Zachodniopomorski Uniwersytet Technologiczny w Szczecinie

Wydział Elektryczny

Radosław Rajczyk

nr albumu: 23804

Implementacja algorytmu Viterbiego z wykorzystaniem biblioteki OpenCL

Praca dyplomowa magisterska

kierunek: Automatyka i Robotyka

specjalność: Systemy sterowania procesami przemysłowymi

Opiekun pracy:

dr hab. inż. Przemysław Mazurek

Katedra Przetwarzania Sygnałów i Inżynierii Multimedialnej

Wydział Elektryczny

Szczecin, 2016

Spis treści

1 Streszczenie **3**

2 Wstęp **4**
2.1 Przetwarzanie obrazu i jego rola w automatyce przemysłowej 4
2.2 Istotność szybkości obliczeń w problemach wizji maszynowej 5
2.3 Cel, zakres i zastosowania pracy 7

3 Metody równoległego przetwarzania danych **9**
3.1 Wielowątkowość CPU dla aplikacji C/C++ 9
3.1.1 Biblioteka POSIX dla systemów Unix 9
3.1.2 OpenMP - wieloplatformowe API 9
3.1.3 Wielowątkowość w standardzie C++11 9
3.2 Programowanie równoległe z wykorzystaniem GPU 9
3.2.1 Architektura GPU i porównanie względem CPU 9
3.2.2 Biblioteka OpenCL 9

4 Algorytm Viterbiego **10**
4.1 Opis działania i zastosowania 10
4.2 Implementacja w języku C++ 10
4.2.1 Wersja szeregowa 10
4.2.2 Wersja równoległa - C++11 10
4.2.3 Wersja równoległa - OpenCL 10

5 Wyniki badań doświadczalnych implementacji algorytmu Viterbiego **11**
5.1 Porównanie czasu działania dla implementacji szeregowej, wielowątkowej oraz z wykorzystaniem biblioteki OpenCL 11
5.2 Porównanie szybkości algorytmów dla różnych konfiguracji sprzętowych 11

6 Wnioski końcowe **12**

7 Załącznik B **13**

8 Załącznik A **14**

9 Bibliografia **15**

Spis rysunków **17**

Rozdział 1

Streszczenie

To jest streszczenie

Rozdział 2

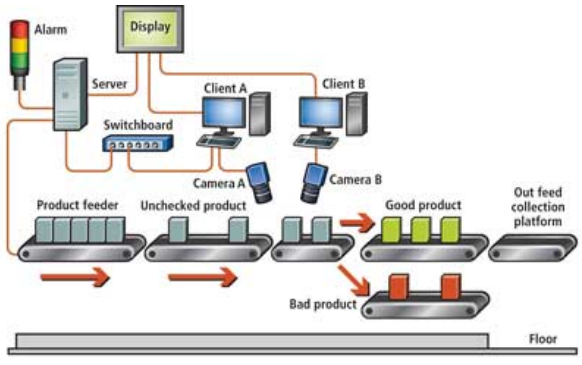
Wstęp

2.1 Przetwarzanie obrazu i jego rola w automatyce przemysłowej

W zagadnieniach technik pomiarowych oraz analizy otoczenia coraz częściej stosowane są rozwiązania wykorzystujące systemy wizyjne. Do najpopularniejszych zastosowań przemysłowych wizji maszynowej należą [6]:

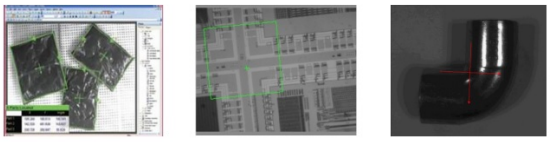
- inspekcja elementów na linii technologicznej
- określanie właściwej orientacji i położenia elementów
- identyfikacja produktów
- pomiary metrologiczne

W automatyce przemysłowej gdzie do zagadnień inspekcji wcześniej niezbędna była ocena wizualna człowieka, obecnie powszechnie stosuje się systemy wizyjne, w których skład wchodzą kamery przemysłowe, czujniki wyzwalające(np. na bazie pozycji) oraz urządzenie odpowiadającego za proces decyzyjny. Występują również rozwiązania w postaci systemów wbudowanych, gdzie inteligentna kamera oprócz akwizycji obrazu zajmuje się jego przetwarzaniem i analizą, wykorzystując własny procesor.[6][7]



Rysunek 2.1: Przykład zautomatyzowanej linii technologicznej wykorzystującej system wizyjny[17]

Sprawdzanie orientacji i położenia elementów w przemyśle jest wykorzystywane między innymi w technologii montażu, gdzie informacje z urządzeń wizyjnych są wykorzystywane przez manipulatory przemysłowe do zautomatyzowanego montażu, sortowania oraz paletyzacji wyrobów.[6]



Rysunek 2.2: Przykład obrazów używanych w testowaniu pozycji i orientacji elementów[6]

Identyfikowanie produktów na bazie obrazu cyfrowego jest wykorzystywane przy sortowaniu oraz monitorowaniu przepływu elementów i lokalizacji wąskich gardeł. Przykładowe metody indentyfikacji to stosowanie kodów kreskowych i kodów DataMatrix.[6]

Rysunek 2.3: Przykład wizyjnej identyfikacji[6]

2.2 Istotność szybkości obliczeń w problemach wizji maszynowej

Większość praktycznych zastosowań przetwarzania obrazu jako dodatkowej informacji w sterowaniu jednym bądź grupą urządzeń, wymaga akwizycji oraz wykonywania obliczeń w czasie rzeczywistym. Oznacza to, że wybrany algorytm wykorzystywany do analizy obrazu cyfrowego, wraz z resztą niezbędnego kodu, musi posiadać czas wykonania spełniający narzucone przez sterowany system.

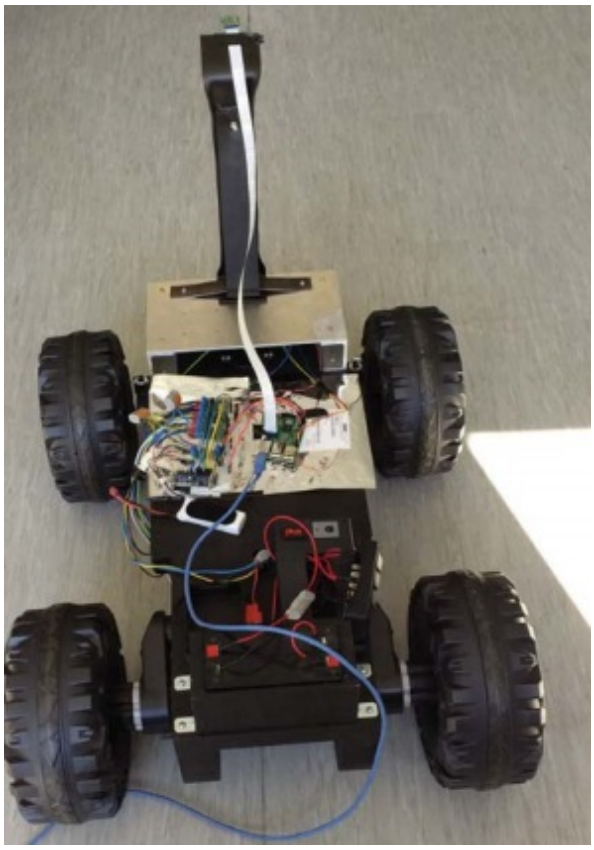
Dla zastosowań przemysłowych. gdzie monitorowane obiekty poruszają się z dużą prędkością, szybkość podjęcia decyzji przez system wizyjny może być wąskim gardłem dla danej gałęzi linii produkcyjnej. Czas na wykonanie decyzji (np. o usunięciu wadliwego produktu z przenośnika taśmowego), składa się na czas akwizycji obrazu, obliczenia sterowania. Standardowe kamery przemysłowe potrafią zrobić nawet powyżej 100 zdjęć na sekundę, a nawet więcej stosując mniejsze rozdzielczości obrazu. Czas przesyłu danych dla standardu popularnego standardu GigE wynosi maksymalnie 125 MB/s [10]. Na podstawie tego można stwierdzić, że główny problem będzie stanowił czas obliczenia sterowania i od niego będzie zależeć szybkość działania systemu wizyjnego[5][3].

Inną dziedziną gdzie stosowane jest przetwarzanie obrazu w czasie rzeczywistym jest robotyka mobilna, gdzie system wizyjny może odpowiadać za:

- Sprzężenie niezbędne do obliczenia zmiany położenia i prędkości robota.
- Lokalizację przeszkód oraz innych robotów(Swarm Robotics).
- Analizę oraz monitorowanie otoczenia.

[9][1]

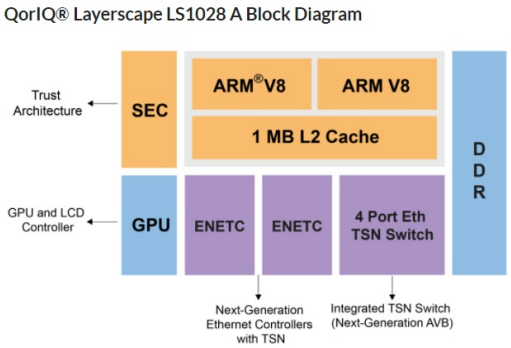
Podobnie jak dla aplikacji przemysłowych odpowiedzialność za wykorzystanie pełnych możliwości układów wykonawczych robota jest szybkość obliczania nowych sterowań. Niezbędne obliczenia mogą być wykonywane bezpośrednio przez urządzenie sterujące silnikami robota, albo z pomocą osobnej



Rysunek 2.4: Robot mobilny do śledzenia linii[9]

stacji, która połączona zdalnie z kontrolerem robota jest odpowiedzialna za przeprowadzanie czasochłonnych obliczeń.

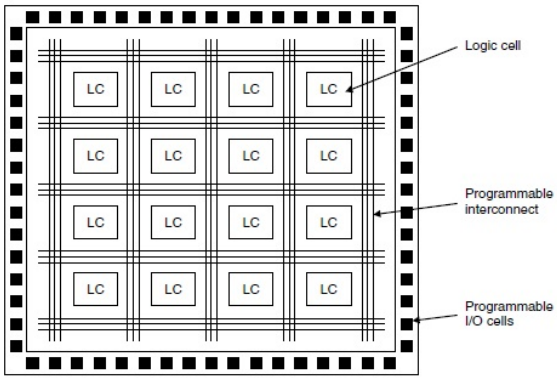
Pierwsze rozwiązanie jest korzystne kiedy nie są wymagane duże rozdzielczości obrazu, skomplikowane i czasochłonne algorytmy przetwarzania obrazu o dużej złożoności obliczeniowej oraz wysokie prędkości ruchu robota, narzucające krótki czas na obliczenia. Stosowane są wtedy najczęściej układy wyposażone mikroprocesory, np. rodzina procesorów ARM oraz x86-64 firmy Intel. Obecne modele są najczęściej wielordzeniowe o taktowaniu nawet powyżej 1GHz [15] [11].



Rysunek 2.5: Procesor *QorlQ Layerscape LS1028* do aplikacji przemysłowych firmy NXP, wyposażony w dwa rdzenie ARMv8[15]

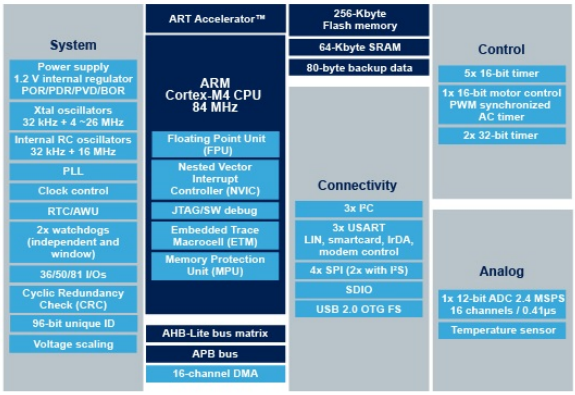
Dla rozwiązań bardziej wymagających pod względem szybkości obliczeń stosowane są układy FPGA, w których logika przetwarzania informacji obrazu jest zaprojektowania w języku HDL(VHDL,

Verilog). Zapewniają one najszybsze prędkości obliczeń ze względu na sprzętową implementację algorytmów. [8][4].



Rysunek 2.6: Architektura układu FPGA[8]

Drugie rozwiązanie gdzie osobne urządzenie jest używane do przetwarzania obrazu, umożliwia użycie mniej kosztownego procesora po stronie robota. Wystarczające do sterowania silnikami jest zastosowanie układu wykorzystującego mikrokontroler(np. z rodziny STM32 bądź Atmel AVR)[16] [2].



Rysunek 2.7: Structura mikrokontrolera *STM32F401CC*[16]

Zewnętrzna jednostka obliczeniowa pozwala na wykorzystanie możliwości konwencjonalnych wielordzeniowych procesorów dla komputerów PC oraz procesora karty graficznej, który jest wyspecjalizowany w obliczeniach równoległych[14] [12][13]. Dzięki kombinacji CPU i GPU możliwe jest przyspieszenie operacji, które mogą być wykonywane równolegle oraz rozdzielenie obciążenia obliczeniowego pomiędzy procesor i kartę graficzną. Podsumowując, szybkość obliczeń systemu wizyjnego w robotyce mobilnej decyduje o tym jakie mogą być maksymalne parametry ruchu robota - prędkość, przyspieszenie, ilości robotów współpracujących w zagadnieniach robotyki roju(Swarm Robotics) oraz poziomie skomplikowania analizy obrazu. W przypadku problemów wizji maszynowej dla zastosowań przemysłowych czas poświęcony na analizę każdego zdjęcia ma wpływ na szybkość działania całej linii produkcyjnej, co ma bezpośredni wpływ na wydajność i koszty produkcji. podrozdział 2

2.3 Cel, zakres i zastosowania pracy

Celem pracy jest implementacja algorytmu Viterbiego w celu wykrywania linii na zaszumionym obrazie cyfrowym oraz analiza porównawcza dla rożnych wersji napisanego algorytmu. Rozpatrywana

będzie implementacja szeregowa i równoległa dla CPU w języku C++ oraz napisana pod procesor karty graficznej z wykorzystaniem biblioteki OpenCL. Implementacja z wykorzystaniem biblioteki OpenCL będzie składała się z dwóch wariantów:

- całkowicie wykonywany przez GPU
- hybrydowy - podział obciążenia obliczeniowego pomiędzy procesor i kartę graficzną.

Następnie dla różnych konfiguracji sprzętowych zostanie zrobione porównanie ich szybkości. Na podstawie powyższej analizy zostanie wybrany najlepszy wariant realizacji algorytmu Viterbiego, co będzie mogło być później zastosowane w sterowaniu ruchem robota mobilnego.

Rozdział 3

Metody równoległego przetwarzania danych

3.1 Wielowątkowość CPU dla aplikacji C/C++

To jest rozdział 1

3.1.1 Biblioteka POSIX dla systemów Unix

To jest podrozdział 1 rozdziału 1

3.1.2 OpenMP - wieloplatformowe API

To jest podrozdział 2 rozdziału 1

3.1.3 Wielowątkowość w standardzie C++11

To jest podrozdział 3 rozdziału 1

3.2 Programowanie równoległe z wykorzystaniem GPU

To jest rozdział 2

3.2.1 Architektura GPU i porównanie względem CPU

To jest podrozdział 1 rozdziału 2

3.2.2 Biblioteka OpenCL

To jest podrozdział 2 rozdziału 2

Rozdział 4

Algorytm Viterbiego

4.1 Opis działania i zastosowania

To jest rozdział 1

4.2 Implementacja w języku C++

To jest rozdział 2

4.2.1 Wersja szeregowa

To jest podrozdział 1 rozdziału 2

4.2.2 Wersja równoległa - C++11

To jest podrozdział 2 rozdziału 2

4.2.3 Wersja równoległa - OpenCL

To jest podrozdział 3 rozdziału 2

Rozdział 5

Wyniki badań doświadczalnych implementacji algorytmu Viterbiego

5.1 Porównanie czasu działania dla implementacji szeregowej, wielowątkowej oraz z wykorzystaniem biblioteki OpenCL

To jest rozdział 1

5.2 Porównanie szybkości algorytmów dla różnych konfiguracji sprzętowych

To jest rozdział 2

Rozdział 6

Wnioski końcowe

To jest zakończenie

Rozdział 7

Załącznik B

To jest załącznik B

Rozdział 8

Załącznik A

To jest załącznik A

Rozdział 9

Bibliografia

[1] Sachin B. Bhosale Amol N. Dumbare, Kiran P.Somase. Mobile robot for object detection using image processing. *International Journal of Advane Research in Computer Science and Managment Studies*, 1(6):81–84, 2013.

[2] Atmel. Atmel avr 8-bit and 32-bit microcontrollers. `http://www.atmel.com/products/microcontrollers/avr/default.aspx`.

[3] Basler. Basler camera portfolio. `https://www.baslerweb.com/en/products/cameras/`.

[4] Pong P. Chu. *FPGA Prototyping by VHDL examples.* John Wiley and Sons, Inc., 2008.

[5] Cognex. Insight 5000 industrial vision systems. `http://www.cognex.com/productstemplate.aspx?id=13915`.

[6] Cognex. Introduction to machine vision. `http://www.assemblymag.com/ext/resources/White_Papers/Sep16/Introduction-to-Machine-Vision.pdf`, 2016.

[7] E.R. Davies. *Computer and Machine Vision: Theory, Algorithms, Practicalities.* Elsevier, 225 WYman Street, Waltham, 02451, USA, 2012.

[8] Ian Grout. *Digital Systems Design with FPGAs and CPLDs.* Elsevier, 2008.

[9] Przemysław Mazurek Grzegorz Matczak. Line following with real-time viterbi trac-before-detect algorithm. *Przegląd Elektrotechniczny*, 1/2017:69–72, 2017.

[10] National Instruments. Choosing the right camera bus. *NI white papers*, 2016.

[11] Intel. Intel processors and chipsets for embedded applications. `http://www.intel.pl/content/www/pl/pl/intelligent-systems/embedded-processors-which-intel-processor-fits-your-project.html`.

[12] David Patterson John Hennesy. *Computer Architecture: A Quantitative Approach.* Elsevier, 2011.

[13] Mike Houston Katvon Fatahalian. A closer look at gpus. *Communications of the ACM*, 51(10):50–57, 2008.

[14] Nvidia. What is gpu-accelerated computing. `http://www.nvidia.com/object/what-is-gpu-computing.html`.

[15] NXP. Arm technology-based solutions - nxp microcontrollers and processors. `http://www.nxp.com/products/microcontrollers-and-processors/arm-processors:ARM-ARCHITECTURE`.

[16] ST. Stm32 32-bit arm cortex mcus. http://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html?querycriteria=productId=SC1169.

[17] Andy Wilson. Industrial inspection : Line-scan-based vision system tackles color print inspection. *Vision Systems Design*, 2014.

Spis rysunków

2.1 Przykład zautomatyzowanej linii technologicznej wykorzystującej system wizyjny[17] . 4
2.2 Przykład obrazów używanych w testowaniu pozycji i orientacji elementów[6] 5
2.3 Przykład wizyjnej identyfikacji[6] 5
2.4 Robot mobilny do śledzenia linii[9] 6
2.5 Procesor *QorlQ Layerscape LS1028* do aplikacji przemysłowych firmy NXP, wyposażony w dwa rdzenie ARMv8[15] 6
2.6 Architektura układu FPGA[8] 7
2.7 Structura mikrokontrolera *STM32F401CC*[16] 7