Grupa 4	Barbara Nowak, Piotr Tomaszewski	26/10/2016

Zadanie 4 - Rozmycie Gaussa w OpenMP

Celem naszego zadania było zaimplementowanie programu rozmywającego wybrane zdjęcie za pomocą algorytmu Gaussa z maską o wymiarach 5x5. Do zrównoleglenia obliczeń wykorzystaliśmy bibliotekę OpenMP.

Program uruchamiany jest z dwoma argumentami: „ilWatkow "- to liczba wątków a pozostałe dwa to odpowiednio lokalizacja pliku źródłowego oraz ścieżka do zapisu zmodyfikowanego pliku. Początkowo sprawdzamy poprawności wprowadzanych danych. Kolejnym krokiem jest wczytanie pliku do zmiennej „zdjecie "typu „Mat "za pomoca funkcji imread.

```
zdjecie = imread(argv[2], CV_LOAD_IMAGE_COLOR);
```

Następnie deklarujemy maskę w postaci 2 - wymiarowej tablicy liczb całkowitych, o wymiarach 5x5. Za pomocą funkcji smasek obliczamy sume masek, która jest niezbędna do prawidłowego wykonania rozmycia Gaussa na wczytanum obrazku.

```
int sMasek(int maska[5][5], int i, int j)
{
    int suma=0;
    for(i = 0; i < 5; ++i)
    {
          for(j = 0; j < 5; ++j)
        {
           suma += maska[i][j];     //Musi byc rozna od 0!
        }
    }
    return suma;
}
```

Tworzymy kopię wczytanego obrazka za pomocą funkcji „clone() ". Kolejnym krokiem jest ustawienie liczby watków dla zrównoleglonych obszarów.

```
omp_set_num_threads(ilWatkow);
```

Etapem końcowym jest zapisanie zmodyfikowanego obrazu do ścieżki z argumentu argv[3] za pomocą funkcji imwrite(), która jako pierwszy argument przyjmuje ścieżkę docelową, a drugim argumentem jest funkcja „rozmycie() "która rozmywa nasz obrazek za pomocą algorytmu Gaussa z maską 5x5.

```
imwrite(argv[3], rozmycie(sumaMasek,zdjecieKopia,zdjecie, maska));
```

W funkcji rozmycie() pierwszą czynnością jaką wykonujemy jest uruchomienie pomiaru czasu, następnie stosujemy procedurę zrónoleglania OpenMP w celu przyspieszenia obliczen.

```
#pragma omp parallel for default(shared) private(i,j,k,l,zmPiksel,r,g,b,zmI
    ,zmJ)
```

Powyższa dyrektywa zawiera następujące parametry: parallel - wskazanie kompilatorowi obszaru kodu, który będzie zrównoleglany, omp - słowo kluczowe odnoszące się do OpenMP, for - informuje że zrównoleglana będzie pętla for, default(shared) utawienie zasiegu zmiennych wewnatrz rownolegle przetwarzanego obszaru, (gdzie parametr śhared"powoduje, ze domyslnie wszystkie zmienne w petli sa wspolne), private (...) - okreslenie zmiennych prywatnych (do ktorych tylko jeden wątek ma dostep.

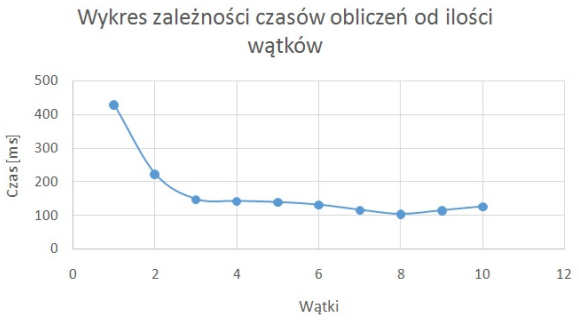
Następnie poruszając się po wierszach i kolumnach wczytanego obrazka, nawigujemy aby wspolrzedne tymczasowe nie wykroczyly poza obraz. Pobieramy wartosci każdego pixela w postaci 3 bitowego wektora (RGB) by następnie obliczyć dla niego nową wartość, kontrolując wartości by nie wyszły poza zakres [0 - 255].

```
// pobranie wartosci pixela w postaci 3 bitowego wektora (RGB)
zmPiksel = zdjecie.at<Vec3b>(Point(zmI, zmJ));
...
r+=zmPiksel.val[0]*maska[i][j];
g+=zmPiksel.val[1]*maska[i][j];
b+=zmPiksel.val[2]*maska[i][j];
...
((r/sumaMasek)>255)? (r=255) : (r=r/sumaMasek);
((g/sumaMasek)>255)? (g=255) : (g=g/sumaMasek);
((b/sumaMasek)>255)? (b=255) : (b=b/sumaMasek);
```

Pod koniec nanosimy nowe wartości pixeli RGB.

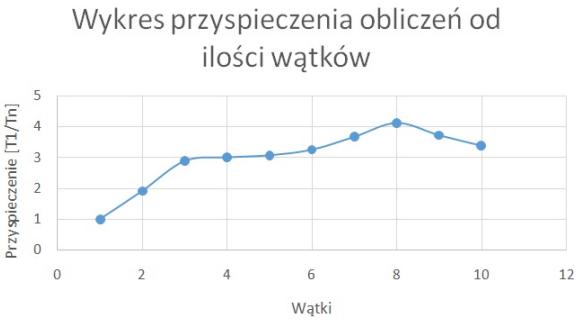
```
zdjecieKopia.at<Vec3b>(Point(l,k))=Vec3b(r,g,b);
```

Wykres zależności czasów obliczeń od ilości wątków (Rysunek 1) ukazuje, że ze wzrostem liczby wątków czas obliczeń znacznie spada. Przy czterech wątkach czas lekko wzrasta, potem płynnie opada aż do 8 wątków, po osiągnięciu których znów wzrasta.



Rysunek 1: Wykres zależności czasu obliczeń od liczby wątków

Zrównoleglenie przy użyciu OpenMP obrazuje poniższy wykres przyspieszenia od ilości watków (Rysunek 2). Program przyspiesza do ośmiu wątków a następnie zwalnia. Serwer uruchomieniowy ma 4-rdzeniowy procesor z technologią Hyper-Threading dlatego przy większej ilości wątków niż ma procesor, czas lekko rośnie.



Rysunek 2: Wykres przyspieszenia

Podsumowanie: OpenMP to rozszerzenie dla języków C/C++ pozwalające mieć wpływ na programowanie wielowątkowe. Każdy z wątków może posiadać własny tymczasowy widok na wspólny obszar pamięci. Zmienne zawarte w bloku paralel można podzielić na private - zmienne do których tylko jeden wątek ma dostęp i shared - dostęp wielu wątków do zmiennych. Używanie większej ilości wątków niż jest to sprzętowo możliwe to nie najlepsze rozwiązanie. Można pomyśleć, że przy nieskończonej ilości wątków czas będzie dążył do zera - to nie jest prawda. Po przekroczeniu możliwości procesora, czas wykonywania programu wzrośnie. Algorytm Gaussa pozwolił nam w prosty i łatwy sposób dokonać rozmycia zadanej fotografii.