**1.**    **Implementatieplan Edgedetection**

**1.1.**                   **Namen en datum**

**Student: Matthijs Koelewijn, Erik de Zeeuw**

**Datum: 6-3-2020**

**1.2.**                   **Doel**

Het doel van deze implementatie is het vinden van de zogenoemde randjes in afbeeldingen. Ook wel “Edge detection” genoemd.

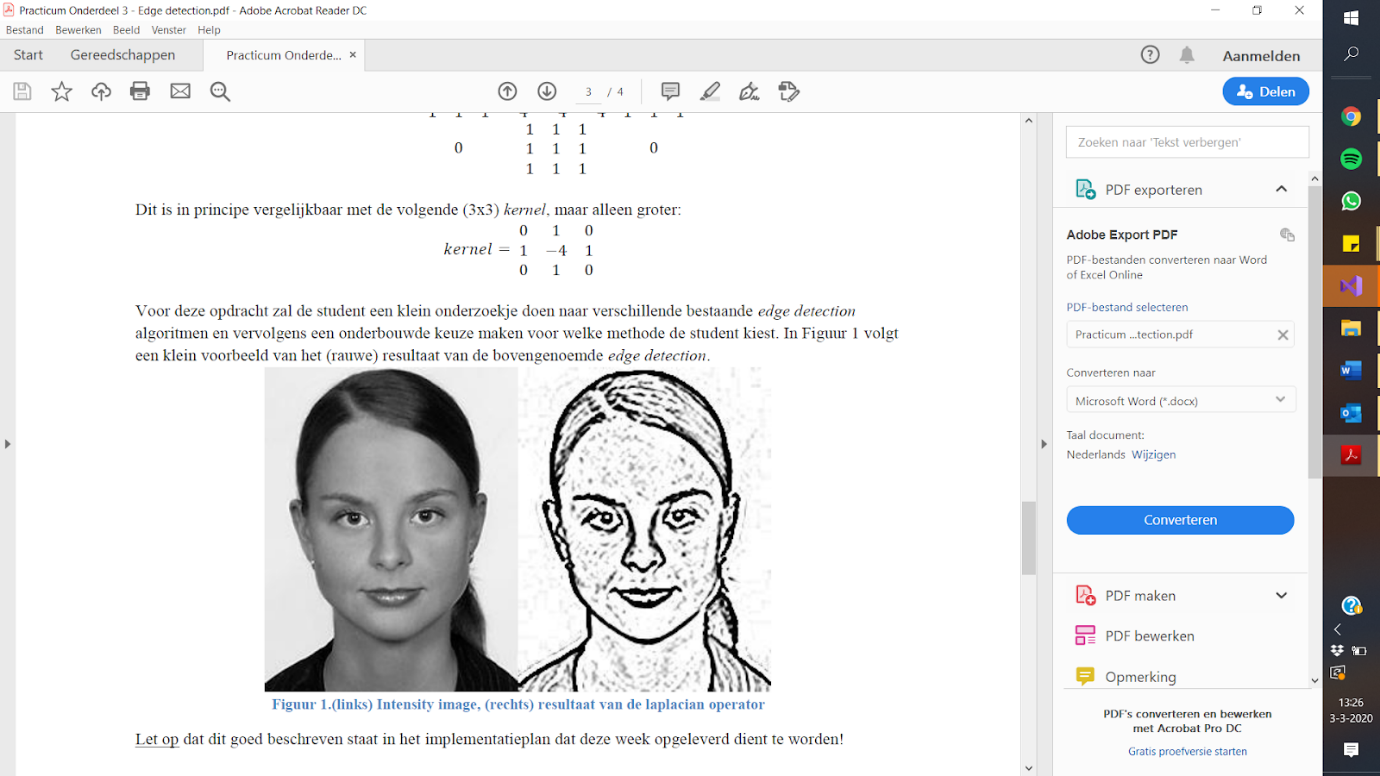
Edge detection is een erg belangrijk onderdeel van de pre-processing. Het zorgt ervoor dat de object recognition veel soepeler verloopt.

**1.3.**                   **Methoden**

Om een goede edge detection te krijgen moet je gebruik maken van high pass filters.

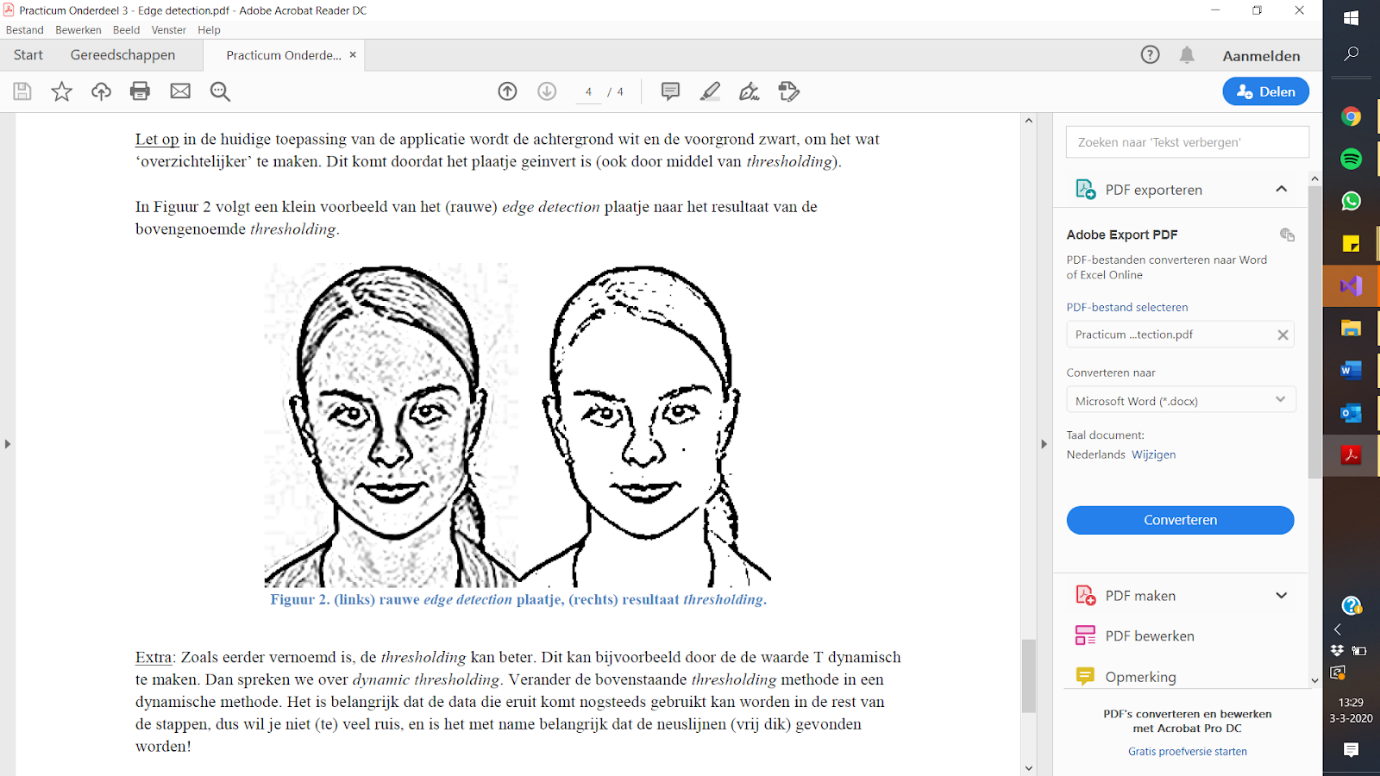
Hiervoor hebben we verschillende informatie gevonden om dit te kunnen gebruiken:

Het ideaal is dat we deze randjes krijgen te zien na een van deze filters:



Voor Na

Nadat we deze filters hebben toegepast willen we alle losse pixels die onbelangrijk zijn voor ons weg halen, dit doen we door middel van thresholding:



Voor Na

Voor dit ideaal zijn uiteraard verschillende methodes te vinden.

Je kunt een van de verschillende Laplacian filters toepassen.

De algemene methode om randjes te vinden is door een kernel te leggen over elke pixel in een plaatje. De kernel gebruik je dan om een nieuwe waarde te berekenen door middel van de pixels die in de buurt liggen. De kernels kunnen verschillende groottes hebben en verschillende waardes, die verschillen per functionaliteit.

Voor Laplacian ziet de kernel er zo uit:

Deze kernel is Laplacian(9x9).

[0,0,0,1,1,1,0,0,0]

[0,0,0,1,1,1,0,0,0]

[0,0,0,1,1,1,0,0,0]

[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]

[0,0,0,1,1,1,0,0,0]

[0,0,0,1,1,1,0,0,0]

[0,0,0,1,1,1,0,0,0]

Deze kernel is Laplacian(3x3).

[0,1,0]

[1,-4,1]

[0,1,0]

Je kunt ook een Sobel filter toepassen, dat is dus eigenlijk twee filters toepassen, een horizontale en een verticale en dan in totaal de wortel nemen van de som van de kwadraten van de horizontale en verticale waarde.

Horizontaal:

 [-1, 0, 1]

 [-2, 0, 2]

 [-1, 0, 1]

Verticaal:

 [-1, -2, -1]

 [0, 0, 0]

 [1, 2, 1]

Of een Prewitt filter, dat is vergelijkbaar met de Sobel filter alleen zijn de getallen anders:

Horizontaal:

 [-1, 0, 1]

 [-1, 0, 1]

 [-1, 0, 1]

Verticaal:

 [-1, -1, -1]

 [0, 0, 0]

 [1, 1, 1]

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

**1.4.**                   **Keuze**

De methode die we gaan implementeren is een Prewitt filter. Dat gaan we doen omdat het een klein filter is, dus dat kost minder tijd dan een groot filter. Het is ook een symmetrisch filter die we dus kunnen optimaliseren dus dat komt de complexiteit ook ten goede.

**1.5.**                   **Implementatie**

We gaan deze keuze implementeren door twee verschillende functies: stepEdgeDetection en stepThresholding. De eerste functie wordt gebruikt om een afbeelding te krijgen waarin de edges te zien zijn met grijswaardes. De tweede functie gaat ervoor zorgen dat elke pixel of wit of zwart wordt door een threshold waarde.

**1.6.**                   **Evaluatie**

Om de implementatie te testen gaan we 5 datasets elk van 20 afbeeldingen verwerken door het zelfgemaakte algoritme en het default algoritme. Die afbeeldingen laten we dan verder verwerken door het bestaande gezichtsherkenning algoritme. Als hij de gezichten op de default manier en op de nieuwe manier met dezelfde of betere accuraatheid kan herkennen, is het algoritme goed.