# Implementatieplan: ImageShell en Intensity

Agterberg, Ole – 1651981

Stoeltie, Ferdi – 1665045

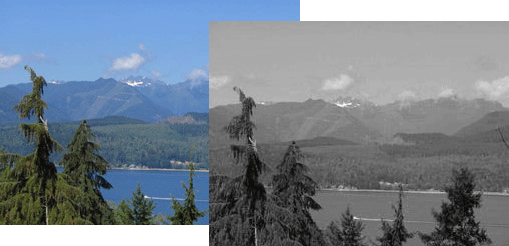


Image: <http://bubble.ro/How_to_convert_an_image_to_grayscale_using_PHP.html>

# Doel

De opdracht bestaat uit twee doelen. Het eerste doel is om Imageshells te maken voor RGB en intensiteit. ­Hiermee kunnen alle pixels van een afbeelding opgeslagen worden. Het tweede doel is om RGB om te zetten naar intensiteit (Grayscale). Dit zorgt er voor dat latere image processing stappen beter werken. Het tweede doel is afhankelijk van het eerste doel. Om deze doelen te realiseren zijn de volgende punten opgesteld:

* Documenten opstellen bestaande uit: Diagrammen, modellen, code en bijbehorende documentatie;
* Maken Imageshell klasse voor RGB en intensiteit;
* RGB-waarden (pixels en afbeelding) om te zetten naar grijs-waarden (Grayscale).
  + Onderzoek doen naar conversie van RGB-waarden naar grijs-waarden;
    - Verschillende methoden vergelijken en testen.
  + Testen in de GUI applicatie;
  + Het maken van een ImageShell klasse voor RGB en voor Intensiteit.

# Methoden

Zoals hier boven beschreven zijn er twee doelen. Om de doelen te behalen zijn ook twee verschillende methoden nodig.

## Data opslag

De pixel data van de image kunnen op verschillende manieren opgeslagen worden, zoals;

* Array
  + - Snel opzoeken van elementen;
    - Snel opslaan van elementen;
    - Constante grootte.
* Vector;
  + - Toevoegen van elementen kan lang duren;
    - Dynamische grootte;
    - Gaat niet efficiënt om met geheugen;
    - Snel elementen opzoeken.
* List;
  + - Toevoegen van element gaat snel;
    - Dynamische grootte;
    - Gaat efficiënt om met geheugen;
    - Opzoeken van elementen duurt lang.
* Set – Een gesorteerde lijst met unieke waarden;
  + - Is niet bruikbaar voor deze toepassing.
* Map – Een gesorteerde lijst met key, value;
  + - Is niet bruikbaar voor deze toepassing.

## Conversie

Voor de conversie naar grijs-waarden zijn twee methoden;

* Gemiddelde
  + Snel door weinig berekeningen
  + Komt niet overeen met werkelijkheid
* Gewogen gemiddelde
  + Extra berekening nodig
  + Komt beter overeen met werkelijkheid

Je geeft hier aan welke methoden er zijn, wat de verschillende tussen de methodes zijn.

Je geeft een onderbouwing over waarom een bepaalde methode is gekozen, en/of waarom bepaalde settings zijn gebruikt.

# Keuze

## Data opslag

Gekozen is om gebruik te maken van een *std::vector* met een vooraf – op basis van de hoeveelheid pixels - gedefinieerde grootte. Een *std::list* is traag om specifieke elementen op te zoeken. Twee nadelen van een vector zijn het inefficient omgaan met geheugen en het toevoegen van nieuwe elementen. Aangezien er alleen elementen (pixels) worden toegevoegd bij initialisatie vormen de nadelen geen probleem (Wicht, 2012).

## Conversie

De gekozen methode is de methode met het gewogen gemiddelde, de extra berekeningen die nodig

# Implementatie

Je geeft aan hoe deze keuze is geimplementeerd in de code

# Evaluatie

Testen van de conversie op:

* Snelheid
* Memory efficiëntie
* Robuustheid
* Volledigheid

Je geeft aan welke experimenten er gedaan zullen worden om de implementatie te testen en te ‘bewijzen’ dat de implementatie daadwerkelijk correct werkt. Dit geeft direct informatie over de meetrapporten die er zullen worden gemaakt.

# Verwijzingen

Wicht, B. (2012, November 26). *C++ benchmark - std::vector VS std::list*. Opgeroepen op 2020, van Baptiste Wicht: https://baptiste-wicht.com/posts/2012/11/cpp-benchmark-vector-vs-list.html