Rick van Mourik

Noah Titarsole

21-02-2020

# Doel

Wij willen een edge detection systeem creëren als 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 het afgeschermde kan. De gezichtskenmerken die we duidelijk willen maken door middel van edge detection zijn: de ogen, de mond en de neus. We willen deze kenmerken gebruiken, omdat deze niet snel zullen veranderen in tegenstelling tot het haar van een persoon. Dit edge detection systeem moet zeer accuraat zijn om de veiligheid te waarborgen. De snelheid van het systeem moet wel in een schappelijke tijd gebeuren voor een hoge beveiligingsinrichting, wij willen dat dit proces minder dan 10 seconde duurt. Het systeem mag wel intensief zijn qua gebruik, want wij verwachten niet dat er veel werknemers tegelijk de scanners zullen gebruiken.

# 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.

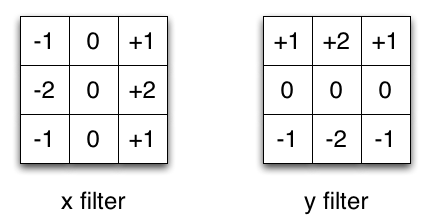
insert working algoritmes here.

Rick: Sobel, prewitt, canny.

noah: robert’s cross, laplacian, laplacian of gaussian

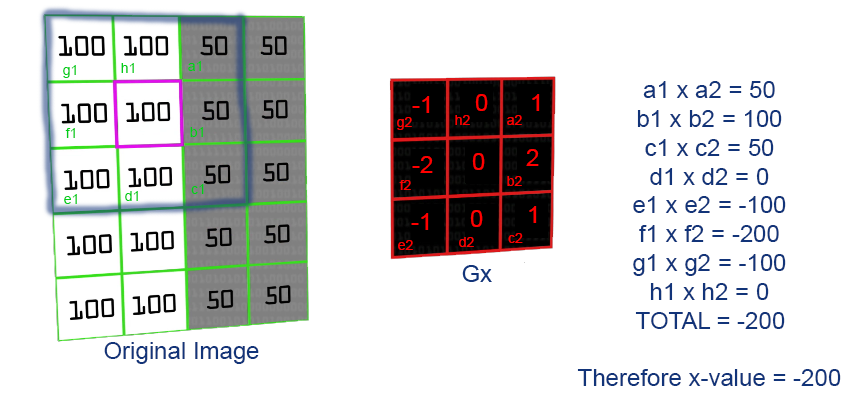
## Sobel

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 berekent met behulp van de volgende filters:

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

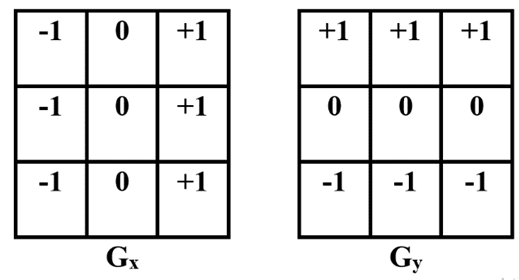
Met deze filters worden 2 nieuwe afbeeldingen die de kernel convolutie geven. Als deze convolutie niet 0 is, dan wordt de pixel gezien als een pixel van een edge. Hier een voorbeeld van zo’n berekening voor een pixel in de x-as:



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

## Prewitt

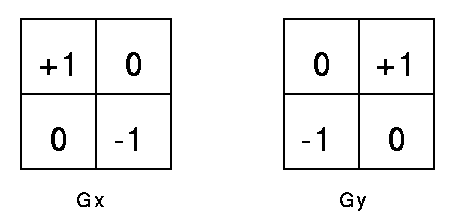
Prewitt gebruikt hetzelfde principe als Sobel, Prewitt maakt ook gebruik van een masker voor de x en y-as en berekent daarmee het verschil in intensiteit tussen de pixels. Wat deze operators verschillend maakt komt door de maskers. Sobel maakt een onderscheidt tussen pixels die direct op de te analyseren pixel sluiten en de pixels die diagonaal op de pixel aansluit, 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. het maakt gebruik van deze 2 kernels:



bron:

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

Omdat er geen middelpunt in deze kernels zit is de uitkomst voor een geïnterpoleerd punt.

bron:

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

## Laplacian

## Laplacian of Gaussian

## Canny

Canny heeft de meeste stappen van alle methodes en combineert verschillende methodes. Om door middel van Canny de edge te detecteren wordt in deze methode 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 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 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 | ★★ | ★★ | ★★ | ★★★ | ★★★★ | ★★★★★ |

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 teveel 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.

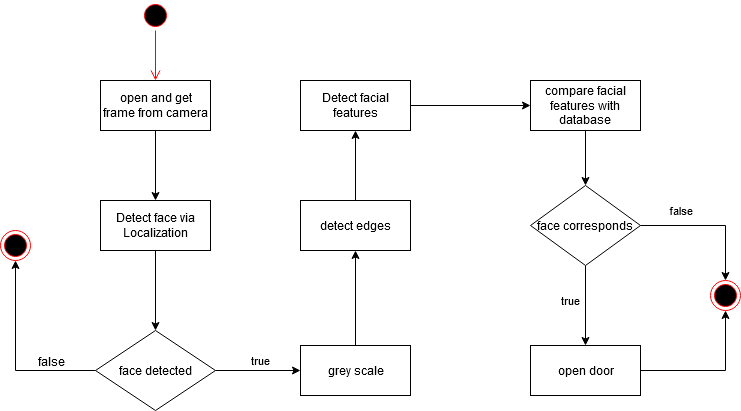
# Keuze

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 niet van belang, aangezien het systeem niet bedoeld is om een continue stroom van afbeeldingen te analyseren, het systeem hoort maar 1 gezicht elke 10-20 seconde te analyseren op een piek tijd. Het systeem zou echter vaak niks 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

# 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>

m2\*n2 + m2\*o2

3x3 mask

m2\*n2 + m2\*o2

3x3 mask

m2\*n2 + m2\*o2

2x2 mask

m\*n2 +