Michał FULARZ, Marek KRAFT
POLITECHNIKA POZNAŃSKA, INSTYTUT AUTOMATYKI I INŻYNIERII INFORMATYCZNEJ,
ul. Piotrowo 3a, 60-965 Poznań

Implementacja w układzie reprogramowalnym wieloprocesorowego systemu realizującego algorytm RANSAC

Mgr inż. Michał FULARZ

Ukończył studia na kierunku Automatyk i Robotyka na Wydziale Elektrycznym Politechniki Poznańskiej w 2010 roku. Obecnie jest uczestnikiem studiów doktoranckich na tym samym wydziale. Jego zainteresowania naukowe obejmują cyfrowe przetwarzanie obrazów, systemy wbudowane, projektowanie i budowę robotów mobilnych, a także układy reprogramowalne.

e-mail: michal.fularz@cie.put.poznan.pl

Mgr inż. Marek KRAFT

Ukończył studia na kierunku Automatyka i Robotyka na Wydziale Elektrycznym Politechniki Poznańskiej w roku 2005. W tym samym roku rozpoczął pracę na stanowisku asystenta w Instytucie Automatyki i Inżynierii Informatycznej tejże uczelni. Jego główne zainteresowania naukowe obejmują projektowanie dedykowanych akceleratorów sprzętowych dla zastosowań w systemach wizyjnych, a także zagadnienia z zakresu przetwarzania obrazów, robotyki mobilnej, oraz projektowania systemów wbudowanych.

e-mail: marek.kraft@put.poznan.pl

Streszczenie

W artykule opisano programową, wieloprocesorową realizację algorytmu RANSAC, który umożliwia odporną estymację modelu matematycznego z danych pomiarowych zawierających znaczący odsetek wartości odstających (ang. outliers). System został zaimplementowany w układzie FPGA w oparciu o konfigurowalne soft procesory MicroBlaze. W pracy przedstawiono opis algorytmu RANSAC, sposób jego podziału w celu przetwarzania równoległego, a także proces konfiguracji systemu wieloprocesorowego. Zaprezentowano również przyrost prędkości przetwarzania w zależności od liczby zastosowanych rdzeni procesorowych, porównano te wyniki do realizacji na komputerze klasy PC i przedstawiono zużycie zasobów układu FPGA.

Słowa kluczowe: układy FPGA, sprzętowa implementacja, systemy wieloprocesorowe, RANSAC, macierz fundamentalna.

FPGA implementation of a multiprocessor system performing the RANSAC algorithm

Abstract

The paper describes a multiprocessor system implementing the RANSAC algorithm [3] which enables robust estimation of a fundamental matrix from a set of image keypoint correspondences containing some amount of outliers. The fundamental matrix encodes the relationship between two views of the same scene. The knowledge of the fundamental matrix enables e.g. the reconstruction of the scene structure. The implemented system is based on three MicroBlaze microprocessors [5] (one master, two slaves) and a dedicated hardware coprocessor connected using fast simplex link (FSL) interfaces [6]. The slave microprocessors perform the task of fundamental matrix computation from point correspondences using singular value decomposition – the so called 8-point algorithm [1, 2] (hypothesis generation). The master processor, along with the connected coprocessor, is responsible for dataflow handling and hypothesis testing using the Sampson error formula (7). The hypothesize and test framework used in RANSAC allows for largely independent task execution. The design is a development of a system described in [5]. The block diagram and dataflow diagram of the proposed solution are given in Figs. 1 and 2, respectively. Tabs. 1 and 2 summarize the use of FPGA resources. With a 100 MHz clock, the designed system is capable of processing the data at the speed which is roughly equivalent to that of the Atom N270 microprocessor clocked at 1,2 GHz. The resulting solution will be targeted at applications for which small size, weight and power consumption are critical. The design is also easily scalable – addition of more slave processors will result in additional increase in the processing speed.

Keywords: FPGA devices, hardware implementation, multiprocessor systems, RANSAC, fundamental matrix.

1. Wstęp

System wizyjny jest jednym z centralnych komponentów systemów sensorycznych współcześnie konstruowanych robotów mobilnych. Dane pomiarowe z systemów wizyjnych wykorzystywane są do wykonywania zadań takich jak śledzenie, autonomiczna nawigacja i konstrukcja mapy otoczenia czy rekonstrukcja trójwymiarowa. Danymi wejściowymi dla wyżej wymienionych algorytmów są najczęściej wyszukiwane i dopasowywane na kolejnych obrazach sekwencji cechy punktowe. Cechami nazywamy tu projekcje tego samego punktu charakterystycznego z obserwowanej sceny na dwa różne obrazy tej samej sceny. Dopasowane między widokami cechy reprezentowane są przez swoje połączone w pary współrzędne obrazowe.

W przypadku systemów przeznaczonych do pracy autonomicznej, algorytmy detekcji i dopasowywania cech działają w sposób automatyczny. Pomimo ciągłego udoskonalania, algorytmy te nie zapewniają stuprocentowej dokładności. W zbiorze dopasowanych cech w zdecydowanej większości przypadków znajduje się pewien odsetek błędnych dopasowań (ang. *outliers*).

W artykule opisano realizację wieloprocesorowego systemu realizującego algorytm RANSAC. System zaimplementowano w układzie programowalnym FPGA. Rozwiązanie proponowane w niniejszej pracy ma za zadanie obliczanie macierzy fundamentalnej (ang. *fundamental matrix)* $\boldsymbol{F}$, która wiąże ze sobą współrzędne obrazowe odpowiadających sobie cech w dwóch obrazach tej samej sceny. Do obliczania macierzy $\boldsymbol{F}$ wykorzystano algorytm 8-punktowy. Ze względu na swą specyfikę, algorytm RANSAC łatwo poddaje się dekompozycji na niezależne zadania, które można wykonywać równolegle w celu przyspieszenia jego działania. Macierz $\boldsymbol{F}$ można w kolejnych etapach przetwarzania wykorzystać do rekonstrukcji trójwymiarowej struktury sceny, lub też do określania ruchu własnego platformy mobilnej. Zaproponowane rozwiązanie może stanowić element systemu wizji komputerowej o niskim poborze mocy, małych wymiarach i wadze. Cechy te mają szczególne znaczenie w przypadku aplikacji mobilnych.

2. Algorytm 8-punktowy

Dla każdej pary prawidłowo dopasowanych cech $x \leftrightarrow x'$ spełniona jest zależność (1).

$$x'^{T} F x = 0 . \tag{1}$$

Symbolami x oraz x' oznaczono wektory reprezentujące jednorodne współrzędne obrazowe dopasowanych punktów, przy czym przez x oznaczono współrzędne w obrazie pierwszym, a przez x' współrzędne w obrazie drugim. Poszczególne składniki zależności (1) można zapisać jako:

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \quad x' = \begin{bmatrix} x'_1 \\ x'_2 \\ 1 \end{bmatrix} . \tag{2}$$

Po wykonaniu mnożenia zgodnie z (1), dla pojedynczej pary dopasowanych punktów otrzymujemy:

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \quad (3)$$

Elementy macierzy $\boldsymbol{F}$ zapisać można w postaci wektora kolumnowego $\boldsymbol{f}$ w porządku wiersz po wierszu. Równanie (1) można wtedy zapisać w formie (4).

$$\begin{bmatrix} x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \end{bmatrix} f = 0. \quad (4)$$

Dla zestawu *n* par punktów zapisać można układ równań (5).

$$Af = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} f = 0 \quad (5)$$

Macierz $\boldsymbol{F}$ ma rząd równy 2 i jest określona co do skali. W związku z tym, aby rozwiązać układ równań (5) potrzeba 8 par punktów – stąd nazwa algorytm 8-punktowy. W praktyce, ze względu na obecność szumu sensora, kwantyzacji, a także innych źródeł błędów, nie istnieje unikalne, jednoznaczne rozwiązanie układu równań (5). W związku z powyższym, układ równań rozwiązuje się w sensie najmniejszych kwadratów. Jedną z metod wykorzystywanych w tym celu jest dekompozycja na wartości osobliwe (dekompozycja SVD) macierzy $\boldsymbol{A}$. Szczegółowe informacje dotyczące algorytmu znaleźć można w [1] i [2].

3. Algorytm RANSAC

Jak wspomniano w rozdziale 1, jako dane wejściowe dla algorytmów wizyjnych (w tym algorytmu 8-punktowego) stosuje się najczęściej pary punktów wykryte i dopasowane w sposób automatyczny. Algorytm 8-punktowy w formie podstawowej nie jest odporny na zakłócenia związane z błędnym dopasowaniem punktów charakterystycznych (cech).

W celu rozwiązania tego problemu, algorytm wspomaga się dodatkowo algorytmem estymacji odpornej. W niniejszej pracy wykorzystano stosowany powszechnie w aplikacjach z dziedziny wizji komputerowej algorytm RANSAC [3].

RANSAC ((ang. Random SAmple Consensus) jest iteracyjnym algorytmem, który umożliwia estymowanie parametrów modelu matematycznego na podstawie zbioru danych pomiarowych, który zawiera pewien odsetek wartości odstających (ang. outliers).

Algorytm działa według następującego schematu:

- ze zbioru danych pomiarowych *S* wybierz losowo próbkę *s*; z wylosowanej próbki oblicz parametry modelu $\hat{M}_i$,
- określ moc zbioru danych, które dopasowane są do modelu $\hat{M}_i$ z błędem nie większym niż Δ_i, (moc zbioru S_i, będącego podzbiorem zbioru *S*),
- jeśli liczba pasujących do modelu, poprawnych danych (moc zbioru S_i) jest wyższa niż *T*, zwróć $\hat{M}_i$ i zakończ (ew. przeprowadź estymację modelu z wykorzystaniem danych ze zbioru S_i),
- jeśli moc zbioru S_i jest niższa niż *T*, powtórz wszystkie powyższe kroki,
- po przeprowadzeniu *N* prób wybierz zbiór S_i o największej mocy i zwróć odpowiadający mu model $\hat{M}_i$ (ew. przeprowadź estymację modelu z wykorzystaniem danych ze zbioru S_i).

W przypadku algorytmu 8-punktowego zbiorem *S* jest zbiór par dopasowanych punktów, próbką *s* jest osiem różnych par dopasowanych punktów, a modelem $\hat{M}_i$ są kolejne obliczane macierze $\boldsymbol{F}$.

Algorytm zwraca prawidłowy wynik z założonym prawdopodobieństwem, a wymagana minimalna liczba iteracji jest silnie zależna od procentowej zawartości wartości odstających w zbiorze pomiarowym. Jeżeli *s* jest rozmiarem próbki, *w* jest prawdopodobieństwem, że wylosowany element zbioru danych pomiarowych należy do danych poprawnych, $\varepsilon = 1-w$ jest prawdopodobieństwem, że wylosowany element należy do danych odstających (błędne dopasowanie), a *p* jest zakładanym prawdopodobieństwem, że estymacja da prawidłowy wynik, to należy przetestować przynajmniej *N* próbek, gdzie *N*:

$$N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)}. \quad (6)$$

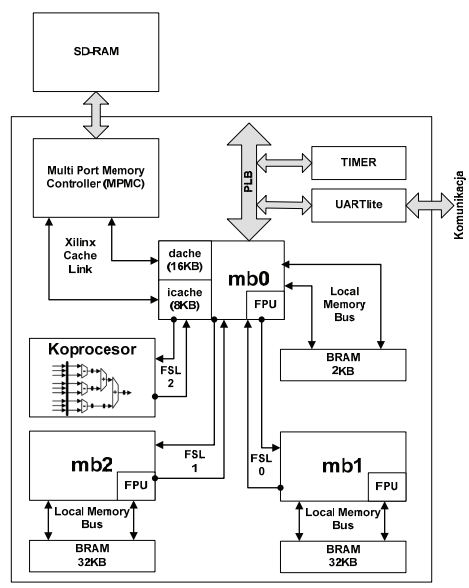
Jak wspomniano wyżej, algorytm RANSAC wymaga zastosowania miary dopasowania danych pomiarowych do modelu matematycznego. Dla celów implementacji wybrano błąd Sampsona, czyli aproksymację pierwszego rzędu błędu geometrycznego, wyrażoną wzorem (7).

$$\Delta_i = \frac{(x'^T_i Fx_i)^2}{(Fx_i)_1^2 + (Fx_i)_2^2 + (F^T x'_i)_1^2 + (F^T x'_i)_2^2}, \quad (7)$$

Symbolem $(Fx_i)_j$ oznaczono *j*-ty element wektora będącego wynikiem iloczynu.

4. Implementacja sprzętowa

Projekt jest rozszerzeniem prac zaprezentowanych w artykule [4]. Istniejący system został poddany szczegółowej analizie w celu zidentyfikowania etapów przetwarzania, która w największym stopniu wpływają na szybkość przetwarzania. Profilowanie kodu wykazało, że najdłużej wykonywany jest proces wyznaczania macierzy fundamentalnej, wykorzystujący algorytm SVD. Algorytm ten ma iteracyjny charakter, przez co wskazana jest realizacja programowa. Sposobem przyspieszenia wykonywania tego typu zadań jest dodanie dodatkowych rdzeni procesorowych ogólnego przeznaczenia. Jest to możliwe, ponieważ poszczególne hipotezy (kolejne macierze fundamentalne) mogą być generowane równolegle, niezależnie od siebie. Schemat blokowy tak zaprojektowanego systemu został zaprezentowany na rys. 1.



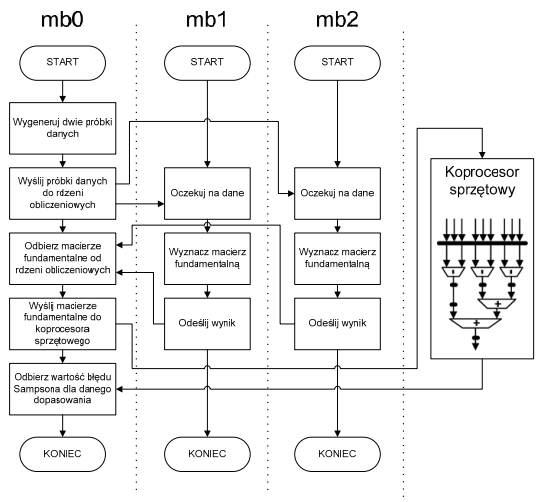
Rys. 1. Schemat blokowy systemu
Fig. 1. Block diagram of the system

Stworzony system wieloprocesorowy ma architekturę gwiazdy, w której nadrzędny procesor – mb0 wysyła do podrzędnych jednostek dane do przetwarzania, zbiera od nich wyniki i przekazuje wyniki obliczeń do nadrzędnego systemu komputerowego. Proces przetwarzania zaprezentowano na rys. 2. W roli procesorów ogólnego przeznaczenia (mb0, mb1, mb2) wykorzystano 32 bitowe mikroprocesory MicroBlaze [5].

Centralny procesor został wyposażony w 64 MB pamięci DDR2 SD-RAM z pamięcią podręczną 16 KB dla danych oraz 8 KB dla instrukcji. Układy peryferyjne (timer, interfejs RS232) zostały podłączone poprzez magistralę PLB (ang. *Processor Local*

Bus). Procesor wyposażono również w rozszerzoną jednostkę zmiennoprzecinkową (FPU), co przyspiesza ok. 20-krotnie operację konwersji typu całkowitego (*integer*) do zmiennoprzecinkowego (*float*). Komunikacja z pozostałymi procesorami odbywała się przy wykorzystaniu magistrali FSL (ang. *Fast Simplex Link*) [6]. Jest to jednokierunkowy, wykorzystujący buforowanie danych interfejsu typu punkt-punkt (ang. *point to point*), bez arbitrażu danych. W celu komunikacji dwukierunkowej niezbędne jest utworzenie dwóch kanałów komunikacyjnych.

Procesory mb1 oraz mb2 są identyczne i zostały wyposażone w podstawową jednostkę zmiennoprzecinkową (FPU) oraz dodatkowe moduły wspomagające operacje obliczeniowe (układ porównania wzorca, jednostkę skalująco-przesuwającą, układ mnożący i dzielący liczby całkowite). Ręczna optymalizacja kodu pozwoliła zmniejszyć jego rozmiar na tyle, że wraz z sekcją stosu i sterty mieści się w 32 kB pamięci złożonej z wewnętrznych bloków BlockRAM układu FPGA. Korzystnie wpływa to na prędkość wykonywania kodu, gdyż pamięć ta jest pamięcią statyczną. Zastosowany w układzie koprocesor sprzętowy, będący sprzętową implementacją wzoru (7) został szczegółowo opisany w pracy [4].



Rys. 2. Schemat przepływu danych w procesie przetwarzania
Fig. 2. Schematic diagram of the dataflow during processing

5. Analiza wyników

Jako platformy sprzętowej użyto płyty rozwojowej V5FXT-EVL wyposażonej w układ FPGA XC5VFX30T-1FFG665. W tabeli 1 zaprezentowano ilość zasobów wykorzystywanych przez cały system oraz poszczególne jego elementy. Należy zwrócić uwagę, że znaczną liczbę elementów BlockRAM pochłonęły bloki pamięci instrukcji oraz danych procesorów mb1 oraz mb2 (każdy po 32KB, czyli 8 elementów BlockRAM).

Tab. 1. Podsumowanie zużycia zasobów układu FPGA przez poszczególne bloki funkcjonalne i cały system
Tab. 1. Summary of FPGA resources used to implement the functional blocks and the whole system

	LUT	FF	BRAM	DSP48
Cały system	18220	13542	39	36
mb0	3408	2514	9	5
mb1	2264	1634	0	5
mb2	2264	1634	0	5
koprocesor	7257	4427	0	21

System pracuje z częstotliwością 100MHz i w ramach testów przetwarzał zbiór 206 par punktów pozyskanych i dopasowanych w sposób automatyczny z dwóch różnych, rzeczywistych obrazów tej samej sceny. W celu porównania prędkości przetwarzania wykonano 10000 operacji wyznaczenia oraz testowania modelu, co pozwoliło wyznaczyć średni czas wykonywania wymienionej pary operacji. Pierwsza wersja implementacji opisana w pracy [4] uzyskała wynik 11,8 ms. Zoptymalizowana, wieloprocesorowa wersja opisana w niniejszej pracy umożliwia przetwarzanie z prędkością 1,16 ms na iterację. Dla porównania energooszczędny mikroprocesor Intel Atom N270 1,6GHz z 1GB pamięci RAM uzyskuje wynik 0,93 ms. Należy zwrócić uwagę, że układ FPGA zużywa, według programu XPower Analyzer, 2,65 W energii, co predestynuje takie rozwiązanie do aplikacji o niskim poborze mocy.

Przedstawione w pracach [7] i [8] implementacje algorytmu RANSAC w układach FPGA wykorzystują inny model matematyczny, przez co bezpośrednie porównanie szybkości wykonywania do zrealizowanego systemu nie jest możliwe.

6. Wnioski

Zaprezentowany system oferuje zadowalającą prędkość przetwarzania, a dodatkowo cechuje się skalowalnością, dzięki czemu w łatwy sposób można przenieść go do większych układów FPGA.

Podsumowując, projektowanie wieloprocesorowych systemów połączonych z koprocesorami sprzętowymi oferuje dużą elastyczność w dostosowywaniu się do wymagań aplikacji. Dodatkowo, dzięki możliwości zastosowania istniejących już programowych bibliotek, skróceniu ulega czas projektowania układu. Należy zwracać szczególną uwagę na ograniczenia mikroprocesorów pod względem realizacji operacji zmiennoprzecinkowych, czy operacji działających na dużych zbiorach danych, przez co należy umiejętnie wybierać zadania przez nie wykonywane. Najważniejszym etapem procesu projektowania staje się w związku z tym etap podziału algorytmu na zadania i przydział tych zadań do poszczególnych jednostek obliczeniowych.

Przyszłe prace będą skupione na przeniesieniu systemu do pojemniejszych układów, co pozwoli na zwiększenie prędkości przetwarzania przez zwiększenie liczby procesorów podrzędnych, a także na integracji opisanego systemu z procesorem do detekcji, deskrypcji i dopasowywania cech w obrazie.

Michał Fularz w roku 2010/2011 jest stypendystą w ramach projektu pt.: „Wsparcie stypendialne dla doktorantów na kierunkach uznanych za strategiczne z punktu widzenia rozwoju Wielkopolski". Poddziałanie 8.2.2 Programu Operacyjnego Kapitał Ludzki, współfinansowanego przez Unię Europejską w ramach Europejskiego Funduszu Społecznego.

Praca naukowa finansowana ze środków na naukę w latach 2010 - 2012 w ramach projektu badawczego N514 213238.

7. Literatura

[1] Hartley R., Zissermann A.: Multiple view geometry in computer vision. Cambridge University Press, 2003.
[2] Cyganek B., Siebert J. P.: An introduction to 3D computer vision techniques and algorithms, Wiley, 2009.
[3] Fischler M. A., Bolles R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, Communications of the ACM, vol. 24, s. 381-395, 1981.
[4] Kraft M.: Sprzętowo-programowa realizacja algorytmu RANSAC do estymacji macierzy fundamentalnej. Pomiary Automatyka Kontrola 7/2010, s. 742-744.
[5] Xilinx Inc.: UG 081 – MicroBlaze Processor Reference Guide, October 2009. http://www.xilinx.com
[6] Xilinx Inc.: DS. 449 – Fast Simplex Link (FSL) Bus (v2.11b), Data Sheet, June 2009. http://www.xilinx.com
[7] Tippetts B., Fowers S., Lillywhite K., Lee D.J., Archibald J.: FPGA implementation of a feature detection and tracking algorithm for real-time applications, Advances in VisualComputing. Proceedings Third International Symposium, ISVC 2007. Part I. Lecture Notes in Computer Science vol. 4841. 2007: 682-91 Springer Verlag, Berlin, Germany.
[8] Martelli S., Marzotto R., Colombari A., Murino V.: FPGA-based robust ellipse estimation for circular road sign detection, Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops CVPR Workshops. 2010: 8 pp., IEEE Computer Society, Los Alamitos, CA, USA.

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