Akademia Górniczo-Hutnicza im. Stanisława Staszica
Wydział Elektrotechniki Automatyki Informatyki i Elektroniki

Rozprawa doktorska

Metodyka zrównoleglania algorytmów przetwarzania i analizy obrazów w systemach przepływowych

Mirosław Jabłoński

Promotor: dr hab. inż. Marek Gorgoń

Kraków, 2009

Ukochanym Rodzicom

dedykuję tę pracę w podziękowaniu
za nieocenione wsparcie i wyrozumiałość
w całym okresie przygotowań do jej redakcji.

Spis treści

Spis treści **i**

I Wstęp **3**

1 Motywacja i kontekst pracy **5**

2 Teza i cel prac **7**

3 Zawartość rozprawy **9**

II Równoległość w systemach wizyjnych **11**

4 Platformy obliczeniowe a systemy wizyjne **13**
4.1 Kryteria wyboru platformy obliczeniowej dla systemów wizyjnych . 13
4.2 Klasyfikacja systemów obliczeniowych 14
4.2.1 Programowane i konfigurowane elementy obliczeniowe . . . 16
4.3 Obliczenia równoległe 17
4.3.1 Przyspieszenie w systemach obliczeniowych 18
4.3.2 Superliniowe przyspieszenie 20
4.3.3 Platformy programowane a konfigurowalne 21
4.3.4 Obliczenia potokowe 22
4.4 Przetwarzanie i analiza sygnału wizyjnego 23
4.4.1 Przetwarzanie wstępne 23
4.4.2 Segmentacja 24
4.4.3 Analiza i rozpoznawanie 24
4.5 Granulacja w systemach obliczeniowych 25
4.5.1 Granulacja systemów rekonfigurowalnych 25
4.5.2 Granulacja danych w przetwarzaniu obrazów 25

5 Strumieniowy system wizyjny 27
5.1 Efektywność strumieniowego systemu wizyjnego 29
5.1.1 Zwiększanie efektywności 30
5.1.2 Realizacja algorytmów o dużym stopniu nieregularności . . 31
5.2 Kryteria oceny przepływowego systemu wizyjnego 32
5.2.1 Opóźnienie transportowe 33
5.2.2 Opóźnienie transportowe w potoku drobnoziarnistym 33
5.2.3 Opóźnienie transportowe w potoku gruboziarnistym 35
5.2.4 Opóźnienie przepływowego systemu wizyjnego 36

III Zrównoleglanie algorytmów wizyjnych 39

6 Implementacja operacji wizyjnych o kontekście przestrzennym 41
6.1 Formowanie sygnału wizyjnego 41
6.1.1 Czasowe parametry strumienia wizyjnego 42
6.1.2 Barwny czujnik wizyjny 46
6.1.3 Interpolacja pikseli 47
6.2 System wizyjny rozpoznający ręcznie pisane cyfry 50
6.3 Algorytmy przetwarzania obrazów 52
6.3.1 Przetwarzanie wstępne 53
6.3.2 Segmentacja 53
6.3.3 Indeksacja 54
6.3.4 Pomiar cech i selekcja obiektów 56
6.3.5 Skalowanie obiektów 58
6.3.6 Rozpoznawanie znaków 60
6.4 Strumieniowa realizacja toru wizyjnego do rozpoznawania znaków . 63
6.4.1 Aspekty obliczeniowe w sprzętowej analizie obrazu 64
6.4.2 Aspekty obliczeniowe w implementacji sieci neuronowej . . 66
6.4.3 Zrównoleglenie w rozpoznawaniu znaków 67
6.5 Równoległa implementacja nieregularnych algorytmów 76
6.5.1 Wpływ równoległości na przepustowość 81
6.5.2 Zrównoleglanie sztucznej sieci neuronowej 87
6.6 Wyniki strumieniowej realizacji systemu rozpoznawania znaków . . 91

7 Implementacja operacji wizyjnych o kontekście temporalnym 95
7.1 Zasoby pamięciowe w kontekstowych operacjach wizyjnych 95
7.2 Algorytm wideodetekcji ruchu drogowego 97
7.3 Strumieniowa realizacja algorytmu wideodetekcji 101
7.3.1 Estymacja zapotrzebowania na zasoby pamięciowe 102
7.3.2 Zrównoleglenie przy ograniczonych zasobach pamięciowych 104
7.3.3 Architektura potokowa a kontekst temporalny 104

7.3.4 Bilans równoległości danych i równoległości operacji 106
7.3.5 Transformacja architektury potokowej 106
7.3.6 Potokowa realizacja podsystemów wideodetektora 109
7.4 Implementacja wideodetektora 113

8 Podsumowanie **117**
8.1 Wyniki prac i wnioski 117
8.1.1 Równoległość 117
8.1.2 Operacje wizyjne o kontekście przestrzennym 118
8.1.3 Operacje wizyjne o kontekście temporalnym 118
8.2 Kierunki dalszych badań 119

Bibliografia **121**

A Wyznaczanie przyspieszenia w zrównoleglaniu wieloetapowym **133**
A.1 Metody wyznaczania przyspieszenia 133
A.1.1 Wyznaczanie przyspieszenia etapami 134
A.1.2 Iteracyjne wyznaczanie przyspieszenia 135

B Przyspieszenie i efektywność w architekturze potokowej **137**
B.1 Wpływ parametrów potoku na przyspieszenie 137
B.2 Wpływ parametrów potoku na efektywność 138

C Zrównoleglony kod źródłowy - przykłady **141**
C.1 Indeksacja obiektów w potoku drobnoziarnistym 141
C.2 Wideodetektor - bufor kontekstu temporalnego 142

Spis skrótów **143**

Spis rysunków **145**

Spis tabel **147**

Podziękowania

Autor składa serdeczne podziękowania Promotorowi dr hab. Markowi Gorgoniowi za poświęcony czas i i pomoc w tworzeniu rozprawy. Wyrazy wdzięczności należą się również całemu zespołowi Laboratorium Biocybernetyki pod kierownictwem profesora dr hab. inż. Ryszarda Tadeusiewicza, a w szczególności:

dr inż. Zbigniewowi Mikrutowi za pomoc w obszarze sieci neuronowych i wideodetekcji,

dr hab. inż Markowi Gorgoniowi za udostępnienie oprogramowania i sprzętu do badań,

dr inż Jaromirowi Przybyło za wartościowe dyskusje, oraz wiele ciekawych pomysłów,

dr n. techn. lek. med. Pawłowi Wołoszynowi za inspirujące rady.

Część I

Wstęp

Rozdział 1

Motywacja i kontekst pracy

Systemy składowania, transmisji, prezentacji analizy i rozpoznawania sygnałów z otaczającego nas świata stają się coraz bardziej obecne w niemal wszystkich sferach ludzkiego życia. Urządzenia do rejestracji, prezentacji i transmisji mediów są już bardzo powszechne. Jakość ich działania oraz możliwości zależą jednak od dostępnych środków obliczeniowych. Jednym z obszarów techniki nabierającym coraz większego znaczenia jest analiza i rozpoznawanie obrazów.

Zgodnie z prawem Moore'a [75] rośnie upakowanie tranzystorów w procesorach. Postęp dokonuje się wielowymiarowo między innymi w kierunku zmniejszenia zapotrzebowania na energię czy koszt jednostkowy. Od kilku lat obserwuje się jednak zatrzymanie wzrostu maksymalnej częstotliwości taktowania systemów cyfrowych a w szczególności procesorów ogólnego przeznaczenia. Zjawisko to sygnalizowano w pracach [41][98]. Wydaje się, że dalszy postęp wydajności systemów obliczeniowych jest możliwy pomimo osiągnięcia granicznej częstotliwości przełączania cyfrowych układów scalonych. Wzrost mocy obliczeniowej uzyskiwany jest poprzez zwielokrotnienie jednostek obliczeniowych: wielordzeniowe procesory ogólnego przeznaczenia [21], niektóre procesory DSP (ang. *Digital Signal Processing*)[7][84], procesory graficzne GPU[82] (ang. *Graphics processing Unit*) i układy rekonfigurowalne FPGA(ang. *Field Programmable Gate Array*) [11][33][56][87][109]. Autor pracy [42] zaznacza, iż rzeczywisty potencjał równoległych obliczeń nie jest należycie wykorzystany.

Jedną z dziedzin, która ze względu na dynamiczny rozwój jest ciągle domeną prac badawczych i specjalistycznych zastosowań, są szeroko rozumiane systemy wizyjne. Zagadnienie realizacji algorytmów widzenia maszynowego jest obecne w nauce i technice przynajmniej od połowy ubiegłego stulecia. Ograniczone możliwości maszyn obliczeniowych stanowiły motywację do poszukiwania efektywnych metod implementacji algorytmów wizyjnych o różnym stopniu złożoności. Postęp dokonywał się nie tylko poprzez dobór metod implementacji algorytmów

na komputerach ogólnego przeznaczenia. Poszukiwano również metod akceleracji obliczeń poprzez budowę dedykowanych systemów obliczeniowych i wykorzystanie akceleratorów sprzętowych. Takie prace były prowadzone również w Laboratorium Biocybernetyki Akademii Górniczo-Hutniczej pod kierownictwem prof. Ryszarda Tadeusiewicza [93]. System Cesaro zbudowany w oparciu o cyfrowe układy scalone TTL umożliwiał wykonanie szeregu operacji przetwarzania obrazu w czasie rzeczywistym, co było niemożliwe do osiągnięcia przy wykorzystaniu dostępnych wówczas komputerów. Kontynuację tych badań stanowił projekt antropomorficznej karty Retina opracowany przez dr inż. Z. Mikruta [74] oraz prace prof. dr hab. inż. K. Wiatra [101] i dr hab. inż. M. Gorgonia [29] dotyczące sprzętowych procesorów obrazu.

Dzięki ciągłemu rozwojowi komputerów osobistych, problem niewystarczającej mocy obliczeniowej pojawia się rzadziej. Systemy widzenia maszynowego nie ograniczają się jednak wyłącznie do typowych zastosowań stacjonarnych. Istnieje potrzeba wbudowania algorytmów wizyjnych w samodzielne urządzenia przenośne lub zorganizowane w sieci sensorów. Problem wydajności systemu obliczeniowego w aplikacjach wizyjnych jest więc nadal istotny.

Wobec braku kompleksowych opracowań dotyczących metodyki zrównoleglania algorytmów wizyjnych i mając na uwadze ciągły rozwój systemów akwizycji i przetwarzania obrazów oraz aktualny stan techniki obliczeniowej, podjęto badania dotyczące sformalizowania oceny stopnia zrównoleglenia w przepływowych systemach wizyjnych. Pod tym kątem przeprowadzono wieloetapowy proces analizy, projektowania i implementacji trzech złożonych algorytmów wizyjnych.

Rozdział 2

Teza i cel prac

Proces implementacji algorytmów przetwarzania i analizy sygnału wizyjnego, jak każda dziedzina nauk technicznych, podlega ocenie ilościowej i jakościowej. Szacowanie efektywności w realizacji systemów wizyjnych wymaga określenia kryteriów oceny. Można zdefiniować szereg wskaźników jakościowych i ilościowych charakteryzujących platformę i sposób projektowania (programowania) algorytmów. Podstawowym dyskretnym wskaźnikiem jest wykonalność algorytmu na danej platformie. Kolejne ilościowe kryteria będące niejednokrotnie ograniczeniami to zapotrzebowanie na energię i pośrednio liczba cykli potrzebnych do wykonania obliczeń czy powierzchnia krzemu zajmowana przez architekturę obliczeniową.

W celu usystematyzowania zakresu prac przyjęto następujące założenia wstępne i wymogi odnośnie projektowanych systemów wizyjnych:

Z1 Przyjęto przepływowy schemat realizacji algorytmów przetwarzania obrazów zestawionych w zintegrowany tor wizyjny. Ma to na celu minimalizację liczby transferów danych oraz zmniejszenie zapotrzebowania na zasoby pamięciowe.

Z2 Jako podstawowy wymóg wykonalności ustalono płynne przetwarzanie strumienia wizyjnego bez pomijania danych niezależnie od treści obrazu oraz stopnia skomplikowania algorytmów wizyjnych.

Z3 Ustalono, że częstotliwość pracy platformy obliczeniowej będzie zdeterminowana tempem transferu pikseli obrazu w strumieniu wizyjnym z czujnika wizyjnego. Przesłanka, jaka przyświeca tej idei wyprowadzona została z faktu, iż moc elektryczna zużywana w elementach obliczeniowych produkowanych przy pomocy współczesnej technologii jest proporcjonalna do częstotliwości przełączania.

Z4 Założono, że przepływowa architektura obliczeniowa będzie w pełni dedykowana do toru wizyjnego przez co będzie pozbawiona strumieni instrukcji, co przyczyni się to do zmniejszenia zajmowanej powierzchni jak i zasobów pamięciowych. Minimalizacja powierzchni nie będzie jednak brane pod uwagę w analizach, chociaż ma ona istotny wpływ na zużycie energii.

Ze względu na duży transfer danych aplikacje przetwarzania sygnału wizyjnego należą do bardziej wymagających obszarów zastosowań techniki obliczeniowej. Jako podstawowy czynnik umożliwiający osiągnięcie tak nakreślonego celu autor rozprawy uznał wykorzystanie współbieżności w realizacji obliczeń. Podstawowa teza pracy została zatem sformułowana następująco:

Zastosowanie oceny stopnia zrównoleglenia algorytmów przetwarzania i analizy obrazów pozwala na określenie realizowalności systemu wizyjnego o dużej szybkości działania.

Wykazanie prawdziwości tej tezy zostanie dokonane poprzez osiągnięcie następujących celów cząstkowych:

a) Identyfikacja rodzajów równoległości możliwych do wykorzystania w przepływowej realizacji wybranych algorytmów wizyjnych.

b) Opracowanie metodyki realizacji algorytmów poprzez ich przekształcanie oraz wybór odpowiedniej architektury obliczeniowej.

c) W odniesieniu do algorytmów o czasie wykonania zależnym od treści strumienia wizyjnego: przebadanie jej wpływu na wykonalność algorytmu w sposób przepływowy.

d) Dobór ilościowych wskaźników umożliwiających szacowanie stopnia spełnienia założeń rozprawy oraz określenie miary zrównoleglenia celem przeprowadzenia analizy porównawczej, umożliwiającej weryfikację tezy rozprawy.

Ze względu na realizację postawionych celów konieczne okazało się dokonanie klasyfikacji algorytmów wizyjnych pod kątem organizacji danych oraz sposobu korzystania z pamięci. Oprócz przyjętych wstępnie założeń, realizacja pracy wymaga również identyfikacji fizycznych ograniczeń platform wykorzystanych do realizacji. Ze względu na wysoki stopień zrównoleglenia będą one dotyczyć głównie organizacji pamięci.

Rozdział 3

Zawartość rozprawy

Rozprawa składa się trzech części. Pierwszą część stanowią rozdziały wprowadzające zawierające motywację badań, tezę, przyjęte założenia oraz określone cele. W części drugiej obejmującej rozdziały 4 i 5 dokonano klasyfikacji platform oraz charakterystyki strumieniowych systemów przetwarzania sygnału wizyjnego. Część trzecia zawierająca rozdziały 6 i 7 przedstawia wyniki badań nad zastosowaniem równoległości w realizacji wybranych algorytmów wizyjnych.

W rozdziale 4 dokonano przeglądu platform obliczeniowych wykorzystywanych w realizacji systemów wizyjnych ze szczególnym uwzględnieniem rodzajów równoległości: równoległość operacji, równoległość danych. Przedstawiono kryteria wyboru platformy do prowadzenia badań, które opisano w części trzeciej rozprawy. Opisano wskaźniki, które pozwalają na ilościową ocenę efektów zrównoleglenia: przyspieszenie i efektywność. Dokonano również charakterystyki czynników, które mają wpływ na efektywność zrównoleglania.

W rozdziale 5 zawarto charakterystykę strumieniowych systemów wizyjnych i sprecyzowano zasadnicze założenia dotyczące sposobu ich działania. Przedstawiono ilościowe wskaźniki pozwalające na ocenę systemu strumieniowego: przepływność i opóźnienie transportowe. Wskazano na podstawowe trudności w strumieniowej realizacji algorytmów iteracyjnych i rekurencyjnych.

W rozdziale 6 opisano sposób wykorzystania różnych rodzajów równoległości dla realizacji strumieniowego toru wizyjnego wykorzystującego przestrzenny (lokalny i globalny) kontekst piksela. W pierwszej sekcji tego rozdziału przedstawiono podsystem formowania strumienia wizyjnego oraz interpolacji kolorów jako przykład zrównoleglonej realizacji algorytmu o lokalnym kontekście przestrzennym. W kolejnych sekcjach dokonano analizy równo-

ległości na poszczególnych etapach strumieniowej realizacji toru wizyjnego rozpoznającego ręcznie pisane cyfry. System, oprócz przestrzennego kontekstu lokalnego, wykorzystuje również algorytmy bazujące na globalnym kontekście przestrzennym.

W rozdziale 7 zaprezentowano etapy zrównoleglania systemu wideodetekcji w ruchu drogowym. Przedstawione tam algorytmy estymacji tła oraz detekcji ruchomych i nieruchomych pojazdów w obszarach detekcji wykorzystują temporalny rodzaj kontekstu. Opisana metoda bilansowania równoległości instrukcji i równoległości danych umożliwiła realizację złożonego systemu wideodetekcji przy istotnych ograniczeniach zasobów pamięciowych platformy rekonfigurowalnej.

W rozdziale 8 podsumowano wyniki badań w kontekście postawionej tezy i wyznaczonych celów. Przedstawiono też propozycję wykorzystania uzyskanych wyników badań oraz zarysowano perspektywę dalszych badań.

Rozdziały 4 i 5 mają charakter opisowy i zawierają aparat pojęciowy, który wykorzystany jest w rozdziałach 6 i 7 do przedstawienia wyników przeprowadzonych prac projektowych i badawczych. W dodatkach umieszczono przykładowe kody źródłowe demonstrujące zrównoleglenie wybranych podsystemów oraz szczegółowe procedury pozwalające na szacowanie przyspieszenia podczas zrównoleglania algorytmów wizyjnych.

Część II

Równoległość w systemach wizyjnych

Rozdział 4

Platformy obliczeniowe a systemy wizyjne

4.1 Kryteria wyboru platformy obliczeniowej dla systemów wizyjnych

Wybór platformy obliczeniowej podlega różnorakim kryteriom, zależnym od specyfiki zastosowania. Są to najczęściej: moc obliczeniowa (lub przyspieszenie w stosunku do innych rozwiązań), rozmiary, zużycie energii, możliwość programowania, zdolność do rekonfiguracji, czy koszt. Najczęściej, wykorzystanie stacjonarnego komputera z racji na któreś z wymienionych kryteriów jest kłopotliwe, lub nawet niemożliwe. Na szczególną uwagę, zwłaszcza na etapie badań i prototypowania zasługują sprzętowe platformy rekonfigurowalne FPGA, które mają coraz większe możliwości i są wykorzystywane w systemach oferowanych komercyjnie zarówno jako komponenty pomocnicze lub jako główne elementy przetwarzające [66][67][78][99]. W klasycznych (opartych o procesory ogólnego zastosowania) systemach widzenia maszynowego wykorzystuje się różnorodne techniki zwiększenia wydajności systemów wizyjnych poprzez: akcelerację obliczeń przy pomocy specjalizowanych jednostek obliczeniowych (ASIC (ang. *Application Specific Integrated Circuit*), procesory DSP, FPGA, GPU), zwielokrotnienie liczby jednostek obliczeniowych [7] lub zastosowanie rozwiązań heterogenicznych [14][27][74]. W rozwiązaniach wbudowanych obserwuje się jednak tendencję to wykorzystania homogenicznych elementów obliczeniowych lub zintegrowanych elementów systemów jednoukładowych SoC (ang. *System on Chip*).

Coraz częściej jako istotne kryterium przyjmuje się koszt w postaci energii niezbędnej do wykonania określonych obliczeń. Kryterium to ma zastosowanie w szeroko rozumianej technice obliczeniowej i jak sygnalizuje autor publikacji [43]

będzie odgrywać w najbliższej przyszłości istotną rolę z racji na kurczące się zasoby źródeł energii. Przykładowo, ilościową ocenę kosztu obliczeń w przenośnych urządzeniach do monitoringu stanu pacjentów w oparciu o sygnał elektrokardiograficzny przedstawiono w pracy [10]. Szczegółowa analiza aspektów energetycznych w projektowaniu systemów wizyjnych, choć wydaje się być istotna, ze względu na zakres pracy nie została uwzględniona w niniejszej rozprawie.

Ponieważ opisywane w rozprawie prace mają charakter badawczy, jako podstawowe kryterium wyboru platformy przyjęto zdolność elementu obliczeniowego do rekonfiguracji. Układy rekonfigurowalne FPGA cechują się dużym stopniem rekonfigurowalności, w tym również na niskim poziomie granulacji. Wytworzenie prototypu architektury obliczeniowej opartej o platformę rekonfigurowalną, jakkolwiek pracochłonne, pozwala uniknąć technologicznej procedury wytworzenia układu scalonego ASIC, co jest znacznie bardziej kosztowne i czasochłonne. Wyprodukowanie specjalizowanego układu scalonego stanowi często drugi etap wdrożenia opracowywanego rozwiązania do powszechnego użytku.

Metod usprawnienia obliczeń upatruje się również w specjalizowanej konstrukcji czujników wizyjnych, optymalnej organizacji danych wizyjnych lub dystrybucji zadań na pracujące współbieżnie węzły sieci czujników wizyjnych [44]. Pomimo dostępnych możliwości obliczeniowych opartych o wielordzeniowe architektury procesorów ogólnego przeznaczenia, nadal istnieje potrzeba poszukiwania bardziej efektywnych metod realizacji systemów wizyjnych.

4.2 Klasyfikacja systemów obliczeniowych

Ze względu na strukturę elementu przetwarzającego, można dokonać podziału platform obliczeniowych dla systemów wizyjnych pod kątem stopnia współbieżności oraz sposobu programowania. Flynn [22] zaproponował podział systemów obliczeniowych na 4 kategorie:

SISD Single Instruction Stream, Single Data Stream,

SIMD Single Instruction Stream, Multiple Data Stream,

MISD Multiple Instruction Stream, Single Data Stream,

MIMD Multiple Instruction Stream, Multiple Data Stream.

Taksonomia ta bazuje na wzajemnych relacjach ilościowych pomiędzy liczbą strumieni rozkazów i strumieni danych. W alternatywnej klasyfikacji systemów obliczeniowych dla systemów wizyjnych [20], częściowo powiązanej z taksonomią Flynn'a, jako podstawowe kryterium przyjęto rodzaj równoległości:

DLP Data Level Parallelism,

ILP Instruction Level Parallelism,

OLP Operation Level Parallelism,

TLP Task Level Parallelism.

W opisie wielordzeniowych architektur współczesnych procesorów CPU ogólnego przeznaczenia również używa się podobnej terminologii (DLP, ILP, TLP). Akronim TLP (ang. Thread Level Parallelism) rozumiany jest tutaj jako równoległość wykonywania wątków w scalonym układzie wieloprocesorowym CMP (ang. Chip Multi-Processor) [8].

Nieco odmienną klasyfikację zaproponowali autorzy pracy [14] w odniesieniu do specjalizowanego akceleratora sprzętowego o nazwie Xputer. Zaproponowano podział poziomów równoległości ze względu na zadania, pętle, instrukcje i operacje:

OLP Operation Level Parallelism,

SLP Statement Level Parallelism,

LLP Loop Level Parallelism,

TLP Task Level Parallelism.

Klasyfikacja ta wynika zarówno ze specyficznej architektury Xputera zorientowanej na dane, jak i narzędzi (w tym kompilatora języka X-C) służących do budowania aplikacji.

W odniesieniu do systemów programowych można zauważyć, że system SIMD oraz wykorzystujący równoległość danych (DLP) są sobie równoważne ze względu na wykonanie tej samej operacji na różnych kwantach danych. Architekturę wykorzystującą równoległość instrukcji (ILP) można porównać z systemem MISD lub MIMD w przypadku jednoczesnej obecności DLP. Analogii można się również doszukać pomiędzy równoległością zadań (TLP) a architekturą MIMD, gdzie występuje wiele strumieni rozkazów i danych. Rozróżnienie pomiędzy równoległością instrukcji (ILP), a równoległością operacji (OLP) ma naturę ilościową i odnosi się do liczby wykonywanych rozkazów. Do systemu OLP zaliczono architektury o bardzo długim słowie instrukcji: VLIW (ang. *Very Long Instruction Word*. Przyjęto bowiem że pojedynczą operację stanowi wystarczająco obszerny zbiór pojedynczych instrukcji.

Każdy cyfrowy system obliczeniowy, niezależnie od architektury i stopnia równoległości potencjalnie można opisać przy pomocy formalizmów stosowanych w opisie i syntezie układów cyfrowych. Podejście to polega na wykorzystaniu pojęcia automatów współbieżnych CFSM (ang. *Concurrent Finite State Machine*) [4][2] opisanych zbiorem stanów wewnętrznych i warunków przejść między stanami. Metoda ta jest wykorzystywana między innymi w badaniu i implementacji reprogramowalnych sterowników logicznych [3] o wysokiej pewności działania. Systemy tego typu zorientowane są raczej na obsługę sekwencji zdarzeń niż przetwarzanie

dużych ilości danych. Stopień złożoności współczesnych systemów obliczeniowych (o dużej liczbie elementów synchronicznych i kombinacyjnych) sprawia jednak, iż ze względów praktycznych, teorię automatów skończonych stosuje się raczej do wybranych elementów architektury obliczeniowej niż do całego złożonego systemu.

Klasyfikacja systemów obliczeniowych ze względu na rodzaj równoległości w pracy [20] stosowana jest wyłącznie w odniesieniu do systemów programowanych z wyraźnie wyszczególnionym strumieniem instrukcji. Autor niniejszej rozprawy uważa jednak, że podział ten ma również zastosowanie w dedykowanych systemach obliczeniowych pozbawionych strumienia instrukcji. Podejście to wymaga jednak odmiennej interpretacji pojęć takich jak *instrukcja* i *operacja*. Wcześniej wspomniano, że są one w pewnym sensie tożsame. Należy zaznaczyć, iż równoległość OLP jest uwarunkowana równoległością DLP, ponieważ jednoczesne wykonanie wielu operacji jest możliwe tylko jeśli dane są dostępne w tym samym czasie.

W systemach procesorowych instrukcją jest kod rozkazu maszynowego przechowywany w wyznaczonym obszarze pamięci. Rozkaz ten determinuje działanie podsystemów procesora celem uzyskania określonych efektów (np. porównanie dwóch liczb, zapis lub odczyt rejestrów).

W systemach sprzętowych nie występuje pamięć programu, która byłaby sekwencyjnie odczytywana w trakcie działania. Każdy podsystem i jego części mają przypisane ściśle określone funkcje a ich działanie jest zdeterminowane przez projektanta na etapie implementacji i nie zmienia się podczas normalnej pracy systemu. W odniesieniu do jednostek funkcjonalnych można raczej powiedzieć, że wykonują one *operacje* a nie *instrukcje*.

4.2.1 Programowane i konfigurowane elementy obliczeniowe

Wcześniej zauważono, że klasyfikacja Flynn'a nie jest w pełni adekwatna do kategoryzacji systemów, w których nie można jednoznacznie wskazać strumienia instrukcji. Istnieją systemy o architekturze algorytmu "zaszytej" niejako w strukturze samego elementu obliczeniowego. Paradygmat ten określono w pracach [14][29][43] jako "obliczenia w przestrzeni" (ang. *computing in space*) w odróżnieniu od sekwencyjnego wykonywania strumienia instrukcji w klasycznych programowanych procesorach. Przykładem takich systemów mogą być dedykowane układy ASIC lub układy PLD (ang. *Programmable Logic Devices*) przeznaczone do wykonywania ściśle określonych zadań: (np. kompresja, dekompresja, kodowanie, dekodowanie danych). Systemy rekonfigurowalne budowane są najczęściej w oparciu o układy FPGA. Podobnie jak specjalizowane układy ASIC, cechują się one architekturą dedykowaną do danego zadania, z tą jednak różnicą, iż strukturę układu rekonfigurowalnego można zmienić zależnie od potrzeb. Możliwe jest jednak zastosowanie technologii ASIC czy układów rekonfigurowalnych do

tworzenia systemów programowanych. Jest to przypadek szczególny choć współcześnie często wykorzystywany [20][107]. Na potrzeby niniejszej pracy zawężono znaczenie następujących pojęć, które pojawiają się w kolejnych rozdziałach, a w języku inżynierskim i w terminologii naukowej mogą być wieloznaczne:

programowany element obliczeniowy jest system cyfrowym zdolnym do pobierania dekodowania i wykonywania strumienia instrukcji (rozkazów) przygotowanych i zadanych przez programistę lub odpowiednie narzędzie, bez zmiany architektury elementu obliczeniowego,

konfigurowany element obliczeniowy jest system cyfrowym o architekturze dedykowanej do wykonywania ściśle określonego zadania do którego realizacji nie jest potrzebny strumień instrukcji, gdyż cała funkcjonalność elementu obliczeniowego jest zakodowana w architekturze elementu obliczeniowego.

W myśl przejętej zasady, element obliczeniowy o architekturze które może ulec zmianie należy określić mianem rekonfigurowany. W literaturze fachowej przejęło się jednak stosowanie pojęć *programowalny* lub *reprogramowalny* również w odniesieniu do układów PLD.

4.3 Obliczenia równoległe

Zdolność do zmiany programu, czy możliwość adaptacji architektury systemu obliczeniowego do konkretnego zadania, same w sobie nie stanowią jednak wyznacznika wydajności czy efektywności. W celu zwiększeniu mocy obliczeniowej powszechnie wykorzystuje się równoległość (4.1).

$$y = F(x) = F^n(x)||F^{n-1}(x)||...||F^2(x)||F^1(x) \tag{4.1}$$

gdzie:

x – dane wejściowe,
y – dane wyjściowe,
$||$ – symbol jednoczesności wykonania operacji,
F^i – poszczególne operatory obrazu, $i \in \{1, ..., n\}$.

Równoległość wykorzystywana jest w systemach SIMD, MIMD, MISD, gdzie występuje zwielokrotnienie strumienia danych lub strumienia instrukcji. Wiele danych może być przetwarzanych jednocześnie i/lub wiele operacji może być wykonywanych jednocześnie na danych wejściowych. W systemach wizyjnych, realizowanych z wykorzystaniem systemów procesorowych, wykorzystuje się geometryczną równoległość [79] na poziomie matrycy pikseli w ramce obrazu oraz

jednoczesność pojawiania się danych reprezentujących pojedynczy piksel obrazu (np. w przypadku obrazu barwnego).

W konfigurowanych i dedykowanych systemach obliczeniowych równoległość również jest wykorzystywana, pomimo iż nie występują tam strumienie instrukcji. Co więcej, w znaczącym stopniu ułatwione jest zrównoleglanie dzięki eliminacji narzutów komunikacyjnych na akwizycję strumieni rozkazów.

4.3.1 Przyspieszenie w systemach obliczeniowych

Do oceny wykonania zadań obliczeniowych w systemach wieloprocesorowych i równoległych [62] stosuje się przyspieszenie wyznaczone formułą (4.2).

$$S_m = \frac{T_1}{T_n} \tag{4.2}$$

gdzie:

S_m – przyspieszenie,
T_1 – czas wykonania programu sekwencyjnego,
T_n – czas wykonania programu zrównoleglonego na n procesorach (elementach obliczeniowych),
n – liczba procesorów (elementów obliczeniowych).

Pełne wykorzystanie równoległości systemu obliczeniowego nie zawsze jest możliwe z powodu natury implementowanych algorytmów. Obserwacja ta sformalizowana jest w prawie Amdahl'a [5][8] w odniesieniu do proporcji pomiędzy ilością kodu zrównoleglonego i wykonywanego sekwencyjnie. Zgodnie z regułą (4.3) teoretyczne przyspieszenie po zastosowaniu równoległych obliczeń jest nadal mocno zależne od tej części programu, która wykonywana jest sekwencyjnie.

$$S_a = \frac{1}{s + \frac{p}{n}} \tag{4.3}$$

gdzie:

S_a – przyspieszenie,
p – czas obliczeń zrównoleglonych dla jednego procesora sekwencyjnego,
s – czas obliczeń niezrównoleglonych,
– p i s są znormalizowane tak, aby spełnić warunek: $p + s = 1.0$

W systemach wieloprocesorowych realna wartość przyspieszenia programu jest mniejsza od teoretycznej (wynikającej z liczby procesorów) ze względu na narzuty komunikacyjne oraz konieczność współdzielenia zasobów takich jak pamięć czy magistrale [6]. Własność tą opisuje wzór na efektywność (4.4) która jest miarą jakości wykorzystania współbieżnych zasobów.

$$E_n = \frac{S}{n} \tag{4.4}$$

gdzie:

S – przyspieszenie,
E_n – efektywność zrównoleglenia $E_n \in (0; 1\rangle$.

W idealnym przypadku efektywność wynosi 1.0 co oznacza przyspieszenie S proporcjonalne do liczby wykorzystanych procesorów lub elementów obliczeniowych. W rzeczywistości jednak, dla wielkiej liczby procesorów, wartość ta najczęściej znacznie odbiega od jedności. Powyżej granicznej liczby procesorów może być nawet malejącą funkcją liczby procesorów n. W powszechnie wykorzystywanych systemach wieloprocesorowych czynnik powodujący obniżenie efektywności jest wartością mierzalną, jednak nie jest zależny jedynie od zaimplementowanego programu. Wpływ na to mają również czynniki takie jak środowisko wykonywania programu (system operacyjny) i sama architektura sprzętowa systemu komputerowego. Rozpatrując jednak aspekty architektury systemu obliczeniowego można uwzględnić we wzorze (4.3) również operacje związane z arbitrażem dzielonych zasobów podczas współbieżnego wykonania programu. Można zauważyć, że przyspieszenie jest wypadkową szeregu czynników, które są przeciwstawne i mocno zależą od skali zadania obliczeniowego.

Gustafson i Baris zaproponowali [34] użycie odmiennego sposobu wyznaczania przyspieszenia, skalowanego do czasu wykonania, a nie rozmiaru problemu (jak w prawie Amdahl'a). W prawie Gustafson-Barsis'a przyspieszenie określa formuła (4.5), która wyznacza liniową zależność od liczby procesorów z uwzględnieniem wpływu wieloprocesorowej implementacji na czas wykonania.

$$S'_n = n - (n-1)s' \tag{4.5}$$

gdzie:

S'_n – przyspieszenie skalowane do czasu wykonania,
s' – znormalizowany czas wykonania sekwencyjnej części algorytmu.

Shi [86] dowodzi, że oba wzory (4.3) i (4.5) są równoważne. Wykorzystano w nich jednak różne miary czasu wykonania niezrównoleglonej części algorytmu: s i s'. W rozdziałach 6 i 7 niniejszej rozprawy zostaną przedstawione wyniki oceny przyspieszenia zrealizowanych sprzętowych systemów wizyjnych z równoczesnym wykorzystaniem obu reguł. Dla każdej z metod oceny otrzymano identyczne wyniki. Ponieważ opisane tam prace przebiegały etapami, przyspieszenie również było wyznaczane w poszczególnych krokach. W dodatku A opisano przyrostową i iteracyjną metodę wyznaczania przyspieszenia, które są przydatne w projektowaniu współbieżnych systemów obliczeniowych i potwierdzają równoważność praw Gustafson'a i Amdahl'a.

Karp i Flatt zapropnowali [58] metrykę (4.6) do oceny wieloprocesorowego wykonania programu poprzez pomiar ułamka e_n przy wykorzystaniu różnej liczby procesorów n. Metodykę wykorzystania pomiaru przyspieszenia przestawiono w pracy [65]. Mała wartość e_n świadczy o dobrej skalowalności programu na danej maszynie, a znaczące zwiększenie wartości e_n, przy wzroście n i małym przyspieszeniu S, świadczy o zbyt dużych narzutach współbieżnego wykonania programu.

$$e_n = \frac{\frac{1}{S_m} - \frac{1}{n}}{1 - \frac{1}{n}} \tag{4.6}$$

gdzie:

e_n – metryka dla n procesorów,
S_m – przyspieszenie zmierzone dla N procesorów.

Jeśli zwiększenie liczby procesorów nie powoduje wzrostu e_n, przyczyną małej wydajności jest sama struktura algorytmu lub sposób jego implementacji. Zastosowanie tej metody oceny wymaga użycia wieloprocesorowego systemu komputerowego skalowalnego w szerokim zakresie ze względu na liczbę procesorów. W rozdziale 6.5.1 zostanie pokazane wykorzystanie metryki do oceny dedykowanych systemów cyfrowych o architekturze i stopniu równoległości dobieranym przez projektanta.

4.3.2 Superliniowe przyspieszenie

W pracy [13] zauważono, iż prawo Amdahl'a zakłada, że przyspieszenie zrównoleglonego kodu jest wprost proporcjonalne do liczby procesorów. W poprzednim akapicie wspomniano, że jest to zależność jedynie teoretyczna. Mimo to w systemach wieloprocesorowych jest możliwe [35] uzyskanie efektywności bliskiej 1 lub większej: $E_n \geqslant 1.0$. Taka zależność zwana "superliniową" jest możliwa dzięki wielopoziomowej strukturze pamięci współczesnych komputerów:

poziom 1 pamięć stronicowana na dysku,

poziom 2 pamięć operacyjna RAM,

poziom 3 pamięć podręczna 1-go i 2-go poziomu,

poziom 4 rejestry.

Ilościowo, efekt ten zależny jest od relacji pomiędzy rozmiarami struktur danych i blokami pamięci na poszczególnych poziomach oraz sposobu partycjonowania danych wejściowych podczas zrównoleglenia. W przypadku systemów wizyjnych realizowanych w oparciu o systemy wieloprocesorowe podstawowym kwantem danych jest ramka obrazu. Dla niewielkich rozdzielczości (512×512) cała ramka obrazu może zostać zapisana w pamięci podręcznej, co pozwala na zaobserwowanie efektu superliniowego przyspieszenia jeśli:

- każdy z procesorów przetworzy inną ramkę obrazu,
- każdy z procesorów wykona inną operację na kopii tej samej ramki obrazu,
- ramka obrazu zostanie podzielona pomiędzy poszczególne procesory.

Efekt ten został zaobserwowany również podczas zrównoleglenia operacji na obrazach barwnych [16] z wykorzystaniem rozszerzeń MMX (ang. *MultiMedia eXtension*) i SSE (ang. *Streaming SIMD Extension*) procesorów ogólnego przeznaczenia. W ogólnym przypadku liczba czynników, która ma wpływ na możliwość wystąpienia efektu superliniowego przyspieszenia jest znacząca. Trudno jest więc ująć wszystkie w prostą formułę analityczną. Wniosek, jaki wynika z tej obserwacji jest jednak istotny nie tylko dla systemów wieloprocesorowych. Hipoteza która mówi, że w efektywnej realizacji obliczeń istotna jest ilość pamięci i sposób dostępu do danych, została potwierdzona doświadczeniem autora rozprawy w odniesieniu do dedykowanych systemów wizyjnych opisanych w rozdziałach 6 i 7 rozprawy.

4.3.3 Platformy programowane a konfigurowalne

Architektury, sposoby programowania i konfiguracji systemów procesorowych oraz rekonfigurowalnych różnią się istotnie. Porównanie ich efektywności poprzez zastosowanie jednorodnych testów wydajnościowych jest trudne do wykonania. Dodatkowo, częstotliwości pracy procesorów w systemach komputerowych są przeważnie dwa rzędy wielkości wyższe niż w przypadku najszybszych układów FPGA wykorzystywanych w prototypowaniu dedykowanych systemów obliczeniowych. W pracy [28] zaproponowano sposób oceny platform obliczeniowych, który pozwala na porównanie platform dedykowanych i procesorowych przy pomocy liczby cykli zegara (4.7) potrzebnych do wykonania określonego zadania. Zniwelowano

w ten sposób wpływ częstotliwości taktowania systemu, która jest pochodną procesu technologicznego wykonania układu scalonego i nic nie mówi o architekturze elementu obliczeniowego ani o sposobie implementacji algorytmu.

$$C_t = \frac{T_t}{f_{clk}} \tag{4.7}$$

gdzie:

C_t – ilość cykli potrzebna do wykonania zadania t,
T_t – czas wykonania zadania t,
f_{clk} – częstotliwość pracy systemu obliczeniowego.

Liczba cykli C_t wyznaczana jest poprzez pomiar czasu wykonania zadania. Zestawienie liczby cykli wykonania tej samej aplikacji na różnych platformach pozwala na ich porównanie. Dotyczy to zarówno systemów dedykowanych o bardzo wysokim stopniu zrównoleglenia jak i komputerów jedno i wieloprocesorowych z uwzględnieniem narzutów komunikacyjnych oraz wynikających ze środowiska uruchamiania programu. Jako miarę zrównoleglenia autor pracy [28] proponuje zastosować iloraz (4.8). Stopień zrównoleglenia S_C jest współczynnikiem łatwym do zmierzenia, jednak jego wartość nie jest porównywalna z realnym przyspieszeniem w sensie formuły (4.2). Mimo wysokiej wartości $S_C > 1.0$ czas wykonania zadania t na platformie $P1$ może być mniejszy niż na platformie $P2$ ze względu na znaczną dysproporcję częstotliwości.

$$S_C = \frac{C_t^{P1}}{C_t^{P2}} \tag{4.8}$$

gdzie:

S_C – stopień zrównoleglenia zadania t na platformie $P2$ w odniesieniu do platformy $P1$,
C_t^{P1} – liczna cykli dla zadania t na platformie $P1$,
C_t^{P2} – liczna cykli dla zadania t na platformie $P2$.

4.3.4 Obliczenia potokowe

Przyspieszanie obliczeń w systemach wizyjnych jest możliwe głównie dzięki równoległości. Nie zawsze jednak równoległość ta jest możliwa do wykorzystania na poziomie danych, instrukcji, czy operacji. Realizacja współbieżnych obliczeń opisanych formułą (4.1) jest mocno ograniczona ze względu na potencjalną możliwość wystąpienia wzajemnych relacji pomiędzy kwantami danych lub kolejnością

wykonywania operacji. Znacznie lepsze rezultaty daje zastosowanie równoległości wraz z potokową organizacją struktur danych w czasie (4.9) jak to zaproponowano w pracy [79].

$$y = F(x) = F^n(F^{n-1}(...(F^2(F^1(x)))...)) \tag{4.9}$$

gdzie:

x – dane wejściowe,

y – dane wyjściowe,

F^i – poszczególne operacje na danych, $i \in \{1, ..., n\}$.

Niskopoziomowe operacje przetwarzania obrazów cechują się najczęściej dużą regularnością struktury danych przestrzennego kontekstu przetwarzanego piksela oraz jej niezmienniczością ze względu na lokalizację w ramce obrazu. Przykładem operacji, która nie spełnia powyższej własności jest np. korekcja zniekształceń obiektywu [68]. Niwelacja zniekształceń geometrycznych wymusza dostęp do danych niezgodny z ich organizacją w strumieniu wizyjnym z kamery. W potoku regularnym zarówno ilość jak i struktura danych wejściowych i wyjściowych są zachowane.

Analiza i rozpoznawanie wykonywane po etapie wstępnym wymagają najczęściej przetwarzania cech, które rozmiarem i strukturą mogą się różnić się zależnie od treści. Dodatkowym czynnikiem powodującym niejednorodność może być również struktura algorytmu, który wykorzystuje np. rekurencję lub wielokrotne iteracje.

4.4 Przetwarzanie i analiza sygnału wizyjnego

Operacje w systemach wizyjnych dzielone są na kilka etapów. Na każdym z nich wykonywane są obliczenia, które pobierają argumenty wejściowe i zwracają rezultaty w określonej postaci. Zależnie od etapu argumentami wejściowymi i rezultatami mogą być dane, informacja i ostatecznie wiedza [90]. W tabeli 4.1 danym odpowiadają piksele i ramki obrazu w strumieniu wizyjnym, informację stanowią cechy obrazu (wyznaczone w procesie analizy) oraz struktury danych opisujące treść obrazu na średnim poziomie abstrakcji. Wiedza reprezentuje opis treści sygnału wizyjnego na najwyższym poziomie uwzględniając rozpoznane obiekty i relacje pomiędzy nimi w czasie i przestrzeni.

4.4.1 Przetwarzanie wstępne

Przetwarzanie wstępne sprowadza się najczęściej do operacji bezkontekstowych na pikselach obrazu oraz filtracji liniowej [54][102] i nieliniowej z regularnym kontekstem przestrzennym.

Tabela 4.1: Klasyfikacja operacji wizyjnych.

etap	operand		kontekst
1. przetwarzanie wstępne	«dane»	piksele,obrazy	punktowy, lokalny
2. analiza i ekstrakcja cech	«informacja»	cechy, struktury danych	lokalny, globalny
3. rozpoznawanie			globalny
4. rozumienie	«wiedza»	opis sceny, relacje między obiektami	

4.4.2 Segmentacja

Kolejną operacją jest zazwyczaj segmentacja obrazu która polega na przydzieleniu pikseli do określonej klasy. Chociaż liczba klas przyporządkowania może przyjmować znaczne wartości, w najprostszym przypadku piksele przydzielane są do dwóch kategorii: $0 \equiv$ tło, $1 \equiv$ obiekt. Najprostszym sposobem segmentacji jest binaryzacja poprzez porównanie bieżącego piksela z zadanymi lub wyznaczonymi automatycznie progami. W bardziej wymagających zastosowaniach wykorzystuje się techniki segmentacji poprzez łączenie lub podział obszarów [95], analizę tekstur jak również metodę przepływu optycznego (ang. *Optical Flow*) [24], elementy metody teorii skal (ang. *Scale Space Theory*) [64], analizę temporalną czy też wielomodalne klasyfikatory statystyczne [23][91].

4.4.3 Analiza i rozpoznawanie

Po segmentacji, obraz poddawany jest najczęściej etykietowaniu w celu identyfikacji obiektów w obrazie oraz określenia ich liczby [95]. Następujące po indeksacji operacje polegają na analizie przestrzennie wydzielonych zbiorów pikseli. Analiza ta obejmuje badanie właściwości kształtów poprzez zastosowanie współczynników kształtów lub wyliczanie momentów dla wydzielonych obiektów [93]. Można również wyznaczyć statystyczne właściwości zbioru pikseli należących do obszaru danego obiektu w oparciu o jasność pikseli w paśmie widzialnym, w podczerwieni [103] lub z uwzględnieniem informacji barwnej [52][81]. Możliwości torów wizyjnych nie ograniczają się jednak do analizy poszczególnych ramek obrazów. Wykorzystuje się również informację temporalną do śledzenia obiektów w kolejnych ramkach obrazu [80].

Ostatni etap obróbki sygnału wizyjnego może stanowić rozpoznawanie oraz automatyczne rozumienie obrazu [76][96]. Jego celem jest automatyczne wygenerowanie opisu treści obrazu z wykorzystaniem odpowiedniej gramatyki dostosowanej do obszaru zastosowań (np. diagnostyka medyczna).

4.5 Granulacja w systemach obliczeniowych

Termin *granulacja* w kontekście prac opisanych w rozprawie należy rozumieć dwojako, zależnie od kontekstu. W pierwszym znaczeniu odnosi się do granulacji elementów obliczeniowych systemu rekonfigurowalnego. Alternatywne rozumienie odnosi się do sposobu kwantyzacji danych które są przetwarzane.

4.5.1 Granulacja systemów rekonfigurowalnych

Granulacja elementów obliczeniowych jest cechą różnicującą rekonfigurowalne platformy obliczeniowe. Poziom granulacji nie zależy od aplikacji realizowanej na danej platformie. W literaturze [40][41] przyjęty został podział systemów rekonfigurowalnych na:

coarse grain architectures architektury gruboziarniste,

fine grain architectures architektury drobnoziarniste,

Do systemów o wysokim poziomie granulacji zaliczają się np. układy FPGA, które umożliwiają rekonfigurację na poziomie pojedynczych bitów. Mianem architektur o niskim poziomie granulacji można określić różnego rodzaju specjalizowane systemy obliczeniowe, które udostępniają wydzielone bloki funkcjonalne o predefiniowanych funkcjonalnościach i ustalonych formatach danych i rozmiarach magistral (np. 4, 8, 16, 32 bity). Przegląd tego typu systemów przedstawiono w pracy [40]. Badania dotyczące automatycznego doboru gruboziarnistego systemu obliczeniowego przedstawiono w pracy [97].

W przypadku współczesnych układów rekonfigurowalnych FPGA [106][108] jednoznaczne przypisanie do jednej z kategorii nie jest możliwe. Oprócz cel logicznych zawierają one dedykowane bloki funkcjonalne, takie jak pamięci blokowe, jednostki arytmetyczne DSP czy sekwencyjne procesory (PowerPC).

Do badań opisanych w rozprawie wybrano platformę o wysokim poziomie granulacji opartą o technologię FPGA, która gwarantuje większą elastyczność, choć jak podaje autor pracy [41], systemy gruboziarniste są pod wieloma innymi względami bardziej efektywne (czas konfiguracji, rozmiar pamięci konfiguracji zużycie energii). W dalszych rozdziałach pracy wszelkie odwołania do granulacji i poziomu rozdrobnienia będą dotyczyły organizacji danych a nie struktury platformy obliczeniowej.

4.5.2 Granulacja danych w przetwarzaniu obrazów

Przedstawiona klasyfikacja operacji i struktur danych jest istotna z punktu widzenia funkcjonalnego. Ma również ogromny wpływ na sposób implementacji algorytmów wizyjnych w systemach komputerowych a w szczególności w systemach

dedykowanych. Realizacja aplikacji wizyjnych na platformach sprzętowych umożliwia korzystanie z masowej równoległości, jednak wiąże się to z ograniczeniami wynikającymi z natury narzędzi projektowych, z organizacji zasobów pamięciowych i sposobu dostępu do danych oraz ze struktury elementu obliczeniowego. Ilościowe relacje pomiędzy liczbą elementów przetwarzających a rozmiarami przetwarzanych operandów stanowią podstawę klasyfikacji elementów współbieżnych systemów wizyjnych ze względu na stopień granulacji danych:

fine grain	wysoki poziom rozdrobnienia (piksele),
medium grain	średni poziom rozdrobnienia (obszary ROI)*,
coarse grain	niski poziom rozdrobnienia (ramki obrazu).

Stopień granulacji ma istotny wpływ na zużycie pamięci i transfer danych pomiędzy elementami przetwarzającymi. Charakterystyka ta dotyczy raczej sposobu implementacji algorytmów wizyjnych i architektur niż funkcji operacji przetwarzania i analizy obrazów. W rozdziale 5.1.2 pokazano, że niektóre algorytmy rekurencyjne lub iteracyjne nie mogą być zrealizowane z wykorzystaniem wyłącznie wysokiego poziomu rozdrobnienia (*fine grain*) ponieważ wymagają globalnego kontekstu lub dostępu do pikseli niezgodnego z ich organizacją w strumieniu wizyjnym.

Aspekty wielokryterialnej optymalizacji dedykowanej architektury SIMD do przetwarzania obrazu na pierwszych trzech etapach szczegółowo przedstawiono w publikacji [20]. Praca koncentruje się wokół takiego doboru mechanizmu komunikacji (danych i rozkazów) dla elementów przetwarzających, który zapewni krótki czas obliczeń oraz niskie zużycie zasobów i energii. Problemy związane z efektywną dystrybucją danych wejściowych, rezultatów, jaki i rozkazów pomiędzy elementami przetwarzającymi wynikają bezpośrednio z definicji kontekstu niezbędnego do wykonania obliczeń dla danej operacji przetwarzania obrazów. Dobrze uwarunkowane są te zadania, które wykorzystują jedynie kontekst lokalny. Jak wynika z przedstawionych tam rezultatów, wykorzystanie kontekstu globalnego lub zastosowanie nieregularnego dostępu do danych wiąże się ze zwiększeniem zużycia zasobów i pobieranej mocy. Wpływ architektury współbieżnego akceleratora konwolucji obrazu na zużycie energii przedstawiono również w pracy [57]. Autorzy tej pracy podają, że wielopoziomowa organizacja pamięci przyczyniła się nie tylko do osiągnięcia wydajności współczesnych temu rozwiązaniu procesorów DSP, ale również pozwoliła na 35-krotną redukcję zużywanej energii przy ponad 8-krotnym zmniejszeniu częstotliwości pracy systemu.

*ang. *Region Of Interest*

Rozdział 5

Strumieniowy system wizyjny

Alternatywne podejście polega na zwiększeniu częstotliwości pracy elementu przetwarzającego i dzieleniu czasu jednostek obliczeniowych pomiędzy różne fragmenty wykonywanego programu. Taki model przetwarzania danych obwiązuje we współczesnych rozwiązaniach opartych o mikrokontrolery i procesory z architekturą harwardzką lub von Neuman'a. Programowalność (na poziomie kodu wykonywalnego) jest niewątpliwą zaletą takich rozwiązań. Jest ona jednak okupiona stosunkowo dużym wpływem ilości przetwarzania danych na wydajność i opóźnienie. Dotyczy to w szczególności systemów obliczeniowych przetwarzających strumieniowane dane. Utrzymanie jakości (opóźnienie, wydajność) przetwarzania przy jednoczesnym wzroście ilości danych wejściowych lub zwiększeniu ilości obliczeń w systemie wymagałoby zwiększenia częstotliwości pracy elementu przetwarzającego i ewentualnie zwiększenia przepustowości magistral do pamięci przechowujących dane. W obu przypadkach efektem ubocznym będzie zwiększenie mocy rozpraszanej w systemie obliczeniowym.

Rekonfigurowalne systemy cyfrowe wprawdzie nie oferują częstotliwości pracy porównywalnej ze współczesnymi procesorami ogólnego przeznaczenia, jednak stopień zrównoleglenia czyni je szczególnie przydatnymi do współbieżnych obliczeń na strumieniowanych danych. Zrównoleglenie wielu niezależnych operacji jest naturalne dla układów cyfrowych ze względu na jednoczesny dostęp do zrównoleglonych zasobów obliczeniowych i pamięciowych. Ma to kluczowe znaczenie w realizacji algorytmów przetwarzania i analizy sygnału wizyjnego.

W technice komputerowej powszechnie przyjął się model systemu obliczeniowego opracowany przez von Neuman'a, w którym nacisk położony jest na szeregowanie instrukcji. W pracy [39] jako alternatywę zaproponowano model architektury zorientowanej na przepływ danych (PDD ang. *Procedurally Data-Driven*) gdzie jednostkę szeregowania instrukcji (ang. *instruction sequencer*) zastąpiono jednostką szeregowania danych (*data sequencer*). Inne określenia użyte w tejże

publikacji i innych [38][62], a dobrze oddające sposób działania tego typu systemu to: "komputer przepływowy", "architektura wyzwalana transportem danych" oraz "*data-stream-driven computing*" (ang. *obliczenia wyzwalane transportem danych*).

Konfiguracja systemu obliczeniowego zorientowanego na transport danych odbywa się poprzez ustalenie schematu komunikacji współbieżnie działających, rekonfigurowalnych jednostek rALU (ang. *reconfigurable Arithemtic-Logic Unit*). Wyeliminowano w ten sposób konieczność formowania strumienia instrukcji przez jednostkę generacji adresu AGU (ang. *Address Generator Unit*). Formowanie strumieni z danych przechowywanych w masowej pamięci nadal wymaga ich adresowania poprzez zbiór jednostek GAG (ang. *Generaic Address Generator*). Źródłem strumieni danych mogą być również jednostki rALU komunikujące się między sobą, wówczas stosowanie jednostek adresujących GAG jest zbędne, o ile dane nie są zapisywane do pamięci masowej. Przedstawiony model jest na tyle ogólny, iż pozwala opisać większość dedykowanych systemów obliczeniowych CCM (ang. *Custom Computing Machine*) stosowanych w systemach wbudowanych jak i w akceleratorach wysoko-wydajnych systemów obliczeniowych HPC (ang. *High Performance Computing*), niezależnie od przyjętej metodologii projektowania czy użytego języka opisu sprzętu lub języka programowania.

W przeważającej większości wejściem systemów wizyjnych jest interfejs, który odbiera strumień sygnał wizyjny z jednej lub z wielu kamer. Zawsze następuje digitalizacja sygnału wizyjnego i zamiana na cyfrowy strumień pikseli obrazu. Format rezultatów przetwarzania i analizy obrazu zależy od rodzaju operacji wizyjnych oraz od architektury systemu. Może to być strumień wizyjny o parametrach zgodnych ze strumieniem wejściowym lub innego rodzaju informacje opisujące treść sygnału wizyjnego. Przedstawiona w założeniach rozprawy idea strumieniowego systemu obliczeniowego, w którym tempo działania wyznacza transport danych w strumieniu, jest więc naturalna do realizacji systemów wizyjnych. Ze względu na transport sygnału wizyjnego, narzucony działaniem czujnika wizyjnego kamery, wydawać by się mogło, że jednostka generacji adresów GAG jest zbędna w tego typu systemach. Stwierdzenie to jest prawdziwe dla większości operacji wstępnego przetwarzania obrazów wykorzystujących lokalny kontekst przestrzenny [53][102]. Istnieją jednak takie operacje analizy i rozpoznawania obrazów, które wymagają temporalnego lub globalnego przestrzennego kontekstu obrazu. Wówczas konieczne jest składowanie ramek obrazu lub formowanie strumienia danych niezgodnego z organizacją strumienia wejściowego[15][48][50]. Takie przypadki zostaną zaprezentowane w trzeciej części rozprawy.

5.1 Efektywność strumieniowego systemu wizyjnego

Alternatywnie do formuły (4.4), efektywność elementów przetwarzających współbieżnego systemu wizyjnego można również zdefiniować jako miarę ich obciążenia w czasie (5.1). Szacowanie czasów aktywności i nieaktywności należy dokonać w horyzoncie czasowym, który będzie reprezentatywny dla stopnia granulacji elementu przetwarzającego z uwzględnieniem wpływu treści danych na relacje czasów.

$$E^i = \frac{T_a^i}{T_a^i + T_m^i} \tag{5.1}$$

gdzie:

E^i – efektywność elementu i,
T_a^i – czas aktywności elementu i,
T_m^i – czas martwy elementu i.

Dla systemu składającego się z wielu elementów obliczeniowych można przyjąć całkowitą efektywność E wyznaczoną w oparciu łączne czasy aktywności i nieaktywności (5.2).

$$E = \frac{\sum T_a^i}{\sum T_a^i + \sum T_m^i} \tag{5.2}$$

gdzie:

E – efektywność złożonego systemu,
T_a^i – czas aktywności elementu i,
T_m^i – czas martwy elementu i.

W przypadku nieregularnego potoku, zarówno całkowity czas wykonania, jak i proporcje czasu martwego i czasu aktywności mogą się znacznie zmieniać w sposób zależny nie tylko od rozmiaru operandu ale również od jego treści.

Postulat ten został sformułowany w [90] dla systemów wieloprocesorowych. Równomierne obciążenie elementów przetwarzających w czasie świadczy o wysokim stopniu zrównoleglenia. Uzyskanie tego efektu wymaga jednak odpowiedniego podejścia w procesie projektowania i implementacji algorytmu na konkretnej platformie. Celem, jaki powinien sobie stawiać projektant lub programista, jest taka organizacja programu, lub architektury systemu obliczeniowego, aby każdy element funkcjonalny w każdym cyklu wykonywał użyteczne obliczenia. Założenie to z góry jest niespełnione w klasycznych procesorach, choćby z racji na wielofunkcyjność jednostki arytmetyczno-logicznej, która nie może jednocześnie

wykonywać mnożenia i dodawania. Jeśli jeden z bloków ALU (ang. *Arithmetic Logic Unit*) wykonuje swoją operację, drugi jest nieaktywny ze względu na wspólny strumień instrukcji. Bardziej rozbudowane procesory (np. procesory sygnałowe) mogą wykonywać pewne operacje współbieżnie jednak ograniczona liczba magistral wewnętrznych i rejestrów oraz natura niektórych algorytmów sprawia, że nie zawsze wszystkie elementy są wykorzystane.

5.1.1 Zwiększanie efektywności

Praca [20] pokazuje, że znaczne zwielokrotnienie liczby elementów przetwarzających w systemie programowanym SIMD wiąże się ze złożonym problemem dystrybucji strumienia instrukcji pomiędzy poszczególnymi elementami przetwarzającymi. To samo dotyczy również strumieni danych, jeśli zaimplementowany algorytm wymaga globalnego lub niedeterministycznego kontekstu.

Ominięcie tej trudności jest możliwe poprzez zastosowanie szeregu opracowanych przez autora rozprawy rozwiązań, których realizacja zostanie szczegółowo omówiona w części trzeciej rozprawy. Należy do nich zaliczyć:

- Zastąpienie programowanego systemu procesorowego dedykowanym konfigurowanym systemem obliczeniowym. Wykorzystanie technologii PLD pozwala na zachowanie elastyczności typowej dla systemów programowanych poprzez konfigurowany element obliczeniowy.
- Wykorzystanie równoległości na poziomie danych i operacji.
- Zastosowanie architektury potokowej na różnych poziomach granulacji danych, adekwatnie do zadania obliczeniowego.
- Dostosowanie sprzętowej architektury obliczeniowej do organizacji strumienia wizyjnego z kamery.
- Dystrybucja i zwielokrotnienie zasobów pamięciowych tak, aby umożliwić równoczesny dostęp do danych elementom potoku.

Układy rekonfigurowalne są obecnie często wykorzystywane w realizacji operacji wstępnego przetwarzania obrazów. Dotyczy to w szczególności realizacji kontekstowych (liniowych i nieliniowych) operacji przetwarzania obrazu (konwolucja, operacje morfologiczne, mediana) w sprzętowych architekturach potokowych [102]. Architektury te są bardzo efektywne pod warunkiem, że przestrzenny kontekst jest lokalny i niezmienny. Tor wizyjny zbudowany w oparciu o taką ideę działa współbieżnie w tempie wyznaczonym przez źródło sygnału wizyjnego. Gwarantuje to płynne przetwarzanie wszystkich pikseli bez przestojów. Powiększenie liczby operatorów w ścieżce toru nie powoduje spadku wydajności przy zachowaniu stałej częstotliwości zegara. Zwiększeniu ulega jedynie opóźnienie trans-

portowe, które określone jest sumą długości linii opóźniających w poszczególnych potokach [53].

5.1.2 Realizacja algorytmów o dużym stopniu nieregularności

Poprzez zwiększanie efektywności należy rozumieć nie tylko przyspieszenie obliczeń. We wbudowanych systemach wizyjnych wystarczy osiągnąć taki poziom przepustowości systemu, aby wszystkie ramki obrazu z kamery zostały przetworzone. Dalsze zwiększanie przepustowości jest bezcelowe i niepotrzebnie podnosi koszty systemu wizyjnego oraz zużycie zasobów i energii. Przy ustalonej przepustowości, jako cel optymalizacji można przyjąć minimalizację zasobów i zapotrzebowania na moc elektryczną.

Architektura potokowa, o bardzo drobnym ziarnie granulacji danych, nie znajduje bezpośredniego zastosowania w implementacji niektórych operacji na obrazie lub przynajmniej niektórych algorytmów wykorzystywanych w ich realizacji. Spośród często wykorzystywanych operacji przetwarzania obrazów autor rozprawy wyszczególnił trzy kategorie, dla których kryterium stanowi sposób dostępu do danych oraz zależność czasu wykonania od treści danych wejściowych:

1. przestrzenny kontekst potrzebny to wyznaczenia ostatecznej wartości pikseli nie jest ograniczony lokalnie, np.: segmentacja przez podział obszaru, segmentacja przez rozrost obszaru, segmentacja przez detekcję krawędzi [95],

2. wykorzystany algorytm jest wieloprzebiegowy i wymaga kilkukrotnej analizy każdej ramki obrazu (ilość iteracji może nawet zależeć od treści obrazu), np.: indeksacja obrazu metodą "tablicy sklejeń"[95], indeksacja obrazu metodą "pożaru prerii"[17],

3. do wyznaczenia nowej wartości piksela niezbędny jest kontekst temporalny (określony w dziedzinie czasu), np.: statystyczna analiza wielomodalna w estymacja tła [91].

Podstawowy problem, wspólny dla wyliczonych przypadków, wiąże się z potrzebą przechowania pośrednich wyników przetwarzania ramki obrazu na kolejnych etapach wykonania algorytmu i jednoczesną obsługą nadchodzących nowych danych z czujnika wizyjnego. Zakładamy przy tym, że wszystkie ramki obrazu, zgodnie z założeniem systemu strumieniowego, przetwarzane są płynnie bez pominięcia danych. Kluczowe jest więc efektywne zarządzanie ilością niezbędnej pamięci oraz sposób organizacji danych. We współczesnych systemach komputerowych wbudowanych i tych powszechnego użytku, ilość dostępnej pamięci nie stanowi ograniczenia. Nie jest jednak trudno osiągnąć górną granicę przepustowości magistrali danych podczas implementacji algorytmów wizyjnych o średnim stopniu skomplikowania. Do realizacji wyżej wymienionych algorytmów w sprzętowych systemach

współbieżnych można wykorzystać potokową architekturę również na niskim poziomie granulacji (*coarse grained*). Ciąg operacji wykonywanych na strumieniu wizyjnym dzielony jest na etapy odpowiadające poszczególnym ramkom obrazu. Pośrednie wyniki obliczeń elementów grubo-ziarnistego potoku składowane są nie w rejestrach lecz w niezależnych bankach pamięci. Dla zintegrowanych torów wizyjnych zawierających wspomniane typy algorytmów, liczba niezbędnych buforów obrazu rośnie wraz z liczbą elementarnych operacji w potoku *coarse grained*. Liczba możliwych do wykorzystania niezależnych banków pamięci limitowana jest jednak ze względu na fizyczne ograniczenia liczby końcówek układów scalonych i moc rozpraszaną podczas cykli odczytu i zapisu danych. Zasadne jest więc poszukiwanie takich architektur, sposobów reprezentacji struktur danych i ich transmisji, aby minimalizować zużycie zasobów pamięciowych i liczbę transferów danych.

5.2 Kryteria oceny przepływowego systemu wizyjnego

W ocenie działania systemów obliczeniowych, często jako kryterium wykorzystuje się czas potrzebny na przetworzenie określonego kwantu danych lub wykonanie zadania obliczeniowego. W celu analizy porównawczej różnych architektur, dokonuje się pomiaru wydajności według reguły (4.4) lub (5.1). W systemach wbudowanych istotnym kryterium może być ilość energii koniecznej do przeprowadzenia określonych obliczeń [10][20][97]. Ze względu na interakcję systemu obliczeniowego z otoczeniem wprowadza się jednak pojęcie przepustowości (5.3), które odpowiada ilości danych przetworzonych w jednostce czasu.

$$P = \frac{D}{T} \tag{5.3}$$

gdzie:

P – przepustowość,

D – liczba przetworzonych danych (bajtów, pikseli lub obrazów),

T – czas pomiaru.

Ta cecha odniesiona do systemu obliczeniowego charakteryzuje graniczną zdolność obliczeniową, najczęściej w długim horyzoncie czasowym. Ten sam współczynnik odniesiony do źródła strumienia danych określa się mianem przepływności. Ilość danych w jednostce czasu stanowi kryterium oceny medium transmisyjnego i szacowania przydatności elementu obliczeniowego w danym zastosowaniu. Zgodnie z założeniem Z1 w rozdziale 1, w systemie nie jest dopuszczalne pominięcie danych na żadnym etapie przetwarzania czy analizy. Oczekiwana przepustowość systemu obliczeniowego jest więc jednoznacznie określona przez przepływność źródła sygnału wizyjnego.

5.2.1 Opóźnienie transportowe

Zależnie od architektury i sposobu implementacji algorytmu elementy przetwarzające strumieniowego systemu wizyjnego mogą się różnić ze względu na opóźnienie transportowe, mimo iż spełniają warunek przepustowości. W systemie strumieniowym bardziej adekwatnym parametrem oceny jest raczej opóźnienie niż czas obliczeń. Nie można wskazać momentu czasowego, w którym ramka obrazu jest gotowa do przetworzenia, jeśli nie zostanie określony bufor ramki obrazu* w którym obraz jest gromadzony. Można natomiast dokładnie określić czas transmisji jednej ramki obrazu. Wynika on z parametrów czujnika wizyjnego kamery. Jedynie dla pojedynczych pikseli występuje oznaczoność co do etapu obliczeń i chwili czasowej. Dla jednego piksela, jako czas przetwarzania, można wskazać interwał pomiędzy pojawieniem się danych na wejściu elementu przetwarzającego, a uzyskaniem rezultatu na wyjściu. Wskaźnik ten również ma charakter opóźnienia transportowego i nie można go potraktować jako globalnego czasu realizacji obliczeń odniesionego do ramki obrazu, ponieważ kolejne elementy obliczeniowe w potoku równocześnie wykonują operacje na różnych pikselach.Wartość opóźnienia transportowego (najczęściej większa niż 1 cykl zegara) pomnożona przez liczbę pikseli dałaby wynik znacznie przekraczający czas trwania ramki. Stoi to w logicznej sprzeczności z zachowaniem regularności strumienia danych. Wobec powyższego, uzasadnione jest przyjęcie opóźnienia transportowego L_p jako parametru oceny działania strumieniowego toru wizyjnego. Interpretacja tego parametru jest dwojaka. Wartość ta, podana w jednostkach czasu [53] określa czas reakcji systemu. Parametr ten może być krytyczny w szczególnych zastosowaniach pomiarowych i przemysłowych [32][46][77]. Opóźnienie transportowe oznaczone bezwymiarowo określa liczbę taktów zegara, jaka jest potrzeba na propagację kwantu danych przez potok obliczeniowy. Niesie również informację o strukturze elementu przetwarzającego i pośrednio, o stopniu złożoności obliczeń oraz ilości elementów pamięciowych czyli rejestrów.

5.2.2 Opóźnienie transportowe w potoku drobnoziarnistym

Dla operacji przetwarzania wstępnego z lokalnym kontekstem przestrzennym, realizowanych w drobnoziarnistej architekturze potokowej, opóźnienie transportowe można wyznaczyć w oparciu o rozmiar kontekstu przy pomocy formuły (5.4). Szczegółową analizę czasową tego typu rozwiązań z uwzględnieniem czynników technologicznych przedstawiono w pracy [100].

$$L_d = (N\frac{(k-1)}{2} + r) \tag{5.4}$$

*Bufor ramki obrazu jest niepotrzebny, jeśli wykorzystana jest klasyczna architektura potokowa, gdzie dane "przepływają" przez element przetwarzający.

gdzie:

L_d – opóźnienie w operacjach o wysokiej ziarnistości danych (wyznaczone w cyklach),

k – pionowy rozmiar okna kontekstu - wartość nieparzysta,

N – liczba pikseli w linii obrazu,

r – opóźnienie dodatkowych rejestrów w cyklach.

Wzór (5.4) dotyczy tych operacji w których rezultatem jest obraz o topologii zgodnej z obrazem wejściowym. Czynnik r odpowiada opóźnieniu wynikającemu z zastosowania dodatkowych rejestrów synchronicznych. Mają one na celu równomierną dystrybucję czasów propagacji pomiędzy elementami logiki obliczeniowej, aby spełniony był warunek (5.5).

$$f_{max} > f_{pixel} \tag{5.5}$$

gdzie:

f_{max} – graniczna częstotliwość pracy.

Dla powszechnie wykorzystywanych rozmiarów ramki obrazu i przestrzennego kontekstu piksela, wartość r jest znacznie mniejsza od czynnika związanego z liniami opóźniającymi. Częstotliwość graniczna f_{max} wyznaczona jest przez czas propagacji najdłuższej ścieżki kombinatorycznej T_{max}.

$$f_{max} = \frac{1}{T_{max}} \tag{5.6}$$

gdzie:

T_{max} – czas propagacji najdłuższej ścieżki kombinatorycznej.

Cechą charakterystyczną opisywanej architektury jest płynność przetwarzania danych: jednemu pikselowi wejściowemu odpowiada dokładnie jeden piksel wyjściowy, a opóźnienie transportowe jest określone głównie przez rozmiary linii opóźniających. Rozmiar kontekstu determinuje liczbę komórek pamięci potrzebnych do przechowania kontekstu $N(k-1)$. Wszystkie próbki strumienia (piksele) wizyjnego poddawane są identycznym operacjom, przez co zapewniony jest pełny determinizm działania co do opóźnienia transportowego, kolejności pikseli w strumieniu i tempa przepływu danych.

5.2.3 Opóźnienie transportowe w potoku gruboziarnistym

W operacjach analizy obrazu, w których rezultatem działania elementu przetwarzającego jest nie obraz, lecz wartość (lub zbiór wartości) charakterystyczna dla całej ramki obrazu (np. suma wartości wszystkich pikseli), lub jeśli kontekst operacji jest globalny, wówczas opóźnienie transportowe można oszacować formułą (5.7). Chwilami czasowymi, które wyznaczają opóźnienie transportowe potoku gruboziarnistego są moment pojawienia się pierwszego piksela ramki na wejściu elementu obliczeniowego oraz moment wyliczenia wyniku z uwzględnieniem ostatniego piksela ramki.

$$L_g = (MN + r) \tag{5.7}$$

gdzie:

L_g – opóźnienie transportowe dla operacji o niskiej ziarnistości danych liczone w sekundach,
N – liczba cykli potrzebnych na akwizycję jednej linii obrazu (wyznaczone w cyklach),
M – liczba wierszy w ramce obrazu,
r – opóźnienie dodatkowych rejestrów.

Nieregularne operacje na strumieniu wizyjnym

W ogólnym przypadku operacje wizyjne na średnim i wysokim poziomie, wymagające globalnego kontekstu mogą mieć opóźnienie L_n uzależnione również od treści obrazu jak w formule (5.8).

$$L_n = \theta(\psi, M, N) \tag{5.8}$$

gdzie:

L_n – czas wykonania nieregularnych operacji przetwarzania obrazów (wyznaczone w cyklach),
ψ – treść obrazu,
$\theta()$ – funkcja zależności liczby cykli wykonania operacji od treści obrazu oraz rozmiarów ramki,
N – liczba kolumn w ramce obrazu,
M – liczba wierszy w ramce obrazu.

Dotyczy to również niektórych operacji, które nie zachowują reprezentacji sygnału w strumieniu wizyjnym lub wymagają zgromadzenia całej ramki obrazu

w buforze pamięciowym. Główną przyczyną braku determinizmu jest blokowanie przepływu danych lub zaburzenie kolejności przetwarzania pikseli w obrębie elementu przetwarzającego.

Opóźnienie L_n ma naturę czasu wykonania operacji a nie opóźnienia transportowego. Zgodnie jednak z założeniem Z2 w rozdziale 1 przyjęto, że również tego typu operacje będą realizowane w przepływowym torze wizyjnym, zatem wyznaczając całkowite opóźnienie transportowe systemu wizyjnego nie można pominąć czasu wykonania nieregularnych operacji przetwarzania obrazu.

Przykładowymi operacjami nieregularnego przetwarzania strumienia wizyjnego są algorytmy indeksacji i selekcji obiektów w torze wizyjnym rozpoznającym ręcznie cyfry. Obydwa przypadki zostały opisane w rozdziale 6.

5.2.4 Opóźnienie przepływowego systemu wizyjnego

Rozpatrując strumieniowy tor wizyjny jako całość można zdefiniować całkowite opóźnienie przepływowego systemu wizyjnego (5.9), które obejmuje opóźnienia poszczególnych elementów przetwarzających.

$$L_s = \sum_{i=0}^{I} L_d^i + \sum_{j=0}^{J} L_g^j + \sum_{k=0}^{K} L_n^k \tag{5.9}$$

gdzie:

- L_s – łączne opóźnienie transportowe w cyklach,
- L_d^i – opóźnienie transportowe i-tego drobnoziarnistego potoku,
- L_g^j – opóźnienie transportowe j-tego gruboziarnistego el. przetwarzającego,
- L_n^k – opóźnienie transportowe k-tego nieregularnego el. przetwarzającego.

Rozróżnienie opóźnień L_d, L_g i L_n jest istotne ponieważ L_d i L_g zależą wyłącznie od struktury elementu przetwarzającego i ewentualnie rozmiaru danych. Wartość L_n uzależniona jest nie tylko od architektury elementu przetwarzającego ale również od rozmiaru i treści danych. Dotyczy to w szczególności algorytmów o dużym stopniu nieregularności przedstawionych w sekcji 5.1.2. Zbyt duża zmienność opóźnienia transportowego L_s może być przyczyną "gubienia" danych podczas strumieniowego przetwarzania. Zjawisko to wiąże się z tym, że realizacja niektórych operacji analizy i rozpoznawania obrazów powoduje blokowanie przepływu danych w strumieniu. Kumulacja czasów przestoju podczas przetwarzania kolejnych ramek może powodować nieograniczony wzrost opóźnienia transportowego w czasie oraz nieograniczone zapotrzebowanie na bufory pamięciowe. Fakt, że źródło sygnału wizyjnego również ma swój czas nieaktywności (rozdział 6.1)

jest wobec powyższego zjawiska korzystny. Relacje ilościowe czasu blokowania i czasu martwego w strumieniu stanowią kryterium wykonalności przepływowego systemu wizyjnego.

Dyskretna funkcja $\theta()$ wymieniona w formule (5.8), która decyduje o opóźnieniu wynikającym z wykonania nieregularnych operacji na strumieniu danych jest trudna do analitycznego wyznaczenia ze względu na rozmiar dziedziny oraz stopień skomplikowania algorytmów wizyjnych. Zbiór wartości argumentów, chociaż jest skończony, nawet przy ustalonych wartościach M i N może być bardzo liczny. Moc zbioru możliwych argumentów wejściowych wynosi 2^{bMN} (gdzie b oznacza liczbę bitów reprezentujących jeden piksel). W rozdziale 6 zostanie przedstawiony sposób szacowania granicznych (maksymalnych i minimalnych) czasów wykonania nieregularnych operacji w torze wizyjnym oraz analiza ich wpływu na wykonalność strumieniowego systemu rozpoznawania znaków.

Część III

Zrównoleglanie algorytmów wizyjnych

Rozdział 6

Implementacja operacji wizyjnych o kontekście przestrzennym

W rozdziale opisano sposób zastosowania różnych rodzajów równoległości dla realizacji strumieniowego toru wizyjnego wykorzystującego kontekst przestrzenny. W pierwszej sekcji tego rozdziału przedstawiono podsystem formowania strumienia wizyjnego oraz interpolacji kolorów, jako przykład równoległej realizacji algorytmu o lokalnym kontekście przestrzennym. W kolejnych sekcjach dokonano analizy równoległości na poszczególnych etapach strumieniowej realizacji toru wizyjnego rozpoznającego ręcznie pisane cyfry. System, oprócz przestrzennego kontekstu lokalnego, wykorzystuje również algorytmy bazujące na globalnym kontekście przestrzennym.

Uzupełnienie rozdziału stanowią dodatki A, B i C.1. Dodatek A przedstawia sposób szacowania przyspieszenia procesie wieloetapowego zrównoleglania algorytmów. Przebieg przyspieszenia i efektywności architektury potokowej w zależności od parametrów potoku przedstawiono w dodatku B. Dodatek C.1 zawiera przykładowy kod źródłowy potokowego algorytmu indeksacji obrazu.

6.1 Formowanie sygnału wizyjnego

Wraz z postępem techniki wytwarzania czujników wizyjnych oraz rozwojem algorytmów przetwarzania i rozpoznawania informacji wizyjnej, systemy obrazowania działające w czasie rzeczywistym znajdują ciągle nowe zastosowania. W urządzeniach przemysłowych, medycznych, jak i coraz częściej ogólnego powszechnego użytku, stosuje się cyfrowe systemy wizyjne. Obecnie na rynku dostępne są sensory oraz kamery kolorowe o rozdzielczościach standaryzowanych przez systemy obrazowania np. popularny SXGA (ang. *Super eXtended Graphics Array*). Podnosi się szybkość akwizycji oraz rozmiary obrazu. Istnieją czujniki wizyjne umożliwiające akwizycję ponad 400-tu ramek w ciągu sekundy w formacie SXGA oraz znacznie większej liczby ramek w przypadku ustalenia mniejszych rozmiarów obrazu [19].

Analogowe formaty transmisji sygnału wizyjnego, takie jak PAL (ang. *Pha-*

se Alternating Line) lub NTSC, z racji na małą przepustowość łącza oraz poziom szumów nie umożliwiają wykorzystania pełnych możliwości zaawansowanych czujników wizyjnych. Istotną przeszkodę w użyciu analogowych standardów stanowi format transmisji zorientowany ściśle na wizualizację. Wielokrotna konwersja RGB → YCrCb → RGB oraz szumy analogowe powodują zaburzenie kolorów w obrazie [81]. Topologiczne zależności pojedynczych pikseli wynikające z przeplotu wprowadzają rozmycie ruchomych obiektów w kolejnych półobrazach, co uniemożliwia poprawne przetwarzanie i analizę obrazów obiektów szybkozmiennych. Ma to krytyczne znaczenie w przypadku niektórych zastosowań przemysłowych i specyficznych zadań badawczych [32].

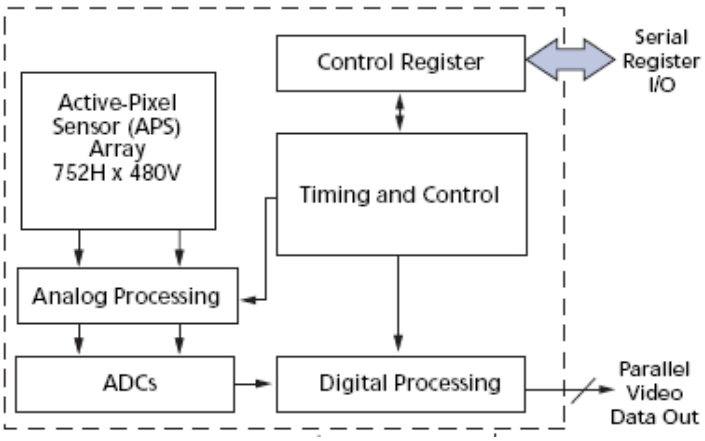
Zapewnienie dobrej jakości sygnału wizyjnego oraz możliwość dopasowania jego parametrów do warunków środowiskowych i właściwości analizowanej sceny stworzyła realną potrzebę standaryzacji komunikacji i sterowania urządzeń komputerowej analizy obrazów (ang. *Machine Vision, Computer Vision*). Obecnie w użyciu są dwie oficjalne definicje promowane przez stowarzyszenie Automated Imaging Association: *Camera Link* oraz *GigaE Vision*. Powszechnie dostępne są kamery cyfrowe wyposażone w interfejsy *IEE1394* (określany jak *FireWire*) oraz *USB 2.0*. Wszystkie z wymienionych interfejsów cechują się dobrze zdefiniowaną i sprawdzoną warstwą fizyczną łącza transmisyjnego, która umożliwiającą połączenie z typowymi platformami komputerów PC (ang. *Personal Computer*). Każdy typ łącza, oprócz styku fizycznego, zawiera również określony protokół i interfejs programistyczny do sterowania parametrami akwizycji obrazu. API (ang. *Application Programming Interface*) udostępnia procedury do konfiguracji parametrów czujnika wizyjnego (wzmocnienie, poziomy referencyjne, ustawienia elektronicznej migawki) jak i do ustalania parametrów transmisji i próbkowania sygnału (współrzędne i rozmiar obszaru próbkowania ROI, format piksela i liczba ramek przechwytywanych w sekundzie).

W specjalizowanych systemach wizyjnych spotyka się również rozwiązania wykorzystujące bezpośrednie połączenie czujnika wizyjnego i elementu obliczeniowego przetwarzającego strumieniowy sygnał wizyjny. Przykład takiego rozwiązania opracowanego przez autora rozprawy opisano w pracy [53]. Dzięki zastosowaniu cyfrowego czujnika wizyjnego MT9V022 [69], system został w całości zrealizowany w postaci cyfrowej bez stosowania dodatkowego przetwornika analogowo-cyfrowego czy framegrabber'a. Dzięki temu znacząco została ograniczona możliwość wystąpienia zakłóceń sygnału wizyjnego. Możliwa stała się też integracja elementu przetwarzającego z sensorem wizyjnym w jednym module.

6.1.1 Czasowe parametry strumienia wizyjnego

W większości czujników wizyjnych powszechnego użytku sygnał wizyjny wysyłany jest w postaci szeregowej piksel po pikselu. Współczesne kamery cyfrowe zawierające zintegrowany przetwornik analogowo-cyfrowy pracują w trybie skanowania

progresywnego i transmisji kolejnoliniowej. Zaletą takiego sposobu formowania danych, w porównaniu do systemów PAL i NTSC jest brak przeplotu. Dzięki temu ramka nie jest dzielona na dwa półobrazy a dane w ramce są spójne.



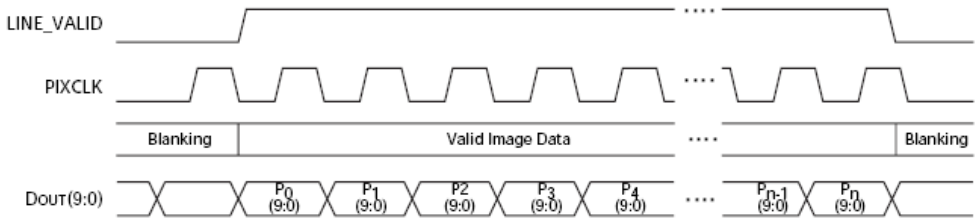
(źródło: [69])

Rysunek 6.1: Schemat blokowy czujnika wizyjnego MT9V022.

Parametry czasowe strumienia wizyjnego zostaną przedstawione na przykładzie źródła sygnału wyżej wspomnianego czujnika wizyjnego MT9V022 [69]. Przetwornik ten jest na tyle uniwersalny, iż może posłużyć jako ogólny model źródła cyfrowego sygnału wizyjnego. Poglądowy schemat czujnika przedstawiono na rysunku 6.1. Jego równoległy interfejs wyjściowy składa się z następujących sygnałów:

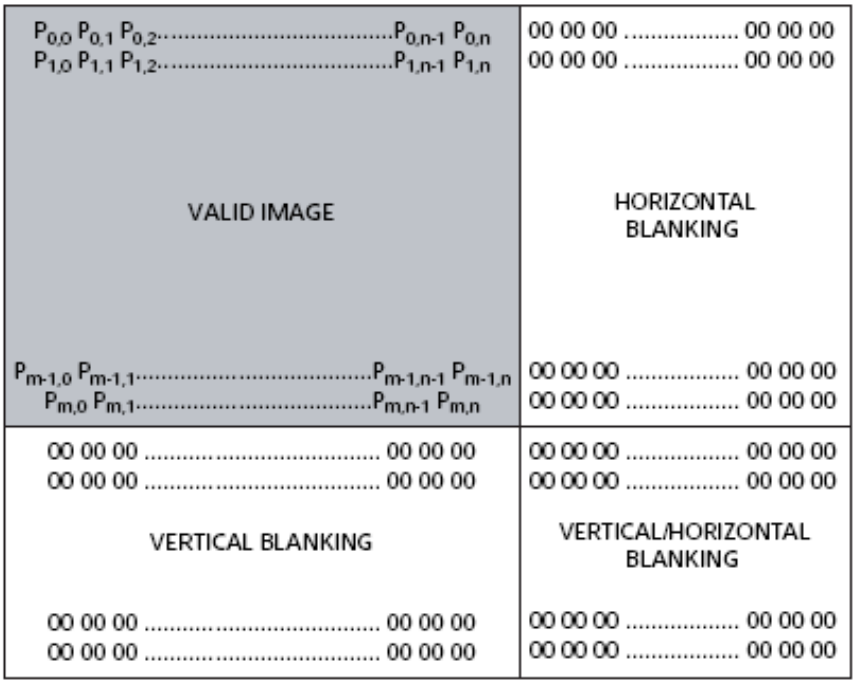
D_{OUT}	10 bitów, magistrala danych do transmisji pikseli,
$F_{RAME}V_{ALID}$	1 bit, sygnał aktywnego obszaru w ramce obrazu,
$L_{INE}V_{ALID}$	1 bit, sygnał aktywnych pikseli w linii obrazu,
PIX_CLK	1 bit, sygnał taktujący transmisję pikseli.

Najmniejszym kwantem danych w strumieniu wizyjnym jest piksel - w tym przypadku 10-bitowy. Z opadającym zboczem zegara PIX_CLK na wyjściu magistrali D_{OUT} pojawia się nowa próbka z czujnika. Piksele w wierszu rastra czujnika obrazu formują linię, której przebieg widoczny jest na rysunku 6.2. O ważności danych na magistrali wyjściowej informuje sygnał $L_{INE}V_{ALID}$. Kolejne linie składające się na ramkę obrazu niosą ważne dane przez cały okres wysokiego stanu sygnału $F_{RAME}V_{ALID}$. Kolejność transmisji pikseli podczas skanowania liniowego odpowiada rozmieszczeniu oznaczeniu komórek światłoczułych sensora jak na rysunku 6.3.



(źródło: [69])

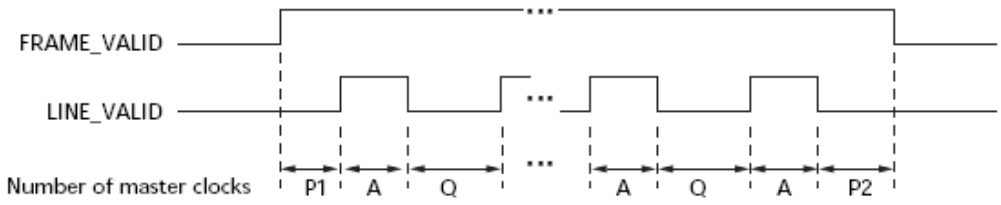
Rysunek 6.2: Transmisja pikseli obrazu.



(źródło: [69])

Rysunek 6.3: Organizacja ramki obrazu podczas skanowania liniowego.

Charakterystyczne czasy w transmisji sygnału wizyjnego przedstawiono na przebiegu ramki obrazu 6.4. Maksymalne (domyślne) wartości tych parametrów zebrano w tabeli 6.1. Czujnik ma rozdzielczość $M \times N$=752×480 zatem w jednej ramce znajduje się 360 960 pikseli. Sygnały $L_{INE}V_{ALID}$ i $F_{RAME}V_{ALID}$ są konieczne ze względu na sposób działania migawki oraz synchronizację urządzeń wyświetlających i przetwarzających strumień wizyjny.



(źródło: [69])

Rysunek 6.4: Czasowe parametry w transmisji ramki obrazu.

Parametr	Opis	Liczba cykli	Wartość	Jedn.
f	Częstotliwość PIX_CLK	1	26.60	MHz
T	Okres PIX_CLK	1	37.60	ns
A	Ważne dane w linii($= M$)	752	28.02	μs
P1	Wygaszanie na początku ramki	71	2.66	μs
P2	Wygaszanie na początku ramki	23	0.86	μs
Q	Wygaszanie poziome	94	3.52	μs
A + Q	Czas trwania linii	846	31.72	μs
BV	Wygaszanie pionowe	38,074	1.43	ms
TF	Całkowity czas ramki	444,154	16.66	ms

Tabela 6.1: Czasowe parametry czujnika wizyjnego

Dla potrzeb dalszych analiz prowadzonych w rozprawie autor rozprawy zdefiniował zapas czasowy strumienia wizyjnego V (6.1) określonego jako względny czas martwy w strumieniu.

$$V = \frac{TF - MN}{TF} = 0.1873 \tag{6.1}$$

gdzie:

V – współczynnik wypełnienia strumienia,
M – liczba aktywnych wierszy w ramce obrazu $M = 480$,
N – liczba aktywnych kolumn w ramce obrazu $N = 752$,
TF – czas akwizycji ramki liczony w cyklach zegara, $TF = 444154$.

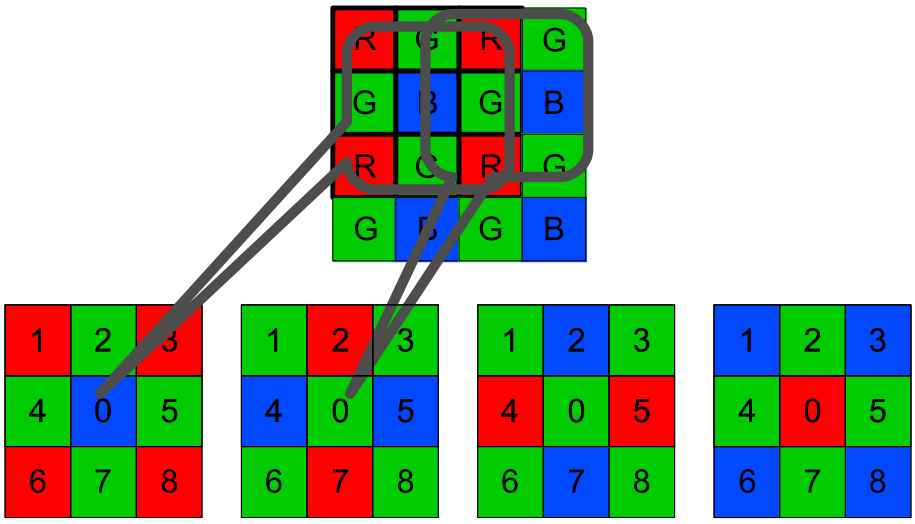
Aktywny obszar sensora, z którego odczytywane są piksele do utworzenia strumienia wizyjnego, jest ustawiany poprzez rejestry czujnika. Iloraz V może zatem przyjmować większe wartości, jeśli aktywny obszar sensora zostanie zmniejszony przy zachowaniu czasów wygaszania.

Parametr V posłuży w podrozdziale 6.5 do oceny wykonalności przepływowej realizacji algorytmów wizyjnych o opóźnieniu zależnym od treści strumienia

wizyjnego. W odniesieniu do elementu przetwarzającego strumień wizyjny, zdefiniowano tam komplementarny parametr (6.23) w postaci rozstępu czasu wykonania v. Spełnienie nierówności $v \leqslant V$ jest warunkiem koniecznym wykonalności algorytmu w systemie strumieniowym.

6.1.2 Barwny czujnik wizyjny

W większości powszechnie stosowanych kamer kolorowych (podobnie jak w kolorwym czujniku MT9V022) światłoczuła matryca sensora zorganizowana jest w postaci mozaiki filtrów: CFA* (ang. *Color Filter Array*). Jedną z pierwszych [12] i nadal stosowanych topologii filtrów jest matryca Bayer'a, której organizacja przedstawiona jest na rysunku 6.5. Filtry promieniowania widzialnego przepuszczające barwę czerwoną R, zieloną G i niebieską B rozłożone są naprzemiennie. Najliczniejsze są komórki z filtrem zielonym. Dzięki temu czułość sensora odpowiada wrażliwości ludzkiego oka, które wykazuje największą wrażliwość w paśmie promieniowania widzialnego odpowiadającego zieleni.

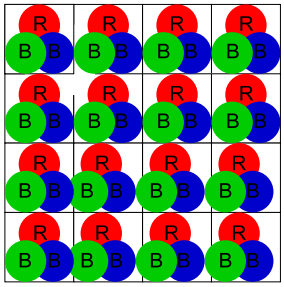


Rysunek 6.5: Konfiguracje komórek filtrów w kolorowym czujniku wizyjnym.

Podstawowym formatem danych w przetwarzaniu obrazów barwnych są składowe przestrzeni barw RGB (ang. *Red Green Blue*) z których wyprowadza się pochodne: HSV (ang. *Hue Saturation Value*), YCrCb, reprezentację w poziomach szarości (ang. *gray level*) i inne [88]. Format obrazu w przestrzeni RGB zakłada, że każdemu pikselowi w rastrze, jak na rysunku 6.6, odpowiadają trzy składowe

*Wyjątek stanowią kamery wizyjne z potrójnym sensorem 3xCCD (ang. *Charge Coupled Device*), 3xCMOS (ang. *Complementary Metal–Oxide–Semiconductor*), czujniki wysokiej rozdzielczości CCD-HR, CCD-SR oraz Foveon-X3.

czerwona R, zielona G i niebieska B. Korzystając z czujnika wizyjnego wyposażonego w filtr mozaikowy, konieczne jest zatem uzupełnienie strumienia wizyjnego z czujnika o brakujące składowe, zależnie od lokalizacji piksela w matrycy.



Rysunek 6.6: Raster obrazu w formacie RGB

6.1.3 Interpolacja pikseli

Uzupełnianie brakujących danych realizowane jest poprzez interpolację w oparciu o otoczenie danego piksela. Jedną z prostszych metod dających akceptowalne rezultaty jest interpolacja liniowa w obszarze kontekstu 3×3 poprzez zastosowanie jednego z czterech równań (6.2) dla poszczególnych orientacji przedstawionych na rysunku 6.5.

$$P_0^{RGB} = \begin{cases} (\frac{P_1+P_3+P_6+P_8}{4} & \frac{P_2+P_4+P_5+P_7}{4} & P_0) & \text{kon. } a) \\ (\frac{P_2+P_7}{2} & P_0 & \frac{P_4+P_5}{2}) & \text{kon. } b) \\ (\frac{P_4+P_5}{2} & P_0 & \frac{P_2+P_7}{2}) & \text{kon. } c) \\ (P_0 & \frac{P_2+P_4+P_5+P_7}{4} & \frac{P_1+P_3+P_6+P_8}{4}) & \text{kon. } d) \end{cases} \tag{6.2}$$

gdzie:

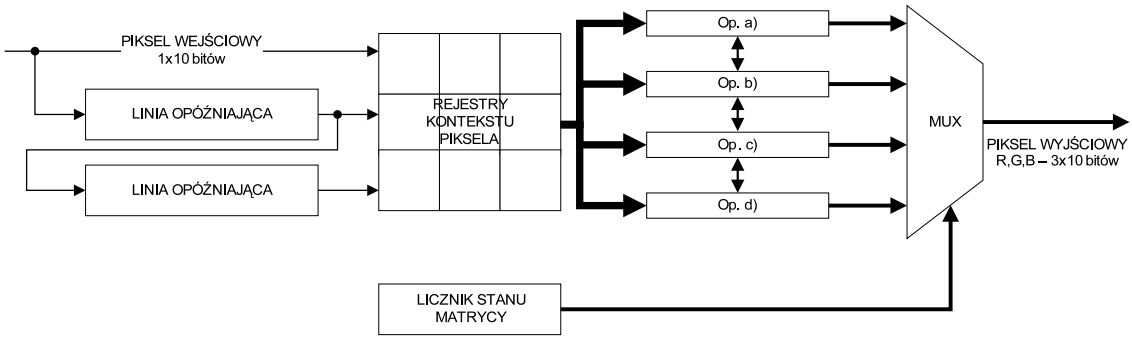
P_i – wartość sygnału komórek matrycy wg opisu z rysunku 6.5,
$P_0^{R,G,B}$ – wynik interpolacji: piksel RGB.

Zestaw równań (6.2) odnosi się bezpośrednio to topologii matrycy Bayer'a przedstawionej na rysunku 6.5. Kolejne piksele, transmitowane naprzemiennie z czujnika odpowiadają konfiguracjom a) b) i c) d) w kolejnych liniach obrazu, zależnie od koordynat interpolowanego piksela.

Organizacja pamięci

W implementacji interpolacji zastosowano potokową architekturę przedstawioną na rysunku 6.7. Wykorzystuje ona zestaw pamięci w postaci linii opóźniających.

Dzięki temu, cały kontekst danego piksela dostępny jest równocześnie w każdym takcie zegara. Organizacja danych w liniach opóźniających oraz kolejność ich przetwarzania jest narzucona przez organizację pikseli w strumieniu danych z czujnika wizyjnego. Dzięki temu nie ma konieczności adresowania danych w pamięci, a odczyt następuje poprzez propagację danych z wejścia na wyjście, w tempie transmisji pikseli z czujnika. Dla omawianego przypadku rozmiar potrzebnej pamięci równy jest dwóm długościom wiersza obrazu. Niezbędne jest również 6 dodatkowych rejestrów które umożliwiają w każdym cyklu dostęp do aktualnie przetwarzanego piksela i ośmiu jego sąsiadów. Taka architektura potokowego przetwarzania jest powszechnie wykorzystywana w sprzętowej realizacji algorytmów wizyjnych o deterministycznym kontekście lokalnym [101] takich jak splot, operacje morfologiczne czy mediana [26].



Rysunek 6.7: Strumieniowa interpolacja barwnego sygnału wizyjnego.

Zrównoleglanie obliczeń

Korzyść wynikająca z zastosowania przepływowego modelu przetwarzania polega na:

- równoczesnym dostępie do całego kontekstu piksela (równoległość DLP), przez co możliwe stało się wykonanie interpolacji (równoległość OLP),
- ograniczeniu rozmiaru zużytych zasobów pamięciowych do rozmiaru równego dwóm długościom wiersza obrazu.

Zrównoleglenie obliczeń polega na współbieżnej ewaluacji wszystkich wyrażeń równania (6.2). Ostateczne ustalenie wartości próbki danych w strumieniu wizyjnym polega na wyborze jednej z czterech wartości w oparciu o stan licznika koordynat. Wykorzystano w ten sposób równoległość operacji, która w tej postaci jest przyczyną niskiego wykorzystania zasobów. Wyniki działań logiki realizującej poszczególne wyrażenia równania (6.2) są ważne w jednym z czterech cykli, co świadczy o niskim wykorzystaniu zasobów w czasie.

Liczba operacji sumowania (wynikająca ze wzoru 6.2) wykonywanych podczas interpolacji jest zależna od bieżącej konfiguracji pikseli i wynosi:

- 12 dla konfiguracji a) i d) łącznie,
- 4 dla konfiguracji b) i c) łączenie,

W każdym takcie należy więc wykonać 16 operacji dodawania. Analiza formuły interpolacji (6.2) prowadzi jednak do wniosku, że poszczególne składowe są symetryczne. W różnych konfiguracjach te same wyrażenia służą do wyznaczania różnych składowych R, G i B.

Wykorzystanie tego faktu prowadzi do rozwiązania, które polega na współdzieleniu wyrażeń cząstkowych pomiędzy równaniami wyznaczającymi piksel w formacie RGB. Składowa R w konfiguracji a) jest wyznaczana identycznie jak składowa B w konfiguracji d) itd. Uwzględnienie tej redundancji i zastosowanie mechanizmu multipleksacji wyników cząstkowych pozwala zredukować liczbę operacji do:

- 6 dla konfiguracji a) i d) łącznie,
- 2 dla konfiguracji b) i c) łączenie,

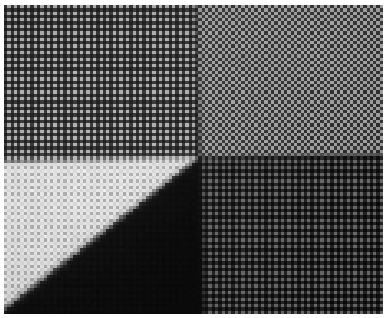
Widać więc, że współdzielenie zasobów obliczeniowych spowodowało dwukrotny spadek ich liczby oraz dwukrotny wzrost stopnia ich wykorzystania. Zjawisko to ma bezpośredni wpływ na stopień zużycia zasobów podczas implementacji w strukturze rekonfigurowalnej, a pośrednio na zużycie energii.

Przedstawiony system cechuje się deterministycznym działaniem, niezależnym od treści obrazu. Opóźnienie grupowe podsystemu interpolacji wynosi $L_s = N+4$ zgodnie z równaniem (5.4). Rezultat interpolacji wraz z pośrednimi wynikami przetwarzania obrazu na poszczególnych etapach przedstawiono na rysunku 6.8.

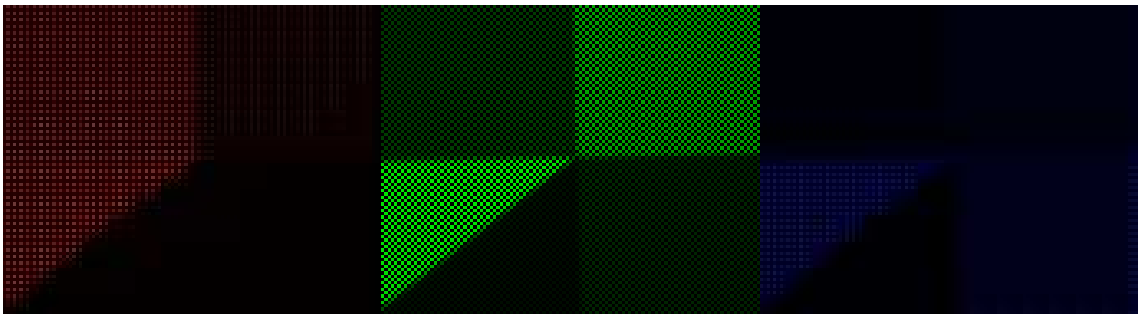
Tabela 6.2: Efektywność wykorzystania zasobów obliczeniowych w interpolacji

	Zrównoleglenie		Współdzielenie	
Operacja	Liczba instancji	Efektywność E	Liczba instancji	Efektywność E
$\frac{P_1+P_3+P_6+P_8}{4}$	2	0.25	1	0.5
$\frac{P_2+P_7}{2}$	4	0.25	1	1.0
$\frac{P_4+P_5}{2}$	4	0.25	1	1.0

Rozpatrując efektywność równoległej implementacji wyrażeń równania (6.2), zgodnie z propozycją szacowania efektywności (5.1) można stwierdzić, że zastosowanie współdzielenia cząstkowych wyników, pozwoliło na czterokrotne zwiększenie efektywności wykorzystania dwóch operatorów sumujących (patrz tabela 6.2) przy czterokrotnym zmniejszeniu liczby instancji.



(a) naturalny obraz uzyskany z czujnika



(b) piksele czerwone (c) piksele zielone (d) piksele niebieskie

(e) obraz RGB po interpolacji

Rysunek 6.8: Etapy interpolacji obrazu testowego

6.2 System wizyjny rozpoznający ręcznie pisane cyfry

Niniejszy podrozdział opisuje wybrane zagadnienia związane z systemem rozpoznawania ręcznie pisanych cyfr w formularzach. Omawiany system wizyjny do rozpoznawania wykorzystuje sztuczną sieć neuronową [70]. Sztuczne sieci neuronowe są stosowane w zadaniach rozpoznawania wzorców i klasyfikacji. Uniwersalność tego narzędzia pozwala na analizę różnorodnych sygnałów. O ile sama implementacja programowa sieci neuronowej nie przysparza obecnie znaczących problemów, to sposób doboru architektury sieci i jej parametrów w znaczący sposób wpływa nie tylko na proces uczenia sieci, ale również na szybkość działania [94]. Ze względu na ilość obliczeń realizowanych podczas symulacji i uczenia sieci,

krytyczne jest stworzenie odpowiedniej reprezentacji danych wejściowych. Należy dążyć do minimalizowania rozmiaru wzorców poddawanych rozpoznawaniu. W wykorzystanym modelu [70] przyjęto, że wektor wejściowy ma postać znormalizowanej matrycy znaku. Wiąże się z tym konieczność wykonania szeregu operacji wstępnych na analizowanym obrazie. Są to:

(a) przetwarzanie wstępne,

(b) segmentacja,

(c) indeksacja obiektów,

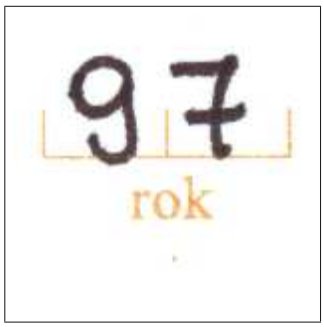
(d) pomiar cech i selekcja obiektów,

(e) skalowanie obiektów do rozmiaru matrycy znaku,

(f) rozpoznawanie znaków.

Operacje wykonywane podczas przetwarzania wstępnego ściśle zależą od jakości obrazu. Szczególnie istotne jest usunięcie zakłóceń, które mogą zniekształcić znaki. Zbinaryzowany obraz poddany jest procesowi segmentacji i indeksacji. Wyodrębnione zostają wówczas poszczególne obiekty oraz określone zostaje ich położenie. Spośród wszystkich zidentyfikowanych obiektów, na podstawie analizy ich cech, odrzucane są te, które nie spełniają kryteriów rozmiaru i współczynników kształtu. Ponieważ obiekty mogą mieć różne gabaryty, a sieć została skonstruowana i wytrenowana dla klasyfikacji znaków o zbliżonych rozmiarach, konieczne jest ich skalowanie jak to opisano w pracy [48]. Zaprojektowana sieć neuronowa jako sygnał wejściowy pobiera matrycę punktów binarnych (wartości 0,1) o rozmiarze 20×20 punktów. Realizacja czynności (a), (b) i (c) lokalizację poetykietowanych obiektów. Operacja (d), na podstawie analizy, pozwala wykluczyć przynależność obiektu do poszukiwanych klas, minimalizując tym samym liczbę wywołań modułu skalującego (e) i modułu sieci neuronowej (f).

Opisany system rozpoznawania znaków wykorzystuje sieć neuronową, z zestawem wag przygotowanym do rozpoznawania ręcznie pisanych cyfr, zapisanych jako matryce znaków o rozmiarach 20×20 binarnych pikseli, przy czym znaki ciągu uczącego zorientowane były pionowo. Fakt ten stanowi pewne ograniczenie w praktycznym zastosowaniu omawianej implementacji systemu: formularz musi mieć ściśle określoną orientację względem kamery, aby znaki znajdowały się zawsze w orientacji pionowej. Nie umniejsza to jednak w żaden sposób ogólności opisywanego rozwiązania. Poprzez odpowiedni dobór rozmiarów sieci neuronowej i zastosowanie ciągu uczącego zawierającego znaki obrócone względem środka matrycy, można przygotować system do rozpoznawania znaków niezależnie od ich orientacji. Zagadnienia doboru struktury sztucznej sieci neuronowej i jej rozmiaru oraz trenowania sieci wykracza jednak poza zakres niniejszej pracy.

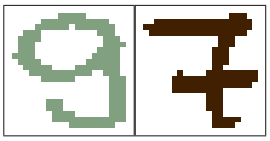


(a) fragment obrazu źródłowego

(b) wynik segmentacji

(c) wynik indeksacji

(d) wynik selekcji



(e) wynik skalowania

9 7

(f) wynik rozpoznania

Rysunek 6.9: Etapy przetwarzania w strumieniowym systemie rozpoznawania znaków.

Rzeczywiste rezultaty działania strumieniowego toru wizyjnego przedstawiono na rysunku 6.9. Widoczny jest wycinek ankiety o rozmiarach 150×150 pikseli oraz odpowiadające mu obrazy wynikowe. Znaki znormalizowane do rozmiaru 20×20 pikseli na obrazie 6.9e), zostały pokazane bez zachowania skali w stosunku do obrazu źródłowego. Paleta kolorów, użyta do prezentacji etykiet w obrazach po indeksacji, została przygotowana wyłącznie w celach prezentacyjnych. W rzeczywistości, etykiety kodowane są jako liczy całkowite.

6.3 Algorytmy przetwarzania obrazów

Z operacji wstępnego przetwarzania i analizy obrazów, opisanych w podrozdziale 6.2, do implementacji sprzętowej wybrano segmentację (b), indeksację (c), selekcję obiektów (d) i skalowanie (e). Realizacja tych algorytmów wymaga zastosowania

różnorodnych metod. Założono, że dane źródłowe maja postać obrazu barwnego o wydzielonych składowych RGB - czerwonej, zielonej i niebieskiej. W wyniku segmentacji uzyskano cyfrowy obraz binarny.

6.3.1 Przetwarzanie wstępne

Podczas eksperymentów przeprowadzonych w warunkach laboratoryjnych z dobrym oświetleniem okazało się, że jakość obrazu pozyskanego poprzez akwizycję obrazu wybranych formularzy z ręcznie pisanymi znakami jest zazwyczaj bardzo dobra. Dotyczy to zwłaszcza odpowiednio przygotowanych arkuszy, gdzie barwa danych wprowadzanych do formularza różni się znacząco od koloru tabel i znaczników zapobiegając ich "zlewaniu się" w procesie segmentacji. Pominięto więc etap przetwarzania wstępnego a drobne artefakty pojawiające się podczas segmentacji usuwane są poprawnie poprzez wstępną klasyfikację obiektów w operacji (d) na podstawie cech (6.7). Wyniki prac autora rozprawy nad automatycznym doborem parametrów akwizycji obrazu przy zmiennych warunkach oświetleniowych przedstawione są w pracy [49].

6.3.2 Segmentacja

W omawianym rozwiązaniu do segmentacji wykorzystano wieloprogową binaryzację sygnału barwnego (6.3).

$$\begin{aligned} P^S = \quad & th_{lo}^R < P^R < th_{hi}^R \quad \wedge \\ & th_{lo}^G < P^G < th_{hi}^G \quad \wedge \\ & th_{lo}^B < P^B < th_{hi}^B \end{aligned} \tag{6.3}$$

gdzie:

P^S – piksel obrazu po segmentacji,
P^R, P^G, P^B – składowe czerwona, zielona i niebieska obrazu wejściowego,
th_{lo}^R, th_{hi}^R – próg wysoki i niski dla składowej czerwonej,
th_{lo}^G, th_{hi}^G – próg wysoki i niski dla składowej zielonej,
th_{lo}^B, th_{hi}^B – próg wysoki i niski dla składowej niebieskiej.

Metoda pozwala na łatwe wydzielenie pikseli potencjalnie należących do znaku z uwzględnieniem jego koloru. Ten sposób segmentacji sprawdza się szczególnie, kiedy barwa kwestionariusza i znaku różnią się, a warunki oświetleniowe są stabilne. Na etapie konfiguracji systemu należy zwrócić szczególną uwagę na dobór progów z uwzględnieniem barwy oświetlenia. W specyficznych warunkach wystarczy dobranie progu jednej składowej P^R, P^G lub P^B przy ustaleniu pozostałych na wartości graniczne.

6.3.3 Indeksacja

Celem indeksacji jest identyfikacja obiektów (ang. *connected components*). W tym procesie piksele należące do danego obiektu otrzymują identyczną etykietę, przy czym etykiety są unikatowe dla poszczególnych obiektów. Spośród szeregu metod indeksacji obrazu wybrano metodę liniowego przeglądania obrazu i sklejeń opisaną w [60][95]. W fachowej literaturze angielskojęzycznej algorytm ten określany jest jako *connected component labeling*.

W niniejszej rozprawie wykorzystano wersję algorytmu z 8-elementowym sąsiedztwem punktu (ang. 8-*connected* - rysunek 6.10a), chociaż w literaturze opisywane są również przypadki wykorzystania uproszczonego kontekstu z 4-elementowym sąsiedztwem (ang. 4-*connected* - rysunek 6.10b) [85][92]. Suzuki podaje w pracy [92], że programowa implementacja algorytmu indeksacji

a) sąsiedztwo 8-pikselowe b) sąsiedztwo 4-pikselowe

Rysunek 6.10: Kontekst przestrzenny w indeksacji.

dla otoczenia 4-pikselowego daje nieznacznie krótsze czasy działania algorytmu, w stosunku do wersji z pełnym kontekstem 8-pikselowym. Czas wykonania zależy liniowo od liczby pikseli obiektów w obrazie. Wyniki badań autora rozprawy przedstawione w rozdziale 6.5 pokazują, że czas indeksacji przy pomocy dedykowanej architektury sprzętowej w niewielkim stopniu zależy od treści obrazu. Czas sprzętowej realizacji liniowo zależy od rozmiarów obrazu, w niewielkim stopniu zaś od liczby obiektów i ich kształtu. Ze względu na równoległość zastosowaną w obliczeniach, kształt analizowanego otoczenia wcale nie wpływa na czas indeksacji. Bardziej korzystny jest zatem pełny kontekst 3×3 przedstawiony na rysunku 6.10a. W rozumieniu kontekstu 8-elementowego dwa piksele są ze sobą w relacji sąsiedztwa bezpośredniego, jeśli ich koordynaty spełniają równanie (6.4). Zakładając przechodniość relacji sąsiedztwa można stwierdzić, że do danego obiektu należą również te piksele, które sąsiadują ze sobą pośrednio.

$$max(|x_1 - x_2|, |y_1 - y_2|) = 1 \tag{6.4}$$

gdzie:

(x_1, y_1) – współrzędne pierwszego piksela,

(x_2, y_2) – współrzędne drugiego piksela.

Algorytm indeksacji metodą liniowego przeglądania i sklejeń dzieli się na trzy fazy:

- indeksacja wstępna - identyfikowane są fragmenty obiektów, równocześnie zapamiętywana jest informacja o ich połączeniach,
- porządkowanie tablicy sklejeń - usuwane są tymczasowe etykiety,
- przekodowanie wstępnie przydzielonych etykiet przy pomocy przekształcenia LUT (ang. *Look Up Table*).

Operacje numeryczne, wykonywane w algorytmie indeksacji sprowadzają się do porównywania wartości całkowitoliczbowych, pozostałe polegają zaś na modyfikacji zawartości tablic obrazu i tablicy sklejeń.

Zaletą przedstawionego algorytmu indeksacji, w odróżnieniu od innych rekurencyjnych i iteracyjnych [9][15][45], jest to, iż wymaga jedynie dwóch przebiegów po tablicy obrazu w kierunku zgodnym z akwizycją strumienia pikseli w ramce obrazu. W wyniku tej operacji każdy, różny od tła punkt obrazu, otrzymuje etykietę unikatową dla obiektu, do którego należy.

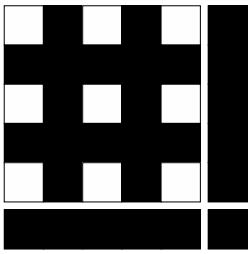
W celu analizy czasu realizacji algorytmu indeksacji dokonano szacowania maksymalnej liczby obiektów w obrazie, zależnie od jego rozmiaru (tj. szerokości M i wysokości N). Zgodnie z definicją sąsiedztwa (6.4) maksymalna liczba etykiet jaka może zostać przydzielona w obrazie określona jest wzorem (6.5).

$$G_{max} = \frac{MN}{4} \tag{6.5}$$

gdzie:

G_{max} – graniczna liczba obiektów zidentyfikowanych w obrazie,
N – szerokość obrazu,
M – wysokość obrazu.

Zależność ta wynika z analizy maksymalnej liczby obiektów jaka może się znaleźć w obrazie binarnym. Graniczny przypadek pokazany jest na sztucznie spreparowanym wzorcu obrazu przedstawionym na rysunku 6.11. Co drugi piksel w co drugiej linii jest pikselem obiektu i ma wartość 1. Łączna liczba pikseli we wzorcu równa jest wartości opisanej równaniem (6.5). Pozostałe piksele stanowią tło, czyli przyjmują wartość 0. Mając na uwadze rysunek 6.11 i równanie (6.4) należy zauważyć, że żaden z pikseli o wartości 1 nie ma sąsiadów w swoim otoczeniu. Każdy z nich stanowi oddzielny obiekt. "Zapalenie" któregokolwiek z pikseli tła powoduje połączenie dwóch przyległych obiektów w jeden większy, a całkowita liczba obiektów spadnie o 1. Nie jest zatem możliwe zbudowanie wzorca obrazu, który miałby większą liczbę obiektów niż G_{max}. Patrząc na rysunek 6.11 można



1	0	1	0	1	0
0	0	0	0	0	0
1	0	1	0	1	0
0	0	0	0	0	0
1	0	1	0	1	0
0	0	0	0	0	0

Rysunek 6.11: Wzorzec do szacowania liczby etykiet w obrazie binarnym.

jednak stwierdzić ze ta sama liczba obiektów o rozmiarze jednego piksela stanowi treść obrazu o rozmiarze 6×6 jak i wydzielonego fragmentu o rozmiarze 5×5. W ostatnim przypadku teoretyczna wartość wyznaczona ze wzoru (6.5) wynosi 6.25 podczas gdy we wzorcu obserwujemy 9 obiektów. Większa liczba możliwych obiektów jest spowodowana nieparzystym rozmiarem obrazu (liczbą kolumn lub liczbą wierszy). Czynnik ten został uwzględniony w równaniu (6.6).

$$G_{max} = ceil\left(\frac{M}{2}\right) ceil\left(\frac{N}{2}\right) \tag{6.6}$$

gdzie:

G_{max} – graniczna liczba obiektów zidentyfikowanych w obrazie,

N – szerokość obrazu,

M – wysokość obrazu,

$ceil(\alpha)$ – funkcja wyznaczająca najmniejszą liczbę całkowitą nie mniejszą od α.

Równanie (6.5) można jednak uznać za obowiązujące ponieważ w praktyce nie wykorzystuje się obrazów o nieparzystych rozmiarach. W implementacjach programowych dąży się nawet do tego aby wymiary obrazu były bez reszty podzielne przez 4.

Należy podkreślić, iż liczba obiektów w rzeczywistych obrazach jest najczęściej znacznie mniejsza niż G_{max}. Obecność obiektów o rozmiarze jednego lub kilku pikseli w obrazie świadczy o złych warunkach akwizycji obrazu lub o niewłaściwie przeprowadzonej filtracji zakłóceń podczas przetwarzania wstępnego. Aby obraz był użyteczny do analizy kształtów i rozpoznawania, zidentyfikowane obiekty powinny składać się z wielu pikseli. Wzrost powierzchni obiektów powoduje jednoczesny spadek ich liczby.

6.3.4 Pomiar cech i selekcja obiektów

Dla każdego obiektu wyznaczonego podczas indeksacji określony zostaje zestaw podstawowych cech w postaci prostokąta opisującego oraz pola powierzchni obiek-

tu. Parametry te pozwalają określić położenie obiektu oraz współczynniki (6.7):

$$\begin{aligned} h &= x_h - x_l \\ w &= y_h - y_l \\ h_{wr} &= \frac{h}{w} \\ a_r &= hw \\ a_{rf} &= \frac{a_f}{a_r} \end{aligned} \tag{6.7}$$

gdzie:

(x_l, y_l, x_h, y_h) – koordynaty prostokąta opisującego,
h – wysokość obiektu,
w – szerokość obiektu,
h_{wr} – proporcje boków prostokąta opisującego,
a_r – pole powierzchni prostokąta opisującego,
a_f – pole powierzchni obiektu,
a_{rf} – współczynnik wypełnienia prostokąta opisującego.

Progowanie powyższych współczynników umożliwia odrzucenie części obiektów. Dotyczy to zwłaszcza artefaktów o małej powierzchni, fragmentów tabel formularzy i innych, które ze względu na rozmiary i kształt nie mogą zostać zakwalifikowane jako poprawne znaki. Drugi etap selekcji odbywa się półautomatycznie w oparciu o współczynnik wysokości względnej (6.8) wyliczany dla każdego kolejnego obrazu oddzielnie.

$$h_r = \frac{h}{h_{max}} \tag{6.8}$$

gdzie:

h – wysokość obiektu,
h_r – wysokość względna obiektu $h_r \in (0; 1\rangle$,
h_{max} – wysokość najwyższego obiektu w danej ramce obrazu.

Obiekty, których wysokość względna h_r jest poniżej określonego progu są również usuwane.

Definicje niektórych współczynników (6.7)(6.8) pokazują, iż mogą one przyjmować wartości ułamkowe. W implementacji nie wystarczy już zatem arytmetyka całkowitoliczbowa. Numeryczne aspekty doboru formatu obliczeniowego oraz precyzji zostaną omówione w rozdziale 6.4.1.

6.3.5 Skalowanie obiektów

Obszary obrazu wyselekcjonowane do rozpoznawania muszą być zgodne z rozmiarem wektora wejściowego sieci. Technikę normalizacji stosowali również autorzy pracy [59] do rozpoznawania twarzy przy pomocy sieci neuronowej zaimplementowanej na zrównoleglonym procesorze sygnałowym o architekturze VLIW. W niniejszej pracy skalowanie obiektów przeznaczonych do rozpoznawania wykonywane jest poprzez próbkowanie wysegmentowanego obiektu w obszarze ograniczonym przez jego prostokąt opisujący. Spośród wielu metod zmiany rozdzielczości przestrzennej wybrano metodę przybliżeń do najbliższego sąsiada. Powszechnie wykorzystywane sposoby interpolacji liniowej czy kubicznej nie są w tym zastosowaniu uzasadnione, zarówno ze względu na naturę obrazu wejściowego jak i nakłady obliczeniowe. Zaobserwowano, że obszary ręcznie pisanego znaku cechują się dużą zwartością tzn. piksele sąsiadujące ze sobą (oprócz pikseli brzegowych) mają zbliżoną barwę i jasność, co zapobiega pojawianiu dziur w wysegmentowanym obrazie potencjalnego znaku. Nie istnieje więc ryzyko wystąpienia efektu aliasingu, który w pewnych przypadkach mógłby doprowadzić do istotnych przekłamań w normalizacji obiektów i ostatecznie uniemożliwiłby poprawne rozpoznanie. Metoda daje dobre rezultaty zarówno w przypadku decymacji (tj. gdy rozmiar obiektu jest większy od wymaganego 20×20) jak i podczas interpolacji, gdy rozmiar obiektu jest mniejszy od docelowego rozmiaru matrycy. Na obszar ograniczony prostokątem (x_l, y_l, x_h, y_h) nałożona zostaje siatka, której węzły wyznaczają miejsca próbkowania. Matryca znaku prezentowanego sieci neuronowej ma rozmiar $K \times K$ punktów, gdzie $K = 20$. Zatem współrzędne (x, y) obrazu wejściowego należy kwantować ze współczynnikami (6.9):

$$\Delta x = \frac{w}{K} \quad , \quad \Delta y = \frac{h}{K} \tag{6.9}$$

gdzie:

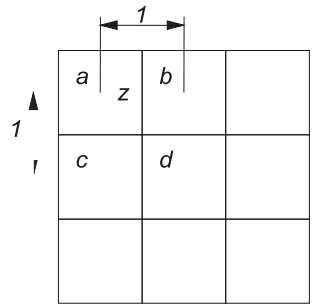
K – rozmiar matrycy znaku, $K = const$,
Δx – współczynnik skalowania szerokości,
Δy – współczynnik skalowania wysokości.

Zakładając, że rozmiary piksela w rastrze obrazu źródłowego wynoszą $(1, 1)$, wartość elementu matrycy $P^M(z)$ znaku określona jest przez najbliższy element obrazu źródłowego P^S według zależności (6.10).

$$P^M(z) = P^S(min(\| z-a \|, \| z-b \|, \| z-c \|, \| z-d \|)) \tag{6.10}$$

gdzie:

$z \in \mathbb{R}$ – współrzędne próbkowania obiektu wyznaczone w oparciu o kwanty koordynat Δx i Δy,
$P^S(z)$ – piksel obiektu w rastrze obrazu źródłowego,
$P^M(z)$ – piksel matrycy znaku,
$a, b, c, d \in \mathbb{N}$ – koordynaty najbliższych sąsiadów punktu z w rastrze ramki obrazu, patrz rys. 6.12,
$\| \cdot \|$ – norma "taksówkowa".



Rysunek 6.12: Orientacja punktów próbkowania względem rastra obrazu.

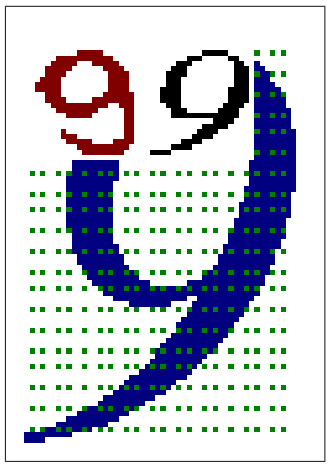
Dla poprawnego wyznaczenia koordynat nie ma konieczności precyzyjnego wyznaczania euklidesowych odległości (jako pierwiastka sumy kwadratów). Wystarczy określić, która z czterech odległości jest minimalna. Zależność (6.10) można więc uprościć do postaci (6.11) z zachowaniem zgodności rezultatów. Dzięki temu zastąpiono wyznaczanie odległości jako pierwiastka różnicy kwadratów poprzez zaokrąglenie współrzędnych stałoprzecinkowych do liczb całkowitych.

$$P^M(z) = P^S(round(z)) \tag{6.11}$$

gdzie:

$round(\alpha)$ – funkcja zaokrąglająca argument α do liczby całkowitej.

Obraz testowy z naniesioną siatką punktów próbkowania przedstawiono na rysunku 6.13. Kolorem niebieskim oznaczono obiekt zakwalifikowany do rozpoznawania. Kolor zielony oznacza punkty, w których próbkowany jest obszar znaku natomiast, kolorem czarnym oznaczono wynik próbkowania obszaru obiektu w oznaczonych punktach. W lewym górnym rogu obszaru znaku (rysunek 6.13) naniesiono wynik rozpoznania w postaci przyporządkowanego wzorca ręcznie pisanego znaku.



Rysunek 6.13: Normalizacja obiektu przez próbkowanie w obszarze ROI.

6.3.6 Rozpoznawanie znaków

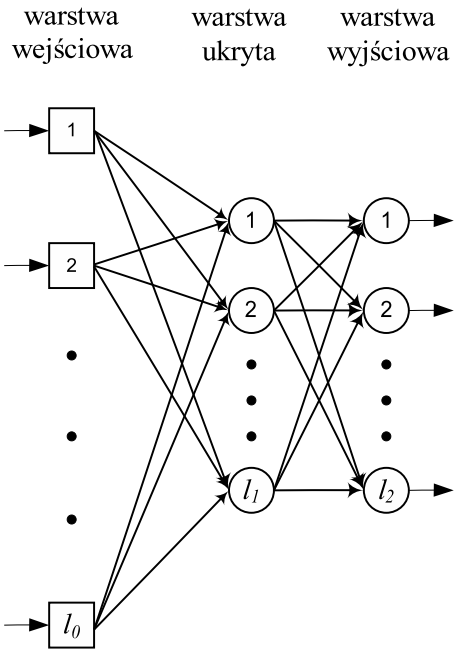
Jednym z narzędzi wykorzystywanych w rozpoznawaniu wzorców - w tym również obrazów - są sztuczne sieci neuronowe (ang. *Artificial Neural Networks*). Struktury te naśladują sposób pozyskiwania wiedzy przez ludzi i nie nakładają przy tym znaczących ograniczeń na postać i rozmiar sygnału wejściowego. Dzięki temu możliwe jest dopasowanie rozmiaru i architektury sztucznej sieci neuronowej do skali problemu i oczekiwanej precyzji rozpoznań.

Przygotowanie sieci neuronowej do działania wymaga (oprócz doboru architektury i testowania) etapu uczenia, w którym ustalane są wagi poszczególnych połączeń synaptycznych. Proces ten polega najczęściej na prezentowaniu sztucznej sieci neuronowej wzorców obiektów w postaci rastrowej lub zestawu cech charakteryzujących zawartość obrazu. W poszczególnych iteracjach wagi połączeń synaptycznych modyfikowane są tak, aby na wyjściu uzyskać wynik zbliżony do oczekiwanego. Zależnie od metody uczenia (z nauczycielem, bez nauczyciela) zadawane są również oczekiwane rezultaty lub oczekiwana liczba klas rozpoznawanych wzorców. W literaturze [94] opisano wiele różnych rodzajów sztucznych sieci neuronowych. Można wyróżnić szereg kryteriów klasyfikacji i przez to rodzajów sztucznych sieci neuronowych:

- liniowość: liniowe, nieliniowe,
- propagacji sygnału: jednokierunkowe, ze sprzężeniem zwrotnym,
- reprezentacja sygnału: sygnał ciągły, sygnał pulsujący,
- sposób uczenia sieci: nadzorowany, nienadzorowany,
- pamięć: statyczne, dynamiczne.

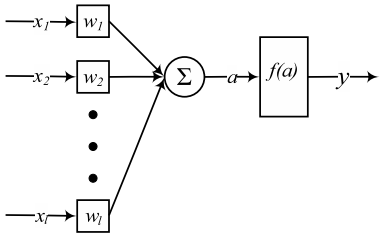
W niniejszej rozprawie do rozpoznawania ręcznie pisanych cyfr została wykorzystana sztuczna sieć neuronowa zaczerpnięta z pracy [70]. Jest to dwuwarstwowa nieliniowa jednokierunkowa sieć wytrenowana metodą wstecznej propagacji błędu (ang. *back propagation*). Ponieważ w opisywanych pracach wykorzystano sieć z gotowym zestawem wag, proces uczenia nie został w rozprawie opisany. Czytelnik znajdzie szereg pozycji literaturowych gdzie szczegółowo opisano tę i inne techniki trenowania sztucznych sieci neuronowych [94].

Strukturę sztucznej sieci neuronowej wykorzystanej w rozprawie przedstawiono na rysunku 6.14. Ten rodzaj sieci określa się w literaturze mianem perceptronu wielowarstwowego [83]. Składa się on z 400 wejść, 10-ciu neuronów w warstwie ukrytej oraz 10-ciu neuronów w warstwie wyjściowej. Liczba wejść wynika z rozmiaru wektora wejściowego jakim jest znormalizowana matryca znaku o liczbie pikseli $l_0 = K^2 = 400$, zaś liczba wyjściowych neuronów $l_2 = 10$ odpowiada liczbie rozpoznawanych znaków tj. 10 cyfrom arabskim: $0, 1, 2, 3, 4, 5, 7, 8, 9$.



Rysunek 6.14: Struktura sieci neuronowej do rozpoznawania znaków.

Każdy z neuronów (rys. 6.15) składa się z sumatora gromadzącego pobudzenie z wejścia lub neuronów poprzedzającej warstwy. Sumowanie odbywa się według formuły (6.12) z uwzględnieniem wagi każdego połączenia.



Rysunek 6.15: Model sztucznego neuronu.

$$a = W_0 + \sum_{k=1}^{l} W_j u_j \tag{6.12}$$

gdzie:

j – numer połączenia synaptycznego,
W_0 – składnik stały,
W_j – waga j-tego połączenia synaptycznego,
u_j – sygnał j-tego neuronu (lub j-tego wejścia w warstwie wejściowej),
a – wartość pobudzenia neuronu.

Sygnał wyjściowy $u = \xi(a)$ neuronu uzyskiwany jest poprzez obliczenie funkcji przejścia $\xi(a)$ dla całkowitego pobudzenia a. Istotnym elementem, który decyduje o szczególnych właściwościach sztucznych sieci neuronowych jest nieliniowa funkcja przejścia. Opisywane w literaturze modele [94] wykorzystują często funkcję sigmoidalną (6.13), która również jest częścią zastosowanego w tej pracy modelu sztucznej sieci neuronowej. Funkcja ta jest ciągła i różniczkowalna w całej dziedzinie argumentu a jej wartości mieszczą się w przedziale $u \in (0; 1)$

$$u = \xi(a) = \frac{1}{1 + e^{-\beta a}} \tag{6.13}$$

gdzie:

a – pobudzenie,
$\xi(a)$ – funkcja przejścia,
u – wartość funkcji przejścia dla pobudzenia a.

Rezultatem rozpoznania jest numer neuronu warstwy wyjściowej i_{max}, którego sygnał wyjściowy jest największy (6.14). Przypisanie znaków do poszczególnych neuronów wyjściowych zostało dokonane na etapie trenowania perceptronu.

Dla uproszczenia można przyjąć, że indeks każdego z dziesięciu wyjść jest jednocześnie rozpoznaną cyfrą.

$$u_{i_{max}} = \max_{i=0}^{9}(u_i) \tag{6.14}$$

gdzie:

i_{max} – indeks neuronu warstwy wyjściowej o największym sygnale,
$u_{i_{max}}$ – największy sygnał wyjściowy,
i – indeks neuronu warstwy wyjściowej, $i \in \{0, 1, ..., 9\}$,
u_i – sygnał i-tego neuronu warstwy wyjściowej.

Nie mniej istotna od samego rozpoznania jest ocena jakości wyniku wytworzonego przez sztuczną sieć neuronową dla zaprezentowanego znaku. Przyjęto, że wysoka wartość sygnału na danym wyjściu sieci (przy jednocześnie niskiej wartości wszystkich pozostałych) świadczy o dużej pewności rozpoznania znaku przyporządkowanego neuronowi wyjściowemu. Obecność wysokiego sygnału na więcej niż jednym wyjściu sygnalizuje małą wiarygodność rozpoznania. Do oceny poprawności wykorzystano wskaźnik wiarygodności (6.15), który wraz z indeksem zwycięskiego neuronu wyjściowego stanowi ostateczny wynik rozpoznania. Wysoka wartość δ świadczy o dużej pewności rozpoznania.

$$\delta = \min_{i=0, i\neq i_{max}}^{9}(|u_{i_{max}} - u_i|) \tag{6.15}$$

gdzie:

δ – wskaźnik pewności rozpoznania,
i_{max} – indeks neuronu warstwy wyjściowej o największym sygnale,
$u_{i_{max}}$ – największy sygnał wyjściowy,
i – indeks neuronu warstwy wyjściowej, $i \in \{0, 1, ..., 9\}$,
u_i – sygnał i-tego neuronu warstwy wyjściowej.

6.4 Strumieniowa realizacja toru wizyjnego do rozpoznawania znaków

Programowy, sekwencyjny model systemu rozpoznawania znaków został zaimplementowany w języku ANSI-C dla platformy PC. Program ten stanowi model referencyjny do weryfikacji strumieniowej realizacji systemu na kolejnych etapach

zrównoleglania. Weryfikacja jest szczególnie istotna na etapie doboru formatu liczbowego i precyzji obliczeń, która w układach konfigurowanych może być dopasowana do konkretnego zadania obliczeniowego. W dalszej części rozdziału opisano sposób doboru reprezentacji liczb ułamkowych jak i przekształcenia, które pozwolą na uniknięcie kosztownej, ze względu na zasoby i czas realizacji, operacji dzielenia.

Proces przenoszenia modelu programowego na platformę sprzętową wiąże się z dopasowaniem do składni języka HandelC, w którym został zaimplementowany system rozpoznawania znaków. Poprawne działanie systemu wymaga również dodania szeregu elementów typowych dla platformy sprzętowej, które nie wymagają ingerencji programisty w przypadku systemu opartego o platformę PC. Chodzi tutaj głównie o zarządzanie zegarem systemowym, organizację pamięci danych oraz interfejsy komunikacji systemu wizyjnego z otoczeniem.

6.4.1 Aspekty obliczeniowe w sprzętowej analizie obrazu

Jak wynika ze wzoru (6.7), obliczenia podczas ekstrakcji cech wykonywane są na liczbach wymiernych. Ze względu na precyzję obliczeń, jak i zużycie zasobów istotny jest dobór formatu obliczeniowego. Programowy model funkcjonalny wykorzystuje zmiennoprzecinkowy typ danych podwójnej precyzji `double`. Realizacja sprzętowa takiej arytmetyki jest zadaniem trudnym z punktu widzenia procesu projektowego, zużycia zasobów i czasu realizacji obliczeń. W omawianym zastosowaniu wystarczające są jednak obliczenia stałoprzecinkowe. Zastosowanie formatu stałoprzecinkowego nie musi oznaczać spadku precyzji obliczeń, o ile ilość bitów na część całkowitą i ułamkową zostanie odpowiednio dobrana. Kolejna trudność wiąże się z realizacją dzielenia. Operacje dzielenia - zarówno liczb całkowitych jak i zmiennoprzecinkowych - w powszechnie stosowanych mikroprocesorach należą do najbardziej czasochłonnych instrukcji. Co więcej, niektóre z nich wcale nie wykonują dzielenia (mikrokontrolery, mikroprocesory RISC ang. *Reduced Instruction Set Computer*), lub obliczają jedynie aproksymację, pozostawiając możliwość programowego doprecyzowania wyniku zależnie od wymaganej precyzji (niektóre procesory DSP np. Blackfin). Struktura omawianego algorytmu umożliwia jednak zastąpienie dzielenia operacją mnożenia. Ponieważ współczynniki a_{rf}, h_{wr} i h_r wykorzystywane są do porównania z wartościami progowymi, wystarczy przekształcić nierówność (6.16), aby otrzymać równoważny funkcjonalnie warunek (6.17).

$$h_{wr} > t_{rh} \tag{6.16}$$

$$h > wt_{rh} \tag{6.17}$$

gdzie:

t_{rh} – próg współczynnika proporcji boków prostokąta opisującego,

h – wysokość obiektu,

w – szerokość obiektu

h_{wr} – proporcje boków prostokąta opisującego.

Analogicznie, w przypadku obliczania współrzędnych elementów matrycy P, skorzystano z faktu, iż K jest wartością stałą; zatem zależności (6.9) można zastąpić przez (6.18).

$$\Delta x = wF \quad , \quad \Delta y = hF \tag{6.18}$$

gdzie:

$F = K^{-1} = const$ – stały współczynnik kwantyzacji koordynat matrycy znaku.

Precyzję obliczeń w arytmetyce stałoprzecinkowej wyznacza liczba bitów przeznaczona na reprezentację części ułamkowej: fb. Tym samym, najmniejszy kwant liczbowy, jaki może zostać bez błędu zapamiętany opisany jest równaniem (6.19), a bezwzględny błąd reprezentacji stanowi jego połowę (6.20).

$$q_{fb} = 2^{-fb} \tag{6.19}$$

$$E = 2^{-(fb+1)} \tag{6.20}$$

gdzie:

q_{fb} – waga najmniej znaczącego bitu części ułamkowej w systemie stałoprzecinkowym,

E – błąd bezwzględny w reprezentacji stałoprzecinkowej o fb bitach przeznaczonych na część ułamkową.

Ze względu na sposób wyznaczania elementów matrycy znaku, do określenia dopuszczalnego względnego błędu reprezentacji należy wziąć pod uwagę wartość F:

$$E_r = \frac{E}{F} \tag{6.21}$$

gdzie:

Er – błąd względny w reprezentacji stałoprzecinkowej odniesiony do wartości F.

Przyjęto, że na część ułamkową w procesie selekcji obiektów zostanie przeznaczone $fb = 16$ bitów. Wartość błędu względnego E_r wynosi w przybliżeniu $1.5 \cdot 10^{-4}$.

6.4.2 Aspekty obliczeniowe w implementacji sieci neuronowej

Dobór precyzji wag dla sztucznej sieci neuronowej został przeprowadzony w oparciu o iteracyjną procedurę symulacji modelu sieci. Dla zakresu precyzji reprezentacji wag przeprowadzono ewaluację sztucznej sieci neuronowej. W trakcie symulacji stwierdzono, że już 10-cio bitowa reprezentacja wag pozwala uzyskać wyniki identyczne z rezultatami działania zmiennoprzecinkowego modelu sieci neuronowej. Mała liczba bitów przeznaczona na reprezentację wag jest istotna ponieważ ma liniowy wpływ na zużycie banków pamięci BlockRAM (pamięć blokowa wewnątrz układu FPGA), które stanowią ograniczone zasoby wnętrza układu rekonfigurowalnych.

Sumatory ukrytej warstwy gromadzą sygnały z 400-tu wejść przemnożonych przez odpowiadające wagi. Liczba bitów potrzebna do poprawnej akumulacji pobudzenia jest wyznaczona formułą (6.22). Dla warstwy ukrytej najmniejsza dopuszczalna rozdzielczość wynosi 19 bitów, a dla sumatorów warstwy ostatniej 14 bitów.

$$2^{nb} \geqslant 2^{nw}(l_i + 1) \tag{6.22}$$

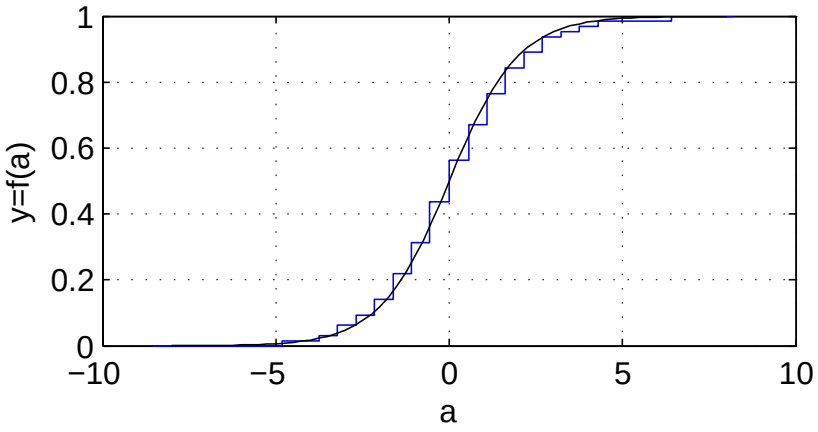
gdzie:

nw – rozdzielczość bitowa wag warstwy l,

nb – rozdzielczość bitowa sumatora warstwy l,

l_i – liczba wejść pobudzających neuronu w warstwie i.

W celu sprzętowej realizacji neuronu dokonano aproksymacji funkcji sigmoidalnej (6.13) przebiegiem schodkowym jak na rysunku 6.16. W literaturze moż-



Rysunek 6.16: Funkcja przejścia oraz jej aproksymacja.

na spotkać szereg opracowań, które skupiają się na możliwie precyzyjnym odwzorowaniu przebiegu funkcji przejścia dla pełnego zakresu wartości pobudzenia [104][63]. W sprzętowej realizacji najbardziej atrakcyjną wydaje się być metoda przybliżania funkcji odcinkami prostych o różnym nachyleniu, przy czym liczba odcinków decyduje o dokładności przybliżenia [104].

W prezentowanej sprzętowej implementacji sztucznej sieci neuronowej wykorzystano przybliżenie sigmoidy poziomymi odcinkami w 19-tu przedziałach pobudzenia. Za wykorzystaniem tak prostej metody przemawia mała ilość zużytych zasobów obliczeniowych przy zachowaniu dobrych wyników rozpoznań. Dodatkowym argumentem jest właściwość, którą sztuczne sieci neuronowe dzielą z biologicznymi sieciami neuronowymi, a mianowicie to, że system neuronowy jest w stanie działać pomimo zakłócenia sygnału wejściowego lub zaburzenia wag. Może również wykazywać odporność na uszkodzenia poprzez całkowitą utratę niektórych połączeń synaptycznych (ang. *fault tolerant computing*) [36]. Wartości funkcji przejścia mieszczą się w przedziale $\langle 0; 1\rangle$ i są reprezentowane w 6-bitowym formacie stałoprzecinkowym bez znaku.

6.4.3 Zrównoleglenie w rozpoznawaniu znaków

W tabeli 6.3 zebrano wszystkie operacje wykonywane w omawianym torze wizyjnym rozpoznającym ręcznie pisane znaki. Przeprowadzono analizę rozmiaru danych przetwarzanych na poszczególnych etapach, aby oszacować wpływ treści obrazu na graniczne (maksymalne i minimalne) czasy wykonania obliczeń. Jako miarę zmienności czasu wykonania poszczególnych operacji v (6.23) wybrano rozstęp[61][89] znormalizowany do maksymalnego czasu wykonania.

Tabela 6.3: Rozmiary danych w systemie rozpoznawania znaków.

Etap	Rodzaj operacji na obrazie	Liczba iteracji	Rozmiar danych w iteracji	Względny rozmiar danych*	Rozstęp względny v
1	segmentacja	1	MN	1.0000	0.000
2	indeksacja	2	$MN + G'$	1.2500	0.200
3	pomiar cech	1	MN	1.0000	0.000
4	selekcja	1	G	0.2500	1.000
5	skalowanie	R	K^2	1.0000	1.000
6	rozpoznawanie	R	K^2	1.0000	1.000
7	ocena wyniku	2R	10	0.0250	1.000
	Wynik zbiorczy			6.8000	0.412

* Względny rozmiar danych liczony jest jako iloraz rzeczywistego rozmiaru i rozmiaru ramki obraz MN z uwzględnieniem maksymalnych wartości R i G wg. przedziałów (6.25)(6.26).

$$v = \frac{C_{max} - C_{min}}{C_{max}} \tag{6.23}$$

gdzie:

v – względny rozstęp czasu obliczeń dla danej operacji,
C_{max} – maksymalna liczba cykli dla danej operacji,
C_{min} – minimalna liczba cykli dla danej operacji.

Względny rozstęp v podany w tabeli 6.3 został wyznaczony analitycznie w oparciu o fakt, że liczba cykli jest wprost proporcjonalna do rozmiaru danych, co jest zgodne ze wstępnym założeniem prac opisanych w niniejszej rozprawie.

Bezpośrednim czynnikiem decydującym o rozmiarach danych na poszczególnych etapach jest liczba obiektów zidentyfikowanych w trakcie pierwszego kroku indeksacji[†] G', rzeczywista liczba obiektów G a następnie liczba R obiektów zaklasyfikowanych wstępnie jako znaki do rozpoznawania.

$$G_{max} \geqslant G' \geqslant G \geqslant R \tag{6.24}$$

gdzie:

G_{max} – graniczna liczba obiektów: $G_{max} = 0.25MN$, por. wzór (6.5),
G' – liczba obiektów zidentyfikowanych w pierwszym etapie indeksacji metodą tablicy sklejeń,
G – rzeczywista liczba obiektów zidentyfikowanych w obrazie,
R – rzeczywista liczba obiektów zaklasyfikowanych do rozpoznawania.

W prezentowanej analizie przyjęto, że liczba obiektów mieści się w przedziale (6.25), choć w praktyce jest znacznie mniejsza od G_{max}. Szacowanie liczby R znaków w rzeczywistym obrazie oparto o dwa proste założenia:

- wszystkie znaki w rzeczywistym obrazie mają rozmiar matrycy znaku,
- obszary znaków nie nakładają się na siebie czyli nie mają pikseli wspólnych.

[†]Liczba obiektów zidentyfikowanych w pierwszym etapie indeksacji metodą tablicy sklejeń może być większa od rzeczywistej liczby obiektów G. Na wartość G' ma wpływ kształt obiektów oraz ich orientacja w obrazie. Dla poprawnych warunków akwizycji wartość G' jest bardzo zbliżona do G więc przyjęto, że $G' = G$. Dowód uzasadniający to uproszczenie zostanie pominięty aby zachować klarowność rozumowania przedstawionego w rozprawie.

Zakładając, że cały obszar ramki obrazu pokryty jest znakami o tym samym rozmiarze, wartość liczby R znaków mieści się w przedziale (6.26).

$$G \in \langle 0, \frac{MN}{4} \rangle \tag{6.25}$$

$$R \in \langle 0, \frac{MN}{K^2} \rangle \tag{6.26}$$

gdzie:

N – szerokość obrazu,

M – wysokość obrazu,

G – rzeczywista liczba obiektów zidentyfikowanych w obrazie,

R – liczba obiektów zaklasyfikowanych do rozpoznawania.

Należy się spodziewać, że w poprawnie dobranych warunkach akwizycji obrazu liczba R obiektów będzie raczej zbliżona do liczby G rozpoznawanych znaków niż do maksymalnej liczby etykiet G_{max}. Zależy to jednak od jakości obrazu oraz ustalonego kryterium klasyfikacji znaku do rozpoznawania.

Podczas projektowania sprzętowej architektury omawianego systemu wizyjnego wykonano szczegółową analizę algorytmów toru wizyjnego pod kątem możliwości zrównoleglenia i przetwarzania strumieniowego na poszczególnych etapach. Zaobserwowano, że realizacja niektórych operacji możliwa jest niejako "w locie" podczas realizacji przesłań międzyrejestrowych. Jedynym kosztem jaki się z nimi wiąże jest niewielkie opóźnienie transportowe i zużyta logika obliczeniowa. Dotyczy to zwłaszcza wieloprogowej binaryzacji (6.3), dzięki której uzyskuje się binarny obraz obiektów. Podobnie pomiar cech obiektów wykonywany jest równocześnie z ostatnim etapem algorytmu indeksacji podczas operacji LUT, która przypisuje ostateczne wartości etykiet poszczególnym pikselom obiektów. Mimo iż jest to operacja bezkontekstowa, to wyznaczenie prostokątów opisujących i pola powierzchni dla poszczególnych obiektów jest możliwe dzięki unikatowym etykietom, które jednoznacznie opisują przynależność danego piksela do określonego obiektu. W tym przypadku, oprócz dodatkowej logiki kombinatorycznej, konieczne jest wykonanie odczytów i zapisów cech obiektów z i do pamięci. Podczas przekodowania LUT nie jest stwierdzane, w którym cyklu analiza danego obiektu została zakończona. Selekcja obiektów wykonywana jest więc dopiero po przekodowaniu całej ramki obrazu. Obiekty, które nie spełniają wcześniej ustalonych kryteriów zostają odrzucone, a pozostałe są zapamiętane w pamięci jako kandydaci przeznaczeni do dalszego rozpoznania przez sieć neuronową. Selekcja realizowana jest w pętli. Liczba iteracji równa jest liczbie poetykietowanych obiektów. Wartość ta, ograniczona z góry poprzez G_{max}, w praktyce jest kilka rzędów wielkości mniejsza od rozmiarów obrazu wejściowego.

Kolejny obieg poetykietowanego obrazu określony jest liczbą wyselekcjonowanych obiektów oraz formatem wektora wejściowego sieci neuronowej. W przeciwieństwie do segmentacji, indeksacji wstępnej i operacji LUT (które przetwarzają wszystkie piksele w buforze ramki, niezależnie od ich lokalizacji, treści i kontekstu) skalowanie wykonywane jest jedynie dla obszarów ograniczonych prostokątem opisującym obiektu, który został zakwalifikowany do dalszego przetwarzania. Dzięki zastosowanej metodzie skalowania obiektów (opisanej w sekcji 6.3.5), dla każdego obiektu odczytywane są z pamięci wyłącznie te piksele, które wchodzą w skład matrycy znaku. Ponieważ odczytywane piksele stanowią bezpośrednio treść matrycy znaku, nie są one ponownie zapisywane do pamięci, lecz są wysyłane do współbieżnie działającego modułu sztucznej sieci neuronowej. Pierwsza warstwa sieci neuronowej przetwarza kolejne piksele skalowanego znaku bez wprowadzania dodatkowych opóźnień w procesie normalizacji. Skalowanie kolejnego znaku wstrzymane jest do momentu ewaluacji wszystkich pozostałych neuronów oraz oszacowania wiarygodności wyniku.

Zrównoleglenie operacji bezkontekstowych

Kolejnym (po interpolacji) przykładem wykorzystania współbieżności na poziomie danych i operacji jest w omawianym systemie segmentacja opisana formułą (6.3). Sekwencyjne wykonanie przebiegające według schematu przedstawionego w tabeli 6.4 składa się z 11-tu operacji i zajmuje 11 cykli. Symbol ↓ reprezentuje

Tabela 6.4: Operacje elementarne w segmentacji.

operacja	pseudokod
1	`f_1 = th_lo_R < P_R`
2	`f_2 = th_hi_R > P_R`
3	`f_3 = th_lo_G < P_G`
4	`f_4 = th_hi_R > P_G`
5	`f_5 = th_lo_B < P_B`
6	`f_6 = th_hi_B > P_B`
↓	
7	`f_7 = f_1 and f_2`
8	`f_8 = f_3 and f_4`
9	`f_9 = f_5 and f_6`
↓	
10	`f_10 = b_7 and b_8`
↓	
11	`P_S = b_9 and b_10`

relację następstwa grup operacji w czasie. Założono przy tym, że hipotetyczna sekwencyjna jednostka przetwarzająca jest w stanie wykonać jednocześnie jedną

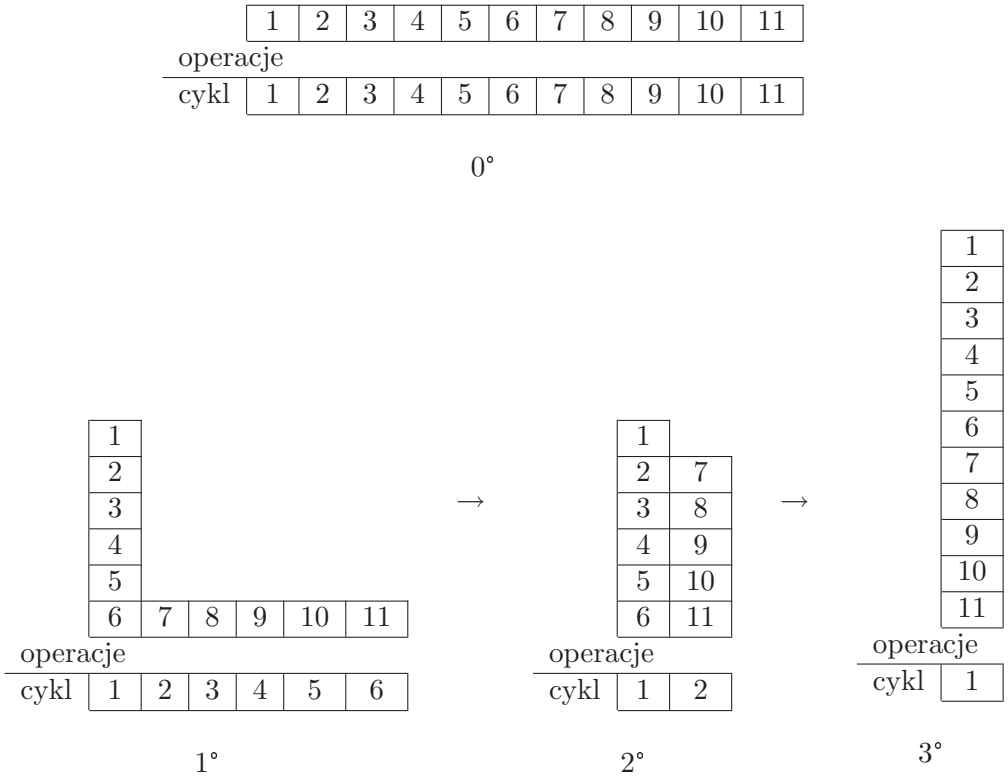
dwuargumentową operację porównania <, > lub jedną operację iloczynu logicznego `and`.

Dla uproszczenia analizy zrównoleglenia segmentacji przedstawionego na rysunku 6.17 przyjęto, że zrównoleglenie realizowane jest etapami na których można obserwować poprawę przyspieszenia 1°, 2°, 3° w stosunku do sekwencyjnego modelu 0°.

1° Zrównoleglenie operacji porównania < >. Progowanie każdej ze składowych może się odbywać współbieżnie ponieważ cząstkowe wyniki binaryzacji nie są od siebie zależne, a składowe barwne są dostępne jednocześnie.

2° Zrównoleglenie koniunkcji `and`. Kolejnym etapem zrównoleglenia jest zastąpienie 5-ciu wywołań dwuargumentowej operacji `and` pięcioma sprzętowymi operatorami działającymi równolegle.

3° Pełne zrównoleglenie. Ostatni etap przyspieszania polega na równoczesnym wykonaniu wszystkich operacji w jednym cyklu.



Rysunek 6.17: Etapy zrównoleglenia segmentacji.

W podrozdziale 4.3, oprócz formuły na doświadczalne wyznaczenie przyspieszenia, podano również prawo Amdahl'a (4.3) i Gustafson'a (4.5), które pozwalają określić przyspieszenie analitycznie: pierwsze w oparciu o ilość zrównoleglonych operacji, drugie w oparciu o proporcje czasu wykonania zrównoleglonych operacji. W tabeli 6.5 zebrano parametry potrzebne do wyznaczenia przyspieszenia oraz rezultaty wyliczeń. Otrzymane rezultaty pokazują że wszystkie metody dają identyczne wyniki.

Tabela 6.5: Ocena zrównoleglenia segmentacji.

	0°	0°- 1°	1°- 2°	2°- 3°
prawo Amdahl'a (4.3)				
n	1	6	5	2
s	1	5/11	1/6	0
p	0	6/11	5/6	1
S_a	1	11/6	6/2	2
prawo Gustafson'a (4.5)				
s'	1	5/6	1/2	0
p'	0	1/6	1/2	1
Sg	1	11/6	6/2	2
przyspieszenie całkowite (4.2)				
T_1	11	11	11	11
T_n	11	6	2	1
S_m	**1.0**	**1.83**	**5.50**	**11.0**
efektywność (4.4)				
E		0.305	0.500	1.0
opóźnienie transportowe				
L	11	6	2	1
przepustowość				
P_c	1/11	1/6	1/2	1

W kolejnych etapach zrównoleglenia rosną następujące wskaźniki: przyspieszenie S, przepustowość P i efektywność E. Równocześnie maleje opóźnienie transportowe L. Jego wartość istotna jest jednak dopiero po osiągnięciu docelowej przepustowość $P_C = 1$, kiedy to spełniony zostaje warunek wykonalności strumieniowej realizacji algorytmu. W pozostałych przypadkach opóźnienie transportowe należy interpretować jako czas obliczeń. Widać więc, że zarówno przyspieszenie, efektywność jak i przepustowość są dobrymi kandydatami na wskaźniki pozwalające szacować stopień zrównoleglenia.

Łączne przyspieszenie równe jest iloczynowi przyspieszeń uzyskanych na poszczególnych etapach (6.27).

$$S^{\Gamma} = \prod_{\gamma=1}^{\Gamma} S^{\gamma} \tag{6.27}$$

gdzie:

S^{Γ} – całkowite przyspieszenie,
S^{γ} – przyspieszenie na γ-tym etapie zrównoleglenia,
γ – etap zrównoleglenia,
Γ – liczba etapów.

Dla przypadku 2° efektywność wynosi zaledwie 1/2 pomimo wykorzystania 11-tu współbieżnych jednostek przetwarzających. Wynik ten jest identyczny po zastosowaniu obydwu formuł na efektywność (4.4) i (5.1), co potwierdza ich równoważność. Tak niska wartość stanowi podstawę do przypuszczeń, że jest możliwe dalsze zwiększenie przyspieszenia poprzez reorganizację sposobu komunikacji zrównoleglonych jednostek obliczeniowych. Ograniczenie komunikacyjne wynika w tym przypadku z sekwencji operacji, która wymusza wykonanie porównania przed iloczynem logicznym (patrz tabela 6.4). Przypuszczenie to znajduje potwierdzenie wyniku uzyskanym w etapie 3°.

Możliwość wyboru pomiędzy opcją 3° a 2° zależy jednak od relacji pomiędzy długością ścieżki kombinatorycznej, a szybkością transmisji pikseli obrazu (6.28).

$$f_g > f_{pixel} \tag{6.28}$$

gdzie:

f_g – graniczna częstotliwość pracy,
f_{pixel} – częstotliwość transmisji pikseli w strumieniu wizyjnym.

Ze względu na dysproporcje w liczbie poziomów logiki pomiędzy komparatorami a bramkami logicznymi można przyjąć, że zmniejszenie częstotliwości granicznej f_g w przypadku 3° jest nieznaczne. Kosztem, jaki wiąże się z zastosowanym tutaj zrównolegleniem jest dodatkowa liczba zużytych zasobów obliczeniowych: komparatorów i bramek `and` (patrz tabela 6.6). Wartość ta jest jednak kompensowana przez zmniejszenie o porównywaną wartość (tj. 10) liczby elementów pamięciowych przechowujących wyniki pośrednie obliczeń. Łączna ilość zasobów przed i po zrównolegleniu utrzymuje się na porównywalnym poziomie.

Należy zwrócić uwagę, że zaprezentowany sekwencyjny model wykonania procesu segmentacji jest bardzo uproszczony. Nie uwzględnia on zasobów potrzebnych do pobrania i dekodowania strumienia instrukcji sterujących zasobami obliczeniowymi i składowaniem wyników w pamięci danych. Sprawia to, że wyniki dla modelu sekwencyjnego są zawyżone jeśli chodzi o przepustowość (tabela 6.5) oraz zaniżone odnośnie zużytych zasobów (tabela 6.6).

Tabela 6.6: Zużycie zasobów w zrównoleglaniu segmentacji.

	0°	1°	2°	3°
komparatory	1	6	6	6
bramki `and`	1	1	5	5
rejestry	10	10	6	0
$\sum$	12	17	17	11

Zrównoleglenie operacji o nieregularnym kontekście przestrzennym

Z przeprowadzonych analiz, prac projektowych i doświadczeń wynika, że jest możliwe zaadaptowanie klasycznej architektury potokowej do realizacji bardziej zaawansowanych algorytmów przetwarzania i analizy obrazów, takich jak np. etykietowanie. Spośród wielu sposobów etykietowania, najlepiej do tego celu nadaje się algorytm "tablicy sklejeń" [60][95] (ang. *connected component labeling*). Jako kryterium wyboru algorytmu ustalono ilość pamięci potrzebnej do etykietowania oraz sposób odwołań do jej zawartości. W literaturze spotyka się również metody rekurencyjne [17][45] lub iteracyjne przeciwbieżne [9][15]. Wadą metod rekurencyjnych jest duże zapotrzebowanie na pamięć stosu oraz duża zależność czasu wykonania od treści obrazu. Przeciwbieżne metody iteracyjne cechują się stosunkowo krótkim czasem realizacji, ale wymagają odwrócenia kolejności odwołań do pamięci, co stoi w sprzeczności z ideą zachowania kolejności przetwarzanych pikseli w strumieniu wizyjnym.

Dla algorytmu indeksacji realizowanego metodą "tablicy sklejeń" można precyzyjnie określić ilość pamięci potrzebnej do wykonania etykietowania. Niezbędny jest bufor pamięci na przechowanie rezultatów pośrednich etykietowania o rozmiarze $M \times N$ oraz pamięć tablicy sklejeń wykorzystywana w sortowaniu etykiet o granicznym rozmiarze G'_{max}. Potokowa wersja wymaga również bufora FIFO (ang. *First In First Out*) do zapamiętania kontekstu przestrzennego przedstawionego na rysunku 6.18. W algorytmie wyszczególnić można trzy oddzielne fazy:

2a etykietowanie wstępne,

2b sortowanie tablicy sklejeń,

2c przekodowanie wyników etykietowania wstępnego uporządkowanymi etykietami (LUT).

A	B	C
D	P	

Rysunek 6.18: Kontekst przestrzenny w pierwszym etapie indeksacji.

Nieoznaczoność kontekstu w algorytmie indeksacji wiąże się z propagacją etykiety danego piksela na ośmiu najbliższych sąsiadów oraz pośrednio na wszystkich kolejnych. Rozłożenie algorytmu etykietowania na trzy wyżej wymienione etapy pozwala uniknąć konieczności analizy nieoznaczonego kontekstu piksela. Redukuje analizowane otoczenie do kontekstu lokalnego (rys. 6.18). Podobnie jak indeksacja wstępna, również bezkontekstowe przekształcenie LUT jest bardzo dobrze uwarunkowane do realizacji strumieniowej.

Etap drugi nie może być rozpoczęty zanim etykietowanie wstępne nie zostanie całkowicie zakończone. Przekształcenie LUT jest możliwe do wykonania dopiero po zakończeniu porządkowania tablicy sklejeń. Sortowanie indeksów w tablicy sklejeń jest więc jedynym źródłem zmienności czasu realizacji w całym algorytmie etykietowania, ponieważ powoduje wstrzymanie obliczeń w pozostałych jednostkach przez czas zależny od liczby i kształtu obiektów w obrazie. Czas porządkowania tablicy sklejeń (wyrażony liczbą cykli zegara) w implementacji zrównoleglonej jest równy rozmiarowi tablicy sklejeń, więc w granicznym przypadku opóźnienie transportowe całego algorytmu wynosi $L_s = MN + G_{max} + r_l$. Aby uniknąć przestoju w strumieniu danych, moduł porządkowania tablicy sklejeń i sama pamięć przechowująca informację o przyporządkowaniach etykiet została zduplikowana. Pozwala to na współbieżne działanie operacji LUT i etykietowania wstępnego pomimo opisanego konfliktu. W omawianej realizacji wszystkie operacje wykonywane są współbieżnie w tempie jednego cyklu na kwant danych, zatem czynnik r_l dokładnie odwzorowuje czas potrzebny na rozbieg potoków w segmentacji wstępnej, operacji porządkowania tablicy sklejeń i operacji LUT. Organizacja strumienia wizyjnego w cyfrowych i analogowych systemach wizyjnych zakłada odstęp czasowy pomiędzy poszczególnym ramkami obrazu T_i. Wielkość interwału zależy od własności czujnika wizyjnego oraz bieżących ustawień migawki elektronicznej. W większości przypadków, jeśli etap przetwarzania wstępnego został prawidłowo wykonany i warunki akwizycji są dobre, można przyjąć, że aktualna liczba poindeksowanych obiektów w obrazie L jest znacznie mniejsza od wartości granicznej G_{max}. Pomimo nieregularności wynikającej z porządkowania tablicy sklejeń, jest mało prawdopodobne, że wystąpi nadpisanie danych w potoku. W przeciwnym wypadku, gdy występuje ryzyko że interwał między ramkami $T_i < L$, można zastosować redundantną strukturę elementów przetwarzających. Dodatkowa jednostka pozwoli na dokładne wykonanie całej operacji niezależnie od aktualnej liczby pośrednich etykiet. Rozwiązanie to wiąże się z dwukrotnym wzrostem zużytych zasobów obliczeniowych i wydłużeniem opóźnienia transportowego.

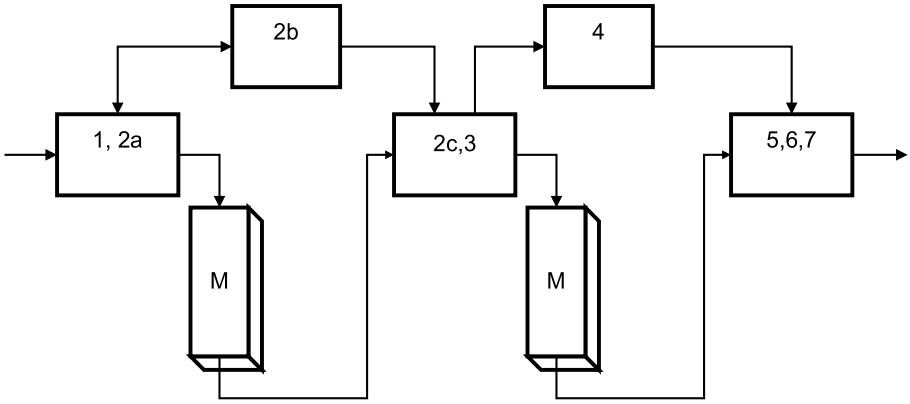
Poprzez zastosowanie techniki zrównoleglenia operacji oraz zwielokrotnienia jednostki porządkowania tablicy sklejeń, potokowa realizacja algorytmu indeksacji (metoda tablicy sklejeń) okazała się możliwa do wykonania z zachowaniem niewielkiej zmienności (20%) opóźnienia transportowego. Co więcej, opóźnienie to jest skompensowane przez interwały w transmisji ramek obrazu w strumieniu wizyjnym.

6.5 Równoległa implementacja nieregularnych algorytmów

W rozdziale 6.4.3 omówiono aspekty związane ze zrównoleglonąą realizacją algorytmów, których czas realizacji zależy od treści obrazu, jednak liczebność oraz postać danych wejściowych i wynikowych jest zachowana. W obu przypadkach jest to obraz o identycznych rozmiarach. Odmienna jest jedynie interpretacja piksela, który na wejściu niesie informację o jasności i ewentualnie o barwie, zaś etykieta piksela wynikowego zawiera informację o jednoznacznym przyporządkowaniu do określonego obiektu.

Poglądowy schemat systemu rozpoznawania znaków wraz z głównymi ścieżkami przepływu danych przedstawiono na rysunku 6.19. Po etykietowaniu, realizowany jest pomiar cech, ich selekcja, a następnie skalowanie obiektów do rozmiaru matrycy znaku. We wszystkich tych operacjach postać oraz liczba danych wejściowych nie jest zachowana na wyjściu. W przypadku wyznaczania cech wynikowe dane mają postać struktury zawierającej parametry prostokąta opisującego, pole powierzchni i numer etykiety, w innych: rezultatem jest również obraz lecz o rozmiarach niezależnych od wielkości obszaru wejściowego. Spośród trzech wymienionych operacji, jedynie selekcja cech zachowuje rozmiary struktur danych, chociaż liczba obiektów wynikowych zaakceptowanych do dalszej obróbki ulega dalszej redukcji.

Zaobserwowano, że pomiar cech (opisany w rozdziale 6.3.4) w systemie strumieniowym spełnia wymogi bezkontekstowej operacji przetwarzania obrazów. To samo dotyczy operacji LUT wykonywanej w końcowej fazie indeksacji. Uzasadniona była więc równoczesna realizacja obu operacji ponieważ działają one na tych samych danych. Wynikiem przekształcenia LUT, oprócz przyporządkowania etykiet, jest zatem opis każdego obiektu, zawierający prostokąt opisujący i pole powierzchni w przydzielonym obszarze pamięci. Wyliczone cechy obiektów poddawane są selekcji ze względu na kryteria opisane w rozdziale 6.3.4. W tym etapie możliwe jest również zebranie statystycznych informacji o cechach, które to informacje służą w kolejnym etapie rozpoznawania do automatycznego ustalenia kryterium selekcji na podstawie średniej lub maksymalnej wysokości znaku h_{max} (6.8). Stopień skomplikowania i równoległości dla operacji selekcji obiektów, jest podobny jak dla porządkowania tablicy sklejeń. Kolejne elementy



Rysunek 6.19: Schemat przepływu danych w systemie rozpoznawania znaków.

Oznaczenia przyjęte w rysunku: 1 - segmentacja, 2a - indeksacja wstępna, 2b - porządkowanie tablicy sklejeń, 2c - przekodowanie LUT, 3 - pomiar cech, 4 - selekcja, 5 - skalowanie, 6 - rozpoznawanie, 7 - ocena wyniku, M - bufor ramki w pamięci.

są odczytywane z tablicy cech i na podstawie klasyfikacji wyliczonych proporcji oraz współczynnika wypełnienia (przez porównanie z zadanymi progami) oznaczane są jako przydatne do dalszej analizy lub odrzucane.

Wybrane obiekty przeznaczone są do skalowania i zaprezentowania sieci neuronowej w celu rozpoznania. Potokowa realizacja omówionej w rozdziale 6.3.5 metody skalowania obiektów gwarantuje regularność w obrębie przetwarzania jednego znaku. Liczba znaków nie jest znana na etapie implementacji algorytmu. Proces skalowania i rozpoznawania jest zatem uruchamiany wielokrotnie, zależnie od liczby R pozytywnie zweryfikowanych obiektów w czasie wykonania.

Moduł sztucznej sieci neuronowej jest połączony z potokiem skalującym poprzez strumień danych, więc do czasu próbkowania obiektu należy dodać czas wiążący się z ewaluacją poszczególnych warstw sztucznej sieci neuronowej. W szczególności, akumulacja pobudzenia w warstwie wejściowej następuje równocześnie z procesem skalowania. Niewielkie narzuty czasowe wiążą się z inicjalizacją akumulatorów poszczególnych neuronów czynnikami stałymi (1 cykl) oraz iteracją pętli warstwy wejściowej sztucznej sieci neuronowej (21 cykli). Poprzez wykorzystanie równoległego przepływu danych w strukturze sztucznej sieci neuronowej liczba operacji przypadających na ewaluację funkcji przejścia warstwy ukrytej i wyjściowej zredukowana została do 31 cykli. Dodatkowe 20 cykli przeznaczone jest na wskazanie neuronu, który wygenerował największe pobudzenie oraz określenie pewności rozpoznania, poprzez wyliczenie różnicy pomiędzy dwoma największymi wynikami. Zapis wynikowej informacji o dokonanym rozpoznaniu zajmuje dodatkowych kilka cykli zależnie od szczegółowości (rozpoznanie, pewność rozpoznania, lokalizacja - ok. 4 cykle). W procesie skalowania narzut na obieg pętli iterującej wynosi 6 cykli. Łączny naddatek dla współbieżnych procesów skalowania i obliczeń sztucznej sieci neuronowej wynosi 83 cykle. W odniesieniu do

liczby pikseli w matrycy znaku ($K^2 = 400$) stanowi to ok. 21% (wynik dla etapu 5 w tabeli 6.7).

W tabeli 6.7 przedstawiono wynik analizy odwołań do danych wizyjnych po zastosowaniu operacji zrównoleglenia opisanych w dalszych podrozdziałach. Z zamieszczonych danych wynika jednoznacznie, że redukcja w odwołaniach do pamięci nastąpiła dla danych o dużym ziarnie granulacji (por. tabela 6.3). Dotyczy to zawartości bufora ramki obrazu oraz matrycy znaków. W pierwszym przypadku wiąże się to ze zmniejszeniem opóźnienia transportowego, a w drugim skutkuje oszczędnością pamięci.

Tabela 6.7: Czasy wykonania operacji w systemie rozpoznawania znaków.

Etap	Rodzaj operacji na obrazie	Liczba iteracji	Liczba cykli w iteracji	Względna liczba cykli*	Rozstęp względny cykli v
1	segmentacja	1	MN	1.0000	0.000
2a	indeksacja wstępna				
2b	porządkowanie etykiet	1	G'	0.2500	1.000
2c	LUT	1	MN	1.0000	0.000
3	pomiar cech				
4	selekcja	1	G	0.2500	1.000
5	skalowanie	R	$K^2 + 83$	1.2075	1.000
6	rozpoznawanie				
7	ocena wyniku				
	Wynik zbiorczy			3.7075	0.461

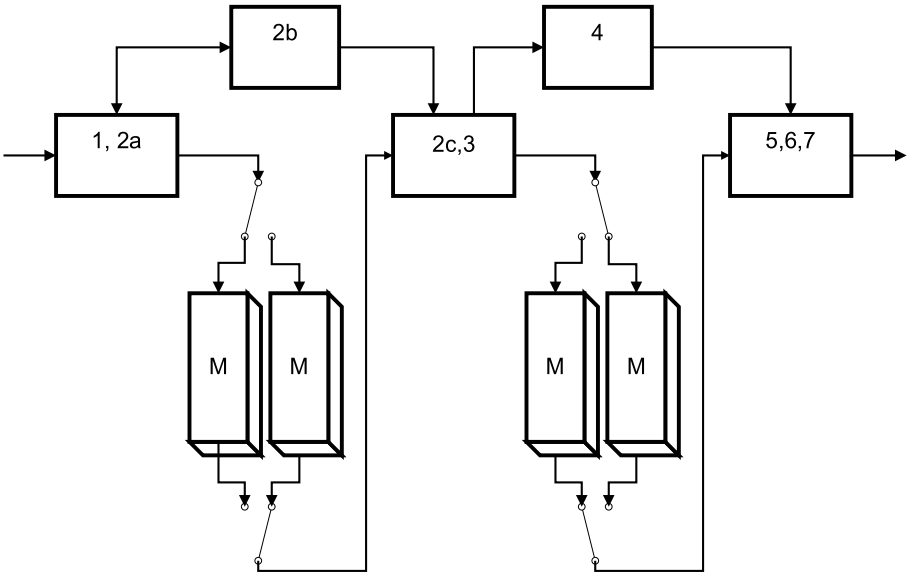
* Względna liczba cykli liczona jest jako iloraz rzeczywistego rozmiaru i rozmiaru ramki obraz MN z uwzględnieniem maksymalnych wartości R i G wg. przedziałów (6.25)(6.26).

W procesie projektowym drugiej części toru wizyjnego (selekcja, skalowanie, ocena, szacowanie wyniku) utworzono dwa niezależne moduły działające współbieżnie (rys. 6.19). Pierwszy - odczytuje cechy wyselekcjonowanych obiektów z tablicy i skaluje obiekty. Drugi potok wykonuje działania sieci neuronowej. Z punktu widzenia ilości danych wejściowych oraz rezultatów obydwa procesy różnią się istotnie. Wejściami dla operacji skalowania są tablica struktur zawierających cechy obiektu oraz poetykietowany obraz. Na wyjściu sztucznej sieci neuronowej pojawia się rozpoznany znak przypisany do danego obszaru obrazu. Obydwa procesy komunikują się poprzez matrycę znaku o stałym rozmiarze $K \times K$ uzyskaną w trakcie skalowania.

Niski poziom granulacji

Założeniem omawianej implementacji systemu było działanie w pełni przypływowe, jednak selekcja cech, podobnie jak porządkowanie tablicy sklejeń po indeksacji

wstępnej, wstrzymuje transfer danych (przez liczbę cykli równą liczbie zidentyfikowanych obiektów G). Jest to przyczyną dużego rozstępu czasu wykonania $v \approx 0,46$ (tabela 6.7) i tym samym uniemożliwia strumieniowe przetwarzanie danych, ponieważ nie jest dogodne buforowanie tak dużej liczby danych, ani nie jest możliwe skompensowanie tego interwału przez czas naświetlania czujnika wizyjnego kamery.



Rysunek 6.20: Potok gruboziarnisty w systemie rozpoznawaniu znaków.

Rozwiązanie tego problemu stanowi alokacja podwójnych bloków pamięci, które służą do buforowania poszczególnych obrazów w strumieniu wizyjnym. Możliwe jest równoczesne działanie indeksacji wstępnej (1) z segmentacją (2a) oraz przekodowania LUT (2c) z wyznaczaniem cech (3). Dzięki topologii systemu jak na rysunku 6.20 uzyskano również efekt zrównoleglenia operacji kłopotliwych w strumieniowej realizacji (porządkowanie etykiet, selekcja obiektów). Przy takiej organizacji przetwarzania danych w strumieniu, nieregularność wyznaczona jest przez liczbę etykiet G oraz liczbę cykli rozpoznawania przekraczającą rozmiary obrazu MN (por. tabela 6.7). Uwzględniając równoległość na niskim poziomie granulacji przedstawioną na rysunku 6.21 należałoby przyjąć wartość (6.29) jako końcowy rozstęp, co w efekcie daje rezultat podobny do przedstawionego tabeli 6.7.

$$v = \frac{G + R(K^2 + 83) - MN}{G + R(K^2 + 83)} = 0.3139 \tag{6.29}$$

Widać więc, że rozstęp czasu przetwarzania danych utrzymuje się na w przybliżeniu na stałym poziomie w trakcie zrównoleglania, choć jak zaznaczono we wcześniejszym akapicie, jest to wartość, która uniemożliwia płynne przetwarzanie danych. Można jednak przypuszczać, że wartość ta jest znacznie zawyżona w sto-

	$1\|\|2a$		$1\|\|2a$		$1\|\|2a$		$1\|\|2a$	...
		$2b$		$2b$		$2b$		...
			$2c\|\|3$		$2c\|\|3$		$2c\|\|3$	...
				4		4		...
					$5\|\|6\|\|7$		$5\|\|6\|\|7$	...
operacje								
cykle:								
$MN\times$max	1.21	0.25	1.21	0.25	1.21	0.25	1.21	...
$MN\times$min	1.00	0.00	1.00	0.00	1.00	0.00	1.00	

Rysunek 6.21: Tor rozpoznawania jako potok gruboziarnisty.

sunku do realnych wartości, ponieważ nie jest możliwa sytuacja, aby jednocześnie G i R uzyskały wartości maksymalne. Fakt ten potwierdzony jest przez doświadczenia i analizę wpływu treści obrazu po segmentacji na liczbę etykiet, a tym samym liczby G obiektów w obrazie. Jakościowo zjawisko to można zobrazować przy pomocy następującego doświadczenia myślowego:

- załóżmy, że $G = G_{max}$, wówczas zgodnie z opisem przedstawionym w rozdziale 6.3.3 wszystkie obiekty muszą mieć rozmiar jednego piksela co sprawia że nie zostaną zaklasyfikowane jako poprawne znaki do rozpoznawania czyli $R = 0$,
- załóżmy, że $R = \frac{MN}{K^2}$ a prostokąty opisujące mają rozmiary $K \times K$: wówczas zgodnie z opisem przedstawionym w rozdziale 6.3.3 nie jest możliwe aby $R = G_{max}$ ponieważ obszar znaku musi tworzyć zbiór sąsiadujących ze sobą pikseli, co tym samym pomniejsza maksymalną liczbę potencjalnych etykiet o czynnik $\frac{K^2}{4}$.

Zjawisko kompensacji nieregularności maksymalnych czasów wykonania krytycznych operacji w potoku można zatem opisać równaniem (6.30):

$$G = G_{max} - R\frac{K^2}{4} \tag{6.30}$$

W efekcie, podstawiając wyliczoną wartość G do równania (6.29) otrzymujemy precyzyjne oszacowanie nieregularności działania potoku (6.31):

$$v = \frac{83}{K^2 + 83} = 0.1718 \tag{6.31}$$

Z przeprowadzonej analizy wynika, że w projektowaniu przepływowych systemów wizyjnych należy brać pod uwagę nie tylko ograniczenia technologiczne, rozmiary i format danych, ale również zawartość sygnału wizyjnego. Przykład ten dobitnie pokazuje, że analiza jakościowa obrazu dostarcza również informacji przydatnych w doborze architektury obliczeniowej.

Przedstawiony system wizyjny obejmujący segmentację, indeksację, analizę obiektów oraz ich rozpoznawanie nie jest wewnętrznie jednorodny. Na poszczególnych etapach przetwarzania zmienia się sposób reprezentacji danych. Zmienna jest nie tylko rozdzielczość obrazu podczas skalowania obiektów, ale również ich liczba. Mimo to jednak, system cechuje się niską wartością granicznego rozstępu $v \simeq 0.17$ która pozawala na płynne przetwarzanie strumienia wizyjnego z kamery.

6.5.1 Wpływ równoległości na przepustowość

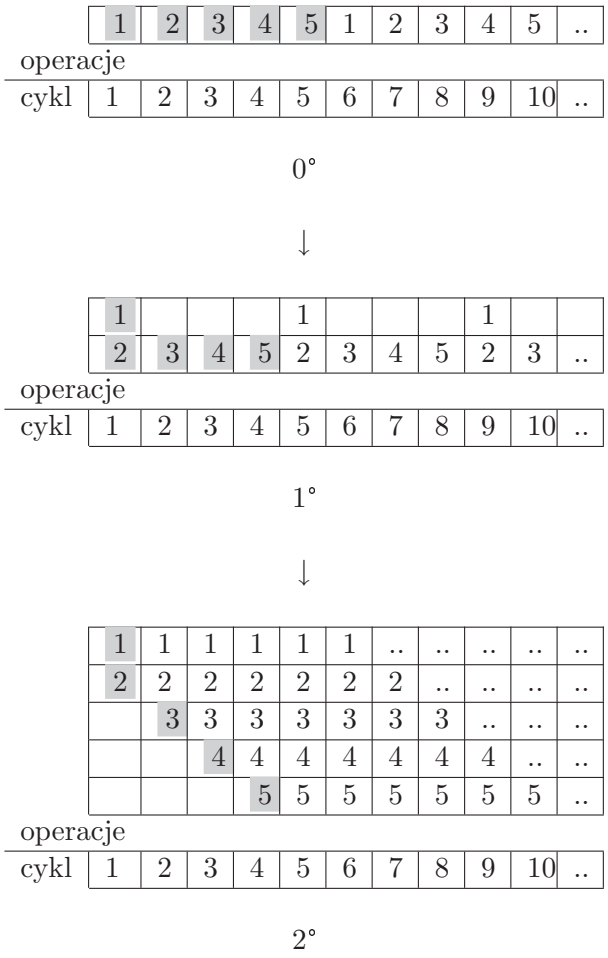
Porównanie obu parametrów (czasu obliczeń i opóźnienia transportowego) zostanie przeprowadzone na przykładzie zrównoleglenia funkcji skalującej obiekty przeznaczone do rozpoznawania przez sztuczną sieć neuronową. Idea normalizacji rozmiarów obiektu została przedstawiona w rozdziale 6.3.5. Przyjęto, że na uproszczoną procedurę skalowania składają się pięć operacji:

1. inkrementacja zmiennych iterujących pętlę,
2. określenie współrzędnych w matrycy,
3. określenie najbliższego sąsiada,
4. odczyt piksela wejściowego z bufora,
5. wysłanie wynikowego piksela do sieci neuronowej.

Analiza potoku skalującego na wysokim poziomie granulacji

Zakładając, że każda operacja realizowana jest przez jeden cykl zegara, sumaryczny czas skalowania wynosi $T_c = T_k = 5K^2$ (rys. 6.22). Wykorzystanie równoległości operacji 1 i 2 w etapie 1° skutkuje zmniejszeniem liczby cykli. Zmienna indeksująca jest inkrementowana równocześnie z wyznaczaniem koordynat próbkowania matrycy znaku $T_c = T_k = 4K^2$. Większy stopień zrównoleglenia jest możliwy do osiągnięcia dopiero zastosowaniu mechanizmu potokowego 2°, który polega na takim uszeregowaniu operacji w torze przetwarzania danych, że operacje zależne od siebie w porządku czasu wykonywane są współbieżnie. Dane będące wynikami poszczególnych operacji dostępne są jednocześnie (co odpowiada równoległości danych DLP) dzięki czemu kolejne operacje mogą być wykonywane jednocześnie. Równoczesność ta jest jednak rozłożona w czasie ze względu na propagację danych w potoku.

Z zastosowaniem potokowej metody przetwarzania (przestawionej na rysunku 6.22) wiąże się konieczność wykonania rozbiegu potoku w celu dostarczenia danych do poszczególnych elementów przetwarzających. W przypadku przedstawionym na rysunku 6.22 inicjalizacja potoku trwa 4 cykle zegara. Ilość następujących iteracji jest dokładnie równa ilości przetwarzanych danych. Pomimo dodatkowych kilku cykli inicjalizacji, zysk czasowy jest ewidentny bo niemal 5-cio krotny w



Rysunek 6.22: Etapy zrównoleglenia w architekturze potokowej.

stosunku do realizacji sekwencyjnej . Ze względu na elementarny kwant danych jakim jest tutaj piksel matrycy znaku należy brać pod uwagę opóźnienie w potoku określone przez $L_p = 4$. Ponieważ jednak proces skalowania wstrzymuje wykonanie obliczeń w potoku głównego strumienia wizyjnego, należy wziąć pod uwagę całkowity czas potrzebny na wykonanie skalowania od inicjalizacji potoku do uzyskania ostatniego piksela matrycy znaku $L_s = R(K^2 + L_p)$. Dla tak określonego sposobu zrównoleglenia obliczeń w potoku można wyznaczyć warunek przydatności (6.32) zrównoleglenia różnych operacji w zależności od stopnia złożoności potoku i ilości danych. Kryterium oceny stanowi tutaj liczba cykli konieczna do przeprowadzenia obliczeń na określonym zbiorze danych. Lewy człon nierówności (6.32) odzwierciedla liczbę cykli konieczną do przeprowadzenia obliczeń w sposób sekwencyjny, a prawy: ten sam parametr po zastosowaniu architektury potokowej dla tego samego zadania obliczeniowego. W pierwszym przypadku czas oczekiwania na rezultat wynosi n_p a w drugim i, przy czym $i \leqslant d$. Realizacja potokowa

charakteryzuje się znacznie większą przepustowościąbo prawie d-razy większą.

$$n_p d < (d + i) \tag{6.32}$$

gdzie:

$d \in \mathbb{N}$ – liczba kwantów danych do przetworzenia,
$n_p \in \mathbb{N}$ – liczba etapów potoku,
$i \in \mathbb{N}$ – liczba cykli potrzebna na rozbieg potoku, $i \leqslant n$.

Przy założeniu, że $i = d$ można wykazać że, nierówność (6.32) niespełniona jest jedynie dla zestawów wartości $n_p = 1, d = 1$ oraz $n_p = 1, d = 2$. Oznacza to, że jeśli wykonywane obliczenia składają się z szeregu elementarnych operacji możliwych do uszeregowania w czasie, korzystne jest zastosowanie równoległości OLP w architekturze potokowej, jeśli liczba danych przekracza 1. Dotyczy to obliczeń, w których kolejność wykonywania elementarnych operacji jest deterministyczna i niezależna od treści danych wejściowych.

Dekompozycja złożonych operacji numerycznych na elementarne jest powszechnie wykorzystywanym zabiegiem w cyfrowych systemach obliczeniowych. Wyznaczanie wartości złożonych wyrażeń wiąże się z określonym czasem propagacji T_{max}, proporcjonalnym do liczby poziomów elementów logiki kombinatorycznej i zależnym od technologii. Największy czas propagacji ścieżki kombinatorycznej bezpośrednio wyznacza graniczną częstotliwość pracy systemu jak w zależności (6.33).

$$f_g = \frac{1}{T_{max}} \tag{6.33}$$

gdzie:

f_g – graniczna częstotliwość pracy,
T_{max} – czas propagacji najwolniejszego elementu przetwarzającego.

Zmniejszenie największego czasu propagacji T_{max} pozwala na osiągnięcie większej granicznej częstotliwości pracy systemu obliczeniowego. Kosztem takiego zabiegu jest zwiększenie opóźnienia transportowego liczonego w dyskretnych cyklach zegara. Mimo to jednak przepustowość systemu ulega najczęściej zwiększeniu ze względu na wzrost częstotliwości f_g.

Rodzaj równoległości wykorzystanej w potokowej realizacji różni się od sposobu zrównoleglenia opisanego w paragrafie 6.4.3. W etapie 2° wykorzystano równoległość OLP, która polega na wykonaniu kilku różnych operacji na różnych danych jednocześnie. W drugim etapie zarówno ilość zasobów jak i opóźnienie

Tabela 6.8: Ocena zrównoleglenia systemu rozpoznawania znaków.

	0°	0°- 1°	1°- 2°
prawo Amdahl'a (4.3)			
n	1	2	4
s	1	2/5	0
p	0	3/5	4
S_a	1	5/4	4
prawo Gustafson'a (4.5)			
s'	1	1/4	0
p'	0	5/4	1
S_g	1	5/4	4
przyspieszenie całkowite (4.2)			
T_1	5	5	5
T_n	5	4	1
S_m	**1.00**	**1.25**	**5.00**
efektywność (4.4)			
E		0.625	1.000
opóźnienie transportowe			
L	5	4	4
przepustowość			
P_c	1/5	1/4	1

transportowe nie zmienia się. Dotyczy to zasobów pamięciowych (5 rejestrów) i elementów obliczeniowych dedykowanych do wykonania ściśle określonych operacji.

Mimo, iż opóźnienie transportowe utrzymuje się na nieznacznie niższym poziomie ($L = 4$), z 5-cio krotnym zrównolegleniem wiąże się 5-cio krotny wzrost przepustowości P_c. W analizie sprzętowych architektur wygodnie jest posługiwać się przepustowością znormalizowaną P_c, która pomija fizyczny czynnik czasu propagacji wynikający z zastosowanej technologii. Znormalizowana przepustowość (6.34), w odróżnieniu od zdefiniowanej we wzorze (5.3), odniesiona jest do cykli zegara a nie do bezwzględnego czasu pomiaru:

$$P_c = \frac{1}{L} = T_{clk} P \tag{6.34}$$

gdzie:

P_c – przepustowość znormalizowana do cykli zegara,

L – opóźnienie transportowe, w cyklach zegara,

T_{clk} – okres zegara,

P – przepustowość bezwzględna określona wzorem (5.3).

Wzrost równoległości pomimo nieznacznego zmniejszenia czasu propagacji przekłada się bezpośrednio na zwiększenie przepustowości i uzyskanie docelowej wartości 1 piksel/cykl (patrz tabela 6.8).

Analizując zmianę przepustowości i opóźnienia transportowego przedstawionego w tabelach 6.5 i 6.8 stwierdzono, że w przypadku wykorzystania prostej równoległości operacji, przepustowość jest odwrotnie proporcjonalna do opóźnienia transportowego. Po zastosowaniu architektury potokowej, przepustowość zwiększa się przy stałym lub rosnącym opóźnieniu transportowym L. Zjawisko to można zaobserwować na rysunku 6.23. Otóż istnieje możliwość takiej (potokowej) reorganizacji segmentacji na etapie zrównoleglenia 2° (rozdział 6.4.3), że przy zachowaniu tej samej ilości zasobów obliczeniowych i pamięciowych opóźnienie transportowe będzie wynosić 2 a przepustowość wzrośnie do 1 pixel/cykl. Przedstawione rozwiązanie cechuje się lepszymi parametrami technologicznymi,

	1||..||6	1||..||6	1||..||6	..	..
		7||..||11	7||..||11	7||..||11	..
operacje					
cykl	1	2	3	4	..

Rysunek 6.23: Przetwarzanie potokowe w drugim etapie zrównoleglenia.

jeśli chodzi o graniczną częstotliwość pracy, zależną od długości ścieżki kombinatorycznej - patrz ograniczenie (6.33). Operacje logicznej koniunkcji wykonywane są w odrębnym cyklu przez wydzieloną, kombinatoryczną część układu cyfrowego.

Analiza potoku skalującego na średnim poziomie granulacji

Rozpatrując potokową architekturę jednostki skalującej, jako element przetwarzający o średnim poziomie granulacji, należy wziąć pod uwagę czas realizacji obliczeń z uwzględnieniem rozmiaru przetwarzanych danych. Jest to uzasadnione, ponieważ proces skalowania powtarzany jest wielokrotnie (dla różnych obiektów w jednym obrazie) i nie można zupełnie zaniedbać czasu poświęconego każdorazowo na rozbieg potoku. Czynnik ten nie został uwzględniony w tabeli 6.8 podczas analizy na wysokim poziomie granulacji. Realną wartość przyspieszenia można opisać posiłkując się formułą (6.35) zaczerpniętą z pracy [62]. Przyspieszenie wyznaczone tą metodą (6.35) przyjmuje wartość $S_n = 3.97$. Jest to liczba znacznie odbiegająca od uzyskanej w tabeli 6.8. Główną przyczyną rozbieżności jest nieuwzględnienie inkrementacji zmiennych iterujących w końcowej liczbie stopni potoku. Przekształcenie formuły (6.35) do postaci (6.36) pozwala uzyskać wiarygodny wynik $S_n = 4.96$.

$$S_{np} = \frac{n_p d}{n_p + d - 1} \tag{6.35}$$

gdzie:

S_{np} – przyspieszenie jednostki potokowej,
$d \in \mathbb{N}$ – liczba kwantów danych do przetworzenia,
$n_p \in \mathbb{N}$ – liczba stopni potoku,

$$S_{ns} = \frac{n_s d}{n_p + d - 1} \tag{6.36}$$

gdzie:

S_{ns} – przyspieszenie jednostki potokowej,
$d \in \mathbb{N}$ – liczba kwantów danych do przetworzenia,
$n_s \in \mathbb{N}$ – liczba sekwencyjnych operacji,
$n_p \in \mathbb{N}$ – liczba stopni potoku.

Po podstawieniu danych ($d = K^2 = 400$, $i = 4$, $S = 5$) do wzoru (6.37) błąd względny e_s przyspieszenia popełniony przy wyznaczaniu przyspieszenia zamieszczonego w tabeli 6.8 wynosi w przybliżeniu 1%.

$$e_s = \left| \frac{\frac{nd}{i+d} - S}{S} \right| \tag{6.37}$$

gdzie:

e_s – błąd względny przyspieszenia,
d – liczba kwantów danych do przetworzenia,
n – liczba elementarnych operacji w potoku,
i – liczba cykli potrzebna na rozbieg potoku,
S – przyspieszenie wyznaczone analitycznie podstawione z etapu 2° w tabeli 6.8.

Można zaobserwować, że błąd e_s dąży do zera przy $i = const$, $n = const$ oraz $d \to \infty$. Rzeczywisty czas potrzebny do przeprowadzenia skalowania wszystkich

obiektów w jednym obrazie zależy liniowo od ich liczby i określony jest równaniem (6.38).

$$C_s = R(i + K^2) \tag{6.38}$$

gdzie:

C_s – czas skalowania w cyklach zegara,

R – liczba obiektów przeznaczonych do skalowania, $R \in \langle 0, G_{max} \rangle$,

i – liczba cykli potrzebna na rozbieg potoku.

Czynnik R zależny od treści obrazu jest przyczyną dużego rozstępu czasu przetwarzania danych przez operator skalowania (patrz tabela 6.7).

6.5.2 Zrównoleglanie sztucznej sieci neuronowej

Ocena działania sekwencyjnego modelu sieci neuronowej sprowadza się do określenia liczby wywołania operatorów sumowania, mnożenia i modułów funkcji przejścia. Dla omawianego przypadku wartości te zebrano w tabeli 6.9.

Tabela 6.9: Szacowanie zasobów modelu sieci neuronowej.

element przetwarzający	**liczba wywołań**	
sumator	$(1 + K^2)l_1 + (1 + l_1)l_2$	4120
mnożenie	$(K^2)l_1 + l_1 l_2$	4100
funkcja przejścia	$l_1 + l_2$	20

gdzie:

K^2 – liczba pikseli w matrycy znaku,

l_1 – liczba neuronów w warstwie ukrytej,

l_2 – liczba neuronów w warstwie wyjściowej.

Dodatkowe oszacowanie można przeprowadzić ze względu na liczbę i sposób reprezentacji wag poszczególnych warstw sieci. Zakładając klasyczny schemat składowania wag we wspólnej, liniowej przestrzeni adresowej, należy zaadresować $(1 + K^2)l_1 + (1 + l_1)l_2$ komórek. Wartość ta bezpośrednio pokrywa się z liczbą sumowań przedstawioną w tabeli 6.9.

Zaimplementowany moduł sztucznej sieci neuronowej cechuje się dużym stopniem równoległości danych wynikającym z rozmiarów warstwy ukrytej sztucznej sieci neuronowej. Na poziomie warstwy wejściowej pełna współbieżność DLP nie

jest wykorzystana, ze względu na szeregowy sposób transmisji wektora wejściowego sieci neuronowej z modułu skalującego. Drugim powodem jest czynnik technologiczny, który nakazuje zastosowanie możliwie małej liczby wejść sumatorów pobudzenia w komórkach pierwszej warstwy ukrytej. Zwiększenie liczby wejść sumatora powoduje znaczący wzrost czasu propagacji poprzez wydłużenie łańcucha przeniesień. Z definicji modelu sztucznej sieci neuronowej przedstawionego w rozdziale 6.3.6 wynika, że każdy neuron pierwszej warstwy ukrytej gromadzi sygnał pobudzający z $K^2 = 400$ wejść warstwy wejściowej. Wobec wielkiej liczby sygnałów do sumowania i ze względu na sposób akwizycji wzorca do rozpoznawania przyjęto, że wyliczanie pobudzenia w każdym z neuronów będzie realizowane przez sprzętowy moduł dodająco-mnożący MAC (ang. *Multiply and Accumulate*) składający się z:

- wejścia sygnału pobudzającego,
- wejścia wagi do mnożenia sygnału pobudzającego,
- wyjścia sygnału aktywacji komórki neuronu,
- jednostki mnożącej,
- jednostki sumatora,
- komórki pamięci (tj. rejestru akumulatora).

Moduł funkcji przejścia wykorzystuje prosty mechanizm progowania wartości wejściowej, aby określić wynikowy sygnał pobudzający. Obie warstwy (ukryta i wyjściowa) posiadają własną instancję modułu funkcji przejścia. Zrównoleglenie implementacji modelu sieci neuronowej przeprowadzono etapami na które składają się:

1° zwielokrotnienie elementów mnożących i akumulatorów,

2° równoczesne wykonanie operacji dodawania i mnożenia w module MAC,

3° przeniesienie wag z pamięci o liniowej przestrzeni adresowej do lokalnych tablic skojarzonych z poszczególnymi neuronami.

Przyspieszenie, które wynika z zastosowania współbieżnych modeli neuronów w etapie 1° i 2° jest niewielkie z powodu sekwencyjnego dostępu do pamięci wag. Dopiero przeniesienie wag połączeń synaptycznych do lokalnych pamięci skojarzonych z poszczególnymi neuronami pozwoliło uzyskać zamierzony efekt w etapie 3°. Dzięki temu wszystkie neurony w warstwie ukrytej mogą gromadzić sygnał pobudzający z określonego neuronu poprzedniej warstwy, z uwzględnieniem wagi danego połączenia. Efekt znacznego wzrostu przyspieszenia w etapie 3° należy kojarzyć z superliniowym przyspieszeniem, o którym pisano w rozdziale 4.3.2.

Poszczególne składowe wektora wejściowego sieci neuronowej podawane są sekwencyjnie na wszystkie komórki kolejnej warstwy jednocześnie. Stopień zrównoleglenia dla warstwy l należy szacować bezpośrednio liczbą neuronów w określonej warstwie l_i. Każda komórka MAC wykonuje operację mnożenia pobudzenia przez wagę oraz dodaje rezultat do zawartości rejestru akumulatora.

Tabela 6.10: Ocena zrównoleglenia modelu sieci neuronowej.

	0°	1°	2°	3°	4°	5°
przyspieszenie całkowite4.2						
n		22	42	42	8240	8240
T_1	8240					
T_n		8240	4140	414	4	1
S_m		**1.00**	**1.99**	**19.90**	2060.00	8240.00
efektywność 4.4						
E		0.05	0.05	0.47	0.25	1.00
opóźnienie transportowe						
L		8240	4140	414	4	4

Ostateczne przyspieszenie jakie uzyskano po zrównolegleniu 3° w danej warstwie ukrytej można przybliżyć równaniem (6.39). Obydwie warstwy: ukryta i wyjściowa mają po 10 komórek, więc w obu przypadkach przyspieszenie jest podobne i zbliżone do zamieszczonego w tabeli 6.10.

$$S_i = \frac{2l_{i-1}li}{l_{i-1}} = 2l_i \tag{6.39}$$

gdzie:

l_{i-1} – liczba neuronów w warstwie poprzedzającej, jednocześnie ilość wejść pobudzających każdego z neuronu warstwy i,

l_i – liczba neuronów w warstwie i,

S_i – przyspieszenie po zrównolegleniu warstwy i.

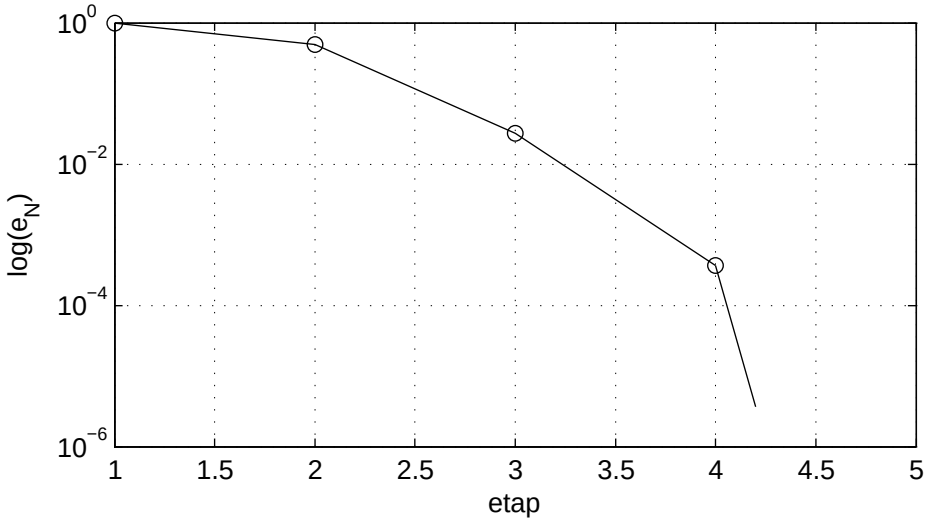
Dla porównania, w kolumnie oznaczonej 4° i 5° przedstawiono przewidywane rezultaty wynikające z potencjalnego stopnia współbieżności sztucznej sieci neuronowej. Czas wykonania $T_n = 4$ wiąże się z przeznaczeniem jednego cyklu na każdą z operacji:

- gromadzenie pobudzenia w warstwie ukrytej,
- ewaluacja funkcji przejścia warstwy ukrytej,
- gromadzenie pobudzenia w warstwie wyjściowej,
- ewaluacja funkcji przejścia w warstwie wyjściowej.

Wzrost efektywności do wartości 1.0 jest możliwy po zastosowaniu potokowej architektury 5°. Podobnie, jak w zrównolegleniu funkcji skalującej, wzrost efektywności nie wiązałby się ze zwiększonym zużyciem zasobów. Scenariusz 4° i 5° dla zastosowanego rozmiaru sieci neuronowej nie jest korzystny do realizacji w technologii cyfrowych układów rekonfigurowalnych. Przyczyną są nieakceptowalne czasy propagacji sumatorów, które w rozpatrywanych przypadkach mają po ok. 400 wejść.

W literaturze spotka się opisy architektur, w których neurony realizowane są w oparciu o analogową lub cyfrowo-analogową technologię półprzewodnikową [37]. W rozwiązaniach tych wykorzystuje się najczęściej model neuronu inspirowany anatomią biologicznych komórek w których pobudzenie ma naturę oscylacyjnych epizodów [25]. W pracy [47] opisano programową, sekwencyjną implementację oscylacyjnego modelu kory mózgowej, działającą w czasie rzeczywistym dla 10^4 neuronów z liczbą połączeń synaptycznych rzędu 10^6. Ponieważ model ten został zbudowany w celach poznawczych, jego autor przyjął zgodność czasu symulacji modelu z tempem zmian sygnału w naturalnych sieciach jako kryterium czasu rzeczywistego. Nie odnosi się więc ono do użytkowych wymogów związanych z tempem przetwarzania danych nadchodzących z otoczenia. Ponieważ pulsacyjny model neuronu opisany jest równaniem różniczkowym, jego sprzętowa implementacja w technice cyfrowej musiałaby zakładać większą częstotliwość pracy i większą złożoność modelu obliczeniowego w stosunku do wykorzystanego w rozprawie modelu cyfrowego. Nie była ona więc brana pod uwagę w badaniach.

Pomimo znaczącego przyspieszenia, efektywność w etapie 3° nie przekracza 0.5. Jak wynika z analizy przypadków 4° i 5° przyczyną niskiej efektywności jest nieoptymalna komunikacja danych na średnim poziomie granulacji. Spowodowane



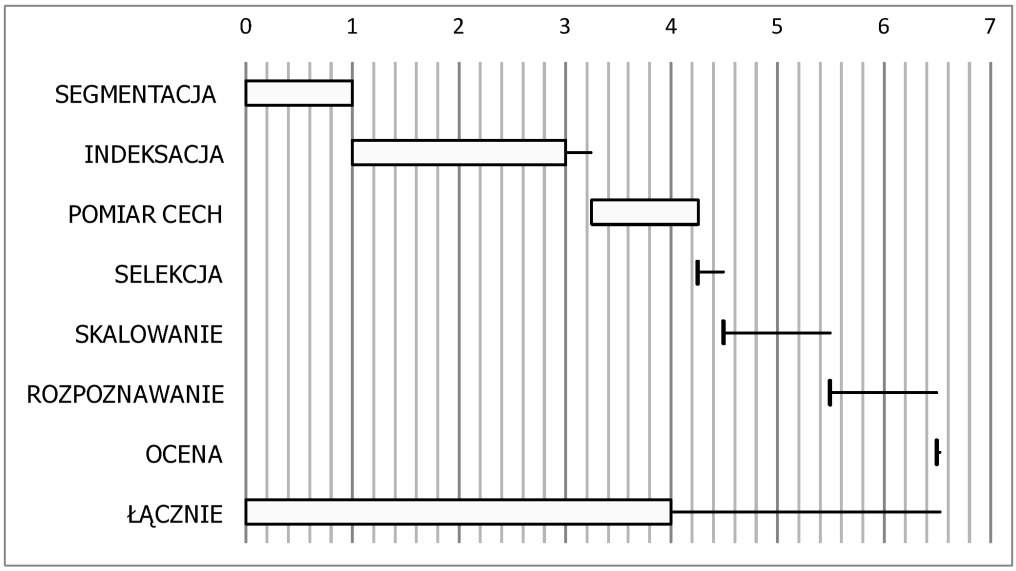
Rysunek 6.24: Wykres metryki Karp-Flatt'a w zrównoleglaniu sieci neuronowej.

jest to niewykorzystaniem równoległości OLP z przyczyn technologicznych.

Rysunek 6.24 pokazuje przebieg metryki e_n w kluczowych etapach zrównoleglania. Wykres zachowuje monotoniczność i dąży do zera na kolejnych etapach zrównoleglenia. Pomimo masowego zrównoleglenia zaproponowanego w etapie 4° efektywność spadła. Zgodnie z regułą wykorzystania metryki (4.6) przedstawioną w rozdziale 4.3 przyczyny należy upatrywać w architekturze systemu obliczeniowego lub sposobie implementacji. Usprawnienie komunikacji poprzez zastosowanie potokowego schematu przetwarzania danych według propozycji 5° mogłoby się przyczynić do wzrostu efektywności i dalszego spadku wartości metryki e_n.

6.6 Wyniki strumieniowej realizacji systemu rozpoznawania znaków

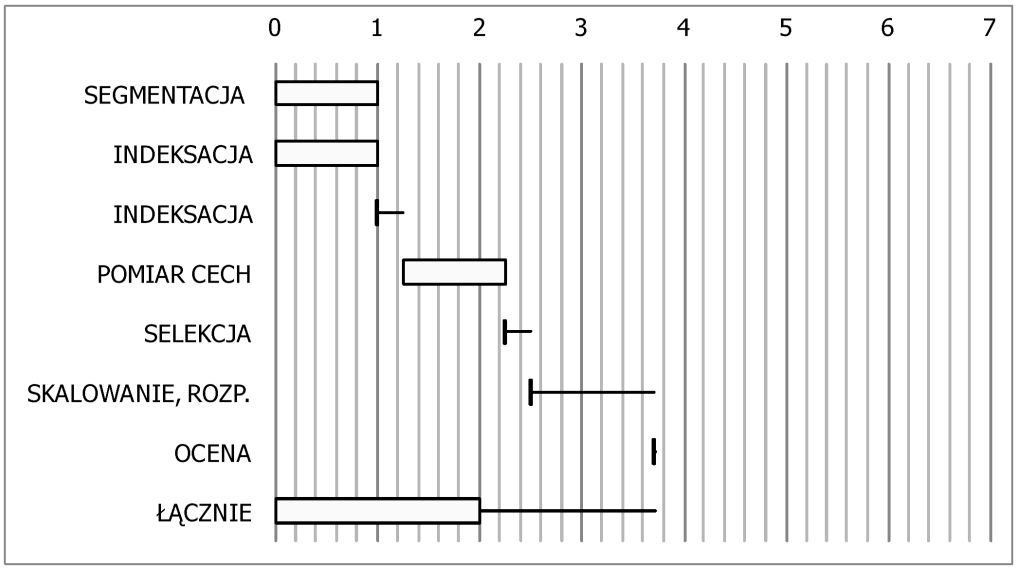
Czasy wykonania poszczególnych operacji toru wizyjnego rozpoznającego ręcznie pisane znaki zebrano w tabelach 6.3 i 6.7. W postaci graficznej został one przedstawione na wykresach 6.25 i 6.26. Poziome osie układu współrzędnych znormalizowane są do rozmiaru ramki obrazu MN. Obszary pokryte pustymi prostokątami odpowiadają operacjom o stałym czasie wykonania. Linie poziome wyznaczają granice w jakich zawierać się może czas wykonania operacji zależnie od treści danych. Wykres na rysunku 6.25 został wyznaczony w oparciu o szacowane rozmiary danych przy założeniu jednego taktu na każdy piksel w danej iteracji.



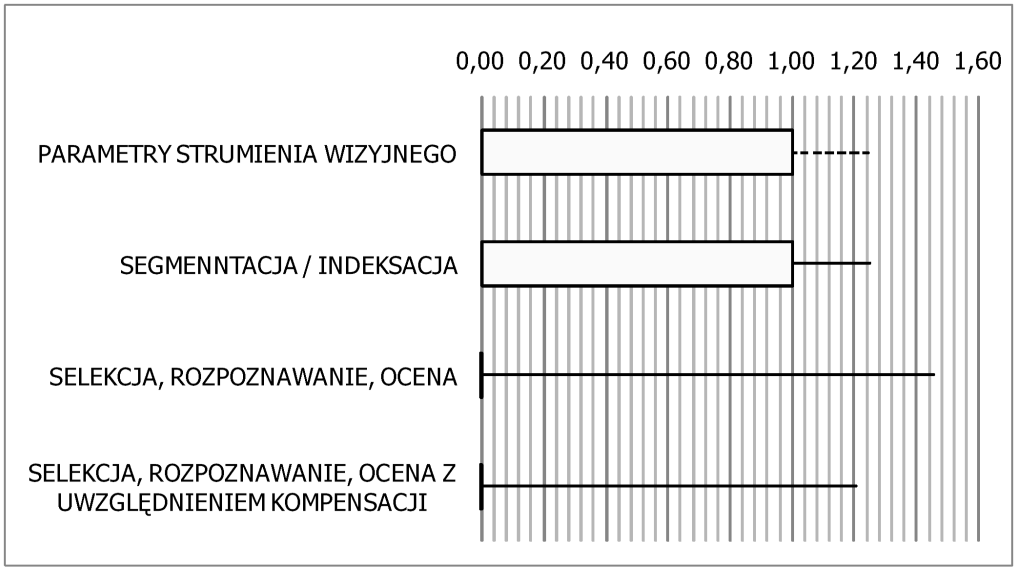
Rysunek 6.25: Czasy wykonania w sekwencyjnym torze wizyjnym.

Rezultaty zaprezentowane na wykresie 6.26 dotyczą strumieniowego systemu po zastosowaniu równoległości danych i operacji na wysokim poziomie granulacji. Na łącznym zestawieniu czasu wykonania widać, że rozstęp całkowity utrzymuje się nadal na poziomie $v \approx 0.5$. Wartość ta znacznie przekracza zapas czasowy modelowego strumienia wizyjnego opisanego w rozdziale 6.1 ponieważ $V = 0.1873$.

Analiza relacji pomiędzy liczbą obiektów zidentyfikowanych podczas indeksacji a potencjalną liczbą znaków do rozpoznawania pozwoliła ustalić dokładne szacowanie rozstępu czasu wykonania na akceptowalnym poziomie $v = 0.1718 < V$. Wykresy na rysunku 6.27 pozwalają skonfrontować zapas czasowy strumienia wizyjnego (oznaczony linią przerywaną) z właściwościami poszczególnych elementów toru wizyjnego. W pierwszej kolejności jest to segmentacja z indeksacją i pomiarem cech, które charakteryzują się małym rozstępem. Kolejny zbiór operacji to selekcja skalowanie, rozpoznawanie i ocena. Wybór ten wynika z faktu, że operacje te zostały zestawione w jeden potok a linia podziału przebiega przez bufory ramki wprowadzone przy zastosowaniu potoku o niskim poziomie granulacji (rysunek 6.20).



Rysunek 6.26: Czasy wykonania w zrównoleglonym torze wizyjnym.



Rysunek 6.27: Kompensacja czasów wykonania w potokowej realizacji toru wizyjnego.

Weryfikacja rozstępu czasu wykonania operacji na wykresie 6.27 odnosi się do czasu transmisji jednej ramki obrazu a nie do całkowitego czasu propagacji sygnału przez tor wizyjny jak na rysunkach 6.25 i 6.26. Jest to efekt zastosowania gruboziarnistego potoku. Obie grupy operacji są wówczas wykonywane równocześnie na różnych ramkach obrazu. O realizowalności strumieniowego toru wizyjnego decyduje zatem opóźnienie transportowe każdego segmentu toru wizyjnego oddzielnie. Można w tym zjawisku zauważyć pozytywny skutek zrównoleglenia na niskim poziomie granulacji. Efektem ubocznym jest wzrost opóźnienia transportowego liczonego w ramkach obrazu. Analizując jednak diagram 6.20 i wartości na powyższych wykresach widać jednak że parametry te są ostatecznie zbliżone co do wartości.

W oparciu o przedstawione dane można jednoznacznie stwierdzić, że integracja wielu zrównoleglonych operacji pozwala na przepływową realizację toru wizyjnego złożonego z algorytmów o czasie wykonania zależnym od treści strumienia wizyjnego.

Rozdział 7

Implementacja operacji wizyjnych o kontekście temporalnym

W rozdziale zaprezentowano etapy zrównoleglania systemu wideodetekcji w ruchu drogowym. Opisane algorytmy estymacji tła oraz detekcji pojazdów w obszarach detekcji wykorzystują temporalny rodzaj kontekstu. Przedstawiona metoda bilansowania równoległości instrukcji i równoległości danych umożliwiła realizację złożonego systemu wideodetekcji przy istotnych ograniczeniach zasobów pamięciowych platformy rekonfigurowalnej. Uzupełnienie rozdziału stanowi dodatek C.2 zawierający przykładowy kod źródłowy bufora kontekstu temporalnego.

7.1 Zasoby pamięciowe w kontekstowych operacjach wizyjnych

W analizie obrazu opisanej w rozdziale 6 wykorzystany jest jedynie kontekst przestrzenny: lokalny i globalny. Przedstawione tam sposoby zrównoleglenia algorytmów w celu wykorzystania w systemie strumieniowym dają dobre rezultaty dzięki zastosowaniu wysoko-zrównoleglonych struktur pamięciowych (rejestrów, linii opóźniających i pamięci dwuportowych) wewnątrz układu scalonego. Struktura danych w strumieniu wizyjnym (organizacja pikseli obrazu w ramki) i ustalony rozmiar kontekstu przestrzennego wyznaczają rozmiary linii opóźniających dla regularnych operacji kontekstowych (7.1).

$$Mem_k = (k-1)N \tag{7.1}$$

gdzie:

Mem_k – liczba komórek pamięci,

k – wysokość okna kontekstu,

N – szerokość obrazu liczona w pikselach.

Dla okna o rozmiarze 3×3 i szerokości obrazu 512 kolumn ilość komórek pamięci wynosi 1024. Wartość ta w niewielkim stopniu konsumuje dostępne zasoby pamięciowe współczesnych układów rekonfigurowalnych. W analizie obrazów wykorzystywane są jednak algorytmy, których zapotrzebowanie na zasoby pamięciowe jest znacznie większe i nie ulega redukcji pomimo udoskonaleniom i zastosowaniu zrównoleglenia. Jako przykład może posłużyć algorytm estymacji tła wykorzystujący analizę wielomodalną [91] lub algorytmy heurystyczne [72] stosowane w systemach wideodetekcji (monitoringu) ruchu drogowego.

Postać danych i parametrów niezbędnych do generacji tła, strukturalnie są zgodne z formatem ramki obrazu. W tabeli 7.1 zestawiono jednostkowe zapotrzebowanie na pamięć dla operacji przetwarzania obrazów o różnym rodzaju kontekstu.

Tabela 7.1: Rodzaj kontekstu a zapotrzebowanie na zasoby pamięciowe.

Rodzaj kontekstu	Rozmiar kontekstu	Zapotrzebowanie na komórki pamięci w bitach
bezkontekstowy	1 piksel	8
lokalny przestrzenny	3×3 piksele	8 192
temporalny	2 ramki	2 097 152

Ilość danych stanowi istotną przeszkodę nie tylko w realizacji sprzętowej takich operacji, ale również przysparza trudności w implementacji programowej na współczesnych komputerach klasy PC, pomimo ich dużej zdolności obliczeniowej i przepustowości magistral.

W najprostszym przypadku, do wyznaczenia tła lub ruchu w całym obszarze ośmiobitowego obrazu o rozmiarze 512×512 potrzeba 2048Kb pamięci. Wartość ta odpowiada maksymalnej pojemności pamięci blokowych największych układów rekonfigurowalnych przeznaczonych do przemysłowych zastosowań [106]. Wykorzystany podczas prac projektowych i testów układ scalony XC2V6000 [105] jest w stanie pomieścić tylko jeden bufor kontekstu temporalnego.

7.2 Algorytm wideodetekcji ruchu drogowego

Jednym z podstawowych elementów techniki wideodetekcji w ruchu drogowym jest poprawnie wyznaczone tło. Stanowi ono referencyjny obraz w oparciu o który przy zastosowaniu metody obrazów różnicowych[30][51] można zlokalizować i zliczyć ruchome i nieruchome pojazdy znajdujące się na pasach ruchu. Automatyczne uzyskanie wiarygodnego obrazu obszarów jezdni zazwyczaj jest jednak trudne ze względu na zmiany warunków oświetleniowych spowodowane warunkami pogodowymi, cyklem pór dnia i pór roku a w szczególności dużym natężeniem ruchu pojazdów.

W Laboratorium Biocybernetyki AGH od szeregu lat prowadzone są badania dotyczące analizy ruchu wideodetekcji i analizy drogowego [71][72][73]. Zbudowany w tym celu system wizyjny składa się z kamery zamontowanej na ruchomej głowicy oraz dwukierunkowego łącza transmisyjnego służącego do wysyłania obrazu z kamery do stanowiska badawczego. Kanał zwrotny służy do sterowania parametrami akwizycji oraz siłownikami mechanicznego systemu pozycjonowania. Fotografie systemu akwizycji sygnału wizyjnego ze skrzyżowania przedstawiono na rysunku 7.1.

Rysunek 7.1: System akwizycji sygnału wizyjnego ze skrzyżowania.

W pracach [72][73] opisano programową implementację heurystycznego algorytmu estymacji tła do analizy ruchu drogowego na platformie wbudowanej PC104. Jego zasadniczą zaletą jest niewrażliwość na nieruchome samochody oczekujące na zielone światło lub stojące w korku. Estymacja tła wykonywana jest dla obszarów określonych liniami, które symulują pętle indukcyjne wykorzystywane alternatywnie jako detektory pojazdów.

Idea ta została rozwinięta w pracach opisanych w publikacjach [30][31]. Autor niniejszej rozprawy, który brał udział w tychże badaniach, przeprowadził analizę heurystycznego algorytmu generacji tła [72] pod kątem możliwości realizacji w systemie przepływowym zawierającym cały tor wizyjny: akwizycję obrazu, estymację tła i detekcję pojazdów ruchomych i nieruchomych. Założono przy tym, że obszary linii zostaną zastąpione obszarami definiowanymi przez użytkownika bez ograniczenia ich kształtu i powierzchni, co miało miejsce w przypadku rozwiązania programowego. W opisanych algorytmach analizy ruchu drogowego określono dwa podstawowe sygnały przypisane do zdefiniowanych obszarów pasów ruchu drogowego:

- VP - obecność nieruchomego pojazdu,
- VM - obecność ruchomego pojazdu.

W referencyjnym (programowym) wideodetektorze sygnały te są wyznaczane dla każdego zbioru linii symulujących pętle indukcyjne zamontowane pod nawierzchnią drogi. Rysunek 7.2 przedstawia przykładową topologię pętli indukcyjnych w obszarze drogi. Idea przetwarzania wybranych pikseli zamiast całego obrazu została przyjęta również w sprzętowej realizacji. Założono jednak, że będzie możliwe wykorzystanie obszarów detekcji o bardziej złożonych kształtach do wykrywania ruchu i obecności (rysunek 7.2b).

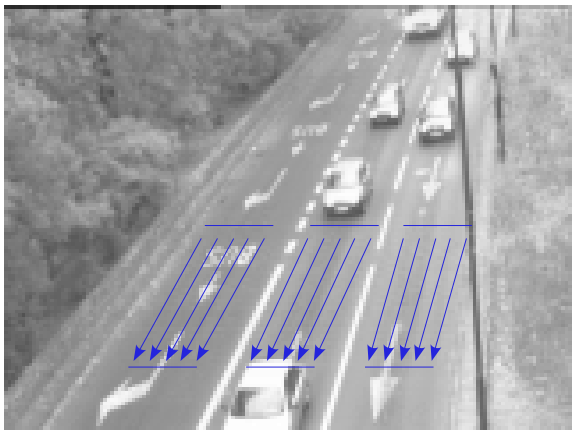
Sprzętowy system wideodetekcji, podobnie jak programowy, umożliwia dostosowanie obszarów detekcji do topologii drogi oraz adaptuje się do zmiennych warunków oświetleniowych, aby wyznaczyć sygnały ruchu i obecności. W celu spełnienia tych wymogów został opracowany następujący scenariusz przetwarzania obrazu:

1. Kolejne ramki obrazu są wykorzystane do wyliczenia tła w niezależnych obszarach detekcji. Tło jest zdefiniowane jako estymacja obszaru nawierzchni drogi bez uwzględnienia wartości pikseli odpowiadających nieruchomym i przemieszczającym się samochodom.

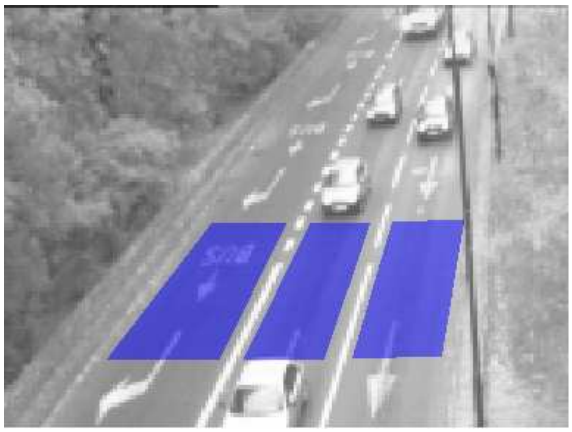
2. Sygnał obecności w danym obszarze detekcji wyznaczany jest poprzez progowanie sumy wartości bezwzględnych różnic pikseli SAD (ang. *Sum of Absolute Differences*) pomiędzy bieżącym obrazu a tłem.

3. Sygnał ruchu w danym obszarze detekcji wyznaczany jest poprzez progowanie sumy wartości SAD dla dwóch kolejnych obrazów (bieżącego i zapamiętanego w buforze).

W literaturze można spotkać wiele metod estymacji tła dla celów wideodetekcji ruchu drogowego czy monitoringu. Podejście bazujące na modelach statystycznych zostało opisane w pracach [23][91]. Autorzy tych prac proponują przypisanie każdemu pikselowi obrazu kilku procesów gaussowskich (ang. *Mixture of*



(a) symulowane pętle indukcyjne



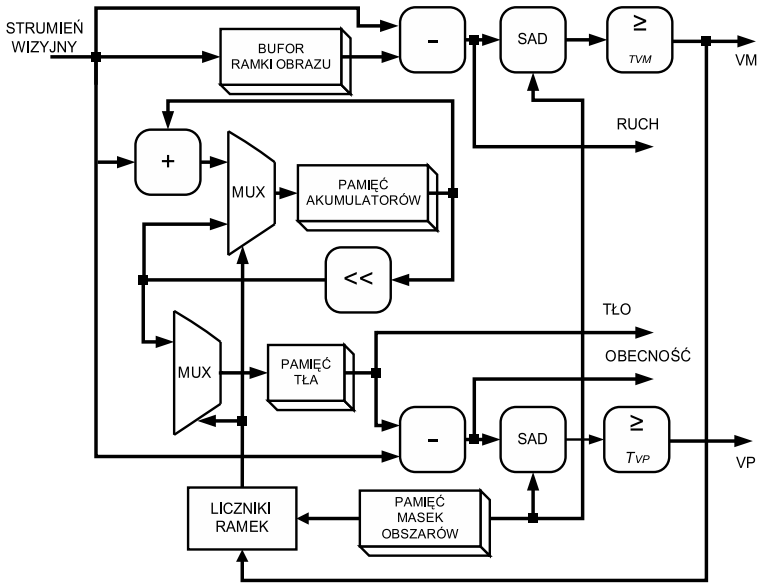
(b) topologia obszarów detekcji

Rysunek 7.2: Topologia pętli indukcyjnych i obszarów wideodetekcji.

Gaussians). Metoda ta jest użyteczna do detekcji tła w zmiennych warunkach oświetleniowych dla wolno-przemieszczających się obiektów i detekcji cieni. Implementacja tego algorytmu w czasie rzeczywistym przysparza jednak wielu problemów. Rozwiązanie tej trudności przedstawili autorzy pracy [55] proponując wykorzystanie sprzętowego akceleratora do segmentacji tła. Na potrzeby niniejszej pracy autor rozprawy wykorzystał jednak heurystyczny algorytm opisany schematem przepływu strumieni jak na rysunku 7.3.

Podczas analizy sekwencji wizyjnych ruchu drogowego autorzy [1][23][72] zaobserwowali, że statystyczny rozkład wartości pikseli w obszarze ruchu jest jednomodalny dla małego natężenia ruchu. Kiedy intensywność ruchu pojazdów wzrasta w rozkładzie wartości pikseli pojawiają się dodatkowe maksima. W pracy [71] stwierdzono, że średnia wartość piksela w czasie jest dobrym przybliżeniem statycznego tła (tj. nawierzchni drogi bez pojazdów) pod warunkiem, że rozkład statystyczny wartości piksela jest jednomodalny.

Piksele odpowiadające pojazdom nieruchomym nie powinny być zatem bra-



Rysunek 7.3: Schemat przepływu strumieni danych w algorytmie wideodetekcji

ne pod uwagę przy wyznaczaniu tła. Wpływ obrazu pojazdów na tło mógłby powodować znacznie większe błędy niż samo przybliżenie wartości oczekiwanej unimodalnego rozkładu średnią wartością pikseli w czasie. Dlatego wartość średnią należy aktualizować jedynie dla tych ramek obrazów, gdzie wykryto ruch pojazdów (sygnał VM). W rzeczywistych warunkach trudno jest zdefiniować globalne kryterium ruchu, więc ruch wyznaczany jest oddzielnie dla każdego obszaru detekcji, aby wyzwolić aktualizację wszystkich pikseli w danym obszarze. W ten sposób brane są pod uwagę nie tylko modele statystyczne zmian stopni szarości pikseli, ale również topologiczne własności pasów ruchu i skrzyżowań. Na potrzeby sprzętowej realizacji zbudowano model matematyczny podsystemu generacji tła w postaci równań (7.2)(7.3) przedstawiony w pracy [50].

$$\left.\begin{array}{l} A^i_{(x,y)} = A^{i-1}_{(x,y)} + P^i_{(x,y)} \\ B^i_{(x,y)} = B^{i-1}_{(x,y)} \\ i = i+1 \end{array}\right\} \quad \text{jeśli} \quad i < 2^I \wedge VM^{i-1} \tag{7.2}$$

$$\left.\begin{array}{l} A^i_{(x,y)} = A^{i-1}_{(x,y)} - A^{n-1}_{(x,y)}/2^J \\ B^i_{(x,y)} = A^i_{(x,y)}/2^I \\ i = 2^{I-J} \end{array}\right\} \quad \text{jeśli} \quad i = 2^I \wedge VM^{i-1} \tag{7.3}$$

gdzie:

(x, y)	–	współrzędne piksela dla danego obszaru detekcji,
i	–	numer ramki obrazu,
2^I	–	okres akumulacji liczony w ramkach,
2^J	–	okres normalizacji liczony w ramkach,
$P^i_{(x,y)}$	–	piksel źródłowy dla ramki i,
$A^i_{(x,y)}$	–	wartość akumulatora dla ramki i,
$B^i_{(x,y)}$	–	wartość tła dla ramki i,
VM^i	–	sygnał ruchu dla ramki i,
VP^i	–	sygnał obecności dla ramki i.

Równania (7.2) i (7.3) opisują obliczenia systemu dla każdego piksela w poszczególnych ramkach strumienia wizyjnego. Reguła aktualizacji akumulatora zawiera okresową normalizację, która zapewnia utrzymanie wartości w ustalonym przedziale poprawnie reprezentowanych wartości $A \in \langle 0, 2^{8+I} - 1 \rangle$ dla ośmiobitowej reprezentacji pikseli monochromatycznego sygnału wizyjnego. Wynikowe tło jest wyznaczane co 2^J ramek w których wykryto ruch. Przez czas gromadzenia danych, tło pozostaje niezmienne. Binarne wartości sygnałów VM i VP wyznaczane są dla każdej ramki obrazu i dla każdego obszaru detekcji oddzielnie zgodnie z równaniami (7.4) i (7.5). Jakkolwiek tło wyznaczane jest automatycznie w oparciu o treść strumienia wizyjnego i zwrotny sygnał ruchu, to należy zaznaczyć, że wartości progów T_{VP} i T_{VM} do wyznaczania ruchu i obecności pojazdów należy uważnie dobrać, aby odpowiadały rozmiarom zdefiniowanych obszarów detekcji.

$$VM^i = \begin{cases} 1 & \text{jeśli } \sum \left| P^i_{(x,y)} - P^{i-1}_{(x,y)} \right| > T_{VM} \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{7.4}$$

$$VP^i = \begin{cases} 1 & \text{jeśli } \sum \left| P^i_{(x,y)} - B^{i-1}_{(x,y)} \right| > T_{VP} \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{7.5}$$

gdzie:

T_{VM}	–	próg binaryzacji dla sygnału ruchu,
T_{VP}	–	próg binaryzacji dla sygnału obecności.

7.3 Strumieniowa realizacja algorytmu wideodetekcji

Sprzętowa architektura zaprojektowana i zaimplementowana na potrzeby opisanego w podrozdziale 7.2 algorytmu wideodetekcji jest przedstawiona na rysunku 7.3. Diagram zawiera bufory obrazu, operatory arytmetyczne, elementy przełączające i sieć połączeń. Ten sposób prezentacji jest wynikiem syntezy równań (7.2), (7.3), (7.4), (7.5). Opis przepływu danych bezpośrednio koresponduje również ze sposobem mapowania opisu funkcjonalnego do struktury sprzętowego systemu

obliczeniowego (HandelC, FPGA). Przedstawiony opis funkcjonalny pomija jednak szereg elementów istotnych ze względu na zarządzanie strumieniami danych i synchronizację współbieżnych procesów. Zadania te zostały oddelegowane do obsługi przez rdzeń biblioteki PixelStreams*[18], która została użyta do realizacji przedstawionego algorytmu.

7.3.1 Estymacja zapotrzebowania na zasoby pamięciowe

Na podstawie równań (7.2)-(7.5) i diagramu przedstawionego na rysunku 7.3 można określić zasoby pamięciowe wymagane do wideodetekcji. Ze schematu wynika, że strumieniowa implementacja sprzętowa wymaga czterech buforów obrazu. Niezbędne są również dodatkowe dwa bufory do akwizycji i wizualizacji. Sumarycznie potrzeba sześć banków pamięci o określonej rozdzielczości bitowej pikseli:

(a) bufor opóźnionej ramki obrazu do detekcji ruchu - 8 bitów,

(b) pamięć akumulatorów do wyznaczania chwilowej średniej - 16 bitów,

(c) pamięć wyznaczonego tła - 8 bitów,

(d) pamięć masek obszarów detekcji definiowanych przez operatora - 8 bitów,

(e) bufor ramki do usuwania przeplotu - 8 bitów,

(f) bufor do prezentacji wybranych strumieni wizyjnych na monitorze - 8 bitów.

Dwie ostatnie pamięci (e) i (f) nie wchodzą w skład procesora estymacji tła jednak są konieczne do akwizycji sygnału wizyjnego i jego prezentacji w celach uruchomieniowych i kontrolnych.

Przyjęto, że obraz w strumieniu wizyjnym ma rozmiar 512×512. Całkowita liczba komórek pamięci niezbędna do realizacji wideodetektora, zgodnie z powyższymi danymi wynosi $Mem_{VD} = 14366$ Kbitów. Szczegółowe zestawienie zapotrzebowania na pamięć dla różnych operacji na obrazie przedstawiono w tabeli 7.2. Dane te odnoszą się bezpośrednio do zawartości tabeli 7.1. Tabela 7.3 pokazuje zasoby pamięci blokowych dostępne w największych układach reprogramowalnych osiągalnych w czasie implementacji wideodetektora.

Arytmetyczne operatory odejmowania i dodawania wymagają alokacji zasobów obliczeniowych. Dzielenie wykorzystane w normalizacji - równanie (7.3) - zrealizowano jako przesunięcie bitowe bez znaczącego wzrostu zużycia zasobów obliczeniowych układu rekonfigurowalnego.

*PixelStreams jest rozszerzeniem języka HandelC dedykowanym do szybkiego prototypowania i implementacji aplikacji przetwarzania i analizy obrazów w cyfrowych systemach sprzętowych, w tym również rekonfigurowalnych.

Tabela 7.2: Zapotrzebowanie na zasoby pamięciowe w wideodetektorze.

Rodzaj kontekstu operacji	Zapotrzebowanie na pamięć Kb
bezkontekstowa	$\backsim 0$
kontekst przestrzenny 3×3	8
kontekst temporalny 2	2 048
cały wideodetektor	14 366

Tabela 7.3: Zasoby pamięciowe w dostępnych układach reprogramowalnych.

Rodzina układów rekonfigurowalnych	Rozmiar pamięci blokowej Kb
Spartan 3	2 268
Virtex II	3 024
Virtex 4	9 936

Zależności czasowe i kolejność operacji wykonywanych w wideodetektorze zostały szczegółowo przeanalizowane w celu sprawdzenia wykonalności algorytmu wideodetekcji. Idea strumieniowego przetwarzania zakłada, że dla każdego piksela wejściowego generowany jest jeden piksel wynikowy*. Aby spełnić ten warunek i umożliwić działanie w czasie rzeczywistym, wszystkie obliczenia opisane równaniami (7.2) - (7.5) muszą być wykonane współbieżnie w jednym cyklu zegara. Prowadzi to do konfliktu na magistralach pamięci które musiałyby by być jednocześnie zapisywane i odczytywane w tym samym cyklu zegarowym (por. rysunek 7.4). Bufory pamięciowe nie mogą być zatem alokowane we wspólnej przestrzeni adresowej. Niezależne (na poziomie elektronicznym) banki pamięci powinny być dołączone do układu reprogramowalnego aby umożliwić współbieżne odwołania do danych w poszczególnych buforach. Dotyczy to również dodatkowych buforów do usuwania przeplotu oraz wizualizacji. Łączna liczba banków pamięci niezbędnych do realizacji systemu wideodetekcji wynosi 6, podczas gdy dostępna do implementacji platforma RC300 oferuje jedynie 4. Ze względów praktycznych zwiększanie liczby pamięci dołączonych do układu scalonego jest niekorzystne. Podnosi to koszt urządzenia, zużycie energii oraz komplikuje proces projektowania platformy jak i wykonania. Uzasadnione jest więc poszukiwanie takich rozwiązań, które pozwolą efektywnie wykorzystać dostępne zasoby tym bardziej, że rozbudowa systemu detekcji o dodatkowe funkcjonalności może zwiększyć zapotrzebowania pamięciowe.

*Nie dotyczy to jednak sygnałów VM i VP, które są wyznaczane po zakończeniu ramki obrazu dla każdego pola detekcji oddzielnie. Mamy tu do czynienia z przestrzennym kontekstem globalnym.

7.3.2 Zrównoleglenie przy ograniczonych zasobach pamięciowych

Autor rozprawy zaproponował cztery wzajemnie uzupełniające się rozwiązania, które umożliwiły realizację systemu wideodetekcji na dostępnej platformie obliczeniowej przy istniejących ograniczeniach pamięciowych:

(a) zastąpienie liniowej przestrzeni adresowej przez współbieżnie działające banki pamięci,

(b) łączenie danych z różnych buforów w słowo danych jednego banku pamięci (tutaj: scalenie bufora tła i maski obszaru detekcji),

(c) reorganizacja danych w pamięci akumulatorów tj. umieszczenie akumulatorów odpowiadających dwóm kolejnym pikselom w jednym słowie pamięci,

(d) podział potokowej fazy na dwa naprzemiennie wykonywane etapy:

- etap odczytu: odczyt danych z pamięci i wykonanie obliczeń,
- etap zapisu: zapis wyników obliczeń do pamięci.

Odwołując się do klasyfikacji przedstawionej w rozdziale 4.2 reorganizacja pamięci przedstawiona w punkcie (a) ma na celu zwiększenie równoległości danych co przyczynia się do wzrostu przyspieszenia poprzez współbieżne wykonanie poszczególnych operacji. Celem modyfikacji (b) jest zmniejszenie liczby wykorzystywanych banków pamięci, co jak wspomniano w poprzednim rozdziale, jest konieczne ze względu na ograniczenie liczby pamięci dostępnych na platformie docelowej. Dalsze zwiększenie równoległości danych w modyfikacji (c) pozwoli uzyskać zamierzoną przepustowość $P_C = 1$. Nie jest to jednak rozwiązanie wystarczające aby uniknąć konfliktu odwołań do pamięci w klasycznym potoku przedstawionym na rysunku 7.4. Dopiero odpowiednie uszeregowanie operacji w potoku (d) umożliwi płynne przetwarzanie pikseli w strumieniu wizyjnym.

7.3.3 Architektura potokowa a kontekst temporalny

operacje \ cykl	0	1	2	3	4	...
	R_0	R_1	R_2	R_3	R_4	...
		O_0	O_1	O_2	O_3	...
			W_0	W_1	W_2	...

Rysunek 7.4: Próba zastosowania potoku - konflikt w odwołaniach do pamięci.

Analizę i ocenę efektów zrównoleglenia w krokach (c) i (d) ułatwi wprowadzenie oznaczeń dla współbieżnie wykonywanych operacji. Przyjęto następującą

notację (7.6) opisu równoległości DLP, kolejno dla operacji na jednym kwancie danych i na dwóch kwantach danych - zgodnie z propozycją (c):

$$F_p \quad F_{p,q} \tag{7.6}$$

gdzie:

F – rodzaj operacji,

p – indeks piksela dla którego wykonywana jest operacja F,

p, q – indeksy dwóch kolejnych pikseli dla których wykonywana jest operacja F.

Zakładając taką notację i uwzględniając wzór (4.1) można zapisać równoważność (7.7), która oznacza współbieżne wykonanie takiej samej operacji F na dwóch kwantach danych: p i q. Ponieważ w rozwiązaniu (c) przyjęto że dwa piksele składowane są w jednym słowie pamięci, zawsze będzie zachodzić równość $q = p + 1$.

$$F_{p,q} \equiv F_p || F_q \tag{7.7}$$

Na wysokim poziomie granulacji, operacje F wykonywane w każdym podsystemie wideodetektora można sklasyfikować ze względu na sposób korzystania z pamięci:

R odczyt danych z pamięci,

O inne operacje: obliczenia na pikselach, generacja adresów odczytu i zapisu,

W zapis danych do pamięci.

Podział ten jest istotny w przedstawionej analizie ponieważ przetwarzanie danych w strumieniu wizyjnym spowodowane jest ograniczeniami zasobów pamięciowych. Na rysunku 7.4 przedstawiono potok zestawiony z wymienionych operacji. Wprawdzie z jego struktury wynika przepustowość $P_c = 1$ jednak już z samej treści powyższej listy operacji widać, że nie jest możliwa jego implementacja ponieważ jednoczesny zapis **W** i odczyt **R** na masowej pamięci jednoportowej jest wykluczony.

Aby zachować czytelność wywodu wszystkie inne operacje i obliczenia zostały tutaj oznaczone zbiorczo jako **O**. W rzeczywistości, na operację **O** może składać się szereg elementarnych obliczeń wykonywanych równolegle lub zorganizowanych w potok. Uproszczenie to nie wpływa jednak na poniższą analizę przy założeniu, że nie wiążą się z nimi podobne konflikty jak przy zapisie i odczycie.

7.3.4 Bilans równoległości danych i równoległości operacji

W potoku przedstawionym na rysunku 7.4 liczba współbieżnych operacji przypadających w cyklu n na krytyczny zasób jakim jest tutaj pamięć wynosi $Q_{OLP} = 2$ ponieważ operacje na pamięci są następujące $O_n^{Mem} \equiv R_n||W_{n-2}$. Ze względów technologicznych taki stopień równoległości jest niemożliwy do realizacji. Obserwacja ta jest kluczowa, ponieważ jednym z możliwych sposobów zrównoleglenia jest taka transformacja potoku, aby te konflikty nie występowały. Istotny jest tutaj fakt, że przyczyną stopnia równoległości Q_{OLP} większego od 1 jest równoległość na poziomie operacji (OLP) a nie danych (DLP). Autor rozprawy zauważył, że zamiana rodzaju równoległości może stanowić rozwiązanie problemu dostępu do danych w pamięci. Co więcej, można tym samym stwierdzić, że do zasobu pamięciowego można przypisać charakterystyczny parametr $Q_{OLP}^{Mem} = 1$ który ogranicza współbieżne wykonanie operacji.

Tabela 7.4: Transformacja równoległości w wideodetektorze.

	Q_{OLP}	Q_{DLP}
stan wyjściowy	2	1
stan końcowy	1	2

gdzie:

Q_{DLP} – aktualny stopień równoległości danych.

Q_{OLP} – aktualny stopień równoległości operacji.

Q_{DLP}^{Mem} – graniczny stopień równoległości danych pamięci, $Q_{DLP}^{Mem} = 2$,

Q_{OLP}^{Mem} – graniczny stopień równoległości operacji pamięci, $Q_{OLP}^{Mem} = 1$.

Kontynuując ten tok rozumowania wprowadzono kolejny parametr równoległości Q_{DLP} oraz cechę pamięci Q_{DLP}^{Mem} wynikające z rodzaju równoległości typu DLP. Ograniczenie $Q_{OLP}^{Mem} = 1$ wynika bezpośrednio z konstrukcji jednoportowej pamięci: nie jest możliwy jednoczesny zapis ani odczyt oraz zapis lub odczyt danych z dwóch lub więcej różnych adresów. Wartość graniczna Q_{DLP}^{Mem} jest równa liczbie kwantów danych składowanych w jednym słowie pamięci. Pośrednio liczba ta wynika z rozmiaru szyny danych pamięci oraz liczby bitów przypadających na jeden piksel obrazu. Analizę ilościową opisującą stan wyjściowy oraz oczekiwane wyniki uzyskane po transformacji potoku, z uwzględnieniem rozwiązań (c) i (d), przy zadanych ograniczeniach pamięci przedstawiono w tabeli 7.5.

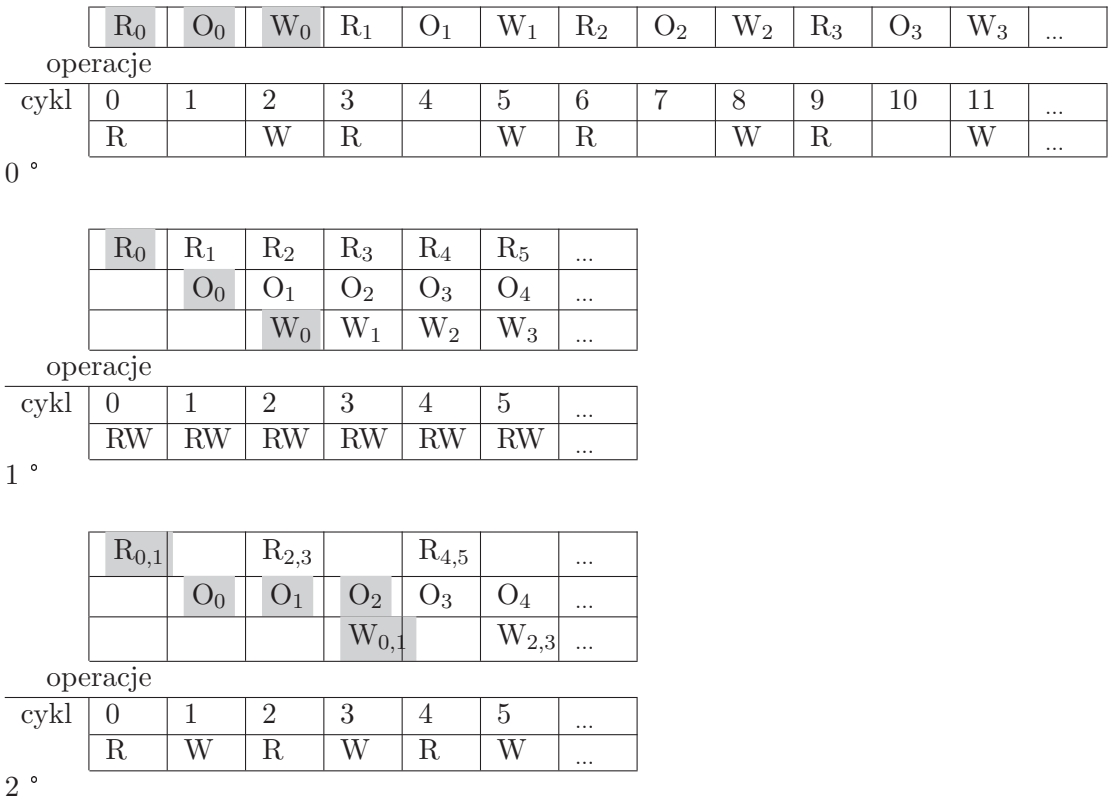
7.3.5 Transformacja architektury potokowej

W celu zobrazowania efektów zastosowania rozwiązań (c) i (d) zdefiniowano następujące etapy transformacji potoku:

0° sekwencyjne wykonanie operacji,

1° zestawienie potoku z pominięciem ograniczeń zasobów pamięciowych,

2° zwiększenie równoległości DLP i reorganizacja cykli odwołań do pamięci.



Rysunek 7.5: Transformacja potoku przy ograniczeniach pamięciowych.

Wymienione etapy transformacji potoku zostały pokazane na rysunku 7.5. Wyniki analizy poszczególnych etapów przedstawiono w tabeli 7.5. Zarówno przyspieszenie, jak i przepustowość przyjmują takie same wartości w etapie 2° jak w etapie 1° chociaż nastąpiła zmiana w stopniach i rodzajach równoległości, a efektywność E spadła do 0.6 w ostatnim etapie. Uzyskano zamierzoną przepustowość $P_c = 1$. Opóźnienie transportowe zwiększyło się w ostatnim etapie do wartości $L = 3$. Wynik ten jest niejednoznaczny ponieważ opóźnienie L przyjmuje wartość 3 dla pikseli parzystych i 2 dla pikseli nieparzystych. Wobec powyższego, można stwierdzić, że ani przyspieszenie S ani efektywność E czy opóźnienie transportowe L_s nie są jedynymi wskaźnikami, które pozwalałyby na skuteczną ocenę kolejnych etapów strumieniowej realizacji algorytmów wizyjnych o kontekście temporalnym. Mimo to jednak strumieniowa implementacja wideodetektora stała się możliwa po

Tabela 7.5: Ocena zrównoleglenia podczas transformacji potoku.

	0°	0°- 1°	0°- 2°
prawo Amdahl'a (4.3)			
n	1	3	5
s	1	0	1/6
p	0	3	5/6
S_a	1	3	3
prawo Gustafson'a (4.5)			
s'	1	0	1/2
p'	0	3	1/2
S_g	1	3	3
przyspieszenie całkowite(4.2)			
T_1	3	3	6
T_n	3	1	2
S_m	1	3	3
efektywność (4.4)			
E		1.000	0.600
opóźnienie transportowe			
L	2	2	3
przepustowość			
P_c	1/3	1	1
równoległość DLP pamięci			
Q_{OLP}	1	2	1
równoległość OLP pamięci			
Q_{DLP}	1	1	2

wprowadzeniu zmian (c) i (d). Autor rozprawy proponuje zatem przyjąć stopień równoległości zasobów pamięciowych Q_{DLP}^{Mem} i Q_{OLP}^{Mem} jako kryterium wykonalności architektury potokowej w strumieniowej realizacji algorytmów o kontekście temporalnym. Łączenie dwóch kolejnych pikseli (i skojarzonych z nimi komórkami akumulatora A do wyznaczania tła) ze strumienia wizyjnego w jednym słowie pamięci pozwoliły osiągnąć zamierzoną wartość $Q_{DLP} = 2$. Dodatkowo, uszeregowanie operacji odczytu i zapisu zrównoleglonych danych do pamięci, pozwoliło na zmniejszenie równoległości operacji $Q_{OLP} = 2$, co przełożyło się na spadek efektywności o prawie połowę. Niska efektywność jest bezpośrednim skutkiem braku równowagi pomiędzy stopniem równoległości DLP a OLP, która wynika ze zidentyfikowanych ograniczeń. Przedstawiona analiza potwierdza zatem przydatność wprowadzonych wskaźników Q_{OLP} oraz Q_{DLP} w strumieniowej realizacji algorytmów wizyjnych o kontekście temporalnym.

7.3.6 Potokowa realizacja podsystemów wideodetektora

Szczegółowa analiza zrównoleglonego modelu sprzętowego wideodetektora obejmuje wszystkie jego podsystemy:

A podsystem identyfikacji obszarów detekcji,

B podsystem detekcji ruchu,

C podsystem generacji tła,

D podsystem detekcji obecności i aktualizacji tła.

Detekcja ruchu i detekcja obecności są operacjami, które łączą w sobie elementy globalnego kontekstu przestrzennego oraz kontekstu temporalnego. Do wyznaczenia rezultatu konieczna jest analiza pikseli z danego obszaru detekcji w bieżącym strumieniu obrazu oraz piksele poprzedniej ramki obrazu zgromadzonej w buforze ramki. W przypadku detekcji obecności jako drugie źródło temporalnych danych obrazu występuje bufor okresowo aktualizowanego tła obrazu. Dla każdego z podsystemów, sekwencja operacji związana z obsługą bieżącego piksela w strumieniu wizyjnym składa się z następujących operacji:

1 ustawienie adresu odczytu danych z pamięci,

2 odczyt danych z pamięci,

3 wykonanie obliczeń zgodnie ze wzorami (7.2)-(7.5),

4 ustawienie adresu zapisu danych z pamięci,

5 zapis danych do pamięci.

Wyszczególnienie operacji 1 i 4 jest konieczne ze względu na zastosowany rodzaj synchronicznej pamięci ZBT RAM. Ten typ pamięci, podobnie jak inne wykorzystywane we współczesnych systemach obliczeniowych, wymaga dwóch cykli na przeprowadzenie operacji zapisu czy odczytu. W pierwszym cyklu ustalany jest adres komórki, a w kolejnym wykonywana jest właściwa operacja na danych: odczyt lub zapis.

Zrównoleglenie podsystemu detekcji obecności

Podsystem **D**, który wykrywa obecność pojazdów w obszarach detekcji odpowiada również za okresową aktualizację tła wyznaczonego przez podsystem **C**. Działanie tego modułu nie wpływa na pozostałe podsystemy i nie będzie uwzględnianie w kolejnym paragrafie aby zachować jego przejrzystość. Moduł detekcji obecności wyłącznie operuje na jednym banku pamięci, w którym przechowywane jest znormalizowane tło. Pamięć ta jest odczytywana podczas detekcji obecności. Zapis danych do tego bufora następuje okresowo, gdy licznik aktualizacji

tła osiągnie górną wartość graniczną zgodnie z warunkiem (7.3). Zrównoleglenie detektora obecności przedstawione na rysunku 7.6 jest typowym przykładem zastosowania potokowej architektury przedstawionej już w rozdziale 6.5.1. Większość operacji na pamięci przedstawionych na rysunku 7.6 to odczyt danych.

	$D1_1$	$D1_2$	$D1_3$	$D1_4$	$D1_5$	$D1_6$	$D1_{...}$	$D1_n$	$D1_{n+1}$	...
		$D2_1$	$D2_2$	$D2_3$	$D2_4$	$D2_5$	$D2_{...}$	$D2_n$	$D2_{n+1}$	...
			$D3_1$	$D3_2$	$D3_3$	$D3_4$	$D3_{...}$	$D3_n$	$D3_{n+1}$	...
				$D4_1$	$D4_2$	$D4_3$	$D4_{...}$	$D4_n$	$D4_{n+1}$	...
					$D5_1$	$D5_2$	$D5_{...}$	$D5_n$	$D5_{n+1}$	...
operacje										
cykl	0	1	2	3	4	5	...	n	$n+1$	...
	R	R	R	R	R	R	R	W	R	...

Rysunek 7.6: Potokowa detekcja obecności.

W cyklu n wykonywana jest operacja zapisu piksela tła. Mimo to, nie występuje konflikt jednoczesnego zapisu i odczytu, ponieważ wartość aktualnie zapisywana do pamięci jest wykorzystana do obliczeń w bieżącym cyklu.

Etapy zrównoleglenia wideodetektora

Kluczowe etapy w przystosowaniu podsystemów **A**,**B** i **C** do implementacji strumieniowej są następujące:

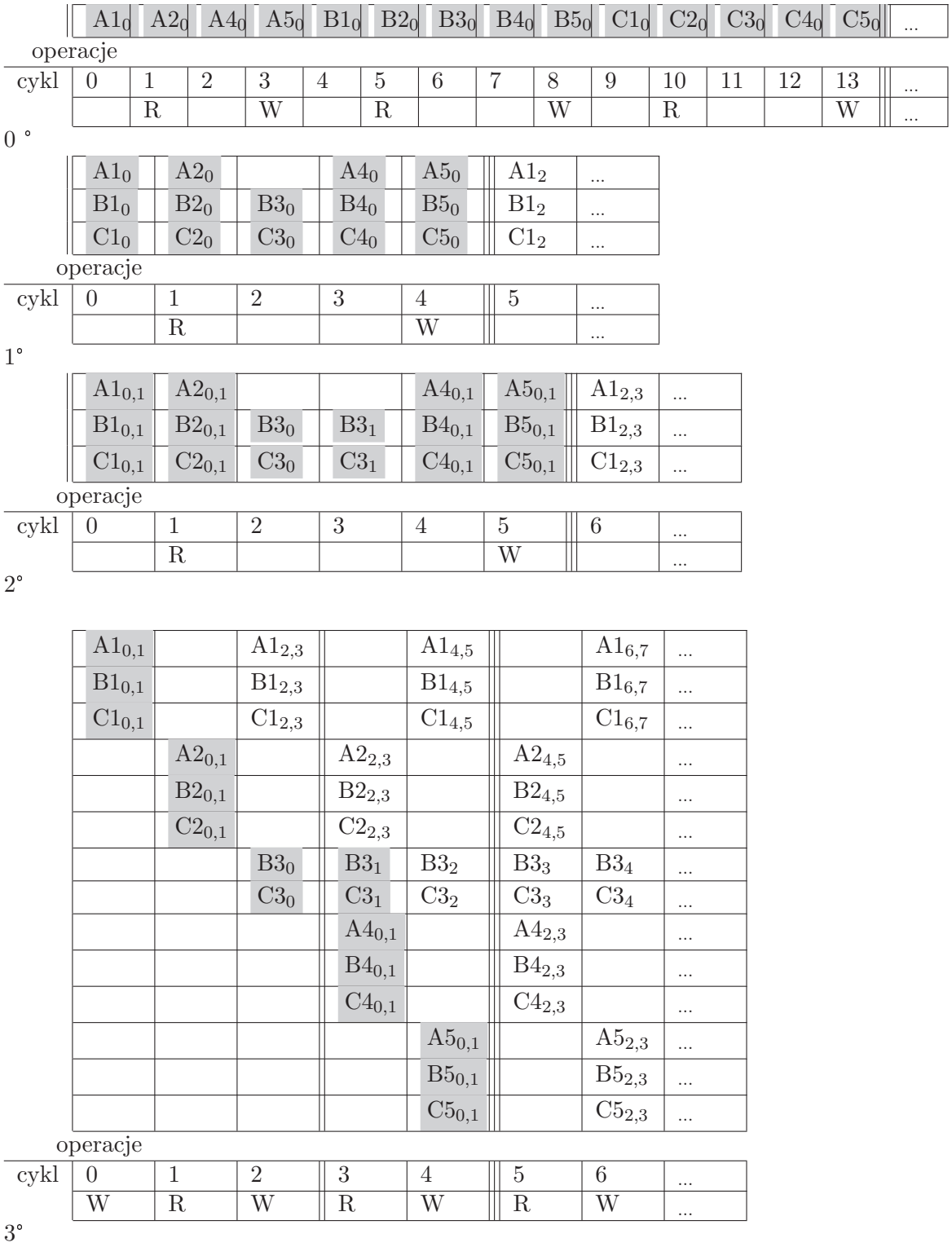
0° sekwencyjne obliczenia na danych we wspólnej, liniowej przestrzeni adresowej,

1° podział liniowej przestrzeni adresowej pamięci na niezależne bufory,

2° upakowanie dwóch kwantów danych w jednym słowie pamięci,

3° reorganizacja potoku.

Pominięto tutaj analizę potokowej realizacji detekcji obecności. Rezultaty zrównoleglenia dla podsystemu detekcji ruchu, aktualizacji tła oraz identyfikacji obszarów detekcji przestawiono w tabeli 7.6. Podstawowym celem przekształcenia zaprezentowanego na rysunku 7.7 było osiągnięcie przepustowości $P_C = 1$. Cel ten został zrealizowany w etapie 3°. W kroku 1° wykorzystano równoległość OLP każdego z podsystemów, uwarunkowaną współbieżnym dostępem do danych w niezależnych bankach pamięci. Niska przepustowość powoduje, że rozwiązanie to nie może być praktycznie wykorzystane. Stanowi jednak dobry punkt wyjściowy do zastosowania równoległości DLP, poprzez łączenie dwóch sąsiednich pikseli w jednym słowie pamięci (krok 2°). Nastąpił dalszy, aczkolwiek niewystarczający wzrost przyspieszenia i wprost proporcjonalne zwiększenie przepustowości. W ostatnim kroku 3° zorganizowano zrównoleglone operacje w potok. Uzyskano



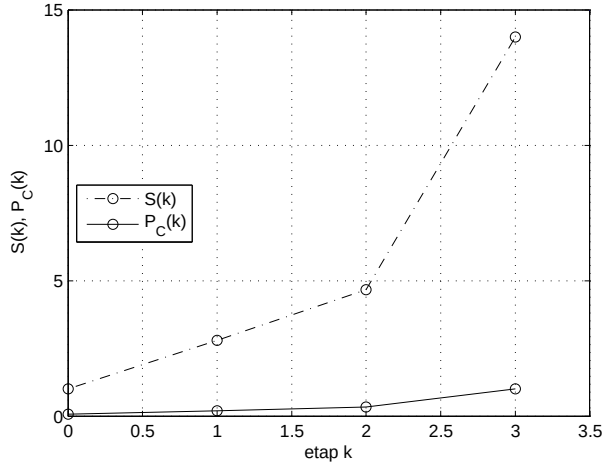
Rysunek 7.7: Zrównoleglenie wideodetektora przy ograniczeniach pamięciowych.

zadawalający efekt w postaci przepustowości $P_C = 1$ oraz przyspieszenia $S = 14$. W danych przedstawionych w tabeli 7.6 i na rysunku 7.8 widać wyraźnie, że przepustowość S i przyspieszenie P_C są do siebie wprost proporcjonalne, a stosunek przyspieszenia do przepustowości jest stały i wynosi $\frac{S(k)}{P_C(k)} = 14.0$. W ocenie realizowalności danego stopnia zrównoleglenia można więc korzystać wymiennie z przyspieszenia lub przepustowości (oczywiście po uwzględnieniu współczynnika proporcjonalności).

Postęp w osiąganiu docelowej przepustowości odzwierciedlony jest również we wzroście równoległości operacji Q_{OLP} potoku jaki i równoległości danych w pamięci Q_{DLP}. Współczynniki te wyznaczono, przyjmując wartości maksymalne dla pełnej sekwencji operacji ograniczonej na rysunku 7.7 podwójną linia pionową. W żadnym etapie nie uzyskano efektowności bliskiej wartości 1. Fakt ten nie dziwi ponieważ poza operacjami B3 i C3 wszystkie inne są wykonywane w jednym z dwóch cykli potoku 3°. Ich realna efektywność wynosi zatem 0.5 i jest to koszt poniesiony w związku z uniknięciem konfliktu dostępu do danych w jednoportowej pamięci.

Tabela 7.6: Ocena zrównoleglenia wideodetektora.

	0°	0°- 1°	1°- 2°	2°- 3°
prawo Amdahl'a (4.3)				
n	1	10	5	5
s	1	4/14	5/10	1/6
p	0	10/14	5/10	5/6
S_a	1	14/5	10/6	6/2
prawo Gustafson'a (4.5)				
s'	1	4/5	5/6	1/2
p'	0	1/5	1/6	1/2
S_g	1	14/5	10/6	3
efektywność (4.4)				
E		0.280	0.333	0.600
przyspieszenie całkowite(4.2)				
T_1	14	14	28	28
T_n	14	5	6	2
S_m	**1.00**	**2.80**	**4.67**	**14.00**
opóźnienie transportowe				
L	14	5	6	2
przepustowość				
P_c	0.071	0.200	0.333	1.000
równoległość OLP potoku				
Q_{OLP}	1	3	3	8
równoległość DLP pamięci				
Q_{DLP}	1	1	2	2



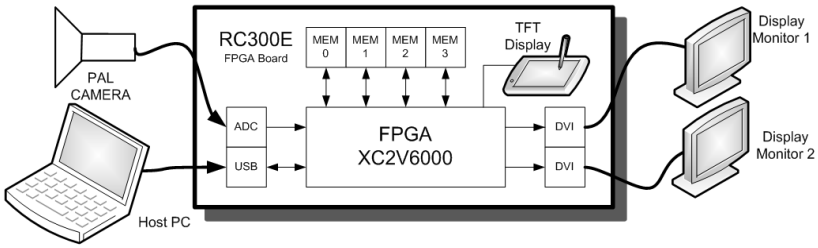
Rysunek 7.8: Przyspieszenie i przepustowość zrównoleglonego wideodetektora.

7.4 Implementacja wideodetektora

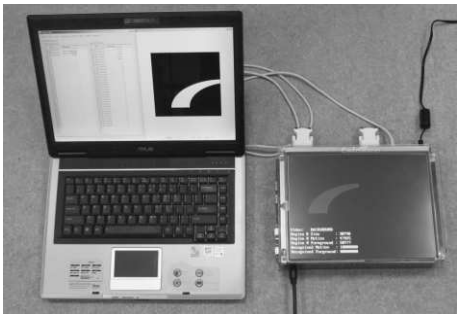
Dzięki zaproponowanej metodzie efektywnego zarządzania pamięcią i szeregowania operacji w systemie strumieniowym, możliwe było uruchomienie aplikacji wideodetektora na sprzętowej platformie wyposażonej jedynie w 4 banki pamięci. Prototyp sprzętowego wideodetektora pobiera sygnał wizyjny PAL z kamery zamontowanej na skrzyżowaniu lub urządzenia odtwarzającego nagraną sekwencję. Dla każdej ramki obrazu wyznaczany binarny sygnał obecności VP i ruchu VM w poszczególnych obszarach detekcji. Sygnały obecności widoczne są na konsoli graficznej karty RC300 (patrz rys. 7.9). Sprzętowy wideodetektor obsługuje strumień obrazów o rozmiarze 512×512 i został zsyntezowany dla $I = 8$ i $J = 5$ (por. równania (7.2), (7.3)). Maksymalna możliwa wartość akumulatora wynosi $A = 2^{8+I} - 1$ i może zostać zapisana w 16-bitowym słowie bez ryzyka przekroczenia zakresu. Okres normalizacji wynosi $2^J = 32$ ramki obrazu. Banki pamięci na platformie RC300 mają 2M-słów 36-bitowych więc w granicznym przypadku może być obsłużona wartość $I = 10$.

Po dołączeniu dodatkowych monitorów do platformy rekonfigurowalnej użytkownik może weryfikować treść strumieni wizyjnych na poszczególnych etapach przetwarzania (tło, stan akumulatorów, obraz różnicowy i status obecności) w trakcie działania aplikacji. Oprogramowanie sterujące uruchomione na komputerze PC jest odpowiedzialne za konfigurację układu FPGA, ładowanie maski obszarów detekcji do banku pamięci oraz ustawianie progów T_{VM} i T_{VP}. Po zakończonej inicjalizacji system wideodetekcji działa zupełnie samodzielnie. W celach testowych użytkownik może przy pomocy aplikacji sterującej dynamicznie zmieniać progi oraz wybierać sygnały do wizualizacji na monitorze DVI.

W tabeli 7.7 znajdują się wyniki implementacji wideodetektora dla jednego



(a) schemat systemu wideodetektora



(b) platforma rekonfigurowalna w środowisku uruchomieniowym

Rysunek 7.9: Prototyp przepływowego wideodetektora

obszaru detekcji. Jedynie 5% zasobów obliczeniowych układu reprogramowalnego zostało skonsumowane a pozostała część wolnych zasobów (obliczeniowych i pamięciowych) może być zużyta dla zwielokrotnienia ścieżki danych, aby obsłużyć więcej obszarów detekcji.

Tabela 7.7: Zużycie zasobów w wideodetektorze.

Wykorzystane zasoby układu FPGA XC2V6000	bezwzględne	względne (%)
cele logiczne (Slice)	1,940	5%
bramki przeliczeniowe	333,497	6%
pamięci BlockRam	4	2%

System pobiera strumień wizyjny z kamery PAL. Po usunięciu przeplotu dane w strumieniu wizyjnym formowane są zgodnie ze standardem XGA (format wizyjny o rozdzielczości 1024×768 i częstotliwości odświeżania 65Hz). Szacowane opóźnienie transportowe L_v toru wizyjnego wideodetektora przy wyznaczaniu sygnałów obecności i ruchu liczone od momentu wejścia sygnału wizyjnego do układu reprogramowalnego mieści się w przedziale od 18 do 274 cykli zegara, czyli od 0.28μs do 4.20μs. Przyczyną dużego rozstępu jest konieczność zsynchroni-

zowania dwóch części toru wizyjnego działających z różnymi przepustowościami. Strumień wizyjny w pierwszej części formowany jest zgodnie ze standardem PAL, w którym występuje przeplot. Kolejno występują półobrazy zawierające linie parzyste i nieparzyste. W buforze znajdującym się w drugiej części toru następuje usunięcie przeplotu poprzez scalenie kolejnych półobrazów. W celu zsynchronizowania dwóch strumieni zastosowano kolejkę FIFO o długości 256, która buforuje dane podczas usuwania przeplotu. Rozbieżności w tempie transmisji danych są przyczyną potencjalnie dużego opóźnienia. Przepływność w formacie XGA jest z definicji większa niż standardzie analogowym PAL, więc nie występuje ryzyko utraty danych w strumieniu, mogą jedynie wystąpić opóźnienia związane z buforowaniem danych [18].

Rozdział 8

Podsumowanie

8.1 Wyniki prac i wnioski

W rozprawie dokonano przeglądu równoległości w kontekście strumieniowych systemów wizyjnych. Analizę przeprowadzono na reprezentatywnych przykładach torów wizyjnych z uwzględnieniem operacji o różnorakim kontekście pikseli w obrazie:

- kontekst przestrzenny lokalny,
- kontekst przestrzenny globalny,
- kontekst temporalny.

8.1.1 Równoległość

Jako środek do realizacji wybranych algorytmów wizyjnych zaproponowano wykorzystanie różnych rodzajów równoległości adekwatnych do zidentyfikowanych ograniczeń:

- działanie strumieniowe (podstawowe założenie pracy),
- relatywnie niska częstotliwość zegara systemu obliczeniowego (podstawowe założenie pracy),
- różne rodzaje kontekstów i ich nieregularność,
- czas wykonania algorytmów zależny od treści strumienia wizyjnego,
- fizyczne ograniczenia platformy obliczeniowej.

Sposób wykorzystania równoległości przedstawiono etapami. Istotnym elementem pracy jest identyfikacja i zastosowanie praw opisujących sposoby zrównoleglania architektur obliczeniowych dedykowanych do wybranych algorytmów. Oryginalne osiągnięcie autora rozprawy stanowi opracowanie metodyki oceny stopnia zrównoleglenia algorytmów wizyjnych celem osiągnięcia efektywnej architektury strumieniowego systemu wizyjnego, wykonalnej dla określonej przepływności źródła sygnału wizyjnego.

Zaproponowana metodyka na którą składa się kilka elementów zaczerpniętych z teorii współbieżnych systemów obliczeniowych, elektroniki, telekomunikacji i automatyki, poszerzyła wachlarz narzędzi, które dotychczas były wykorzystywane do tworzenia znanych i opisywanych w literaturze strumieniowych systemów przetwarzania obrazów.

8.1.2 Operacje wizyjne o kontekście przestrzennym

Zbadano relacje pomiędzy rodzajem kontekstu piksela a zapotrzebowaniem na zasoby pamięciowe i ograniczeniami w realizacji strumieniowej. We wszystkich przypadkach zaproponowano architektury umożliwiające płynne strumieniowe przetwarzanie sygnału wizyjnego bez utraty danych. Szczególną uwagę zwrócono na realizację algorytmów wizyjnych cechujących się dużym rozstępem czasu wykonania uzależnionym od treści obrazu. Zaproponowano sposób szacowania rozstępu czasu wykonania oraz metodę jego zmniejszania na przykładzie przepływowego toru wizyjnego rozpoznawania znaków.

Szczególnym osiągnięciem w tym zakresie jest obserwacja, która pozwala na skuteczny dobór architektury strumieniowego toru wizyjnego w oparciu o zależności wynikające z granicznych warunków nakreślonych przez możliwy rozkład pikseli w obrazie. Podejście takie pozwala na lepszą ocenę wykonalności danego algorytmu w postaci strumieniowego systemu wizyjnego.

8.1.3 Operacje wizyjne o kontekście temporalnym

Szczególną uwagę zwrócono na realizację operacji wizyjnych o kontekście temporalnym na przykładzie sprzętowej aplikacji wideodetektora w ruchu drogowym. Zaproponowano sposób organizacji danych oraz szeregowania elementarnych operacji w potokowym systemie wideodetekcji, który pomimo dużego przepływu danych i ograniczeń zasobów pamięciowych platformy sprzętowej, pozwolił na płynne przetwarzanie wielu obszarów detekcji. Udało się to osiągnąć dzięki zaproponowanej przez autora rozprawy metodzie bilansowania różnych rodzajów równoległości: danych i operacji.

Wnioski wynikające z prac nad strumieniową architekturą algorytmu wideodetekcji opartego o estymację tła są następujące:

- prosty schemat potokowego przetwarzania strumienia wizyjnego nie jest wy-

starczający do implementacji operacji przetwarzania obrazów o kontekście temporalnym,

- poprzez odpowiednie zarządzanie dostępem do pamięci i transformację architektury potokowej możliwa jest strumieniowa realizacja operacji przetwarzania obrazu o kontekście temporalnym.

System strumieniowego wideodetektora został zaimplementowany dla jednego obszaru detekcji o kształcie i rozmiarze dopasowanym przez operatora do topologii monitorowanego obszaru jezdni. Opracowana metodologia zarządzania pamięcią i organizacji operacji w potoku, umożliwi skalowanie systemu wideodetekcji do 32 obszarów detekcji (dla użytej platformy prototypowej). Moduł IP (ang. *Intellectual Property*) zgodny ze standardem biblioteki PixelStreams został zaprojektowany w języku HandelC aby umożliwić wideodetekcję w czasie rzeczywistym. Funkcjonalna weryfikacja utworzonych komponentów systemów została przeprowadzona na etapie prototypowania i po implementacji z wykorzystaniem nagrań sygnału wizyjnego z kamery umieszczonej na skrzyżowaniu (patrz rys. 7.1). Podstawowe funkcjonalności referencyjnego modelu wideodetektora dla komputera PC zostały zweryfikowane na prototypie platformy sprzętowej. Co więcej dodano możliwość definiowania obszarów detekcji o dowolnym kształcie i rozmiarze bez wprowadzenia dodatkowych ograniczeń, co do wydajności, opóźnienia transportowego systemu strumieniowego, czy dodatkowych zasobów.

8.2 Kierunki dalszych badań

Jak wcześniej wspomniano, zaproponowane metody zrównoleglania oraz sposoby oceny ich rezultatów mogą być pomocne przy doborze efektywnych architektur obliczeniowych do danego zadania lub szacowaniu działania zrównoleglonych platform ogólnego przeznaczenia. Kierunki dalszych badań dotyczących oceny przydatności opisanej metodologii zrównoleglania, opartej o ilościową ocenę efektów mogą być różnorakie:

- Wykorzystanie metodologii w realizacji systemów wizyjnych wykorzystujących bardziej zaawansowane algorytmy przetwarzania analizy rozpoznawania obrazów 2D.

- Ocena przydatności opisanej metodologii w przetwarzaniu analizie i rozpoznawaniu strumienia zawierającego dane wielowymiarowe (np. 3D) i multimodalne.

- Rozwinięcie badań nad wpływem treści przetwarzanego strumienia danych na dobór efektywnej architektury, zwłaszcza w kontekście bardziej zaawansowanych algorytmów wizyjnych.

Zagadnieniem nie mniej istotnym jest próba wbudowania uzyskanej wiedzy w narzędzia projektowania (kompilatory i narzędzia do syntezy systemów cyfrowych), które pozwoliłyby na lepsze wykorzystanie zasobów współbieżnych i rekonfigurowalnych platform obliczeniowych. Automatyzacja procesu zrównoleglania w projektowaniu aplikacji mogłaby znacznie skrócić czas implementacji systemów wizyjnych i przyczynić się do lepszej efektywności działania w sensie rozmiaru, zapotrzebowania na energię czy przepustowość. Wiąże się to jednak z problemem automatycznego szacowania równoległości w oparciu o formalny opis implementowanego algorytmu.

Bibliografia

[1] A. Adamski and Z. Mikrut. The cracovian prototype of videodetector's feedback in transportation systems. In Transactions on Transport Systems Telematics, Monograph, pages 140–151. Wydawnictwo Politechniki Śląskiej, Gliwice, 2004. [cytowanie na str. 99]

[2] M. Adamski. Projektowanie układów cyfrowych systematyczną metodą strukturalną, volume 49 of Monografie. Wyd. Wyższej Szkoły Inżynierskiej, Zielona Góra, Lipiec 1990. [cytowanie na str. 15]

[3] M. Adamski. Parallel controller implementation using standard pld software. In FPGAs: International Workshop on Field Programmable Logic and Applications, pages 296–304, Abingdon, 1991. Abingdon EE&CS Books. [cytowanie na str. 15]

[4] M. Adamski. Formal logic design of reprogrammable controllers. In Design of embedded control systems, ISBN: 0-387-23630-9, pages 15–26. Springer, New York, 2005. [cytowanie na str. 15]

[5] G. Amdahl. Validity of the single processor approach to achieving large-scale computing capabilities. Proceedings of AFIPS Spring Joint Computer Conference, pages 483–485, April 1967. [cytowanie na str. 18]

[6] G. Amit, Y. Caspi, R. Vitale, and A. Pinhas. Scalability of multimedia applications on next-generation processors. In IEEE International Conference on Multimedia and Expo, volume 7, pages 17–20, USA, July 2006. [cytowanie na str. 19]

[7] Analog Devices„ Norwood USA. ADSP-BF561 Blackfin Embedded Symmetric Multiprocessor, rev. d edition, February 2009. [cytowanie na str. 5, 13]

[8] M. Annavaram, E. Grochowski, and J. Shen. Mitigating amdahl's law through epi throttling. In 32nd International Symposium on Computer Architecture, pages 298–309. IEEE, June 2005. [cytowanie na str. 15, 18]

[9] K. Appiah, A. Hunter, P. Dickinson, and J. Owens. A run-length based connected component algorithm for fpga implementation. In International Conference on Field-Programmable Technology. 2008. [cytowanie na str. 55, 74]

[10] P. Augustyniak. Detecting patient's emergency - a minimum-computation procedure for pervasive cardiac monitoring. In 30th Annual International Conference

on Engineering in Medicine and Biology Society, pages 1439–1442. IEEE, August 2008. [cytowanie na str. 14, 32]

[11] Z. Baker, M. Gokhale, and J. Tripp. Matched filter computation on fpga, cell and gpu. In 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pages 207–218, USA, April 2007. IEEE. [cytowanie na str. 5]

[12] B. Bayer. Color imaging array. July 1976
. http://www.pat2pdf.org/patents/pat3971065.pdf. [cytowanie na str. 46]

[13] G. Beane. The effects of microprocessor architecture on speedup in distributed memory supercomputers. Master's thesis, The University of Maine, 2004. [cytowanie na str. 20]

[14] J. Becker and K. Schmidt. Automatic parallelism exploitation for fpl-based accelerators. In Thirty-First Hawaii International Conference on System Sciences, volume 7, pages 169–178, USA, January 1998. [cytowanie na str. 13, 15, 16]

[15] K. Benkrid, S. Sukhsawas, D. Crookes, and S. Belkacemi. A single-fpga implementation of image connected component labelling. In Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field Programmable Gate Arrays, pages 238 – 238. ACM, 2003. [cytowanie na str. 28, 55, 74]

[16] Z. Bubliński and M. Jabłoński. Rozszerzenia simd w przetwarzaniu obrazów. In Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, volume 12(3), pages 609–614, Kraków, 2008. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne. [cytowanie na str. 21]

[17] Z. Bubliński, M. Jabłoński, and Z. Mikrut. Analiza sekwencji filmowych w środowisku virtualdub w oparciu o platformę fpga. In Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, volume 10(3), pages 323–333, Kraków, 2006. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne. [cytowanie na str. 31, 74]

[18] Celoxica. Platform Developer's Kit: PixelStreams Manual. Celoxica, 2006. [cytowanie na str. 102, 115]

[19] Cypress Semiconductor Corporation, San Jose, USA. LUPA-1300 1.3MPixel High Speed CMOS Image Sensor, 2005. http://download.cypress.com.edgesuite.net/design_resources/datasheets/contents/lupa_1300_8.pdf. [cytowanie na str. 41]

[20] H. Fatemi. Processor Architecture Design for Smart Cameras. PhD thesis, Technische Universiteit Eindhoven, 2007. [cytowanie na str. 14, 16, 17, 26, 30, 32]

[21] H. Feng, E. Li, Y. Chen, and Zhang Y. Parallelization and characterization of sift on multi-core systems. In IEEE International Symposium on Workload Characterization, pages 14–23. IEEE, Sptember 2008. [cytowanie na str. 5]

[22] M. Flynn. Some Computer Organizations and Their Effectiveness, volume C(21), page 948. IEEE Trans. Comput., 1972. [cytowanie na str. 14]

[23] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. In Proceedings of Annual Conference on Uncertainty in Artificial Intelligence, pages 175–181, 1997. [cytowanie na str. 24, 98, 99]

[24] Sasă Galić and Sven Lončarić. Spatio-temporal image segmentation using optical flow and clustering algorithm. In First International Workshop on Image and Signal Processing and Analysis, IWISPA 2000, pages 63–68. IEEE, June 2000. [cytowanie na str. 24]

[25] W. Gerstner. Spiking neurons. In C. M. Bishop W. Maass, editor, Pulsed Neural Networks, pages 3–53. MIT Press, 1998. [cytowanie na str. 90]

[26] M. Gorgoń. Ocena specjalizowanych procesorów sprzętowych pod kątem wstępnego przetwarzania obrazów. PhD thesis, AGH, Kraków, 1995. [cytowanie na str. 48]

[27] M. Gorgoń. Retina - karta dokonująca readresacji i transformacji obrazu w czasie rzeczywistym. Technical Report 94/2000, AGH, 2000. [cytowanie na str. 13]

[28] M. Gorgoń. Zrównoleglanie operacji na obrazach cyfrowych w strukturach rekonfigurowalnych. In Materiały seminarium: Przetwarzanie i analiza sygnałów w systemach wizji i sterowania, pages 79–84, Słok k/Bełchatowa, Czerwiec 2002. [cytowanie na str. 21, 22]

[29] M. Gorgoń. Architektury rekonfigurowalne do przetwarzania i analizy obrazu oraz dekodowania cyfrowego sygnału wideo. Rozprawy Monografie. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne, 2007. [cytowanie na str. 6, 16]

[30] M. Gorgoń, P. Pawlik, M. Jabłoński, and J. Przybyło. Fpga-based road traffic videodetector. In Digital System Design, DSD'07, pages 412–419, Lubeck, Germany, April 2007. IEEE Computer Society. [cytowanie na str. 97, 98]

[31] M. Gorgoń, P. Pawlik, M. Jabłoński, and J. Przybyło. Pixelstreams-based implementation of videodetector. In proceedings of the 15th Annual Symposium on Field-Programmable Computing Machines - FCCM'07, Napa, USA, April 2007. [cytowanie na str. 98]

[32] C. Graetzel, S. Fry, and B. Nelson. A 6000 hz computer vision system for real-time wing beat analysis of drosophila. In Proceedings The First IEEE/RAS-EMBS International Conference on of Biomedical Robotics and Biomechatronics, BioRob 2006., volume 0, pages 243–270. IEEE, 2006. [cytowanie na str. 33, 42]

[33] A.K. Gupta, S. Nooshabadi, D. Taubman, and M. Dyer. Realizing low-cost high-throughput general-purpose block encoder for jpeg2000. In IEEE Transactions on Circuits and Systems for Video Technology, volume 16(7), pages 843–858. IEEE, July 2006. [cytowanie na str. 5]

[34] J. Gustafson. Reevaluating amdahl's law. In Communications of the ACM, volume 31, pages 532–533. 1987. [cytowanie na str. 19]

[35] J. Gustafson. Fixed time, tiered memory, and superlinear speedup. In Proceedings of the Fifth Distributed Memory Computing Conference, pages 532–533, October 1990. [cytowanie na str. 20]

[36] D. Hammerstrom. A survey of bio-inspired and other alternative architectures. In Rainer Waser, editor, Information Technology, Nanotechnology. John Wiley & Sons. [cytowanie na str. 67]

[37] D. Hammerstrom. Cortical models onto cmol and cmos—architectures and performance/price. In Rainer Waser, editor, IEEE Transactions on Circuits and Systems, volume 54. IEEE, November 2007. [cytowanie na str. 90]

[38] R. Hartenstein. Handbook of Nature-Inspired and Innovative Computing, volume 2006, chapter Morphware and Configware, pages 343–386. Springer US, March. [cytowanie na str. 28]

[39] R. Hartenstein. Custom computing machines vs. hardware/software co-design: From a globalized point of view. In R. Hartenstein and M. Gelsner, editors, Field-Programmable Logic, Smart Applications, New Paradigms and Compilers, pages 65–76, Darmstadt, Germany, September 1996. Springer. [cytowanie na str. 27]

[40] R. Hartenstein. Coarse grain reconfigurable architectures. In R. Hartenstein and M. Gelsner, editors, Proceedings of the ASP-DAC 2001 Asia and South Pacific Design Automation Conference, pages 564–569. IEEE, February 2001. [cytowanie na str. 25]

[41] R. Hartenstein. Are we really ready for the breakthrough? In Parallel and Distributed Processing Symposium, page 7. IEEE, April 2003. [cytowanie na str. 5, 25]

[42] R. Hartenstein. Reconfigurable hpc: torpedoed by deficits in education? In Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region, pages 428–429. IEEE, July 2004. [cytowanie na str. 5]

[43] R. Hartenstein. The von neumann syndrome. In Stamatis Vassisliadis Memorial Symposium, pages 42–54, Delft, The Netherlands, August 2007. [cytowanie na str. 13, 16]

[44] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan. Mesheye: a hybrid-resolution smart camera mote for applications. In Proc. of 6th International Conference on Information Processing in Sensor Networks (IPSN/SPOTS), pages 360–369, April 2007. [cytowanie na str. 14]

[45] Intel. Image and video processing. In Reference Manual: Intel® Integrated Performance Primitives for Intel® Architecture, volume 2, pages 871–879. ippiman.pdf, 2005. [cytowanie na str. 55, 74]

[46] Y. Ito and K. Nakano. Component labeling for k-concave binary images using an fpga. In Proceedings of International Symposium on Parallel and Distributed Processing, pages 1 – 8. IEEE, 2008. [cytowanie na str. 33]

[47] E. Izhikevich. Some Computer Organizations and Their Effectiveness, volume 14, pages 1569–1572. IEEE Transactions on Neural Netowrks, 2003. [cytowanie na str. 90]

[48] M. Jablonski. Od obrazu do matrycy znaku - implementacja sprzętowa. In Materiały seminarium: Przetwarzanie i analiza sygnałów w systemach wizji i sterowania, pages 38–43, Słok k/Bełchatowa, Czerwiec 2002. [cytowanie na str. 28, 51]

[49] M. Jabłoński. Inteligentna kamera - podsystem automatycznej kalibracji barwnej. In Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, volume 11(3), pages 245–257, Kraków, 2007. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne. [cytowanie na str. 53]

[50] M. Jabłoński. Reconfigurable fpga-based platform enables real-time video detection. In Transportation and Logistics Intergrated Systems ITS-ILS'07, pages 38–47, Kraków, April 2007. ISBN 978-83-88309-86-1. [cytowanie na str. 28, 100]

[51] M. Jabłoński, J. Przybyło, and M. Gorgoń. Real-time implementation of motion detection algorithm based on pixelstreams. In Z. Bradac, editor, IFAC workshop on Programmable Devices and embedded Systems: PDeS 2006, pages 186—-190, Brno, February 2006. IFAC. [cytowanie na str. 97]

[52] M. Jabłoński, J. Przybyło, and P. Wołoszyn. Automatyczna segmentacja twarzy dla potrzeb interfejsu człowiek-komputer. In Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, volume 9(3), pages 599—600, Kraków, 2005. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne. [cytowanie na str. 24]

[53] M. Jabłoński and Bubliński Z. Integracja toru wizyjnego na platformie rekonfigurowalnej. In Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, volume 12(3), pages 657–667, Kraków, 2008. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne. [cytowanie na str. 28, 31, 33, 42]

[54] E. Jamro. Parameterised automated generation of convolers implemented in FPGAs. PhD thesis, AGH-University of Science and Technology, Kraków, June 2001. [cytowanie na str. 23]

[55] H Jiang, H. Ardo, and V. Owall. Hardware accelerator design for video segmentation with multi-modal background modelling. In Poceedings of IEEE International Symposium on Circuits and Systems - ISCAS 2005, volume 2, pages 1142–1145, Kobe, Japan, May 2005. IEEE. [cytowanie na str. 99]

[56] H. Jiang, H. Ardo, and V. Owall. A hardware architecture for real-time video segmentation utilizing memory reduction techniques. In IEEE Transactions on Circuits and Systems for Video Technology, volume 19(2), pages 226–236. IEEE, February 2009. [cytowanie na str. 5]

[57] H Jiang and V. Owall. Fpga implementation of real-time image convolutions with three level of memory hierarchy. In IEEE International Conference on Field-Programmable Technology, pages 424–427. IEEE, December 2003. [cytowanie na str. 26]

[58] A. Karp and H. Flatt. Measuring parallel processor performance. In Communication of the ACM, volume 33(5), pages 539–543. May 1990. [cytowanie na str. 20]

[59] R. Kleihorst, M Reuver, B. Kriise, and H Broers. A smart camera for face recognition. In Proceedings of International Conference on Image Processing (ICIP 2004), pages 2849–2852, 2004. [cytowanie na str. 58]

[60] T. Kong and A. Rosenfeld. Topological algoritms for digital image processing, volume 19. Elsevier, Amsterdam, 1996. [cytowanie na str. 54, 74]

[61] J. Koronacki and J. Mielniczuk. Statystyka dla studentów kierunków technicznych i przyrodniczych. Wydawnictwa Naukowo-Techniczne, Warszawa, 2004. [cytowanie na str. 67]

[62] S. Kozielski and Szczerbiński Z. Komputery równoległe - architektura, elementy oprogramowania. Wydawnictwa Naukowo-Techniczne, Warszawa, 1993. [cytowanie na str. 18, 28, 85]

[63] M. Kwiczala and K. Wiatr. Sprzętowa implementacja sieci neuronowych – aproksymacja funkcji aktywacji. In Computer Methods and Systems, pages 83–88. AGH, ONT, November 2005. [cytowanie na str. 67]

[64] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. In Journal. of Applied Statistics, volume 21, pages 224–270, 1994. [cytowanie na str. 24]

[65] D. Lyons and I. Giselle. Evaluation of a Parallel Architecture and Algorithm for Mapping and Localization. Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation, June 2007. [cytowanie na str. 20]

[66] Matrox, Canada. Matrox Iris E-Series with Design Assistant, August 2008. http://www.matrox.com/imaging/products/odyssey_xproplus/b_odyssey_xpro_plus.pdf. [cytowanie na str. 13]

[67] Matrox, Canada. Matrox Odyssey Xpro+ Scalable vision processor board with customizable co-processor FPGA, vision processors edition, October 2008. http://www.matrox.com/imaging/products/odyssey_xproplus/b_odyssey_xpro_plus.pdf. [cytowanie na str. 13]

[68] B. Mesman, H. Fatemi, H. Corporaal, and T. Basten. Dynamic-simd for lens distortion compensation. In Proceedings of the 17th IEEE Conference on Application-specific Systems, Architectures, and Processors (ASAP), pages 261––264, Steamboat Springs, Colorado, USA, September 2006. IEEE Computer Society. [cytowanie na str. 23]

[69] Micron Technology Inc. 1/3-Inch, Wide-VGA CMOS Digital Image Sensor, 2006. PDF:09005aef8201ffc3/Source: 09005aef81ff2525 MT9V022_Product_Brief - Rev. A 1/06 EN. [cytowanie na str. 42, 43, 44, 45]

[70] Z. Mikrut. Rozpoznawanie ręcznie pisanych cyfr za pomocą sieci neuronowych o różnych strukturach. In Automatyka: Zeszyty Naukowe Akademii Górniczo-Hutniczej im. Stanisława Staszica, volume 66, pages 31–58. PWN, Kraków, 1993. [cytowanie na str. 50, 51, 61]

[71] Z. Mikrut. Road traffic measurement using videodetection. In Image Processing and Communications, volume 3(3-4), pages 19–30, Bydgoszcz, 1997. Institute of Telecommunications. [cytowanie na str. 97, 99]

[72] Z. Mikrut. Algorytmy wideodetekcji w analizie ruchu drogowego. In Materiały seminarium: Przetwarzanie i analiza sygnałów w systemach wizji i sterowania, pages 67–72, Słok k/Bełchatowa, Czerwiec 2002. [cytowanie na str. 96, 97, 98, 99]

[73] Z. Mikrut. The cracovian videodetector - from ideas to embedding. In Transportation and Logistics Intergrated Systems ITS-ILS'07, pages 29–37, Kraków, April 2007. ISBN 978-83-88309-86-1. [cytowanie na str. 97]

[74] Z. Mikrut and Z. Stanek. Low-cost board for image digitalization, remapping and processing. In Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, volume 3(2), pages 487–497. PWN, Kraków, 1999. [cytowanie na str. 6, 13]

[75] G. Moore. Cramming more components onto integrated circuits. Electronics, 38(8), April 1965. [cytowanie na str. 5]

[76] M. Ogiela, R. Tadeusiewicz, and L. Ogiela. Cognitive vision techniques in medical image processing and analysis. In 7th International Workshop on Enterprise networking and Computing in Healthcare Industry, pages 120–123. IEEE, June 2005. [cytowanie na str. 24]

[77] Photon Phocus. Welded joint inspection in vehicle production. In Image Processing. Photon Phocus AG, April 2004. [cytowanie na str. 33]

[78] Photonfocus AG, Switzerland. MV-D1024E-PP01 CameraLink®- CMOS Area Scan Camera, December 2008. http://www.photonfocus.com/upload/manuals/MAN043_V1_0_MV-D1024E-PP01.pdf. [cytowanie na str. 13]

[79] I. Pitas. Introduction to parallel digital image processing. In Pitas I., editor, Parallel algorithms for digital image processing, computer vision and neural networks, pages 1–24. John Wiley & Sons, 1993. [cytowanie na str. 17, 23]

[80] J. Przybyło. Śledzenie cech charakterystycznych twarzy w systemie rozpoznawania mimiki. In Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, volume 11(3), pages 257–266, Kraków, 2007. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne. [cytowanie na str. 24]

[81] J. Przybyło, M. Jabłoński, and P. Wołoszyn. Detekcja markerów dla celów automatycznej anotacji mimiki twarzy. In Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, volume 10(3), pages 413—425, Kraków, 2006. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne. [cytowanie na str. 24, 42]

[82] M. Rofouei, M. Moazeni, and M. Sarrafzadeh. Fast gpu-based space-time correlation for activity recognition in video sequences. In IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia, pages 33–38. IEEE, October 2008. [cytowanie na str. 5]

[83] F. Rosenblatt. The perceptron: A probabilistic model for information storage in the brain. In Psychological review, volume 65, pages 368–408. 1958. [cytowanie na str. 61]

[84] H. Seo and S. Kim. Openmp directive extension for blackfin 561 dual core processor. In The Sixth IEEE International Conference on Computer and Information Technology, pages 49 – 55. IEEE, Sptember 2006. [cytowanie na str. 5]

[85] L. Shapiro and G. Stockman. Computer vision, pages 63–67. Prentice Hall, 2001. [cytowanie na str. 54]

[86] Y. Shi. Reevaluating Amdahl's Law and Gustafson's Law. http://www.cis.temple.edu/~shi/docs/amdahl/amdahl.html. [cytowanie na str. 20]

[87] O. Sims and J. Irvine. An fpga implementation of pattern-selective pyramidal image fusion. In International Conference on Field Programmable Logic and Applications, pages 1–4. IEEE, August 2006. [cytowanie na str. 5]

[88] W. Skarbek. Segmentation of Colour Images. PWN, 1994. [cytowanie na str. 46]

[89] M. Sobczyk. Statystyka. Wydawnictwa Naukowe PWN, Warszawa, 2007. [cytowanie na str. 67]

[90] L. Sousa and M. Piedad. Low level parallel image processing. In I. Pitas, editor, Parallel algorithms for digital image processing, computer vision and neural networks, pages 25–52. John Wiley & Sons, 1993. [cytowanie na str. 23, 29]

[91] C. Stauffer and W. Grimson. Adaptive background mixture models for realtime tracking. In Proc. IEEE conf. Computer Vision and Pattern Recognition, 1999. [cytowanie na str. 24, 31, 96, 98]

[92] K. Suzuki, I. Horiba, and N. Sugie. Linear-time connected-component labeling based on sequential local operations. volume 89 of Computer Vision and Image Understanding, pages 1–23. Elsevier Science Inc, 2003. [cytowanie na str. 54]

[93] R. Tadeusiewicz. Systemy wizyjne robotów przemysłowych. Wydawnictwo Naukowo-Techniczne, Warszawa, 1992. [cytowanie na str. 6, 24]

[94] R. Tadeusiewicz. Sieci neuronowe. Akademicka Oficyna Wydawnicza, Warszawa, 1993. [cytowanie na str. 50, 60, 61, 62]

[95] R. Tadeusiewicz and P. Korohoda. Komputerowa analiza i przetwarzanie obrazów. Wydawnictwo Fundacji Postępu Telekomunikacji, 1997. [cytowanie na str. 24, 31, 54, 74]

[96] R. Tadusiewicz and M. Ogiela. Automatic understanding of images. In Soft Computing, Intelligent System and Information Technology - ICSIIT 2007, pages 13–38, Indonesia, July 2007. [cytowanie na str. 24]

[97] I. Taniguchi, M. Jayapala, P. Raghavan, K.and Takeuchi Y. Catthoor, F.and Sakanushi, and M.; Imai. Systematic architecture exploration from vast solution space is a complex problem in embedded system design. In Asia and South Pacific ASP-DAC 2009 Design Automation Conference, pages 449–454. IEEE, January 2009. [cytowanie na str. 25, 32]

[98] I. Tuomi. The lives and death of moore's law. First Monday, 7(11), November 2002. http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/issue/view/151. [cytowanie na str. 5]

[99] Videor, Germany. Smart and Freely Programmable Open Source Camera Systems under Linux, October 2008. http://www.videor.com/mall/1/assets/brochure/59728_en_bro.zip. [cytowanie na str. 13]

[100] K. Wiatr. Architektura specjalizowanych procesorów sprzętowych do wstępnego przetwarzania obrazów w systemach wizyjnych czasu rzeczywistego, volume 67 of Rozprawy i Monografie. Wydawnictwa AGH, Kraków, 1998. [cytowanie na str. 33]

[101] K. Wiatr. Sprzętowe implementacje algorytmów przetwarzania obrazów w systemach wizyjnych czasu rzeczywistego. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne, 2002. [cytowanie na str. 6, 48]

[102] K. Wiatr and E. Jamro. Implementation image data convolutions operations in fpga reconfigurable structures for real-time vision systems. In Proceedings of International conference on Information technology: coding and computing, pages 152–157, Los Alamitos, USA, 2000. [cytowanie na str. 23, 28, 30]

[103] P. Wołoszyn, J. Przybyło, and M. Jabłoński. Analiza przydatności metod komunikacji z komputerem w tworzeniu interfejsu dla osób niepełnosprawnych. In Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, volume 7(3), pages 399—408, Kraków, 2003. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne. [cytowanie na str. 24]

[104] M. Wrzesiński, M. Gorgoń, and Mikrut Z. Implementacja perceptronu wielowarstwowego w układach fpga typu virtex i spartan-ii. In Materiały seminarium: Przetwarzanie i analiza sygnałów w systemach wizji i sterowania, pages 209–217, Słok k/Bełchatowa, Czerwiec 2002. [cytowanie na str. 67]

[105] Xilinx. Virtex-II Platform FPGAs: Complete Data Sheet, volume DS031 (v3.5). http://www.xilinx.com/support/documentation/data_sheets/ds031.pdf, 2007. [cytowanie na str. 96]

[106] Xilinx. Extended Spartan-3A Family Overview, volume DS706 (v1.0). http://www.xilinx.com/support/documentation/data_sheets/ds706.pdf, 2008. [cytowanie na str. 25, 96]

[107] Xilinx. MicroBlaze Processor Reference Guide, volume UG081 (v9.0). http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf, 2008. [cytowanie na str. 17]

[108] Xilinx. Virtex-5 Family Overview, volume DS100 (v5.0). http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, 2009. [cytowanie na str. 25]

[109] J. Xu, Y. Dou, J. Li, X. Zhou, and Q. Dou. Fpga accelerating algorithms of active shape model in people tracking applications. In 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, pages 432–435. IEEE, August 2007. [cytowanie na str. 5]

Dodatki

Dodatek A

Wyznaczanie przyspieszenia w zrównoleglaniu wieloetapowym

A.1 Metody wyznaczania przyspieszenia

Jak wspomniano we wstępnie niniejszej rozprawy przyspieszenie można wyznaczyć na szereg sposobów korzystając z pomiaru (A.1) lub analizy zastosowanego rozwiązania w oparciu o prawo Amdahl'a (A.2) lub Gustafson'a a (A.3). Na potrzeby realizacji sprzętowej algorytmów wizyjnych autor dokonał analizy porównawczej poszczególnych metod i wykazał ich równoważność przy przyjętych założeniach.

$$S_m = \frac{T_1}{T_n} \tag{A.1}$$

gdzie:

S_m – przyspieszenie wyznaczone w oparciu o pomiar czasu wykonania,
T_1 – czas sekwencyjnego wykonania algorytmu na jednej jednostce obliczeniowej,
T_n – czas wykonania algorytmu zrównoleglonego przy pomocy n współbieżnych jednostek obliczeniowych,
n – liczba jednostek obliczeniowych.

$$S_a = \frac{1}{s + \frac{p}{n}} \tag{A.2}$$

gdzie:

S_a – przyspieszenie wyznaczone w oparciu o prawo Amdahl'a,
p – znormalizowana ilość obliczeń zrównoleglonych,
s – znormalizowana ilość obliczeń niezrównoleglonych,
n – liczba zrównoleglonych jednostek obliczeniowych,
p i s są znormalizowane aby spełnić warunek: $p + s = 1$.

$$S_g = n - (n-1)s' \tag{A.3}$$

gdzie:

S_g – przyspieszenie wyznaczone w oparciu o prawo Gustafson'a,
s' – znormalizowany czas wykonania niezrównoleglonej części algorytmu,
p' – znormalizowany czas wykonania zrównoleglonej części algorytmu,
n – liczba zrównoleglonych jednostek obliczeniowych,
p' i s' są znormalizowane aby spełnić warunek: $p' + s' = 1$.

W kontekście niniejszej rozprawy przyjęto że czas wykonania operacji jest skwantowany, a każda operacja będąca przedmiotem zrównoleglenia lub sekwencyjnego wykonania trwa dokładanie jeden cykl zegarowy.

A.1.1 Wyznaczanie przyspieszenia etapami

Z definicji (A.1) widać że przyspieszenie jest miarą skrócenia czasu wykonania obliczeń. Całkowite przyspieszenie, będące wynikiem kolejnych zabiegów zrównoleglenia jest zatem iloczynem przyspieszeń S^γ na kolejnych etapach γ.

$$S^\Gamma = \prod_{\gamma=1}^{\Gamma} S^\gamma \tag{A.4}$$

gdzie:

S^Γ – całkowite przyspieszenie,
S^γ – przyspieszenie na γ-tym etapie zrównoleglenia,
γ – etap zrównoleglenia,
Γ – liczba etapów zrównoleglenia.

A.1.2 Iteracyjne wyznaczanie przyspieszenia

Załóżmy, że na wykonanie pewnego sekwencyjnego algorytmu potrzeba C_0 cykli natomiast po zrównolegleniu na wykonanie tego samego zadania potrzeba C_Γ cykli. Przyjmijmy również, że liczba etapów zrównoleglenia Γ jest równa różnicy $C_0 - C_\Gamma$. Możemy wówczas zauważyć, że podczas każdego etapu zrównoleglenia liczba zrównoleglonych obliczeń elementarnych trwających jeden cykl wynosi $n = 2$. Wykorzystując własność (A.4) i prawo Amdahl'a (A.2) wyliczamy przyspieszenie $S_a^{(0-\Gamma)}$ zgodnie z przekształceniami (A.5). Identyczny wynik (A.6) uzyskano wychodząc z prawa Gustafson'a (A.3) wyznaczając $S_g^{(0-\Gamma)}$.

$$S_a^{(1-\Gamma)} = \prod_{C_1}^{C_\Gamma+1} \left(\frac{1}{\frac{c-2}{c} + \frac{2}{c}\frac{1}{2}} \right) = \prod_{C_1}^{C_\Gamma+1} \left(\frac{1}{\frac{c-1}{c}} \right) = \prod_{C_1}^{C_\Gamma+1} \left(\frac{c}{c-1} \right) = \frac{C_1}{C_\Gamma} \tag{A.5}$$

$$S_g^{(1-\Gamma)} = \prod_{C_1}^{C_\Gamma+1} \left(2 - (2-1)\frac{1}{c-1} \right) = \prod_{C_1}^{C_\Gamma+1} \left(\frac{c}{c-1} \right) = \frac{C_1}{C_\Gamma} \tag{A.6}$$

Widać również że wyznaczona zależność znajduje potwierdzenie we wzorze na przyspieszenie (A.1) bazującym na pomiarze czasu wykonania ponieważ $T_1 = C_1/f$ a $T_n = C_\Gamma/f$ gdzie f jest częstotliwością taktowania systemu. Warto zauważyć, że w iteracyjnej metodzie wyznaczania przyspieszenia nie występuje całkowita liczba zrównoleglonych jednostek $n_{1-\Gamma}$. Można ją jednak dokładnie oszacować jak we wzorze (A.7) celem podstawienia do równań (A.2) lub (A.3).

$$n_{1-\Gamma} = C_1 - C_\Gamma + 1 \tag{A.7}$$

Dodatek B

Przyspieszenie i efektywność w architekturze potokowej

B.1 Wpływ parametrów potoku na przyspieszenie

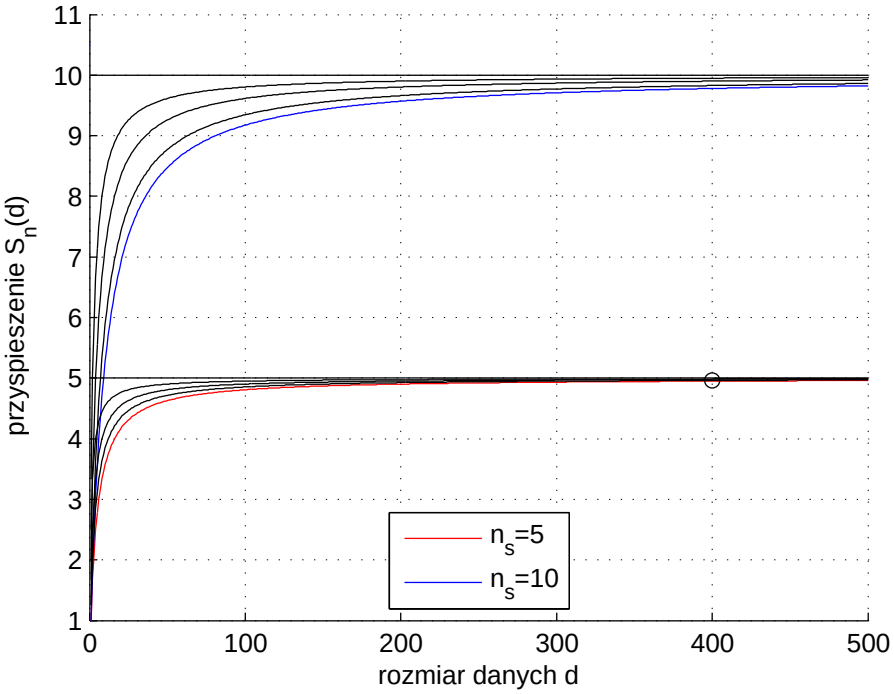
Na rysunku B.1 przedstawiono rodziny krzywych opisujących przyspieszenie w zależności od rozmiaru danych które są przetwarzane. Oznaczone kolorami przebiegi odpowiadają sytuacjom w których liczba operacji i liczba etapów są sobie równe $n_s = n_p$ (równanie B.1). Poziome linie mają znaczenie dwojakie. Po pierwsze określają asymptoty które w granicy $d \to \infty$ wyznaczają wartość przyspieszenia. Wysokość na której umieszczona jest prosta odpowiada realnemu przyspieszeniu w przepływowym przetwarzaniu danych ze strumienia, pod warunkiem zachowania formatu danych. Druga interpretacja odnosi się do przyspieszenia stałego w stosunku rozmiaru danych. Dotyczy to zrównoleglenia zbioru operacji które nie są od siebie zależne i mogą być wykonane współbieżnie bez szeregowania w czasie ($n_p = 1$).

$$S_{ns} = \frac{n_s d}{n_p + d - 1} \tag{B.1}$$

gdzie:

S_{ns} – przyspieszenie jednostki potokowej,
$d \in \mathbb{N}$ – liczba kwantów danych do przetworzenia,
$n_s \in \mathbb{N}$ – liczba sekwencyjnych operacji,
$n_p \in \mathbb{N}$ – liczba stopni potoku.

Przebiegi przedstawione na rysunku B.1 wyznaczono dla wartości podstawionych



Rysunek B.1: Wykres przyspieszenia potoku

Punkt na wykresie oznaczony kołem odpowiada aktualnym parametrom architektury potokowej przedstawionej w rozdziale 6.5.1.

Tabela B.1: Parametry potoku

	n_s	n_p		n_s	n_p
czerwony	5	5	niebieski	10	10
	5	4		10	8
	5	3		10	5
	5	2		10	3
	5	1		10	1

z tabeli B.1 do wzoru B.1. Duża liczba operacji sekwencyjnych n_s daje większe możliwości zrównoleglenia a efekt jest tym lepszy im mniej jest etapów potoku n_p. Należy więc dążyć do takiej organizacji operacji aby liczba zależności pomiędzy operacjami była jak najmniejsza. Uwzględniając aspekt technologiczny związany z czasem propagacji wynikającym ze złożoności etapów potoku, uzasadnione jest rozbijanie złożonych operacji na mniejsze.

B.2 Wpływ parametrów potoku na efektywność

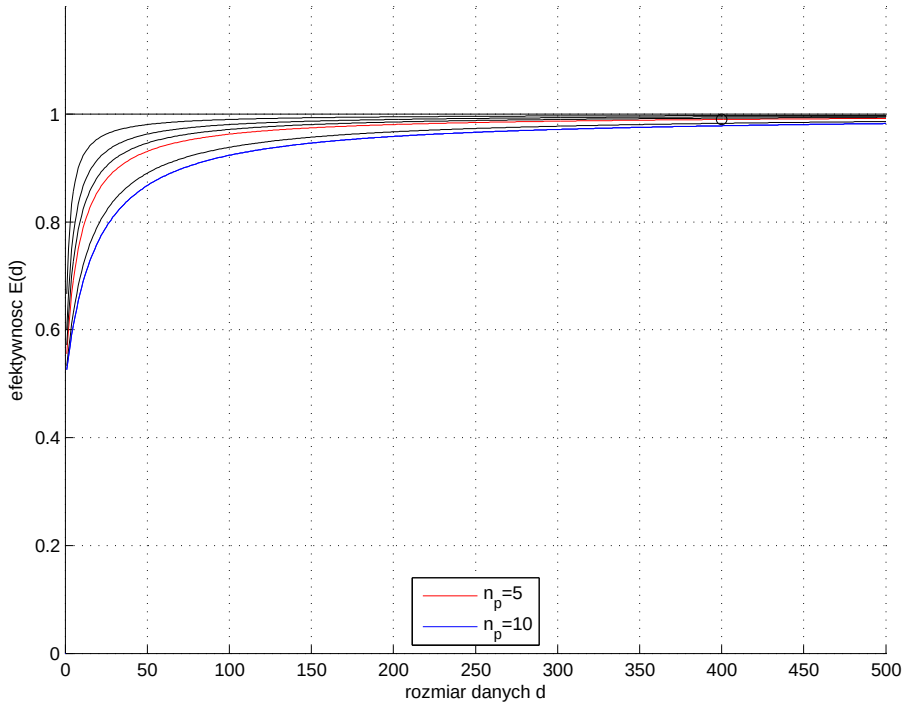
Na rysunku B.2 przedstawiono rodzinę krzywych opisujących efektywność architektury potokowej. Efektywność zrównoleglenia (patrz wzór B.2) zależy przede

wszystkim od rozmiaru przetwarzanych danych oraz liczby etapów w potoku. Nie jest natomiast wrażliwa na całkowitą liczbę operacji. Efektywność maleje ze wzrostem liczby etapów jednak w granicy przy $d \to \infty$ lub $n_p = 1$ efektywność dąży do wartości 1.0. Zatem w przetwarzaniu ciągłego strumienia danych można przyjąć że efektywność potoku $E_{np} = 1.0$.

$$E_{np} = \frac{n_p + d}{2n_p + d - 2} \tag{B.2}$$

gdzie:

E_{np} – efektywność jednostki potokowej,
$d \in \mathbb{N}$ – liczba kwantów danych do przetworzenia,
$n_p \in \mathbb{N}$ – liczba stopni potoku.



Rysunek B.2: Wykres efektywności potoku

Punkt na wykresie oznaczony kołem odpowiada aktualnym parametrom architektury potokowej przedstawionej w rozdziale 6.5.1.

Dodatek C

Zrównoleglony kod źródłowy - przykłady

C.1 Indeksacja obiektów w potoku drobnoziarnistym

```
\include{label.hch}
void main(void)
{
   unsigned LOG2_LABEL objCnt[NUM_PROC];
   static unsigned 1 i=0;
   par{
        par{
              DataSource();
              Label(&TAB,&objCnt));
              seq{
                 delay;
                 AuxSink();
              }
        }
        par{
              AuxSource();
              Lut(&TAB);
              seq {
                 delay;
                 delay;
                 DataSink();
              }
        }
    }
    ReindexTab(&TAB,&objCnt);
}
```

C.2 Wideodetektor - bufor kontekstu temporalnego

Zaimplementowany moduł IP jako rozszerzenie biblioteki PixelStreams.

```
BEGIN_FILTER (PxsFrameSampler)
    (INPUT (In),
     INPUT (InHold),
     OUTPUT (Out),
     PARAMX (PL1RAM, "PalPL1RAMCT(0)", PXS_CONSTANT),
     PARAM  (ClockRate, ClockRate, PXS_CONSTANT))
{
   signal unsigned 36 ram_data;
   unsigned 32 read_data;
   static unsigned 32 frame_counter = 0;  //frame counter
   static unsigned 1 update = 0;
   PXS_SAME (Stage1,In);
   PXS_SAME (Stage1_H, InHold);
   par{
        PalPL1RAMRun    (PL1RAM, ClockRate);
        PalPL1RAMEnable (PL1RAM);
      while (1) par{
         PxsCopyVAPCSH (In, &Stage1);
         PxsCopyVAPCSH (InHold, &Stage1_H);
         PxsCopyVACSH (&Stage1, Out);
         if (In->Valid && In->Active)par{
            if (InHold->Pixel.U8[PXS_M])par{
               PalPL1RAMSetWriteAddress(PL1RAM,0@((unsigned 10)In->Coord.Y[9:0]@
                                                  (unsigned 10)In->Coord.X[10:1]));
            }else par{
               PalPL1RAMSetReadAddress(PL1RAM,0@((unsigned 10)In->Coord.Y[9:0]@
                                                  (unsigned 10)In->Coord.X[10:1]));
            }
         }else
            delay;
         if (Stage1.Valid && Stage1.Active)par{
            if (InHold->Pixel.U8[PXS_M])par{
               PalPL1RAMWrite(PL1RAM,0@Stage1.Pixel.U8[PXS_M]);
               read_data = 0@Stage1.Pixel.U8[PXS_M];
             }else par{
             PalPL1RAMRead(PL1RAM,&ram_data);
               read_data = ram_data[31:0];
             }
         }else
            delay;
         if (Out->Valid && Out->Active)par{
            Out->Pixel.U8[PXS_M] = read_data<-8;
         }else
            delay;
      }
   }
}
END_FILTER
```

Spis skrótów

Skrót	Opis	Odwołanie
API	ang. Application Programming Interface	strona 42
ASIC	ang. Application Specific Integrated Circuit	strona 13
ALU	ang. Arithemtic-Logic Unit	strona 30
AGU	ang. Address Generation Unit	strona 28
ANN	ang. Artificial Neural Network	strona 60
BlockRAM	pamięć blokowa wewnątrz układu FPGA	strona 66
CCM	ang. Custom Computing Machine	strona 28
CFSM	ang. Concurrent Finite State Machine	strona 15
CMP	ang. Chip Multi-Processor	strona 15
CFA	ang. Color Filter Array	strona 46
DLP	ang. Data Level Parallelism	strona 15
DSP	ang. Digital Signal Processing	strona 5
DVI	ang. Digital Video Interface	strona 113
FIFO	ang. First In First Out	strona 74
FPGA	ang. Field Programmable Gate Array	strona 5
GAG	ang. Generic Address Generator	strona 28
GPU	ang. Graphics Prcessing Unit	strona **??**
HPC	ang. High Performance Computing	strona 28
HSV	ang. Hue Saturation Value	strona 46
ILP	ang.Instruction Level Parallelism	strona 15
IP	ang. Intellectual Property	strona 119
LUT	ang. Look Up Table	strona 55
MAC	ang. Multiply and ACcumulate	strona 88
MIMD	ang. Multiple Instruction Multiple Data	strona 14
MISD	ang. Multiple Instruction Single Data	strona 14
MMX	ang. MultiMedia eXtension	strona 21
NTSC	ang. National Television System Committee	strona 42
OLP	ang. Operation Level Parallelism	strona 15
PC	ang. Personal Computer	strona 42
PDD	ang. Procedurally Data-Driven architecture	strona 27
PLD	ang. Programmable Logic Device	strona 16

Skrót	Opis	Odwołanie
rALU	ang. Reconfigurable Arithemtic-Logic Unit	strona 28
RGB	ang. Red Green Blue	strona 46
RISC	ang. Reduced Instruction Set Computing	strona 64
ROI	ang. Region of Interest	strona 26
PAL	ang. Phase Alternating Line	strona 41
PDK	ang. Platform Development Kit	strona **??**
SAD	ang. Sum of Absolute Differences	strona 98
SIMD	ang. Single Instruction Multiple Data	strona 14
SISD	ang. Single Instruction Single Data	strona 14
SoC	ang. System on Chip	strona 13
SSE	ang. Streaming SIMD Extension	strona 21
SXGA	ang. Super eXtended Graphics Array	strona 41
TLP	ang. Task Level Parallelism	strona 15
VLIW	ang. Very Long Instruction Word	strona 15
YCrCb	Y-luminancja; Cr, Cb- sygnały różnicowe: czerwony i niebieski	strona 46
ZBT	ang. Zero Bus Turnaround	strona 109

Spis rysunków

6.1 Schemat blokowy czujnika wizyjnego MT9V022. 43
6.2 Transmisja pikseli obrazu. 44
6.3 Organizacja ramki obrazu podczas skanowania liniowego. 44
6.4 Czasowe parametry w transmisji ramki obrazu. 45
6.5 Konfiguracje komórek filtrów w kolorowym czujniku wizyjnym. 46
6.6 Raster obrazu w formacie RGB 47
6.7 Strumieniowa interpolacja barwnego sygnału wizyjnego. 48
6.8 Etapy interpolacji obrazu testowego 50
6.9 Etapy przetwarzania w strumieniowym systemie rozpoznawania znaków. 52
6.10 Kontekst przestrzenny w indeksacji. 54
6.11 Wzorzec do szacowania liczby etykiet w obrazie binarnym. 56
6.12 Orientacja punktów próbkowania względem rastra obrazu. 59
6.13 Normalizacja obiektu przez próbkowanie w obszarze ROI. 60
6.14 Struktura sieci neuronowej do rozpoznawania znaków. 61
6.15 Model sztucznego neuronu. 62
6.16 Funkcja przejścia oraz jej aproksymacja. 66
6.17 Etapy zrównoleglenia segmentacji. 71
6.18 Kontekst przestrzenny w pierwszym etapie indeksacji. 75
6.19 Schemat przepływu danych w systemie rozpoznawania znaków. 77
6.20 Potok gruboziarnisty w systemie rozpoznawaniu znaków. 79
6.21 Tor rozpoznawania jako potok gruboziarnisty. 80
6.22 Etapy zrównoleglenia w architekturze potokowej. 82
6.23 Przetwarzanie potokowe w drugim etapie zrównoleglenia. 85
6.24 Wykres metryki Karp-Flatt'a w zrównoleglaniu sieci neuronowej. 90
6.25 Czasy wykonania w sekwencyjnym torze wizyjnym. 91
6.26 Czasy wykonania w zrównoleglonym torze wizyjnym. 92
6.27 Kompensacja czasów wykonania w potokowej realizacji toru wizyjnego. 93

7.1 System akwizycji sygnału wizyjnego ze skrzyżowania. 97
7.2 Topologia pętli indukcyjnych i obszarów wideodetekcji. 99
7.3 Schemat przepływu strumieni danych w algorytmie wideodetekcji 100
7.4 Próba zastosowania potoku - konflikt w odwołaniach do pamięci. 104
7.5 Transformacja potoku przy ograniczeniach pamięciowych. 107
7.6 Potokowa detekcja obecności. 110
7.7 Zrównoleglenie wideodetektora przy ograniczeniach pamięciowych. 111
7.8 Przyspieszenie i przepustowość zrównoleglonego wideodetektora. 113

7.9 Prototyp przepływowego wideodetektora 114

B.1 Wykres przyspieszenia potoku 138
B.2 Wykres efektywności potoku 139

Spis tabel

4.1 Klasyfikacja operacji wizyjnych. 24

6.1 Czasowe parametry czujnika wizyjnego 45
6.2 Efektywność wykorzystania zasobów obliczeniowych w interpolacji 49
6.3 Rozmiary danych w systemie rozpoznawania znaków. 67
6.4 Operacje elementarne w segmentacji. 70
6.5 Ocena zrównoleglenia segmentacji. 72
6.6 Zużycie zasobów w zrównoleglaniu segmentacji. 74
6.7 Czasy wykonania operacji w systemie rozpoznawania znaków. 78
6.8 Ocena zrównoleglenia systemu rozpoznawania znaków. 84
6.9 Szacowanie zasobów modelu sieci neuronowej. 87
6.10 Ocena zrównoleglenia modelu sieci neuronowej. 89

7.1 Rodzaj kontekstu a zapotrzebowanie na zasoby pamięciowe. 96
7.2 Zapotrzebowanie na zasoby pamięciowe w wideodetektorze. 103
7.3 Zasoby pamięciowe w dostępnych układach reprogramowalnych. 103
7.4 Transformacja równoległości w wideodetektorze. 106
7.5 Ocena zrównoleglenia podczas transformacji potoku. 108
7.6 Ocena zrównoleglenia wideodetektora. 112
7.7 Zużycie zasobów w wideodetektorze. 114

B.1 Parametry potoku 138