Mariusz KAPRUZIAK
ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY

Zrandomizowana transformata Hough'a (RHT) w strukturze FPGA

Dr inż. Mariusz KAPRUZIAK

Obronił prace doktorska na Politechnice Szczecińskiej w 2006r na temat "Opracowanie koncepcji i realizacja dedykowanego procesora radia programowalnego". Obecnie prowadzi badania dotyczące systemów rekonfigurowanych oraz widzenia maszynowego na Politechnice Szczecińskiej.

e-mail: mkapruziak@wi.ps.pl

Streszczenie

W artykule przedstawiona jest modyfikacja algorytmu RHT w celu efektywnego wykorzystania jej w strukturze FPGA. Głównym problemem implementacji RHT w FPGA jest duża i dynamicznie zorganizowana pamięć parametrów. Autor proponuje alternatywne rozwiązanie organizacji tej pamięci. Postuluje się, że pamięć parametrów i punktów aktywnych mogłaby być wspólna i niezorganizowana. Zadanie organizacji można przerzucić na proces losowy. Tak zmodyfikowany algorytm można skutecznie zaimplementować w układzie FPGA.

Słowa kluczowe: RHT, FPGA, Randomized Hough Transform.

Randomized Hough Transform in FPGA

Abstract

The paper presents modification of a randomized Hough transform (RHT) for efficient implementation in FPGA. The RHT has one significant advantage in comparison with the Hough Transform (HT) - much shorter processing time. It results from the fact that two points constitute together precisely one line (1), not the whole family of them (HT). Therefore only one accumulation has to be done in the parameter space for a single pair of points. The main problem of RHT implementation is a large and dynamic memory structure required for the parameter space. The author proposes an alternative solution. It is suggested that the parameters and active points memory could be kept in the same unorganized memory unit. Organization of the parameter space might be ensured afterwards by a random process, the same used for selecting two points to be put together into a single line. The problem of voting for "non-existing" lines (Fig. 1), common for algorithms in [1] and [3], is solved by connecting lines together (Fig. 2). The algorithm for such a transform has been proposed (Fig. 3) and implemented in FPGA (Figs. 5, 6, 7). It has resulted in a small and compact memory required in case of implementing the RHT. Moreover, the memory size is treated here as a parameter for implementation, allowing to trade the range of lines to be processed simultaneously and the efficiency of memory utilization. It works well even with a little memory supporting only tens or hundreds lines.

Keywords: RHT, FPGA, Randomized Hough Transform.

1. Wstęp

Rozpoznawanie obiektów w obrazie można w niektórych warunkach znacznie uprościć wykorzystując transformatę Hough [1]. Dotyczy to zarówno prostych obiektów, jak na przykład linia jak również bardziej złożonych jak litera czy znak firmowy. Jedynym praktycznym ograniczeniem jest liczba możliwych parametrów wzorcowego obiektu, która nie może być zbyt duża ze względu na konieczność zbudowania i operowania na wielowymiarowej przestrzeni parametrów.

Transformata Hough ma kilka wad: <1> wymaga dużej ilości pamięci do reprezentacji przestrzeni parametrów, <2> dla każdego punktu obrazu wymaga wielu wpisów do przestrzeni parametrów, <3> proces znajdowania punktów „aktywnych" w obrazie wejściowym dodaje się do złożoności obliczeniowej lub powiększa rozmiar pamięci do reprezentacji tablicy tych punktów. Zaproponowano kilka modyfikacji redukujących koszt tych problemów [1, 2, 3]. Ten artykuł dotyczy implementacji jednej z nich, zwanej zrandomizowaną transformatą Hough (RHT, ang. *Randomized Hough Transform*) [1]. W celu zachowania przejrzystości rozważań w artykule przedstawia się wersję tej transformaty do znajdowania linii prostych w obrazie wejściowym.

Wykorzystując zrandomizowaną transformatę Hough głównie próbuje się zredukować złożoność problemu <2>. Rozwiązanie można oprzeć na obserwacji, że dobierając punkty w pary na podstawie układu równań (wzór (1)) można określić tylko jeden punkt w przestrzeni parametrów [1]. Problemem takiego rozwiązania jest fakt, że dla zbioru N aktywnych punktów istnieje aż $N\times(N-1)/2$ par. Zamiast analizować wszystkie możliwości, pary punktów wybierane są losowo aż do momentu, gdy wystarczająco wyraźnie ujawnią się maksyma w przestrzeni parametrów.

$$\begin{cases} y_0 = ax_0 + b \\ y_1 = ax_1 + b \end{cases}; \tag{1}$$

gdzie:

x_0, y_0, x_1, y_1 - współrzędne pary punktów,

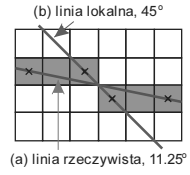
a, b - parametry prostej na której leżą oba punkty.

Pozostałe dwa problemy rozwiązuje się w artykule w zmodyfikowany sposób. Problem <1> w oryginalnej pracy [1] rozwiązany został za pomocą listy dynamicznej, który autor uznał jako mało efektywny przy implementacji w FPGA. Problem <3> natomiast nie został tam podjęty. Autor w modyfikacji algorytmu RHT postuluje, że mogłaby istnieć jedna globalna tablica punktów aktywnych i punktów reprezentujących figury z przestrzeni optymalizacji. Byłoby możliwe zestawianie wpisów takiej tablicy ze sobą, tworząc nowe figury. Możliwe zatem byłoby głosowanie na prostą nie tylko przez dwa punkty przez która ta prosta przechodzi ale także dwie proste mogą głosować na nową prostą.

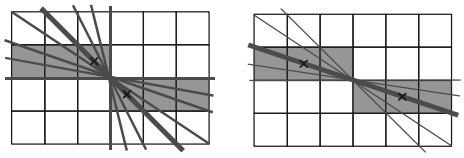
Rozwiązanie takie ma szczególne znaczenie przy implementacji strukturalnej w FPGA. Kłopotliwa byłaby tu przede wszystkim duża pamięć wymagająca sortowania i wstawiania elementów w środek. Dodatkowo niewielką pamięć wspólną dla punktów i linii można zorganizować w ten sposób, aby umożliwić wiele odczytów zapisów do niej jednocześnie. Umożliwi to zrównoleglenie procesu łączenia linii.

2. Modyfikacja algorytmu RHT

Głosowanie prostych na nową prostą wymaga dodatkowego komentarza. W świetle wzoru (1) wydaje się, że dwa punkty w sposób jednoznaczny tworzą nową prostą. Tak jednak nie jest. Związane jest to z niezerowym rozmiarem piksela. Na rysunku 1 pokazano przykład linii dla której określono jej nachylenie dla pary pikseli leżących w jej środku. Wynikowy kąt nachylenia okazuje się różny od kąta rzeczywistego ((a) 11.25^o w stosunku do (b) 45^o).



Rys. 1. Błąd określenia kąta linii dla punktów środkowych
Fig. 1. Error of line slope estimation for middle-points



Rys. 2. Rodzina możliwych prostych dla pary punktów. Czym punkty bardziej odległe tym węższy zakres kątów.
Fig. 2. Family of lines for a pair of points. The farther the points, the narrower the family

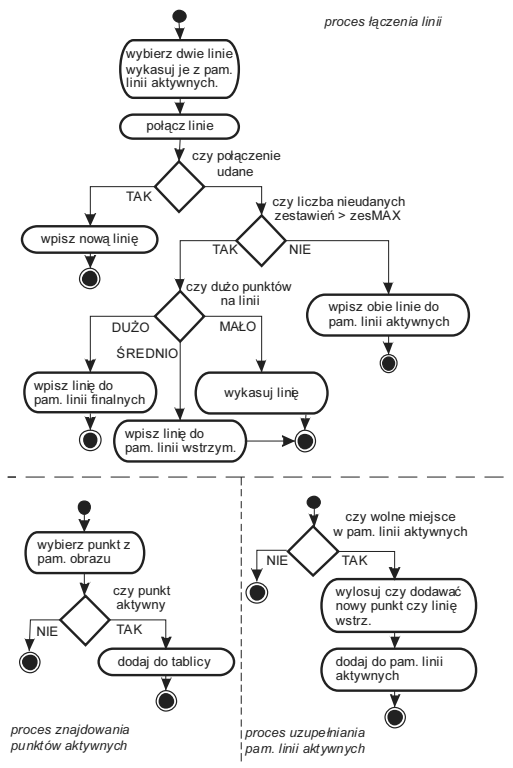
Czym dalsze punkty zostaną wybrane tym lepiej zostanie oszacowany prawdziwy kąt (rys. 2). W pracach [1, 3] pojawia się w ten sposób wiele głosów na proste nie występujące w obrazie. Propozycja autora aby zestawiać proste z innymi prostymi ma na celu tworzenie nowych bardziej ogólnych prostych z lepiej określonymi parametrami (jak nachylenie oraz przesunięcie).

Konsekwentnie autor próbuje pójść dalej tym torem, traktując wszystkie punkty tak jakby były bardzo krótkimi prostymi. Wtedy znika konieczność rozróżniania pomiędzy przestrzenią parametrów i tablicą punktów aktywnych. Wszystko przechowywane może być we wspólnej pamięci.

Powstała w ten sposób wspólna pamięć nie musi być w szczególny sposób uporządkowana (na przykład posortowana, jak przestrzeń parametrów w pracy [1]). Linie są z niej wybierane losowo i poddawane dalszej analizie. Pamięć ta też nie musi być na tyle duża aby pomieścić wszystkie punkty aktywne i możliwe linie. W miarę pracy algorytmu nowe punkty oraz nowe analizowane linie mogą być dodawane jak również są na bieżąco usuwane. W ten sposób wielkość pamięci staje się parametrem rozwiązania i może nie być duża.

3. Opis algorytmu

Algorytm ostatecznie opiera się na kilku prostych warunkach (rys. 3). Linie wybierane są parami losowo z pamięci. Każda para linii może łączyć się ze sobą tworząc nową lub nie. Jeśli się łączą w jedną to linie tworzące są wykasowywane, jeśli nie to pozostają w pamięci. Gdy linia przez większą liczbę iteracji nie łączy się z żadną jest kasowana z pamięci.

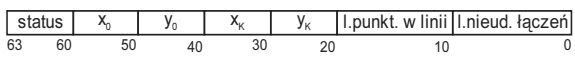


Rys. 3. Algorytm zmodyfikowanej RHT
Fig. 3. Algorithm of modified RHT

Kasowanie linii nie łączących się z innymi może także wynikać z chwilowo niekorzystnego doboru linii uwzględnionych do zestawiania. Nie zawsze kasowanie takie jest korzystne. Wprowadzono dlatego dwie dodatkowe pamięci: linii „finalnych" oraz linii „wstrzymanych". Pamięć linii finalnych to pamięć linii ostatecznie uznanych za skończone w przetwarzaniu i zaakceptowane jako wynik algorytmu (linie z dużą liczbą punktów leżących na nich). Pamięć linii wstrzymanych to natomiast pamięć linii, które potencjalnie mogłyby zostać wykorzystane w późniejszym przetwarzaniu, nawet jeśli chwilowo nie łączą się z innymi liniami (liczba punktów na linii nie wystarczy do uznania jej jako finalną). Linie z małą liczbą punktów i nie łączące się z innymi liniami są bezpowrotnie kasowane.

Pamięć z której na bieżąco trwa losowanie linii do zestawiania nazwana została pamięcią linii aktywnych. Rozróżnienie na pamięć linii aktywnych i wstrzymanych wynika z technicznego problemu budowy tej pierwszej. Problem polega na konieczności zapewnienia wielu zapisów/odczytów do takiej pamięci jednocześnie.

Rekord w każdej z pamięci wygląda w ten sam sposób (rys. 4). Zawiera współrzędne punktu początkowego i końcowego linii, liczbę nieudanych prób połączenia linii z innymi liczoną od ostatnio udanego połączenia oraz liczbę punktów tworzących tą linię. Na podstawie dwóch ostatnich parametrów podejmowane są decyzje co robić z linią w razie niepowodzenia najbliższego połączenia dwóch linii (zgodnie z algorytmem, rys. 3).

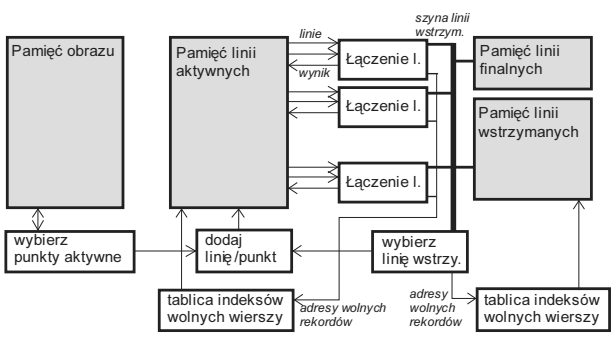


Rys. 4. Format rekordu w pamięci z opisem linii
Fig. 4. Record format for a line in memory

Nieustannie trwa także uzupełnianie pamięci linii aktywnych nowymi punktami z obrazu jak również liniami odłożonymi wcześniej do pamięci linii wstrzymanych (rys. 3, część dolna rysunku). Dobór tego, co wstawić w zwolnione miejsce w pamięci linii aktywnych (czy nowy punkt czy linię wstrzymaną oraz który punkt/linie konkretnie) wykonywany jest losowo.

4. Implementacja algorytmu w układzie FPGA

W układzie FPGA wykonano strukturę taką jak na rysunku 5. Na wejściu dostarczony jest obraz binarny (0-punkt nieaktywny, 1-punkt aktywny; punkt aktywny to taki który może tworzyć linię, punkt nieaktywny to tło). Autor zakłada, że obraz do takiej postaci doprowadzony został uprzednio za pomocą dwóch operacji: detekcji krawędzi oraz progowania (*thresholding*).

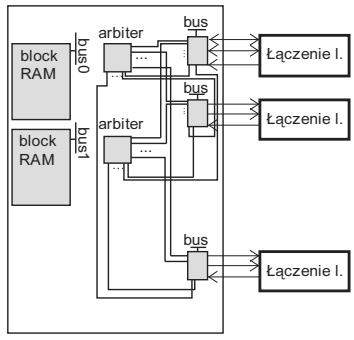


Rys. 5. Architektura procesora do realizacji zmodyfikowanej transformaty RHT
Fig. 5. Processor architecture for modified RHT

Istnieją trzy istotne problemy z którymi trzeba się zmierzyć projektując strukturę dla rozważanego w pracy algorytmu: <1> jak zrobić pamięć z możliwością odczytu/zapisu jednocześnie wielu zmiennych, <2> jak efektywnie śledzić które komórki w pamię-

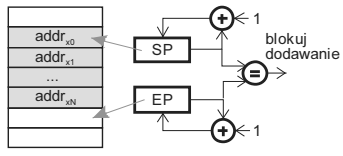
ciach są wolne, gdy kasowanie danych z tych pamięci następuje losowo oraz <3> jak efektywnie znajdować punkty „aktywne" w obrazie w którym większość punktów jest nieaktywnych.

Problem <1> rozwiązano poprzez złożenie pamięci z wielu mniejszych bloków pamięci RAM. Każdy moduł łączenia linii wystawia adres wiersza w pamięci. Na podstawie adresu jednoznacznie identyfikowany jest blok RAM gdzie dane są zapisane. Z każdym modułem blok RAMu związany jest arbiter. Do niego wysyłane jest żądanie dostępu do pamięci. Arbiter odpowiada sygnałem zezwolenia. Dopiero po otrzymaniu zezwolenia można wystawić adres na szynę odpowiedniego bloku RAM. Odwołanie do pamięci trwa zatem dwa cykle. W pierwszym wykonywana jest negocjacja zezwolenia na odwołanie w drugim następuje odwołanie jako takie. Ze względu na losowy charakter odwołań oczekuje się, że statystycznie wszystkie bloki w każdym cyklu będą równo obciążone. W przypadku odmowy możliwości odczytu danej w danym cyklu moduł łączenia linii próbuje ponownie, aż do skutku.



Rys. 6. Architektura pamięci linii aktywnych
Fig. 6. Active lines memory organisation

Problem <2> rozwiązano poprzez wprowadzenie dodatkowej tablicy przechowującej wolne adresy w pamięci (rys.7). Tablica ta ma formę bufora kołowego. W przypadku gdy się zapełni wszelkie nowe zgłaszane wpisy są ignorowane. Informacje o wolnych rekordach pochodzą z bloków łączenia linii (dla pamięci linii aktywnych), wysyłane dla wybranych linii pustych, lub modułu wybierania linii wstrzymanej (dla pamięci linii wstrzymanych). Odrzucenie tych informacji nie skutkuje ich trwała utratą, gdyż odpowiedni moduł zgłosi ją ponownie po ponownym napotkaniu takiego rekordu.



Rys. 7. Tablica wolnych adresów dla pamięci linii aktywnych oraz pamięci linii wstrzymanych
Fig. 7. Table of free records for active/postponed lines memory

Problem <3> rozwiązano poprzez wprowadzenie osobnego modułu zajmującego się wyłącznie losowaniem punktów w pamięci obrazu, sprawdzaniem czy jest to punkt aktywny (jedynka w pamięci) i jeśli tak wpisaniem współrzędnych do bufora. Moduł „dodaj linię/punkt" (rys. 5) w odpowiednim czasie przepiszę tę wartość do pamięci linii aktywnych. Małe prawdopodobieństwo trafienia w punkt aktywny rekompensowane jest szybkim i niezależnym od reszty algorytmu wybieraniem i sprawdzaniem tych punktów (nowy punkt sprawdzany w każdym cyklu zegara).

Ostatnią rzeczą wymagającą krótkiego komentarza jest moduł łączenia linii. Jest on wykonany w sposób proceduralny, wymagając od 11 do 15 cykli na zakończenie operacji. Algorytm samego łączenia wykonano w relatywnie prosty sposób. Najpierw znajdowane są najbardziej odległe od siebie punkty z czterech (odpowiednio P_{00} i P_{0K} dla pierwszej linii i P_{10} oraz P_{1K} dla linii drugiej). Najbardziej odległe punkty tworzą linię. To czy linia jest poprawna oceniane jest na podstawie tego, czy pozostałe dwa punkty należą do tak utworzonej linii. Jeśli tak to liczba punktów na nowo powstałej linii równa się sumie liczby punktów obu linii.

5. Dyskusja i podsumowanie

Główną zaletą przedstawionej w pracy modyfikacji jest niskie zapotrzebowanie na pamięć. Klasyczna HT dla obrazu 100x100 i rozdzielczości 1^o nachylenia wymaga macierz parametrów 360x200 15-bitowych punktów. Wymaga także wykonania średnio 270 wpisów dla każdego punktu aktywnego. Transformata RHT albo pozostawia niezmienioną macierz parametrów albo wprowadza ją w wersji dynamicznej [1]. Wersja dynamiczna wymaga sortowania oraz wstawiania nowych danych w środek innych (przesuwania lub dynamicznego powiązania danych).

W rozwiązaniu przedstawionym w pracy pamięć aktywna może zawierać klasycznie kilkadziesiąt/kilkaset linii. Nie wymaga także żadnego uporządkowania, pozostawiając wszystko procesowi losowego doboru. Wielkość pamięci może być mała ze względu na eliminacje problemu fałszywych linii (opis rozdział 2, rys. 1) z transformaty RHT. Nie rozróżnia się także pamięci punktów aktywnych, traktując je jako krótkie linie, podczas gdy oryginalnie (RHT) pamięć punktów aktywnych może sięgać maksymalnie połowy liczby wszystkich punktów w obrazie (max 5000 wpisów dla obrazu 100x100 punktów).

Rozwiązanie przedstawione w pracy posiada jeden, istotny z punktu widzenia autora, problem. Dobór punktów do pamięci linii aktywnych jest losowy i konsekwentnie nie zawsze najlepszy dla danej sytuacji. Skutkuje to w miarę częstym przerzucaniem linii do i z pamięci linii wstrzymanych. W sytuacji gdy linie są już dość pewne (wystarczająca liczba punktów na linii) można sprawdzić „jakość" linii (liczbę punktów) bezpośrednio na obrazie. Skomplikuje to wzajemne powiązania między modułami, gdyż moduł łączenia linii musiałby móc adresować dane w pamięci obrazu. Czas wykonania operacji w tym module stałby się także jeszcze bardziej nieprzewidywalny. Skuteczność takiego rozwiązania już w wersji podstawowej (dla każdej pary punktów natychmiast następuje sprawdzanie) może być natomiast dość duża, jak przedstawiono w artykule [4]. Autor stosunkowo niedawno miał okazję zapoznać się z treścią artykułu [4] i nie uwzględnił ich w przedstawionym opracowaniu. W najbliższym czasie planuje się prace w tym kierunku.

6. Literatura

[1] L. Xu, E. Oja, P. Kultanen: A New curie detection metod: Randomized Hough Transform (RHT), Pattern Recognition Letters 11, 1990, pp 331-338.

[2] H. Illingworth, J. Kittler: The adaptive Hough transform, IEEE Transactions on Pattern Analysis and Machine Intelligence 1987.

[3] M. Delalandre, B. Simon, S. Guillas, J.M. Ogier, K. Bertet: Stright Line Detection based on the Hough Transform a System and its Performance Evaluation, Draft, 2006.

[4] Ch. Linpeng, Z. Guoliang, J. Guangming, T. Qi: A new algorithm for line detection based on the Randomized Hough Transform, The Eighth International Conference on Electronic Measurement and Instruments, ICEMI'2007.

Artykuł recenzowany