AUTOMATYKA • 2011 • Tom 15 • Zeszyt 3

Tomasz Kryjak*, Mateusz Komorkiewicz*, Marek Gorgoń*

Implementacja algorytmu generacji tła wraz z modułem segmentacji obiektów ruchomych i eliminacji cieni w układach FPGA serii Spartan 6**

1. Wprowadzenie

Współcześnie obserwuje się intensywny rozwój wizyjnych systemów nadzoru przestrzeni publicznej. Kamery instalowane są w takich miejscach jak dworce, lotniska, sklepy, budynki administracji, szkoły, uczelnie, muzea, galerie handlowe, a także, coraz częściej, jako element zabezpieczenia domów prywatnych. Od kilku lat coraz większe znaczenie zaczynają odgrywać systemy nadzoru wizyjnego, oparte na cyfrowej transmisji sygnału wizyjnego. Wzrost popularności tego typu rozwiązań związany jest z jednej strony z próbą zapewnienia coraz wyższego poziomu bezpieczeństwa, a z drugiej z malejącymi kosztami instalacji systemów i ułatwieniem, jakim jest technologia kamer IP (transmisja obrazów poprzez sieć Ethernet). Dużym wyzwaniem jest analiza uzyskanej informacji wizyjnej. W przeważającej większości przypadków, analizy dokonuje operator, który podejmuje decyzję o wykrywaniu sytuacji niebezpiecznych na podstawie obserwacji kilku monitorów. Dodatkowo strumienie wizyjne z kamer są zapisywane i archiwizowane w celu ewentualnej późniejszej analizy.

Obecnie trwają prace nad opracowaniem automatycznych algorytmów analizy sekwencji wideo z monitoringu wizyjnego, których zadaniem jest wykrywanie sytuacji niebezpiecznych – na przykład: porzuconych przedmiotów (potencjalne ładunki wybuchowe), przypadków naruszenia strefy zabronionej (np. w muzeum), kradzieży (zaginięcie przedmiotu chronionego), bójek, nagłych zgromadzeń ludzi i podobnych. Wykorzystywane w tych pracach algorytmy, charakteryzują się dużą złożonością obliczeniową i dlatego ich akceleracja sprzętowa jest w wielu przypadkach bardzo pożądana.

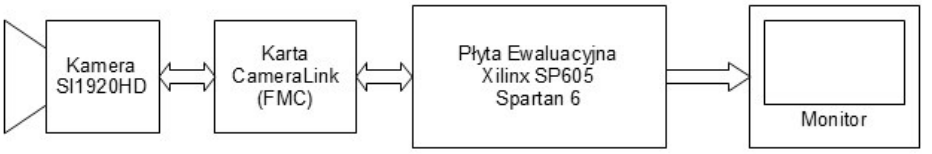
* AGH Akademia Górniczo-Hutnicza, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, Katedra Automatyki, Laboratorium Biocybernetyki, al. A. Mickiewicza 30, 30-059 Kraków

** Praca wykonana w ramach projektu „SIMPOZ" z Ministerstwa Nauki i Szkolnictwa Wyższego nr 0128/R/t00/2010/12

W artykule zaprezentowano system do detekcji obiektów ruchomych zrealizowany z wykorzystaniem zasobów układu rekonfigurowalnego FPGA serii Spartan 6. Podstawą działania systemu jest wykrywanie ruchu za pomocą odejmowania tła z wykorzystaniem operacji generacji tła oraz segmentacji obiektów na podstawie trzech kryteriów: jasności, koloru i tekstury. Ponadto do realizacji systemu konieczne było zaimplementowanie szeregu modułów w układzie FPGA: komunikacji z kamerą, konwersji sygnału kolorowego z matrycy kamery, tzw. transformacji Bayera, konwersji przestrzeni barw RGB do CIE Lab, wydajnej komunikacji z zewnętrzną pamięcią RAM, obsługi monitora oraz zaprojektowanie i wykonanie modułu sprzętowego do podłączenia kamery (standard Camera Link) do układu FPGA. W rozdziale 2 opisano ogólną koncepcję systemu, a w kolejnych rozdziałach od 3 do 7 budowę poszczególnych modułów. Artykuł zakończony jest prezentacją uzyskanych wyników i ich omówieniem.

2. Koncepcja systemu

W trakcie projektowania przyjęto założenie, że system na wejściu będzie pobierał obraz z kamery, a na wyjściu zwracał maskę obiektów ruchomych, która wyświetlana będzie na monitorze LCD. Wszystkie konieczne obliczenia wykonywane będą w zasobach układu FPGA. Podczas prac wykorzystano płytę SP605 firmy Xilinx z układem FPGA Spartan 6 (XC6SLX45T). Schemat systemu zaprezentowano na rysunku 1.



Rys. 1. Schemat systemu detekcji obiektów ruchomych

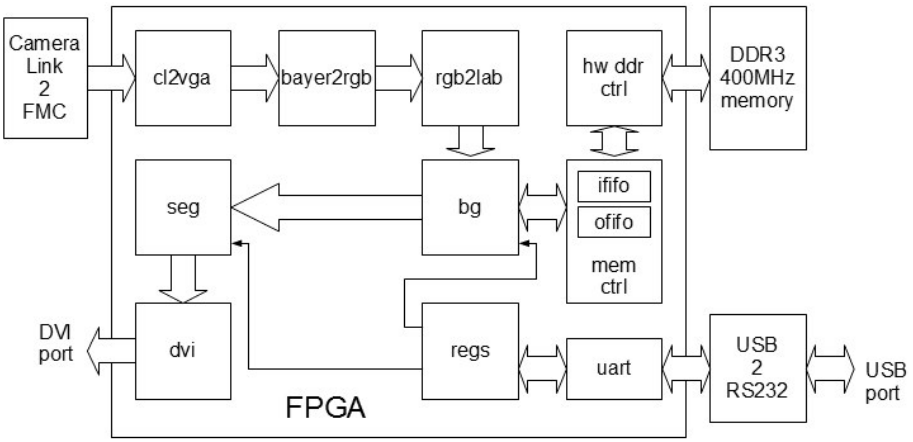
W skład systemu wchodzą następujące elementy:

- kamera SI 1920HD firmy Silicon Imaging z interfejsem Camera Link;
- konwerter magistrali Camera Link na standard FMC (FPGA Mezzanine Card) zaprojektowany i wykonany przez jednego ze współautorów, który umożliwia podłączenie kamery do karty przetwarzającej obraz;
- płyta ewaluacyjna SP605 z układem FPGA Spartan 6 firmy Xilinx, jako zasadnicza platforma obliczeniowa, na której wykonywane są wszystkie operacje związane z przetwarzaniem sygnału wizyjnego;
- monitor LCD do wizualizacji przetworzonego obrazu.

Schemat funkcjonalny systemu zrealizowanego w układzie FPGA przedstawiono na rysunku 2. Składa się on z następujących modułów:

- *cl2vga* – odbiór sygnału z kamery pracującej w standardzie Camera Link i przetworzenie go na standard VGA,

– *bayer2rgb* – odtworzenie koloru – wykonanie tzw. transformacji Bayera,
– *rgb2lab* – konwersja przestrzeni barw z RGB na CIE Lab,
– *bg* – realizacja generacji tła,
– *seg* – realizacja segmentacji obiektów ruchomych,
– *dvi* – konwersja sygnału do postaci możliwej do wyświetlania na monitorze,
– *regs* – rejestry z parametrami algorytmu,
– *uart* – transmisja danych do rejestrów z komputera PC (sterowanie algorytmem),
– *mem ctrl* – kontroler pamięci, wraz z buforami FIFO,
– *hw ddr ctrl* – sprzętowy kontroler pamięci DDR3.



Rys. 2. Schemat funkcjonalny systemu zaimplementowanego w układzie FPGA

3. Przetwarzanie sygnału z kamery

3.1. Odbiór danych z kamery (moduł *cl2vga*)

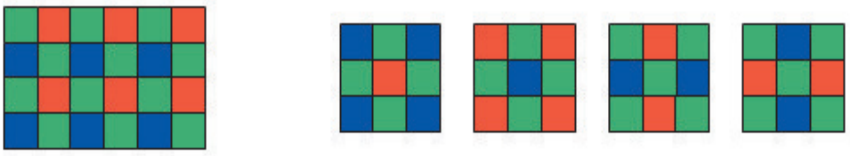
Zadaniem modułu jest odebranie sygnału wizyjnego wysyłanego przez kamerę SI-1920HD-CL transmitującą dane za pomocą protokołu Camera Link. Następnie sygnał ten jest konwertowany do postaci, która może być bezpośrednio wyświetlona na monitorze komputerowym. Wymaga to dopasowania sygnałów synchronizacji pionowej i poziomej (*hsync, vsync*) oraz częstotliwości zegara dla obu urządzeń, a także odpowiedniego ułożenia danych.

Aby możliwe było bezpośrednie wyświetlenie obrazu z kamery na monitorze, konieczna jest taka konfiguracja kamery, w której dostarczany sygnał jest możliwie jak najbardziej zgodny ze standardem VGA. Z całej gamy dostępnych trybów wybrana została rozdzielczość 640×480 przy 60 klatkach na sekundę (zegar transmisji danych w łączu Camera Link wynosi 25 MHz).

Warto zauważyć, że kamera SI-1920HD-CL z powodu ograniczeń magistrali Camera Link nie jest w stanie pracować z zegarem pikseli niższym niż 40 MHz. Ponieważ standard VGA przewiduje częstotliwości pikseli równą 25 MHz, wybrany został tryb pracy z zegarem 50 MHz (co przy rozdzielczości 640×480 odpowiada 120 klatkom na sekundę), z myślą o dalszej redukcji strumienia wizyjnego. W celu dostosowania sygnału do standardu VGA, należało albo odrzucić co drugą ramkę (poprzez buforowanie i odczyt z dwa razy mniejszą szybkością) albo podnieść rozdzielczość do 1280×480 i w tak otrzymanym obrazie odrzucić co drugi piksel. Z uwagi na prostotę implementacji sprzętowej wybrane zostało drugie rozwiązanie. Opisywane rozwiązanie ma charakter badawczo-rozwojowy. W procesie wdrażania, w rozwiązaniach dedykowanych do określonych systemów monitoringu, należy dobrać odpowiednie kamery w zależności od stosowanej rozdzielczości obrazu, co pozwoli uniknąć problemu niedopasowania zegara piksela kamery i urządzenia przetwarzającego oraz parametrów urządzeń wizualizujących i rejestrujących obraz.

3.2. Konwersja kolorów na RGB (moduł *bayer2rgb*)

Czujniki obrazu zarówno CMOS (*Complementary Metal-Oxide-Semiconductor*), jak i CCD (*Charge Coupled Device*) są czułe na zmianę jasności. Uzyskanie obrazu kolorowego wymaga optycznej filtracji składowych barwnych. W kamerach kolorowych spotyka się dwa rozwiązania, które pozwalają na rejestrację obrazu kolorowego: wykorzystanie trzech czujników, każdy z nich odpowiedzialny jest za rejestrację jednej składowej barwnej (np. 3×CCD, 3×CMOS) lub wykorzystanie jednego czujnika, w którym matryca czujnika pokryta jest mozaiką filtrów CFA (*Color Filter Array*). Drugie rozwiązanie, z uwagi na niższe koszty, jest chętniej wykorzystywane. Jedną z pierwszych i nadal stosowanych topologii filtrów jest matryca filtrów Bayera [3]. W matrycy Bayera każda komórka światłoczuła pokryta jest filtrem przepuszczającym światło o określonej barwie (czerwonej, niebieskiej, zielonej). Sposób rozmieszczenia filtrów zaprezentowano na rysunku 3.



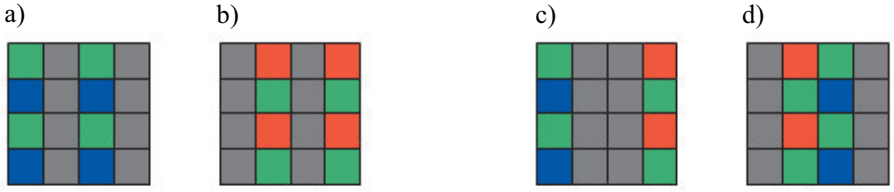
Rys. 3. Siatka Bayera i cztery warianty otoczenia

Zatem dla matrycy o ustalonym rozmiarze, dane o kolorze czerwonym i niebieskim znajdują się tylko w 1/4 wszystkich komórek. Natomiast kolor zielony jest rejestrowany w połowie wszystkich pikseli. W celu uzyskania informacji o wartości trzech kolorów w każdym punkcie matrycy stosowana jest interpolacja.

Interpolacji dokonuje się na podstawie analizy otoczenia o rozmiarze 3×3 dla danego piksela. W pierwszym kroku konieczne jest ustalenie, jaki wariant otoczenia występuje dla

danej lokalizacji (warianty zostały zaprezentowane na rysunku 3). Wartość koloru jest odczytywana bezpośrednio dla piksela o danym kolorze, wartość dwóch pozostałych kolorów jest wyliczana jako średnia z wartości sąsiadów o danym kolorze (tzw. interpolacja dwuliniowa). Zagadnienie odtwarzania koloru na podstawie matrycy Bayera i implementacja potokowa algorytmu opisane zostały szerzej m.in. w pracach [10, 11].

Zgodnie z opisem z punktu 3.1, obraz 1280×480 ma być zredukowany do rozmiaru 640×480 przez odrzucenie co drugiego piksela. Jeśli zastosuje się takie podejście bezpośrednio do matrycy Bayera (jak zaprezentowano na rysunku 4a i b), to utracona zostanie informacja o jednej ze składowych barwnych. W związku z tym, piksele należy odrzucać w inny sposób (rys. 4c i 4d).



Rys. 4. Redukcja pikseli na siatce Bayera: a) i b) warianty niepoprawne; c) i d) warianty poprawne

W zrealizowanym systemie zdecydowano się na rozwiązanie przedstawione na rysunku 4c. W ten sposób zredukowana została ilość danych, przy jednoczesnym zachowaniu informacji o kolorze. Skutkiem ubocznym redukcji rozdzielczości jest nieznaczna deformacja linii i brzegów.

3.3. Konwersja RGB na CIE Lab (moduł *rgb2lab*)

W przypadku realizacji kolorowego systemu wizyjnego istotny jest wybór przestrzeni barw. W pracy [4] Autorzy przebadali szereg przestrzeni barw i uzyskane wyniki wskazują, że do segmentacji z jednoczesną redukcją cieni, która jest jednym z celów omawianego systemu, najlepiej nadaje się przestrzeń barw CIE Lab i CIE Luv. Dlatego też zdecydowano się w niniejszej pracy wykorzystać przestrzeń CIE Lab.

W systemie CIE Lab trzy komponenty, mówiące o natężeniu poszczególnych barw RGB, zostały zastąpione parametrami L, a, b (L – luminancja, a, b – składowe chrominancji). Konwersja pomiędzy przestrzenią RGB, a CIE Lab jest dwuetapowa [8, 9]. W pierwszym kroku obraz z RGB konwertowany jest do przestrzeni CIE XYZ zgodnie z formułą:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0{,}412453 & 0{,}357580 & 0{,}180423 \\ 0{,}212671 & 0{,}715160 & 0{,}072169 \\ 0{,}019334 & 0{,}119193 & 0{,}950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

Z przestrzeni CIE XYZ do przestrzeni kolorów CIE Lab prowadzi następujące przekształcenie:

$$\begin{aligned} L &= 116 \cdot f(Y/Y_n) - 16 \\ a &= 500 \cdot [f(X/X_n) - f(Y/Y_n)] \\ b &= 200 \cdot [f(Y/Y_n) - f(Z/Z_n)] \end{aligned} \tag{2}$$

gdzie: $X_n = 0{,}950456$, $Y_n = 1$, $Z_n = 1{,}088754$ to stałe odpowiadające za punkt bieli, a funkcja $f(t)$ dana jest równaniem:

$$f(t) = \begin{cases} t^{\frac{1}{3}} & \text{dla} \quad t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & w\,p.p. \end{cases} \tag{3}$$

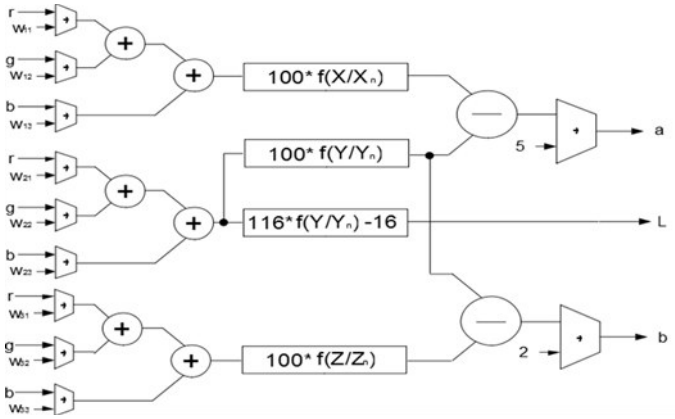
W celu zaimplementowania konwersji przestrzeni barw na platformie FPGA, wszystkie mnożenia zostały zrealizowane jako stałoprzecinkowe i wykonane z wykorzystaniem mnożarek sprzętowych DSP48, dostępnych w układzie Spartan 6. Ponieważ implementacja operacji pierwiastkowania (wzór (3)) w układzie FPGA wymagałaby wykorzystania dużej ilości zasobów sprzętowych, jak również wprowadziłaby znaczne opóźnienie w przetwarzaniu sygnału, dalsze operacje zostały stablicowane i umieszczone w zasobach blokowej pamięci wewnętrznej układu FPGA (BRAM). Ponieważ X_n, Y_n, Z_n są stałymi, utworzono 4 tablice, w których zapisano na stałe następujące parametry:

$$\begin{aligned} xlut(t) &= 100 \cdot (t/X_n) \\ ylut(t) &= 100 \cdot (t/Y_n) \\ zlut(t) &= 100 \cdot (t/Z_n) \\ llut(t) &= 116 \cdot (t/Y_n) \end{aligned} \tag{4}$$

W ten sposób problem przekształcono do postaci:

$$\begin{aligned} L(X,Y,Z) &= llut(Y) \\ a(X,Y,Z) &= 5 \cdot (xlut(X) - ylut(Y)) \\ b(X,Y,Z) &= 2 \cdot (ylut(Y) - zlut(Z)) \end{aligned} \tag{5}$$

Schemat moduł konwersji RGB na CIE Lab został przedstawiony na rysunku 5. Moduł został napisany w języku Verilog. Poprawność działania została potwierdzona symulacyjnie; uzyskane wyniki były zgodne z modelem stworzonym w programie Matlab. Maksymalna częstotliwość pracy raportowana przez narzędzie do syntezy wynosi 252 MHz. Zużycie zasobów FPGA przedstawiono w tabeli 1.



Rys. 5. Schemat modułu konwersji RGB na CIE Lab

Tabela 1
Zużycie zasobów układu FPGA dla modułu *rgb2lab*

Zasób	Użyte	Dostępne	Procent
FF	94	54576	1%
LUT6	154	27288	1%
SLICE	57	6822	1%
DSP48	9	58	15%
BRAM	10	116	9%

4. Generacja tła (moduł *bg*)

Generacja tła jest najczęściej wykorzystywaną metodą wykrywania ruchu przy założeniu, że obraz rejestrowany jest przez statyczną kamerę. Stanowi ona podstawę wielu zaawansowanych systemów monitoringu wizyjnego. Ogólna koncepcja polega na wykrywaniu ruchu na zasadzie odejmowania aktualnej ramki obrazu (z kamery) od tła referencyjnego, które jest rezultatem działania algorytmu generacji tła. W ciągu ponad 20 lat badań w tej dziedzinie powstało wiele algorytmów i ich modyfikacji. Bardzo dobry i wyczerpujący przegląd metod zaprezentowany został w pracy [6].

Podczas analizy algorytmów generacji tła, z uwagi na możliwość ich implementacji w zasobach układu FPGA, istotny jest podział na metody nierekurencyjne i rekurencyjne. Metody nierekurencyjne, do których można zaliczyć średnią z N ostatnich ramek, medianę z N ostatnich ramek, metodę W4, charakteryzują się wysoką adaptacyjnością i nie zależą od historii sprzed N ramek. Podstawową wadą jest jednak duża złożoność pamięciowa (przykładowo bufor $N = 30$ ramek o rozdzielczości 640×480 w kolorze RGB (24 bit) to ok. 26 MB).

W technikach rekurencyjnych model tła aktualizowany jest tylko na podstawie bieżącej ramki. Podstawową zaletą takiego podejścia jest niewielka złożoność pamięciowa, a wadą – mała odporność na zakłócenia powstałe w tle (są one długo eliminowane). Do metod rekurencyjnych zaliczyć można metodę sigma-delta, metodę pojedynczego rozkładu Gaussa [19], filtr Kalmana, metodę wielokrotnego rozkładu Gaussa (MOG) [18] oraz klasteryzację [5]. Warto zwrócić uwagę na podział metod na przechowujące reprezentację jednego wariantu tła (sigma-detla, pojedynczego rozkładu Gaussa) i przechowujące wiele wariantów tła (MOG, klasteryzacja). Metody wielowariantowe lepiej sprawdzają się w warunkach dynamicznie zmieniającego się oświetlenia (np. cienie rzucane przez chmury) oraz pozwalają rozwiązać problem inicjalizacji tła w przypadku obecności obiektów ruchomych na scenie.

Implementacja modułu generacji tła w zasobach układu FPGA pozwala: odciążyć procesor (CPU) komputera, którego moc obliczeniowa może zostać wykorzystana do realizacji dalszych etapów zaawansowanego systemu nadzoru wizyjnego (śledzenia obiektów, klasyfikacji obiektów, oceny zachowania itp.), lub stworzyć system typu inteligentna kamera (smart-camera), w którym detekcja obiektów ruchomych będzie przebiegała w kamerze, a wyniki wykorzystane zostaną np. do inteligentnej kompresji rejestrowanego obrazu. Dodatkowo, w przypadku wielowariantowych metod generacji tła architektura układu FPGA pozwala na zrównoleglenie operacji związanych z uaktualnianiem modelu.

W literaturze można znaleźć kilka prac związanych z implementacją generacji tła w FPGA. W artykule [2] autorzy zaprezentowali implementację metody bazującej na algorytmie MOG, z uwzględnieniem specyfiki obliczeń w układzie FPGA. W pracy [12] autorzy zaprezentowali implementację metody MOG w zasobach układu FPGA (Virtex 2 1000). Podstawowym problemem przy sprzętowej realizacji tego algorytmu jest dostęp do zewnętrznej pamięci RAM – przy założonej przez cytowanych autorów precyzji obliczeń, pojedynczy rozkład Gaussa w przestrzeni RGB zajmuje 124 bity. Dlatego też, schemat kompresji zaproponowany w cytowanej pracy opierał się na założeniu, że sąsiednie piksele mają podobny rozkład. Wyniki symulacyjne wskazują, że zaproponowana metoda pozwoliła o ok. 60% zredukować zapotrzebowanie na przepustowość do pamięci RAM. Implementacje generacji tła w układach FPGA, oparte na innych algorytmach, opisano również w pracach [1, 7, 13, 16].

4.1. Wykorzystany algorytm

Zarówno analiza literatury, jak i prowadzone badania wstępne pokazały, że podstawowym zagadnieniem do rozwiązania, w trakcie implementacji algorytmu generacji tła w zasobach rekonfigurowalnych, jest problem wydajnego dostępu do zewnętrznej pamięci RAM. Stąd, punktem wyjścia do wyboru algorytmu była analiza jego wymagań pod kątem wymaganej przepustowości do pamięci RAM. Na wstępie przyjęto następujące założenia odnośnie do wykorzystywanego algorytmu: ma działać na obrazach kolorowych oraz być wielowariantowy. Wykorzystanie koloru ma poprawić jakość analizy obrazu. Wielowariantowość pozwoli na adaptacyjność modelu tła zarówno w warunkach szybkozmiennej, jak i wolnozmiennej zmiany natężenia oświetlenia. Założenia te wykluczyły z dalszej analizy następujące

algorytmy: pojedynczy rozkład Gaussa, sigma-delta i filtr Kalmana. Spośród dwóch powszechnie opisywanych w literaturze metod wielowariantowych: MOG i klasteryzacji zdecydowano się wybrać klasteryzację, głównie z uwagi na prostsze niż w MOG obliczenia i łatwiejszą inicjalizację (nie jest potrzebne wyznaczenie początkowego rozkładu np. algorytmem EM (*Expectation Maximization*).

Dla potrzeb omawianej implementacji dokonano kilku modyfikacji algorytmu opisanego w pracy [5], które zaznaczono w opisie metody. Pierwsza z nich, to wybór przestrzeni barw. Zdecydowano się wykorzystać przestrzeń CIE Lab, jako lepiej nadającą się do detekcji i usuwania cieni (na podstawie wyników pracy [4]). W przestrzeni CIE Lab składowa luminacji L jest liczbą z zakresu 0–100 (7 bitów), a składowe chrominancji (*ab*) zmieniają się od –127 do 127 (8 bitów). Wychodząc od wartości 128-bitów (dostępna szerokość magistrali do pamięci RAM na karcie SP605), założono liczbę wariantów (klastrów) tła $K = 4$, a reprezentację pojedynczego klastra jako: składowa L: 9 bitów (7 część całkowita + 2 ułamkowa), składowe a i b: 8 bitów, waga 6 bitów, zatem jeden klaster wymaga 31 bitów, a cztery 124 bity.

Dla pierwszej wczytanej ramki następuje inicjalizacja tła – wartość piksela staje się pierwszym klastrem. Dla kolejnych ramek przeprowadzane są następujące operacje (dla każdego piksela niezależnie):

A) Wyznaczenie odległości pomiędzy nowym pikselem a każdym z klastrów. Odległości wyznaczane są osobno dla składowej luminancji, a osobno dla chrominancji na podstawie wzorów:

$$dL = \left|L_F - L_{Mi}\right| \tag{6}$$

$$dC = \left|Ca_F - Ca_{Mi}\right| + \left|Cb_F - Cb_{Mi}\right| \tag{7}$$

gdzie:

L_F, Ca_F, Cb_F – wartość piksela z aktualnej ramki,
L_{Mi}, Ca_{Mi}, Cb_{Mi} – wartość piksela z i-tego klastra.

B) Wskazanie klastra, który jest najbliższy aktualnemu pikselowi oraz sprawdzenie, czy dla tego piksela wartości dL i dC są mniejsze od zdefiniowanych progów (*luminanceTh* i *colourTh*).

C) W przypadku gdy klaster spełnia założenia z punktu B), następuje aktualizacja zgodnie ze wzorem:

$$M_{akt} = \alpha_1 F + (1 - \alpha_1) \cdot M \tag{8}$$

gdzie:

M – klaster,
M_{akt} – zaktualizowany klaster,
F – ramka,
α_1 – parametr określający szybkość uaktualniania tła.

Ponadto następuje inkrementacja wagi klastra. Z uwagi na przyjętą reprezentację dla wagi (6 bitów), może ona maksymalnie osiągnąć wartość 63. Następnie wykonuje się sortowanie klastrów (według malejącej wagi). Warto zauważyć, że dopuszczalne jest uproszczenie, w postaci założenia, że klaster, którego waga została inkrementowana, może zamienić się miejscem tylko z klastrem będącym bezpośrednio przed nim. W znakomitej większości przypadków założenie jest prawdziwe i pozwala znacznie uprościć sortowanie (zarówno programowo, jak i sprzętowo).

D) W przypadku gdy nie znaleziono dopasowanego klastra w oryginalnej propozycji algorytmu, klaster o najmniejszej wadze zastępowany był aktualnym pikselem, a jego waga zerowana. Podczas testów algorytmu okazało się, że takie podejście powoduje zbyt szybkie przenikanie obiektów do modelu tła (np. osób, które się na chwilę zatrzymały). Zdecydowano się zatem, wprowadzić modyfikację. Polega ona na wykorzystaniu schematu aktualizacji (równanie (8)) z parametrem α_2 (zerowanie wagi zachowano) zamiast bezpośredniego przypisania. W ten sposób do modelu tła mogą przeniknąć tylko obiekty, które w danej lokalizacji pozostaną wystarczająco długo (w zależności od dobranej wartości α_2), aby się „wtopić" w ostatni klaster.

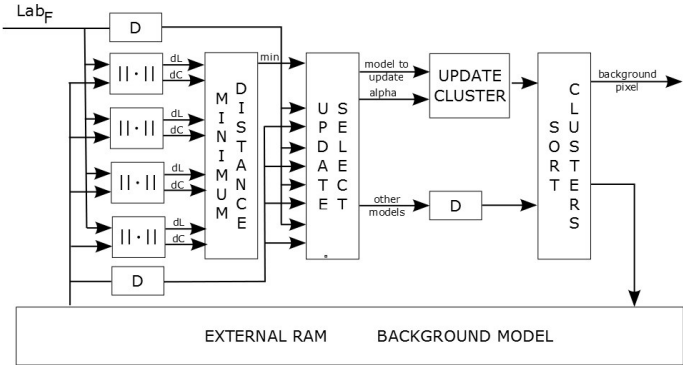
Oryginalny algorytm zakładał działanie w dwóch fazach – inicjalizacji (uczenia) tła i działania właściwego. W fazie inicjalizacji algorytm działa w sposób opisany powyżej. Następnie, jeżeli wagi (ew. waga) pierwszych *b* klastrów jest większa od progu, dla wszystkich pikseli tworzony jest model tła (składający się z *b* klastrów) i na jego podstawie prowadzona jest dalsza detekcja obiektów. W tej fazie nie jest możliwe dodawanie nowych elementów do tła, a jedynie jego aktualizacja na podstawie równania (8), celem kompensacji niewielkich zmian oświetlenia. W opisywanej implementacji zdecydowano się na działanie cały czas w fazie uczenia, tak aby model tła mógł dopasowywać się do zmian na scenie. Decyzja o wyborze trybu działania algorytmu może zależeć od charakteru rozpoznawanej sceny. W przyszłości planuje się dodanie do modułu sprzężenia zwrotnego od dalszych etapów przetwarzania i rozumienia obrazu, które pozwoli na lepsze zarządzanie modelem tła.

Kolejną modyfikacją jest pominięcie mechanizmu wyznaczania obiektów ruchomych, zaproponowanego w oryginalnej implementacji. Maska obiektów ruchomych powstaje w innym module (opisanym w rozdziale 5), a moduł generacji tła dostarcza jedynie opisu tła dla każdej lokalizacji. Przy czym, za prawidłowy klaster uznaje się tylko taki, którego waga przekracza pewien ustalony próg (*weightTh*). Ostatni klaster (o najmniejszej wadze), z uwagi na to, że stanowi bufor pomiędzy aktualną ramką a modelem tła, nie jest rozpatrywany jako kandydat na tło.

4.2. Sprzętowa implementacja algorytmu

Schemat zaproponowanego modułu generacji tła zaprezentowano na rysunku 6. Moduł został opisany w języku VHDL z wykorzystaniem sprzętowych modułów IP Core Xilinx (mnożenie, linie opóźniające). Syntezę oraz implementację wykonano przy użyciu narzędzia ISE 13.1 firmy Xilinx dla układu Spartan 6 LX 45T. Przeprowadzone w programie ModelSim 6.5c symulacje (behawioralne i po fazie *place & route*) wykazały pełną zgodność modułu sprzętowego ze stworzonym w pakiecie Matlab 2009b, modelem programo-

wym algorytmu. Statyczna analiza czasowa wykazała, że w obecnej formie moduł może działać z częstotliwością 148 MHz. Jest to wartość w zupełności wystarczająca do przetwarzania obrazu o rozdzielczości 640×480/60 klatek na sekundę (dla takich parametrów zegar piksela wynosi 25 MHz). Zużycie zasobów FPGA przedstawione zostało w tabeli 2.



Rys. 6. Schemat modułu generacji tła

Opis użytych modułów:

- || || – wyznaczanie odległości pomiędzy aktualnym pikselem a klastrem (punkt A);
- D – opóźnienie (w celu zachowania pełnej potokowości implementacji);
- MINIMUM DISTANCE – wybranie klastra o najmniejszej odległości od aktualnego piksela (punkt B);
- UPDATE SELECT – wybór klastra do uaktualnienia;
- UPDATE CLUSTER – implementacja równania (3) oraz uaktualnienie wagi; dla parametru α, który jest z zakresu [0;1), przyjęto stałoprzecinkową reprezentację 10-bitową; mnożenie zrealizowano z wykorzystaniem dostępnych w układzie FPGA Spartan 6, sprzętowych mnożarek DSP48;
- SORT CLUSTERS – sortowanie klastrów oraz wybór aktualnej reprezentacji tła.

Warto podkreślić, że moduł zużywa tylko niewielką część zasobów dostępnych w układzie Spartan 6 LX 45T, co pozwala na implementację w tym samym układzie również dalszych etapów przetwarzania danych, w ramach wizyjnego systemu nadzoru.

Tabela 2
Zużycie zasobów FPGA przez moduł generacji tła

Zasób	Użyte	Dostępne	Procent
FF	905	54576	1%
LUT 6	1244	27288	4%
SLICE	460	6822	7%
DSP48	6	58	10%

5. Segmentacja obiektów ruchomych

Detekcja i śledzenie obiektów jest podstawą wielu aplikacji, w szczególności jest jednym z najważniejszych elementów inteligentnych systemów nadzoru wizyjnego. Podstawową metodą segmentacji jest progowanie różnicy pomiędzy aktualną ramką a modelem tła. Podejście to zastosowano również w niniejszej pracy, jednakże zdecydowano się wykorzystać oprócz informacji o jasności i kolorze informację o teksturze oraz tak skonstruować algorytm, aby zminimalizować wpływ cieni na rezultaty segmentacji.

Wykrywanie cieni opiera się zasadniczo na dwóch założeniach: cień nie zmienia koloru elementów, na które pada, a jedynie jasność (testy w warunkach oświetlenia słonecznego pokazały, że założenie to nie zawsze jest prawdziwe), oraz cień nie zmienia tekstury elementów, na które pada. Redukcję wpływu cieni zrealizowano na podstawie wymienionych założeń, wykorzystując rezultaty prac: [4] – wybór przestrzeni barw jako CIE Lab, SILTP [17] – wybór deskryptora teksturowego.

Przeprowadzone badania wstępne pokazały, że przydatność informacji teksturowej przy detekcji cieni bywa w wielu wypadkach ograniczona, ponieważ bardzo często elementy sceny nie mają wyraźniej tekstury (albo nie jest ona wykrywana przez deskryptor) lub głębokie cienie zasłaniają teksturę (np. słoneczny dzień na zewnątrz). Jednakże dodanie informacji o teksturze poprawia wyniki segmentacji obiektów ruchomych i dlatego zdecydowano się na implementację metody SILTP, której realizacja sprzętowa jest dość prosta przy wynikach lepszych niż np. przy użyciu gradientu Sobela. Ponieważ implementacja w systemach nadzoru powinna działać w czasie rzeczywistym, zdecydowano się dążyć do potokowej implementacji wszystkich algorytmów w układzie FPGA. Warto zwrócić uwagę, że z powodu przyjęcia tego założenia, dobierając algorytmy skoncentrowano się na lokalnych cechach obrazu (piksel, kontekst o rozmiarze 3×3, 5×5). Niestety takie podejście ogranicza w znaczący sposób skuteczność algorytmu usuwania cieni. Większość opisanych w literaturze metod, które dobrze wykrywają cienie, operuje na poziomie obiektów, tj. po etapie etykietowania analizowane są cechy całych obszarów. W obecnej implementacji jakość wykrywania cienia nie jest priorytetem, niemniej zastosowana metoda umożliwia usuwanie cienia w wielu przypadkach, przez co poprawia jakość analizowanej sekwencji wideo.

5.1. Sprzętowa implementacja segmentacji

Jedyna znana autorom praca (po dokonaniu analizy baz INSPEC), dotycząca implementacji detekcji cieni w układach FPGA, to artykuł [15]. Opisano w nim implementację metody detekcji cieni w przestrzeni barw YCbCr (osobne progowanie każdej składowej) wraz z dodatkowym wykorzystaniem informacji o krawędziach.

W zaproponowanej w niniejszym artykule metodzie zdecydowano się wykorzystać trzy informacje: jasność (składowa L z modelu CIE Lab), kolor (składowe ab z modelu Lab) oraz deskryptor teksturowy SILTP. Odległość aktualnej ramki od tła (jasność i kolor) jest liczona zgodnie z wzorami (6) i (7). Definicję deskryptora teksturowego SILTP można znaleźć w pracy [17].

Wartości trzech wykorzystywanych miar (jasność, kolor, tekstura) zostały znormalizowane z wykorzystaniem metody podobnej do zaprezentowanej w artykule [14]:

$$dN = \begin{cases} 1, if\ d > \max(d) \cdot \beta \\ d / \max(d), w\ p.p. \end{cases} \tag{9}$$

gdzie: β to parametr z zakresu (0;1] (w eksperymentach 0,75), a max(d) oznacza maksymalną wartość miary dla danego obrazu (w implementacji sprzętowej wykorzystano maksymalną wartość z poprzedniej ramki). Dodatkowo dla chrominancji zaimplementowano analogiczny mechanizm eliminacji niewielkich wartości (źródło szumu), a dla SILTP ustalono współczynnik $\beta = 1$.

Na podstawie znormalizowanych wartości zaproponowano i przebadano kombinację trzech miar:

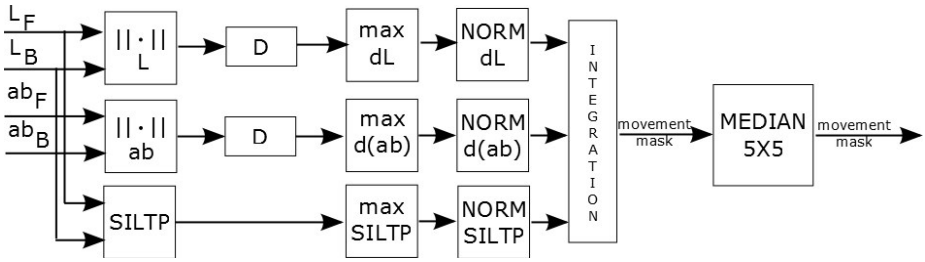
$$LCT = w_L dNL + w_C dN(ab) + w_T SILTP_N \tag{10}$$

gdzie:

w_L, w_C, w_T – wagi (eksperymentalnie ustalone na odpowiednio 1, 3 i 2),
dNL – znormalizowana różnica w jasności,
$dN(ab)$ – znormalizowana różnica w kolorze,
$SILTP_N$ – znormalizowany deskryptor SILTP.

W ostatnim kroku algorytmu dokonano binaryzacji współczynników LCT z wybranym progiem (0,9375 w zrealizowanych eksperymentach). Do przetwarzania końcowego zdecydowano się użyć binarnej filtracji medianowej z maską o rozmiarze 5×5.

Schemat kompletnego modułu segmentacji zaprezentowano na rysunku 7. Moduł, podobnie jak opisana wcześniej generacja tła, został opisany w języku VHDL z wykorzystaniem sprzętowych modułów IP Core (mnożenie, linie opóźniające). Do syntezy i implementacji użyto narzędzi ISE 13.1 firmy Xilinx, a do symulacji działania modułu programu ModelSim 6.5c. Przeprowadzone testy wykazały zgodność modułu sprzętowego z modelem programowym (opisanym w pakiecie Matlab 2009b). Do implementacji założono rozmiar obrazka jako 640×480. Ma on wpływ na długość linii opóźniających użytych w modułach SILTP oraz MEDIAN 5×5 i ostateczne zużycie zasobów, jak i opóźnienie wprowadzane przez moduł. Statyczna analiza czasowa wykazała, że w obecnej formie moduł może działać z częstotliwością 390 MHz. Zużycie zasobów FPGA przedstawione zostało w tabeli 3.



Rys. 7. Schemat modułu segmentacji

Opis modułów:

- || || L i || || ab – obliczanie odległości między aktualną ramka a tłem, zgodnie ze wzorami (6) i (7),
- SITLP – moduł obliczający deskryptor SILTP,
- D – opóźnienie (umożliwienie działania całego systemu w sposób potokowy),
- max dL, d(ab) SILTP – moduły wyznaczające maksymalną wartość w poprzedniej ramce,
- NORM – moduły odpowiadające za normalizację wartości do zakresu [0:1]
- INTEGRATION – moduł odpowiadający za integrację informacji o jasności, kolorze i teksturze oraz końcowe progowanie (decyzja obiekt/tło),
- MEDIAN 5×5 – binarna mediana z oknem o rozmiarze 5×5.

Tabela 3
Zużycie zasobów FPGA przez moduł segmentacji

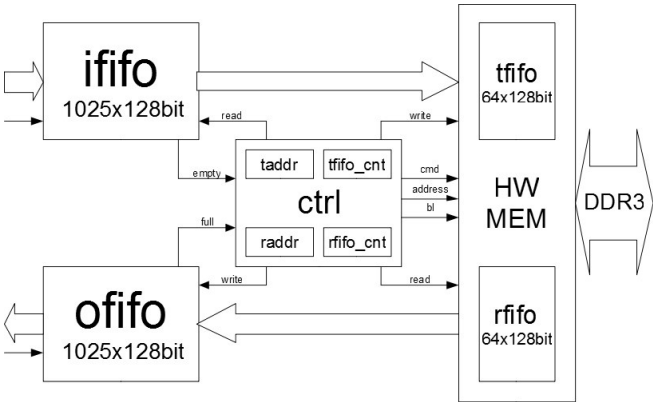
Zasób	Użyte	Dostępne	Procent
FF	1059	54576	1%
LUT 6	1405	27288	5%
SLICE	401	6822	3%
DSP48	3	58	5%

6. Obsługa zewnętrznej pamięci (*memctrl*)

W układach FPGA Spartan 6 firmy Xilinx dostępny jest zintegrowany sprzętowy kontroler obsługi zewnętrznej pamięci RAM DDR. Zapewnia on wygenerowanie odpowiednich sygnałów dostępu do pamięci, jej odświeżanie oraz kalibrację portów wejścia/wyjścia. Od strony użytkownika dostarcza on konfigurowalną liczbę portów o konfigurowalnej szerokości słowa. Każdy port ma własne bufory dla danych i komend, a jego obsługa sprowadza się do ustawienia adresu, wybrania komendy oraz podania liczby bajtów, które mają zostać odczytane lub zapisane. Użytkownik musi jednak zapewnić, że jego komenda

jest możliwa do wykonania (wystarczająca ilość danych w buforze w przypadku zapisu i miejsce na dane przy odczycie). Ponieważ moduł generacji tła wykorzystuje słowo o szerokości 128 bitów, wybrana została konfiguracja z jednym dwukierunkowym portem dostępu do kontrolera pamięci o długości 128 bitów.

Generacja tła wymaga zapisania i odczytania 128 bitów informacji (2×16 bajtów = 32B) w każdym takcie zegara, w którym na magistrali danych obrazu znajdują się poprawne piksele. Zatem przy rozdzielczości 640×480 i przy 60 klatkach na sekundę jest to 18,5 Mpróbek/s * 32 B, co daje ok. 590 MB/s. Pamięć dostępna na płycie SP605 to pamięć DDR3 o 16-bitowym słowie, taktowana zegarem 400 MHz. Teoretyczna przepustowość tej pamięci to 400MHz * 2(DDR) * 2 bajty, zatem prawie 1600 MB/s. Należy jednak pamiętać, że pamięć DDR jest pamięcią dynamiczną, korzystając z niej, nie można założyć stałej długości czasu potrzebnego do odczytania/zapisania danych. Wiąże się to między innymi z dostępem do pamięci w trybie *burst*, koniecznością otwierania rzędów, kolumn lub banków pamięci, jak również koniecznością odświeżania pamięci dynamicznej (raz na około 64 ms dostęp do banku zostaje wstrzymany i następuje jego odświeżenie). Z tego powodu, bufor w sprzętowym kontrolerze o pojemności 64 słowa 128-bitowe nie jest w stanie zapewnić wystarczającego zapasu danych do utrzymania stałej przepustowości na poziomie 590MB/s dla modułu generującego tło. W tym celu dodano dodatkowe bufory pośredniczące o pojemności 1025 słów 128-bitowych w celu zapewnienia nieprzerwanego strumienia danych. Dodatkowo bufory pozwalają na wykorzystanie pamięci w czasie, gdy w sygnale VGA nie ma właściwych danych wizyjnych, a tylko odbywa się synchronizacja.



Rys. 8. Schemat kontrolera pamięci RAM

Zaimplementowany kontroler dostępu do pamięci ustala adresy do odczytu i zapisu oraz oblicza informacje o stanie napełnienia buforów w bloku sprzętowym i na tej podstawie generuje odpowiednie komendy. Steruje również przepływem pomiędzy tymi buforami, a dużymi kolejkami FIFO, które są bezpośrednio połączone z modułem generacji tła. Schemat kontrolera pamięci przedstawiono na rysunku 8.

7. Pozostałe moduły

7.1. Ustawianie parametrów (*regs*, *uart*)

Aby umożliwić zmianę parametrów systemu w czasie rzeczywistym, wykorzystano znajdujący się na płycie SP605 układ PL2031 będący konwerterem magistrali USB na RS232. Po stronie FPGA zaimplementowany został moduł UART do obsługi RS232, który umożliwia odczyt i zapis do 32 rejestrów 16-bitowych. Rejestry są podłączone do odpowiednich modułów, przez co możliwa jest zmiana parametrów systemu.

Dodatkowo zaimplementowano możliwość przekierowania sygnału RS232 tak, aby zamiast sterować zapisem do rejestrów karty, był on przekazywany poprzez kartę FMC do kamery, co umożliwia zmianę parametrów kamery podczas działania systemu (np. zmiana kontrastu, parametrów korekcji gamma itd.).

7.2. Wizualizacja (*dvi*)

Ostatni moduł ma za zadanie sterowanie układem ch7301c (enkoder DVI znajdujący się na płycie SP605) oraz przesyłanie do niego danych wizyjnych (na dwóch zboczach zegara, przy wykorzystaniu buforów DDR). W tym celu wykorzystywany jest procesor Picoblaze, dostarczony w formie IP Core przez firmę Xilinx, którego zadaniem jest konfiguracja układu 7301c za pomocą magistrali I^2C.

8. Integracja systemu

Wszystkie opisane w rozdziałach 3–7 moduły zostały zintegrowane zgodnie ze schematem załączonym na rysunku 2. Projekt został zsyntezowany dla układu Xilinx Spratan 6 oznaczonego symbolem XC6slx45t-3fgg484. Wykorzystano narzędzie Xilinx ISE 13.1. Raportowana maksymalna częstotliwość zegara wyniosła (po fazie *place&route*) 119 MHz i znacznie przewyższała wymagane 25 MHz. Analiza przeprowadzona za pomocą narzędzia *Xilinx XPower Analyzer* wskazuje, że moc pobierana przez układ (*On-Chip*) wynosi ok. 0,9 W, a moc zasilania układu ok. 1,2 W. Zużycie zasobów FPGA zostało zaprezentowane w tabeli 4.

Analiza danych zaprezentowanych w tabeli 4 pokazuje, że nawet w relatywnie nieduży układzie FPGA serii Spartan 6 można zrealizować dość złożony algorytm wizyjny, wykorzystując niespełna 30% dostępnych zasobów sprzętowych. Pozostałą logikę można wykorzystać do implementacji wstępnej filtracji obrazu (eliminacja szumów z kamery), implementacji filtracji medianowej pomiędzy modułami generacji tła i segmentacji lub innych operacji przetwarzania obrazu, z wyłączeniem tych, które wymagają wykorzystania zewnętrznej pamięci RAM.

Tabela 4
Zużycie zasobów FPGA przez cały projekt

Zasób	Użyte	Dostępne	Procent
FF	4737	54576	8%
LUT 4	5188	27288	19%
SLICE	1938	6822	28%
DSP48	27	58	58%
BRAM	24	116	20%

9. Rezultaty i wnioski

9.1. Generacja tła

Zaimplementowany algorytm okazał się generować wyniki lepsze od metod opartych na jednym wariancie tła (np. pojedynczy rozkład Gaussa), ale rezultatami ustępuje metodzie MOG. Algorytm przetestowano symulacyjnie na szeregu sekwencji, które należy uznać za trudne. Rejestracja odbywała się w wietrzny dzień z szybkimi zmianami oświetlenia. Doświadczenia z algorytmem wykazują, że wielowariantowość pozwala na redukcję wpływu ruchu drobnych obiektów (np. krzewów na wietrze), z zastrzeżeniem jednak, że ruchy te nie mogą być za duże, a tło, na jakim znajduje się obiekt, nie może być skomplikowane. W innym przypadku, metoda wykorzystująca 4 warianty tła, nie jest w stanie zamodelować każdej możliwej wartości piksela i ruch obiektu, de facto należącego do tła, jest wykrywany.

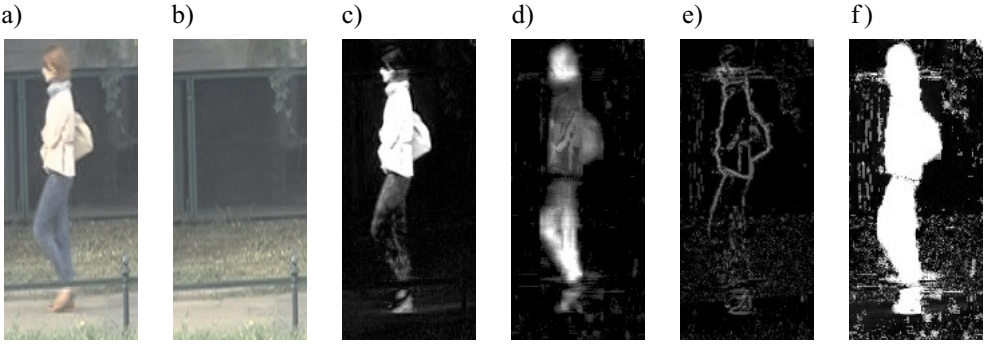
Podczas testów stwierdzono, że bardzo istotny dla działania algorytmu generacji tła, opartego na aktualizacji zgodnie z równaniem (8), jest dobór precyzji obliczeń i reprezentacji modelu tła. Przyjęta reprezentacja (9 bitów składowa L, i 8 bitów składowa a i b) okazała się być wystarczająca jedynie dla dużych wartości współczynnika α. (0,75, 0,125). Przy mniejszych wartościach, tło nie było poprawnie uaktualniane – zaokrąglanie powodowało, że udział aktualnej ramki (αF) był zbyt mały, aby wpłynąć na nową wartość M, która była dostosowywana do wybranej reprezentacji. W przypadku implementacji programowych, eliminacja problemu jest dość prosta i wymaga przechowywania modeli tła w postaci liczby rzeczywistej (najczęściej zmiennoprzecinkowej typu *float* lub *double*).

W przypadku opisywanej implementacji sprzętowej FPGA, w której głównym ograniczeniem jest szerokość dostępu do zewnętrznej pamięci RAM (omówienie w rozdz. 6), eliminacja problemu wiąże się z wyborem: duża liczba wariantów tła lub duża precyzja reprezentacji wariantu. W obecnym etapie prac zdecydowano się wykorzystać 4 warianty tła z dość małą precyzją. Na platformie SP605 szerokość dostępu do pamięci RAM wynosząca 128 bitów jest wartością zbliżoną do maksymalnej, zatem zwiększenie precyzji obliczeń wiązałoby się ze zmniejszeniem liczby wariantów tła. Dalsze badania tego zagadnienia będą

koncentrować się na określeniu zależności pomiędzy wartością α, a wymaganą szerokością reprezentacji bitowej. Ponadto podjęte zostaną próby przeniesienia algorytmu na inną platformę sprzętową, na której dostępna jest szersza magistrala do pamięci RAM.

9.2. Segmentacja obiektów ruchomych

Zaproponowana w pracy metoda segmentacji obiektów ruchomych zakłada integrację trzech informacji: jasności, koloru i tekstury w celu polepszenia wyników oraz dodatkowo eliminacji cieni. Podczas prowadzonych badań udało się wskazać sytuację, w których tego rodzaju podejście daje lepsze rezultaty niż wykorzystanie jedynie jasności. Wynik stanowi również potwierdzenie sensowności wykorzystania modelu tła w kolorze, mimo iż wymaga on trzykrotnie więcej zasobów pamięciowych i obliczeń niż model w skali szarości.



Rys. 9. Przykład segmentacji: a) aktualna ramka; b) tło; c) różnica w jasności; d) różnica w kolorze; e) tekstura SILTP; f) integracja informacji

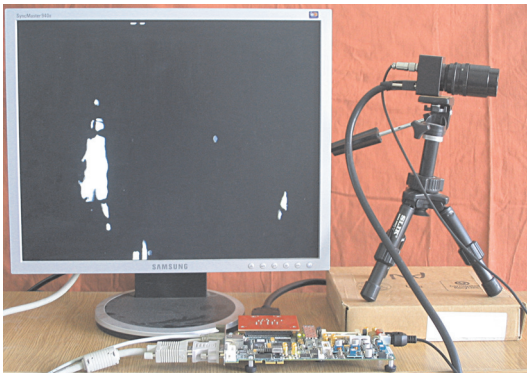
Na rysunku 9 zaprezentowano przykład takiej sytuacji. Na obrazie 9c można zaobserwować, że jasność włosów i spodni osoby i jasność tła są bardzo zbliżone i praktycznie niemożliwe jest zaproponowanie progu binaryzacji, który pozwoli na poprawną segmentację całej sylwetki. Informacja o kolorze (rys. 9d) pozwala na poprawną segmentację obszaru głowy (włosów) i spodni. Tekstura (rys. 9e) w omawianym przypadku ma znaczenie pomocnicze. Integracja cech (rys. 9f) zgodnie z równaniem (10) pozwala na poprawną segmentację sylwetki.

Eliminacja cieni, przy założeniu, że wykorzystywana jest jedynie informacja lokalna (piksel, mały kontekst), w wielu przypadkach okazuje się zawodna. Przeprowadzone badania potwierdzają tę tezę. Jednakże da się wskazać sytuację (rys. 10a i b), kiedy z wykorzystaniem zaproponowanej metody udaje się uzyskać redukcję wpływu cieni. W sytuacji przedstawionej na rysunku 10c i d, przy mocniejszym oświetleniu, cienie stają się głębsze i zaproponowany algorytm nie jest w stanie dokonać poprawnej segmentacji sylwetek. Warto jeszcze zauważyć, że zastosowana metodologia jest mniej wrażliwa na dobór ostatecznego progu binaryzacji niż zastosowanie tylko jednej informacji np. jasności.

Rys. 10. Przykłady redukcji wpływu cieni. Redukcja poprawna (słabsze oświetlenie): a) scena; b) maska obiektów ruchomych. Redukcja niepoprawna (silne oświetlenie – głębokie cienie); c) scena; d) maska obiektów

9.3. Uruchomienie systemu

Opisany system do detekcji obiektów ruchomych został pomyślnie uruchomiony i poddany testom. Uzyskano zakładaną rozdzielczość 640×480 przy 60 klatkach na sekundę oraz pracę w kolorze w przestrzeni CIE Lab. System działa poprawnie i zgodnie z przewidywaniami. W przyszłości planowane jest dalsze ulepszenie rozwiązania i poszerzanie jego funkcjonalności. Zdjęcie wykonanego stanowiska badawczego wraz z przykładowym rezultatem działania segmentacji przedstawiono na rysunku 11. Na skonstruowanym stanowisku przetwarzanie obrazu następuje w czasie rzeczywistym: 60 klatek na sekundę.



Rys. 11. Wykonane stanowisko (przetwarzanie 60 klatek/s, rozdzielczość 640×480 pikseli)

10. Podsumowanie

W artykule przedstawiono opis implementacji systemu segmentacji obiektów ruchomych z redukcją cieni w układzie reprogramowalnym FPGA. Zadanie wymagało zaprojektowania szeregu modułów sprzętowych (opisanych w językach Verilog i VHDL):

komunikacji z kamerą, transformacji Bayera, konwersji przestrzeni barw RGB -> CIE Lab, generacji tła, segmentacji, komunikacji z zewnętrzną pamięcią RAM, komunikacji szeregowej z komputerem oraz wyświetlania obrazu na monitorze. W pracy zaimplementowano wielowariantowy algorytm generacji tła, będący sprzętową realizacją, zmodyfikowanej przez autorów, opisanej w literaturze metody klasteryzacji oraz segmentacji z wykorzystaniem informacji o jasności, kolorze i teksturze. Zasadniczym osiągnięciem opisywanych prac jest stworzenie działającego w czasie rzeczywistym (przetwarzanie 60 klatek na sekundę, rozdzielczość 640×480 pikseli), systemu do segmentacji obiektów ruchomych. Wyniki świadczą o wysokiej przydatności układów FPGA do implementacji złożonych algorytmów analizy obrazu w systemach wizyjnych. Zaprojektowane moduły stanowią bazę, która w przyszłości umożliwi dalszą rozbudowę funkcjonalności systemu.

Literatura

[1] Abutaleb M.M., Hamdy A., Abuelwafa M.E., Saad E.M., *FPGA-based object-extraction based on multimodal Σ–Δ background estimation.* 2nd International Conference on Computer, Control and Communication, 2009. IC4 2009, 17–18 Feb. 2009, 1–7.

[2] Appiah K., Hunter A., *A single-chip FPGA implementation of real-time adaptive background model*. [w:] IEEE 2005 Conference on Field-Programmable Technology (FPT' 05), December 2005, 11–14.

[3] Bayer B.E., *Color imaging array*. US Patent No. 3971065.

[4] Benedek C., Sziranyi T., *Study on color space selection for detecting cast shadows in video surveillance*. Articles. Int. J. Imaging Syst. Technol., 17, 3, 2007, 190–201.

[5] Butler D., Sridharan S., Bove VMJr., *Real-time Adaptive Background Segmentation. Acoustics, Speech, and Signal Processing*. Proc. (ICASSP '03), 2003 IEEE Int. Conf. on April 2003, 349–52.

[6] Elhabian S.Y., El-Sayed K.M., Ahmed S.H., *Moving object detection in spatial domain using background removal techniques, state-of-art*. Recent Patents on Computer Science, 1, 2008, 32–54.

[7] Gorgoń M., Pawlik P., Jabłoński M., Przybyło J., *PixelStreams-based implementation of videodetector.* Preliminary proceedings of the 15th Annual Symposium on Field-Programmable Computing Machines – FCCM'07, 23–25 April 2007, Napa, USA 2007.

[8] ICC.1:2004-10 Specification (Profile version 4.2.0.0) Image technology colour management – Architecture, profile format, and data structure.

[9] ITU-R Recommendation BT.709, *Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange (1990)*. [formerly CCIR Rec. 709], ITU, 1211 Geneva 20, Switzerland.

[10] Jabłoński M., Bubliński Z., *Integracja toru wizyjnego na platformie rekonfigurowalnej – Video pipeline integration on reconfigurable platform.* Automatyka (półrocznik AGH), t. 12, z. 3, 2008, 657–667.

[11] Jabłoński M.: *Metodyka zrównoleglania algorytmów przetwarzania i analizy obrazów w systemach przepływowych.* Kraków, 2009 (rozprawa doktorska).

[12] Jiang H., Ardo H., Owall V., *Hardware accelerator design for video segmentation with multimodal background modeling*. International Symposium on Circuits and Systems, ISCAS 2005, vol. 2, May 2005, 1142–1145.

[13] Juvonen M.P.T., Coutinho J.G.F., Luk W., *Hardware Architectures for Adaptive Background Modelling*. 3rd Southern Conference on Programmable Logic, 2007. SPL '07, Feb. 2007, 149–154.
[14] Li L., Leung M.K.H., *Integrating Intensity and Texture Differences for Robust Change Detection*. IEEE Transactions on Image Processing, vol. 11, Iss. 2, 2002, 105–112.
[15] Musial M., Dybek D., Wojcikowski M., *Hardware realization of shadow detection algorithm in FPGA.* 2nd International Conference on Information Technology (ICIT), 2010, 201–204.
[16] Oliveira J., Printes A., Freire R.C.S., Melcher E., Silva I.S.S., *FPGA architecture for static background subtraction in real time*. Proc. of the 19th annual symposium on Integrated circuits and systems design (SBCCI '06), ACM, New York, NY, USA, 2006.
[17] Qin R., Liao S., Lei Z., Li S.Z., *Moving Cast Shadow Removal Based on Local Descriptors*. 20th International Conference on Pattern Recognition (ICPR), 23–26 Aug. 2010, 1377–1380.
[18] Stauffer C., Grimson W.E.L., *Adaptive background mixture models for real-time tracking*. Proc. IEEE CVPR, June 1999, 24&252.
[19] Wren C., Azarhayejani A., Darrell T., Pentland A.P., *Pfinder: real-time tracking of the human body.* IEEE Trans. on Pattern Anal. and Machine Intelligence, vol. 19, No. 7, 1997, 780–785.