ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: AUTOMATYKA z. 113 Nr kol. 1246

Zygmunt KUŚ
Konrad WOJCIECHOWSKI

PRZETWARZANIE STOSOWE

Streszczenie. W pracy zostały przedstawione podstawowe właściwości sposobu przetwarzania obrazów nazywanego przetwarzaniem stosowym - filtracją stosową (stack filtration).

Zdefiniowano takie podstawowe pojęcia z dziedziny przetwarzania obrazów, jak: obraz jednowymiarowy, obraz dwuwymiarowy i odpowiednie dla nich apertury. Następnie podano definicje progowej dekompozycji, właściwości układania w stos oraz inne definicje i twierdzenia związane z metodą przetwarzania stosowego, a także przykłady wykonania przetwarzania stosowego na obrazach dwuwymiarowych o 16 poziomach szarości.

W pracy przedstawiono również metodę projektowania funkcji boolowskich realizujących przetwarzanie ze statystyką porządkową dla apertury o dowolnej szerokości i kwantyla dowolnego rzędu.

Z przedstawionych rozważań wynika, że podejście do przetwarzania obrazów oferowane przez metodę przetwarzania stosowego jest szczególnie atrakcyjne ze względu na możliwość zwiększenia szybkości działania, uproszczenia konstrukcji oraz stworzenia wielu nowych rodzajów przetworników.

STACK FILTRATION

Summary. In this work basic properties of stack filtration are presented. The basic terms of image processing such as 1D image, 2D image and appropriate apertures are defined. Afterwards, the definition of threshold decomposition, stacking property and some other definitions and thesis concerning stack filtration are given as well as examples of stack filtration of 16 - valued 2D images.

The work presents also a desing method for Boolean function proceeding with image processing in rank order for an aperture with arbitrary width and order of quantile.

The work emhasises great advantages of stack filter image processing coming from availability of speed increase for procedures required, desing simplification and creation of variety of new filters.

СТЕКОВАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ

Резюме. В работе представлены основные свойства стековой фильтрации. Определены также основные понятия из области обработки изображений такие, как: одномерное изображение, двухмерное изображение и соответствующие апертуры. Даны определения порогового разбиения, свойства построения стека и другие определения и теоремы связанные с методом стековой обработки изображений. Приведены примеры стековой обработки для двухмерных изображений с 16 уровнями серого. Представлен также метод проектирования логических функций реализирующих обработку с порядковой статистикой для апертуры произвольной ширины и квантили произвольного порядка. Показано, что стековой подход к обработке изображений особо привлекательный из-за возможности увеличения скорости действия, упрощения конструкции и построения видов преобразователей.

1. WSTĘP

W dziedzinie przetwarzania obrazów występują obecnie dwa trendy. Po pierwsze prowadzone są poszukiwania nowych algorytmów przetwarzania, a po drugie poszukuje się nowych sposobów realizacji znanych już algorytmów. Powodem tych poszukiwań jest dążenie do przyspieszenia działania układów i oprogramowania oraz zmniejszenia kosztów produkcji. Pod względem szybkości działania szczególnie duże wymagania stawia przetwarzanie obrazów w czasie rzeczywistym.

Poszukiwania możliwości zwiększenia szybkości działania oraz uproszczenia konstrukcji powodują, że podejście do przetwarzania obrazów oferowane przez metodę przetwarzania stosowego jest szczególnie atrakcyjne. Metoda polega na dekompozycji ciągu wielowartościowego na ciągi binarne, wykonaniu operacji przetwarzania na tych ciągach, co jest szczególnie proste, a następnie powrocie do ciągu wielowartościowego przez "sumowanie" wynikowych ciągów binarnych.

Wspomniana powyżej dekompozycja pozwala na realizację znanych algorytmów przetwarzania np. filtracji medianowej w odmienny niż do tej pory sposób. Filtracja medianowa dla ciągu wielowartościowego jest "złożeniem" binarnych filtracji medianowych wykonanych na uzyskanych w wyniku dekompozycji ciągu wielowartościowego ciągach binarnych. Z kolei dzięki temu, że wyjście binarnej filtracji medianowej jest znajdowane przez sumowanie liczby jedynek w podciągu obejmowanym przez aperturę i dzięki porównaniu wyniku z progiem można ominąć etap porządkowania występujący w konwencjonalnej realizacji filtracji medianowej. W konsekwencji filtr medianowy zrealizowany z wykorzystaniem metody, *przetwarzania stosowego* składa się jedynie z sumatorów i układów progujących i może być wykonany

Praca finansowana z grantu wewnętrznego Politechniki Śląskiej BK 465.

w technologii VLSI (Very Large Scale Integration). Wykorzystanie technologii VLSI umożliwia wzrost szybkości działania pozwalający za zrealizowanie przetwarzania obrazów w czasie rzeczywistym.

Kolejną ważną cechą przetworników stosowych jest możliwość konstrukcji dużej liczby różnych przetworników. Właściwości wielu z nich nie zostały dotychczas jeszcze zbadane. Dla ilustracji tej różnorodności powiedzmy tylko za [2], że przetworników z aperturą szerokości 3 jest 20, z aperturą 5 jest ich 7581, a z aperturą 7 jest ich co najmniej 2^{35}.

Powyższe względy sprawiają, że przetwarzanie stosowe, któremu niniejsza praca jest poświęcona, otwiera nowe możliwości w realizacji znanych już algorytmów przetwarzania oraz w poszukiwaniu nowych.

W pracy zdefiniowano w rozdziale 2 takie podstawowe pojęcia z dziedziny przetwarzania obrazów, jak: obraz jednowymiarowy, obraz dwuwymiarowy i odpowiednio dla nich apertury. Następnie w rozdziale 3 podano definicje progowej dekompozycji, właściwości układania w stos oraz inne, definicje i twierdzenia związane z metodą przetwarzania stosowego, a także przykłady wykonania przetwarzania stosowego na obrazach dwuwymiarowych o 16 poziomach szarości. Z kolei w rozdziale 4 opisano metodę projektowania funkcji boolowskich realizujących przetwarzanie ze statystyką porządkową dla apertury o dowolnej szerokości i kwantyla dowolnego rzędu.

Zarówno rozważania teoretyczne, jak i praktyczna programowa realizacja filtrów dla obrazów o rozmiarach 256x256 punktów i 16 poziomach szarości pokazują, że przetwarzanie stosowe pozwala na wykonanie w prostszy sposob wielu znanych już rodzajów filtracji oraz oferuje ogromne możliwości w konstrukcji nowych rodzajów przetwarzania obrazów.

2. PODSTAWOWE DEFINICJE I POJĘCIA

Przechodząc do definicji podstawowych pojęć wykorzystywanych przy przetwarzaniu obrazów wyróżniamy: obraz jednowymiarowy, obraz dwuwymiarowy, aperturę dla obrazu jednowymiarowego oraz aperturę dla obrazu dwuwymiarowego.

Definicja 2.1

Obrazem jednowymiarowym nazywamy wektor:

$$X = [x_i], \qquad x_i \in \{0,1,\ldots M-1\}, \qquad i = 0,1,\ldots I-1$$

gdzie:

x_i - jest punktem obrazu (pikslem) w i-tym miejscu obrazu lub wartością sygnału w i-tej chwili.

M - odpowiada liczbie poziomów szarości, jakie mogą przyjmować punkty obrazu lub liczbie wartości, które może przyjmować sygnał.

I - jest liczbą punktów obrazu.

□

Przykład 2.1

Obrazem jednowymiarowym przy I=10, M = 4 może być na przykład ciąg:

$$(1, 3, 0, 0, 1, 2, 2, 2, 1, 3).$$

□

Definicja 2.2

Obrazem dwuwymiarowym nazywamy macierz:

$$X = [x_{i,j}]$$

$$x_{i,j} \in \{0,1,\ldots M-1\}, \quad i = 0,1,\ldots I-1, j = 0,1,\ldots J-1$$

gdzie:

$x_{i,j}$ - jest punktem obrazu (pikslem) w i-tym wierszu i j-tej kolumnie,

M - odpowiada liczbie poziomów szarości, jakie mogą przyjmować punkty obrazu,

I·J - liczba punktów obrazu.

□

Przykład 2.2

Obrazem dwuwymiarowym przy I = 5, J = 5, M = 4 może być na przykład macierz:

$$\begin{bmatrix} 0 & 2 & 3 & 0 & 1 \\ 1 & 1 & 0 & 3 & 2 \\ 2 & 1 & 2 & 3 & 3 \\ 3 & 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 2 & 3 \end{bmatrix}.$$

□

Definicja 2.3

Aperturą dla obrazu jednowymiarowego nazywamy zbiór A:

$$A = \{r_1, r_2, \ldots, r_k, \ldots, r_\sigma\}, \quad r_k \in Z,$$

σ - szerokość apertury, Z - zbiór liczb całkowitych, którego elementami są liczby całkowite.

□

Jeżeli w danej chwili ustalimy w obrazie i-ty punkt obrazu, to nałożenie apertury na obraz jest równoważne wybraniu z obrazu podzbioru punktów o następujących współrzędnych:

$$\{i + r_1, \quad i + r_2, \ldots, \quad i + r_k, \ldots, \quad i + r_\sigma\}.$$

Możliwe są różne uporządkowania podzbioru punktów wybranego z obrazu. Jednym z możliwych jest uporządkowanie według porządku naturalnego w zbiorze A.

Tak wybrany podzbiór punktów obrazu można wykorzystać w procesie przetwarzania obrazu. W wyniku wykonania na tym podzbiorze punktów operacji określonych przez rodzaj przetwarzania otrzymujemy nowy poziom szarości (nową wartość sygnału) w i-tym punkcie obrazu.

W praktyce stosuje się apertury o szerokości σ (liczbie określanych przez aperturę punktów) nieparzystej, zawierające w sobie wyróżniony element nazywany punktem centralnym apertury oraz posiadające symetrię środkową względem niego.

Definicja 2.4

Punktem centralnym apertury jednowymiarowej A nazywamy element (0).

□

Definicja 2.5

Mówimy, że apertura jednowymiarowa A posiada symetrię środkową względem punktu centralnego, jeżeli:

$$\text{i)} \quad (0) \in A,$$

$$\text{ii)} \quad \big((r) \in A\big) \Rightarrow \big((-r) \in A\big)$$

□

Poniżej podano przykład apertury jednowymiarowej zawierającej punkt centralny i posiadający symetrię środkową względem niego.

Przykład 2.3

Nałożenie apertury A = {-2, -1, 0, 1, 2} na obraz z przykładu 2.1. dla piątego punktu obrazu (i = 4) powoduje otrzymanie podzbioru punktów {0, 0, 1, 2, 2}.

□

Definicja 2.6

Aperturą dla obrazu dwuwymiarowego nazywamy zbiór A:

$$A = \{(r_1, s_1), (r_2, s_2), \ldots (r_k, s_k) \ldots (r_\sigma, s_\sigma)\},$$

$$r_k, s_k \in Z, \quad r - \text{w osi } X, \quad s - \text{w osi } Y,$$

Z - zbiór liczb całkowitych, σ - szerokość apertury,
którego elementami są pary liczb całkowitych.

□

Jeżeli w danej chwili ustalimy w obrazie punkt o współrzędnych (i, j), to nałożenie apertury na obraz jest równoważne wybraniu z obrazu podzbioru punktów o następujących współrzędnych:

$$\{(i+s_1, j+r_1), (i+s_2, j+r_2), \ldots, (i+s_k, j+r_k), \ldots, (i+s_\sigma, j+r_\sigma)\}.$$

Uporządkowanie podzbioru punktów wybranych z obrazu może być przeprowadzone dowolnie i jest osobnym zagadnieniem, które nie będzie tutaj szczegółowo rozpatrywane. Brak tak jednoznacznie narzucającego się uporządkowania jak w przypadku obrazów jednowymiarowych wprowadza pewną dowolność. W wyniku wykonania na tym podzbiorze punktów pewnych operacji, określonych przez rodzaj przetwarzania, otrzymujemy nowy poziom szarości w punkcie obrazu o współrzędnych (i, j).

W ogólnym przypadku aperturę dwuwymiarową określa dowolny zbiór par liczb całkowitych. W praktyce stosuje się apertury posiadające pewne dodatkowe właściwości, jak istnienie punktu centralnego apertury i symetrię środkową względem niego oraz mające nieparzystą szerokość σ (liczbę określanych przez aperturę punktów).

Definicja 2.7

Punktem centralnym apertury dwuwymiarowej nazywamy element (0,0).

□

Definicja 2.8

Mówimy, że apertura dwuwymiarowa A posiada symetrię środkową względem punktu centralnego, jeżeli:

$$\text{i)} \quad (0,0) \in A,$$

$$\text{ii)} \quad ((r,s) \in A) \Rightarrow ((-r,-s) \in A)$$

□

Poniżej podano przykład apertury dwuwymiarowej zawierającej punkt centralny i posiadający symetrię środkową względem niego.

Przykład 2.4

Ustalmy aperturę następującej postaci:

$$A = \{(0,0), (0,1), (0,-1), (-1,0), (1,0)\}.$$

W dziedzinie przetwarzania obrazów przyjęła się konwencja graficznej interpretacji apertury z punktem centralnym w postaci punktów w prostokątnym układzie współrzędnych o początku w punkcie centralnym apertury. Interpretacja graficzna apertury z przykładu ma następująca postać:

$$\begin{bmatrix} & * & \\ * & * & * \\ & * & \end{bmatrix}.$$

Jeśli jako ustalony przyjmiemy punkt obrazu, dla którego i = 3 oraz j = 4, to przez nałożenie takiej apertury na obraz z przykładu 2.2. otrzymamy następujący podzbiór punktów obrazu: {3, 3, 1, 2, 3}, który odpowiada wybraniu następującego fragmentu obrazu:

$$\begin{bmatrix} & 3 & \\ 2 & 3 & 3 \\ & 1 & \end{bmatrix}.$$

Uporządkowanie punktów tego podzbioru nie jest jednoznacznie określone. Można je traktować jako ciąg (3, 2, 3, 3, 1), ciąg (2, 3, 3, 1, 3), ciąg (1, 2, 3, 3, 3) lub ciąg tych samych liczb w jeszcze innej kolejności.

□

Na przykład w realizacji programowej uporządkowanie podzbioru punktów wyczytanych z obrazu zostało określone kolejnością wprowadzania punktów apertury przy jej definiowaniu w pracy przy programu.

Mając zdefiniowane powyżej podstawowe obiekty przejdźmy do idei przetwarzania obrazów. Polega ona na zastąpieniu poziomu szarości każdego punktu obrazu oryginalnego przez poziom szarości wyliczony na podstawie poziomów szarości punktów z jego otoczenia. O tym, które i ile punktów z otoczenia będzie brane do obliczeń, przesądza kształt i szerokość apertury, natomiast sposób obliczania nowego poziomu szarości zależy od rodzaju przetwarzania. Jeżeli zarówno kształt apertury, jak i rodzaj przetwarzania dobieramy pod katem eliminacji pewnych szczególnych elementów obrazu (zakłóceń), to możemy przetwarzanie nazwać filtracją.

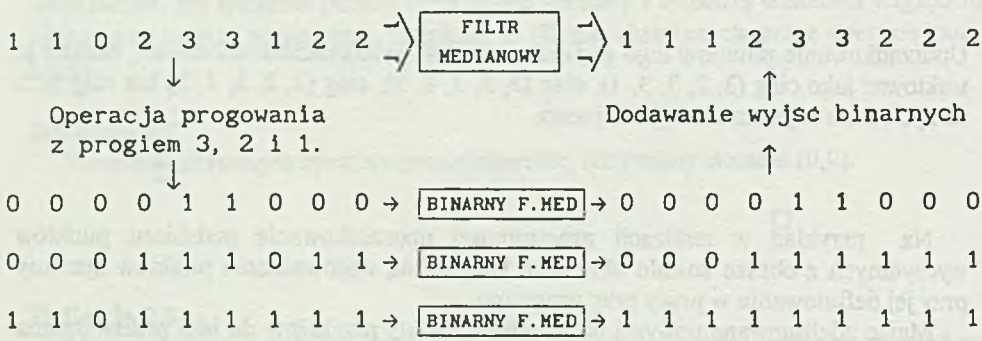
Jak wspomniano przy definiowaniu apertury dla obrazów dwuwymiarowych, problem uporządkowania podzbioru punktów wybranych z obrazu dwuwymiarowego w jednowymiarowy ciąg jest osobnym problemem, który tutaj nie będzie szczegółowo omawiany. Jest on jednak bardzo znaczący, gdyż do wyliczania poziomów szarości w obrazie po przetworzeniu wykorzystuje się funkcje o argumentach w postaci jednowymiarowych ciągów zmiennych (funkcje boolowskie, operatory progowania, operatory porządkowania). Do niektórych z nich kolejność argumentów ma znaczenie (np. dla nieprzemiennych funkcji boolowskich), a dla niektórych nie np. dla operato-

rów porządkowania lub przemiennych funkcji boolowskich). Dalsze rozważania będą prowadzone, przy założeniu że ustaliliśmy pewien sposób porządkowania punktów wyczytanych z obrazu dwuwymiarowego i mamy "jednowymiarowy" ciąg na wejściu przetwornika stosowego.

3. IDEA PRZETWARZANIA STOSOWEGO

Przetwarzanie stosowe jest rodzajem przetwarzania obrazów, zawierającym w sobie (jako szczególne przypadki) wszystkie rodzaje przetwarzania z operatorami porządkowymi oraz wiele innych rodzajów przetwarzania obrazów.

Pojęcie przetwarzanie stosowe zawiera w sobie szeroką klasę przetworników wykonanych za pomocą progowej dekompozycji (treshold docomposition) z uwzględnieniem właściwości układania w stos (stacking property). Konstrukcję takiego przetwornika prześledźmy za [1] na przykładzie filtru medianowego o aperturze szerokości 3, zrealizowanego techniką przetwarzania stosowego, z ciągiem o następujących parametrach: I = 9, M = 4, na wejściu, (rys. 3.1).



Rys. 3.1. Filtracja medianowa z aperturą o szerokości 3 wykonana za pomocą progowej dekompozycji

Fig. 3.1. Window width 3 median filtering by threshold decomposition

Właściwość progowej dekompozycji sprawia (co ilustruje rys. 3.1.), że przejście M-wartościowego ciągu przez filtr ze statystyką porządkową jest **równoważne** następującej procedurze:

1) Dekompozycji M-wartościowego ciągu wejściowego w zbiór M-1 ciągów binarnych, k-ty ciąg binarny, gdzie k jest liczbą całkowitą ze zbioru {1, 2,..., M-1}, jest otrzymywany przez progowanie ciągu wejściowego progiem o wartości k. Daje to wartość 1 dla elementu ciągu większego od lub równego k i 0 dla elementu ciągu mniejszego k. Zauważmy że sumowanie tych M-1 ciągów binarnych zawsze daje nam oryginalny ciąg wejściowy (rys. 3.1.).
2) Filtracji każdego ciągu binarnego niezależnie, za pomocą jego własnego filtru ze statystyką porządkową (np. binarnego filtru medianowego). Wszystkie te operacje mogą być wykonane równolegle. Również każdy z filtrów, dzięki ciągom binarnym na wejściu, ma bardzo prostą realizację. Każdy z nich po prostu dodaje bity fragmentu ciągu (fragmentu określonego przez aperturę) i porównuje wynik z zadaną liczbą całkowitą r: wyprowadzając zero, jeśli suma jest mniejsza od r, a jedynkę, jeśli suma jest większa lub równa r. Jeżeli szerokość apertury b jest równa 2r + 1, to filtr jest filtrem medianowym.
3) Dodaniu elementów ciągów pojawiających się jednocześnie na wyjściach filtrów binarnych (binarnych operatorów porządkowych).

Ten nowy sposób realizacji filtru medianowego oraz innych filtrów z operatorami porządkowania bardzo dobrze nadaje się do wykonania w technologii VLSI (Very Large Scale Intergration). Jest to najbardziej równoległy sposób przetwarzania informacji, a operacje progowania i binarne filtry porządkujące są łatwe do wykonania technologią VLSI. Osobnym problemem jest wykonanie części rekonstruującej, ponieważ musi ona dodać jednocześnie wyjścia wszystkich binarnych operatorów porządkowych. Trudności w realizacji zaczynają się dla więcej niż kilku ciągów binarnych. Na przykład, jeśli oryginalny ciąg wejściowy składa się z ośmiobitowej danej, to trzeba jednocześnie dodać 255 sygnałów binarnych w części rekonstruującej. Tak więc widać, że wykonanie tego rozwiązania w technologii VLSI nie bardzo jest możliwe, ponieważ sumator, który to realizuje, mógłby zająć na swoje potrzeby cały chip. Rozwiązanie tego problemu jest możliwe przy uwzględnieniu, że wyjścia binarnych operatorów ze statystykami porządkowymi posiadają właściwość układania w stos. Sprawia ona, że jeżeli binarne ciągi wyjściowe są "ułożone jeden nad drugim", w porządku zgodnym z poziomami progowania, to powstaje struktura (patrz rys. 3.1.), w której zawsze nad kolumną jedynek występuje kolumna zer. Poziom, na którym następuje w kolumnie zmiana z "1" na "0", jest poszukiwanym wyjściem przetwornika stosowego. Poziom ten znajdujemy dokonując binarnego przeszukiwania wyjść filtrów binarnych. W ten sposób możemy szybko określić wyjście przetwornika stosowego używając jednego bitu w celu detekcji zmiany z 0 na 1, startując od najwyższego poziomu progowania. Przeprowadzenie szczegółowej analizy realizacji sprzętowej takiego przetwornika stosowego w technologii VLSI nie jest przedmiotem tej pracy, jednak już zamieszczone powyżej podstawowe informacje pokazują, że jest ona prosta i zapewnia dużą szybkość działania.

Wymienione powyżej zalety sprawiły, że zadano sobie pytanie, jakie jeszcze inne przetworniki (oprócz przetworników ze statystyką porządkową, dla których właściwość

układania w stos została dowiedziona w [3] i [4], można zrealizować za pomocą progowej dekompozycji z wykorzystaniem właściwości układania w stos. W dalszej części pracy będzie pokazane, że przetworniki o wyżej wymienionych właściwościach otrzymuje się przez zastąpienie binarnego przetwornika porządkowego odpowiednią funkcją boolowską (np. na rys. 3.1. binarnego filtru medianowego odpowiednią funkcją boolowską). Ze względu na posiadaną przez wszystkie te przetworniki właściwość układania w stos oraz ze względu na jej wagę (bez niej nie byłaby możliwa ich realizacja w technologii VLSI) ten dział przetwarzania obrazów nazwano "przetwarzaniem stosowym" (lub filtracją stosową - Stack Filtration).

Funkcje boolowskie, którymi chcielibyśmy zastąpić B F M z rys. 3.1, muszą posiadać właściwość układania w stos. Zastanówmy się teraz, jakie funkcje boolowskie będą te właściwość posiadały. Aby to zrobić, konieczne jest podanie zapisu i definicji progowej dekompozycji oraz właściwości układania w stos.

Notacja:

- zapis wektorowy będzie używany do oznaczenia ciągów, tak więc wejściowy sygnał s o długości n będzie zapisany jako $\bar{s} = (s_1, s_2, ..., s_n)$. Jeśli ciąg $\bar{s}_n$ jest ciągiem binarnym, to piszemy $\bar{s}_{b,n}$ dla przejrzystości. Wtedy kiedy wartość n będzie jasno wynikała z kontekstu, pominiemy n w zapisie $\bar{s}_b$.
- Relacje pomiędzy ciągami. Niech $\bar{x}_{b,n} = (x_1, x_2, ..., x_n)$ i $\bar{y}_{b,n} = (y_1, y_2, ..., y_n)$ będą dwoma binarnymi ciągami o długości n. Mówimy, że $\bar{x}_{b,n} = \bar{y}_{b,n}$ wtedy i tylko, gdy $x_i = y_i$ dla każdego i. Mówimy, że $\bar{x}_{b,n} \le \bar{y}_{b,n}$, jeśli zachodzi $\{(x_i = 1) \Rightarrow (y_i = 1)\}$, czyli $x_i = 1$ implikuje $y_i = 1$ dla każdego i; piszemy $\bar{x}_{b,n} < \bar{y}_{b,n}$, jeżeli $\bar{x}_{b,n} \le \bar{y}_{b,n}$ i jednocześnie nie zachodzi $\bar{x}_{b,n} = \bar{y}_{b,n}$.
- Zapis dla funkcji logicznych. Binarna funkcja B o n wejściach będzie zapisywana Bn. Jeśli binarna funkcja logiczna zostanie zastosowana do wykonania operacji na argumencie $\bar{x}_{b,n}$, to wyjście zapisujemy jako Bn $(\bar{x}_{b,n})$ albo Bn $(x_1, x_2, ..., x_n)$. Przykładem ilustrującym ten zapis dla funkcji niech będzie filtracja medianowa o aperturze równej 3. Jeśli na wejściu mamy tylko sygnały binarne, to filtr medianowy staje się binarną funkcją, którą zapisujemy FM3. Jeśli ten filtr zastosować do ciągu $\bar{s}_b = (1, 0, 0)$, to wynikiem będzie zero, co zapisujemy FM3 (1, 0, 0) = FM3 $(\bar{s}_b)$ = 0. Podobnie FM3 (0, 1, 0) = 0 i FM3 (1, 1, 0) = 1. Uogólniając, wyjście FM3 jest równe 1, gdy co najmniej dwie pozycje argumentu zajmuje 1. Moglibyśmy przedstawić taki filtr binarny jako tablicę stanów, jednakże jeśli umiemy wyprowadzić regułę definiującą wyjście filtru, możemy przedstawić filtr bardziej zwięźle jako wyrażenie boolowskie. W takim wyrażeniu przedstawiamy OR przez dodawanie, AND przez mnożenie, a dopełnienie zmiennej X przez $\overline{X}$. Na przykład dla FM3 otrzymujemy zapis dla całego filtru:

$$FM3(x_1,x_2,x_3) = x_1x_2 + x_2x_3 + x_1x_3 \tag{3.1}$$

Filtr ten można wykonać przez umieszczenie binarnego operatora **B** na każdym poziomie progowania (rys. 3.2). Filtr nazywamy filtrem stosowym S_B, jeśli **B** ma właściwość układania w stos.

Za pomocą przedstawionego zapisu możemy teraz ściślej określić - zdefiniować progową dekompozycję i właściwość układania w stos.

Definicja 3.1

Progowa dekompozycja M-wartościowego ciągu $\bar{s}_N$ jest to zbiór M - 1 binarnych ciągów, zwanych ciągami sprogowanymi:

$$\vec{T1}_b, \vec{T2}_b, \ldots, \overrightarrow{T(M-1)}_b,$$

których elementy definiujemy jako

$$Ti_b(j) = \begin{cases} 1 & \text{jesli} \quad \bar{s}(j) \geq i \\ 0 & \qquad\quad \bar{s}(j) \leq i \end{cases} \tag{3.2}$$

□

Jako przykład pokazujący zbiór sygnałów sprogowanych lewą stronę rys. 3.1.:

	j = 1	2	3	4	5	6	7	8	9	
$\vec{T3}_b$ = (	0	0	0	0	1	1	0	0	0	) i =3
$\vec{T2}_b$ = (	0	0	0	1	1	1	0	1	1	) i = 2
$\vec{T1}_b$ = (	1	1	0	1	1	1	1	1	1	) i = 1

Zauważmy, że sygnały sprogowane zawsze posiadają właściwość układania w stos, którą definiujemy następująco:

Definicja 3.2

Uporządkowany zbiór L binarnych ciągów $\vec{S1}_b, \vec{S2}_b, \ldots \vec{SL}_b$, w którym ciągi mają długość n, posiada właściwość układania w stos, jeżeli:

$$\vec{S1}_b \geq \vec{S2}_b \geq \vec{S3}_b \geq \vec{SL}_b \,. \tag{3.3}$$

□

Możemy zatem zdefiniować właściwość układania w stos dla funkcji binarnych, tak jak zdefiniowaliśmy ją dla uporządkowanych zbiorów ciągów.

Definicja 3.3

O funkcji binarnej **Bn** z aperturą o szerokości **n** mówimy, że ma właściwość układania w stos, jeżeli:

$$\bar{x} \geq \bar{y} \Rightarrow Bn(\bar{x}) \geq Bn(\bar{x}). \tag{3.4}$$

□

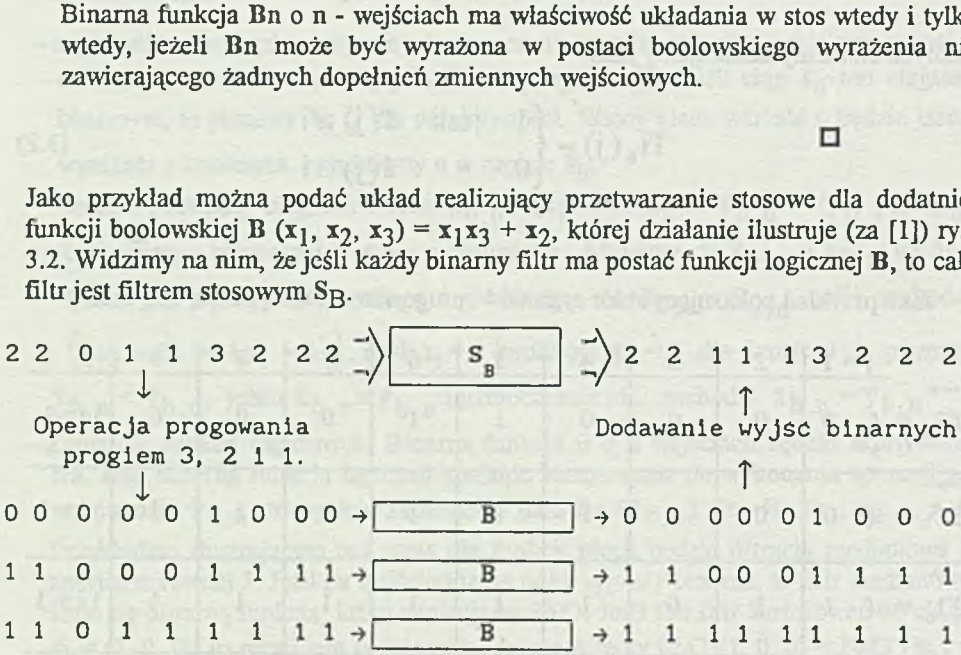
Następnie określimy warunek konieczny i wystarczający posiadania przez funkcję binarną właściwości układania w stos. Warunek ten został wyprowadzony przez E.N. Gilberta w [2].

Twierdzenie 3.1

Binarna funkcja **Bn** o **n** - wejściach ma właściwość układania w stos wtedy i tylko wtedy, jeżeli **Bn** może być wyrażona w postaci boolowskiego wyrażenia nie zawierającego żadnych dopełnień zmiennych wejściowych.

□

Jako przykład można podać układ realizujący przetwarzanie stosowe dla dodatniej funkcji boolowskiej **B** $(x_1, x_2, x_3) = x_1x_3 + x_2$, której działanie ilustruje (za [1]) rys. 3.2. Widzimy na nim, że jeśli każdy binarny filtr ma postać funkcji logicznej **B**, to cały filtr jest filtrem stosowym S_B.



Rys. 3.2. Filtr stosowy z aperturą o szerokości 3. Dodatnia funkcja boolowska realizującą powyższy filtr ma postać:

$$B(x_1, x_2, x_3) = x_1x_3 + x_2,$$

gdzie x_i, i = 1, 2, 3 są bitami pojawiającymi się w aperturze. Filtr ten zachowuje krawędzie, usuwa zakłócenia punktowe o wartości mniejszej od tła (22011) → (22111), ale pozostawia zakłócenia punktowe o wartości większej od tła (11322) → (11322).

Fig. 3.2. Window width 3 stack filtering based on positive Boolean function:

$$B(x_1, x_2, x_3) = x_1x_3 + x_2,$$

where x_i, i = 1, 2, 3 are bits appearing in the window. This filter preserves shapes, removes impulsive noise value less then background (22011)→(22111), but preserves single point noise value greater then background (11322)→(11322).

Analogicznie do filtru usuwającego zakłócenia o wartości mniejszej od tła istnieje filtr usuwający zakłócenia o wartości większej od tła. Otrzymujemy go przez wstawienie w miejsce funkcji logicznej B funkcji o postaci $B(x_1, x_2, x_3) = x_1x_2 + x_2x_3$.

Porównanie działania filtrów skonstruowanych za pomocą $B(x_1, x_2, x_3) = x_1x_3 + x_2$ i $B(x_1, x_2, x_3) = x_1x_3 + x_2x_3 + x_1x_2$ (filtracja medianowa) dla $A = \{(-1,0), (0,0), (1,0)\}$ pokazuje rysunek 3.3.

E.N. Gilbert nazywa funkcje spełniające warunek z twierdzenia 3.1. funkcjami frontalnymi (Frontal Functions) [2], gdzie indziej są one nazywane dodatnimi funkcjami boolowskimi (Positive Boolean Function) i to drugie określenie będzie używane w tej pracy.

Zauważmy, że warunek zapewniający właściwość układania w stos nie ogranicza zbyt mocno liczby dodatnich funkcji boolowskich, a ponieważ każda taka funkcja definiuje filtr stosowy (przez wprowadzenie jej do konstrukcji z rys. 3.2.), stąd możliwych jest wiele filtrów stosowych. W [2] podano, że jest dokładnie 20 takich filtrów dla apertury o szerokości 3, 7581 filtrów dla apertury o szerokości 5 i więcej niż 2^{35} dla apertury o szerokości 7. Dokładna liczba dodatnich funkcji boolowskich dla apertury o dowolnej szerokości nie jest obecnie znana [1].

Dodatnie funkcje boolowskie są podzbiorem funkcji unitarnych [5], [6]. Funkcje unitarne mają taką cechę, że można je wyrazić w Zminimalizowanej Postaci Sumacyjnej (ZPS). Tak więc każda dodatnia funkcja boolowska i wskutek tego każdy filtr stosowy może być wyrażony w ZPS. Wyrażenie dla FM3 w ZPS podano w (3.1).

Dla niektórych dodatnich funkcji boolowskich wygodnie jest stosować jeszcze inną postać. Filtry binarne wykorzystane do realizacji filtrów ze statystyką porządkową za pomocą progowej dekompozycji są zasadniczo operatorami dodawania i progowania. Te binarne operatory porządkujące są szczególnymi przypadkami funkcji progujących [5], [6]. Podajmy zatem definicję funkcji progujących.

Definicja 3.4

n-wejściowa funkcja progująca $T(x_1,x_2,...,x_n)$ z wagami $a_1, a_2,...,a_n$ i progiem c jest zdefiniowana jako:

$$T(x_1,x_2,..,x_n) = 1 \quad \Leftrightarrow \quad \sum_{i=1}^{n} a_i x_i \geq c \tag{3.5}$$

$a_i, c \in R$, R - zbiór liczb rzeczywistych.

□

Następnie pokażemy, jakie funkcje progujące są dodatnimi funkcjami boolowskimi. Jeżeli $\bar{x}_b = (x_1, x_2, ..., x_n)$ i $\bar{a} = (a_1, a_2, ..., a_n)$, to funkcję progującą możemy przedstawić jako:

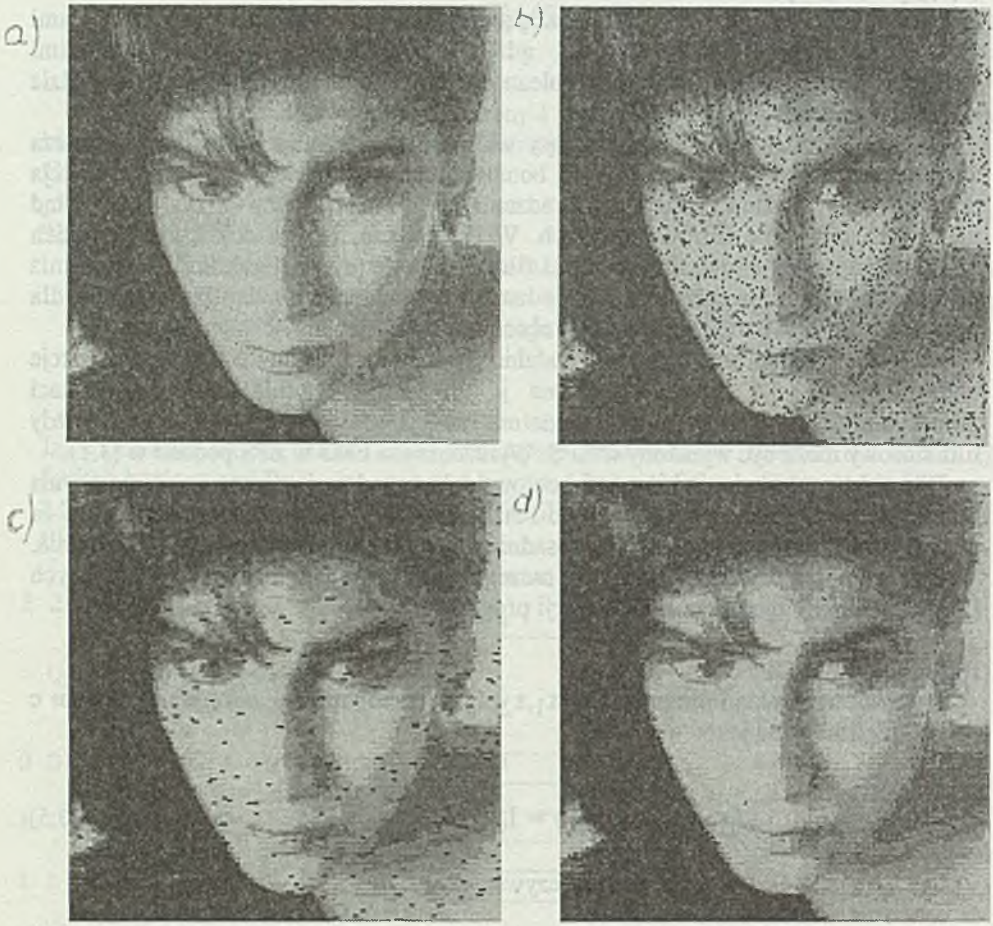
$$T(\bar{x}_b) = 1 \Leftrightarrow \bar{a} \cdot \bar{x}_b \geq c. \tag{3.6}$$

$$(\bar{x}_b \leq \bar{y}_b \quad \text{i} \quad a_1, a_2, ..., a_n \geq 0) \Rightarrow (\bar{a} \cdot \bar{x}_b \leq \bar{a} \cdot \bar{y}_b).$$

Jeżeli

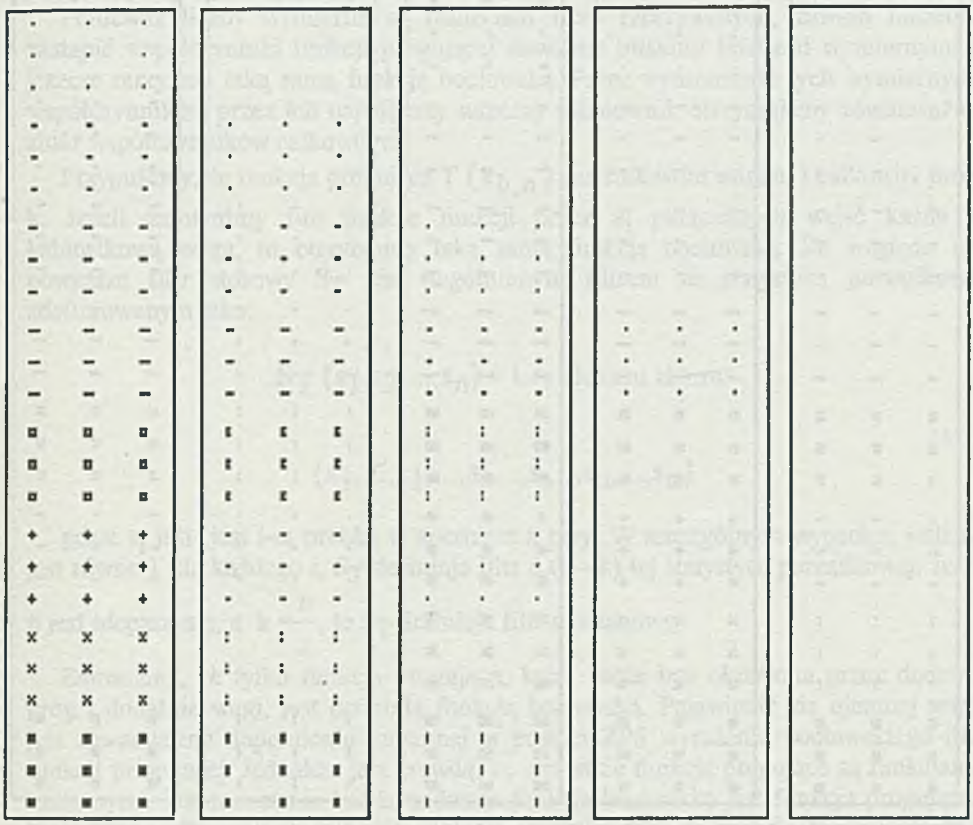
$$\left(\bar{a}\cdot\bar{x}_b > c\right) \Rightarrow \left(\bar{a}\cdot\bar{y}_b > c\right), \text{ to}$$

$$\left(T(\bar{x}_b) = 1\right) \Rightarrow \left(T(\bar{y}_b) = 1\right).$$



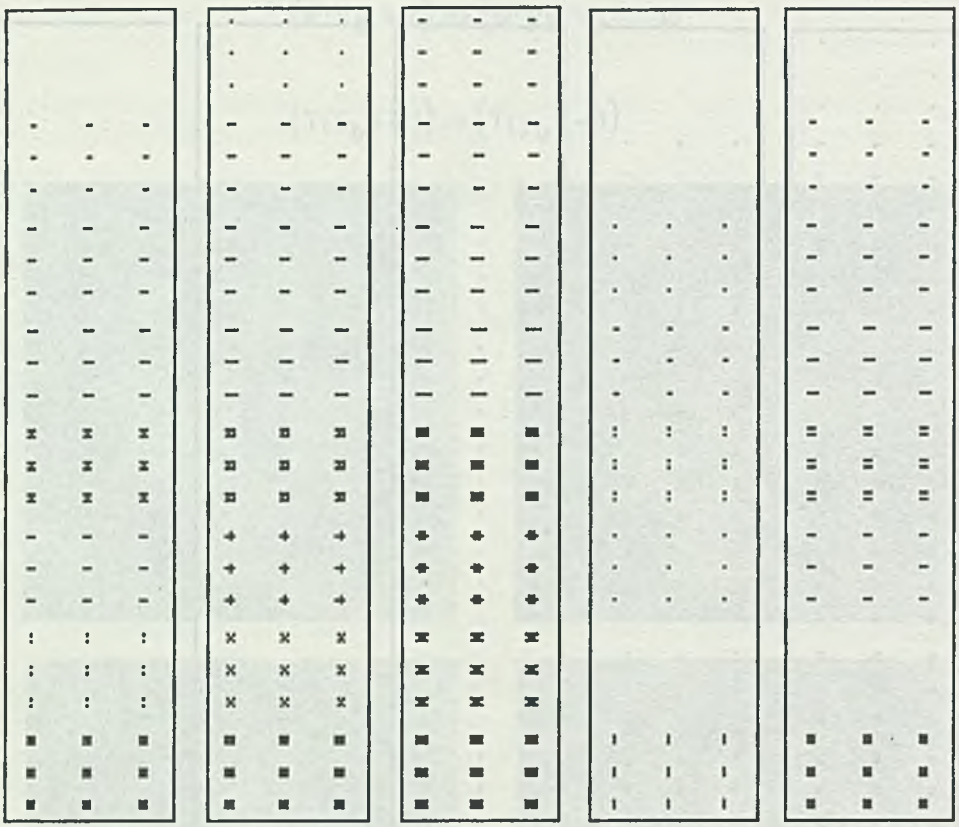
Rys. 3.3. Porównanie filtracji medianowej i szczególnego przypadku filtracji stosowej ($A = \{(-1,0), (0,0), (1,0)\}$). a) obraz oryginalny, b) obraz oryginalny plus zakłócenie stochastyczne punktowe, c) obraz b) po dwukrotnej filtracji filtrem stosowym dla $B = x_1x_2 + +x_1x_2 + x_2x_3$ (filtr medianowy), d) obraz b) po dwukrotnej filtracji dla $B = x_2 + x_1x_3$

Fig. 3.3. Comparing median filter and stack filter ($A = \{(-1,0), (0,0), (1,0)\}$). a) original image, b) impulsive noise corrupted image, c) image b) after two passings by stack filter with $B = x_1x_2 + x_1x_2 + x_2x_3$ (median filter), d) image b) after two passings by stack filter with $B = x_2 + x_1x_3$



Rys. 3.4. Wielokrotne przejście obrazu (różne rozkłady czarnych punktów na białym tle) przez filtr stosowy z $B = x_2 + x_1x_3$ przy aperturze: $A = \{(-1,0), (0,0), (1,0)\}$. a) obraz oryginalny, b) obraz a) po jednokrotnym przejściu przez filtr, c) obraz a) po dwukrotnym przejściu przez filtr, d) obraz a) po trzykrotnym przejściu przez filtr, e) obraz a) po czterokrotnym przejściu przez filtr

Fig. 3.4. Several passing image (different distribution of black points on white background) by stack filter $B = x_2 + x_1x_3$ at window: $A = \{(-1,0), (0,0), (1,0)\}$. a) original image, b) image a) after one passing by filter, c) image a) after two passings by filter, d) image a) after three passings by filter, e) image a) after four passings by filter



Rys. 3.5. Jednokrotne przejście obrazu oryginalnego z rys. 3.4. przez różne filtry stosowe przy aperturze:

$$A = \{(-1,0), (0,0), (1,0)\}$$

a) $B = x_1x_2 + x_1x_3 + x_2x_3$,
b) $B = x_1x_2 + x_2x_3$,
c) $B = x_1x_2x_3$,
d) $B = x_1 + x_2 + x_3$,
e) $B = x_1 + x_2x_3$.

Fig. 3.5. One passing original image with fig. 3.4. by different stack filters using window:

$$A = \{(-1,0), (0,0), (1,0)\}$$

a) $B = x_1x_2 + x_1x_3 + x_2x_3$,
b) $B = x_1x_2 + x_2x_3$,
c) $B = x_1x_2x_3$,
d) $B = x_1 + x_2 + x_3$,
e) $B = x_1 + x_2x_3$.

Zatem jeśli funkcja progująca będzie określona przy nieujemnych wagach i nieujemnym progu, to jest dodatnia funkcja boolowska.

Ponieważ liczby wymierne są obcięciem liczb rzeczywistych, zawsze możemy zastąpić współczynniki funkcji progującej dowolnie bliskimi liczbami wymiernymi i jeszcze otrzymać taką samą funkcję boolowską. Przez wymnożenie tych wymiernych współczynników przez ich największy wspólny mianownik otrzymujemy równoważny zbiór współczynników całkowitych.

Przypuśćmy, że funkcja progująca $T(\bar{x}_{b,n})$ ma całkowite wagi a_i i całkowity próg k. Jeżeli zamienimy i-te wejście funkcji przez a_i połączonych wejść każde z jednostkową wagą, to otrzymamy taką samą funkcję boolowską. Ze względu na powyższe filtr stosowy S_T jest uogólnionym filtrem ze statystyką porządkową zdefiniowanym jako:

$$S_T(s_1,s_2,\ldots,s_n) = \text{k-ty element zbioru:}$$

$$\{s_1,\ldots,s_1,\ldots,s_i,\ldots,s_i,\ldots,s_n,\ldots,s_n\} \tag{3.7}$$

gdzie s_i jest jest i-tą próbką w aperturze a razy. W szczególnym wypadku, jeśli a_i jest równe 1 dla każdego i, S_T definiuje filtr o (n - k)-tej statystyce porządkowej. Jeśli n jest nieparzyste, a $k = \frac{n}{2}$, to S_T definiuje filtr medianowy.

Zauważmy, że tylko funkcja progująca, która może być określona przez dodatni próg i dodatnie wagi, jest dodatnią funkcją boolowską. Pojawienie się ujemnej wagi jest równoważne dopełnieniu zmiennej w postaci ZPS wyrażenia boolowskiego dla funkcji progującej. Jednakże jest prawdą, że wszystkie funkcje progujące są funkcjami unitarnymi. Natomiast nie każda dodatnia funkcja boolowska jest funkcja progującą. Wszystkie dodatnie funkcje boolowskie trzech zmiennych są funkcjami progującymi, jednakże z 7581 dodatnich funkcji boolowskich pięciu zmiennych tylko 3287 to funkcje progujące [6]. Oznacza to, że jest wiele filtrów stosowych, których nie da się wyrazić jako "proste" funkcje zmiennych wejściowych.

Mając już jakąś dodatnią funkcję boolowską, możemy poszukiwać innych dodatnich funkcji boolowskich. Są dwie podstawowe operacje, które przekształcają jedną DFB w inną DFB. Pierwsza polega na dowolnym przestawieniu zmiennych wejściowych DFB; na przykład możemy odwrócić kolejność zmiennych wejściowych. Druga operacja polega na zanegowaniu wszystkich wejść dodatniej funkcji boolowskiej i zanegowaniu wyjścia całej funkcji. Otrzymujemy wtedy nową funkcję, którą nazywamy funkcją dualną do funkcji **B** i zapisujemy B^d. Ta operacja jest równoważna wstawieniu w miejsce **AND OR**, a w miejsce **OR AND** w wyrażeniu **AND-OR**. Wskutek tego jeśli **B** jest dodatnie, to B^d jest dodatnie również. W szczególnym wypadku jeśli **B** jest dodatnią funkcją progującą, to B^d jest nią również. Oczywiście odwrócenie kolejności zmiennych odwraca sens "czasu" dla filtru stosowego S_B. Zamiana **B** w progowej dekompozycji przez jego dualną funkcję B^d odwróci wartości poziomów dla również odwróconych wartości wejść. Na przykład jeśli pierwotny filtr daje na wyjściu {Z(t)} dla wejścia {S(t)}, to nowy filtr da wyjście {M-1Z(t)} dla wejścia {M-1-S(t)}.

4. METODA PROJEKTOWANIA PRZETWORNIKÓW STOSOWYCH REALIZUJĄCYCH STATYSTYKI PORZĄDKOWE

Zastanówmy się teraz, jaką postać będzie miała funkcja boolowska realizująca przetwarzanie ze statystyką porządkową dla apertury o dowolnej szerokości (dowolna liczba zmiennych w funkcji) i kwantyla dowolnego rzędu.

Najpierw wyprowadźmy funkcję boolowską dla trzech zmiennych i kwantyla k = 0,5 (filtracja medianowa).

Tablica prawdy (wartości funkcji) jest przedstawiona na rys. 4.1.

x_1	x_2	x_3	$B(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Rys. 4.1. Tablica prawdy $B_{FM}(x_1, x_2, x_3)$
Fig. 4.1. True table $B_{FM}(x_1, x_2, x_3)$

Do otrzymania funkcji posłużymy się tablicą Karnaugha, która dla tej funkcji ma postać przedstawiona na rys. 4.2.

Postać sumacyjna

$x_1 \backslash x_2x_3$	0 0	0 1	1 1	1 0
0	0	0	1	0
1	0	1	1	1

$B(x_1, x_2, x_3) = x_1x_3 + x_1x_2 + x_2x_3$

Postać iloczynowa

$x_1 \backslash x_2x_3$	0 0	0 1	1 1	1 0
0	0	0	1	0
1	0	1	1	1

$B(x_1, x_2, x_3) = (x_1+x_3)(x_1+x_2)(x_2+x_3)$

Rys. 4.2. Tablice Karnaugha dla $B_{FM}(x_1, x_2, x_3)$
Fig. 4.2. Karnaugh's table for $B_{FM}(x_1, x_2, x_3)$

Przy czym oczywiście obie postacie funkcji realizują to samo działanie.

Wyprowadźmy teraz tę samą funkcję dla pięciu zmiennych. Wiemy, że wyjście funkcji boolowskiej ma wynosić 1, gdy co najmniej trzy jej argumenty są równe jeden. Zatem tablica Karnaugha będzie miała postać jak na rys. 4.3 i rys. 4.4.

Postać iloczynowa

$x_1x_2x_3$ \ x_4x_5	0 0	0 1	1 1	1 0
0 0 0	0	0	0	0
0 0 1	0	0	1	0
0 1 1	0	1	1	1
0 1 0	0	0	1	0
0 1 0	0	1	1	1
0 1 1	1	1	1	1
0 0 1	0	1	1	1
0 0 0	0	0	1	0

$$B(x_1,x_2,x_3,x_4,x_5) =$$
$$(x_1+x_4+x_5)(x_1+x_1+x_4)(x_1+x_2+x_3)$$
$$(x_2+x_3+x_5)(x_2+x_3+x_4)(x_1+x_2+x_5)$$
$$(x_2+x_4+x_5)(x_1+x_3+x_4)(x_1+x_3+x_5)$$
$$(\bar{x}_1+x_3+x_4+x_5)(x_1+x_3+x_4+x_5) =$$
$$=(x_1+x_4+x_5)(x_1+x_1+x_4)(x_1+x_2+x_3)$$
$$(x_2+x_3+x_5)(x_2+x_3+x_4)(x_1+x_2+x_5)$$
$$(x_2+x_4+x_5)(x_1+x_3+x_4)(x_1+x_3+x_5)$$
$$(x_3+x_4+x_5)$$

Rys. 4.3. Tablica Karnaugha dla $B_{FM}(x_1, x_2, x_3, x_4, x_5)$
Fig. 4.3. Karnaugh's for B $B_{FM}(x_1, x_2, x_3, x_4, x_5)$

$x_1x_2x_3$ \ x_4x_5	0 0	0 1	1 1	1 0
0 0 0	0	0	0	0
0 0 1	0	0	1	0
0 1 1	0	1	1	1
0 1 0	0	0	1	0
0 1 0	0	1	1	1
0 1 1	1	1	1	1
0 0 1	0	1	1	1
0 0 0	0	0	1	0

Postać sumacyjna

$$B(x_1,x_2,x_3,x_4,x_5) =$$
$$x_1x_2x_5+x_1x_2x_4+x_1x_3x_5+$$
$$x_1x_3x_4+x_1x_2x_3+x_1x_4x_5+$$
$$\bar{x}_1x_3x_4x_5+x_1x_3x_4x_5+$$
$$\bar{x}_1x_2x_4x_5+x_1x_2x_4x_5+$$
$$\bar{x}_1x_2x_3x_5+x_1x_2x_3x_5+$$
$$\bar{x}_1x_2x_3x_4+x_1x_2x_3x_4 =$$
$$= x_1x_2x_5+x_1x_2x_4+x_1x_3x_5+$$
$$x_1x_3x_4+x_1x_2x_3+x_1x_4x_5+$$
$$x_3x_4+_5x_2x_4x_5+x_2x_3x_5+x_2x_3x_4$$

Rys. 4.4. Tablica Karnaugha dla $B_{FM}(x_1, x_2, x_3, x_4, x_5)$
Fig. 4.4. Karnaugh's table for B $B_{FM}(x_1, x_2, x_3, x_4, x_5)$

Przedstawiony powyżej tok postępowania można zastąpić prostszym algorytmem pozwalającym otrzymać funkcję boolowską realizującą przetwarzanie ze statystyką porządkową dla apertury o dowolnej szerokości (dowolna liczba zmiennych w funkcji) i kwantyla dowolnego rzędu.

Twierdzenie 4.1.

Jeżeli mamy aperturę o szerokości 2N+1 i chcemy otrzymać funkcję boolowską wybierającą z binarnego ciągu o długości 2N+1 kwantyl rzędu k, to postępujemy w następujący sposób:

a) Wybieramy wszystkie możliwe kombinacje l-elementowe (l = 2N · K+l) ze zbioru 2N + 1, elementowego {1,2,..., 2N+1}. Otrzymujemy l-elementowe podzbiory liczb. Będzie ich $\binom{2N+1}{l} = \frac{(2N+1)!}{l!(2N+1-l)!}$.

b) Podzbiory te są zbiorami indeksów zmiennych, z których należy utworzyć iloczyn (dla postaci sumacyjnej B) lub sumę (dla postaci iloczynowej B).

c) Tak powstałe iloczyny dodajemy do siebie (sumy wymnażamy przez siebie) i otrzymujemy funkcje boolowską. Oczywiście przez dodawanie i wymnażanie rozumiemy logiczne operacje OR i AND.

□

Funkcja w obu postaciach: sumacyjnej i iloczynowej wykonuje to samo działanie. Dla mediany liczbę składników wyrażenia w zależności od szerokości apertury przedstawia tabela 4.1.:

Tabela 4.1

Liczba punktów w aperturze	Liczba składników wyrażenia B
3	$\binom{3}{2} = 3$
5	$\binom{5}{3} = 10$
7	$\binom{7}{4} = 35$
9	$\binom{9}{5} = 126$

Jak z tego widać, wraz ze wzrostem szerokości apertury bardzo szybko rośnie liczba składników wyrażenia boolowskiego. W realizacji programowej zastosowano dwa warianty funkcji logicznej wybierającej medianę z ciągu: sumowanie jedynek i progowanie przy progu N+1 oraz wyrażenie boolowskie. To drugie rozwiązanie powoduje skrócenie czasu filtracji. Jednak ze względu na szybki wzrost liczby składników w wyrażeniu zysk czasu maleje ze wzrostem szerokości apertury. Dla przykładowego obrazu przedstawia to tabela 4.2.:

Tabela 4.2

Szerokość apertury	Σ jedynek ≥ N+1 (t_1)	wyrażenia Boolowskie (t_2)	$\frac{t_2}{t_1}$
3	1 min. 49 sek.	21 sek.	0.19
5	1 min. 56 sek.	35 sek.	0.30
7	2 min. 04 sek.	1 min. 07 sek.	0.54

Przedstawmy teraz dowód twierdzenia 4.1.:

Dowód twierdzenia 4.1.

Założenie 1.: funkcja ma mieć wartość 1 ⇔ co najmniej 1 zmiennych z 2N+1 zmiennych jest równych 1.

Jeżeli co najmniej 1 zmiennych z 2N+1 zmiennych jest równych 1, to co najmniej jeden z iloczynów (w postaci sumacyjnej) składa się z tych 1 zmiennych, które są równe 1 (ponieważ wybraliśmy wszystkie możliwe iloczyny 1 zmiennych z 2N+1 zmiennych). Ale wtedy cała funkcja jest równa 1.

Założenie 2.: funkcja ma mieć wartość 0 ⇔ więcej niż (2N+1-1) zmiennych z 2N+1 zmiennych jest równych 0.

Jeżeli więcej niż (2N+1-1) zmiennych jest równych 0, to w każdym iloczynie (w postaci sumacyjnej) jest co najmniej jedna zmienna równa 0, a zatem nie ma ani jednego iloczynu równego 1 i cała funkcja przyjmuje wartość 0.

Otrzymanie wyjścia funkcji równego 1 przy założeniu 1 i otrzymanie wyjścia funkcji równego 0 przy założeniu 2. jest równoważne spełnieniu wymagań nakładanych na funkcję.

Analogiczne rozumowanie można przeprowadzić dla zminimalizowanej postaci iloczynowej (ZPI) funkcji boolowskiej.

□

5. ZAKOŃCZENIE

W pracy zostało wprowadzone pojęcie przetwarzanie stosowe - filtracja stosowa (stack filtration).

Jeżeli zauważymy, że wszystkie operacje wykonywane w trakcie przejścia obrazu przez filtr stosowy sprowadzają się do:

a) progowania wielowartościowego sygnału,

b) obliczania wartości funkcji boolowskiej,

c) detekcji najwyższego poziomu, dla którego wartość funkcji boolowskiej wynosi 1,

to widzimy, że z powodu prostoty wykonania tych operacji w technologii VLSI wszystkie przetworniki stosowe mogą być wykonane w ten sposób. Umożliwia to wzrost szybkości działania pozwalający na przetwarzanie obrazów w czasie rzeczywistym.

Również realizacja programowa, dzięki dekompozycji problemu złożonego na szereg prostszych, staje się łatwiejsza do wykonania oraz powoduje skrócenia czasu przetwarzania.

Natomiast z powodu warunku, jaki musi spełniać funkcja boolowska, niezbyt mocno ograniczającego liczbę dostępnych funkcji, pojawia się ogromna liczba możliwych do zrealizowania filtrów. Już dla trzech punktów w aperturze (funkcja boolowska trzech zmiennych) otrzymujemy bardzo użyteczne filtry. Przykład z rys. 3.3. ilustruje przydatność zastosowania funkcji boolowskiej usuwającej zakłócenia punktowe o wartości mniejszej od tła, gdy na obrazie pojawiają się właśnie takie zakłócenia. Filtracja medianowa usuwająca zakłócenia punktowe zarówno o wartości większej od tła, jak i mniejszej od tła daje tu wyraźnie gorsze wyniki. Z kolei rysunki: 3.4. i 3.5. pokazują przejście elementów obrazu o różnych kształtach wielokrotnie przez jeden filtr lub jednokrotnie przez różne filtry. Szczegółowa analiza otrzymanych rezultatów nie była tutaj przeprowadzona, a jedynie chodziło o pokazanie, jak różne wyniki można otrzymać dla różnych funkcji boolowskich.

Widząc, jak wielką różnorodność rodzajów filtracji oferuje filtracja stosowa przy większych liczbach punktów w aperturze, można zadać sobie pytanie: jak wybrać ten optymalny filtr dla konkretnego zakłócenia, dla danego obrazu czy dla konkretnych warunków zakłócenia obrazu. Ogromną liczbę możliwości powiększa jeszcze fakt, że na różnych poziomach progowania mogą być różne filtry, jeżeli tylko posiadają one względem siebie właściwość układania w stos. Na te niezwykle interesujące pytania można znaleźć już częściowo odpowiedź np. w [7]).

LITERATURA

[1] Coyle Edward, Gallagher Neal, Wendt Peter: "Stack Filters" IEEE Transactions On Acoustics, Speech And Signal Processing, vol. ASS-34, pp. 898-911, August 1986.

[2] Gilbert E.N.: Lattice-teoretic properties of frontal switching functuions, J. Math. Phys., vol. 33, pp. 57-67, April 1954.

[3] Coyle Edward, Lin Jean-Hsang, Sellke Thomas: Adaptive stack filtering under the mean absolute error criterion, IEEE Transactions On Acoustics, Speech And Signal Processing, vol 38, pp. 938-953, June 1990.

[4] "Threshold decomposition of multidimensional ranked-oreder operations:, IEEE Trans. Circuits Syst., vol. CAS-32, pp. 445-450, May 1985.
[5] Gallagher N. C. Nodes T.A.: Theoretical results on the properties of median type filters, School Elec. Eng., Purdue Univ., West Lafayette, IN, Tech, Rep, EE 82-32, September 1982.
[6] Muroga S.: Threshold logic and Its Applications. New York: Wiley, 1971.
[7] Coyle Edward, Lin Jean-Hsang: Stack Filters and the Mean Absolute Error Criterion, IEEE Transactions On Acoustics, Speech And Signal Processing, vol. 36, No 8, Agust 1988.

Recenzent: Doc. dr inż. Bogdan **Wolczak** Pol. Szczecińska

Wpłynęło do Redakcji 31.01. 1992 r.

Abstract

In this work basic properties of stack filtration are presented. The basic terms of image processing such as 1D image, 2D image and appropriate apertures are defined. Afterwards, the definition of threshold decomposition, stacking property and some other definitions and thesis concerning stack filtration are given as well as examples of stack fillration of 16 - valued 2D images.

The idea of stack filration is to decompose multivalued input image into binary ones, then to process binary the images and finaly to create an output multivalued image as a result of the processed binary image summation.

The work illustrates the existence of a wide varietyt of stack filters, since the only factor limiting their number is necessity of using a Boolean function which has the stacking property. Such property is characteristic for positive Boolean functions presented in the paper.

The threshold decomposition mentioned above enables implementation of already known processing algorithms, e.g. median filtration, in completely different way then they been hitherto realized. Median filtration made by the means of stack filtration for multivalued image is a composition of binary median filtrations applied to binary images got in the result of multivalued image decomposition. The output of the binary median filtration is calculated by adding particular "1" numbers appearing in aperture and afterwards compared to the threshold. This causes that ordering operation found in conventional approach to median filtration can be skipped. As a result, a median filter based on the stack processing method contains only summators and thresholding elements, thus providing feasibility of constructing it using the VLSI technology.

The work presents also a desing method for Boolean function proceeding with image processing in rank oreder for an aperture with arbitrary width and order of quantile. Subsequently some examples of processing repeatidly differently shaped elements of the image by a sole filter and processing these elements by various filters are shown.