Meetrapport, Snelheid - Scaling

Dimitry Volker- 1661152
Jasper van Hulst - 1660498

6 februari 2018

Inhoudsopgave

1 Doel

Het schalen van een afbeelding is op diverse manieren voor elkaar te krijgen. Er bestaan 3 algoritmes die veel op elkaar lijken "Nearest-neighbour Interpolation", "Bilinear Interpolationën "Bicubic Interpolation". Wanneer er een situatie voorkomt waar je bijvoorbeeld heel veel afbeeldingen moet scalen, dan is het belangrijk om te weten wat de snelste is, ongeacht de kwaliteit. Daarom stellen wij de volgende vraag:

"Welk algoritme is het snelst voor scaling?".

2 Hypothese

Het lijkt ons dat de Nearest-neighbour op alle experimenten het snelste zal presteren. Er zijn bijna geen calculaties en er word alleen gekeken naar de "dichtstbijzijnde"pixel.

3 Onderzoeksmethode

Om een goed beeld te krijgen naar de snelheid van de drie algoritmes worden de experimenten tien keer uitgevoerd per schalingsfactor. Hier kijken wij naar het verkleinen en vergroten van een afbeelding vanaf de orginele grote.

Figuur 1: Orginele afbeelding

0.5x
0.75x
1.25x
1.50x

Tabel 1: Geteste schalingsfactoren

3.1 Code

```
#include <time.h>
clock_t start, finish;
...

for (int i = 0; i < 10; i++) {
    result = ImageFactory::newIntensityImage(WIDTH, HEIGHT);
    start = clock();

    //scaling here //

    finish = clock();
    std::cout << "Time for x (seconds):"
    << ((double)(finish - start)) / CLOCKS_PER_SEC;
    delete result;
}
```

Listing 1: Code voor timen

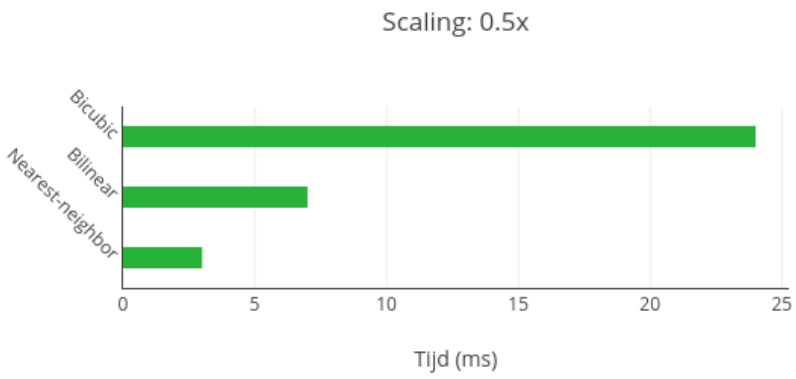
4 Resultaten

Zoals aangegeven in hoofdstuk 3, word de afbeelding geschaald met diverse schalingsfactoren. Om een goed resultaat te krijgen word elke test 10 keer uitgevoerd.

4.1 Schalingsfactor 0.5

Nearest-neighbor interpolation	Bilinear Interpolation	Bicubic Interpolation
0,005s	0,009s	0,026s
0,003s	0,006s	0,024s
0,004s	0,007s	0,024s
0,003s	0,006s	0,023s
0,003s	0,007s	0,024s
0,003s	0,007s	0,024s
0,003s	0,007s	0,023s
0,004s	0,006s	0,023s
0,003s	0,006s	0,024s
0,003s	0,007s	0,027s

Tabel 2: Resultaten schalen met factor 0.5

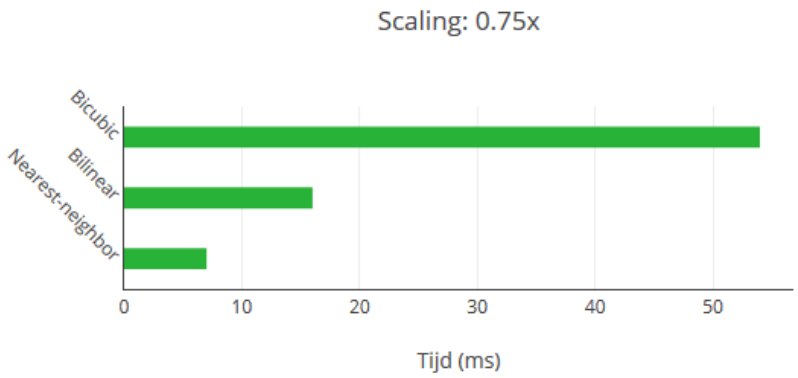


Figuur 2: Gemiddelde lengte van schalen met factor 0.5

4.2 Schalingsfactor 0.75

Nearest-neighbor interpolation	Bilinear Interpolation	Bicubic Interpolation
0,008	0,015	0,055
0,007	0,015	0,059
0,007	0,015	0,056
0,007	0,015	0,053
0,007	0,015	0,051
0,009	0,016	0,052
0,007	0,021	0,058
0,007	0,015	0,055
0,007	0,015	0,052
0,007	0,017	0,052

Tabel 3: Resultaten schalen met schaalfactor 0.75

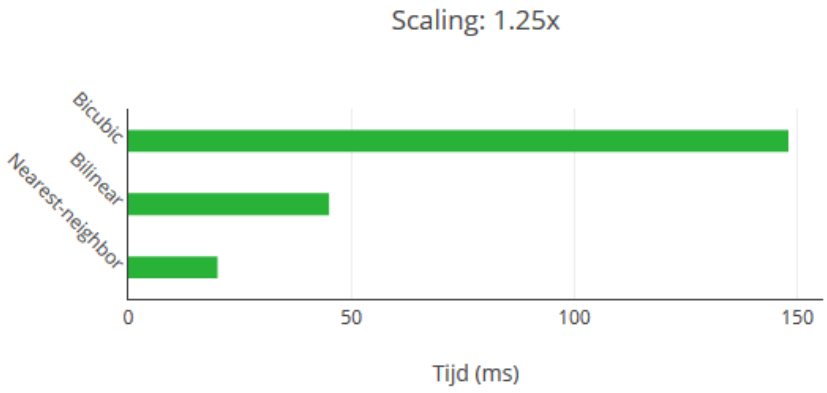


Figuur 3: Gemiddelde lengte van schaling met factor 0.75

4.3 Schalingsfactor 1.25

Nearest-neighbor interpolation	Bilinear Interpolation	Bicubic Interpolation
0,020	0,048	0,162
0,020	0,047	0,146
0,020	0,052	0,152
0,022	0,043	0,146
0,023	0,044	0,145
0,019	0,041	0,145
0,019	0,042	0,145
0,019	0,050	0,144
0,018	0,044	0,146
0,019	0,041	0,145
0,020	0,045	0,148

Tabel 4: Resultaten schalen met schaalfactor 1.25

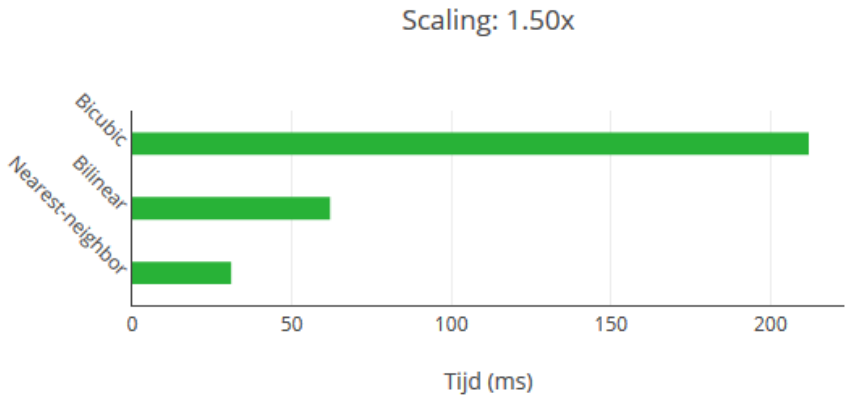


Figuur 4: Gemiddelde lengte van schaling met factor 1.25

4.4 Schalingsfactor 1.5

Nearest-neighbor interpolation	Bilinear Interpolation	Bicubic Interpolation
0,027	0,059	0,222
0,034	0,062	0,220
0,034	0,063	0,216
0,033	0,070	0,210
0,036	0,061	0,211
0,033	0,062	0,210
0,028	0,061	0,208
0,029	0,062	0,206
0,028	0,063	0,206
0,028	0,060	0,210

Tabel 5: Resultaten schalen met schaalfactor 1.50



Figuur 5: Gemiddelde lengte van schaling met factor 1.50

5 Conclusie

Uit de resultaten kunnen we concluderen dan, ongeacht vergroten of verkleinen, de "Nearest-Neighbour Interpolatie"het snelste algoritme van de 3 is.