1. Meetrapport Image Shell

1.1. Namen en datum

Jeroen Steendam (1607288)
Timon van den Brink (1664554)
27/4/2016

1.2. Doel

Het doel van ons experiment is om aan te tonen of onze implementatie sneller is dan de default implementatie. Ook wordt er gekeken hoe een vector hier tegen op weegt. Hierbij kijken we naar geheugen gebruik en ook kijken we naar de snelheid waarmee deze wordt uitgevoerd. Hiermee willen we aantonen of onze hypothese juist is.

1.3. Hypothese

Wij verwachten dat de snelheid van StudentImage implementatie zo goed als gelijk is aan de PrivateImage implementatie. We verwachten dat het geheugen gebruik bij StudentImage implementatie lager zal zijn dan bij de PrivateImage implementatie. Ook verwachten we dat de IntesityImage drie maal zo snel zal zijn als RGBImage, doordat InstesityImage drie maal zo weinig data opslaat. Hierdoor kunnen aanpassingen, naar onze verwachtingen, drie maal zo snel verlopen.

1.4. Werkwijze

We draaien de tests op een Windows 10 machine. Deze machine draait Visual Studio 2015 debug mode met een i7 6700K met 16 gb ram. We laden de testafbeelding in voor RGBImagePrivate, RGBImageStudent en RGBImageVector, dit wordt ook gedaan voor IntensityImage.

Vervolgens wordt gemeten met behulp van een timer hoelang de uitvoer tijd van de access functies zijn, denk aan setPixel() en getPixel(). Met de diagnostics tools van Visual Studio wordt het geheugengebruik gemeten.

Hiervan zal de standaard afwijking worden berekend, waardoor wij kunnen bepalen hoe constant de functies zijn qua snelheid.

De resultaten zullen worden verwerkt in een spreadsheet waarmee er grafieken kunnen worden genereren. Deze grafieken zijn vervolgens in dit document geplaatst.

1.5. Resultaten

Geheugengebruik kleuren en intensiteit afbeelding
Test afbeelding van 198*255 pixels, totaal 50490 pixels.

ShellType	Total size (bytes)	Pixel list size (bytes)	Overhead (bytes)
RGBImagePrivate	151.486	151.470	16
RGBImageStudent	151.486	151.470	16
RGBImageVector	151.529	151.505	59
IntesityImagePrivate	50.490	50.474	16
IntesityImageStudent	50.490	50.474	16
IntesityImageVector	50.525	50.501	24

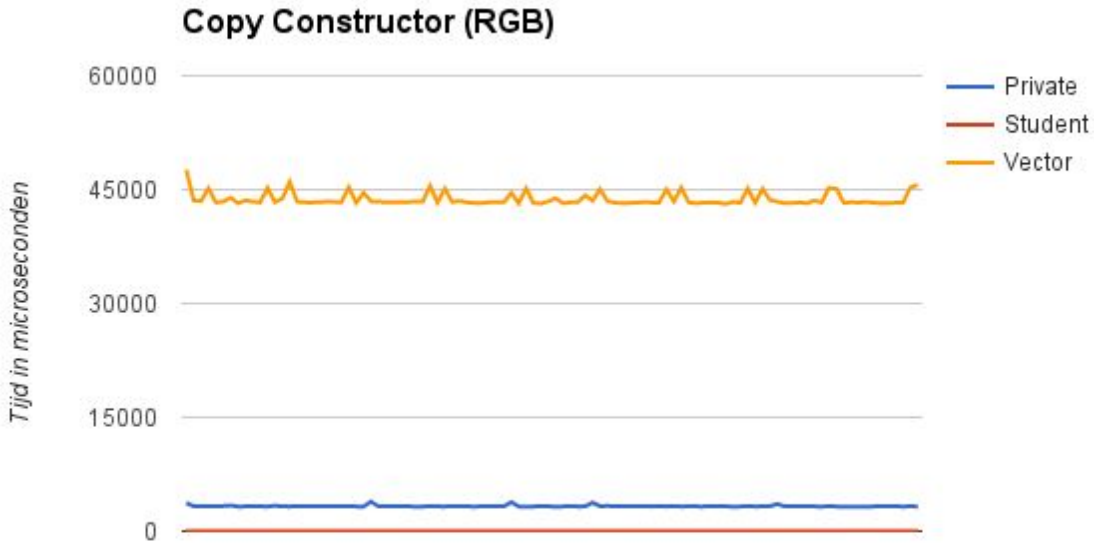
De gemiddelde snelheid in nanoseconden of microseconden per functie van de implementaties. De copy constructor, de getpixel en de setpixel zijn per stuk 100 keer getest.

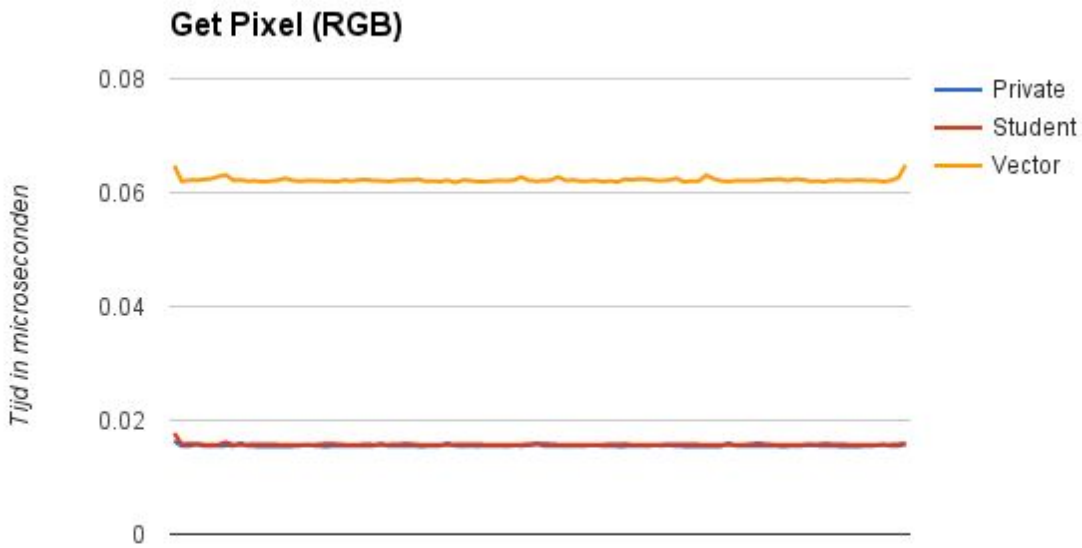
Functie	getPixel	setPixel	copyConstructor
RGBImagePrivate	15,55 ns	34,40 ns	3294,31 µs
RGBImageStudent	15,79 ns	34,65 ns	2,57 µs
RGBImageVector	62,22 ns	88,51 ns	43740,43 µs
IntesityImagePrivate	15,74 ns	16,10 ns	1500,75 µs
IntesityImageStudent	16,10 ns	16,38 ns	2,60 µs
IntesityImageVector	69,21 ns	67,87 ns	25979,36 µs

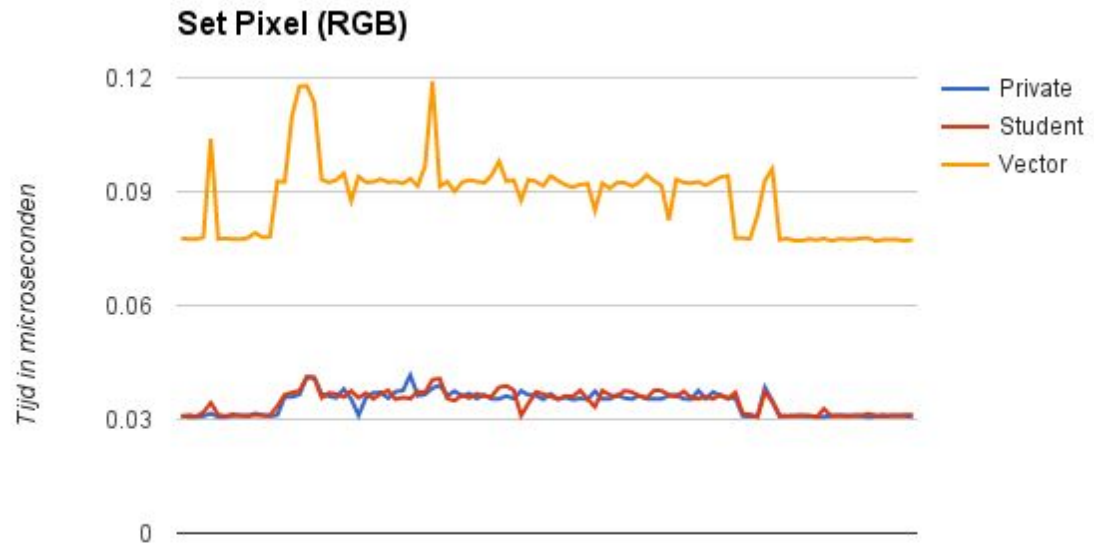
Standaarddeviatie per functie

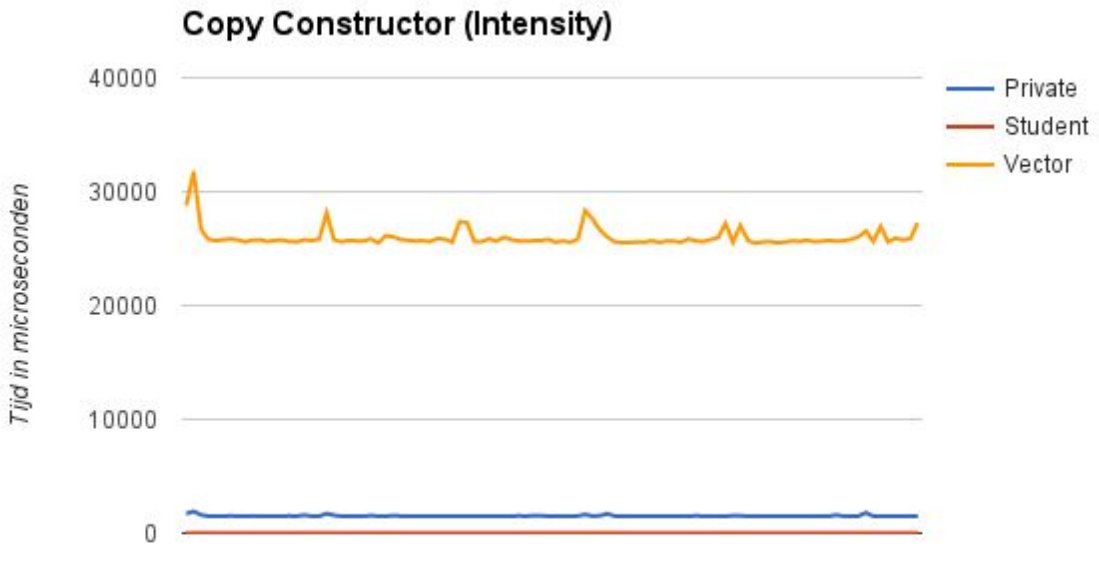
Functie	getPixel	setPixel	copyConstructor
RGBImagePrivate	0,14 ns	2,93 ns	112,67 µs
RGBImageStudent	0,21 ns	2,91 ns	0,41 µs
RGBImageVector	0,44 ns	9,68 ns	834,82 µs
IntesityImagePrivate	1,48 ns	1,44 ns	66,51 µs
IntesityImageStudent	1,70 ns	1,53 ns	0,48 µs

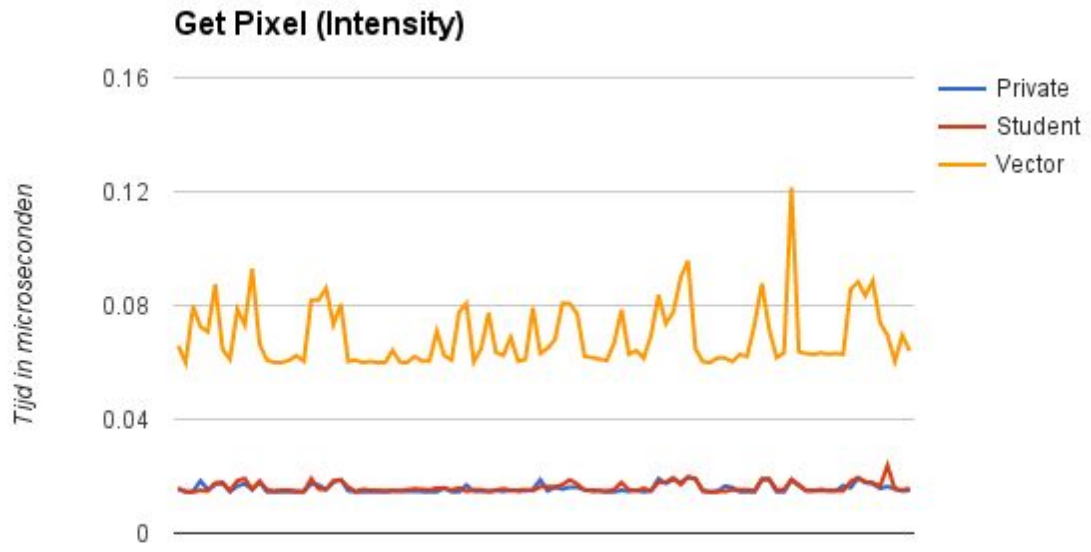
IntesityImageVector	10,89 ns	11,69 ns	854,62 µs

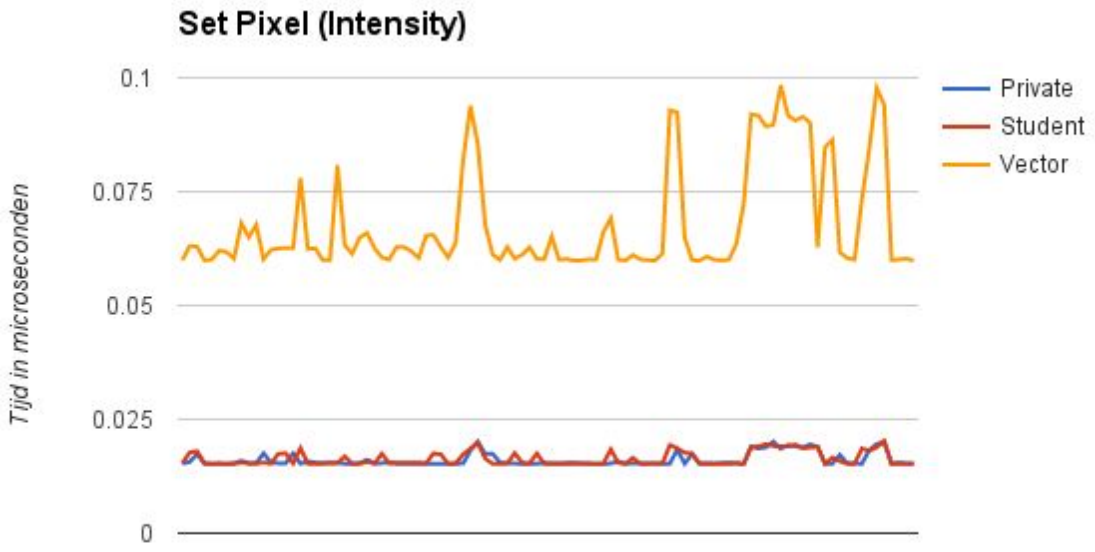





Set Pixel (RGB)
0.12
0.09
0.06
0.03
0
Tijd in microseconden
Private
Student
Vector

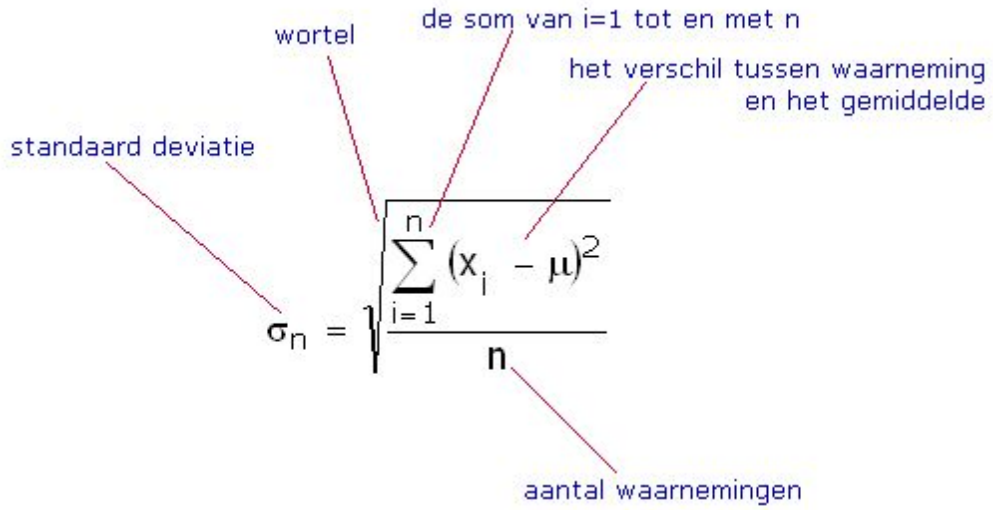

Copy Constructor (Intensity)
40000
30000
20000
10000
0
Tijd in microseconden
Private
Student
Vector


Get Pixel (Intensity)
0.16
0.12
0.08
0.04
0
Tijd in microseconden
Private
Student
Vector


Set Pixel (Intensity)
0.1
0.075
0.05
0.025
0
Tijd in microseconden
Private
Student
Vector

1.6. Verwerking

We hebben gebruik gemaakt van de standaard deviatie om aan te tonen hoe constant de functies zijn.



De copyConstructor van RGBImagePrivate heeft een gemiddelde meetwaarde van 3294,31 µs en een standaard deviatie van 122,67 µs. Daarom kan met een zekerheid van 95% worden gezegd dat de meetwaarde binnen de 3294,31 - (2 * 122,67) en 3294,31 + (2 * 122,67) valt.

1.7. Conclusie

In dit hoofdstuk wordt een conclusie getrokken uit de verwerking van de meetresultaten.

Het geheugengebruik verschilt niet veel tussen de implementaties. De private en de student implementaties gebruiken precies evenveel geheugen, alleen de vector implementatie heeft meer overhead. Bij de intensity image is dat 8 bytes en bij de rgb image is dit 43 bytes dit is verwaarloosbaar bij groottes van 50.000 bytes en 150.000 bytes.

Geconcludeerd kan worden dat de std::vector duidelijk meer tijd nodig heeft om bewerkingen te doen. Ook is std::vector instabiel dit kunnen we zien in de tabel met standaarddeviaties. Wij concluderen dat std::vector niet handig is voor een imageShell en alleen gebruikt zou moeten worden bij korte lijsten.

De Private implementaties, RGBImagePrivate en IntensityImagePrivate, en de Student implementaties, RGBImageStudent en IntensityImageStudent, zijn ongeveer gelijk aan elkaar, maar de copyConstructor van de Student implementaties zijn veel sneller, ook is hierdoor de standaarddeviatie van Student's copyConstructor lager.

1.8. Evaluatie

In dit hoofdstuk wordt een verband getrokken tussen de conclusie en het doel van het experiment (de hypothese). Daarbij wordt ook gekeken naar eventuele meetonzekerheden.

Hypothese
Wij verwachten dat de snelheid van StudentImage implementatie zo goed als gelijk is aan de PrivateImage implementatie. We verwachten dat het geheugen gebruik bij StudentImage implementatie lager zal zijn dan bij de PrivateImage implementatie. Ook verwachten we dat de IntesityImage drie maal zo snel zal zijn als RGBImage, doordat InstesityImage drie maal zo weinig data opslaat. Hierdoor kunnen aanpassingen, naar onze verwachtingen, drie maal zo snel verlopen.

Conclusie op hypothese
In de hypothese verwachtte wij dat de snelheid van de StudentImage gelijk zou zijn aan die van PrivateImage. Dit was het geval voor getPixel en setPixel maar als we kijken naar de copyConstructor was de StudentImage sneller.

Het geheugengebruik was, tegen onze verwachting in, voor zowel de PrivateImage als StudentImage gelijk. De VectorImage gebruikte een klein beetje meer, dit was verwaarloosbaar.

De snelheid van de implentatie van IntensityImage was op sommige plekken bijna gelijk aan de RGBImage implementatie. Maar op andere plekken scheelde het bijna factor 2.

Meetonzekerheden
De metingen van de IntesityImage implementaties zijn op een ander moment uitgevoerd dan de metingen van de rgbimage implementaties. Dit zou gevolgen kunnen hebben voor de testresultaten, ookal achten wij deze kans klein.