Przemysław MAZUREK
ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE,
KATEDRA PRZETWARZANIA SYGNAŁÓW I INŻYNIERII MULTIMEDIALNEJ, ul. 26. Kwietnia 10, 71-126 Szczecin

Optimization of Track-Before-Detect Systems for GPGPU

PhD Eng. Przemysław MAZUREK

Assistant professor in the Department of Signal Processing and Multimedia Engineering at the Faculty of Electrical Engineering, West-Pomeranian University of Technology in Szczecin. Author of more then 80 papers related to the digital signal processing, estimation of object kinematics, biosignals acquisition and processing.

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

Abstract

A computation speed of Track-Before-Detect algorithm with GPGPU implementations are compared in the paper. The conventional and subpixel variants for different thread processing block sizes are compared. Decimation of the state space for reduction of the external memory accesses is assumed. The GPGPU code profiling technique by the source code synthesis is applied for finding of the best parameters and code variants for particular GPGPU.

Keywords: Estimation, Tracking, Parallel Image Processing, GPGPU, Track-Before-Detect.

Optymalizacja systemów śledzenia przed detekcją dla GPGPU

Streszczenie

Systemy śledzenia oparte na schemacie śledzenia przed detekcją (TBD) umożliwiają śledzenia obiektów o niskim stosunku sygnału do szumu (SRN<1), co jest ważne dla zastosowań cywilnych i wojskowych. Konwencjonalne systemy śledzenia oparte na detekcji i śledzeniu nie są odpowiednie z uwagi na dużą ilość fałszywych lub utraconych detekcji. Najważniejszą wadą algorytmów TBD jest skala obliczeń, ponieważ wszystkie hipotezy (trajektorie) powinny być testowane, nawet jeśli nie ma obiektu w zasięgu. Proponowana metoda [8] oparta o decymację daje istotną (kilka razy) redukcję czasu przetwarzania na GPGPU. Programowalne karty graficzne (GPGPU) zawierają dużą ilość jednostek przetwarzania (procesorów strumieniowych) z bardzo małą, ale szybką pamięcią współdzieloną oraz dużą, ale bardzo wolną pamięcią globalną. Proponowana metoda [8] została w artykule przetestowana z wykorzystaniem algorytmu Spatio-Temporal TBD z dodatkowym profilowaniem kodu z wykorzystaniem platformy przetwarzania Nvidia CUDA. Kompilator CUDA jest dodatkowo używany do optymalizacji czasu przetwarzania z różnymi rozmiarami bloku przetwarzania. Przestrzeń stanów jest przetwarzana wewnętrznie z wykorzystaniem pamięci współdzielonej i przechowywana w pamięci globalnej po pewnej określonej liczbie kroków czasowych. Podejście z okienkowaniem jest używane do przetwarzania wejściowych danych pomiarowych 2D przechowywanych w pamięci globalnej.

Słowa kluczowe: Estymacja, śledzenie ruchu, równoległe przetwarzanie obrazów, GPGPU, śledzenie przed detekcją.

1. Introduction

Tracking systems are used in numerous applications like a road, space, air, marine, underwater surveillance [1, 2]. They give ability of tracking numerous parameters related to the object (estimation of the availability, position, velocity, etc.). From signal processing perspective tracking is a filtering technique because the input signal (measurements) is noised and the result (output of tracker) is better in comparison to the input signal. There are a lot of tracking algorithms and the most important are: the Benedict-Bordner Filter, Kalman Filter, and Bayes Filter [1, 2]. Every tracking algorithm have own advantages and disadvantages and should be carefully selected and applied for particular applications. Computation cost for tracking filter is important factor, especially for real-time tracking systems. Implementation of such systems for single object tracking is rather simple but for multiple objects could be very sophisticated. Typical tracking systems consist of three parts: detection (using signal processing algorithms, like threshold), tracking, and assignment. Assignment and tracking parts could cooperate for the track maintenance.

Fig. 1. Model of typical tracking system
Rys. 1. Model typowego systemu śledzenia

Assignment and tracking parts reduce the false detection influence only for high SNR (Signal-to-Noise Ratio). If SNR is very low (SNR<1) there are too much false detections and system can not work properly. Alternative method based on the Track-Before-Detect (TBD) scheme is used for such situations [2, 3]. TBD algorithms assume availability of objects in all available states (e.g. positions, velocities) and update state space using allowable transition jumps. Detection of the object is possible after tracking due to accumulative approach typically [2, 3].

Fig. 2. Model of Track-Before-Detect system
Rys. 2. Model systemu śledzenia przed detekcją

Accumulative approach increase SNR using multiple observations what gives ability of object's tracking even if this object is hidden in a noise (object's signal is below a noise floor). Detection of such weak signal (dim object) is important in numerous applications, especially military (e.g. tracking of 'stealth' targets). TBD systems could be used for tracking object of any-kind. If SNR is high a more convenient are conventional systems due to very large computation cost for TBD. Calculation of all trajectories is necessary even if no one object is in the range for reliable tracking systems.

2. Types of TBD algorithms

There are two kinds of TBD algorithms based on recurrent and non-recurrent processing scheme. Recurrent algorithms are preferred due to reduction of memory requirements and less computation cost. Spatio-Temporal (or Spatial-Temporal) TBD algorithm has a following structure:

Start

$$P(k=0,s)=0 \qquad \text{// initialization:} \tag{1a}$$

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's matrix.

Such algorithm could be simplified for the linear trajectories and the constant velocity object. Such assumption reduces possibility of tracking for maneuvering object but is very simple in implementation [4]:

$$P_h = \alpha P_{h-1} + (1-\alpha) X_h \,. \tag{2}$$

TBD systems could be also used for subpixel object tracking [5]. Subpixel object is measured as an excitation of a single sensor cells but due to movements the resolution of the sensor could be multiplied virtually. For non-moving object a single cell is excited so tracking resolution is equal to the real resolution of the sensor. Similar situation occurs for exact vertical and horizontal trajectories where the single column or row cells are excited only. This situation gives ability of tracking with different horizontal and vertical virtual resolution.

$$P(k,s) = \alpha P^{-}(k,s) + (1-\alpha) X(k, \lfloor s/R \rfloor) \tag{3}$$

Simplification of subpixel TBD algorithms is also possible and in such algorithm there is significant difference between state-space and measurement space sizes. Single excited cell value (e.g. pixel value in imaging sensors) is used for update of multiple state-space cells (3) and R is spatial divider (subpixel ratio). For example for R=3 there are 9 subpixels for every cell for 2D sensor. This behavior is very important in implementations because reduce memory read operations in information update formula. For example, if there are 9 subpixels only single memory read operation is necessary for this formula.

3. Optimization of TBD algorithm

TBD systems are computation demanding and for real-time processing are used VLSI (Very Large Scale Integration) chips, FPGAs (Field Programmable Gate Arrays), DSPs (Digital Signal Processors) and multicore clusters. Fortunately TBD algorithms have algorithm structure that fit quite well in today available computing devices. There are two limitation related to the Spatio-Temporal TBD algorithm: the time of calculations and the memory transfers for state-space and new input data. State-space is multidimensional for TBD system and is the main limitation. For example: if input space has 1000x1000 size and there are 10 motion vectors for simplified formula (transitions only between current state-space motion vector) there are 10 millions of read and write operations related to the state-space.

GPGPU (General Purpose Graphics Processing Unit) has possibility in parallel processing using set of threads organized into blocks [6, 7]. High computation speed is only achievable if the main bottleneck problem (global memory transfers) is reduced. The modification of algorithm (Fig.3) based on decimation (downsampling) is proposed for TBD systems [8].

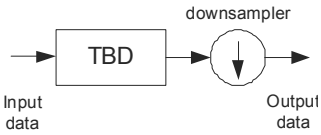

Fig. 3. Model of TBD system with downsampled output
Rys. 3. Model systemu TBD z decymatorem

Accumulation approach needs a lot of data and overall TBD system works as a low pass filter so sampling ratio for TBD output could be reduced without significant disadvantages.

Modified processing scheme [8] is proposed (Fig.4) introducing a delay for output data delivery. Instead processing of every incoming frame one by one a set of input frames is stored into global memory. Data are processed using window approach. For selected position of window corresponding areas in input measurement and state space is selected. It gives ability of utilization of internal a very fast shared memory as storage for state-space temporal results. At the beginning an appropriate area of state-space is one-time fetched from the global memory and stored into shared memory. In next steps a values stored in the shared memory are updated by measurements. Result (state-space stored in shared memory) is returned to the global memory and is available as a final output. Such approach reduce significantly a memory transfers for the state-space what is very important for TBD system (Fig. 5).

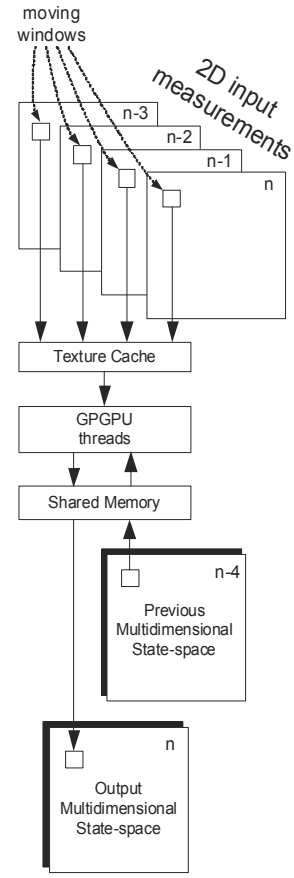

Fig. 4. Processing model for TBD downsampled system
Rys. 4. Model przetwarzania dla systemu TBD z decymacją

GPU code writing 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 the CUDA platform (Compute Unified Device Architecture, actual version 2.3) [6, 7] are used, and utilize extended C-language for description of processing in parallel using threads. CUDA implements SIMT processing (Single-Instruction Multiple-Threads). For tests 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) is used. Floating (32-bit coded) values are used for input and output data. Input measurement has 1024x1024 size and the output state space has 13x1024x1024 size so there are 13 of the motion vectors assumed.

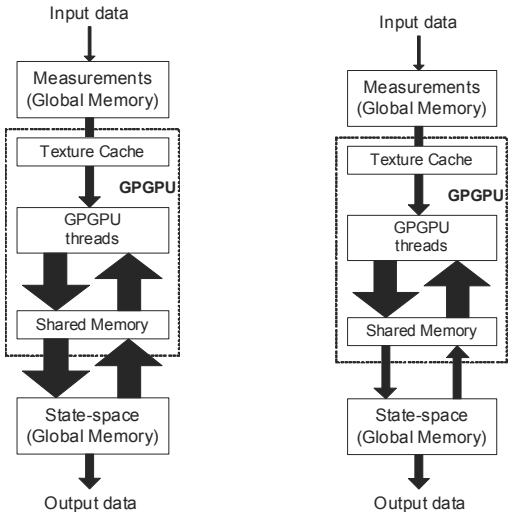

Fig. 5. Comparison of conventional TBD (left) and downsampled (right) TBD systems. Size of the arrows visualize amount of transferred memory data
Rys. 5. Porównanie realizacji systemów TBD: konwencjonalnego (z lewej) i z decymacją (z prawej). Rozmiar strzałek symbolizuje ilość przesyłanych danych

4. Code profiling

Optimization of code is possible using algorithm analysis but is optimal for particular GPGPU. There are multiple parameters that influent on final performance, and manual code synthesis is time consuming operation. For this algorithm a two hours were necessary for testing all cases (8 block widths, 16 block heights, and 20 downsampling ratios) using CUDA and Microsoft Visual Studio C Compiler controlled by author's profiling tool (a set of scripts).

In Fig. 6 are shown example results for the system without downsampling and some configuration of width and height of processing window (thread block) are rejected and there are not available results (e.g. width=64 and height=16). There are visible low performance results for widths: 8, 24, 40, and 56, and the highest performance are for 16, 32, 48, and 64 block widths, what is result of better coalescence accesses [6, 7].

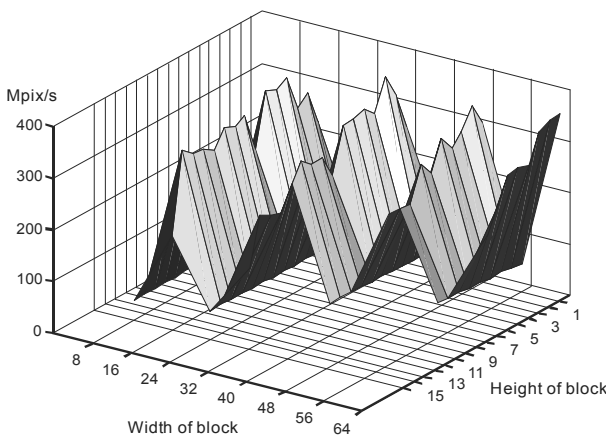

Fig. 6. Example set of processing speed for conventional TBD system
Rys. 6. Przykładowy zbiór szybkości przetwarzania dla systemu konwencjonalnego TBD

The best and worst results are compared for variable downsampling ratio (Fig. 7). Random selection of parameters is not recommended because difference between both boundaries is large (a few times higher performance for best case in comparison to the worst case). Downsampling ratio equal to the 1 is the case of conventional TBD system (without downsampler). The performance of TBD algorithm dependent on downsampling ratio is very well visible (a few time higher performance in comparison to the conventional). Subpixel TBD algorithm performance is only a few percent higher in comparison to the regular TBD.

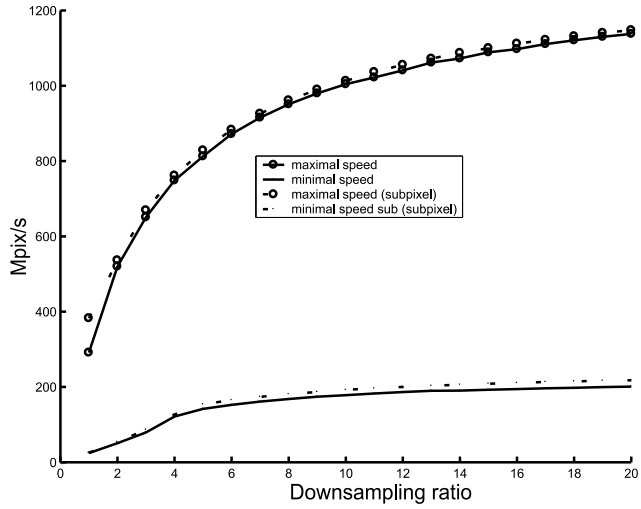

Fig. 7. Best and worst processing speed for conventional and subpixel Spatio-Temporal TBD algorithm dependent on downsampling ratio
Rys. 7. Prędkość przetwarzania dla najgorszych i najlepszych przypadków dla zwykłego i podpikselowego algorytmu Spatio-Temporal TBD w zależności od stopnia decymacji

5. Conclusions

In the paper are compared two TBD systems (conventional and downsampled) for two cases of implementation (information update formula: pixel and subpixel based). System with subpixel processing has a few percent higher performance due to multiple utilization of the input values.

The main bottlenecks of recurrent TBD algorithm implementation on GPGPU are the memory transfers. Using downsampling approach a few times higher performance could be obtained. Application of GPGPU for TBD systems is very important due to performance to cost high ratio and market's availability.

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] Blackman S., Poupoli R.: Modern Tracking Systems, Artech House, 1999.
[2] Bar-Shalom Y.: Multitarget-Multisensor Tracking: Applications and Advances, vol II, 1998.
[3] Stone L.D., Barlow C.A., Corwin T.L.: Bayesian Multiple Target Tracking, Artech House 1999.
[4] Mazurek P.: Implementation of Spatio-Temporal Track-Before-Detect Algorithm using GPU. Pomiary Automatyka Kontrola, vol. 55 nr 8, 657-659, 2009.
[5] Daca W., Mazurek P.: Subpikselowe śledzenie przed detekcją dla matrycowych detektorów optycznych. VI Krajowa Konferencja Elektroniki, Darłówko Wschodnie 2007, 429-434, 2007.
[6] NVIDIA CUDA - Compute Unified Device Architecture. Reference Manual v2.0, Nvidia 2008.
[7] NVIDIA CUDA - Compute Unified Device Architecture. Programming Guide v2.0, Nvidia 2008.
[8] Mazurek P.: Optimization of Bayesian Track-Before-Detect Algorithms for GPGPUs Implementations. Electrical Review 2010 (in print).

otrzymano / received: 10.05.2010
przyjęto do druku / accepted: 04.06.2010 *artykuł recenzowany*