Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ

AUTOREFERAT ROZPRAWY DOKTORSKIEJ

MATEUSZ KOMORKIEWICZ

AKCELERACJA ALGORYTMÓW DETEKCJI, ANALIZY I KLASYFIKACJI OBIEKTÓW NA PODSTAWIE STRUMIENIA WIDEO W UKŁADACH REPROGRAMOWALNYCH

PROMOTOR:
dr hab. inż. Marek Gorgoń,
prof. n. AGH

Kraków 2014

1. Wprowadzenie

Systemy wizyjne są coraz częściej wykorzystywane jako źródło sprzężenia zwrotnego w automatycznych układach regulacji. Dzieje się tak, ponieważ rejestracja i analiza obrazu pozwala na bezkontaktowe zebranie informacji o stanie obiektu. W wielu przypadkach jest to jedyny sposób na uzyskanie danych potrzebnych do wypracowania sterowania. Jako przykład takiego wykorzystania systemów wizyjnych można wymienić: bezdotykową inspekcję obiektów na taśmie produkcyjnej, sterowanie autonomicznymi pojazdami lub kontrolę nad robotami wykorzystywanymi na sali operacyjnej.

W odróżnieniu od standardowych czujników, takich jak np. termopara czy czujnik ciśnienia, gdzie mierzona wartość napięcia w bezpośredni sposób odpowiada rzeczywistej wartości mierzonego parametru, uzyskanie interesującej informacji w systemach wizyjnych wymaga przetworzenia danych wejściowych na zupełnie innym poziomie.

Obraz cyfrowy jest przechowywany jako zbiór liczb, które dokładnie i jednoznacznie określają jasność lub kolor pikseli. Taki sposób reprezentacji nie pozwala jednak na automatyczne rozpoznanie tego, co znajduje się na obrazie. Informacja ta jest natomiast konieczna w celu stworzenia zaawansowanych systemów wizyjnych. Procedura umiejscowienia, identyfikacji i analizy obiektów, którą ludzki mózg wykonuje w ułamkach sekundy, w systemach cyfrowych ciągle pozostaje otwartym zagadnieniem badawczym. Co więcej, stosunkowo rzadko może być wykonana w czasie rzeczywistym.

W związku z tym detekcja, analiza i klasyfikacja obiektów jest ciągle dynamicznie rozwijającą się dziedziną badań naukowych. Świadczy o tym duża liczba odbywających się co roku konferencji poświęconych tej tematyce. Są to m. in. IEEE Conference on Computer Vision and Pattern Recognition, International Conference on Image Analysis and Recognition, International Conference on Image Processing, Computer Vision and Pattern Recognition i inne. Ważność tematu potwierdzają również wysokie nakłady finansowe na rozwój tych zagadnień oraz liczne projekty badawcze [Bubliński i inni, 2011],[INDECT, 2010]. Wyrazem zainteresowania są również konkursy o zasięgu międzynarodowym, w których różne algorytmy mają za zadanie możliwie najdokładniej wykryć obiekty znajdujące się na obrazie [Everingham i inni, 2012], [Fiscus i inni, 2009].

Pomimo wielu lat badań, aktualnie dostępne algorytmy nie potrafią poradzić sobie z rozpoznaniem i analizą obiektów w każdych warunkach oświetlenia lub w dowolnych położeniach kamery. Co więcej algorytmy, które charakteryzują się wysoką skutecznością, wymagają wykonania tak dużej ilości obliczeń, że nie ma możliwości ich wykorzystania w systemach czasu rzeczywistego. W przypadku części zastosowań, dozwolone jest czekanie na wynik działania algorytmu (wykonanie operacji off-line). Istnieją jednak takie aplikacje, gdzie cały proces powinien być wykonywany z klatki na klatkę w taki sposób, aby system nie przeoczył ważnych wydarzeń i mógł na nie odpowiednio szybko zareagować [Komorkiewicz, 2013]. Dobrym przykładem jest konieczność natychmiastowego wykrycia wtargnięcia pieszych na jezdnię przy sterowaniu pojazdem autonomicznym.

Z drugiej strony, można zaobserwować ciągły rozwój platform sprzętowych, na których mogą być realizowane wyżej wymienione algorytmy [Kryjak i inni, 2011]. Procesory ogólnego przeznaczenia, DSP, układy GPU czy układy reprogramowalne są coraz szybsze i posiadają coraz więcej zasobów. Zauważalna jest tendencja do zwiększania mocy obliczeniowej głównie poprzez zwiększanie liczby elementów, które równolegle wykonują operacje na danych (instrukcje SSE, AVX, potoki renderujące, elementy logiczne). Spośród wymienionych wyżej rozwiązań właśnie układy reprogramowalne, umożliwiają osiągnięcie najwyższego stopnia zrównoleglenia przetwarzania danych. Jednakże ich efektywne wykorzystanie wymaga dogłębnej wiedzy o architekturze platformy, jak również o naturze implementowanego algorytmu.

Celowym wydaje się więc prowadzenie badań w kierunku możliwości akceleracji algorytmów detekcji, klasyfikacji i analizy obiektów w układach reprogramowalnych. Z jednej strony polegają one na takich modyfikacjach oryginalnych algorytmów lub tworzeniu nowych, aby była możliwa ich równoległa implementacja w układach FPGA. Z drugiej strony, badane są nowe architektury sprzętowe, które umożliwiają akcelerację już istniejących algorytmów [Wiatr, 2003],[Gorgoń, 2007].

W ramach niniejszej pracy, przedstawiono kilka implementacji wybranych algorytmów detekcji, klasyfikacji i analizy obiektów na obrazie w układach reprogramowalnych. Zaproponowane nowe architektury sprzętowe zostały dokładnie przebadane pod kątem ilości używanych zasobów logicznych oraz maksymalnej częstotliwości pracy. Dokładnie sprawdzono również wpływ zastosowanych rozwiązań oraz wybranych reprezentacji liczbowych na zmianę dokładności implementowanych algorytmów i uzyskane rezultaty.

Zaproponowane architektury sprzętowe umożliwiły wykonywanie algorytmów detekcji, analizy i klasyfikacji obiektów na ciągłym strumieniu danych wideo. Dzięki temu, możliwe jest tworzenie zaawansowanych systemów wizyjnych, które są w stanie analizować strumień obrazu w czasie rzeczywistym.

1.1. Teza i cel rozprawy

We wstępie wskazano na problemy związane z dużą liczbą operacji, które muszą być wykonane w celu realizacji algorytmów detekcji, analizy i klasyfikacji obiektów. Wraz ze wzrostem rozdzielczości obrazów jak również liczbą klatek, które muszą być przetwarzane, aby możliwa była ciągła praca na strumieniu wideo, zagadnienie to jeszcze bardziej się komplikuje. Podkreśla się również, konieczność zachowania reżimów czasu rzeczywistego przy realizacji tych algorytmów w wielu systemach wizyjnych oraz fakt, że procesory ogólnego przeznaczenia nie są w stanie zapewnić tego warunku dla części algorytmów. Różnice pomiędzy procesorami ogólnego przeznaczenia, a układami reprogramowalnymi, które umożliwiają dużo większe zrównoleglenie wykonywanych operacji, jak również lepsze dostosowanie architektury do wykonywania konkretnych zadań, prowadzą do sformułowania tezy rozprawy:

Sprzętowa akceleracja algorytmów detekcji, analizy i klasyfikacji obiektów na podstawie strumienia wideo, pozwoli na wykonywanie tych zaawansowanych algorytmów w czasie rzeczywistym.

W celu udowodnia tezy, w kolejnych rozdziałach przedstawiono szereg implementacji sprzętowych wybranych algorytmów detekcji, analizy i klasyfikacji obiektów. Starano się wybrać algorytmy, które są istotne i często wykorzystywane w praktyce przetwarzania obrazów. Celem pracy było takie ich przyśpieszenie względem istniejących implementacji sprzętowych lub programowych, aby możliwe było przetwarzanie strumienia wideo o jak najwyższej rozdzielczości i liczbie klatek na sekundę. Aby zweryfikować zaprojektowane moduły w rzeczywistych zastosowaniach, zaprezentowane zostały trzy kompletne systemy wizyjne, realizujące funkcje detekcji, analizy i klasyfikacji obieków na podstawie strumienia wideo.

1.2. Zawartość pracy

Rozprawa składa się z sześciu rozdziałów oraz jednego dadatku, poświęconych akceleracji algorytmów detekcji, analizy i klasyfikacji obiektów na podstawie strumienia wideo w układach reprogramowalnych. W kolejnych rozdziałach zostały omówione następujące zagadnienia:

Rozdział 1. Wstęp: przedstawiono istotę systemów wizyjnych w automatyce. Omówiono problemy związane z wykorzystaniem informacji wizyjnej jak również powody, dla których konieczna jest akceleracja algorytmów detekcji, analizy i klasyfikacji obiektów. W rozdziale tym wskazane zostały różnice pomiędzy układami reprogramowalnymi, a procesorami ogólnego

przeznaczenia GPP. Dodatkowo omówiono różne aspekty akceleracji algorytmów. Przedstawiono również problemy związane z przetwarzaniem obrazów w postaci strumienia wideo.

Rozdział 2. Akceleracja algorytmu obliczania przepływu optycznego metodą Horna-Schuncka: W rozdziale zaprezentowano sprzętową implementację algorytmu obliczania przepływu optycznego metodą Horna-Schuncka. Zaproponowano w pełni potokową architekturę, która umożliwia osiągnięcie bardzo wysokiej wydajności przetwarzania danych wizyjnych. Przedstawiono również metodologię, która pozwoliła na zmierzenie wpływu przyjętej reprezentacji stałoprzecinkowej zarówno na dokładność obliczanego przepływu optycznego, jak i na jakość detekcji (segmentacji) obiektów ruchomych na obrazie. Ponadto w rozdziale przedstawiono wyniki weryfikacji sprzętowej. Zaprezentowano działający na karcie ewaluacyjnej system, który w czasie rzeczywistym oblicza przepływ optyczny dla strumienia wizyjnego z kamery wysokiej rozdzielczości.

Rozdział 3. Akceleracja algorytmów analizy obiektów pierwszego planu: W rozdziale zaprezentowano dwie architektury sprzętowe realizujące jednoprzebiegową analizę obiektów pierwszego planu. Charakteryzują się w pełni potokową pracą i nie wymagają zewnętrznej pamięci RAM do przechowywania danych. Zaprojektowane moduły umożliwiają wyznaczenia takich parametrów jak:

– określenie liczby segmentów maski binarnej, ich pola powierzchni, liczby pikseli brzegowych oraz położenia i rozmiarów prostokąta ograniczającego,

– obliczenie deskryptora teksturowego GLCM dla obszarów obrazu, którym odpowiadają kolejne segmenty maski binarnej.

Zaprezentowano również wyniki sprzętowej weryfikacji zaprezentowanego modułu indeksacji i obliczania deskryptora GLCM na karcie ewaluacyjnej z układem Virtex 6 firmy Xilinx.

Rozdział 4. Metoda zmiany kolejności danych dla szybkiego obliczania histogramów blokowych:

W rozdziale zaprezentowano nowatorską metodę zmiany kolejności danych w strumieniu wizyjnym, która umożliwia zaprojektowanie architektur sprzętowych pracujących z bardzo wysoką częstotliwością. Omówiono jak różne konfiguracje latencji sumatorów i pamięci BRAM wpływają na czasy odczytu i przetwarzania danych. Przedstawiono również architekturę sprzętową, która pozwala na wydajną, potokową zmianę kolejności danych. Zaprezentowano przykład zastosowania opisanej metody do rozwiązania problemu obliczania histogramów blokowych dla pola przepływu optycznego. Badania przeprowadzono zarówno dla elementów obliczeniowych działających na zmienno- jak i stałoprzecinkowej reprezentacji danych.

Rozdział 5. Akceleracja algorytmu detekcji i klasyfikacji obiektów Histogram of Oriented Gradients (HOG):

W rozdziale przedstawiono sprzętową implementację algorytmu *Histogram of Oriented Gradients*, który jest szczególnie chętnie wykorzystywany w systemach wizyjnych do wykrywania obiektów na obrazach. W pierwszej kolejności omówiono oryginalny algorytm. W dalszej kolejności przebadano, jak modyfikacje różnych aspektów obliczeń wpływają na uzyskane rezultaty. Ponadto sprawdzono, jak implementacja poszczególnych kroków w reprezentacji stałoprzecinkowej wpływa na zmianę wyników detekcji. Zaproponowano kilka różnych architektur sprzętowych, które pozwalają uzyskać zarówno dużą dokładność wykrywania obiektów, jak również wysoką wydajność przetwarzania danych. Zaprezentowano moduł w pełni zgodny z wynikami uzyskiwanymi przez implementację algorytmu HOG w bibliotece OpenCV. Dokonano również jego weryfikacji sprzętowej na karcie ewaluacyjnej ML605 z układem Virtex 6 firmy Xilinx.

Rozdział 6. Podsumowanie: W rozdziale zawarto podsumowanie prac opisanych w rozprawie oraz omówiono uzyskane wyniki. Podano wykaz publikacji, które powstały na podstawie przedstawionych badań. Zaprezentowano również kierunki dalszych prac nad akceleracją algorytmów detekcji analizy i klasyfikacji obiektów.

Dadatek A: W dodatku przedstawiono inżynierski aspekt niniejszej rozprawy. Zaprezentowano karty ewaluacyjne, które były wykorzystywane do implementacji algorytmów detekcji, analizy i klasyfikacji obiektów. Opisano również implementacje modułów pomocniczych, które umożliwiały komunikację karty z otoczeniem i wydajny dostęp do zewnętrznej pamięci RAM.

2. Wybrane wyniki

2.1. Obliczanie przepływu optycznego

Czasy obliczania przepływu optycznego dla obrazów o różnych rozdzielczościach w układzie FPGA serii Virtex 7, w zależności od konfiguracji modułów, zostały przedstawione w tabeli 2.1. Ponieważ w architekturze sprzętowej, każda iteracja algorytmu jest realizowana przez osobne, dodatkowe moduły sprzętowe, wykonanie kolejnych iteracji nie wydłuża czasu obliczeń. Wpływa na niego jedynie rozmiar obrazu oraz maksymalna częstotliwość pracy uzyskanej architektury.

W tej samej tabeli, przedstawiono również czasy obliczeń dla procesora Core i7 2600K (3,4 GHz) obliczającego przepływ optyczny metodą Horna-Schuncka przy wykorzystaniu funkcji dostępnych w bibliotece OpenCV. Procesor ten ma wsparcie dla instrukcji AVX pozwalającą na przetwarzanie do 8 zmiennych pojedynczej precyzji w tym samym czasie.

Jeśli przyjmiemy, że przez przetwarzanie obrazu w czasie rzeczywistym, należy rozumieć zdolność systemu do przetwarzania 60 klatek na sekundę, to czas obliczeń dla pojedynczej

	640×480			1280×1024			1920×1080		
Iteracje	CPU	FPGA	Prz.	CPU	FPGA	Prz.	CPU	FPGA	Prz.
1	5,8	1,63	3,6×	25,44	7,00	3,6×	38,62	9,63	4,0×
2	5,56	1,73	3,2×	27,61	7,41	3,7×	50,23	10,19	4,9×
4	9,08	1,76	5,2×	37,6	7,56	5,0×	72,71	10,40	7,0×
8	15,92	1,78	8,9×	66,88	7,63	8,8×	119,09	10,49	11,4×
16	30,2	1,64	18,4×	133,71	7,03	19,0×	214,53	9,67	22,2×
32	56,79	2,17	26,2×	244,37	9,33	26,2×	399,97	12,82	31,2×
64	116,72	2,24	52,1×	498,43	9,63	51,8×	769,61	13,24	58,1×
128	216,61	2,40	90,3×	953,75	10,29	92,7×	1509,44	14,14	106,7×

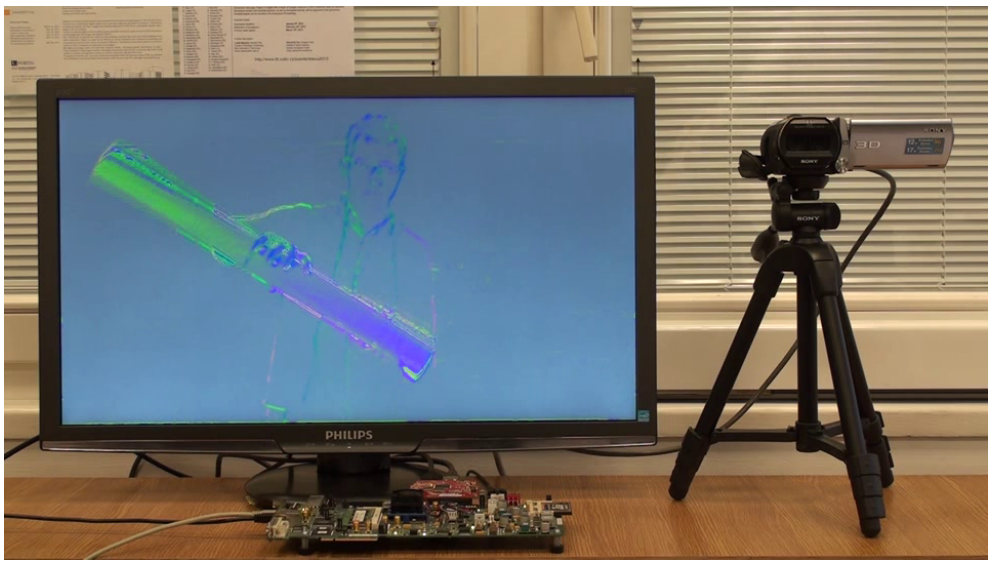
Tabela 2.1: Czas wyznaczania przepływu optycznego w milisekundach dla układu FPGA i procesora CPU w zależności od rozmiaru obrazu i liczby iteracji algorytmu oraz uzyskane przyspieszenie

klatki obrazu musi być mniejszy niż 16,6 ms. Komputer z procesorem Core i7, z włączonym wsparciem dla instrukcji SIMD potrafi wykonać maksymalnie od 8 do 9 iteracji na obrazie VGA w czasie rzeczywistym.

Warto zwrócić uwagę, że w przypadku układu FPGA przetworzenie obrazu HD o rozdzielczości 1920×1080 pikesli wymaga jedynie 14,14 ms. Przetworzenie tego samego obrazu na najszybszej z badanych architektur procesorowych (Core i7) wymaga 1509,44 ms. Uzyskane przyspieszenie wynosi w tym przypadku 107×. Dzięki tak znacznej akceleracji, system jest w stanie przetwarzać strumień wideo w czasie rzeczywistym.

W celu umożliwienia weryfikacji sprzętowej modułów do obliczania przepływu optycznego, zaprojektowano kompletny system wizyjny, który został zrealizowany na karcie ewaluacyjnej. Założono, że powinien on umożliwiać obliczanie przepływu optycznego dla strumienia wizyjnego odbieranego z kamery wysokiej rozdzielczości. Na rysunku 2.1 zostało przedstawione zdjęcie, prezentujące działanie opisywanego systemu. Kamera z wyjściem HDMI jest źródłem obrazu, który jest przetwarzany przez kartę z układem reprogramowalnym ML605 firmy Xilinx [Xilinx, 2012]. Obliczony przepływ jest przekształcany na odpowiednią mapę kolorów i wyświetlany na ekranie monitora.

Na rysunku widoczny jest zarys osoby, która obraca podłużnym przedmiotem (tuba plakatowa) dookoła osi przechodzącej przez jego środek. W ten sposób, punkty które leżą na prawo od osi, przesuwają się w przeciwną stronę niż punkty na lewo od osi (np. jedne w górę, drugie w dół), dodatkowo ponieważ jest to ruch obrotowy, punkty leżące dalej od osi obrotu posiadają większą prędkość liniową niż punkty bliżej osi. Obserwując uważnie przedstawiony rysunek, widać, że przepływ został obliczony poprawnie. Jedna część tuby jest zielona, podczas gdy druga jest niebieska, co świadczy o różnych zwrotach wektora prędkości. Dodatkowo można



Rysunek 2.1: Działający system vizyjny obliczający przepływ optyczny

zauważyć, że punkty leżące bliżej środka obrotu posiadają jaśniejszy kolor, co świadczy o tym, że mają mniejszą prędkość.

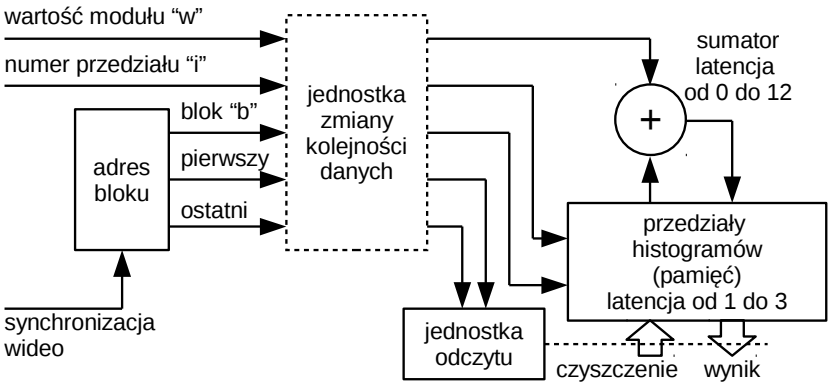
2.2. Zmiana kolejności danych

Sprzętowa architektura do obliczania histogramów z bloków pola wektorowego przepływu optycznego została zaprezentowana na rysunku 2.2. Maksymalny rozmiar obsługiwanego obrazu został ustalony na 1920×1088 pikseli, co pozwala na przetwarzanie obrazu o rozdzielczości Full HD (dodatkowe 8 pikseli w pionie wynika ze stosowania bloków 16×16 pikseli).

W klasycznym przypadku obliczania histogramu (gdy latencja sumatora wynosi 0, a latencja dostępu do pamięci 1) nie jest wykorzystywana jednostka zmiany kolejności danych i wartości wejść są przesyłane bezpośrednio do modułu obliczającego wartość histogramu. Jeśli suma latencji sumatora i dostępu do pamięci przekracza jeden cykl zegara, konieczne jest wykorzystanie jednostki zmieniającej kolejność danych w celu oddzielenia następujących po sobie próbek, które należą do tych samych bloków.

Na etapie obliczania histogramu przetestowano wiele konfiguracji. W pierwszej kolejności przebadano zarówno stało-, jak i zmiennoprzecinkowe sumatory. Latencja była zmieniana od 0 do 2 cykli zegara dla sumatorów stałoprzecinkowych. Dla sumatorów zmiennoprzecinkowych przetestowano konfiguracje z latencją 0, 1, 4, 8, 12 cykli zegara. Dodatkowo przebadano 2 konfiguracje sumatorów: jedną – opartą na elementach logicznych LUT6, drugą – wykorzystującą bloki DSP48, dedykowane do operacji arytmetycznych. Sprawdzono także wiele możliwych konfiguracji pamięci. Latencja była ustawiana na 1, 2 (rejestry typu core lub primitive) lub 3 (pełne rejestry wyjściowe) cykle zegara.

Możliwość różnych opcji konfiguracji doprowadziły do stworzenia i przetestowania 58 różnych architektur sprzętowych. Wyniki testowania dla wybranych architektur zostały przedstawione w tabelach 2.2 i 2.3. W każdej z nich zaprezentowano najważniejsze parametry roz-



Rysunek 2.2: Sprzętowa architektura do szybkiego obliczania histogramu

ważanych architektur. Latencje sumatora i pamięci zostały podane w 2 pierwszych wierszach. W dalszej kolejności podano zużycie zasobów. Poszczególne skróty oznaczają: FF (*flip-flops*) – przerzutniki, LUT 6 (*look up table*) – 6-wejściowe tablice logiczne z jednym wyjściem, SLICE – komórki typu *slice* układu FPGA, BRAM 18 i 36 – wewnętrzne pamięci blokowe o rozmiarze 2 kB i 4kB [Xilinx, 2011], DSP48 – dedykowane bloki arytmetyczne składające się z sumatora i mnożarki. Podano również maksymalne częstotliwości pracy otrzymanych architektur uzyskane symulacyjnie w programie Xilinx ISE DS po fazie *place and route* dla układu XC6VLX240T z serii Virtex 6.

W drugiej kolumnie każdej z tabel znajduje się konfiguracja z latencją sumatora równą 0 i latencją dostępu do pamięci równą 1. Jest to jedyna konfiguracja, która do poprawnego działania nie wymaga jednostki zmiany kolejności danych (klasyczna konfiguracja modułu obliczania histogramu). Konfiguracja ta jest wykorzystywana jako punkt odniesienia do pomiarów uzyskanego przyspieszenia. Została podana w celu pokazania, w jak znacznym stopniu użycie jednostki zmiany kolejności danych pozwala poprawić maksymalną częstotliwość pracy modułu obliczającego histogramy.

W tabeli 2.2 przedstawiono rezultaty testowania architektury do stałoprzecinkowego obliczania histogramu (32 bity). Warto zauważyć, że klasyczna architektura pozwala na osiągnięcie maksymalnej częstotliwości pracy równej 170 MHz.

Dzięki zastosowaniu jednostki zmieniającej kolejność danych możliwe jest wykorzystanie elementów przetwarzających o większej latencji. Wydłużenie latencji z 0 do 1 cyklu zegara pozwala zwiększyć maksymalną częstotliwości pracy o około 40% przy tej samej konfiguracji pamięci. Z kolei wykorzystanie pamięci o większej latencji dostępu (dodatkowe buforowanie odczytu) umożliwia zwiększenie częstotliwości od 27% do ponad 68%. Konfiguracja, w której sumator ma latencję równą jednemu cyklowi zegara, a odczyt pamięci wprowadza latencję 3

Latencja pamięci	1	1	2	2	3	3
Latencja sumatora	0	1	0	1	0	1
FF	92	237	205	237	493	525
LUT 6	114	150	182	179	242	220
SLICE	47	62	64	65	212	242
BRAM 18	0	0	0	0	0	0
BRAM 36	8	9	9	9	9	9
DSP48	0	0	0	0	0	0
MAX CLK	170 MHz	236 MHz	216 MHz	329 MHz	276 MHz	396 MHz
Przyspieszenie	1×	1,4×	1,3×	1,9×	1,6×	2,3×

Tabela 2.2: Architektury do obliczania histogramów ze stałoprzecinkową reprezentacją danych

cykle zegara, pozwala na osiągnięcie maksymalnej częstotliwości pracy wynoszącej 396 MHz. W stosunku do klasycznej architektury jest to przyspieszenie o ponad 2,3 raza.

W dalszej części eksperymentów, 32-bitowy sumator stałoprzecinkowy został zamieniony na sumator zmiennoprzecinkowy pojedynczej precyzji. Latencja sumatora zmiennoprzecinkowego może wynosić od 0 do 12 cykli zegara. W czasie testów przebadane zostały konfiguracje z latencją 0, 1, 4, 8 i 12 cykli zegara. Okazało się, że najlepsze rezultaty uzyskano dla sumatora pracującego z maksymalną latencją (12 cykli zegara). Jak zostało pokazane w tabeli 2.3, architektura, która nie wykorzystuje jednostki zmiany kolejności danych i posiada łączną latencję równą jednemu cyklowi zegara, może pracować z maksymalną częstotliwością równą 61 MHz. W efekcie dodania jednostki zmieniającej kolejność danych i zwiększenia latencji sumatora do 12 cykli zegara maksymalna częstotliwość pracy wzrasta czterokrotnie – do 252 MHz. Skonfigurowanie pamięci do pracy z większą latencją pozwala na dalszą poprawę tego wyniku. Najlepsza z architektur, której łączna latencja wynosi 15 cykli zegara, może pracować z maksymalną częstotliwością równą 383 MHz, co jest ponad 6,3 raza lepszym wynikiem w porównaniu do klasycznej architektury.

Latencja pamięci	1	1	2	2	3	3
Latencja sumatora	0	12	0	12	0	12
FF	92	746	205	746	493	1034
LUT 6	465	509	520	557	568	585
SLICE	149	211	177	195	321	401
BRAM 18	0	0	0	0	0	0
BRAM 36	8	9	9	9	9	9
DSP48	0	0	0	0	0	0
MAX CLK	61 MHz	252 MHz	65 MHz	369 MHz	71 MHz	383 MHz
Przyspieszenie	1×	4,1×	1,1×	6,0×	1,2×	6,3×

Tabela 2.3: Architektury obliczające histogram z precyzją zmiennoprzecinkową

2.3. Wykrywanie obiektów algorytmem HOG

W celu porównania wydajności przeprowadzono pomiary czasu obliczania algorytmu HOG na komputerze wyposażonym w procesor Core i7 2600 (3,4 GHz). Program był skompilowany w 2 wersjach: z wykorzystaniem instrukcji SIMD i bez nich. W tym przypadku były to instrukcje AVX pozwalające na wykonywanie tej samej operacji na 8 wartościach pojedynczej precyzji jednocześnie. Obliczono również, ile czasu potrzeba na ukończenie tej samej operacji

Implementacja	HOG	Przyspieszenie
CPU bez SIMD	89,21 ms	0,41 ×
CPU z SIMD	36,59 ms	1,0 ×
FPGA ZP	16,03 ms	2,28 ×
FPGA SP v. 0	3,16 ms	11,58 ×
FPGA SP v. 1	2,44 ms	15,00 ×
FPGA SP v. 2	2,37 ms	15,44 ×

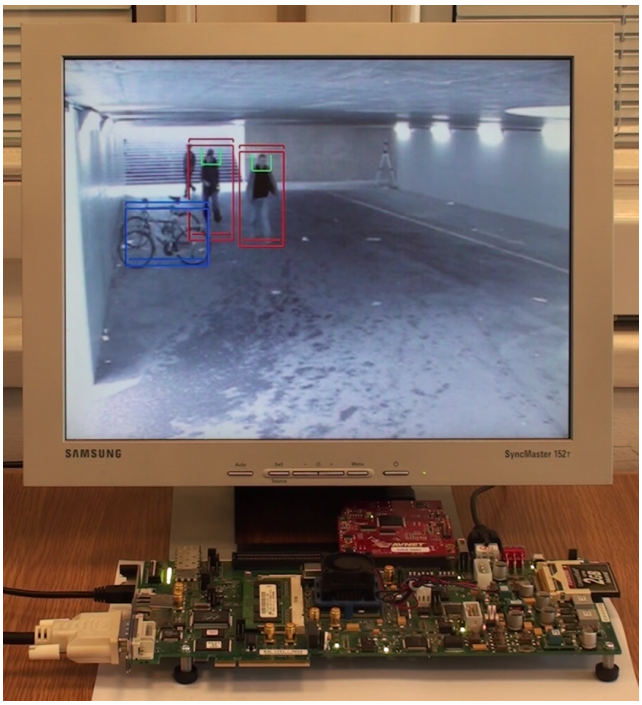
Tabela 2.4: Czas przetwarzania klatki o rozmiarze 640x480 pikseli, przy oknie detekcji równym 7x15 bloków. ZP - moduł zmiennoprzecinkowy, SP - moduł stałoprzecinkowy

w układzie reprogramowalnym FPGA. Do testów przyjęto rozmiar obrazu 640×480 pikseli i rozmiar okna klasyfikacji 7×15 bloków.

Jeśli przyjmie się, że aby mówić o przetwarzaniu strumienia wizyjnego w czasie rzeczywistym, w ciągu jednej sekundy musi zostać przetworzonych 60 klatek (standardowa wartość w monitorach LCD), to przetworzenie jednej ramki powinno zajmować nie więcej niż 16,7 ms. Żadna z implementacji programowych nie jest w stanie spełnić tego wymagania. Oryginalna implementacja z biblioteki OpenCV wykorzystująca instrukcje SIMD, realizowana na wydajnym procesorze, wymaga ponad 36 ms na przetworzenie pojedynczej klatki obrazu. Wersja bez instrukcji SIMD realizuje tę samą operację w czasie ponad 89 ms. W tabeli 2.4 przedstawiono również przyspieszenie, które uzyskano dzięki implementacji algorytmu HOG na platformie FPGA, w porównaniu do najszybszej wersji realizowanej na procesorze CPU. Można zauważyć, że architektura sprzętowa obliczająca oryginalną wersję algorytmu w wersji zmiennoprzecinkowej uzyskuje ponad dwukrotne przyspieszenie i umożliwia przetwarzanie obrazu w czasie rzeczywistym. Z kolei architektury stałoprzecinkowe pozwalają na uzyskanie maksymalnie ponad piętnastokrotnego przyspieszenia względem algorytmu realizowanego na procesorze CPU.

Do weryfikacji sprzętowej wybrano wersję architektury która realizuje najbardziej złożoną wersję algorytmu w zmiennoprzecinkowej reprezentacji danych. Celem opisywanej implementacji było także sprawdzenie, czy układy rekonfigurowalne mają potencjał, aby wykonywać oryginalne wersje algorytmów bez żadnych uproszczeń numerycznych oraz czy mogą konkurować na tym polu z układami CPU i GPU. Zaprezentowany został działający system, który dokonuje klasyfikacji 3 typów obiektów (głowa, człowiek, rower) z oknami detekcji o rozmiarach 32×32, 64×128 i 112×80 pikseli. Przetwarza on strumień wideo o rozmiarach 640×480 pikseli w czasie rzeczywistym (60 klatek na sekundę). Wykonywany jest oryginalny algorytm HOG w wersji z biblioteki OpenCV bez żadnych uproszczeń, które zmniejszałyby jego dokładność.

Opisana implementacja została przetestowana z wykorzystaniem karty ewaluacyjnej



Rysunek 2.3: Działający system realizujący algorytm HOG, przetwarzanie 60 klatek na sekundę o rozdzielczości 640×480

ML605 z układem Virtex 6 VC6VLX240T wyprodukowanej przez firmę Xilinx. Do odbioru strumienia wizyjnego użyto karty rozszerzeń Avnet DVI I/O FMC. Projekt został zweryfikowany przez porównanie wyników implementacji programowej z wynikiem symulacji oraz przez zebranie danych z działającego systemu (wykorzystano logowanie do pamięci BRAM z przesyłaniem danych do komputera z wykorzystaniem jednostki UART). Wyniki przetwarzania sekwencji wideo z bazy danych [Berclaz i inni, 2011] zostały zaprezentowane na rysunku 2.3.

2.4. Podsumowanie

Celem badań przedstawionych w niniejszej rozprawie była akceleracja algorytmów detekcji, analizy i klasyfikacji obiektów w układach reprogramowalnych FPGA. Zagadnienie to jest bardzo istotne w przypadku tworzenia zaawansowanych automatycznych systemów wizyjnych, które wymagają realizacji złożonych obliczeniowo algorytmów. W pracy zaprezentowano szereg metod i architektur pozwalających na wydajną realizację tych zadań w czasie rzeczywistym.

Uzyskane wyniki w pełni potwierdzają tezę rozprawy i pokazują, że układy FPGA są dobrą platformą do implementacji i akceleracji zaawansowanych algorytmów wizyjnych. Umożliwiają one ścisłą integrację sensorów wizyjnych z elementami przetwarzającymi dane w systemach wbudowanych. Ich zastosowania obejmują automatyczny monitoring wizyjny, autonomiczne pojazdy, medyczne systemy obrazowania i wiele innych.

Bibliografia

[Berclaz i inni, 2011] Berclaz, J., Fleuret, F., Turetken, E., i Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(9):1806–1819. (Cytowane na stronie 13)

[Bubliński i inni, 2011] Bubliński, Z., Chmiel, W., Jabłoński, M., Kadłuczka, P., Kryjak, T., Mikrut, Z., Pawlik, P., i Tadeusiewicz, R. (2011). System inteligentnego monitoringu przestrzeni i obiektów szczególnego znaczenia simpoz. W *Pomiary Automatyka Robotyka*, numer 69. (Cytowane na stronie 2)

[Everingham i inni, 2012] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., i Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. (Cytowane na stronie 2)

[Fiscus i inni, 2009] Fiscus, J., Garofolo, J., Rose, T., i Michel, M. (2009). Avss multiple camera person tracking challenge evaluation overview. W *Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on*, strony 219–219. (Cytowane na stronie 2)

[Gorgoń, 2007] Gorgoń, M. (2007). *Architektury rekonfigurowalne do przetwarzania i analizy obrazu oraz dekodowania cyfrowego sygnału wideo*. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne, Kraków. (Cytowane na stronie 3)

[INDECT, 2010] INDECT (Październik 2010). Indect description of system architecture 2.2. The INDECT Consortium. (Cytowane na stronie 2)

[Komorkiewicz, 2013] Komorkiewicz, M. (2013). Real-time detection of movement in prohibited direction for video surveillance system. *Image Processing and Communications*, 17(4):251–264. (Cytowane na stronie 3)

[Kryjak i inni, 2011] Kryjak, T., Komorkiewicz, M., i Gorgon, M. (2011). Real-time moving object detection for video surveillance system in fpga. W *Design and Architectures for Signal and Image Processing (DASIP), 2011 Conference on*, strony 1–8. (Cytowane na stronie 3)

[Wiatr, 2003] Wiatr, K. (2003). *Akceleracja obliczeń w systemach wizyjnych*. Wydawnictwa Naukowo-Techniczne, Warszawa. (Cytowane na stronie 3)

[Xilinx, 2011] Xilinx (2011). *Virtex-6 FPGA Memory Resources, User Guide, UG363 v1.6*. Xilinx, San Jose CA. `www.xilinx.com`. (Cytowane na stronie 10)

[Xilinx, 2012] Xilinx (2012). *UG534 ML605 Hardware User Guide v1.8*. Xilinx, San Jose CA. `www.xilinx.com`. (Cytowane na stronie 8)