Edge Detection

Jeroen Kok

Stefan

23-02-2017

# 1. Doel

In opdracht van de docent moesten wij een onderdeel kiezen uit het Face Recognition programma, daar onderzoek naar doen en dat dan verbeteren.

Wij hebben gekozen voor Edge-detection omdat wij denken daar het meeste verschil in te kunnen maken. Wij denken dat de edge-detection sneller kan zijn dan het nu is.

Ons doel is de huidige implementatie voor de Edge-detection te vervangen met een algoritme dat sneller is dan degene die nu wordt gebruikt. De betrouwbaarheid van het systeem mag echter niet te veel omlaag gaan. Hiervoor gaan wij onderzoek doen naar een aantal methodes om te kijken welke de beste verbetering in het systeem zal brengen.

Wij zullen daarna een aantal methodes uitkiezen die wij vervolgens grondig zullen testen op basis van verschillende attributen, zoals de tijd die het kost en betrouwbaarheid, om te kijken welke de beste is.

# 2. Methoden

Voor dit onderzoek hebben wij zoveel mogelijk methodes proberen te vinden en hebben wij kort gekeken waar welke goed voor is. De methode die wij hebben gevonden zijn:

* Canny
* Deriche
* Sobel
* Prewitt
* Robert Cross
* Hough Transformatie
* Laplacian
* Basic Gradient Detection

Canny & Deriche

Canny is een methode die veel wordt gebruikt. Maar het is niet wat wij zoeken; het is meer een stappenplan dat als kernel verwijst naar Sobel, Prewiit, ect. Het is zelf geen edge-detection, maar een systeem voor ‘goede’ edge-detection. Ditzelfde geldt voor Deriche.

Hough Transformation

Hough Transformation ziet er voor ons erg ingewikkeld uit en kost ook veel tijd dus deze is niet geschikt

Sobel, Prewitt, Roberts Cross, laplacian en simplistic

Deze methode zijn kernels waar in Canny en Deriche naar worden verwezen. Als je een van deze kernels over een plaatje haalt, dan krijg je een plaatje terug met alle edges. Hoewel ze allemaal kernels zijn; gaat het om de getallen die in de kernels staan en hoe groot ze zijn. En die leveren elk een ander resultaat. Als een kernel groter is zal deze trager werken maar het resultaat zal niet per se slechter zijn, Daarom lijkt het ons slim om meer op de kleinere kernels, zoals Basic Gradient Detection en Roberts te focussen.

Mexican Hat(laplacian)

Er is ook nog Maxican Hat. Dit is een Laplacian kernel met tegelijkertijd een gaussian ruis filter. Deze moeten echter erg groot zijn (groter dan 3x3) en niet deelbaar in 2 kleine kernels. Wiij denken dat hij daarom niet geschikt zal zijn.

3. Sobel [2][5]

- Grof bij hoge frequenties

~ Edge detection op basis van gradiënt

4. Prewitt [5]

+ Kernel deelbaar (m\*n2)

+ Goedkoop en simpel

~ Edge detection op basis van gradiënt

- Erg grof

5. Roberts Cross [5]

+ Simpel

~ Edge detection op basis van gradiënt

6. Hough Transformation [1][7]

- Zeer ingewikkeld (at the time of writing)

- Zeer duur (Kost veel tijd)

- LED can niet worden opgedeelt in meerdere kernels. Daardoor is de de orde M2\*N2 ipv M\*N2, wat erg kostbaar kan zijn bij grotere kernels.

8. Basic Gradient Detection [11]

+ Erg simpel, omdat er maar weinig moet worden gedaan.

- Erg grof

- Gevoelig voor ruis (omdat er geen filter in zit)

# 3. Keuze

Wij hebben voor Basic Gradient Detection (BGD) gekozen. Omdat wij BGD erg simpel vinden, denken wij dat dit de beste methode voor ons doel is. Door de simpliciteit denken wij dat het een snellere methode zal zijn dan bijvoorbeeld Robert Cross, Prewitt of Sobel.

Maar omdat het hier om enkel de inhoud van de kernels gaat, willen wij ook Robert, Prewitt en Sobel testen om te zien of de grootte van de kernels verschil maakt in de snelheid en kwaliteit van het resultaat.

Wij kijken in eerste instantie niet naar Canny of Deriche omdat zij zelf geen edge detection zijn, maar kernels van anderen gebruiken.

De hough transformatie lijken ons te complexe methoden om uit te zoeken in verband met de tijd die wij er voor hebben.

# 4. Implementatie

} De code zal binnen de bestaande software worden geïmplementeerd zonder het gebruik van nieuwe (hulp) klasse. In StudentPreProcessor.cpp binnen de functie “stepEdgeDetection”.

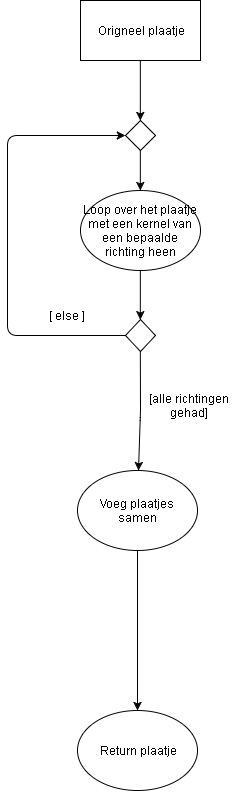
Wij hebben besloten alleen naar de kernels zelf van elke methoden te kijken.

} We nemen elke kernel en gaan daarmee een bepaald aantal keer over een originele plaatje. Daarna voegen we de resulterende plaatjes bij elkaar en geven die aan het programma terug..

voor het BGD moet er meerdere keren over het originele plaatje worden gegaan, omdat elk kernel maar voor één richting geschikt is. Als je dat niet doet krijg je niet alle edges terug. Bijvoorbeeld alleen die van zwart naar wit gaan, maar niet van wit naar zwart.

Je kunt de kernels ook niet samenvoegen omdat dit dan de edge-detecting eigenschappen van de kernel ongedaan maakt.

Wij nemen aan dat BGD het snelst zal zijn. Maar dat het zwakkere resultaten levert dan Roberts, Prewitt en Sobel.



## BGD (1x2)

x

|  |  |
| --- | --- |
| 1 | -1 |

y

|  |
| --- |
| 1 |
| -1 |

## Roberts cross (2x2)

|  |  |
| --- | --- |
| 1 | 0 |
| 0 | -1 |

|  |  |
| --- | --- |
| 0 | 1 |
| -1 | 0 |

## Laplacian

x

|  |  |  |
| --- | --- | --- |
| 1 | -2 | 1 |

y

|  |
| --- |
| 1 |
| -2 |
| 1 |

## Prewitt (3x3)

x

|  |  |  |
| --- | --- | --- |
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

y

|  |  |  |
| --- | --- | --- |
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

## Sobel (3x3)

x

|  |  |  |
| --- | --- | --- |
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

y

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

# 5. Evaluatie

} Doel : Kijken of BGD het efficienste kernel is  
} Aantoenen dat BGD het snelste kernel is, zonder te slechte resultaten te leveren.

} Relevantie: In de praktijk wil je veel verschillende gezichten herkennen in zo min mogelijk tijd

} Wat meten: Hoelang doet het programma er over met de verschillende kernels.

} Hoe: dat stukje code timen

} Wij denken dat BGD sneller zal werken dan de andere kernels.Q. Maar niet zoveel dat het de gezichtsherkenning zal vermoeilijken.

Bronnen

1. <http://users.polytech.unice.fr/~lingrand/Ens/up/Lesson7and8-segmentation.pdf>
2. <https://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator>
3. <https://pdfs.semanticscholar.org/55e6/6333402df1a75664260501522800cf3d26b9.pdf>
4. <http://sites.google.com/site/setiawanhadi2/1CannyEdgeDetectionTutorial.pdf>
5. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.1860&rep=rep1&type=pdf>
6. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.1860&rep=rep1&type=pdf>
7. <http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm>
8. <https://ena.etsmtl.ca/pluginfile.php/59679/mod_resource/content/0/Deriche%20Wikipedia.pdf>
9. <http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/laplacian.html>
10. <http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm>
11. <https://www.cis.rit.edu/people/faculty/rhody/EdgeDetection.htm>dia