1. Implementatieplan Image Shell, grayscale

1.1. Namen en datum

- Bryan Campagne (1724053)
- Gerrit van Os (1719977)
- 19-2-2019 start met maken van plan

1.2. Doel

- Een image shell creëren voor zowel een RGB afbeelding als een intensiteit afbeelding.
    - Zo memory efficiënt mogelijk.
    - De get en set moeten zo snel mogelijk een waarde returnen of aanpassen.
- Tevens het maken van een conversie van een RGB afbeelding naar een intensiteit afbeelding.
    - een zo goed mogelijke image generen zodat de face recognition minder lang hoeft te duren. Op deze manier kan het dus zijn dat het grayscalen zelf langer duurt maar de rest van het proces sneller gaat.
    - Zo nauwkeurig mogelijk houden.

1.3. Methoden

Image shell:

De belangrijkste keuzes die hier mogelijk zijn hebben te maken met de manier van de pixels opslaan. De keuzes hierin zijn uiteraard groot, denk aan een c stijl array of 1 van de stl containers zoals std::array of std:vector. Daarnaast is het nog de vraag hoe de pixels opgeslagen gaan worden. Alle pixels achter elkaar in 1 container of een hoofd container met daarin voor elke rij of kolom een ander container die daadwerkelijk de informatie bevat.

De verschillen hierin zijn zowel in snelheid als in memory efficiency terug te zien.

Grayscale:

- Intensity
  De RGB waardes bij elkaar optellen daarna delen door drie. De uitkomst is de nieuwe grijswaarde.
- Luma
- Luminance <-
  De RGB waardes worden met een coëfficiënt vermenigvuldigd die er voor zorgt dat de foto lijkt op hoe het menselijk oog het ziet
- Lightness
- Value
  De hoogste van de drie RGB waardes wordt de nieuwe grijswaarden.
- Luster
  de hoogste van de drie RGB waardes wordt gedeeld door 2 = de laagste van de drie RGB waardes. De uitkomst hiervan wordt de nieuwe grijswaarden.
- Single color channel
  1 kleur kiezen en deze converteren naar gijs.

1.4. Keuze

Image Shell:

- We gaan kiezen voor een 1d vector waar alle pixels in staan, dit lijkt sneller omdat er op deze manier minder geheugen nodig is als bij een 2d vector.
- Tevens is het getten dan vrij snel omdat er maar 1x array acces gedaan hoeft te worden. De x,y value omrekenen naar een i lijkt ook wat sneller te zijn.

Grayscale:

- We gaan kiezen voor Luminance omdat het menselijk oog het ook zo ziet.
- Wij denken dat gezichtsherkenning gedeelte sneller werkt als de grayscaling zo realistisch mogelijk is.
- relatief eenvoudige berekening omdat de coëfficiënt lineair is.

1.5. Implementatie

- De RGB image shell zal gemaakt worden in de RGBImageStudent klasse
- De Intensiteit image shell zal gemaakt worden in de IntensityImageStudent klasse
    - Voor beide image shells zullen er een 1 of meerdere private variabelen aangemaakt moeten worden om de pixel informatie op te slaan.
- Het grayscale algoritme zal geschreven worden in de volgende functie stepToIntensityImage de implementatie hiervan is te vinden in de file StudentPreProcessing.cpp

1.6. Evaluatie

- De output van de foto's zal vergeleken worden met de standaard daarbij wordt er gekeken of alle facial parameters nog steeds goed gevonden worden.
- Er zal getimed worden of het gray scalen en het toevoegen van de foto in de image shell sneller is.

Bronnen:

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0029740
http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/