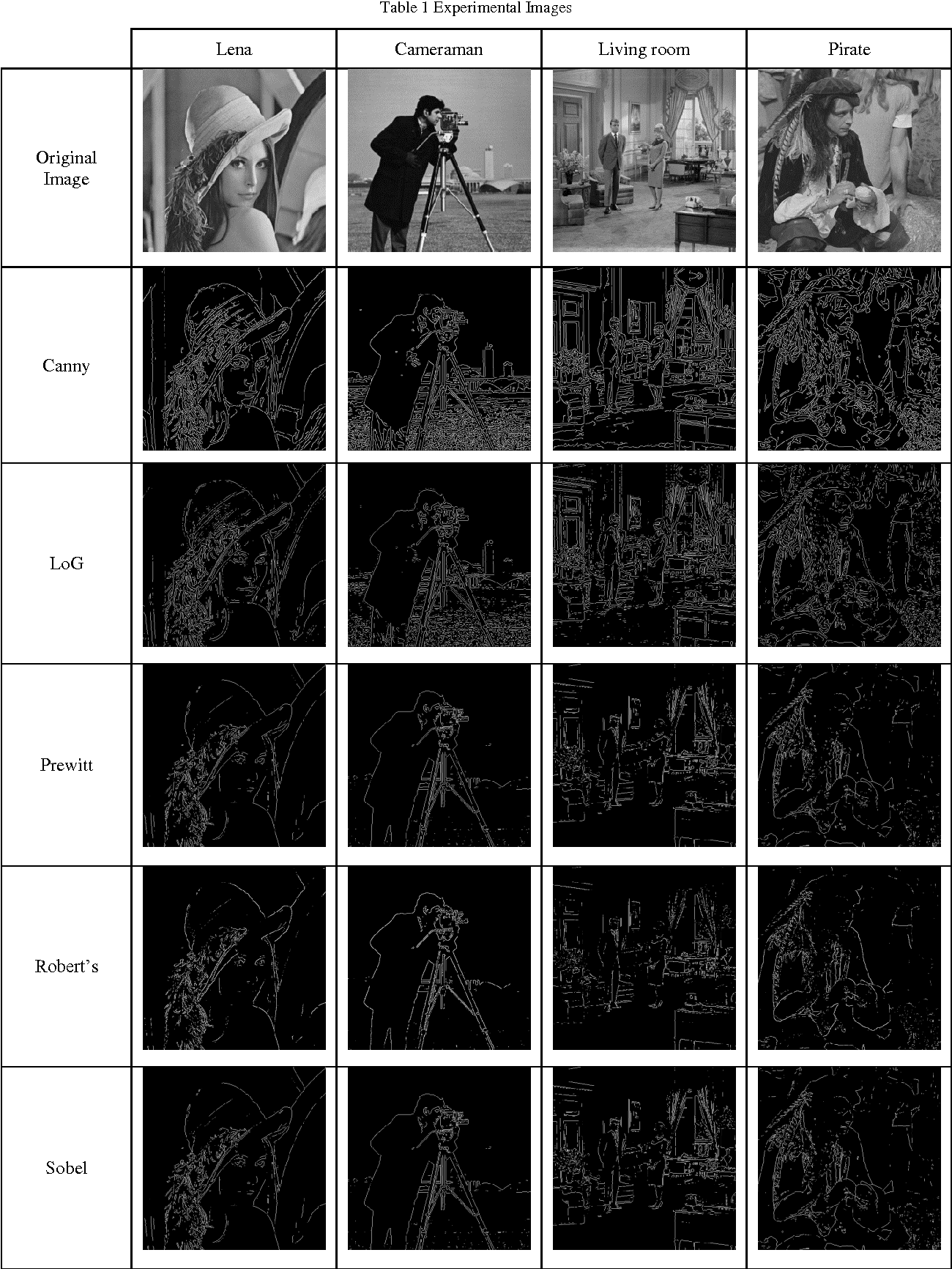
## Canny

Canny heeft de meeste stappen van alle methodes en combineert verschillende methodes. Om door middel van Canny de edge te detecteren wordt eerst een gaussian blur toegepast op de afbeelding om zo de ruis te verminderen. Als de gaussian blur toegepast is wordt hetzelfde masker als sobel gebruikt om de richting van een edge te detecteren.

Met behulp van deze richting wordt vervolgens de rand in de afbeelding afgeleid. Vervolgens worden de pixels bij de rand horen nog eens uitgedund, zodat je een dunne lijn krijgt. Dit wordt gedaan door de pixels te vergelijken met de andere pixels in de rand richting. Als als een pixel waarde lager is dan de andere rand pixels in dezelfde richting, wordt deze pixel niet meer gezien als rand.

Ten slotte is er nog een laatste check om in plaats van strepen een vloeiende lijn te krijgen. Hierbij worden 2 thresholds gebruikt, T1 en T2. Als een pixel boven de T1 threshold komt dan wordt deze gezien als een pixel van de lijn. Vervolgens wordt er naar de omringende pixels gekeken of deze boven T2 zitten, als dit het geval is dan worden zij ook onderdeel van de rand. Dit leidt tot een vloeiende en dunne lijn, als de thresholds correct zijn gekozen.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Sobel | Prewitt | Robert’s cross | Laplacian | Laplacian of Gaussian | Canny |
| snelheid | ★★ | ★★ | ★★★ | ★★★★★ | ★★★★ | ★ |
| geheugen  gebruik | ★★ | ★★ | ★★ | ★★★★★ | ★★★★★ | ★ |
| robuustheid | ★★ | ★★ | ★ | ★★★ | ★★★★ | ★★★★★ |
| accuraatheid | ★★ | ★★ | ★★ | ★★★ | ★★★★ | ★★★★★ |



bron: <https://www.semanticscholar.org/paper/Edge-Detection-Operators%3A-Peak-Signal-to-Noise-Poobathy-Chezian/ebada25e647fd0501f5b2386b3f17b210cecb4af>

## Voor- en nadelen

De voor- en nadelen van de verschillende methodes op een rij.

Sobel, prewitt en robert’s cross zijn alle 3 relatief eenvoudige algoritmes waardoor ze snel zijn en weinig geheugen gebruiken. Deze algoritmes zijn echter minder volledig en niet erg robuust. Bij ruis werken deze algoritmes niet goed en ze zijn ook te sensitief, waardoor er te veel wordt gezien als een edge.

Laplacian Is net zoals de vorige algoritmes gevoelig voor ruis, maar is wel sneller dan de vorige, omdat dit algoritme gebruik maakt van separatie van convolutie kan de nieuwe matrix in m\*n2 berekend worden in plaats van m2\*n2waar m de originele matrix is.

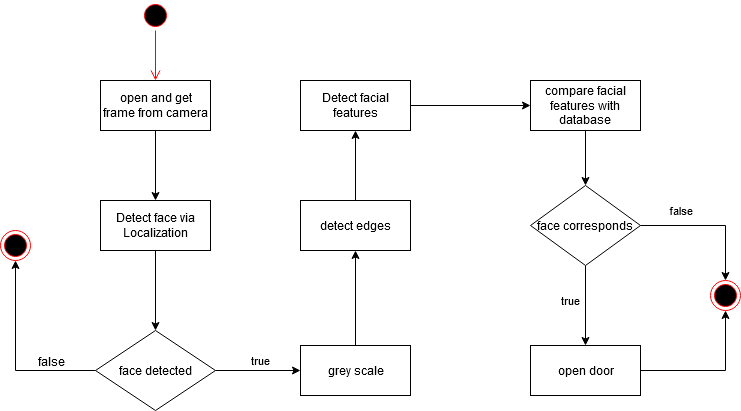
Canny heeft als voordelen dat het algoritme zelf de ruis vermindert, het algoritme heeft een ingebouwde error marge om de edges smoother te maken en het algoritme volgt de edge zeer nauw. De nadelen van dit algoritme is dat het langzaam is vanwege de vele en complexe berekeningen.

# 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 weinig mogelijk fouten. Voor ons systeem is de snelheid en het geheugengebruik van weinig belang, aangezien het systeem niet bedoeld is om een continue stroom van afbeeldingen te analyseren en het systeem hoort maar 1 gezicht elke 10-20 seconde te analyseren op een piektijd. Het systeem zit voornamelijk niets te doen. Het implementeren van Canny edge lijkt ons uitdagend, maar wel goed te doen. Daarnaast is deze methode makkelijk op te splitsen in verschillende onderdelen, waardoor er efficiënter gewerkt kan worden.

# Implementatie

De Canny edge detection zal ter vervanging dienen van het huidige edge detection systeem in ons beveiligingssysteem. Het huidige edge detection systeem maakt te veel fouten naar onze mening en is een belangrijke reden voor de fouten die het programma maakt voor het gezichtsherkennen. Deze fouten leiden ertoe dat de klant zijn werknemers vaak meerdere malen ons programma moeten gebruiken voordat zij toegang krijgen tot de afdeling waar ze heen willen. Wij verwachten dat Canny, ondanks dat het langzamer is per gebruik, ons beveiligingssysteem versneld aangezien het programma in de huidige staat vaak meerdere malen gebruikt moet worden.



# Evaluatie

Wij willen aantonen dat de door ons gekozen methode (Canny edge) beter werkt dan de huidige edge detection. Dit is relevant voor ons doel, een beveiligingssysteem, omdat je niet per ongeluk de verkeerde persoon binnen wilt laten of buiten wilt houden. 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.

# Bibliografie

Tsankashivii N. 21-02-2020. Comparing Edge Detection Methods. Geraadpleegd van <https://medium.com/@nikatsanka/comparing-edge-detection-methods-638a2919476e>

Maini R & Dr. Aggarwal H. 21-02-2020. Study and Comparison of Various Image Edge Detection Techniques. Geraadpleegd van <https://www.cscjournals.org/manuscript/Journals/IJIP/Volume3/Issue1/IJIP-15.pdf>

Argialas D. P. & Mavrantza O. D. 21-02-2020. COMPARISON OF EDGE DETECTION AND HOUGH TRANSFORM TECHNIQUES FOR THE EXTRACTION OF GEOLOGIC FEATURES. Geraadpleegd van

<https://pdfs.semanticscholar.org/3a0e/a78f7508356f6130e506df0f1010ea6f55b5.pdf>

Ashish K. 08-03-2020. Understanding Edge Detection (Sobel Operator). Geraadpleegd van

<https://medium.com/datadriveninvestor/understanding-edge-detection-sobel-operator-2aada303b900>

Green B. 12-03-2020. Canny edge detection tutorial. Geraadpleegd van <https://www.sites.google.com/site/setiawanhadi2/1CannyEdgeDetectionTutorial.pdf>

Shrivakshan G.T. & Dr. Chandrasekar C. 13-03-2020. A Comparison of various Edge Detection Techniques used in Image Processing. Geraadpleegd van

<https://www.researchgate.net/publication/303142762_A_Comparison_of_various_Edge_Detection_Techniques_used_in_Image_Processing>