# 1. Meetrapport Edge detection – snelheid 2

## 1.1. Namen en datum

Bianca Krieger en Marianne Delmaar

## 1.2. Doel

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

Het doel van dit experiment is uit te vinden of de 3x3 kernel sneller is dan de 5x5 kernel. Ook is het belangrijk dat het uiteindelijke resultaat van de edge detection nog voldoende is.

## 1.3. Hypothese

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

Ik verwacht dat de 3x3 kernel beduidend sneller is, maar dat op hetzelfde moment het uiteindelijke resultaat van de 5x5 kernel beter is.

## 1.4. 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.

Om te testen hoe snel mijn 5x5 kernel is, vergelijk ik met de BaseTimer de snelheid van de 5x5 kernel met de snelheid van de 3x3 kernel. Dit doe ik met de volgende code.

```
#include "Basetimer2.h"
#include "Basetimer2.cpp"
#include "Exectimer.h"

int i = 0;
	BaseTimer2 bt = BaseTimer2();
	bt.start();
	for (i; i < 10; i++){
	//Execute the four Pre-processing steps
	if (!executor->executePreProcessingStep1(true)) {
		std::cout << "Pre-processing step 1 failed!" << std::endl;
		return false;
	}

	if (!executor->executePreProcessingStep2(true)) {
		std::cout << "Pre-processing step 2 failed!" << std::endl;
		return false;
	}

	if (!executor->executePreProcessingStep3(true)) {
		std::cout << "Pre-processing step 3 failed!" << std::endl;
		return false;
	}
```

```
	ImageIO::saveIntensityImage(*executor->resultPreProcessingStep3,
ImageIO::getDebugFileName("Pre-processing-3.png"));

	if (!executor->executePreProcessingStep4(true)) {
		std::cout << "Pre-processing step 4 failed!" << std::endl;
		return false;
	}
	ImageIO::saveIntensityImage(*executor->resultPreProcessingStep4,
ImageIO::getDebugFileName("Pre-processing-4.png"));

	}
	bt.stop();
	std::cout << "test: " << bt.elapsedMicroSeconds() << "ms/m" << std::endl; int i =
0;
	BaseTimer2 bt = BaseTimer2();
	bt.start();
	for (i; i < 10; i++){
```

## 1.5. Resultaten

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

| 3x3 kernel | Meettijden per 10 keer | 5x5 kernel | Meettijden per 10 keer |
|---|---|---|---|
| | 582844 | | 2078780 ms |
| | 482506 | | 1190032 ms |
| | 338541 | | 1276822 ms |
| | 516995 | | 1261216 ms |
| | 495911 | | 928054 ms |

## 1.6. Verwerking

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.

Omdat elke meettijd per tien keer uitvoeren gemeten is, wordt het gemiddelde uitgerekend van alle meettijden / 50. Dit betekent dat ik vijf keer een meting heb gedaan van de tien keer uitgevoerde functie.

| 3x3 kernel | | 5x5 kernel | |
|---|---|---|---|
| Gemiddelde van 50 keer: | 37996,04 ms | Gemiddelde van 50 keer: | 134698,08 ms |

## 1.7. Conclusie

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

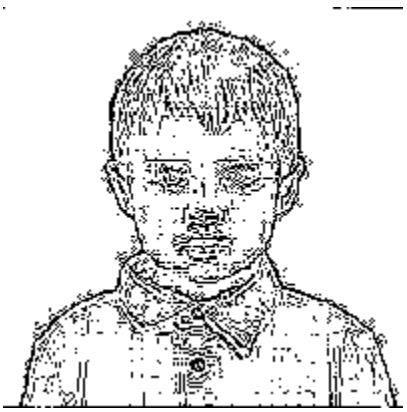
De 3x3 kernel is veel sneller dan de 5x5 kernel. De kwaliteit van de edge detection van de 3x3 kernel is helaas een stuk lager dan de kwaliteit van de edge detection van de 5x5 kernel.

## 1.8. 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.

De 3x3 kernel is een stuk sneller dan de 5x5 kernel. Dit komt omdat er minder rekenwerk zit in de 3x3 kernel. Het betekent helaas wel dat de edge detection van de 3x3 kernel een stuk minder accuraat is. Dit is ook te zien in de uitvoer van een image.

3x3 kernel



5x5 kernel