Zofia Jankowska, 405601,
Wykonano: 20.03.2024

Sprawozdanie 6.

Cel ćwiczenia

Celem ćwiczenia była implementacja dolnoprzepustowego zorientowanego filtru Gaussa o rozmiarze 3x5 w wersji 2D oraz separowalnej.

Wykonanie ćwiczenia

W celu wykonania ćwiczenia należało zmodyfikować pliki źródłowe aplikacji AdvancedConvolution, tak aby wykonywany był operator Gaussa z zadanymi współczynnikami, zakodowanymi jako wartości stałe w kodzie źródłowym kerneli.

Wykorzystano wagi wyznaczone na drodze wykonywania programu w Matlabie, podanego w instrukcji do ćwiczenia i dodano je do pliku Filtercoeff.h.

```
float GAUSSIAN_FILTER_3x5[3*5] = {
0.00342918917027285f, 0.0493526028129635f, 0.120046527317579f,
0.0493526028129635f, 0.00342918917027285f,
0.00834124702533771f, 0.120046527317579f, 0.292004228746064f,
0.120046527317579f, 0.00834124702533771f,
0.00342918917027285f, 0.0493526028129635f, 0.120046527317579f,
0.0493526028129635f, 0.00342918917027285f};

float GAUSSIAN_FILTER_3x5_pass1[3] =
{-0.222222165955144f, -0.540538852976128f, -0.222222165955144f};

float GAUSSIAN_FILTER_3x5_pass2[5] =
{-0.0154313551734758f, -0.222086768891117f, -0.540209509711154f,
-0.222086768891117f, -0.0154313551734756f};
```

Aby kod źródłowy aplikacji AdvancedConvolution wykorzystać dla filtru Gaussa o rozmiarze 3x5 należało wprowadzić zmiany.

W pliku AdvancedConvolution.hpp zadeklarowano zmienną *filterSize2* typu *cl_uint* w celu późniejszego użycia w pliku AdvancedConvolution.cpp.

W AdvancedConvolution.cpp zmodyfikowano metodę *int AdvancedConvolution::readInputImage(std::string inputImageName).*

Do warunku sprawdzającego czy typ filtra ma prawidłową wartość, dodano informację, że jeśli przyjmuje wartość 3 jest to filtr Gaussa 3x5 i aplikacja nie powinna zwracać błędu:

```
if (filterType !=0 && filterType != 1 && filterType !=2 && filterType !=3 )
{
std::cout << "Filter Type can only be 0, 1 or 2 or 3 for Sobel, Box,
Gaussian and Gaussian 3x5 filters respectively." << std::endl;
return SDK_EXPECTED_FAILURE;
}
```

Ponadto, dodano fragment kodu odpowiedzialny za przypisanie maski, rozmiarów filtra oraz wartości filtrów dla wiersza i kolumny dla filtru 3x5 oraz w pozostałych przypadkach przypisano wartość zmiennej *filterSize2 = filterSize* w celu uniknięcia późniejszych błędów:

```
switch (filterType)
{
case 0: /* Sobel Filter */
 ...
filterSize2 = filterSize;
break;

case 1: /* Box Filter */
...
filterSize2 = filterSize;
break;

case 2: /* Gaussian Filter */
...
filterSize2 = filterSize;
break;

case 3: /* Gaussian Filter 3x5 */
mask = GAUSSIAN_FILTER_3x5;
rowFilter = GAUSSIAN_FILTER_3x5_pass1;
colFilter = GAUSSIAN_FILTER_3x5_pass2;
filterSize2 = 5;
filterSize = 3;
break;
}
```

Zmodyfikowano bufor maski oraz flitra kolumny, tak aby miały one rozmiary uwzględniające prostokątny rozmiar okna:

```
maskBuffer = clCreateBuffer(context, CL_MEM_READ_ONLY |
CL_MEM_USE_HOST_PTR, sizeof(cl_float ) * filterSize * filterSize2,
mask, &status);

colFilterBuffer = clCreateBuffer(context, CL_MEM_READ_ONLY |
CL_MEM_USE_HOST_PTR, sizeof(cl_float ) * filterSize2,
colFilter, &status);
```

Tak zmodyfikowane pliki zbudowano, a następnie przystąpiono edycji kernela. Aby dostosować implementację do prostokątnego filtra, w każdym wystąpieniu zmiennej *FILTERSIZE* sprawdzano, czy odnosi się ona do kolumn czy wierszy i jeśli dotyczyła ona wierszy, zmieniano ją na *FILTERSIZE2.*

Po zakończeniu modyfikacji kernela, uruchamiano aplikację z flagą *-f 3* i sprawdzono jej działanie zarówno na GPU (NVIDIA GeForce RTX 3050 Ti Laptop) oraz na CPU (11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz) dla obrazu *Cameraman* w rozmiarach 256x256, 1024x1024 oraz 4096x4096.
Poniżej przedstawiono wyniki filtracji dla obrazu 256x256 wykonanej na GPU.

Rysunek 1. Obraz *Cameraman* w oryginale

Rysunek 2. Obraz po zastosowaniu filtra nieseparowalnego

Rysunek 3. Obraz po zastosowaniu filtra separowalnego

Jak można zauważyć, filtr został poprawnie zastosowany, jednak dla filtru separowalnego obraz jest ciemniejszy, problem ten jednak nie występuje na CPU.

Porównano czas wykonywania algorytmów na GPU i CPU oraz w Matlabie, a wyniki przedstawiono w tabeli:

	Filtr separowalny			Filtr nieseparowalny		
Rozmiar	GPU	CPU	Matlab	GPU	CPU	Matlab
256x256	1.78955e-05	0.000184204	0.049936	1.51855e-05	0.00013689	0.049965
1024x1024	0.000110498	0.00201831	0.059413	0.000125415	0.00154028	0.057039
4096x4096	0.0022134	0.0264436	0.185842	0.00171428	0.0192126	0.184835

Filtr separowalny był wolniejszy od nieseparowalnego w 8/9 przypadkach.

Zastosowanie filtra o zróżnicowanych wymiarach

Taki filtr można by zastosować w przypadku, w którym poziom zaszumienia jest bardziej widoczny wzdłuż jednej z osi. Dzięki temu istnieje możliwość skutecznej redukcji szumów wykorzystując mniejsze zasoby obliczeniowe.