# Implementatieplan Jip en Tim

## Namen en datum

Teamlid 1: Jip Galema
Teamlid 2: Tim IJntema
Datum: 17-2-2017

## Doel

**Kwestie:**
Voor het vak vision bestaat een implementatie van gezichtsherkenning software. Deze is echter niet snel/efficiënt genoeg. Om deze sneller/efficiënter te maken zullen we deel van de grijsomzetting gaan verbeteren.

**Ideale situatie:**
Een snelle, efficiënt en werkende inlees functionaliteit voor een plaatje daarnaast ook een goede omzetting van een rgb-waarde naar een grayscale waarde.

## Methoden

Je geeft hieraan welke methoden er zijn, wat de verschillende tussen de methodes zijn, en wat de voor en nadelen hiervan zijn

Er zijn verschillende manieren om een image op te kunnen staan. Hierbij zijn er verschillende afwegingen die we gemaakt hebben. De eerste afweging is een array op de stack of heap. De heap is wel handig aangezien de array dynamisch te vergroten en verkleinen is. Wel is een array op de heap trager aangezien de stack in het snelle CPU cache geheugen zit. De 2$^{e}$ afweging is een 1D array of een 2D array. Hierbij is de 1D array een stuk sneller. Dit komt doordat de rijen van een 2D array in het geheugen verspreid worden op meerdere plekken. Als laatste hebben we nog gekeken naar het gebruik van een vector voor de opslag in het geval dat de heap gebruikt word. Het gebruik van een vector zou geheugen allocatie en deallocatie verzorgen voor ons. Dat zorgt voor minder kans op geheugen lekken. Wel is het net iets trager dan een normale array op de heap.

Bij het zoeken van een methode voor de conversie van RGB naar grijswaarde hebben we gekeken naar 5 methodes. De eerste is heel simpel. Je telt de RGB waardes bij elkaar op en deelt het antwoord door 3, oftewel je neemt het gemiddelde van de 3 waardes. Deze optie is goed toe te passen met een look-up tabel en is dus makkelijk dynamisch te maken.
De nadelen van deze methode hebben vooral met de type grijstinten te maken. Deze optie is namelijk niet zo goed in het weergeven van verschillende soorten grijs en de helderheid van deze tinten. De tweede methode is vergelijkbaar. Het enige verschil is dat de formule voor deze methode een correctie toe voor het menselijke oog. Dit doet men, omdat mensen bepaalde kleuren sterker zien dan anderen. De formule is: Gray = (Red * 0.3 + Green * 0.59 + Blue * 0.11). Dit kost wat meer reken operaties maar is nog steeds best wel efficiënt. Ook hebben we nog de methode die te maken heeft met desaturation. Deze methode zorgt voor een donker grijsachtige tint. Maar hierbij wordt het HSL color space gebruikt, er zou dus een soort omzetting moeten plaatsvinden, dit kan voor extra rekentijd zorgen voor het programma. Decomposition is de volgende methode. Wat hierbij gedaan wordt is het minimum of maximum van de rgb waardes pakken. De simpelste en laatste methode is om een enkele kleur als grayscale waarde te pakken. Je kan bijvoorbeeld voor elke pixel de R-waarde pakken als grayscale waarde. Dit is erg snel en efficent aangezien er geen berekeningen plaats hoeven te vinden. Verder hoeft alleen de waarde die je als grayscale gebruikt opgehaald te worden uit het geheugen.

## Keuze

Voor de imageshell hebben we besloten om in eerste instantie een vector te gebruiken. Dit heeft als voordeel dat de array goed gealloceerd en gedeblokkeerd wordt. Dit is ook een veilige optie (weinig kans op geheugen dat overschreven wordt)

Voor het opzetten van RGB-waardes hebben we in eerste instantie gekozen voor “Luminance”. Maar we gaan een aantal logische omzettingen testen voor efficiëntie en de hoeveelheid geheugen die ze innemen. Hierop zullen we nog een keer kijken op we de goede keuze hebben gemaakt.

## Implementatie

Onze code komt in de volgende classes te zitten: IntensityImageStudent, RGBImageStudent en StudentPreProcessing

## Evaluatie

De volgende testen zouden we graag uit willen voeren:

- Welke grijstinten omzetting is het handigste (kost het minste tijd en geheugen)
- Hoe groot is het verschil tussen een vector of een array op de heap (tijd en geheugen gebruik).

Om onze implementatie te bewijzen:

- Tijd meten van methode in de implementatie en eigengemaakte methode
- Plaatjes onderling vergelijken voor preciezere herkenning van gezichtseigenschappen

In onze meetrapporten is te vinden hoe we deze dingen willen gaan testen.

## Bibliografie

*1d or 2d array whats faster*. (sd). Opgehaald van Stack overflow: http://stackoverflow.com/questions/17259877/1d-or-2d-array-whats-faster

*Do not waste time with STL vectors*. (sd). Opgehaald van Daniel Lemire's blog: http://lemire.me/blog/2012/06/20/do-not-waste-time-with-stl-vectors/

*Seven grayscale conversion algorithms (with pseudocode and VB6 source code)*. (sd). Opgehaald van Tanner Helland (dot) com: http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/

*Stack memory vs heap memory*. (sd). Opgehaald van Stack overflow: http://stackoverflow.com/questions/5836309/stack-memory-vs-heap-memory