

Sobel methode

De code:

```
auto start = high_resolution_clock::now();

cv::Mat imageContainer;

HereBeDragons::HerLoveForWhoseDearLoveIRiseAndFall(src, imageContainer);

cv::Mat sobelx = (cv::Mat_<float>(3, 3) << -1, 0, 1, -2, 0, 2, -1, 0, 1);
cv::Mat sobely = (cv::Mat_<float>(3, 3) << -1, -2, -1, 0, 0, 0, 1, 2, 1);

cv::Mat sobelxResult;
cv::Mat sobelyResult;

filter2D(imageContainer, sobelxResult, -1, sobelx, cv::Point(-1, -1));
filter2D(imageContainer, sobelyResult, -1, sobely, cv::Point(-1, -1));

IntensityImage* edgeDetectionImage = ImageFactory::newIntensityImage();

cv::Mat weighted;
cv::addWeighted(sobelxResult, 3, sobelyResult, 3, 0, weighted, -1);

HereBeDragons::NoWantOfConscienceHoldItThatICall(weighted, *edgeDetectionImage);

auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);
std::cout << "-----" << duration.count() << std::endl;

return edgeDetectionImage;
```

Prewitt methode

De code:

```
auto start = high_resolution_clock::now();

cv::Mat imageContainer;
HereBeDragons::HerLoveForWhoseDearLoveIRiseAndFall(src, imageContainer);

cv::Mat prewittx = (cv::Mat_<float>(3, 3) << 1, 0, -1, 1, 0, -1, 1, 0, -1 );
cv::Mat prewitty = (cv::Mat_<float>(3, 3) << 1, 1, 1, 0, 0, 0, -1, -1, -1 );

cv::Mat prewittxResult;
cv::Mat prewittyResult;

filter2D(imageContainer, prewittxResult, -1, prewittx, cv::Point(-1, -1));
filter2D(imageContainer, prewittyResult, -1, prewitty, cv::Point(-1, -1));

IntensityImage* edgeDetectionImage = ImageFactory::newIntensityImage();

cv::Mat weighted;
cv::addWeighted(prewittxResult, 3, prewittyResult, 3, 0, weighted, -1);

HereBeDragons::NoWantOfConscienceHoldItThatICall(weighted, *edgeDetectionImage);

auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);
std::cout << "-----" << duration.count() << std::endl;

return edgeDetectionImage;
```

Default Laplacian methode

De code:

```
auto start = high_resolution_clock::now();

cv::Mat OverHillOverDale;

HereBeDragons::HerLoveForWhoseDearLoveIRiseAndFall(src, OverHillOverDale);

cv::Mat ThoroughBushThoroughBrier = (cv::Mat_<float>(9, 9) << 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
    -4, -4, -4, 1, 1, 1, 1, 1, 1, -4, -4, -4, 1, 1, 1, 1, 1, 1,
    -4, -4, -4, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
    1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0);

cv::Mat OverParkOverPale;

filter2D(OverHillOverDale, OverParkOverPale, CV_8U, ThoroughBushThoroughBrier, cv::Point(-1, -1), 0, cv::BORDER_DEFAULT);
IntensityImage* ThoroughFloodThoroughFire = ImageFactory::newIntensityImage();

HereBeDragons::NoWantOfConscienceHoldItThatICall(OverParkOverPale, *ThoroughFloodThoroughFire);

auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);
std::cout << "-----" << duration.count() << std::endl;

return ThoroughFloodThoroughFire;
```

Voor de handmatige test hebben we bewust gekozen om dit eenmalig op een laptop te doen, omdat het anders erg veel tijd zou gaan kosten. De specificaties van deze laptop zijn een intel i7-7700HQ cpu met een kloksnelheid van 2.80 GHz, het heeft 16.0 GB geheugen met een snelheid van 2400 MHz en de grafische kaart is een NVIDIA GeForce GTX 1050.

We hebben er voor gekozen om ook een handmatige test te doen, waarbij elke keer de applicatie opnieuw wordt geopend en gesloten na een test, dit betekent dat we niet al te veel testen handmatig kunnen doen omdat het erg veel tijd kost. We hebben hiervoor gekozen, omdat we ook de resultaten wilde zien als een plaatje elke keer opnieuw werd geopend.

De handmatige testen die zijn uitgevoerd geven geen resultaat, omdat ze soms heel snel runnen en soms heel sloom.

Voor de volgende test zullen we wel gebruik maken van meerdere computers

computer 1: intel i7-7700HQ cpu met een kloksnelheid van 2.80 GHz, het heeft 16.0 GB geheugen met een snelheid van 2400 MHz en de grafische kaart is een NVIDIA GeForce GTX 1050.

Computer 2: Intel i3-6100 met een kloksnelheid van 3.70 GHz, het heeft 8 GB geheugen met een snelheid van 2133MHz en de grafische kaart is NVIDIA GeForce GTX 1050

Computer 3: intel i3-5005u met een kloksnelheid van 2.00 GHz, het heeft 4.0 GB geheugen met een snelheid van 1600 MHz en de grafische kaart is een intel HD Graphics 5500

1.1. Doel

Geef aan wat het doel van het experiment is, bijvoorbeeld in de vorm van een te controleren hypothese.

Het doel van het experiment is om te ontdekken welke edge detection methode het snelste is en of hardware er invloed op heeft.

1.2. Hypothese

Voordat je aan de proef begint stel je een hypothese op; wat verwacht je dat het antwoord zal zijn op je onderzoeksvraag?

Op basis van het onderzoek dat wij voor het implementatieplan hebben moeten doen verwachten wij dat de Sobel en Prewitt methode sneller zijn dan de standaard geïmplementeerde versie in de applicatie waarvan er gebruik wordt gemaakt, de laplacian methode.

1.3. Werkwijze

Geef een korte beschrijving van het experiment. (Het overschrijven van de practicumhandleiding is niet nodig.) Maak indien nodig een tekening van de proefopstelling, waarin grootheden kunnen worden aangegeven.

We zullen twee experimenten uitvoeren om te bewijzen welke het snelste is.

Het eerste experiment is een handmatige test waarbij we elke foto tien keer opnieuw laten runnen voor alle drie de methodes. We hebben hiervoor gekozen, omdat het misschien net iets andere resultaten zal opleveren omdat elke keer de applicatie wordt geopend en gesloten. Dit experiment wordt alleen op een computer uitgevoerd, omdat het anders heel veel tijd kost.

Om alle methodes een gelijke kans te geven, maken ze ook allemaal gebruik van dezelfde threshold.

De code ziet er zo uit:

Prewitt:

```

auto start = high_resolution_clock::now();

cv::Mat imageContainer;
HereBeDragons::HerLoveForWhoseDearLoveIRiseAndFall(src, imageContainer);

cv::Mat prewittx = (cv::Mat_<float>(3, 3) << 1, 0, -1, 1, 0, -1, 1, 0, -1 );
cv::Mat prewitty = (cv::Mat_<float>(3, 3) << 1, 1, 1, 0, 0, 0, -1, -1, -1 );

cv::Mat prewittxResult;
cv::Mat prewittyResult;

filter2D(imageContainer, prewittxResult, -1, prewittx, cv::Point(-1, -1));
filter2D(imageContainer, prewittyResult, -1, prewitty, cv::Point(-1, -1));

IntensityImage* edgeDetectionImage = ImageFactory::newIntensityImage();

cv::Mat weighted;
cv::addWeighted(prewittxResult, 3, prewittyResult, 3, 0, weighted, -1);

HereBeDragons::NoWantOfConscienceHoldItThatICall(weighted, *edgeDetectionImage);

auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);
std::cout << "-----" << duration.count() << std::endl;

return edgeDetectionImage;

```

Sobel:

```

auto start = high_resolution_clock::now();

cv::Mat imageContainer;

HereBeDragons::HerLoveForWhoseDearLoveIRiseAndFall(src, imageContainer);

cv::Mat sobelx = (cv::Mat_<float>(3, 3) << -1, 0, 1, -2, 0, 2, -1, 0, 1);
cv::Mat sobely = (cv::Mat_<float>(3, 3) << -1, -2, -1, 0, 0, 0, 1, 2, 1);

cv::Mat sobelxResult;
cv::Mat sobelyResult;

filter2D(imageContainer, sobelxResult, -1, sobelx, cv::Point(-1, -1));
filter2D(imageContainer, sobelyResult, -1, sobely, cv::Point(-1, -1));

IntensityImage* edgeDetectionImage = ImageFactory::newIntensityImage();

cv::Mat weighted;
cv::addWeighted(sobelxResult, 3, sobelyResult, 3, 0, weighted, -1);

HereBeDragons::NoWantOfConscienceHoldItThatICall(weighted, *edgeDetectionImage);

auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);
std::cout << "-----" << duration.count() << std::endl;

return edgeDetectionImage;

```

Laplacian:

```
auto start = high_resolution_clock::now();

cv::Mat OverHillOverDale;

HereBeDragons::HerLoveForWhoseDearLoveIRiseAndFall(src, OverHillOverDale);

cv::Mat ThoroughBushThoroughBrier = (cv::Mat_<float>(9, 9) << 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
                                                                    0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
                                                                    -4, -4, -4, 1, 1, 1, 1, 1, 1, -4, -4, -4, 1, 1, 1, 1, 1, 1,
                                                                    -4, -4, -4, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
                                                                    1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0);

cv::Mat OverParkOverPale;

filter2D(OverHillOverDale, OverParkOverPale, CV_8U, ThoroughBushThoroughBrier, cv::Point(-1, -1), 0, cv::BORDER_DEFAULT);
IntensityImage* ThoroughFloodThoroughFire = ImageFactory::newIntensityImage();

HereBeDragons::NoWantOfConscienceHoldItThatICall(OverParkOverPale, *ThoroughFloodThoroughFire);

auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);
std::cout << "-----" << duration.count() << std::endl;

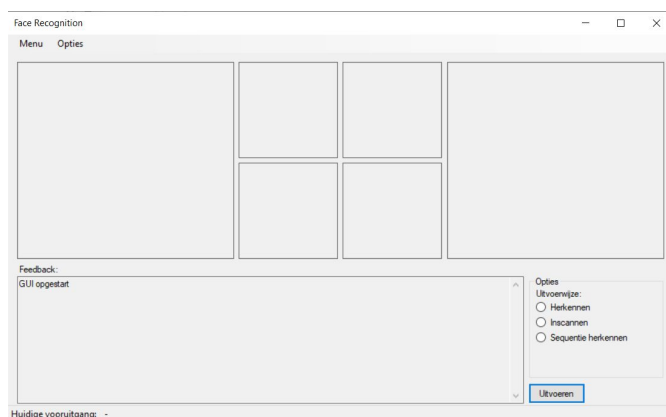
return ThoroughFloodThoroughFire;
```

We maken hiervoor gebruik van de library chrono uit de c++ standaard, deze zal zodra de functie edge detection wordt aangeroepen een start punt opvragen, en aan het einde het volgende punt opvragen en dat dan min elkaar doen om de tijdsduur te meten.

We kunnen nu de applicatie uitvoeren, dit doen we door op start te drukken. Dat is te vinden boven aan het beeld in visual studio.



Als we dat gedaan hebben, start de applicatie en krijgen we een scherm die er zo uit ziet:



Vervolgens willen we een foto selecteren, dat doen we door op Menu te klikken en vervolgens op “Kies Afbeelding” dat zal daarna nog een scherm openen waar onze testsets in staan



Als we een van de foto's hebben gekozen gaan we naar de opties en kiezen we de algoritme opties

Algoritme opties


Welk algoritme wordt er gebruikt?	Default	Student
ImageShells	<input checked="" type="radio"/>	<input type="radio"/>
Stap #1: Pre-processing		
Converteer naar Intensity	<input checked="" type="radio"/>	<input type="radio"/>
Scaling	<input checked="" type="radio"/>	<input type="radio"/>
Edge detection	<input type="radio"/>	<input checked="" type="radio"/>
Thresholding	<input type="radio"/>	<input checked="" type="radio"/>
Stap #2: Localiseren		
Hoofd, links en rechts	<input checked="" type="radio"/>	<input type="radio"/>
Neus, mond en kin	<input checked="" type="radio"/>	<input type="radio"/>
Kincontouren	<input checked="" type="radio"/>	<input type="radio"/>
Neusvleugels, linker- en rechteroog	<input checked="" type="radio"/>	<input type="radio"/>
Ogen	<input checked="" type="radio"/>	<input type="radio"/>
Stap #3: Extraheren		
Ogen	<input checked="" type="radio"/>	<input type="radio"/>
Neusgaten	<input checked="" type="radio"/>	<input type="radio"/>
Mondhoeken	<input checked="" type="radio"/>	<input type="radio"/>



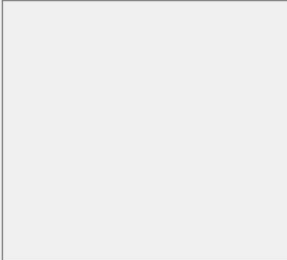
Opslaan

In dat nieuwe schermpje zorgen we er voor dat de Thresholding in alle gevallen die van ons is, dus de studenten versie. Verder nemen we de studenten edge detection als we sobel of prewitt willen testen en als we laplacian willen testen nemen we de default. Dat slaan we vervolgens op, daarna hoeven we alleen nog het vakje herkennen te selecteren en op uitvoeren te drukken rechts onderin het beeld zoals te zien is op de afbeelding hieronder.

Face Recognition

Menu Opties



Feedback:

GUI opgestart

Huidige vooruitgang: -

Opties

Uitvoerswijze:

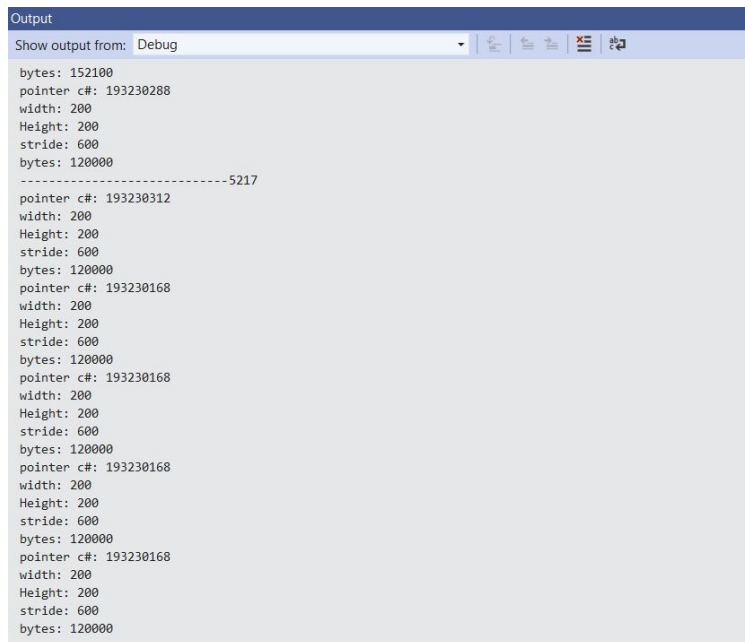
☒ Herkennen

☐ Inscannen

☐ Sequentie herkennen

Uitvoeren

Daar komt vervolgens een resultaat uit waar we niet echt heel erg in geïnteresseerd zijn, het is nu de bedoeling dat de applicatie wordt gesloten zodat we de gegevens die we nodig hebben te zien krijgen.



```
Output
Show output from: Debug

bytes: 152100
pointer c#: 193230288
width: 200
Height: 200
stride: 600
bytes: 120000
-----5217
pointer c#: 193230312
width: 200
Height: 200
stride: 600
bytes: 120000
pointer c#: 193230168
width: 200
Height: 200
stride: 600
bytes: 120000
pointer c#: 193230168
width: 200
Height: 200
stride: 600
bytes: 120000
pointer c#: 193230168
width: 200
Height: 200
stride: 600
bytes: 120000
pointer c#: 193230168
width: 200
Height: 200
stride: 600
bytes: 120000
```

Er zal dan een lijst aan getallen zichtbaar horen te worden in visual studio, zoals hierboven te zien is. We zijn alleen geïnteresseerd in een getal en dat is het getal met de stippelijntjes er voor, Dus in dit geval de 5217 zoals in de afbeelding hierboven staat.

Dit wordt dus handmatig tien keer per foto per methode uitgevoerd, dus 210 keer. Dat zetten we vervolgens in tabellen per foto met een gemiddeld resultaat er onder, zodat we kunnen vergelijken welke foto's het beter doen bij welke methode.

Het tweede experiment zal soortgelijk verlopen alleen hoeven we het geen 210 keer uit te voeren, maar per computer een keer per foto. Hiervoor hebben we een net iets andere code geschreven. Het staat nu in een for loop en zal 1000 keer runnen. Er kan niet vanuit gegaan worden dat de resultaten hetzelfde zullen zijn als in het eerste experiment, omdat de applicatie niet constant gesloten en geopend wordt, wat mogelijk invloed zou kunnen hebben op het eerste experiment.

De code voor dit experiment ziet er als volgt uit

Prewitt:

```
IntensityImage* edgeDetectionImage = ImageFactory::newIntensityImage();
microseconds totalDuration = milliseconds(0);

for (int i = 0; i < 1000; i++) {
    auto start = high_resolution_clock::now();

    cv::Mat imageContainer;
    HereBeDragons::HerLoveForWhoseDearLoveIRiseAndFall(src, imageContainer);

    cv::Mat prewittx = (cv::Mat_<float>(3, 3) << 1, 0, -1, 1, 0, -1, 1, 0, -1);
    cv::Mat prewitty = (cv::Mat_<float>(3, 3) << 1, 1, 1, 0, 0, 0, -1, -1, -1);

    cv::Mat prewittxResult;
    cv::Mat prewittyResult;

    filter2D(imageContainer, prewittxResult, -1, prewittx, cv::Point(-1, -1));
    filter2D(imageContainer, prewittyResult, -1, prewitty, cv::Point(-1, -1));

    cv::Mat weighted;
    cv::addWeighted(prewittxResult, 3, prewittyResult, 3, 0, weighted, -1);

    HereBeDragons::NoWantOfConscienceHoldItThatICall(weighted, *edgeDetectionImage);

    auto stop = high_resolution_clock::now();
    totalDuration += duration_cast<microseconds>(stop - start);
}

std::cout << "-----" << totalDuration.count() / 1000 << std::endl;

return edgeDetectionImage;
```

Sobel:

```
IntensityImage* edgeDetectionImage = ImageFactory::newIntensityImage();
microseconds totalDuration = milliseconds(0);

for (int i = 0; i < 1000; i++) {
    auto start = high_resolution_clock::now();

    cv::Mat imageContainer;

    HereBeDragons::HerLoveForWhoseDearLoveIRiseAndFall(src, imageContainer);

    cv::Mat sobelx = (cv::Mat_<float>(3, 3) << -1, 0, 1, -2, 0, 2, -1, 0, 1);
    cv::Mat sobely = (cv::Mat_<float>(3, 3) << -1, -2, -1, 0, 0, 0, 1, 2, 1);

    cv::Mat sobelxResult;
    cv::Mat sobelyResult;

    filter2D(imageContainer, sobelxResult, -1, sobelx, cv::Point(-1, -1));
    filter2D(imageContainer, sobelyResult, -1, sobely, cv::Point(-1, -1));

    cv::Mat weighted;
    cv::addWeighted(sobelxResult, 3, sobelyResult, 3, 0, weighted, -1);

    HereBeDragons::NoWantOfConscienceHoldItThatICall(weighted, *edgeDetectionImage);

    auto stop = high_resolution_clock::now();
    totalDuration += duration_cast<microseconds>(stop - start);
}

std::cout << "-----" << totalDuration.count() / 1000 << std::endl;

return edgeDetectionImage;
```

Laplacian:

```
IntensityImage* ThoroughFloodThoroughFire = ImageFactory::newIntensityImage();
microseconds totalDuration = milliseconds(0);

for (int i = 0; i < 1000; i++) {

    auto start = high_resolution_clock::now();

    cv::Mat OverHillOverDale;

    HereBeDragons::HerLoveForWhoseDearLoveIRiseAndFall(src, OverHillOverDale);

    cv::Mat ThoroughBushThoroughBrier = (cv::Mat_<float>(9, 9) << 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
        0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
        -4, -4, -4, 1, 1, 1, 1, 1, -4, -4, -4, 1, 1, 1, 1, 1,
        -4, -4, -4, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
        1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0);
    cv::Mat OverParkOverPale;

    filter2D(OverHillOverDale, OverParkOverPale, CV_8U, ThoroughBushThoroughBrier, cv::Point(-1, -1), 0, cv::BORDER_DEFAULT);

    HereBeDragons::NoWantOfConscienceHoldItThatICall(OverParkOverPale, *ThoroughFloodThoroughFire);

    auto stop = high_resolution_clock::now();
    totalDuration += duration_cast<microseconds>(stop - start);
}

std::cout << "-----" << totalDuration.count() / 1000 << std::endl;

return ThoroughFloodThoroughFire;
```

Om te kunnen kijken hoeveel invloed de hardware er op heeft hebben we gebruik gemaakt van drie computers met verschillende specificaties. Waarbij computer 1 over het algemeen het sterkste is gevolgd door computer 2 en daarna computer 3.

computer 1: intel i7-7700HQ cpu met een kloksnelheid van 2.80 GHz, het heeft 16.0 GB geheugen met een snelheid van 2400 MHz en de grafische kaart is een NVIDIA GeForce GTX 1050.

Computer 2: Intel i3-6100 met een kloksnelheid van 3.70 GHz, het heeft 8 GB geheugen met een snelheid van 2133MHz en de grafische kaart is NVIDIA GeForce GTX 1050

Computer 3: intel i3-5005u met een kloksnelheid van 2.00 GHz, het heeft 4.0 GB geheugen met een snelheid van 1600 MHz en de grafische kaart is een intel HD Graphics 5500

Om er ook nog achter te komen hoeveel invloed de hardware op de snelheid van de edge detection methodes heeft zullen we per methode en foto moeten kijken naar wat het resultaat is per computer. Dit resultaat zal het verschil tussen de tijd per computer zijn, vervolgens zullen we dit vergelijken tussen de methodes, waardoor we te zien krijgen welke methode veel beïnvloed wordt door de hardware en welke er minder om geeft.

1.4. Resultaten

De snelheids experiment levert deze resultaten op:

Computers:	Comput er 1			Compu ter 2			Comput er 3		
Methode:	Sobel	Prewitt	Laplacian	Sobel	Prewitt	Laplacian	Sobel	Prewitt	Laplacian
child-1.png	5502	5383	5555	4889	4866	4785	11578	11991	12466
female-1.png	5321	5165	5618	5004	4856	4839	11403	11882	12289
female-2.png	2167	2175	2314	2138	2245	2038	4914	5190	5338

female-3.png	5167	5153	5614	5018	5188	5141	11403	11800	12381
male-1.png	5371	5146	5539	5186	5361	5261	11569	11672	12502
male-2.png	5088	5177	5578	4841	4893	5259	11691	11669	12407
male-3.png	5128	5135	5530	5029	4584	5245	11536	11891	12090

Geef de meetresultaten overzichtelijk weer in de vorm van een tabel en/of diagram.

1.5. Verwerking

De resultaten van de snelheids experiment hebben we omgezet naar een gemiddelde percentage. Deze percentages zijn berekend door het gemiddelde te nemen van de resultaten die horen bij de computers en methodes per afbeelding.

	Sobel sneller dan Laplacian	Prewitt sneller dan Laplacian	Sobel sneller dan Prewitt
child-1.png	3,80991397	2,544964029	1,233556375
female-1.png	4,685198822	3,848787837	0,8054123711
female-2.png	5,109013993	0,8324661811	4,241240916
female-3.png	7,170650361	4,493925297	2,561608301
male-1.png	5,315014011	5,063348212	0,2395371961
male-2.png	7,511563367	6,923041538	0,5504162812
male-3.png	5,402664454	5,807496529	-0,3826119025
Gemiddeld percentage:	5,572002711	4,216289946	1,321308505

Zo is "Sobel sneller dan Laplacian" berekent als volgt in een excel sheet.

child-1.png	= (((D59+G59+J59) / 3) / ((B59+E59+H59) / 3) * 100) - 100
Gemiddeld percentage:	=SOM(L59:L65) / 7

Laat zien hoe je de meetresultaten verwerkt om een conclusie te kunnen trekken. Het is niet nodig om alle berekeningen op te schrijven, als je bijvoorbeeld maar laat zien welke formule(s) je gebruikt voor het verwerken van de meetresultaten en daar zo nodig één voorbeeldberekening aan toevoegt.

1.6. Conclusie

Uit de verwerking kunnen we zien dat Sobel gemiddeld 5,572002711% sneller is dan de Laplacian methode en 1,321308505% sneller is dan de Laplacian methode.

De Prewitt methode is dan 4,216289946% sneller dan de Laplacian methode.

Uit deze percentages kunnen we uitmaken dat uit de drie methodes, de Laplacian methode het slechtst presteert met daarna de Prewitt en als het snelst is de Sobel methode.

Geef aan welke conclusie kan worden getrokken uit de verwerking van de meetresultaten.

1.7. Evaluatie

Leg een verband tussen de getrokken conclusie en het doel van het experiment (en de hypothese). Ga daarbij ook in op bijvoorbeeld de meetonzekerheid als gevolg van de gebruikte meetmethoden of eventuele meetfouten.