# Practicum Vision: Face Recognition

## Practicum onderdeel #3:Edge detection

**Auteurs:**
Alexander Hustinx
Rolf Smit

**Versie:**
1.0

**Cursus:**
Vision (TCTI-V2VISN1-13)

# Practicum onderdeel #3: Edge detection

In het onderdeel #3 zal er gewerkt worden aan een *edge detection* en *thresholding* methode voor de pre-processing van de *face recogntion* applicatie.
Na het toevoegen van de door de student gemaakte code zal het via de GUI getest kunnen worden en zal het moeten resulteren in een vergelijkbaar of beter resultaat dan verkregen met de aangeleverde code.

In dit document zal er duidelijk gemaakt worden wat de leerdoelen van dit onderdeel van het practicum zijn en hoe deze behaald dienen te worden. Daarna zal er beschreven worden hoe het practicum beoordeeld gaat worden en hoe en wanneer de student het practicum dient in te leveren. Als laatst wordt de opdracht en hoe de student te werk moet gaan in detail beschreven.

**Leerdoelen**
De leerdoelen voor dit onderdeel van het practicum zijn het leren werken met *kernel*-based technieken, ontdekken hoe *edge detection* in zijn gang gaat, leren hoe en waar *edge detection* bruikbaar voor kan zijn en het implementeren van statische 2D software voor een vision taak. Deze leerdoelen zullen behaald worden door het toepassen van de volgende algoritmen en visiontechnieken:

- Het maken en implementeren van een *edge detection* methode;
- Het maken van een afweging tussen de verschillende bruikbare *edge detection* algoritmen;
- Het analyseren van de resultaten van het gebruikte *edge detection* algoritme.

Hoe deze punten helpen met het behalen van de genoemde leerdoelen is doordat studenten op zoek moeten naar een *edge detection* algoritme geschikt voor de huidige doelstellingen en scope, ze moeten weten hoe *edge detection* werkt en waar er op gelet moet worden. Ook is *edge detection* een *kernel*-based techniek, dus tijdens het maken hiervan zal de studenten ook leren omgaan met de basis van *kernels*.

**Beoordelings- en inlevermethoden**
Dit onderdeel van het practicum wordt beoordeeld aan de hand van de volgende punten:

- Snelheid;
- Memory efficiency;
- Robuustheid;
- Volledigheid;
- Afweging tussen methoden (implementatieplan).

Hoe meer van deze punten behaald/voldaan zijn, hoe hoger de resulterende beoordeling zal zijn. Dit is met uitzondering van plagiaat. Plagiaat resulteert in een NVD of een 1.

Er dienen een aantal (zelf te bepalen, maar meer dan één) meetrapporten te worden gemaakt die aantonen dat de code en oplossingen daadwerkelijk werken. Daarbij kan gedacht worden aan snelheidsmetingen, geheugengebruik, vergelijk met de ‘base’-implementaties, etc. Deze rapporten dienen in PDF formaat te worden opgeslagen in de ‘meetrapporten’ folder van de repository. Bronbestanden voor deze rapporten kunnen in de ‘working’ folder worden opgeslagen, NIET in de ‘meetrapporten’ hoofd-folder.

*Vergeet niet om het implementatieplan te schrijven en deze in de folder ‘implementatieplan’ op te slaan!!*

**Practicumopdracht(en)**
In deze paragraaf wordt behandeld wat de practicumopdracht precies is en wat er van de student verwacht wordt.

Dit onderdeel van het practicum bestaat uit twee opdrachten:

1. Het maken van een *edge detection* methode;
2. Het maken van een thresholding methode.

Deze opdrachten worden hieronder beschreven.

**1. Het maken van een *edge detection* methode**
*Edge detection* is een erg belangrijk onderdeel van de *pre-processing*. Het zorgt ervoor dat de *object recognition* veel soepeler verloopt. In de applicatie wordt op dit moment gebruik gemaakt van de volgende (9x9) *Laplacian operator* met als *kernel*:

$$kernel = \begin{matrix} & & & 1 & 1 & 1 & & & \\ & 0 & & 1 & 1 & 1 & & 0 & \\ & & & 1 & 1 & 1 & & & \\ 1 & 1 & 1 & -4 & -4 & -4 & 1 & 1 & 1 \\ 1 & 1 & 1 & -4 & -4 & -4 & 1 & 1 & 1 \\ 1 & 1 & 1 & -4 & -4 & -4 & 1 & 1 & 1 \\ & & & 1 & 1 & 1 & & & \\ & 0 & & 1 & 1 & 1 & & 0 & \\ & & & 1 & 1 & 1 & & & \end{matrix}$$

Dit is in principe vergelijkbaar met de volgende (3x3) *kernel*, maar alleen groter:

$$kernel = \begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Voor deze opdracht zal de student een klein onderzoekje doen naar verschillende bestaande *edge detection* algoritmen en vervolgens een onderbouwde keuze maken voor welke methode de student kiest. In Figuur 1 volgt een klein voorbeeld van het (rauwe) resultaat van de bovengenoemde *edge detection*.

**Figuur 1.(links) Intensity image, (rechts) resultaat van de laplacian operator**

Let op dat dit goed beschreven staat in het implementatieplan dat deze week opgeleverd dient te worden!

**2. Het maken van een thresholding methode**

Omdat *edge detection* ook vaak gecombineerd wordt met *thresholding* zal dat ook in dit onderdeel van het practicum behandeld worden. In dit geval zal het alleen gaan om de *thresholding* van de *pre-processing*. Dit is een van de simpelere vormen van *thresholding*, maar er kan veel meer mee gedaan worden.

In deze opdracht zal de student een basis *thresholding* methode maken. Dit houdt in dat als een pixel een hogere waarde heeft dan de *threshold* (T) dan wordt hij wit, anders wordt hij zwart. Op deze manier kan er een binaire afbeelding gemaakt worden. In de huidige toepassing is de gebruikte *threshold*:

$$T = 220$$

Let op in de huidige toepassing van de applicatie wordt de achtergrond wit en de voorgrond zwart, om het wat 'overzichtelijker' te maken. Dit komt doordat het plaatje geinvert is (ook door middel van *thresholding*).

In Figuur 2 volgt een klein voorbeeld van het (rauwe) *edge detection* plaatje naar het resultaat van de bovengenoemde *thresholding*.

**Figuur 2. (links) rauwe *edge detection* plaatje, (rechts) resultaat *thresholding*.**

Extra: Zoals eerder vernoemd is, de *thresholding* kan beter. Dit kan bijvoorbeeld door de de waarde T dynamisch te maken. Dan spreken we over *dynamic thresholding*. Verander de bovenstaande *thresholding* methode in een dynamische methode. Het is belangrijk dat de data die eruit komt nogsteeds gebruikt kan worden in de rest van de stappen, dus wil je niet (te) veel ruis, en is het met name belangrijk dat de neuslijnen (vrij dik) gevonden worden!