Dominik ŻUREK, **Maciej WIELGOSZ**, Ernest JAMRO, Kazimierz WIATR
AKADEMIA GÓRNICZO-HUTNICZA, AKADEMICKIE CENTRUM KOMPUTEROWE CYFRONET,
ul. Nawojki 11, 30-950 Kraków

Implementacja w układach FPGA wybranych fragmentów metody szybkiej segmentacji obrazów

Inż. Dominik ŻUREK

Ukończył studia na AGH (2011),wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki na kierunku Elektronika i Telekomunikacja. Obecnie jest studentem I roku studiów II stopnia na tym samym kierunku. Pracuje z ACK Cyfronet AGH z działem zespołu Akceleracji Obliczeń. Jego zainteresowania naukowe dotyczą przetwarzania obrazów oraz akceleracji obliczeń.

e-mail: zurek@student.agh.edu.pl

Dr inż. Maciej WIELGOSZ

Ukończył studia na AGH (2005), wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki na kierunku Elektronika i Telekomunikacja. Obecnie jest doktorantem w Katedrze Elektroniki AGH i bierze czynny udział w pracach badawczych realizowanych w zespole rekonfigurowalnych systemów obliczeniowych. Jego zainteresowania naukowe dotyczą sprzętowej akceleracji obliczeń, kompresji obrazu i sieci neuronowych.

e-mail: wielgosz@agh.edu.pl

Dr inż. Ernest JAMRO

Ukończył studia na AGH na kierunku Elektronika oraz na University of Huddersfield (UK) na kierunku Elektronika i Telekomunikacja. Obronił pracę doktorską w 2001 roku na AGH na wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki. Aktualnie jest adiunktem w Katedrze Elektroniki na AGH. Jego zainteresowania naukowe to sprzętowa akceleracja obliczeń, niskopoziomowe przetwarzanie obrazów, sieci neuronowe.

e-mail: jamro@agh.edu.pl

Prof. dr hab. inż. Kazimierz WIATR

Studia AGH Kraków (1980), dr nauk technicznych (1987), dr habilitowany (1999) i profesor (2002). Profesor zwyczajny na Akademii Górniczo-Hutniczej oraz Dyrektor Akademickiego Centrum Komputerowego Cyfronet AGH. Prowadzone prace badawcze dotyczą komputerowego sterowania procesami, systemów wizyjnych, systemów wieloprocesorowych, układów programowalnych, rekonfigurowalnych systemów obliczeniowych i sprzętowych metod akceleracji obliczeń.

e-mail: wiatr@agh.edu.pl

Streszczenie

Prezentowane w pracy badania dotyczą segmentacji obrazów metodą wektorów wspierających (ang. Support Vector Machine – SVM). Metoda ta opiera się na grupie kilkunastu wektorów wspierających, które posiadają cechy wybranych obiektów w obrazie. Implementacja przedstawionej procedury klasyfikacji wektorów wspierających została wykona zarówno programowo w języku C++ na procesorze ogólnego przeznaczenia AMD AthlonII P320 Dual-Core2.10 GHz, jak i sprzętowo w języku VHDL. Moduł klasyfikacji wektorów wspierających został zaimplementowany w układzie Xilinx Spartan 6.

Słowa kluczowe: segmentacja obrazów, wektory wspierające, FPGA, układy rekonfigurowalne.

FPGA implementation of selected parts of the fast image segmentation algorithm

Abstract

The paper presents preliminary implementation results of image segmentation for the SVM (Support Vector Machine) algorithm. SVM is a dedicated mathematical formula which allows extracting selective objects from an input picture and assign them to an appropriate class. Consequently, a black and white images reflecting occurrence of the desired feature are derived from an original picture fed into the classifier. This work is primarily focused on the FPGA implementation aspects of the algorithm as well as on comparison of the hardware and software performance. A human skin classifier was used as an example and implemented both in AMD AthlonII P320 Dual-Core2.10 GHz and Xilinx Spartan 6 FPGA. It is worth emphasizing that the critical hardware components were designed using HDL, whereas the less demanding standard ones such as communication interfaces, FIFO, FSMs were implemented in HLL (High Level Language). Such an approach allowed both shortening the design time and preserving high performance of the hardware classification module. This work is a part of the Synat project embracing several initiatives aiming at creation of a repository of images to which are to be assigned descriptive name according to their contents. Such a database of tagged images will significantly reduce the search time, since only picture tags will be processed instead of images, so the process will involve simple string operations rather than image recognition. The project is a huge challenge due to an immense volume of data collected over the past years denoted today as the Internet resources. Therefore, the core part of the undertaking is to design and implement a classification system which should be both reliable and fast. In order to achieve the high performance of a search engine, the most computationally intensive operations are to be ported to hardware.

Keywords: picture segmentation, supportive vectors, FPGA, reconfigurable logic.

1. Wstęp

Pod pojęciem segmentacji obrazu rozumiemy dzielenie obrazu na mniejsze fragmenty, odpowiadające poszczególnym obiektom, które można w nim wyróżnić. Celem segmentacji jest zatem takie przetworzenie danych zawartych na obrazie, aby uzyskać jego podział, który pomoże w rozpoznaniu obiektów w nim zawartych i ich interpretacji. Zmniejsza ona również nadmiarową informację, grupuje piksele o zbliżonych cechach charakterystycznych oraz łączy piksele opisujące poszczególne elementy. Segmentacja zatem pozwala na wydzielenie pewnych obszarów obrazu spełniających określone kryteria jednorodności.

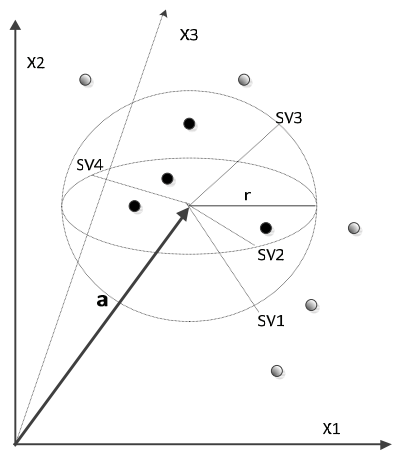
Opracowano wiele metod segmentacji obrazów i są one wciąż udoskonalane. O tym, czy dana metoda będzie szerzej wykorzystywana decydują przede wszystkim szybkość oraz skuteczność jej działania.

1.1. Support Vector Machine (SVM)

Charakterystyczne cechy obiektów poszukiwanych w obrazie zostają podane do modułu realizującego operację segmentacji w postaci wektora. Sposób ten jest użyteczny w sytuacjach, gdy ilość charakterystycznych punktów opisujących dany obiekt jest znacznie mniejsza niż ilość innych punktów, które są nieznane (nie możemy nic powiedzieć o ich cechach charakterystycznych, tj. kolor poziom szarości, tekstura etc.). W metodzie tej system OC-SVM (ang. One Class Support Vector Machine) jest trenowany cechami definiującymi obiekt, który będzie szukany. Cechami tego wektora mogą być elementy pochodzące z ortogonalnej bazy kolorów wraz z elementami pochodzącymi ze Structure Tensor (macierz gradientu funkcji, mówiąca o domniemanym kierunku gradientu w określonym otoczeniu punktu) lub, wektora kolorów RGB.

1.2. Klasyfikacja danych

Przetwarzanie danych realizowane jest w tzw. przestrzeń cech, w której klasyfikacja może być dokonana przy pomocy linearnej hiper-sfery (rys. 1). Dane są podawane w postaci OC–SVM, w której klasyfikator jest wytrenowany do rozpoznawania obiektów należących tylko do jednej klasy. Wspomniana hiper-sfera opisana jest przy pomocy centrum ***a*** i promienia ***r***. Jej objętość powinna jak najciaśniej zawrzeć w sobie wektor OC–SVM. Ta objętość jest wprost proporcjonalna do r^n. Dla uproszczenia obliczeń przyjmujemy, że minimalizacja r^n odbywa się ze szczególnym uwzględnieniem r^2. Za pomocą takiej realizacji zawsze, gdy podajemy wektor charakterystycznych cech obiektu OC–SVM, klasyfikator sprawdza tylko te punkty, które leżą na powierzchni hiper–sfery, natomiast pozostałe punkty po wytrenowaniu są odrzucane. Dzięki takiemu rozwiązaniu nie są sprawdzane wszystkie punkty, przez co uzyskiwana jest duża szybkość działania.



Rys. 1. Hiper-sfera wektorów wspierających
Fig. 1. N-sphere

Przynależność punktu do klasy poszukiwanego obiektu sprawdzana jest za pomocą zależności:

$$\sum_{i\in Idr(Sv)} \alpha_i K(X_x, X_i) \geq \sum_{i\in Idr(Sv)} \alpha_i K(X_s, X_i) = \tau \tag{1}$$

gdzie: τ - wartość obliczana jest za pomocą wektorów SVM, X_s jest to jeden z wektorów wspierających (SVM) uzyskanych w procesie uczenia klasyfikatora OC-SVM, X_x – jest to jeden z pikseli z wczytanego obrazu, X_i – są to kolejne wektory SVM.

Niezależnie od tego, który z wektorów zostanie użyty do obliczeń, wartość ta powinna pozostawać stała i wynosić τ. Prawa strona nierówności dotyczy testowanego punktu $X_{x,}$ natomiast suma po „i" dotyczy wszystkich wektorów wspierających. Współczynnik α jest wartością skalarną otrzymaną w procesie optymalizacji z tzw. mnożników Lagrange'a. Wyrażenie $K(x_s,x_i)$ określa jądro naszej hiper-sfery. W niniejszej implementacji używane jest Gaussowskie jądro, które określone jest za pomocą zależności:

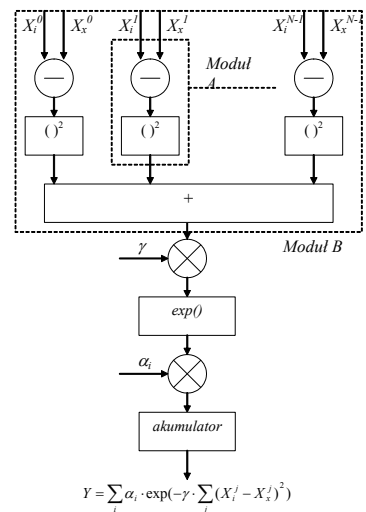
$$K = \mathrm{e}^{-\gamma \| X_i - X_j \|^2} \tag{2}$$

Parametr γ kontroluje tzw. "rozpiętość jądra". Jeśli nierówność (1) jest spełniona, to znaczy, że dany punkt klasyfikujemy jako należący do szukanego obiektu, jeśli nie, punkt klasyfikujemy jako tło. Z badań wynika, że w przypadku klasyfikatorów typu SVM znacznie lepsze wyniki uzyskuje się normalizując dane (w niektórych sytuacjach normalizacja jest zbędna) tak, aby mieściły się one w przedziale [-1;1].

2. Implementacja sprzętowa

2.1. Schemat

Sprzętowy moduł realizującego operację klasyfikacji składa się z pięciu modułów sprzętowych. Operacja klasyfikacji wykonywana jest zgodnie z zależnością (1) i (2). Wszystkie moduły są parametryzowane oraz potokowe, co pozwala na przetwarzanie danych co takt zegara oraz adaptację szerokości strumienia wejściowego (otrzymanego z procesora). Wszystkie dane, na których wykonywane są poszczególne operacje są w formacie stałoprzecinkowym. Podstawowym formatem liczbowym jest format 32-bitowy, 16-bitów przed przecinkiem i 16-bitów po przecinku. Każdy moduł posiada dodatkowe wejście gotowości transferu *strb* oraz przepełnienia *overflow*. Pierwszy z nich synchronizuje pracę modułów, natomiast sygnał *overflow* informuje o wystąpieniu przepełnienia.



Rys. 2. Schemat blokowy modułu
Fig. 2. Block diagram of the module

2.2. Moduł eksponenty (exp)

Algorytm sprzętowy realizujący funkcję wykładniczą exp() składa się z dwóch kroków: redukcji zakresu liczbowego, aproksymacji we wcześniej zdefiniowanym przedziale. Jest on wzorowany na podobnym module ale realizującym operacje na liczbach zmiennoprzecinkowych [5] a jego schemat blokowy został przedstawiony na rys. 3.

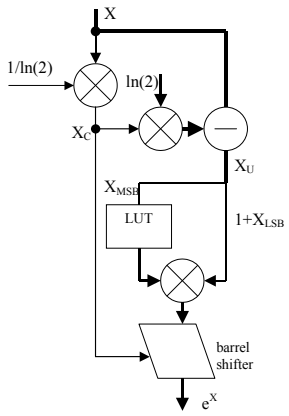
Redukcja zakresu danej wejściowej funkcji exp() związana jest z tym, że funkcja wykładnicza jest silnie malejąca (argument funkcji jest zawsze ujemny) i relatywnie niewielka (co do modułu) dana wejściowa może powodować ustawienie wyjścia w stan zero. Na przykład, dla wyjściowej reprezentacji: 16-bitów po przecinku, dana wejściowa może być reprezentowana tylko na 4 bitach całkowitych, jedynka na starszych bitach generuje stan wyjściowy równy zero.

Następną operacją jaką należy dokonać jest zamiana podstawy wykładnika z liczby e na liczbę 2. Jest to związane z tym, że wykonywanie operacji 2^{Xc} (gdzie X_c część całkowita liczby X) jest łatwa do obliczenia i jak to zostanie dalej pokazane jest zastępowane modułem szybkiego rejestru przesuwającego (ang. barrel shifter). Aby zmienić podstawę funkcji wykładniczej wystarczy pomnożyć daną wejściową przez 1/ln(2) zgodnie z poniższą tożsamością matematyczną:

$$\mathrm{e}^{x} = 2^{\frac{X}{\ln(2)}} \tag{3}$$

Otrzymany wynik mnożenia przez 1/ln(2) dzielimy na część ułamkową *Xu* oraz całkowitą *Xc*. Warto podkreślić, że mnożenie przez (1/ln(2)) można dokonać z małą dokładnością. Podobnie

mnożenie odwrotne przez ln(2) nie wymaga dużych zasobów sprzętowych ponieważ szerokość bitowa części całkowitej X_u jest niewielka [5].



Rys. 3. Schemat blokowy modułu obliczającego eksponentę
Fig. 3. Block diagram of the exponent module

Najbardziej skomplikowane jest obliczenie wartości eksponenty dla części ułamkowej, które bazuje na poniższym przekształceniu matematycznym:

$$e^{Xu} = e^{Xmsb+Xlsb} = e^{Xmsb} \cdot e^{Xlsb} \approx e^{Xmsb} \cdot (1+X_{lsb}) \quad (4)$$

gdzie: X_u - w pierwszym przybliżeniu część ułamkowa danej wejściowej X (zob. rys. 3), X_{msb} - najbardziej znaczące bity danej X_u, X_{lsb} - młodsze bity danej X_u.

Operacja e^{Xmsb} jest wykonywana z wykorzystaniem tablic LUT (Look-Up Table) implementowana w formie pamięci blokowej BRAM występującej w układach FPGA. Operacja e^{Xlsb} jest obliczana za pomocą przybliżenia Taylora: $e^{Xlsb} \approx 1+X_{lsb}$, jest to możliwe, ponieważ dana wejściowa X_{lsb} jest na tyle mała, że pominięcie następnych składników szeregu Taylora nie wpływa na poprawność otrzymanego wyniku. Omawiana metoda charakteryzuje się dużą szybkością działania oraz dokładnością ok. 18-bitów dla szerokości adresowej pamięci LUT równej 9-bitów (optymalnej ze względu na wielkość pamięci BRAM).

3. Wyniki implementacji

3.1. Implementacja sprzętowa

Implementacja sprzętowa została wykonana w układzie Xilinx Spartan 6 (XC6SLX75), a jej wyniki zostały przedstawione poniżej.

Tab. 1. Wyniki implementacji wybranych bloków składowych modułu
Tab. 1. Implementation results of the building blocks

Moduł	#LUT	#Przerzutniki
Moduł obliczający kwadrat różnicy (A)	660[1%]	220[1%]
Moduł obliczający skalar [j=3] (B)	1,953 [4%]	760 [1%]
Moduł exp()	312[1%]	277[1%]

Tab. 2 Wyniki implementacji modułu końcowego – realizującego klasyfikację
Tab. 2. Implementation results of the classification module

Liczba współrzędnych wektora [j]	LUT	Przerzutniki
2	2,689[5%]	1,048[1%]
3	3,377[7%]	1,311[1%]
5	6,319[13%]	1,834[2%]

Moduł A (zob. rys. 2) zajmuje niewielką ilość zasobów logicznych, co pozwala na wprowadzenie zrównoleglenia obliczań na poziomie modułów.

Prezentowany moduł może pracować z częstotliwością około 200 MHz, opóźnienie potokowe zależy od przyjętej parametryzacji: liczby współrzędnych wektora, szerokości ścieżki danych w ramach modułu.

3.2. Implementacja programowa

Wyżej opisana implementacja sprzętowa może być zastąpiona poprzez implementację programową na procesorze ogólnego przeznaczenia. Implementacja programowa została wykona w środowisku C++ przy użyciu biblioteki OpenCV, która wspomaga przetwarzanie obrazów. Na potrzeby projektu zostało wygenerowanych 17 wektorów wspierających posiadających cechy ludzkiej skóry, z których każdy posiada trzy współrzędne pochodzące z wektora kolorów RGB.

Realizacja wyodrębnienia obiektów ludzkiej skóry składa się z następujących elementów:
- wczytanie danych uczących,
- obliczanie na podstawie tychże danych wartości τ zgodnie z (1),
- wczytanie obrazu z dysku,
- przeskalowanie obrazu,
- klasyfikacja według wzoru(1),
- filtracja w celu wyeliminowania szumów oraz obiektów skóropodobnych.

Czas potrzebny na wykonanie wyżej opisanego algorytmu na procesorze ogólnego przeznaczenia: AMD Athlon II P320 Dual-Core 2.10 GHz (wykorzystywano tylko jeden rdzeń), dla obrazu o wymiarach 480×480 pikseli wynosił średnio około 0.015 s.

Sprzętowa implementacja klasyfikacji według (1) przy założeniu braku konieczności oczekiwania na dane wejściowe wymaga następującą liczbę taktów zegara: 17 (wektorów wspierających) × 480 (pikseli) ×480 (pikseli)≈ 4×10^6 taktów zegara. W rezultacie teoretyczny czas przetwarzania przy założeniu, że częstotliwość zegara wynosi 200MHz wynosi 0.02s. Dzięki niewielkiej zajętości zasobów możliwe będzie zaimplementowanie kilku lub kilkunastu tego typu modułów w FPGA, co pozwoli uzyskać przyspieszenie obliczeń.

4. Wnioski

W ramach projektu została zaimplementowana metoda klasyfikacji wektorów wspierających – SVM, programowo (z użyciem języka C++) jak również sprzętowo (VHDL). W artykule zostały przedstawione wstępne wyniki implementacji elementów składowych modułu oraz całej jednostki realizującej operację klasyfikacji. W następnym kroku moduł zostanie uruchomiony i przetestowany w systemie Pico M-502 [6] jako element składowy jednostki przetwarzania oraz klasyfikacji treści internetowych.

Praca finansowana ze środków NCBiR w ramach projektu Synat.

5. Literatura

[1] Rafajłowicz E., Rafajłowicz W., Rusiecki A.: Algorytmy przetwarzania obrazów i wstęp do pracy z biblioteką OpenCV, Wrocław 2009.
[2] Cyganek B.: Framework for Object Tracking with Support Vector Machines, Structural Tensor and the Mean Shift Method, LNCS 2009, vol. 5863/2009, s. 399-408.
[3] Wiatr K.: Akceleracja obliczeń w systemach wizyjnych, Warszawa 2003.
[4] Wielgosz M., Wiatr K.: Implementacja w układach FPGA wybranych operacji zmiennoprzecinkowych, Warszawa 2010, Akademicka oficyna wydawnicza EXIT.
[5] Wielgosz M., Jamro E., Wiatr K.: Highly Effcient Structure of 64-Bit Exponential Function Implemented in FPGAs, ARC 2008, Lecture notes in Springer-Verlag, London LNCS 4943, pp. 274 – 279.
[6] http://www.picocomputing.com/m_series.html

otrzymano / received: 13.05.2011
przyjęto do druku / accepted: 04.07.2011 *artykuł recenzowany*