# 1. Implementatieplan titel

## 1.1. Namen en datum

Teamlid 1: Lex Ruesink

Teamlid 2: Thomas Theil

Datum: 14/04/2019

## 1.2. Doel

Het doel van de implementatie is om in de lokalisatiefase van het gezichtsherkenningsproces de locatie van de ogen sneller te maken dan de “originele” implementatie binnen het framework. De snellere implementatie mag niet te inaccuraat zijn tegenover het origineel.

## 1.3. Methoden

Er zijn verschillende methodieken om ogen te detecteren in een image:

- Haar-like features om vormen binnen afbeelding te vinden
- Een circulaire Hough-transformatie uitvoeren om de distinctieve ovale vorm van het oog te detecteren
- Uitgaan van de natuurlijke verhoudingen van het menselijk gezicht

## 1.4. Keuze

Voor onze implementatie is gekozen voor het uitgaan van de natuurlijke verhoudingen van het menselijk gezicht. Deze variant is zeer snel (op de cpu) en relatief accuraat. Binnen het framework wat wij moeten gebruiken voor de implementatie krijgen wij alleen een intensiteitsmatrix aangeleverd; voor circulaire Hough-transformaties of Haar-like features is de originele/grijswaarden of edge-detected afbeelding nodig.

## 1.5. Implementatie

Binnen de code worden de volgende stappen ondernomen. Er wordt gestart met de intensiteitsmatrix die wordt aangeleverd vanuit het framework:

1: Er wordt erosion uitgevoerd met de volgende kernel:

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

Het doel van de erosion stap is het verwijderen van noise uit de afbeelding; deze noise zou BFS-proces later in het stappenplan kunnen verstoren.

2: Er wordt dilation uitgevoerd met de volgende kernel:

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

De pixelgroepen die zijn achtergebleven na de erosion stap worden weer naar originele dikte hersteld. Anders zou de rectangle met de ogen in het eindresultaat te klein zijn tegenover de originele afbeelding.

3: Er wordt een rij-scorelijst opgebouwd. De score wordt berekend in de volgende subset van de afbeelding:

- X = 0
- Y = Afbeeldingshoogte / 2 + (10% van de hoogte)
- Width = Afbeeldingsbreedte
- Height = Afbeeldingshoogte – Y – headTop (Y-positie van het begin van het hoofd, gegeven door `FEATURE_HEAD_TOP`)

De score wordt berekend door van links naar rechts door de rij heen te lopen. Elke zwarte (gekleurde) pixel in de intensiteitsmatrix hoogt de score van de rij op met 1. De rij met de hoogste score wordt geselecteerd. Dit is in bijna alle gevallen een rij op de wenkbrouw.

Bij de rij (de Y-waarde, niet de score) wordt Afbeeldingshoogte / 2 + (10% van de hoogte) opgeteld.

4: Het linkeroog wordt opgezocht. Op de geselecteerde rij wordt er vanaf links naar het midden gewerkt. Bij de eerste zwarte (gekleurde) pixel wordt BFS uitgevoerd met een maximale diepte van 24. Van deze lijst wordt de minimale X en Y en de maximale X en Y-waarden opgezocht. Hier wordt een rectangle van gemaakt.

5: Het rechteroog wordt opgezocht. Op de geselecteerde rij wordt er vanaf rechts naar het midden gewerkt. Bij de eerste zwarte (gekleurde) pixel wordt BFS uitgevoerd met een maximale diepte van 24. Van deze lijst wordt de minimale X en Y en de maximale X en Y-waarden opgezocht. Hier wordt een rectangle van gemaakt.

6. Beide rectangles worden opgevoerd in het framework als features.

## 1.6. Evaluatie

Na implementatie en metingen hebben wij bevonden dat het algoritme functioneel is en een snelheidswinst bereikt tegenover het oude algoritme. Omdat de snelheidswinst behaald is door bepaalde assumpties te nemen, is het algoritme ook gevoeliger voor foutieve input. Zo zijn er binnen de testset twee inputs waarbij de neus onder de mond wordt geplaatst. Dit is een nadeel van dit algoritme.

Omdat binnen het framework enkel een edge-detected intensity image binnen komt, zijn wij beperkt geweest in de mogelijke algoritmen; veel algoritmen vereisen een grayscale-variant van de originele afbeelding.