Grupa 1

Zadanie 3 - Rozmycie Gaussa w CUDA

Celem zadania było stworzenie programu rozmywającego podane na wejściu obrazu, za pomocą algorytmu Gaussa z maską 5x5. Do zrównoleglenia obliczeń należało wykorzystać technologię CUDA.

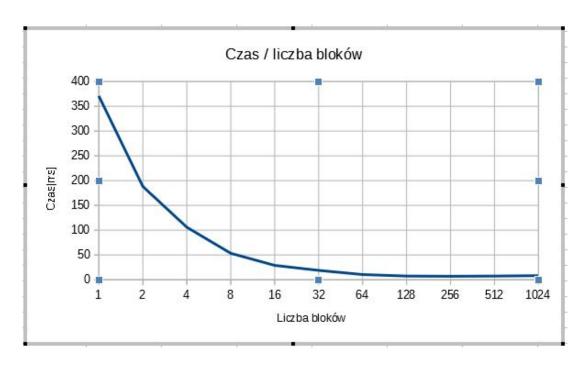
```
cudaMalloc<unsigned char> (&cudaSrcImage, totalImageSize);
       cudaMalloc<unsigned char> (&cudaOutImage, totalImageSize);
2
3
       cudaMemcpy(cudaSrcImage, inputImg.ptr(), totalImageSize,
4
           cudaMemcpyHostToDevice);
       \dim 3 \operatorname{dim} \operatorname{Grid} = \dim 3 (\operatorname{GRID}_{\operatorname{SIZE}});
6
       \dim 3 \dim Block = \dim 3(BLOCK\_SIZE);
       cudaEventCreate(&startEvent);
9
       cudaEventCreate(&endEvent);
10
11
       cudaEventRecord(startEvent, NULL);
12
       gaussianBlurCuda <<<dimGrid, dimBlock >>> (cudaSrcImage, cudaOutImage,
13
           inputImg.rows, rowLengthInBytes, inputImg.channels());
       cudaEventRecord(endEvent, NULL);
14
       cudaEventSynchronize(endEvent);
15
       cudaEventElapsedTime(&elapsedTime, startEvent, endEvent);
16
17
       cudaMemcpy(outputImg.ptr(), cudaOutImage, totalImageSize,
18
           cudaMemcpyDeviceToHost);
```

Na powyższym fragmencie kodu widać wykorzystanie API Cudy.

```
int col = (byteCounter % rowSizeInBytes) / channelsSize;
1
2
          if (byteCounter < 2 * rowSizeInBytes) continue;
          if (byteCounter > totalImageSize - 2 * rowSizeInBytes) continue;
4
5
          if (col < 2) continue;
          if (col >= (rowSizeInBytes / channelsSize) - 2) continue;
          int sum = 0;
9
10
          for (int i = 0; i < 5; i++)
11
12
               for (int j = 0; j < 5; j++)
13
                   sum += globalMask[i][j] * imageSrc[byteCounter + (i - 2) *
15
                      rowSizeInBytes + (j - 2) * channelsSize];
16
17
          imageOut[byteCounter] = (int) (sum / deviceMaskWeight);
18
```

Powyższy listing przedstawia kod kernela, wykonywany równolegle przez wiele wątków na GPU.

Obliczenia wykonywane były na uczelnianym serwerze. Do testów używano obrazka w rozdzielczości 4K. Jak widać na podstawie wykresów, dzięki wykorzystaniu technologii CUDA, można wykorzystać moc kart obliczeniowych. Ich architektura sprawia, że są one



Rysunek 1: Wykres zależności czasu wykonywania obliczeń od liczby bloków



Rysunek 2: Wykres przyspieszenia działania programu w zależności od liczby bloków

wręcz idealne do pracy nad zadaniami, w których istotnym problemem są takie same, proste obliczenia wykonywane wielokrotnie na olbrzymim zbiorze danych.