AUTOMATYKA • 2006 • Tom 10 • Zeszyt 3

Marek Gorgoń*, Piotr Pawlik*, Mirosław Jabłoński*, Jaromir Przybyło*

Modelowanie i realizacja algorytmów wideodetekcji na platformie FPGA**

1. Wprowadzenie

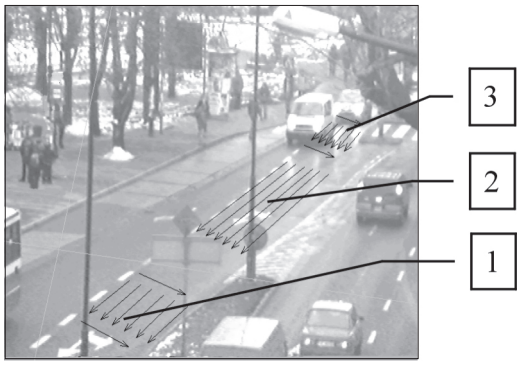
Wraz ze wzrostem mocy obliczeniowej i dostępności komputerów przemysłowych oraz spadkiem cen na urządzenia monitorujące zwiększa się zainteresowanie wideodetekcją, czyli automatycznym obliczaniem parametrów ruchu drogowego na podstawie wprowadzonego do komputera obrazu z kamery. Wideodetektory, czyli systemy zlokalizowane na skrzyżowaniu, złożone z kilku kamer i komputera analizującego obrazy cyfrowe, mają szansę stać się alternatywą dla powszechnie stosowanych w drogownictwie pętli indukcyjnych [2]. Przy niewielkich dodatkowych nakładach, obrazy z kamer mogą być transmitowane do centrum dyspozytorskiego, stanowiąc cenne źródło informacji o ewentualnych zagrożeniach płynności ruchu.

Ogrom informacji, który musi zostać przetworzony przez komputer analizujący obraz, powoduje, że w algorytmach wideodetekcji konieczne jest stosowanie redukcji danych wejściowych oraz optymalizowanie oprogramowania. Duże możliwości otwiera zastosowanie specjalizowanych procesorów przetwarzania obrazów w czasie rzeczywistym, opartych o technologię FPGA (*Field Programmable Gate Array*).

Na obecnym etapie rozwoju wideodetektorów wskazane jest, aby symulowały one działanie pętli indukcyjnych [1] (por. rys. 1). Ma to na celu zachowanie zgodności z istniejącym oprogramowaniem sterowników świateł. Zgodnie z zasadą działania pętli detekcja pojazdów odbywa się dwojako: wykrywa się ruch lub obecność pojazdu [2].

Wykrywanie pojazdów w ruchu opiera się na progowaniu obrazu, będącego różnicą dwóch kolejnych ramek [5]. W celu wykrycia pojazdów nieruchomych (np. oczekujących na zielone światło), analizuje się obraz różnicowy, otrzymany w wyniku odjęcia aktualnej ramki od obrazu referencyjnego tzw. tła. Obydwa te zadania (wykrywanie pojazdów ruchomych i nieruchomych) są realizowane z wykorzystaniem algorytmu SAD (*Sum of the Absolute value of Differences*), opisanego dokładniej w kolejnych rozdziałach.

* Akademia Górniczo-Hutnicza w Krakowie
** Praca finansowana ze środków na naukę, grant nr 4T11C01725



Rys. 1. Obiekty symulujące trzy pętle indukcyjne

Poprawna generacja tła dla wideodetekcji jest warunkiem przeprowadzenia poprawnych procesów wykrywania pojazdów. W aglomeracjach miejskich, w szczególności dla wielopasmowych arterii o dużym i bardzo dużym natężeniu ruchu, poprawne generowanie tła napotyka na liczne trudności. O ile w warunkach niskiego natężenia ruchu wystarczające okazują się algorytmy uśredniające, o tyle nie sprawdzają się one w warunkach wysokiego natężenia ruchu. W niniejszej pracy pokazano opracowaną modyfikację metody generacji tła (przedstawioną w [6]) dla wideodetektora oraz wyniki badań mających na celu adaptacje algorytmu wideodetekcji do implementacji na platformie FPGA. W szczególności chodzi tu o sposób realizacji algorytmu SAD dla dowolnie zdefiniowanych przez użytkownika fragmentów obrazu ROI (*Region of Interest*).

2. Modyfikacja algorytmu generacji tła

Programową realizację algorytmu generacji tła przedstawiono w pracy [6]. Bazuje ona na spostrzeżeniu, że zmiany wartości poszczególnych pikseli w kolejnych klatkach obrazu, po pewnym czasie (ok. 6000 klatek), tworzą funkcję, którą w uproszczeniu można określić jako jednomodalną. Na tej podstawie zauważono, że skutecznym sposobem estymacji tła może być obliczanie dla każdego piksela średniej ważonej przybieranych przez niego wartości. W pracy [6] do obliczania średniej ważonej dla k-tego piksela korzystano ze wzoru

$$M_k = \frac{\sum_{i=0}^{255} p_{i,k} \cdot X_{i,k}}{\sum_{i=0}^{255} p_{i,k}} \tag{1}$$

gdzie:

$X_{i,k}$ – wartość i-tej zmiennej losowej (od 0–255, ponieważ rozpatrywanych jest 256 poziomów szarości) dla k-tego piksela,

$p_{i,k}$ – liczba wystąpień i-tej wartości zmiennej losowej dla k-tego piksela (wartość z histogramu czasowego).

Wymagało to jednak pamiętania liczby wystąpień poszczególnych wartości każdego piksela w całym czasie działania algorytmu (czyli ciągłej aktualizacji histogramów czasowych przyporządkowanych każdemu pikselowi). W pracy [6] zaproponowano modyfikację metody średniej ważonej – elementy tła były wyznaczane jako lokalne maksima histogramów czasowych leżące w pobliżu wyznaczonej średniej ważonej.

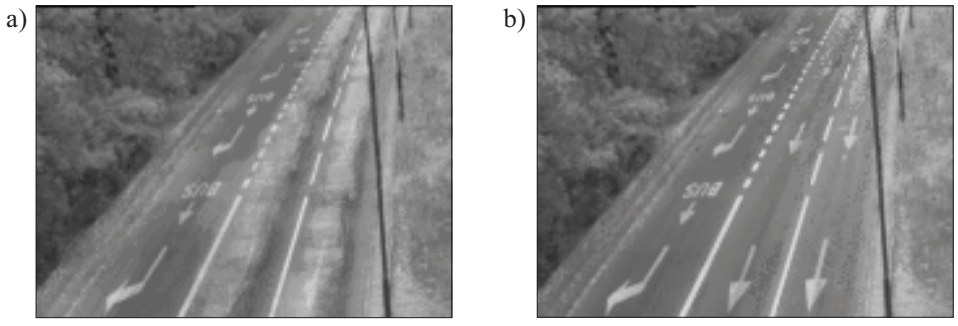
Powyższa metoda stawia wysokie wymagania pamięciowe (przykładowo dla rozdzielczości 512×512 potrzebne jest 134 217 728 bajtów, jeśli przyjmie się, że każdy histogram zajmuje 256 liczb dwubajtowych) oraz wymaga przeliczania wartości każdego histogramu celem wyliczenia średniej. Przy analizie zmierzającej do przeniesienia opisanej metody na platformę sprzętową zauważono, iż zamiast liczyć średnią ważoną z histogramu, można wykorzystać akumulację kolejnych wartości piksela

$$M_k = \frac{\sum_{t=0}^{T} X_t}{T} \tag{2}$$

gdzie X_t – wartość piksela dla chwili czasowej (ramki) t.

Powyższy sposób liczenia pozwala uzyskać identyczne tło jak przy użyciu histogramów czasowych określonych we wzorze (1). Nie można jednak dokonać za jego pomocą wspomnianej powyżej modyfikacji metody histogramowej, polegającej na wyznaczaniu lokalnego maksimum w histogramie. Z drugiej strony otrzymuje się poważną oszczędność wykorzystania pamięci (potrzebne jest do 256 razy mniej pamięci!), oraz pewną oszczędność czasu obliczeń (nie ma potrzeby liczenia licznika we wzorze (1)). W trakcie przeprowadzonych prób okazało się, że zysk ze stosowania modyfikacji związanej z maksimum lokalnym (por. rys. 2a) jest niewspółmiernie mały w porównaniu z ponoszonymi nakładami na pamiętanie i przeliczanie histogramów.

Drugą, o wiele istotniejszą modyfikacją metody średniej ważonej, zawartą w pracy [6], była propozycja wstrzymania wyznaczania wartości tła w czasie oczekiwania samochodów na zmianę świateł. Wyznaczanie tła zostało tam powiązane zwrotnie z detekcją obecności i ruchu samochodów.



Rys. 2. Tło wygenerowane na podstawie: a) histogramów czasowych – widoczne wyraźnie smugi w obrębie pasów ruchu; b) średniej ważonej (włączony warunek ruch)

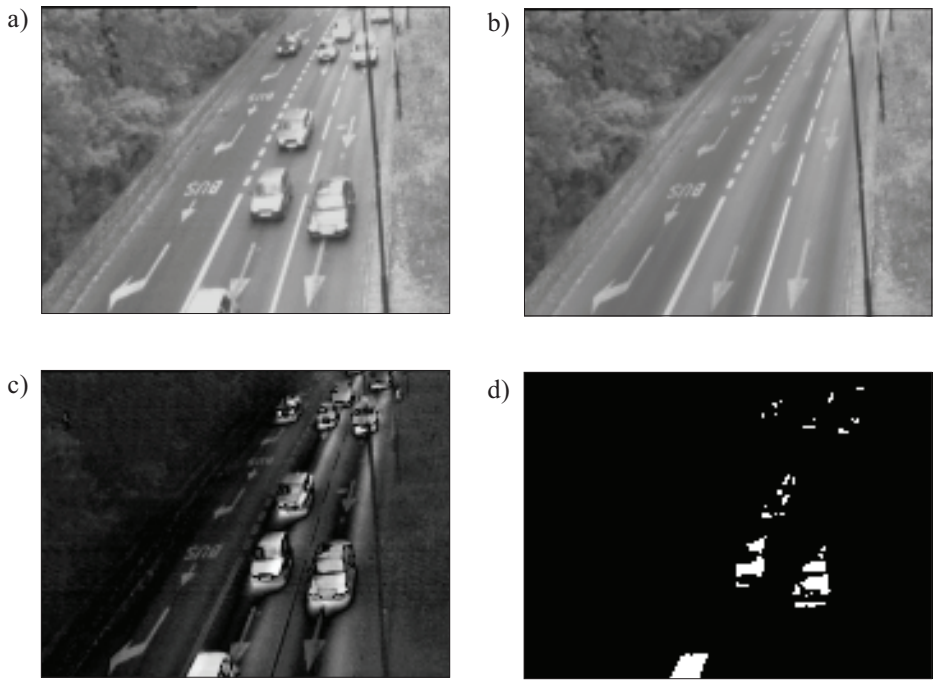
Szczegółowa obserwacja materiału filmowego, przeprowadzona w trakcie obecnych badań, doprowadziła do postawienia tezy, że w okresach dużego natężenia ruchu rozkład poziomów szarości dla punktów pasa drogowego, w momencie obecności i ruchu, jest mniej korzystny z punktu widzenia poprawnego wyznaczenia tła niż rozkład wynikły z analizy fragmentów, gdzie występuje ruch, a obecność nie jest sprawdzana. Akumulacja z włączonymi jednocześnie warunkiem ruchu i detekcją obecności doprowadza bowiem do wykrywania punktów pojazdów, a nie punktów tła. Przy akumulacji w okresach ruchu w polach wideodetekcji występują poziomy szarości odpowiadające za tło (odcień jezdni lub namalowanego znaku poziomego) oraz odcienie szarości związane z pikselami pojazdów. Przy czym statystycznie, piksele pojazdów rozkładają się równomiernie, a dominująca (dla obliczenia średniej) jest wartość związana z tłem. Analiza histogramów czasowych potwierdziła, że własność ta jest obserwowana po kilkuset klatkach obrazu, co w procesie generacji tła jest czasem krótkim.

Na potrzeby implementacji sprzętowej został opracowany wariant powyższej modyfikacji, w którym generacja tła następowała tylko podczas ruchu samochodów. Jeżeli sumaryczny ruch wykrywany przez odjęcie dwóch kolejnych klatek był mniejszy od założonego progu, to generacja tła była wstrzymywana. Wyniki wykazują znaczną poprawę w stosunku do metody proponowanej w pracy [6] (por. rys. 2 b).

Należy zwrócić uwagę, że algorytm wykorzystujący akumulację w okresach ruchu sprawdza się w warunkach dużego i średniego natężenia pojazdów, lecz w okresach incydentalnego ruchu na całej jezdni, lub choćby dla pojedynczego pasa, tło nie będzie odświeżane. Stąd w dalszej części prac określony zostanie dodatkowy warunek uruchamiania akumulacji w sytuacjach braku ruchu przez określony czas. Jest też konieczne zróżnicowanie rodzaju używanego algorytmu dla poszczególnych pasów ruchu, ponieważ brak ruchu może wystąpić jedynie na jednym z pasów ruchu (np. na pasie do skrętu). Stąd konieczne stało się określenie osobno sprawdzanych obszarów detekcji dla poszczególnych pasów ruchu.

Implementacja sprzętowa narzucała także pewne ograniczenia algorytmu wyznaczania średniej ważonej. Bezpośrednie zastosowanie wzoru (2) powodowałoby konieczność dzielenia zmiennoprzecinkowego, którego realizacja w zastosowanej platformie sprzętowej byłaby kosztowna pamięciowo i nieefektywna ze względów czasu obliczeń. Rozwiązanie praktycznego braku operacji dzielenia zostało powiązane z innym problemem występującym w obu wspomnianych metodach. Zarówno w rozwiązaniu „histogramowym", jak i we wzorze (2) występuje sumowanie wartości z kolejnych klatek (akumulacja w czasie). Przy ciągłym działaniu aplikacji opartej o powyższe algorytmy prowadziło by to do przepełnień. Stąd wprowadza się okresowe „obniżanie" histogramów (w pierwszym przypadku) i sumy wartości pikseli (w drugim przypadku), realizowane przez okresowe dzielenie narastających wartości (wraz z dzieleniem licznika klatek). Na potrzeby implementacji sprzętowej dokonano następującej modyfikacji algorytmu. Do pewnej wartości licznika klatek (będącej potęgą dwójki – eksperymentalnie stwierdzono, że wystarczająca jest wartość 512) następuje tylko sumowanie wartości pikseli bez wyznaczania średniej ważonej. Dzięki osiągnięciu przez licznik klatek wartości będącej potęgą dwójki, dzielenie występujące we wzorze (2) można zastąpić przesunięciem (operacją *shift*). Jednocześnie dokonuje się cy-

klicznego obniżenia wartości sumy pikseli oraz odpowiednio licznika klatek, przy czym dzielenie zostaje zastąpione łatwiej realizowalnym i szybszym mnożeniem stałoprzecinkowym przez odwrotność. Mnożnik jest tak dobrany, aby zmniejszyć licznik klatek o żądaną wartość (wartość ta określa, co ile klatek ma być wyliczane tło – w prezentowanym przykładzie przyjęto 32). Po ponownym osiągnięciu przez licznik klatek zadanej wartości (512), operacja jest ponawiana. Dzięki temu zabiegowi suma wartości pikseli oscyluje, zamiast rosnąć do przepełnienia, a niejako „przy okazji" kłopotliwe dzielenie można zastąpić o wiele szybszym przesuwaniem.



Rys. 3. Przykładowe zastosowanie wyznaczonego tła do detekcji obecności pojazdów:
a) klatka (kadr) filmu; b) wygenerowane tło; c) obraz różnicowy kadru i tła;
d) wynik segmentacji binaryzacją procentową

Na rysunku 3 pokazano wyniki kolejnych etapów wideodetekcji. Po wygenerowaniu tła dalszy etap wideodetekcji stanowi wyliczenie obrazu różnicowego (rys. 3c) pomiędzy klatką bieżącą (rys. 3a) a bieżącym obrazem tła (rys. 3b). Na potrzeby testowania jakości metody generacji tła zastosowano jako ostatni etap metodę segmentacji wykorzystującej binaryzację procentową. Wyniki segmentacji procentowej, na obecnym etapie badań są orientacyjną, ale wystarczającą miarą poprawności działania algorytmu generacji tła. Przykładowo, brak jasnych punktów na rysunku 3d w miejscach występowania znaków poziomych na jezdni oraz niewystępowanie smug na pasach o dużym natężeniu ruchu (pasy do jazdy na wprost) świadczy o poprawnym działaniu algorytmu generacji tła.

3. Algorytm SAD

3.1. Programowa implementacja referencyjna

Algorytm SAD wykorzystywany jest w wielu aplikacjach, w których konieczna jest estymacja ruchu, głównie jako miara dopasowania wzorców (*template matching*). Przykładowe aplikacje – kompresja MPEG sygnału wideo – przedstawione zostały przez autorów prac: [4, 7].

W powyższych przypadkach operacja SAD może być bardzo czasochłonna, ponieważ wyznaczana jest dla dużych obszarów obrazu oraz dla wielu różnych szablonów. Istnieje wiele prób przyspieszenia z wykorzystaniem implementacji sprzętowej. Przykładowo w pracy [8] przedstawiona została analiza implementacji sprzętowej algorytmu SAD w układach FPGA. Autorzy rozdzielili operacje SAD na dwa etapy:

1) wyznaczanie wartości bezwzględnej,
2) sumę.

W oparciu o estymaty oczekiwanej prędkości oraz obszaru, zaimplementowana została operacja SAD przy wykorzystaniu generatora przeniesienia (etap wartości bezwzględnej) oraz drzewo sumatorów (etap wyznaczania sumy). Do implementacji wykorzystany został język VHDL.

Równanie dwuwymiarowego dyskretnego algorytmu SAD jest następujące

$$C(j,k) = \sum_{m=0}^{(Mt-1)} \sum_{n=0}^{(Nt-1)} abs(I(m+j, n+k) - T(m,n)) \tag{3}$$

gdzie:

$0 \le j < Mi - Mt + 1$,

$0 \le k < Ni - Nt + 1$,

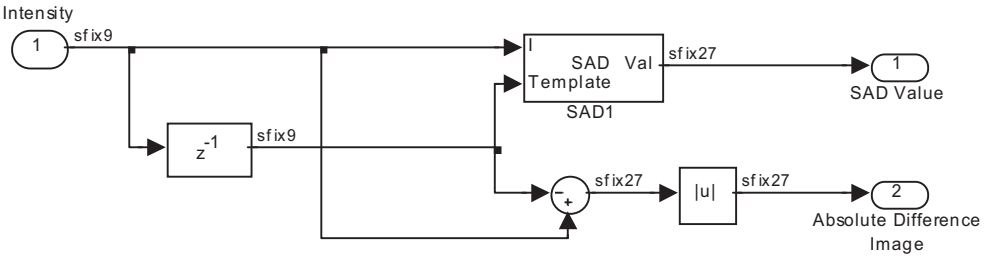
I – obraz wejściowy [$Mi \times Ni$],

T – szablon [$Mt \times Nt$].

Bezpośrednie podejście w obliczaniu algorytmu SAD, wykorzystane w niniejszej pracy, składa się z następujących etapów:

1. obliczenie różnicy pomiędzy poszczególnymi pikselami obrazu a szablonu $I_i - T_i$;
2. dla pikseli, dla których $I_i - T_i$ jest mniejsza od zera, wyznaczenie $T_i - I_i$ jako modułu z różnicy;
3. akumulacja (suma) po wszystkich wyznaczonych wartościach.

Na rysunku 4 przedstawiono model algorytmu SAD przygotowany w Simulinku, wykorzystywany później jako algorytm referencyjny do implementacji FPGA.



Rys. 4. Schemat blokowy algorytmu SAD

3.2. Przygotowanie algorytmu referencyjnego SAD do implementacji sprzętowej

Algorytm referencyjny zaimplementowany w Simulinku, w podstawowej wersji, został przystosowany do obliczeń na liczbach zmiennoprzecinkowych pojedynczej precyzji. Realizacja arytmetyki zmiennoprzecinkowej w układach reprogramowalnych jest zadaniem trudnym i wymagającym wykorzystania dużych zasobów. Algorytm SAD może jednak zostać, bez utraty dokładności, przystosowany do pracy z arytmetyką całkowitoliczbową lub stałoprzecinkową. Dzięki temu realizacja sprzętowa jest bardzo efektywna – szybka i niewymagająca wykorzystania dużych zasobów układu FPGA.

Aby przystosować ww. algorytm do implementacji sprzętowej, dokonana została analiza zakresu danych wymaganych na poszczególnych etapach obliczeń. Przy założeniu, iż wartość SAD wyliczana jest dla prostokątnego obszaru obrazu o wielkości [M×N], i typie danych – 8-bitowe liczby bez znaku (obraz w 256 odcieniach szarości), wymagane zakresy danych są następujące:

– do obliczania różnicy pomiędzy obrazem a szablonem – 9-bitowe liczby ze znakiem,
– akumulator wartości SAD – wymagana liczba bitów reprezentacji liczbowej może zostać wyznaczona z następującego wzoru

$$NrOfBits = \text{ceil}(\max(\log_2(MaxSAD))) \tag{4}$$

gdzie: $MaxSAD = M * N * 255$

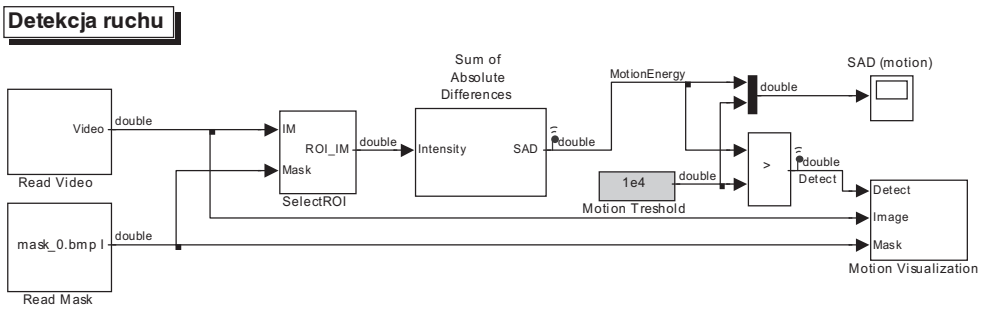
– pozostałe części algorytmu mogą wykorzystywać reprezentację liczb 8-bitowych bez znaku.

3.3. Detekcja ruchu – model referencyjny

Przedstawiony algorytm SAD został wykorzystany w algorytmie detekcji ruchu. Schemat blokowy modelu referencyjnego przedstawiony został na rysunku 5.

Algorytm przystosowano do pracy z wybranym regionem zainteresowań (ROI), definiowanym przez binarną maskę. Z kolejnych klatek wejściowej sekwencji video, wybierane są piksele odpowiadające odpowiednim wartościom maski (operacja AND). Następnie dla obszaru ROI wyznaczana jest wartość SAD pomiędzy aktualną, a poprzednią klatką

(sygnał *Motion energy*). Wartość ta porównywana jest następnie z zadanym progiem (*Motion Treshold*). Jeśli wartość SAD przekroczy zadany próg, sygnalizowane jest zdarzenie wykrycia ruchu (sygnał *Detect*). Wizualizacja rezultatów odbywa się w podsystemie Motion Visualization.



Rys. 5. Schemat blokowy algorytmu detekcji ruchu

Sygnały *Motion energy*, *Detect* wykorzystywane są następnie jako wektory testowe do weryfikacji implementacji sprzętowej algorytmu w języku Handel-C. W tym celu konwertowane są na odpowiedni format (tekstowy plik wartości heksadecymalnych), akceptowany przez środowisko Handel-C.

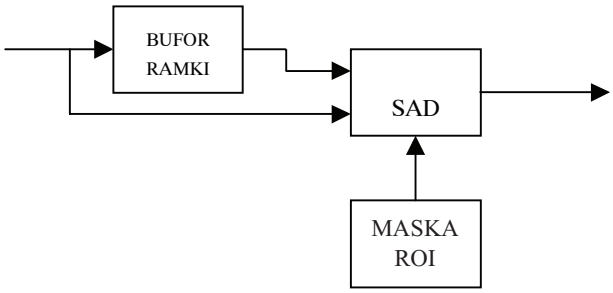
4. Algorytm SAD dla obszarów nieregularnych – realizacja sprzętowa

Zmodyfikowany algorytm SAD wyznacza miarę różnicy dwóch obrazów w dowolnie określonych, również nieregularnych, obszarach. Obszary zainteresowania ROI mogą być łatwo definiowane przez operatora systemu w postaci maski – binarnej mapy bitowej, lub też wyznaczane dynamicznie w czasie działania algorytmu analizy obrazu. Multiplikacja jednostek elementarnych jednostek obliczeniowych i zwielokrotnienie pamięci masek ROI pozwala łatwo zwiększyć liczbę analizowanych obszarów, przy czym w zależności od potrzeb, obszary te mogą na siebie zachodzić lub stanowić zbiór rozsianych grup pikseli obrazu.

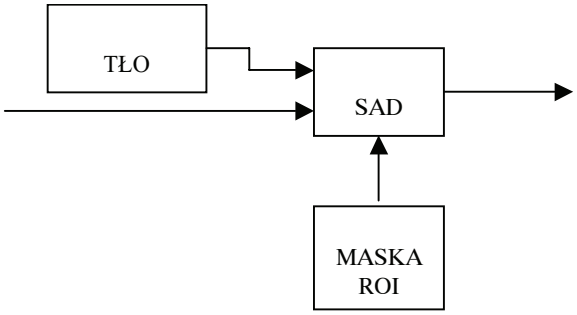
Pomimo swojej prostoty, algorytm znajduje wielorakie zastosowania na różnych etapach procesu analizy obrazu. Najprostszą i zarazem w pełni kompletną aplikację stanowi algorytm detekcji ruchu wykorzystujący moduł SAD i bufor ramki obrazu (rys. 6). Sygnałem wejściowym systemu jest obraz w postaci strumienia pikseli, a wyjście binarne awizuje o detekcji ruchu wyznaczonego z dwóch kolejnych ramek obrazu.

Kolejny schemat aplikacyjny (rys. 7) pokazuje sposób określania miary rozbieżności obrazu ze zdefiniowanym lub dynamicznie wyznaczonym wzorcem. W szczególności, obrazem referencyjnym może być wyznaczane w sposób ciągły tło obrazu, a wynik obliczeń awizuje chwilową obecność statycznych obiektów w wyznaczonym obszarze. Algorytm SAD dla obszarów nieregularnych został zaimplementowany w języku Handel-C jako kom-

ponent biblioteki PixelStreams. Zarys wykorzystanego środowiska rozwojowego przedstawiono w pracy [3]. Cztery współbieżnie działające jednostki obliczeniowe wyznaczają wartość sygnału różnicowego oraz binarny rezultat dla masek zdefiniowanych w pamięciach RAM.



Rys. 6. Detektor ruchu



Rys. 7. Detektor obecności

Sprzętowy model funkcjonalny był symulowany w środowisku DK4 i weryfikowany wektorami testowymi otrzymanymi z modelu napisanego w Simulinku. Aplikacja testowa detektora ruchu została zrealizowana i przetestowana na platformie multimedialnej RC300 z układem FPGA XC2V6000. Wyniki implementacji algorytmu SAD wyliczanego równolegle dla czterech obszarów ROI przedstawiono w tabeli 1.

Tabela 1
Rezultaty implementacji sprzętowej algorytmu SAD (dla podziału obrazu na cztery obszary ROI)

Zasoby układu FPGA	Użyte w implementacji	W stosunku do zasobów układu %
Liczba użytych elementów Slices:	3,465	11
Liczba użytych elementów IOBs:	194	23
Liczba użytych elementów Block RAMs:	8	5
Łączna liczba przeliczeniowych bramek logicznych	649,799	11

5. Wnioski

W pracy przedstawiono algorytm generacji tła dla zastosowań w wideodetekcji pojazdów. Algorytm przebadano dla sekwencji testowej o dużym natężeniu ruchu pojazdów. Zaproponowany algorytm dokonywał akumulacji obrazów jedynie w okresach ruchu pojazdów. Wygenerowane tło pozbawione było smug stanowiących ślad po poruszających się pojazdach. Smug takich, będących zakłóceniami, nie udawało się usunąć we wcześniejszych badaniach. Wskazano na konieczność prowadzenia akumulacji w okresach całkowitego braku ruchu pojazdów. Szczegółowe kryteria akumulacji tła w takich warunkach należy określić w dalszych badaniach. Przedstawiono realizację programową i sprzętową algorytmu SAD. W implementacji sprzętowej zaproponowano wprowadzenie maski, pozwalającej definiować do 32 niezależnych obszarów ROI o dowolnym kształcie. Umożliwi to prowadzenie wideodetekcji w sposób równoległy dla określonych fragmentów obrazu, wyznaczanych jako odpowiedniki pól detekcji pętli fazowych.

Literatura

[1] Adamski A., Mikrut Z.: *Traffic video-detector application to intelligent control*. Proc. of Int. Conf. on Modeling and Management in Transportation, Poznań-Kraków, 1999

[2] Leśko M., Guzik J.: *Sterowanie ruchem drogowym. Sygnalizacja świetlna i detektory ruchu pojazdów*. Wydawnictwo Politechniki Śląskiej, Gliwice, 2000

[3] Jabłoński M., Przybyło J., Gorgoń M.: *Real time implementation of motion detection algorithm based on Pixelstreams.* Proceedings of IFAC workshop on Programmable Devices and Embedded Systems, IFAC, Brno, 2005, 186–190

[4] Koga, T., *et al.*: *Motion-compensated interframe coding for video conferencing*. In: Nat. Telecommun. Conf., New Orleans, LA, 1981, G5.3.1–5

[5] Mikrut, Z.: *Road Traffic Measurement Using Videodetection*. Image Processing and Communications, vol. 3, nr 3–4, 1997, 19–30

[6] Olechowski R., Nowak M.: *Konfiguracja i współpraca programów analizy ruchu drogowego w systemie Linux*. Wydział EAIiE, AGH, Kraków, 2003 (praca dyplomowa)

[7] Wang Y., Ostermann J., Zhang Y.-Q.: *Video Processing and Communications*. Prentice Hall, Upper Saddle River, NJ, 2002

[8] Wong S., Stougie B., Cotofana S.: *An Investigation on FPGA based SAD Hardware Implementations*. In: Proceedings of the 13th Annual Workshop on Circuits, Systems, and Signal Processing (PRORISC2002), Veldhoven, The Netherlands, 2002, 568–573