Wykrywanie krawędzi metodą Robertsa

Kamil Susek

Analiza zadania

$$W_{i,j} = \begin{pmatrix} a_{i,j} & a_{i+1,j} \\ a_{i,j+1} & a_{i+1,j+1} \end{pmatrix}$$

$$Z_{i,j} = |a_{i,j} - a_{i+1,j+1}| + |a_{i+1,j} - a_{i,j+1}|$$

Algorytm Robertsa wykorzystuje okno o rozmiarze 2x2 piksele (macierz W). Na podstawie wartości pikseli w oknie obliczana jest nowa wartość piksela, która zapisywana jest w tablicy wynikowej (macierz Z).

Realizacja zadania

Wykorzystanie rozkazów wektorowych w algorytmie Robertsa wymagało dopasowania zbioru danych. W tym celu tablica P zawierająca przetwarzany obraz została podzielona według następującej zależności:

Dla każdego i = 0, 1, 2, ..., 2 * WYSOKOŚĆ_OBRAZU * SZEROKOŚĆ_OBRAZU

$$A_{2i} = P_i, A_{2i+1} = P_{i+1}$$
$$B_{2i} = P_{i+1+W}, B_{2i+1} = P_{i+W}$$

Realizacja zadania

Dane podzielone na tablice A i B są przetwarzane w następujący sposób:

$$Z_i = |A_i - B_i|$$

```
movdqu xmm1, [rbx + 0*SIZEOF BYTE]
movdqu xmm2, [rdx + 0*SIZEOF BYTE]

psubb xmm1,xmm2

pabsb xmm1,xmm1

movdqu [rcx],xmm1;
```

Realizacja zadania ASM

```
operateOnPixelsAsm PROC

    movdqu xmm1, [rbx + 0*SIZEOF BYTE]      ; load vector 1
    movdqu xmm2, [rdx + 0*SIZEOF BYTE]      ; load vector 2

    psubb xmm1,xmm2                         ; subtract vectors

    pabsb xmm1,xmm1                         ; get absolute value

    movdqu [rcx],xmm1                       ; load absolute value to result array

    ret
operateOnPixelsAsm ENDP
```

Realizacja zadania CPP

```
extern "C"  void  __declspec(dllexport)  operateOnPixelsCpp(unsigned char *pixels, unsigned char *copy1, unsigned char *copy2)
{
    //vector a and b
    __m128i a, b;
    // loading 128 bits of tab copy1 and copy 2
    a = _mm_loadu_si128((__m128i*)copy1);
    b = _mm_loadu_si128((__m128i*)copy2);
    // subtract b from a and get absolute value, next store result in pixels array
    _mm_storeu_si128((__m128i*)pixels, (_mm_abs_epi8(_mm_sub_epi8(a, b))));
}
```

Podsumowanie

Efektywne wykorzystanie rozkazów wektorowych wymaga dopasowania zbioru danych. Koszt dopasowania danych czasami może być większy, niż zysk wynikający z wykorzystania rozkazów SIMD.

Najbardziej optymalne wyniki czasowe zostały uzyskane, gdy liczba aktywnych wątków nie przekraczała maksymalnej liczby wątków procesora.