Przemysław MAZUREK
WEST POMERANIAN UNIVERSITY OF TECHNOLOGY, SZCZECIN,
ELECTRICAL FACULTY

Implementation of Spatio-Temporal Track-Before-Detect Algorithm using GPU

Dr inż. Przemysław MAZUREK

Assistant professor of West-Pomeranian University of Technology, Faculty of Electrical Engineering, Chair of Signal Processing and Multimedia Engineering. In 1998 he graduated from the Szczecin University of Technology. In 2002 he defended his doctor of science thesis on the electrical engineering. He has written over 80 scientific papers. His research areas are related to the digital signal processing, digital image processing, pattern recognition and kinematics estimation.

e-mail: przemyslaw.mazurek@zut.edu.pl

Abstract

Track-Before-Detect (TBD) Algorithms are especially suitable for tracking low-observable targets. For low signal-to-noise ratio (SNR <1) cases tracking of such target is possible using TBD approach. Using accumulative approach and more than single measurements a noise level can be reduced in algorithm way, and gives SNR value enhancement. Due to the target's dynamic the possible motion vectors should be considered. In this article in parallel processing approach based on GPU (Graphics Processing Unit) and CUDA (a software platform for GPU programming) is discussed. GPU gives ability of using high number of stream processors and high clocking frequency for parallel algorithms. Because TBD algorithms have abilities of processing in parallel way they are well suited for GPU implementations and real-time processing. Using sparse characteristic of Markov's matrix the Spatio-Temporal TBD algorithm is considered and different implementations schemes (texture, global memory, global with shared memory) for state space access are compared and real-time processing for typical image sizes are obtained.

Keywords: estimation, Track-Before-Detect, digital image processing, parallel image processing.

Implementacja przestrzenno-czasowego algorytmu śledzenia przed detekcją z wykorzystaniem GPU

Streszczenie

Algorytmy śledzenia przed detekcją (TBD – Track-Before-Detect) umożliwiają realizacje systemów estymacji parametrów kinematycznych obiektów także przy warunku SNR<1 (Signal-to-Noise Ratio), co pozwala na śledzenie obiektów, których sygnał jest poniżej wartości szumów. Wykorzystując podejście akumulacyjne oraz więcej niż jeden pomiar możliwe jest zmniejszenie poziomu szumów, a przez to zwiększenie wartości SNR. Z uwagi na dynamikę obiektu konieczne jest uwzględnienie możliwych wektorów ruchu obiektu. Wymagania te powodują, że algorytmy te mają olbrzymi koszt obliczeniowy niezależny od ilości śledzonych obiektów. W artykule zaproponowano rozwiązanie przetwarzania równoległego w czasie rzeczywistym dla obrazów, z wykorzystaniem GPU (Graphical Processing Unit) i platformy programowej CUDA. Zaletą wykorzystania GPU jest możliwość użycia bardzo dużej liczby procesorów strumieniowych, charakteryzujących się prostą budową i wysoką częstotliwością taktowania, co pozwala na efektywną czasowo realizację algorytmów przetwarzania równoległego. Ponieważ algorytmy śledzenia przed detekcją mają cechy predysponujące je do przetwarzania równoległego, więc wykorzystanie GPU jest rozwiązaniem pozwalającym na przetwarzanie w czasie rzeczywistym. W artykule rozpatrywane jest zastosowanie algorytmu rekurencyjnego: przestrzenno-czasowego śledzenia przed detekcją, ze szczególnym uwzględnieniem możliwości redukcji ilości obliczeń dla rzadkich macierzy Markowa. Porównano różne warianty implementacji dla dostępu do wielowymiarowej przestrzeni stanów, która jest przechowywana w pamięci karty graficznej. Dane wejściowe także przechowywane są w pamięci karty graficznej, a dostęp realizowany za pomocą odczytu tekstury, co pozwala na realizację także ułamkowych wektorów ruchu, dzięki wbudowanej interpolacji dwuliniowej. Przestrzeń stanów jest czterowymiarowa i dostęp do niej obciąża znacząco magistralę pamięci. Przetestowano warianty: odczytu z wykorzystaniem tekstur oraz zwykłego dostępu do pamięci, oraz zapisu bezpośredniego i z synchronizowanym buforowaniem w pamięci współdzielonej, uzyskując zbliżone wyniki czasu przetwarzania. Ponieważ w architekturze CUDA nie jest możliwy zapis do tekstury, dlatego konieczne jest dodatkowe kopiowanie wyników przestrzeni stanów do obszaru tekstury, co jednak nie powoduje znaczącego obciążenia w systemie. Wykazano, że możliwa jest realizacja systemów śledzenie przed detekcją z wykorzystaniem GPU pracującym w czasie rzeczywistym. Dla obrazów o rozmiarze 256x256 pikseli osiągnięto ponad 200 klatek na sekundę przy 13 wektorach ruchu, a dla 1024x1024 osiągnięto 15 klatek na sekundę, przy wykorzystaniu procesora G80 (GeForce 8800 GTS).

Słowa kluczowe: estymacja, śledzenie przed detekcją, cyfrowe przetwarzanie obrazów, przetwarzanie równoległe obrazów.

1. Introduction

Tracking algorithms are used in numerous applications and the most known applications of them are radar and video image surveillance systems. Typical tracking systems process measurements in following way: detection algorithms are used for reduction of amount of input data and for separation true data from clutter; detections are processed by state estimator for update and filtering, and the last one is assignment used for track management. Especially two last stages (estimation and assignment) are used together for best performance. As a detection algorithm filtering and threshold (fixed or adaptive) algorithms are used so position of moving target is well established in discreet space. For estimation Benedict-Bordner, Kalman, Extended Kalman or Bayesian algorithms are typically used. There are also a lot of assignment algorithms [1] for track maintenance used for creating and removal particular trajectories. For mulitsensor and multitarget tracking systems algorithms are much more complicated and computation requirements are higher, and for example some assignment problems are NP-hard [1].

Tracking low SNR (Signal-to-Noise Ratio) targets is especially complicated because if detection fails, targets are not detected and tracked (in a case of too high threshold value). If noise clutter is at signal level there are a lot false detections (in a case of too low threshold values). If target signal is at noise or even below noise level floor such target can not be tracked using conventional tracking approach.

There is only one solution that is well destined to solving such scenarios – Track-Before-Detect (TBD) systems. TBD approach test all possible trajectories over a few input scans (tracking phases) and make decision with one trajectory is possible (detection phase). Even if there are no single target available giant amount of trajectories are tested what is computational not optimal, but gives guaranty of detection for assumed set of tested trajectories. Tracking low SNR targets is especially important for airborne, space and submarine surveillance due to availability of 'stealth' technologies.

There are number of TBD algorithms [2, 3] and the most known are spatio-temporal [4], likelihood ratio [5], particle filters [6, 7] algorithms. A lot of computation requirements are typical for every TBD algorithms, but it is worth to be noted that computation cost is less significant for military surveillance applications.

TBD algorithms need high performance processing units and should be carefully designed for particular application. There are VLSI and FPGA solutions for specific systems, e.g. FLIR marine low-horizon [8] detectors but GPU processing is more convenient due to flexibility and low cost.

2. Spatio-Temporal (Spatial-Temporal) TBD

This algorithm is very interesting because is derived from set of FIR filters. The simplest conceptually is non-recurrent version (Moving Average FIR filter) that accumulate three dimensionally pixels from set of stacked 2D images. There are a lot of trajectories and for nonmoving targets it accumulate values over pixels at exactly the same pixel coordinates (this is kind of image denoising technique by accumulative approach). For moving targets accumulation is calculated with an additional motion vector over stacked images (Fig.1) and number of hypotheses is increased.

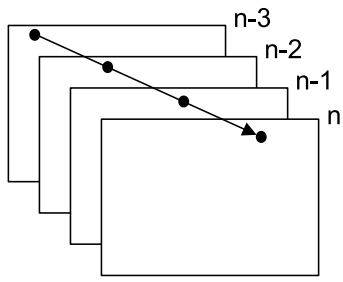

Fig. 1. Model of FIR based Spatio-Temporal TBD (single trajectory)
Rys. 1. Model realizacji przestrzenno-czasowego algorytmu FIR TBD (pojedyncza trajektoria)

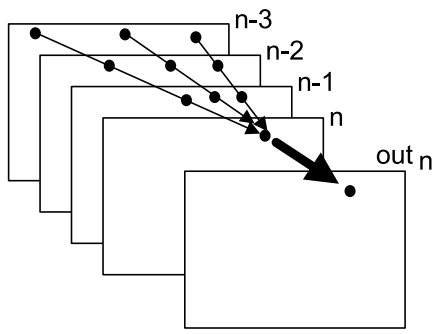

Fig. 2. Data fusion example (using three trajectories)
Rys. 2. Przykład fuzji danych (z wykorzystaniem trzech trajektorii)

There is also a recurrent version that is much more feasible for implementation because less computation is necessary. This is especially important for today available computing devices.

Recurrent version of spatio-temporal TBD algorithm uses following pseudocode and formulas:

Start
// initialization:

$$P(k=0,s)=0 \tag{1a}$$

For $k \geq 1$
//motion update:

$$P^{-}(k,s)=\int_{S} q_k(s \mid s_{k-1}) P(k-1,s_{k-1}) ds_{k-1} \tag{1b}$$

//information update:

$$P(k,s)=\alpha P^{-}(k,s)+(1-\alpha)X(k,s) \tag{1c}$$

EndFor
Stop

where:

k – iteration number,
s – particular space,
X – input data,
P^{-} – predicted TBD output,
P – TBD output,
α – weight (smoothing coefficient),
$q_k(s \mid s_{k-1})$ – Markov matrix.

Similar structure has Bayesian Likelihood Ratio TBD algorithm formulated in [5]. The results obtained in this paper can be extended to this and other classes but excluding Particle Filters TBD [6] algorithms due to theirs statistical state sampling behavior.

The state space has four dimensions so calculating 4D integrals is time consuming task. Integrals should be calculated for every trajectory and there are millions or billions of them for every iteration step.

Assuming linear or semi-linear trajectories the Markov matrix is very sparse and motion update can be reformulated to the much simpler form:

$$P^{-}(k,s)=\int_{L} P(k-1,s_{k-1}) dl_{k-1}\,, \tag{2}$$

where:

l – particular linear trajectory.

If hypothesis trajectories sample only single TBD output they can be simplified to the set of exponential filters separated for every motion vector. For such case there are no dispersion of information during motion update part like previously what gives better results in comparison to general form (1b), but only if hypothesis and real trajectories are fitted. Because information update, sharpen motion update results such case is very interesting. It is worth to be noted that for point target size it is not so good but for extended targets (a few pixels are excited by target's) it can be acceptable.

Every trajectory for particular motion vector is calculated using following formula:

$$P_h=\alpha P_{h-1}+(1-\alpha)X_h\,, \tag{3}$$

For such simplification there are less computation what gives ability of real-time implementation for high frame rate of input images and quite high resolution of sensors using today available computing devices what is crucial for implementation.

3. GPU implementation results

TBD algorithms are very important for military tracking applications and there is only very small part of literature available (most papers are restricted) so it is hard to say about possible GPU implementations and results made by other authors.

Writing GPU code is now possible using C or C-like languages. Efficient GPU programming needs optimization of algorithm and memory accesses what is developer's task. For Nvidia's graphics cards is used CUDA platform (Compute Unified Device Architecture, actually version 2.1) [9, 10] that supports extended C-language for description of parallel processing using threads and CUDA implements SIMT processing (Single-Instruction Multiple-Threads).

For test Nvidia G80 GPU (GeForce 8800 GTS, 128 stream processors, 650MHz core clock, 1625MHz shader clock, 1944MHz memory data rate, 256-bit memory interface, PCI Express x16) was used. Floating values (32-bit coded) was used and four implementation cases were tested.

All TBD algorithms need a very large data transfers between global memory and stream processors. For real-time applications fast shared memory can not be used for state-space storage unfortunately due to very small size. Only input image is common for all state spaces and should be fetched using texture unit (that has own very small cache) or using shared memory.

One of the most important limitations is additional memory transfer required for texture area in every iteration step. Because CUDA does not support writing to the texture area (textures are read-only) results stored in global memory are transferred to the texture area. This is important for state-space because calculation non-texture transfers can be implemented by double-buffer and switching them only by address switching.

Every thread fetch input pixel value and calculate results for all motion vectors. There are 13 motion vectors used in test. Threads can cooperate (e.g. using shared memory) and 16x16 blocks are used.

There is significant difference between texture and regular memory read because texture unit has ability of filtering what is very important for fractional motion vectors. Instead using stream processor for calculating 2D interpolated value a bilinear interpolation results can be obtained from texture unit directly without additional cost. Texture units have own dedicated computing resources so it is very efficient method.

Tab. 1. Performance measurement results of TBD algorithm
Tab. 1. Wyniki pomiaru wydajności algorytmu TBD

Algorithm variant	Frame processing time	Internal memory transfer time	Frames per second
256x256 input frame size			
Input image (texture) Input state space (texture) Output state space (global memory)	4.38 ms	0.45 ms	207 fps
Input image (texture) Input state space (texture) Output state space (global memory via shared memory)	4.30 ms	0.45 ms	210 fps
Input image (texture) Input state space (global memory) Output state space (global memory)	4.51 ms	Not required	222 fps
Input image (texture) Input state space (global memory) Output state space (global memory via shared memory)	4.42 ms	Not required	226 fps
1024x1024 input frame size			
Input image (texture) Input state space (texture) Output state space (global memory)	64.13 ms	0.95 ms	15 fps
Input image (texture) Input state space (texture) Output state space (global memory via shared memory)	63.24 ms	0.95 ms	15 fps
Input image (texture) Input state space (global memory) Output state space (global memory)	65.88 ms	Not required	15 fps
Input image (texture) Input state space (global memory) Output state space (global memory via shared memory)	65.10 ms	Not required	15 fps

In Table 1 are presented results for four implementations and it is shown that results are similar for every algorithm.

The most important is that TBD algorithm can be implemented using today available hardware at quite low cost for typical image sizes and frame rates.

4. Conclusions

Data transfer from global memory to the texture buffer consumes time but it is not so important in comparison to the frame processing.

Data fusion (similar to the shown in Fig. 2) can be implemented using GPU or CPU and is not considered in this paper but necessary for real application. Depending on application it can be simple sum of all trajectories for particular position over all motion vectors (subspaces) or every subspace should be considered separately.

More advanced TBD algorithm implementations on GPU is possible and it will be considered in further research.

5. Acknowledgements

This work is supported by the MNiSW grant N514 004 32/0434 (Poland).

This work is supported by the UE EFRR ZPORR project Z/2.32/I/1.3.1/267/05 "Szczecin University of Technology - Research and Education Center of Modern Multimedia Technologies" (Poland).

6. References

[1] S. Blackman, R. Poupoli: Modern Tracking Systems, Artech House, 1999.

[2] Y. Boers, F. Ehlers, W. Koch, T. Luginbuhl, L. D. Stone, R. L. Streit (eds.): Track Before Detect Algorithms, EURASIP Journal on Applied Signal Processing, 2008.

[3] M. Dragovic: Velocity Filtering for Target Detection and Track Initiation, Weapons Systems Division Systems Sciences Laboratory, DSTO-TR-1406, 2003.

[4] Y. Bar-Shalom: Multitarget-Multisensor Tracking: Applications and Advances, vol II, 1998.

[5] L. D. Stone, C. A. Barlow, T. L. Corwin: Bayesian Multiple Target Tracking, Artech House 1999.

[6] B. Ristic, S. Arulampalam, N. Gordon: Beyond the Kalman Filter: Particle Filters for Tracking Applications, Artech House, 2004.

[7] M. G. Rutten, B. Ristic, N. J. Gordon: A Comparison of Particle Filters for Recursive Track-Before-Detect, 7th International Conference on Information Fusion (FUSION 2005), 2005.

[8] R. C. Warren: A Bayesian Track-before-Detect Algorithm for IR Point Target Detection, Weapons Systems Division Aeronautical and Mari-time Research Laboratory, DSTO-TR-1281, 2002.

[9] NVIDIA CUDA - Compute Unified Device Architecture. Reference Manual v2.0, Nvidia, 2008.

[10] NVIDIA CUDA - Compute Unified Device Architecture. Programming Guide v2.0, Nvidia, 2008.

Artykul recenzowany