Meetrapport: conversie snelheid

Stoeltie, Ferdi – 1665045  
Agterberg, Ole – 1651981

Manon – 113ietsmeteentwee

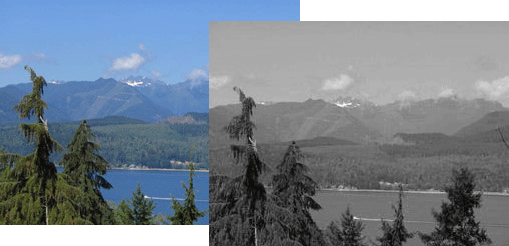
18-03-2020

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

# Doel

Het doel van het meetrapport is om te kijken of snelheid van de conversie van RGB naar intensiteit is verbeterd ten op zichten van de bestaande conversie. De snelheid wordt beïnvloed door het wegschreven van pixels naar de intensity klassen, naar het verkrijgen van de RGB pixels uit de RGB klassen en natuurlijk de conversie van RGB naar grayscale.

Er zijn vier test categorieën. Elke categorie geeft een uniek scenario weer. Elke foto zal getest worden in de vier categorieën.

* O2S -> O2 (optimized for speed), single-threaded
* O2M -> O2 (optimized for speed), multi-threaded
* DM -> Default (not optimized), multi-threaded
* DS -> Default (not optimized), single threaded

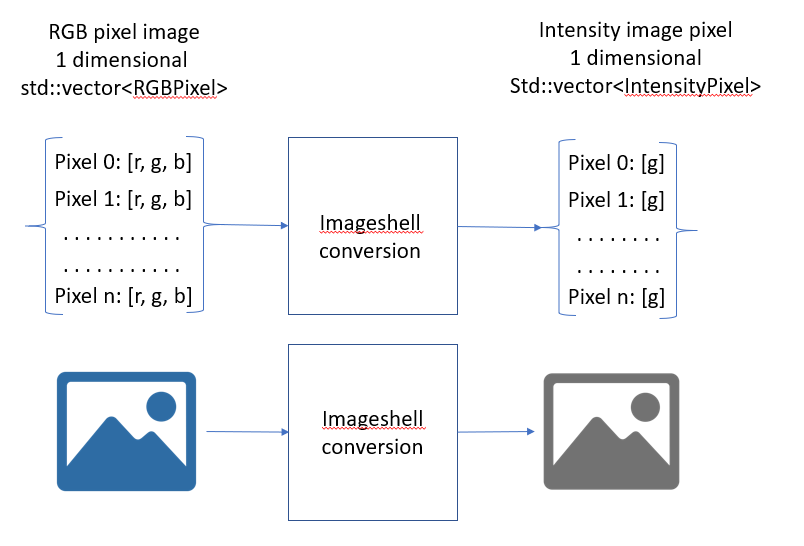
|  |  |  |
| --- | --- | --- |
| Variable  Category | O2 | Multi-threaded |
| O2S | **✓** | x |
| O2M | **✓** | **✓** |
| DM | x | **✓** |
| DS | x | x |

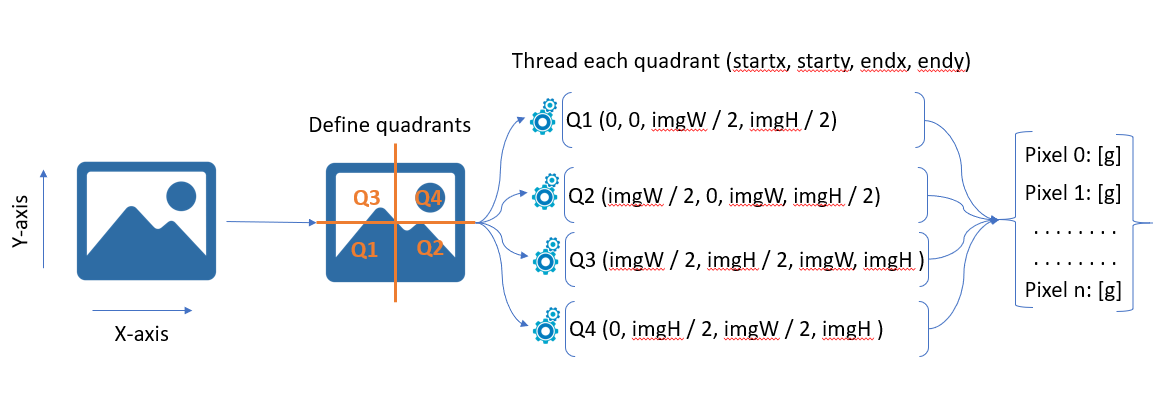
# Hypothese

Wij verwachten dat de student implementatie minder snel zal zijn, omdat we niet de snelste conversie methode hebben gekozen. Daarnaast hebben we gekozen om de data op te slaan in een vector. Vector is voor het toevoegen van elementen niet snel. We denken dat de default implementatie gebruik maakt van een array, die erg snel elementen kan toevoegen.

Er is gekozen voor een vector waarvan de grootte, eenmalig gedefinieerd wordt door de hoeveelheid pixels N van de image. Hierdoor heeft de vector een constante grootte N en hoeft niet te worden verplaatst wanneer de pointers naar de pixels worden gezet. Schalen van een vector kost veel tijd en dat hoeft niet met deze aanpak.

Verder worden de RGB pixels in Intensity pixels omgezet in vier threads tegelijk. Hierdoor is het mogelijk om de conversie parallel uit te voeren.





Elke thread handelt afzonderlijk een kwart van de foto af, en vult een kwart van de vector.

# Werkwijze

Voor elke test zal de functie executePreProcessingStep1 meerdere malen (ongeveer 300 keer) aangeroepen worden. De tijd die nodig was om de test uit te voeren wordt uitgeprint. Deze test wordt vijf keer uitgevoerd voor de zelf gemaakte code en vijf keer voor de default code. Hierna zal het verschil in tijd in percentage berekend worden.

# Resultaten

Eerste keer:

|  |  |
| --- | --- |
| **Default** | **Student** |
| 2749848000 | 1810264500 |
| 2240876900 | 1912801500 |
| 2214771900 | 1791599800 |
| 2213354700 | 1801846600 |
| 2203699000 | 1773925600 |
| **2324510100** | **1818087600** |

Tweede keer:

|  |  |
| --- | --- |
| **Default** | **Student** |
| 2893738500 | 2035345200 |
| 2257994300 | 2048420100 |
| 2252571900 | 1924862200 |
| 2248997400 | 2042516700 |
| 2436216400 | 1886146000 |
| **2417903700** | **1987458040** |

# Verwerking

# Conclusie