Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

Inżynieria Oprogramowania

Temat:

Optymalizacja czasowa

Janusz Gajda

Grupa 4, sekcja 1

HistogramStreching

Połączenie pętli wyszukiwania minimum i maksimum; dodanie warunku dla min = 0 i max = 255 – ponad 2x szybciej

Przeniesienie *fScale* przed pętle, razem z warunkiem *nMin != nMax* – pojedyncze miliseknudy

Zamiana dzielenia w *fVal* na mnożenie, co wymusiło zamianę licznika z mianownikiem w *fScale* – pojedyncze milisekundy

Połączenie *fVa*l z *nVal* – praktyczny brak wpływu

Jako iż mamy 255 możliwych wyników; możemy zastosować tablicowanie, czyli wcześniejsze wyliczenie wartości, a potem ich wyszukiwanie. W tym przypadku wprowadzamy tablicowanie dla wielkości obrazu większej niż 256 pikseli, ponieważ dla mniejszych obrazów szansa powtórzenia wartości jest mała i operacja tablicowania dałaby efekt odwroty od zamierzonego. Przy okazji pozbyliśmy się zmiennej *nVal* poprzez bezpośrednie przypisanie wyliczanej wartości do *pImage*. Zamieniona została także kolejność pętli iterujących przez obraz, aby zachować liniowy odczyt pamięci. – ponad 10x szybciej

Ostatnim krokiem była zmiana typów zmiennych; zmienna *fScale* została zmieniona na *const float*, wszystkie indeksy w pętlach na *unsigned int* – pomijalne zmiany czasowe

Łączny wynik – 25 razy szybciej

ImageFiltering

Usunięcie zerowania *pOutImg,* przekształcenie *GetIndex* na makrodefinicję, zamiana wewnętrznych pętli na ręczne wywołania i likwidacja nadmiarowych wywołań poprzez modyfikację dzielników, eliminacja wywołań dla rozmiarów 0xN i Nx0 – 2x szybciej

Usunięcie zerowania *fSum*, dodanie zmiennej index, aby nie wywoływać cały czas *GetIndex*, dodanie zmiennych *width* i *height* – znikomy efekt (pojedyncze milisekundy)

Zastąpienie dzielenia przez wagę przesunięciami bitowymi – 1.5x szybciej

Ostatnim krokiem jest zamienienie kolejnością pętli iterujących przez obraz, aby uzyskać liniowy odczyt pamięci – 2.5x szybciej

Łączny wynik – 8.5 razy szybciej

Matrix

Zastosowanie transpozycji jednej z macierzy pozwoliło skrócić czas o około 25%.

Wnioski

Wykonanie powyższych czynności pozwoliło na znaczące skrócenie czasu wykonania programów. Szczególnie efektywne było zastosowanie tablicowania; jednak to rozwiązanie najlepiej sprawuje się, gdy duży zbiór elementów ma stosunkowo mały zbiór rozwiązań.

Największym zaskoczeniem była dla mnie kolejność zachowanie liniowości odczytu pamięci. Tak prosta zmiana jak zmiana kolejności wywołania pętle w przypadku, gdy jedna pętla jest zagnieżdżona w drugiej może spowodować drastyczny wzrost wydajności.