Marek KRAFT
POLITECHNIKA POZNAŃSKA, INSTYTUT AUTOMATYKI I INŻYNIERII INFORMATYCZNEJ,
ul Piotrowo 3A, 60-965 Poznań

Implementacja algorytmu do estymacji ruchu własnego robota w układzie FPGA

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 implementację w układzie FPGA systemu, realizującego zadanie szacowania ruchu własnego urządzenia (np. robota mobilnego), wyposażonego w pojedynczą kamerę. Zrealizowano ją w architekturze hybrydowej, sprzętowo-programowej. W artykule przedstawiono szczegółowy opis wynikowej architektury, jak również użycie zasobów układu programowalnego, oraz analizę wydajności systemu, wraz z porównaniem z alternatywnym rozwiązaniem opartym o komputer PC.

Słowa kluczowe: FPGA, odometria wizyjna, system wizyjny, system wbudowany.

Implementation of the Robot Ego-Motion Estimation Algorithm in FPGA Circuits

Abstract

The paper presents implementation of the robot ego-motion estimation algorithm in a single FPGA. The input data for the algorithm are feature correspondences detected in the image sequence registered by a single camera. The implemented system, based on the Microblaze microprocessor along with a dedicated hardware coprocessor, performs all stages of the algorithm – computation of the essential matrix using the 8-point algorithm employing singular value decomposition, robust estimation of the correct essential matrix using the RANSAC algorithm as well as computation of the rotation matrix and the translation vector (up to a scale) from the essential matrix [1, 2]. The system was implemented in a Virtex 5 PFGA and is capable of working with a clock speed of 100MHz. The microprocessor is used to find successive essential matrices using singular value decomposition. The solutions are tested for correctness using the coprocessor with the RANSAC algorithm [3]. The coprocessor employs a reduced, 23-bit floating point number representation to reduce resource usage. Upon successful completion of the essential matrix estimation, rotation and translation are computed. Additional sensors are used to deal with rotation and translation sign ambiguity. Table 1 presents the summary of resources used for implementation. Figure 1 outlines the system architecture. The results obtained are satisfactory and promising. The availability of inexpensive, low power, small footprint solution for ego-motion estimation is desirable for many applications.

Keywords: FPGA, visual odometry, vision system, embedded system.

1. Wstęp

Systemy wizyjne są kluczowym elementem systemów sensorycznych większości współcześnie konstruowanych robotów mobilnych. Obrazy rejestrowane przez systemy wizyjne zawierają wiele informacji i przetworzenie ich wymaga zaangażowania znaczącej mocy obliczeniowej. Informacja wizyjna wykorzystywana jest przez roboty mobilne najczęściej do wspomagania nawigacji. Współczesne mikroprocesory z powodzeniem radzą sobie nawet ze złożonymi algorytmami do przetwarzania i analizy obrazu, lecz najczęściej okupione jest to zwiększeniem kosztu i poboru mocy przez urządzenie końcowe.

W niniejszym artykule opisano hybrydową, sprzętowo-programową implementację algorytmu estymacji ruchu własnego robota. Na podstawie par dopasowanych punktów charakterystycznych (cech) z dwóch obrazów tej samej sceny, zarejestrowanych przy pomocy skalibrowanej kamery, system umożliwia estymację macierzy względnej rotacji $\boldsymbol{R}$ i wektora względnej translacji $\boldsymbol{t}$ (z dokładnością co do skali), wiążących dwie pozycje robota, w których obrazy zostały zarejestrowane. Implementacja obejmuje obliczanie macierzy zasadniczej $\boldsymbol{E}$ z par dopasowanych punktów za pomocą algorytmu 8-punktowego, odporną estymację macierzy $\boldsymbol{E}$ za pomocą algorytmu RANSAC, a także dekompozycję macierzy $\boldsymbol{E}$ na macierz $\boldsymbol{R}$ i wektor $\boldsymbol{t}$. System może być wykorzystany jako część systemu wizyjnego o niskim poborze mocy, wadze i rozmiarach. Cechy takie są szczególnie pożądane w aplikacjach mobilnych.

2. Estymacja parametrów modelu dla estymacji ruchu własnego

W rozdziale tym opisano kolejne kroki zaimplementowanego algorytmu estymacji ruchu własnego kamery.

Macierz fundamentalna i zasadnicza

W wizji komputerowej, macierze fundamentalna $\boldsymbol{F}$ i zasadnicza $\boldsymbol{E}$ są macierzami 3x3 rzędu 2, które wiążą odpowiadające sobie punkty (wyrażone w jednorodnych współrzędnych obrazowych) w dwóch obrazach tej samej sceny. Jeśli projekcję punktu sceny (wyrażoną we współrzędnych jednorodnych) na obraz pierwszy oznaczymy przez $\boldsymbol{x}$, a na obraz drugi przez $\boldsymbol{x'}$, to każda para odpowiadających sobie punktów $\boldsymbol{x}\leftrightarrow\boldsymbol{x'}$ powiązana jest przez macierz fundamentalną zależnością [1, 2]:

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

Macierz zasadnicza $\boldsymbol{E}$ powiązana jest z macierzą $\boldsymbol{F}$ zależnością:

$$E = K'^{T} FK . \tag{2}$$

Macierze $\boldsymbol{K}$ i $\boldsymbol{K'}$ to macierze kamer, którymi zarejestrowano oba obrazy sceny. Wartości elementów macierzy uzyskuje się w procesie kalibracji kamery.

Algorytm 8-punktowy

Jeśli poszczególne składniki zależności (1) mają postać:

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

to po zastosowaniu jej do pojedynczej pary dopasowanych punktów otrzymujemy:

$$\begin{aligned} &x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + \\ &+ y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \end{aligned} . \tag{3}$$

Macierz fundamentalna ma 8 stopni swobody, gdyż określona jest co do skali. Aby jednoznacznie ją wyznaczyć, potrzeba układu 8 równań, utworzonych na podstawie 8 par odpowiadających sobie punktów. Po zapisaniu elementów macierzy $\boldsymbol{F}$ w postaci wektora kolumnowego $\boldsymbol{f}$ w porządku wiersz po wierszu, układ równań można zapisać w formie:

$$Af = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_8x_8 & x'_8y_8 & x'_8 & y'_8x_8 & y'_8y_8 & y'_8 & x_8 & y_8 & 1 \end{bmatrix} f = 0 . \tag{5}$$

Ze względu na niedokładności związane z obecnością szumu i efektami kwantyzacji, układu nie da się rozwiązać metodami liniowymi i konieczne jest rozwiązanie go metodą najmniejszych kwadratów. W tym celu wykorzystuje się dekompozycję na wartości osobliwe SVD (ang. *singular value decomposition)* macierzy ***A***. Rozwiązaniem jest prawy wektor osobliwy odpowiadający najmniejszej wartości osobliwej [1, 2].

Estymacja odporna przy użyciu algorytmu RANSAC

W praktyce, pary punktów wykorzystywane do obliczeń są dopasowywane w sposób automatyczny. W zbiorze tak dopasowanych par punktów (zbiorze pomiarowym) zwykle występują również pary dopasowane błędnie. Rozwiązanie układu równań (5), w którym choćby jedno równanie składowe utworzone zostało na podstawie błędnie dopasowanych punktów, da błędny wynik. Ten brak odporności algorytmu 8-punktowego na wyżej wymienione zakłócenia wymusza konieczność zastosowania algorytmu estymacji odpornej. Na potrzeby opisywanych prac wykorzystano algorytm RANSAC.

RANSAC (ang. *Random Sample Consensus)* jest algorytmem, który umożliwia odporną estymację modelu matematycznego (w tym przypadku macierzy fundamentalnej) na podstawie zbioru danych pomiarowych (par dopasowanych punktów), które zawierają pewien odsetek wartości odstających (par nieprawidłowo dopasowanych punktów) [3]. Algorytm składa się z dwóch wykonywanych iteracyjnie kroków:

1. *Generowanie hipotezy*, czyli obliczanie parametrów modelu na podstawie *najmniejszej próbki* (najmniejszej możliwej liczby elementów zbioru pomiarowego, umożliwiającej pełne obliczenie parametrów modelu), pobieranej ze zbioru danych pomiarowych – w przypadku algorytmu 8-punktowego jest to 8 par współrzędnych dopasowanych punktów.
2. *Testowanie hipotezy*, czyli sprawdzanie zgodności wygenerowanego modelu z całym zbiorem pomiarowym – etap ten zwraca tzw. *zbiór zgodny* (ang. *consensus set*).

Algorytm kończy działanie, gdy prawdopodobieństwo znalezienia większego zbioru zgodnego spada poniżej pewnej ustalonej wartości granicznej. Liczba iteracji algorytmu wymaganych, aby z ustalonym prawdopodobieństwem znaleźć prawdziwy zbiór zgodny, rośnie silnie nieliniowo w funkcji liczby elementów najmniejszej próbki.

Dla celów implementacji na funkcję dopasowania danych pomiarowych do modelu wybrano błąd Sampsona, czyli aproksymację pierwszego rzędu błędu geometrycznego, wyrażoną wzorem:

$$\Delta_i = \frac{(x'^T_i\, E x_i)^2}{(Ex_i)_1^2 + (Ex_i)_2^2 + (E^T x'_i)_1^2 + (E^T x'_i)_2^2}, \tag{6}$$

gdzie $(Fx_i)_j$ oznacza *j*-ty element wektora [1].

Obliczanie względnej rotacji i translacji pomiędzy obrazami

Znajomość macierzy zasadniczej umożliwia obliczenie macierzy rotacji ***R*** i wektora translacji ***t*** (z dokładnością co do skali) przy pomocy dekompozycji SVD. Niech dekompozycja macierzy ***E*** dana będzie przez równanie:

$$SVD(E) = UDV^T, \text{ gdzie } D = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{7}$$

Dodatkowo wprowadzone zostaną macierze pomocnicze, oznaczone przez ***W*** i $\boldsymbol{W}^{\mathrm{T}}$:

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad W^T = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Jeśli oznaczymy pierwszą lokalizację kamery przez ***P***, przy rotacji i translacji ***P***=[***I*** | ***0***] (zerowa rotacja, oznaczona macierzą jednostkową ***I*** o odpowiednim rozmiarze, zerowa translacja), a drugą lokalizację kamery przez ***P'***, to istnieją cztery rozwiązania dla rotacji i translacji [1, 2]:

$$\begin{aligned} P' &= [UWV^T \mid u_3], \quad P' = [UWV^T \mid -u_3], \\ P' &= [UW^TV^T \mid u_3], \quad P' = [UW^TV^T \mid -u_3]. \end{aligned} \tag{8}$$

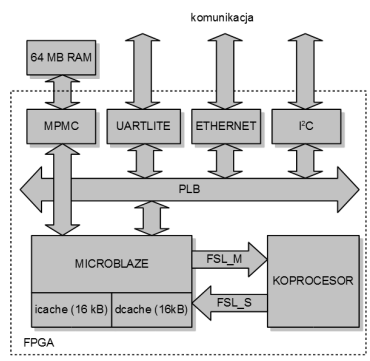
Przez $\boldsymbol{u_3}$ oznaczono trzecią kolumnę macierzy ***U***. Spośród czterech podanych przez (8) rozwiązań tylko jedno jest prawidłowe w sensie fizycznym; istnieje kilka metod jego wyboru. Na potrzeby pracy zdecydowano się na użycie prostej fuzji danych sensorycznych. Do określenia właściwego kierunku rotacji i translacji wykorzystano dane pomiarowe z magnetometru i akcelerometru.

3. Implementacja sprzętowa

Zaimplementowany system przystosowano do pracy z danymi znormalizowanymi, tzn. współrzędnymi par punktów przemnożonymi przez stosowne macierze kalibracji kamer.

Dekompozycja SVD macierzy wymaga zastosowania stosunkowo złożonych algorytmów iteracyjnych, których translacja na dedykowaną architekturę sprzętową jest problematyczna. W związku z tym, dekompozycję SVD zaimplementowano w formie programu, wykonywanego przez soft-procesor Microblaze. Program wykonywany przez mikroprocesor generuje kolejne hipotezy na podstawie zbioru danych pomiarowych (współrzędnych par dopasowanych punktów). Wyniki przekazywane są następnie do dedykowanego koprocesora sprzętowego, wykorzystywanego do testowania hipotez.

Koprocesor jest implementacją wzoru na błąd Sampsona (6), zaprojektowaną specjalnie do współpracy z mikroprocesorem Microblaze. Wzór (6) podzielić można na pojedyncze operacje matematyczne, z powodzeniem poddające się translacji na potokową, równoległą architekturę sprzętową. Koprocesor połączony jest z mikroprocesorem Microblaze za pomocą dedykowanego interfejsu FSL (ang. *Fast Simplex Link*). FSL jest szybkim, jednokierunkowym interfejsem punkt-punkt z buforowaniem FIFO. Mikroprocesor wyposażono w specjalizowane, szybkie instrukcje do obsługi tego interfejsu, dzięki czemu komunikacja z jego wykorzystaniem jest szybsza, niż komunikacja z wykorzystaniem systemowej magistrali PLB (ang. *Processor Local Bus*). Architekturę systemu przedstawiono w sposób schematyczny na rysunku 1.



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

W systemie zastosowano mikroprocesor Microblaze w wersji z 5-stopniowym potokiem oraz jednostką wspomagającą operacje zmiennoprzecinkowe. Jest to wariant o wyższej wydajności i wyższym zapotrzebowaniu na zasoby układu programowalnego. Dalszy wzrost wydajności uzyskano dzięki wyposażeniu go w pamięć podręczną (*cache*) dla danych i instrukcji o pojemności 16 kB każda. Jako platformę testową wykorzystano płytę ewaluacyjną firmy Avnet, na której zamontowano układ XC5VFX30T-1FFG665 z rodziny Virtex 5 firmy Xilinx. Dodatkowe wyposażenie płyty obejmuje również 64 MB pamięci DDR2 RAM, 16 MB pamięci FLASH, a także interfejsy komunikacyjne, w tym gigabitowy Ethernet i port szeregowy.

W celu zmniejszenia zapotrzebowania na zasoby sprzętowe układu programowalnego zdecydowano się na zastosowanie niestandardowej reprezentacji liczb zmiennoprzecinkowych w koprocesorze. Dla przykładu, dzięki zastosowaniu reprezentacji 23-bitowej w miejsce 32-bitowej, zużycie dedykowanych bloków DSP48E, koniecznych do zaimplementowania funkcji mnożenia liczb zmiennoprzecinkowych, zmniejsza się z trzech do jednego.

Bieżący model, wyznaczony drogą programową, przesyłany jest do koprocesora przy użyciu jednego z kanałów FSL i zapisywany w wewnętrznych rejestrach koprocesora. Tym samym kanałem przesyłane są następnie kolejne współrzędne punktów w celu przetestowania zgodności modelu z danymi pomiarowymi. Drugi kanał FSL jest kanałem zwrotnym. Przesyłane są nim wyniki obliczeń z koprocesora (wartości błędu dopasowania do modelu dla każdej z par dopasowanych punktów). Do sterowania przepływem danych oraz odpowiedniego ich konwertowania opracowany został dedykowany blok kontrolera. Oba z kanałów FSL są buforowane kolejkami FIFO, w których zapisać można do 1024 słów. Zapewnia to odpowiednią płynność transmisji danych.

W momencie zakończenia procesu odpornej estymacji, wynikowa macierz zasadnicza jest poddawana dekompozycji SVD w celu określenia rotacji i translacji. Operacja ta wykonywana jest przez program działający na mikroprocesorze Microblaze. Do wyboru właściwego rozwiązania spośród czterech możliwości (patrz równania (8)) wykorzystywane są dane pomiarowe z czujnika MEMS, zawierającego w swojej strukturze 3-osiowy akcelerometr i 3-osiowy magnetometr, podłączony do systemu za pośrednictwem magistrali I^2C. Dane pomiarowe z czujnika MEMS po odpowiednim przetworzeniu i filtracji dają informację o kierunku obrotu wokół wszystkich osi oraz kierunku ruchu w przestrzeni (po podwójnym scałkowaniu przyspieszeń). Ponieważ wykorzystywane są jedynie informacje o kierunku (znaku) przemieszczeń i obrotów, nie jest wymagana wysoka dokładność pomiarów.

4. Analiza wyników

System, na którym wykonywano testy, taktowany był sygnałem zegarowym o częstotliwości 100 MHz. Raporty z implementacji wskazują, że sam koprocesor może być taktowany z częstotliwością do 170 MHz. Zużycie zasobów przez cały system oraz przez sam koprocesor sprzętowy podano w tabeli 1.

Tab. 1. Podsumowanie zużycia zasobów układu FPGA przez koprocesor i cały system
Tab. 1. Summary of FPGA resources used to implement the coprocessor and the whole system

zasób	koprocesor	system
BlockRAM	0	23
FF	4511	11429
LUT	7672	14906
DSP48E	21	26

Testy wydajności wykonywane były z wykorzystaniem zbioru rzeczywistych danych pomiarowych, składających się z 208 par dopasowanych punktów (cech). Jako detektora cech użyto detektora Harrisa, a do dopasowania cech pomiędzy dwoma obrazami wykorzystano miarę podobieństwa SSD (suma kwadratów różnic pikseli w otoczeniu cechy). Kamera użyta do zarejestrowania obrazów testowych została uprzednio skalibrowania, co umożliwiło odpowiednie przekształcenie (znormalizowanie) surowych danych pomiarowych (współrzędnych punktów).

Ponieważ wymaganej do znalezienia rozwiązania liczby iteracji algorytmu RANSAC nie da się oszacować z góry, testy porównawcze z innymi platformami przeprowadzono wykonując na danej platformie 10000 iteracji algorytmu. Każda z iteracji wykonywana była na 8 losowo wybranych ze zbioru danych pomiarowych parach punktów. Testy umożliwiły określenie średniego czasu wykonywania pojedynczej iteracji. Jest to o tyle istotne, że obliczenia mające na celu estymację macierzy zasadniczej zajmują zdecydowanie więcej czasu niż późniejsza jej dekompozycja na komponenty rotacji i translacji. Dla implementacji czysto programowej na mikroprocesorze Microblaze czas wykonywania wynosi 21,4 ms. Dodanie koprocesora wspomagającego testowanie wygenerowanych hipotez skraca ten czas do 12,1 ms. Dla porównania, jedna iteracja implementacji programowej tego samego algorytmu na komputerze z mikroprocesorem Intel Atom N270 1,6 GHz i 1 GB pamięci RAM, pracującym pod kontrolą systemu Ubuntu Linux 9.10 zajmuje 2,3 ms. Badania wykazały, że zredukowanie precyzji reprezentacji liczb zmiennoprzecinkowych w koprocesorze wprowadza błąd nie większy niż 3% (w stosunku do reprezentacji 32-bitowej), co w opisywanej aplikacji jest akceptowalne.

Zastosowanie koprocesora skraca czas testowania hipotezy ponad 50-krotnie w stosunku do implementacji programowej w mikroprocesorze Microblaze i około 5-krotnie w stosunku do implementacji programowej na mikroprocesorze Intel Atom. Należy tu nadmienić, że im większy zbiór danych pomiarowych, tym mniejsza względna różnica w szybkości wykonywania jednej iteracji algorytmu, ze względu na zwiększenie się udziału operacji testowania hipotezy w całkowitym czasie obliczeń.

Dalsze zwiększenie wydajności można uzyskać na wiele sposobów. Koprocesor może być taktowany sygnałem zegarowym o częstotliwości wyższej, niż sygnał taktujący mikroprocesora (interfejs FSL umożliwia komunikację między dwoma różnymi domenami zegarowymi), co spowoduje przyspieszenie działania procesu testowania kolejnych hipotez. Możliwe jest również wykorzystanie odmiany algorytmu RANSAC charakteryzującej się większą szybkością działania [4]. Kolejną drogą do zwiększenia ogólnej prędkości przetwarzania jest powielenie systemu – jest ono możliwe, ponieważ hipotezy mogą być testowane niezależnie od siebie.

Z punktu widzenia kompletnej aplikacji interesująca wydaje się możliwość połączenia opisywanego systemu z dedykowanymi blokami sprzętowymi, wykonującymi operacje wstępnego przetwarzania obrazu, takie jak filtracja czy detekcja cech. Dedykowane bloki sprzętowe, przy odpowiedniej organizacji wejściowych danych obrazowych, umożliwiają uzyskanie kilkadziesiąt razy wyższej prędkości przetwarzania niż przy wykorzystaniu komputerów PC [5].

5. Wnioski

W artykule zaprezentowano implementację systemu do estymacji ruchu własnego robota na podstawie danych wizyjnych, wspomaganych odczytami z innych czujników, wykonaną w oparciu o układ programowalny. Uzyskane wyniki są zadowalające. Możliwe jest dalsze przyspieszenie wykonywania obliczeń, co będzie tematem dalszych prac. Planowane jest również zintegrowanie systemu z innymi dedykowanymi blokami sprzętowymi, wykonującymi operacje filtracji i wstępnego przetwarzania obrazu [6], w celu uzyskania kompletnego, zintegrowanego systemu wizyjnego o małym poborze mocy, gabarytach i masie.

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

6. 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] Chum O., Matas J.: Optimal randomized RANSAC. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30(8), s. 1472-1482, 2008.
[5] Draper B. A., Beveridge J. R., Bohm A. P., Ross C., Chawathe M.: Accelerated image processing on FPGAs. IEEE Transactions on Image Processing vol 12, s. 1543-1551, 2003.
[6] Kraft M., Fularz M.: Porównanie sprzętowych implementacji dwóch popularnych detektorów cech punktowych. Pomiary Automatyka Kontrola 8/2009, s. 618-620.

otrzymano / received: 15.10.2010
przyjęto do druku / accepted: 01.12.2010 *artykuł recenzowany*