

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.

```

1  cudaMalloc<unsigned char> (&cudaSrcImage , totalImageSize);
2  cudaMalloc<unsigned char> (&cudaOutImage , totalImageSize);
3
4  cudaMemcpy(cudaSrcImage , inputImg.ptr() , totalImageSize ,
5             cudaMemcpyHostToDevice);
6
7  dim3 dimGrid = dim3(GRID_SIZE);
8  dim3 dimBlock = dim3(BLOCK_SIZE);
9
10 cudaEventCreate(&startEvent);
11 cudaEventCreate(&endEvent);
12
13 cudaEventRecord(startEvent , NULL);
14 gaussianBlurCuda<<<dimGrid , dimBlock>>>(cudaSrcImage , cudaOutImage ,
15      inputImg.rows , rowLengthInBytes , inputImg.channels());
16 cudaEventRecord(endEvent , NULL);
17 cudaEventSynchronize(endEvent);
18 cudaEventElapsedTime(&elapsedTime , startEvent , endEvent);
19
20 cudaMemcpy(outputImg.ptr() , cudaOutImage , totalImageSize ,
21            cudaMemcpyDeviceToHost);

```

Na powyższym fragmencie kodu widać wykorzystanie API Cudy.

```

1  int col = (byteCounter % rowSizeInBytes) / channelsSize;
2
3  if (byteCounter < 2 * rowSizeInBytes) continue;
4  if (byteCounter > totalImageSize - 2 * rowSizeInBytes) continue;
5
6  if (col < 2) continue;
7  if (col >= (rowSizeInBytes / channelsSize) - 2) continue;
8
9  int sum = 0;
10
11 for (int i = 0; i < 5; i++)
12 {
13     for (int j = 0; j < 5; j++)
14     {
15         sum += globalMask[i][j] * imageSrc[byteCounter + (i - 2) *
16             rowSizeInBytes + (j - 2) * channelsSize];
17     }
18     imageOut[byteCounter] = (int) (sum / deviceMaskWeight);

```

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.