1. Implementatieplan titel

1.1. Namen en datum

Nick Goris & Roxanne van der Pol

19-03-2018

1.2. Doel

Een werkende implementatie van de ImagShell en een paar verschillende greyscale algoritmen, om deze te kunnen vergelijken met de standaard implementatie.

1.3. Methoden

Voor de ImageShell is gekozen voor een array van pixels – bij de RGBimages RGB pixels (chars in een struct), en bij de IntensityImage IntensityPixels (char). De manier waarop de pixels worden opgeslagen is hetzelfde als bij de standaard implementatie. Voor de opslag van de afbeeldingen (bij ons in een array) is dit niet bekend.

Voor de greyscaling is voor een single channel implementatie gekozen, waarbij bij het omzetten van de RGB images naar Intensity Images de waarde van de groene pixel uit de RGB gebruikt wordt voor de nieuwe waarde van de intensity pixel. (Helland, 2011)
Het tweede greyscale algoritme dat geimplementeerd is een intensity image gebaseerd op het gemiddelde van de 3 kleurkanalen. (Helland, 2011)

1.4. Keuze

De opslag van pixels in een array was de eerste methode die bij ons op kwam. We wilden dit eerst in een matrix doen, maar in C++ kun je de tweede of hogere dimensie in een matrix niet dynamisch allokeren.

Voor het greyscalen is gekozen voor een single channel implementatie, omdat dit een makkelijk en snel te implementeren methode is, waarvan wij verwachten dat deze zal volstaan bij het omzetten van een RGB image naar een Intensity image.
Het tweede geimplementeerde greyscale algoritme is iets complexer wat rekenen betreft, maar het resulterende grijze plaatje is wellicht mooier en beter te begrijpen voor het menselijk oog dan het grijswaarde plaatje uit de eerste methode.

1.5. Implementatie

Voor de ImageShell zijn de RGBImageStudent.h en .cpp en de IntensityImageStudent.h en .cpp geïmplementeerd. Voor deze beide is er een public attribuut toegevoegd om de pixels voor de afbeeldingen op te kunnen slaan.

Voor het greyscalen is in de file StudentPreProcessing.cpp de stepToIntensityImage() functie geïmplementeerd.

1.6. Evaluatie

Wij zullen de code compileren en runnen. Als er dan een gegrayscaled plaatje gegenereerd wordt en hier een gezicht in herkend wordt, weten we dat onze implementatie werkt. De tijd die het volledige programma duurt zal gemeten worden, evenals de grootte van de resulterende code.
Wij denken dat onze eerste implementatie sneller is dan de standaardimplementatie, en dat onze tweede

implementatie een mooier resultaat oplevert – ten koste van geheugen en snelheid. Wij verwachten dat onze eerste implementatie een programma oplevert dat minder geheugen gebruikt.

2. Bibliography

Helland, T. (2011, October 1). *Seven grayscale conversion algorithms (with pseudocode and VB6 source code)*. Retrieved March 19, 2018, from tannerhelland: http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/