ImageShell en Grayscale

## Namen en datum

Wilco Matthijssen, Daan Zimmerman van Woesik

25-03-2020

## Doel

Ons doel van dit experiment was om te onderzoeken of we een manier konden vinden waardoor een foto sneller kon omgezet worden met zo min mogelijk verlies in kwaliteit.

## Hypothese

Wij verwachten dat onze implementatie sneller zal zijn dan de gegeven implementatie gegeven door school zonder teveel verlies van de kwaliteit.

## Werkwijze

Wij gaan kijken of onze implementatie sneller is dan het gegeven algoritme van school. Dit gaan we doen door meerdere afbeeldingen om te zetten naar grayscale en de tijd dat dit kost te vergelijken met de andere implementatie. Wij voeren dit meerdere keren uit zodat eventuele pieken niet zoveel invloed hebben op het gemiddelde.

Daarna kijken we niet alleen of het sneller is maar ook of het omgezette afbeelding dezelfde of misschien betere kwaliteit heeft. Wij gaan dit doen door de omgezette afbeelding door de rest van de imageprocessing te laten doorlopen gegeven door school. De gebruikte afbeeldingen zijn ook verschillende soorten mensen en ook objecten. Hiervoor is gekozen om te kijken of de imageprocessing misschien een gezicht ziet die normaal niet wordt gezien bij het algoritme waar we ons tegen vergelijken. Tijdens deze vergelijking geldt, als een algoritme verder komt dan de andere is deze dan ook beter.

## Resultaten

Tijdens het testen van de onze implementatie hebben we het gemiddelde en de standaarddeviatie genoteerd. (Zie Bron 1.0) Daarnaast hebben we ook de kwaliteit bekeken van de algoritmes en genoteerd. (Zie Bron 1.1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Naam afbeelding (Resolutie foto) | Default gemiddelde(us) | Student gemiddelde(us) | Default  Standaarddeviatie | Student  Standaarddeviatie |
| **blue.png** (275x183) | 410,74 | 243,14 | 40,00 | 13,10 |
| **child-1.png**  (225x225) | 420,74 | 246,31 | 69,51 | 14,75 |
| **female-1.png**  (195x258) | 434,59 | 254,72 | 40,90 | 29,76 |
| **female-2.png**  (198x255) | 402,37 | 243,36 | 28,78 | 12,21 |
| **female-3.png**  (400x267) | 940,29 | 507,02 | 120,04 | 18,51 |
| **female-4.jpg**  (300x300) | 713,78 | 425,71 | 36,24 | 9,68 |
| **female-5.jpg**  (337x300) | 957,30 | 478,43 | 46,64 | 11,64 |
| **male-1.png**  (194x259) | 417,66 | 239,42 | 30,50 | 5,79 |
| **male-2.png**  (194x259) | 417,26 | 238,69 | 20,98 | 6,18 |
| **male-3.png**  (198x255) | 416,31 | 239,57 | 25,25 | 3,94 |
| **male-4.jpg**  (226x301) | 545,80 | 322,41 | 31,26 | 8,34 |
| **male-5.jpg**  (214x300) | 530,62 | 303,77 | 26,88 | 4,62 |
| **male-6.jpg**  (208x300) | 496,85 | 297,22 | 18,74 | 17,74 |
| **male-7.jpg**  (200x301) | 483,87 | 284,58 | 20,58 | 8,71 |
| **rock-1.jpg**  (250x251) | 513,38 | 297,48 | 27,48 | 10,78 |
| **rock-2.jpg**  (400x301) | 1048,28 | 568,68 | 55,10 | 15,54 |
| **sky-1.jpg**  (400x200) | 633,11 | 378,76 | 23,63 | 16,65 |
| **sky-2.jpg**  (400x275) | 934,01 | 523,60 | 60,25 | 14,86 |
| **tree-1.png**  (400x255) | 914,24 | 484,93 | 50,10 | 12,38 |
| **tree-2.jpg**  (278x276) | 628,19 | 365,25 | 31,56 | 17,19 |

**Bron 1.0:** Snelheid resultaten van het Default en Student Algoritme.

|  |  |  |  |
| --- | --- | --- | --- |
| Naam afbeelding (Resolutie foto) | Default implementatie | Student implementatie | Correcte uitkomst |
| **blue.png**  (275x183) | Step 2 | Step 2 | Step 2 |
| **child-1.png**  (225x225) | X | X | X |
| **female-1.png**  (195x258) | Step 5 | X | X |
| **female-2.png**  (198x255) | X | X | X |
| **female-3.png**  (400x267) | Step 2 | Step 2 | X |
| **female-4.jpg**  (300x300) | Step 2 | Step 2 | X |
| **female-5.jpg**  (337x300) | Step 5 | X | X |
| **male-1.png**  (194x259) | X | X | X |
| **male-2.png**  (194x259) | Step 5 | Step 2 | X |
| **male-3.png**  (198x255) | X | Step 3 | X |
| **male-4.jpg**  (226x301) | Step 4 | Step 5 | X |
| **male-5.jpg**  (214x300) | Step 4 | Step 5 | X |
| **male-6.jpg**  (208x300) | Step 5 | Step 2 | X |
| **male-7.jpg**  (200x301) | Step 4 | Step 4 | X |
| **rock-1.jpg**  (250x251) | Step 2 | Step 2 | Step 2 |
| **rock-2.jpg**  (400x301) | Step 2 | Step 2 | Step 2 |
| **sky-1.jpg**  (400x200) | Step 2 | Step 2 | Step 2 |
| **sky-2.jpg**  (400x275) | Step 2 | Step 2 | Step 2 |
| **tree-1.png**  (400x255) | Step 2 | Step 2 | Step 2 |
| **tree-2.jpg**  (278x276) | Step 2 | Step 2 | Step 2 |

**Bron 1.1:** Image processing slagings tabel. Hier betekend een hogere step hoe verder het gezicht herkenning algoritme is gekomen met het gezicht herkennen. Een X betekend hierbij dat het gezicht is gevonden. Kleuren zijn toegevoegd voor verduidelijking, Groen staat gelijk aan een verwacht resultaat hebben. Geel staat gelijk aan niet een verwacht resultaat maar beter of hetzelfde resultaat als het andere algoritme. Rood staat gelijk aan een niet verwacht resultaat en ook slechter dan het andere algoritme.

## Verwerking

**Bron 1.2**: Gemiddelde verandering vergelijken met de default implementatie.

Bij de gekregen resultaten hebben we gekeken naar wat de percentuele veranderingen zijn tussen de default implementatie en die van ons. Hierbij hebben we gebruik gemaakt van de formule:

Hierbij hebben we gekeken naar het gemiddelde en de standaarddeviatie (Zie Bron 1.2). Hieruit is best te concluderen dat onze implementatie aanzienlijk beter is dan de Default implementatie. Buiten dat de afbeeldingen veel sneller worden omgezet naar grayscale, worden de gegenereerde afbeeldingen veel dichter richting het gemiddelde getrokken. Dit omdat de standaarddeviatie flink is verlaagd. Het gemiddelde is met ongeveer 75% gestegen en de standaarddeviatie met een flinke 256%. Echter zijn deze waardes helemaal niks waard als de kwaliteit van de afbeeldingen verslechterd.

Gelukkig is dit niet het geval (Zie Bron 1.1). Hoewel onze implementatie nog op sommige afbeeldingen beter is dan de Default implementatie is het bij andere afbeeldingen het tegenovergestelde. Ons doel was ook niet om dat te verbeteren maar wel om sneller te zijn. Echter hebben wij onze testen voornamelijk getest op relatief kleine afbeeldingen. Dit verschil zal voornamelijk te merken zijn bij grotere afbeeldingen als we uitgaan van een gemiddelde verbetering van 75%.

## Conclusie

Met de verwerking van de resultaten is gebleken dat de Student implementatie een stuk sneller is dan de Default implementatie, met een gemiddelde van 75% sneller. Al is de snelheid wel het doel, is de accuraatheid ook iets wat zo min mogelijk verloren moest worden wat ook onderzocht is. Met de aanzienlijke snelheidsverbetering is de accuraatheid grotendeels gelijk gebleven.(Zie Bron 1.1)

## Evaluatie

Zoals bij de conclusie is Student implementatie sneller dan de Default implementatie wat overeen komt het doel en hypothese van dit meetrapport. Er waren problemen met het meten van de snelheid. De resultaten werden bij een klein deel beïnvloed door andere factoren. Deze factoren zorgden bij sommige resultaten voor uitschieters die niet te maken hebben met wat er gemeten werd. We vermoeden dat deze door Windows scheduler en andere achtergrondprogramma’s komt. Omdat dit al van te voren was verwacht maar op een stuk mindere mate hebben we deze test 1000 keer per foto gedaan en hiervan het gemiddelde en standaarddeviatie genomen.