Meetrapport Memory Consumption

Jullian van Doorn en Jari van Dam 26-3-2018

Doel

Het doel van dit experiment is bepalen welke implementatie een kleinere memory footprint heeft. De implementaties zijn als volgd: OpenCV's edge detection en een eigen edge detection implementatie gebaseerd op Sobel. De onderzoeksvraag is als volgd:
Hoeveel bytes gebruikt de door studenten geïmplementeerde edge-detection techniek minder of meer dan de standaard optie die OpenCV aanbiedt?

Hypothese

Er wordt verwacht dat de student implementatie een kleinere memory footprint heeft dan de OpenCV canny edge detection.

Werkwijze

De memory consumption is aan de hand van soort van breakpoints gemeten. Er is een macro geschreven die de huidige memory consumption aan de Windows API vraagt en vervolgens wegschrijft in een bestand.
Er worden zeven unieke afbeeldingen van gezichten ingeladen gedurende de meting. De hele set wordt gedurende één sessie maal in serie ingeladen. De afbeeldingen zijn opgeslagen onder *"/testsets/Set A/TestSet Images"*. De volgorde van de afbeeldingen gaat als volgd:
child-1, female-1, male-1, female-2, male-2, female-3, male-3.
Er moeten meerdere sessies met metingen worden gedaan om kleine afwijkingen te middelen. Er zullen nu drie sessies van metingen plaatsvinden. Onder een sessie wordt het openen tot het sluiten van het programma verstaan.

Het onderstaande stuk code zorgt ervoor dat de macro *TRACE_MEMORY_USAGE* beschikbaar komt. Deze macro kan overal worden geplaatst waar de huidige memory usage moet worden weggeschreven. Deze macro is onder elke statement gezet die een grote impact kan hebben op de memory consumption. Met name grote geheugen allocaties en object instantiaties. Wanneer nu de DLL wordt gecompileerd zal hij zijn eigen memory usage telkens dumpen. Deze macro heeft wel invloed op de memory consumption met maar liefst enkele bytes. Maar wanneer deze macro meerdere keren in serie wordt gebruikt heeft dit geen opstapelend effect omdat de code in zijn eigen scope staat.

```
#pragma once

#include "windows.h"
#include "psapi.h"
#include "stdio.h"

#define MEMORY_USAGE_PROFILER_FILE_OUT "MemoryConsumption.log"

#define TRACE_MEMORY_USAGE { \
           PROCESS_MEMORY_COUNTERS_EX __pmc; \
           GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS*)&__pmc, sizeof(__pmc)); \
           SIZE_T __virtualMemUsedByMe = __pmc.PrivateUsage; \
           FILE* f; \
           fopen_s(&f, MEMORY_USAGE_PROFILER_FILE_OUT, "a"); \
           printf("%d\n", __virtualMemUsedByMe); \
           fprintf(f, "%d\n", __virtualMemUsedByMe); \
           fflush(f); \
           fclose(f); \
}
```

Figuur 1. TRACE_MEMORY_USAGE

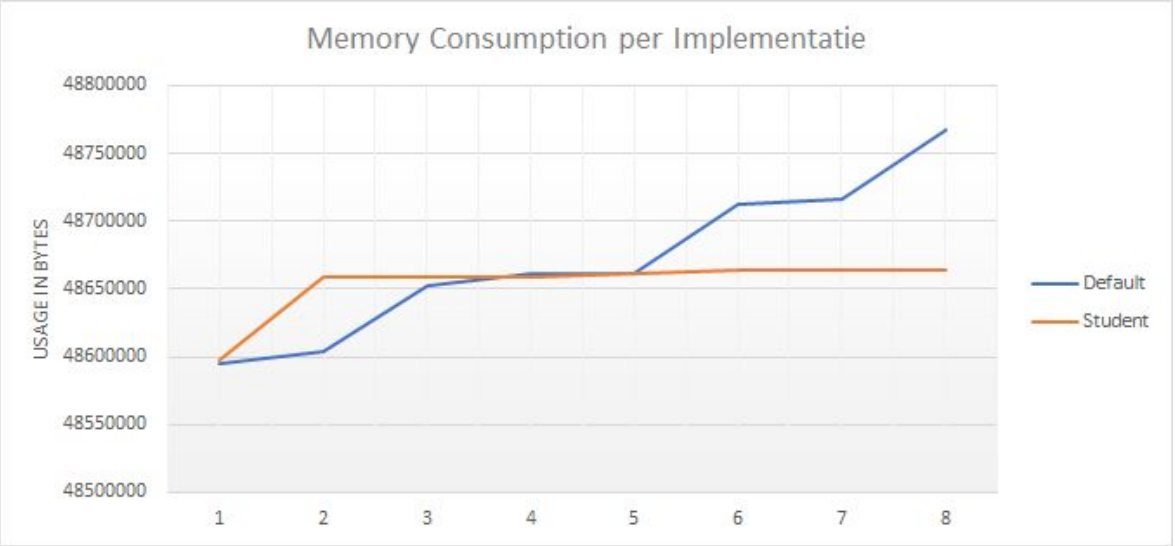
Het onderstaande script is gebruikt om een sessie uit te voeren. Dit schrijft verschillende dingen weg naar MemoryConsumption.log. Evenals de *TRACE_MEMORY_USAGE* macro. Er is een variant die dit voor de student implementatie doet. In dat geval zijn is al het “default” veranderd naar “student”. Dan zal het script de ExternalDLL configureren om de student implementatie te gebruiken.

```
@echo Default session begin >> MemoryConsumption.log
@echo Child 1 >> MemoryConsumption.log
Debug\ExternalDLL.exe "\TCTI-V2VISN1-13\testsets\Set A\TestSet Images\child-1.png" default
@echo Female 1 >> MemoryConsumption.log
Debug\ExternalDLL.exe "\TCTI-V2VISN1-13\testsets\Set A\TestSet Images\female-1.png" default
@echo Male 1 >> MemoryConsumption.log
Debug\ExternalDLL.exe "\TCTI-V2VISN1-13\testsets\Set A\TestSet Images\male-1.png" default
@echo Female 2 >> MemoryConsumption.log
Debug\ExternalDLL.exe "\TCTI-V2VISN1-13\testsets\Set A\TestSet Images\female-2.png" default
@echo Male 2 >> MemoryConsumption.log
Debug\ExternalDLL.exe "\TCTI-V2VISN1-13\testsets\Set A\TestSet Images\male-2.png" default
@echo Female 3 >> MemoryConsumption.log
Debug\ExternalDLL.exe "\TCTI-V2VISN1-13\testsets\Set A\TestSet Images\female-3.png" default
@echo Male 3 >> MemoryConsumption.log
Debug\ExternalDLL.exe "\TCTI-V2VISN1-13\testsets\Set A\TestSet Images\male-3.png" default
@echo Default session end >> MemoryConsumption.log
pause
```

Figuur 2. Test script

Resultaten

Aan de hand van de aangegeven werkwijze zijn er resultaten opgesteld. Deze resultaten zijn in een grafiek geplaatst. Aangezien de student-implementatie minder TRACE_MEMORY_USAGE's heeft, is de student lijn uitgerekt over dezelfde tijdsduur.



Figuur 3. Grafiek gemiddelde memory usage traces over 21 images waarvan 7 unieke images

In de grafiek is te zien dat de student implementatie aan het begin al zijn variabelen alloceert. Terwijl de default implementatie incrementeel steeds meer variabelen krijgt. Dit is terug te zien in de code. De student implementatie alloceert namelijk als eerste een nieuwe, lege, image. Dan gaat hij aan de hand van de input deze image invullen. In de loop van dit process wordt er niks groots meer gealloceerd en blijft de memory consumption bijna gelijk.

Verwerking

In de grafiek in figuur 3 is te zien dat de student-implementatie qua piek memory consumption ruim 100KB lager ligt. Hoewel dit in een hedendaagse desktop omgeving verwaarloosbaar is, maakt het de hypothese correct.

Conclusie

Hoeveel bytes gebruikt de door studenten geïmplementeerde edge-detection techniek minder of meer dan de standaard optie die OpenCV aanbiedt? Het antwoord is doormiddel van het hercompileren van de DLL zodat deze dumpt hoeveel memory usage zijn eigen proces verbruikt gevonden.

Omdat de memory consumption variabel is aan de instellingen kunnen kan er alleen een estimate worden gegeven. Er zijn meerdere tests op dezelfde set images gedaan en gemiddeld is het verbruik ruim 100KB minder.

Evaluatie

De eerste insteek van dit meetrapport was om de Visual Studio heap profiler te gebruiken. Dit was een slechte keus omdat deze niet in unmanaged environments werkt en voornamelijk bedoeld is voor .NET omgevingen. Nu was het geen optie om C++ te compileren voor .NET dus moest er worden gezocht naar een alternatief.

Omdat er op Windows 10 werd gewerkt was er geen goede CLI optie om van één specifieke command de gebruikte resources te loggen. Er moest dus een alternatief voor beiden komen. Dit alternatief was dus TRACE_MEMORY_USAGE, die aan de hand van de Windows API ophaalt wat de memory usage is van zijn eigen proces. Dit kan dan worden gedumpt in een log file en konden de resultaten gewoon worden verwerkt.

Wat er in principe nog verbeterd kan worden aan de memory consumption zelf kan worden is de gebruikte datatypes, meer chars gebruiken in plaats van ints, en wellicht meer template-metaprogramming zodat er meer werk tijdens compilatie kan worden gedaan zodat het programma tijdens runtime minder variabelen bevat.