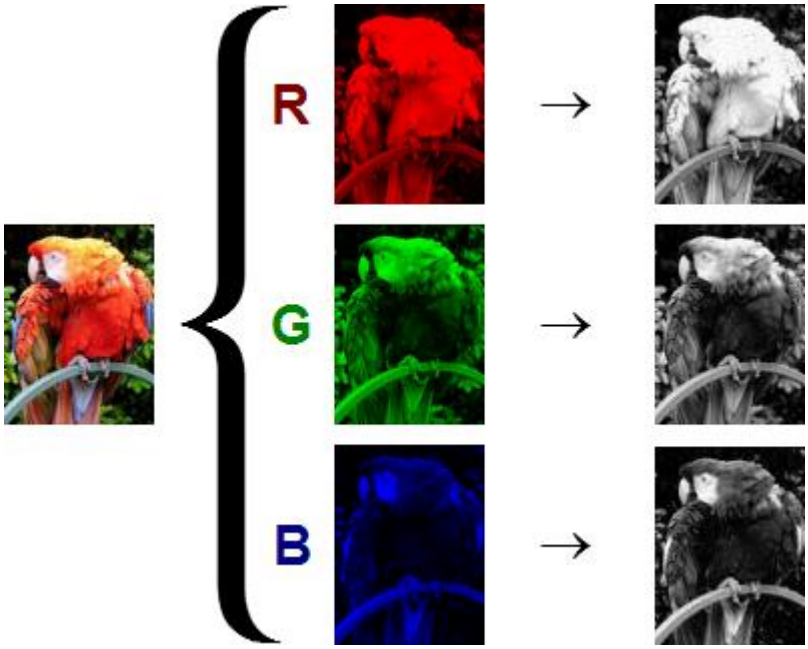
Implementatieplan Conversie naar grijswaarde algoritmes



Gemaakt door:

Patrick Dekker & Mark Gasse

Datum:

14 April 2019

Inhoudsopgave

Doel

Het doel van de implementatie is om een inkomend RGBImage(kleuren afbeelding) om te zetten naar IntersityImage(grijswaarden afbeelding). Hiervoor moet elke pixel een nieuwe waarde krijgen, dit wordt gedaan met behulp van een algoritme. De nieuwe waarde is afhankelijk van de oude RGB kleuren. Er zijn verschillende algoritmes die een ander resultaat als doel einde hebben. Dit kan bijvoorbeeld snelheid zijn of dat de operatie goedkoop is.

Methoden

In het onderstaande gedeelte worden een aantal algoritmes voorgesteld met hun voor- en nadelen en wat de formule van het algoritme is.

Algoritmes:

- Averaging (aka “quik and dirty”)
    - Formule: Gray = (Red + Green + Blue) / 3
    - Snel & makkelijk maar geeft niet goed weer hoe mensen de helderheid van grijs tinten zien.

- Luma/Luminace
    - Formule: Gray = (Red * 0.3 + Green * 0.59 + Blue * 0.11)
    - Grijs tinten op basis van hoe mensen het zien.
    - Onenigheid over juiste formule dus probeer ook:
        - Gray = (Red * 0.2126 + Green * 0.7152 + Blue * 0.0722)
        - Gray = (Red * 0.299 + Green * 0.587 + Blue * 0.114)

- Desaturation
    - Formule: Gray = (Max(Red, Green, Blue) + Min(Red, Green, Blue)) / 2
    - Dit geeft de afbeelding weinig contrast

- Decomposition
    - Max formule: Gray = Max(Red, Green, Blue)
    - Min formule: Gray = Min(Red, Green, Blue)
    - Max maakt de grijs waarde helderder
    - Min maakt de grijs waarde donkerder

- Single color channel
    - Snelste methoden maar het is moeilijk om het resultaat te schatten.
    - Gebruik maar 1 kleur waarde dus een van de onderstaande:
        - Gray = Red
        - Gray = Green
        - Gray = Blue

Keuze

Wij hebben er voor gekozen om er drie te implementeren, zodat er getest kan worden welk algoritme het meest geschikt is. De drie waarvoor we gekozen hebben zijn:

- Averaging
- Luma
- Single color channel

Voor deze drie hebben we gekozen omdat we het verschil willen zien/testen tussen een algoritme dat het best is voor het menselijk oog, een die het aller snelst is maar waarschijnlijk minder goed resultaat heeft en een die er tussen in zit.

Implementatie

Het algoritme wordt geïmplementeerd in de StudentPreProcessing.cpp. In dat bestand wordt in de functie stepToIntensityImage een nieuwe afbeelding gemaakt, De functie gaat over elke pixel van de originele afbeelding en zet ze in de nieuwe afbeelding nadat de pixel is veranderd doormiddel van de formule van het algoritme. De functie geeft de nieuwe afbeelding (grijswaarde afbeelding) terug.

Evaluatie

Er zullen experimenten gedaan moeten worden om de verschillende algoritmes te testen. Dit zal gedaan moeten worden met verschillende afbeeldingen omdat alle afbeeldingen andere eigenschappen hebben, zoals andere kleuren. Dit kan effect hebben hoe goed het programma werkt. Verder kan de student implementatie vergeleken worden met de default implementatie.

Bronvermelding

Helland, T. (2012, 4 juni). Seven grayscale conversion algorithms (with pseudocode and VB6 source code). Geraadpleegd op 14 april 2019, van http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/

Wikipedia contributors. (2019, 16 maart). one in which the value of each pixel is a single sample representing only an amount of light, that is, it carries only intensity information;composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. Geraadpleegd op 14 april 2019, van https://en.wikipedia.org/wiki/Grayscale