1. Meetrapport Edge detection - snelheid

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 de snelheid van de base implementatie vergelijken met mijn eigen implementatie.

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 mijn code langzamer zal zijn. Het is een hoop rekenwerk wat gedaan moet worden.

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 base implementatie met mijn eigen implementatie. 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.

Base implementatie	Meettijden per 10 keer	Eigen implementatie	Meettijden per 10 keer
	831254 ms		1162447 ms
	877984 ms		1277781 ms
	821338 ms		1252448 ms
	827127 ms		727041 ms
	816086 ms		1192572 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.

Base implementatie		Eigen implementatie	
Gemiddelde van 50 keer:	83475,78 ms	Gemiddelde van 50 keer:	112245,78 ms

1.7. Conclusie

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

De eigen implementatie is beduidend langzamer dan de base implementatie.

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.

Het is zo dat de base implementatie sneller is dan de eigen implementatie. Ik denk dat dit zo is omdat er in de eigen implementatie vaker inefficiënte code wordt gebruikt.

Base implementatie

Eigen implementatie