Chapter 7
Accelerated Rendering and Fast Reconstruction of EEG Data in Real-Time BCI

Ning Wang, Peng Lu, Lipeng Zhang, Shijie Li and Hanghang Hu

Abstract In real-time BCI (Brain Computer Interface), the *ITRs* (Information Transmission Rates) is one of the most common criteria for evaluating the performance of the whole system, and one of the key factors is the duration of one single trial. This paper aims at improving the *ITRs* by decreasing the duration. Accelerated rendering is used for drawing raw EEG (Electroencephalogram) data in presentation thread, and thread scheduling based on adaptive one-sided fuzzy inference and the mechanism of mutual exclusion and synchronization with semaphore is adopted to recombine intervening data blocks in reconstruction thread.

Keywords Real-time BCI · The mechanism of mutual exclusion · EEG data · Thread scheduling · Fuzzy inference

7.1 Introduction

The target of BCI (Brain Computer Interface) is to transform awareness into real-time commands of controlling external devices and communication tools [1, 2]. Compared with offline BCI, continuous and real-time judgement of brain status is needed in online BCI. The key strategy of a high-performance BCI system is the immediate processing of raw EEG (Electroencephalogram) data [3]. By far, the usual policy is adopting general software platform, for instance, LabVIEW

N. Wang · P. Lu (✉) · L. Zhang · S. Li · H. Hu
The 27th Research Institute of China Electronics Technology Group Corporation, Zhengzhou 450047, China
e-mail: lupeng@zzu.edu.cn

N. Wang · P. Lu · L. Zhang · S. Li · H. Hu
School of Electric Engineering, Zhengzhou University, Zhengzhou 450001, China

Z. Deng and H. Li (eds.), *Proceedings of the 2015 Chinese Intelligent Automation Conference*, Lecture Notes in Electrical Engineering 336,
DOI 10.1007/978-3-662-46469-4_7

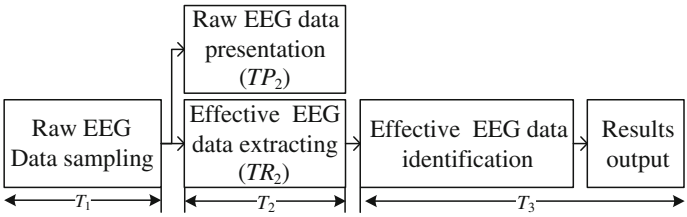

Fig. 7.1 General procedures of online BCI

components are used to handle data streams in document [4], and BCI2000 is adopted in document [5]. While general software packages fixed function interfaces or mix too many other function modules, their flexibility and efficiencies are limited for online BCI system when facing complicated problems. This paper focuses on solving the instantaneity of online BCI through deeper threads mechanism.

The main performance index of an online BCI system is the ITRs (Information Transmission Rates) calculated with the following equation (unit: bits/min),

$$\mathrm{ITRs} = B * 60/T, \tag{7.1}$$

where

$$B = \mathrm{lb}N + P\mathrm{lb}P + (1-P)\mathrm{lb}[(1-P)/(N-1)], \tag{7.2}$$

indicates the amount of information transmission in a single trial, N indicates the classification number of awareness recognition, P indicates the accuracy rate of recognition, and T is the period (unit: s) of a single trial. When N is certain, ITRs have positive relationship with P and inverse relationship with T. Thus decreasing T is one of the keys to improve ITRs.

In general, online BCI system [6] with synchronous stimulation, one single trial can be decomposed into three stages $T1$, $T2$, and $T3$, as shown in Fig. 7.1. The system samples raw EEG data in $T1$, presents and reconstructs them in $T2$, and analyzes them in $T3$. Before effective EEG data comes in the next trial, the whole procedure has to be finished in time. Document [7] lists each parameter's influences on ITRs, which still assumes that each trial runs serially.

In our experiment, we found that stage $T2$, $T3$ always consume more time than stage $T1$, and there are even losses of effective EEG data in some trials. The primary reason is that a single thread cannot finish the handling of analysis in the time between the front and rear two sections of effective EEG data. In synchronous BCI, before the next section of effective EEG data comes, the tasks in $T2$ and $T3$ should be completed in time. This paper focuses on the time factor for ITRs and aims at decreasing $T2$ through concurrent thread mechanisms.

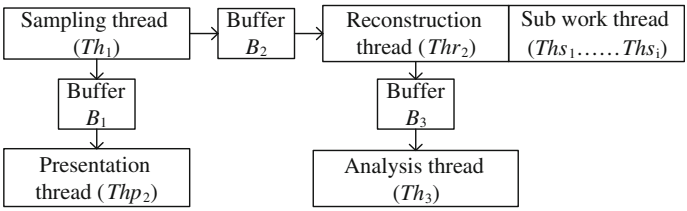

Fig. 7.2 Threading module structure

7.2 Method

Through mechanism of thread concurrency, we decompose a single trial into more subtasks further, allowing the front subtask enter next trial while the rear subtask is executing, and make concurrent subtasks execute among adjacent trials.

Decompose the whole online BCI system into 3 main concurrent modules, with the second module including 2 main threads: the presentation thread and the reconstruction thread. Duration of the presentation thread and the reconstruction thread is our research target. Accelerated rendering is used for drawing raw EEG data in presentation thread, and thread scheduling based on adaptive one-sided fuzzy inference [8] and the mechanism of mutual exclusion and synchronization [9] with semaphore is adopted to recombine intervening data blocks in reconstruction thread.

The thread module structure of the online BCI system is designed as Fig. 7.2. There are four main threads, namely the sampling thread Th1, the presentation thread Thp2, the reconstruction thread Thr2, and the analysis and recognition thread Th3. Besides that, there are more subthreads dominated by Thr2 for fitting better the complicated change of EEG data size.

7.2.1 Accelerated Rendering of Raw EEG Data

In general, EEG acquisition system such as Neuroscan, the raw EEG data is presented in a method of superposition by column from left to right with the ordinary GDI (Graphics Device Interface) drawing [10]. We found that the method of superposition by column only draws in a local area each time; thus it has a relative small workload.

This kind of presentation has two main shortages:

1. Each drawing process contains an operation of clearing the local area so as to cover the data curves of last drawing. Moreover, as the number of EEG sampling channels increases, and the sampling rate speeds up, this phenomenon becomes more serious. Because the GDI drawing is based on CPU [11], more operations means more burdens for CPU.

2. The procedure of superposition by column is continuous in time but not in spatial presentation; from Fig. 7.1, we can see that later EEG data may appear on the left of earlier EEG data. If the EEG data curves can be presented from left to right all the time, then the presentation will look more intuitive.

Based on the two factors proposed, the accelerated rendering technology is an available choice. As most computers support GPU-accelerated rendering currently [12], delivering the presentation to GPU and taking full advantage of the graphics card can contribute to fluent presentation of large-scale EEG data and relieve the burden of CPU to some extent. What is more, this can decrease duration of stage *T*2, and reserve more CPU resources for the analysis process of stage *T*3.

Due to the fixed drawing area in a practical EEG system, a circular buffer of fixed size is feasible, and in the mode of hardware acceleration.

Assume that the sampling rate is *S*, the EEG channel number and column number of each data block transmitted from the network is *N* and *C*, the pixel width and height of the drawing area is *W* and *H*, and the duration for the presentation of the whole drawing area is Tp2 (unit: s).

The linguistic descriptions for the procedure of accelerated rendering are as follows:

1. Buffer construction. Construct a circular buffer *B*1 with a length of *S**Tp2**N* as the frame buffer unit, and initialize the data pointer pHead and pTail to point the head of *B*1; construct a vector buffer with a length of *S**Tp2 for the accelerated rendering of graphics card.
2. Matrix transposition. Due to the fact that EEG data of all channels exist continuously in each data block from the network, a matrix transposition is needed so that the EEG data of the same channel can be attached directly, as shown in Fig. 7.2.
3. Data block connection. Now the data block can be attached directly, so put it into *B*1 directly and move the tail pointer at the same time. When the length of EEG data in *B*1 is *S**Tp2, transpose it again to a matrix with row *N* and column *S**Tp2.
4. Coordinate mapping. According to the position of the drawing area, map the EEG data values to the screen coordinates and send them to the vector buffer. Then take each row of data in the vector buffer as one unit and draw them to the hardware off-screen buffer; after drawing, send them to hardware frame buffer to finish the accelerated rendering of one row.
5. Pointer handling. When the tail pointer pTail arrives to the tail of *B*1, it automatically moves from the head next time. When pTail exceeds the head pointer pHead, pHead automatically moves to the next data block after pTail.

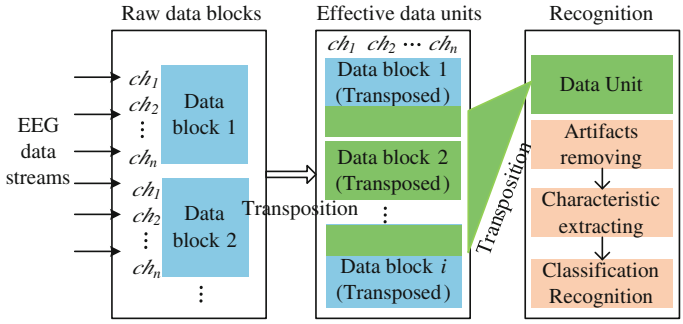

Fig. 7.3 Reconstruction of effective EEG data

7.2.2 Thread Scheduling Against Variable EEG Data

Reconstruction of effective EEG data is a procedure of recombining the data in working status before recognition. The duration of this procedure is marked as Tr2 in Fig. 7.1, and the target of this section is to decrease Tr2.

Assume the column number of effective EEG data unit for one-time recognition is L, due to the uncertain appearing time of event mark value in raw EEG data block, an effective EEG data unit may not start exactly at the beginning of an EEG data block, so the number of raw EEG data block needed to recombine an effective EEG data unit should be $L/C + 1$, in which C is the column number of a raw EEG data block mentioned in Sect. 7.2.1. The flowchart for reconstruction of effective EEG data is shown in Fig. 7.3, in which chi indicates the ith sampling channel.

In Fig. 7.3, the green fill area represents an effective EEG data unit, and the processing steps of reconstruction are as follows:

1. Buffer construction. Construct a second buffer named B_2 for storing an effective data unit, and initialize the data pointer pHead and pTail to point the head of B_2;
2. Matrix transposition. Read the event mark value in the raw EEG data block, and transpose the data block from a matrix with N rows and C columns to another with C rows and N columns.
3. Data blocks recombining. The data blocks transposed can be put in B_2 and stored sequentially. Move the head pointer pHead to the position pos_1 where the event value begins to appear, and the tail pointer pTail to the position $pos_2 = pos_1 + k*C$ when the following kth data block comes, so the length between pHead and pTail is always an integer multiple of that for a data block. When the length between pHead and pTail is L, transpose the matrix of the whole data unit again and use the effective data unit with row N and column L for analysis afterward.
4. Pointer reset. Move pHead and pTail to the head of B_2 and begin the next reconstruction of another effective data unit.

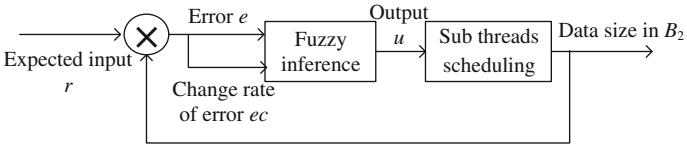

Fig. 7.4 Thread scheduling based on one-sided fuzzy inference

Different sampling rate and the data reading speed will lead to increase and decrease variations of data size. Our experiments show that, due to the accelerated rendering process, the EEG data in buffer $B1$ can always be consumed in time, whereas in buffer $B2$, the EEG data size may generate all kinds of increase and decrease variations.

Predicting the changes of data size effectively, and handling the EEG data flexibly, is the key to prevent the space complexity from increasing, reduce system delay, and improve the efficiency of the system. The online BCI system has a typical environment of time-varying multitasking, and compared with traditional methods, fuzzy inference has a prominent advantage in predicting data variations [13]. Therefore, a method of fuzzy inference is designed to solve the problems of predicting data streams variations in nonlinear environment, as shown in Fig. 7.4. The reconstruction process is fulfilled by the reconstruction thread Thr2 and more subworking threads Ths. The input $r = 0$ indicates the desired data size in buffer $B2$, the error e indicates the current actual data size, the error rate ec = de/dt indicates the change rate of data streams, and u indicates number of desired subworking threads. Data size and the number of subworking threads cannot be negative, as a result, the structure above is a procedure of one-sided fuzzy inference.

Set M as the initial total number of subworking threads, K as the number of the subworking threads inferred by fuzzy inference. Take e, ec, and u as the input and output of fuzzy inference, use positive domain to fuzzily e and u, and use two-sided domain to fuzzily ec, and adopt the triangle membership function and the Mamdani minimax reasoning method to design one-sided fuzzy inference rules, as shown in Table 7.1.

Adopt gravity method for the defuzzification process, and assume the ith fuzzy rule is:

$$R_i: \text{ If } x \text{ is } A_i \text{ and } y \text{ is } \mathrm{B}_i, \text{ then } z \text{ is } \mathrm{C}_i,$$

Table 7.1 One-sided fuzzy inference rules

e	ec						
	NB	NM	NS	Z	PS	PM	PB
Z	Z	Z	Z	Z	PS	PS	PM
PS	Z	Z	Z	PS	PS	PM	PM
PM	Z	Z	PS	PS	PM	PM	PB
PB	Z	PS	PS	PM	PM	PB	PB

where Ai, Bi, and Ci are, respectively, the fuzzy subset of input variables x, y, and output variable z, thus fuzzy set of z can be obtained through (7.3);

$$R_{\mathrm{i}} = (A_i \text{ and } B_i) \rightarrow C_i;\ R = \bigcup_{i=1}^{n} R_i; C^T = (A^T \times B^T) \circ R = \bigcup_{i=1}^{n} C_i^T, \tag{7.3}$$

where A^T, B^T are the fuzzy sets of inputs. The implication operation "$\rightarrow$" fuzzy "and" operator, the synthetic operation "$\circ$" adopts maximum–minimum method, and the minimum, minimum indicate fuzzy "and," "or" operator. Accurate output z_0 can be converted from fuzzy quantity through (7.4).

$$z_0 = \frac{\sum_{i=1}^{n} z_i \int \mu_C(z_i)}{\sum_{i=1}^{n} \int \mu_C(z_i)}, \tag{7.4}$$

where $\int \mu_C(z_i)$ is the area of conclusion membership function for the ith rule, $z_i (i = 1, 2, \ldots, \mathrm{n})$ indicates the center of each conclusion membership function.

For Mamdani fuzzy inference process with double inputs and one output, set the error e and its change rate ec as input, and u as output, considering the influences of quantization factor k_e, k_{ec}, and scaling factor k_u on the system is not monotonous, influences of different stages are distinct and restrict for each other. Dynamic correction factors are adopted to adjust quantization and scaling factors k_e, k_{ec}, and k_u, the adjustment rules are as follows:

$$\begin{aligned} k_e(n+1) &= k_e(n) + \delta_1 \\ k_{ec}(n+1) &= k_{ec}(n) + \delta_2 \\ k_u(n+1) &= k_u(n) + \delta_3 \end{aligned}, \tag{7.5}$$

where $\delta_1, \delta_2, \delta_3$ are, respectively, the dynamic correction factors of the quantization factor k_e, k_{ec}, and scaling factor k_u.

In order to make the adjustment rules play a part in the whole domain, use another variable ε as a threshold value to make the control effect more accurate for small errors, so as to ensure the control accuracy. As $e > 0$, Thr2's online automatic adjustment strategies of parameters are set as follows:

$$\begin{aligned} &\mathrm{Rule}_1 : \text{If } e > 0 \text{ and ec} > 0 \text{ then} \\ &\delta_1 = \Delta k_e, \delta_2 = 0, \delta_3 = -\Delta k_u; \\ &\mathrm{Rule}_2 : \text{If } e > 0 \text{ and ec} < 0 \text{ and } e > \varepsilon \text{ then} \\ &\delta_1 = \Delta k_e, \delta_2 = 0, \delta_3 = \Delta k_u; \\ &\mathrm{Rule}_3 : \text{If } e > 0 \text{ and ec} < 0 \text{ and } e < \varepsilon \text{ then} \\ &\delta_1 = -\Delta k_e, \delta_2 = \Delta k_{ec}, \delta_3 = -\Delta k_u; \end{aligned}$$

In rules proposed above, $\Delta k_e, \Delta k_{\mathrm{ec}}, \Delta k_u$ are, respectively, the minimum increment of k_e, k_{ec}, and k_u, whose values can be set according to actual situations.

7.2.3 Restricted Access to Shared Resources

Concurrent multitasks can make full use of system resources, and improve the performance of online BCI data analysis system [14]. However, influenced by scheduling properties of the operating system [15], intermediate results are not ordered. In order to get true awareness instructions, a reliable mechanism of mutual exclusion and synchronization to ensure the correct recombination of raw EEG data is necessary.

For the analysis process in real-time BCI, the procedure of EEG data processing is that Th1 receives data and put it in $B2$, Thr2 control the scheduling of subthreads Ths, Ths read, recombine EEG data to the form of data units and put them in buffer $B3$. Th3 reads data units from $B3$ and identify them. Direct production–consumption relationship exists between Th1 (P) and Ths ($C1$, $C2$, …, Cn), Ths ($P1$, $P2$, …, Pn) and Th3 (C).

Since Th1, Ths, and Th3 are related to the access of shared resources in $B2$, $B3$, a reliable mechanism of mutual exclusion and synchronization is the key to manage shared resources. Therefore, to solve the mutex and synchronization problem, the system kernel object such as the mutex lock, the event, and semaphore object are adopted.

Mutual exclusion and synchronization access rules on $B2$ are as follows:

1. $B_2 = 0$, all Ths get into synchronous waiting state;
2. $B_2 \neq 0$, one Ths consumes data block in a method of mutual exclusion.

Set up a semaphore object between Thr2 and Ths, and initialize the resource count with 0 and the maximum resource count with M; Set a mutex lock among multiple Ths with its initial state signaled. For thread Thr2, the result K of fuzzy inference determines the implement times of V.

Mutual exclusion and synchronization access rules on $B3$ are as follows:

1 Multiple Ths put data units into B_3 in the correct sequence exclusively;
2 $B_3 = 0$, Th_3 get into the synchronous waiting state.

Set a mutex lock (hMutex2) for multiple Ths with initial state signaled; Set an event object (hEvent2) between Ths and Th3 with initial state nonsignaled; Set two global variables (TR, TW) protected by the mutex as the token numbers to control the correct recombination of intermediate results. Each Ths carries the token number information through a TLS (Thread Local Storage) [16] variable with the same name.

7.3 Results and Discussion

Subject selections: Select graduate students with healthy physical and normal vision correction, and the ratio of male to female is 1:1. Subjects and the EEG acquisition system are in a shielded room. Indoor lighting is darker to reduce the EOG (Electro-Oculogram) artifacts and distraction caused by surrounding environment.

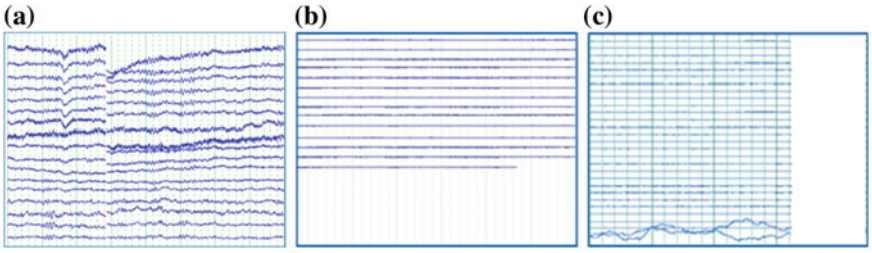

Fig. 7.5 Delay of screen refresh. **a** Neuroscan's normal drawing status. **b** Neuroscan's row drawing delay in repainting. **c** Our client's column drawing delay in repainting with ordinary drawing

System structure configurations: CPU Intel (R) Celeron (R) 2.5 GHz, 2.5 GHz; RAM: 4.00 GB; Operating system platform: Windows 7, 32 bit; Signal stimulation equipment: E-Prime2.0; Data acquisition system: Neuroscan 4.5; Offline analysis environment: MATLAB R2010a.

Experimental paradigms: Stimulation time of each trial: 3 s; Stimulus onset asynchrony: 2 s; Presentation order: random; Use 10–20 system electrode cap with 64 channels, and choose original raw EEG data of C3, C4, FC3, and FC4 channels for multimodal analysis.

We chose six students (male:female = 3:3), and let each one have two online synchronous stimulus experiments. In each one's two experiments, one was with ordinary drawing and thread scheduling, while the other was with accelerated rendering and adaptive thread scheduling. Moreover, the latter experiment's effective EEG data units were immediately saved as the experiment was in progress, so that we could verify the accuracy of online BCI experiment results through offline analysis. Each experiment contains 80 trials.

First, we compared the two kinds of drawing effects and durations of completing a whole window screen repainting.

Figure 7.5 displays the effect of ordinary drawing method. In normal running state, this kind of presentation through superposition by column seems not problematic; however, when encountering the restoring of a window from the minimized state, the window screen began to show an evident sign of stuck drawing process due to tremendous EEG data.

In contrast, Fig. 7.6 shows the accelerated rendering for all kinds of status. Dynamic drawing processes reflect the rapidity and stability of accelerated rendering.

Moreover, the duration of drawing time for a whole window screen is shown in Fig. 7.7 for the six subjects, from which, we can see that the accelerated rendering consumes significantly less time than the other.

Then we set a monitoring window to display dynamic changes of data size in $B2$ and corresponding number of active subthreads, and compared thread scheduling under different conditions, including the ordinary methods just according to the EEG data size in $B2$, and the adaptive methods based on the adaptive one-sided

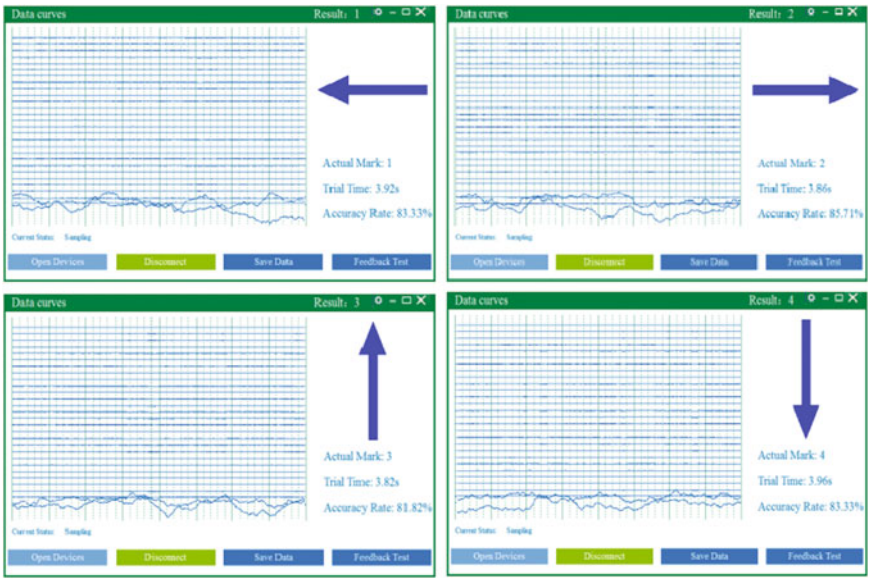

Fig. 7.6 Presentation of accelerated rendering

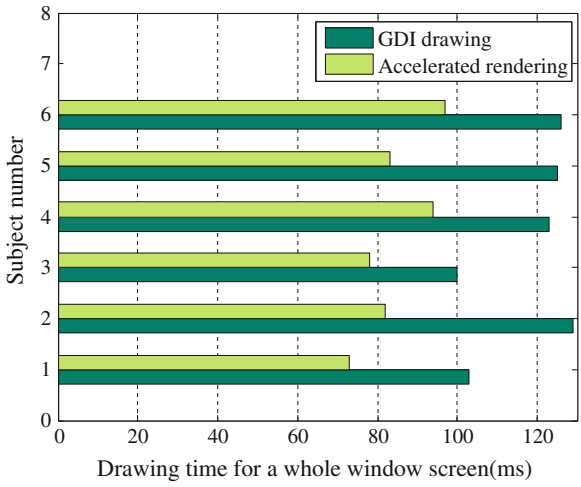

Fig. 7.7 Comparison of two kinds of drawing time for a whole window screen repainting

fuzzy inference according to both the EEG data size and its variation trend. Figure 7.8 shows dynamic variation of data size in $B2$ and number of active subthreads following time under condition of ordinary thread scheduling.

Next, we tested the adaptive fuzzy inference process in the main working thread Thr2, and provided a set of suitable parameters according to the actual situation. The basic parameters include basic domain and fuzzy subset domain of the error e,

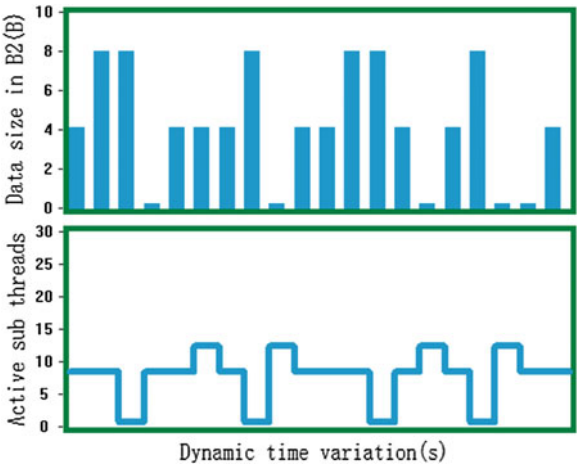

Fig. 7.8 Thread scheduling through ordinary methods

Table 7.2 Setting of domain, quantization factor, and scale factor

Variable	Basic domain	Fuzzy subset domain	Quantization/scale factor
e	[0, 5]	[0, 6]	$k_e = 1.2$
ec	[−20, 20]	[−6, 6]	$k_{ec} = 0.3$
u	[0, 30]	[0, 6]	$k_u = 0.2$

the error rate of change ec, and the output u, and the corresponding quantization, scale factor k_e, k_{ec}, and k_u, as shown in Table 7.2.

The minimum increments of k_e, k_{ec}, and k_u were set as $\Delta k_e = 0.5k_e$, $\Delta k_{de} = 0.25k_{de}$, $\Delta k_u = 0.125k_u$, the switch threshold of error change was set as $\varepsilon = 3$, and the parameters of the fuzzy inference process were adjusted according to Rule1, Rule2, and Rule3 proposed above in Sect. 7.2.2. Afterward, we re-ran the system and observed dynamic variation of data size in $B2$ and corresponding number of active subthreads following time under adaptive fuzzy inference, as shown in Fig. 7.9.

Compare Figs. 7.9 with 7.8, data size in $B2$ changes obviously and K keeps a trend of oscillation for ordinary thread scheduling, whereas on condition of adaptive fuzzy inference, data size in $B2$ becomes relatively stable, and with the increase of K from the initial stage, only some but small overshoot appears, moreover, K is always approaching a stable state following time variation.

At last, we compared the average delay time of the two kinds of online handling methods in each trial. The raw EEG data was saved for offline analysis at the same time. Final statistical results are shown in Fig. 7.10 and Table 7.3, in which the offline recognition time amounts to the serial time of one trial, i.e., $T1 + T2 + T3$, considering some overlaps existing between two adjoining trials on condition of online real-time concurrence, the duration of one trial in online BCI is an average of all trials from the start time to the end time.

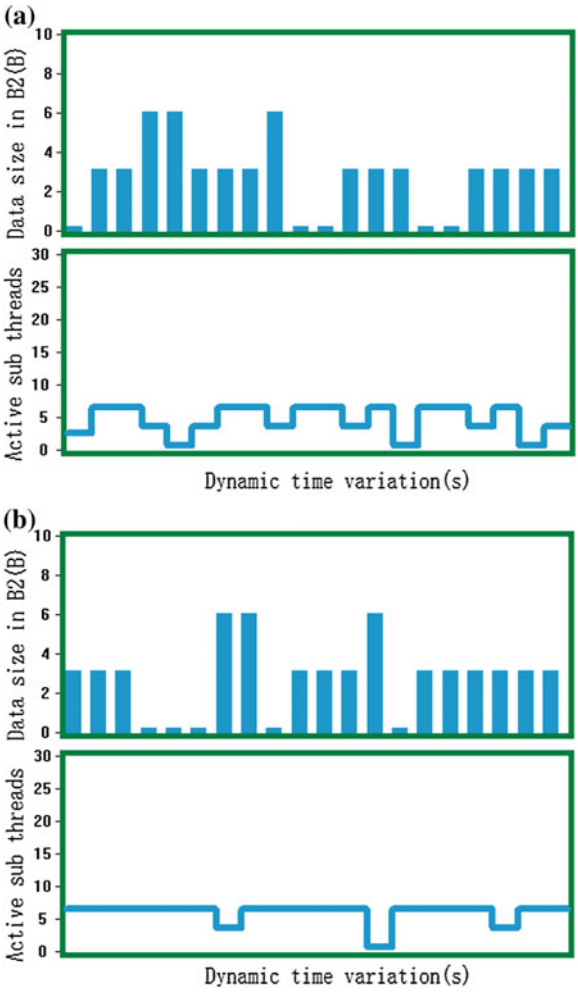

Fig. 7.9 Thread scheduling through adaptive fuzzy inference. **a** System running in the initial stage. **b** System running in the stable stage

Based on the same recognition algorithms, the procedure of offline analysis amounts to implementation of serial trials, thus has the longest time, as can be seen from Fig. 7.10. Concurrence of online BCI data analysis system makes the delay time overlapped for each trial, thus reduces the average delay time of each trial. What is more, from Fig. 7.8 and Table 7.3 we can see that ordinary thread scheduling is less adaptive to high-speed EEG data streams, thus has a relatively low recognition rate due to some losses of effective EEG data, whereas adaptive thread scheduling decreases the average delay time to about 2.1 s with a comparative accuracy rate, which is significant enough to the improvement of ITRs.

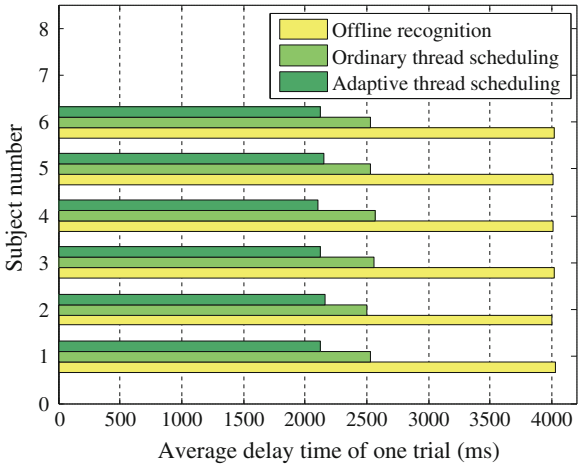

Fig. 7.10 Delay time of one trial under different conditions

Table 7.3 Result comparison for 80 trials

Subject number	Ordinary online recognition		Adaptive online recognition		Offline recognition	
	Correct results	Accuracy rate (%)	Correct results	Accuracy rate (%)	Correct results	Accuracy rate (%)
1	64	80.00	64	80.00	65	81.25
2	61	76.25	65	81.25	66	82.50
3	59	73.75	64	80.00	66	82.50
4	60	75.00	64	80.00	65	81.25
5	59	73.75	66	82.50	67	83.75
6	60	75.00	64	80.00	65	81.25
Average		75.62		80.63		82.08

Figure 7.11 is an overall running status of the whole system showing the event mark value and the result on condition of adaptive thread scheduling, in which the bars indicate the actual event mark values and the curves indicate the recognition results. The whole system runs stably and keeps relatively short delay time from beginning to end.

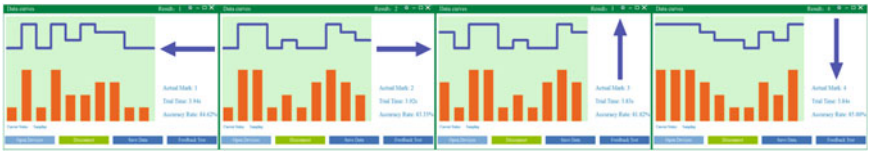

Fig. 7.11 Running status of our system showing the event mark value and the result

7.4 Conclusion

Accelerated rendering and fast reconstruction proposed by this paper is an effective way to improve the rapidity and stability of online real-time BCI system. Through the mechanism of thread concurrency, each module can make full use of the system resources; through accelerated rendering, the presentation of raw EEG data can be more fluent; and through fast reconstruction, effective EEG data unit can be analyzed timely. What is more, thread scheduling based on adaptive one-sided fuzzy inference in the process of reconstruction can enhance the robustness of the whole system for responding to the complicated data streams on condition of different sampling rate. As the statistical results show, integrative design scheme decreases the average delay time of one single trial, and improve the ITRs.

Acknowledgments Fund Project: The National Natural Science Fund (NO.60841004, 60971000, 61172152).

References

1. Allison BZ, Brunner C, Altstätter C et al (2012) A hybrid ERD/SSVEP BCI for continuous simultaneous two dimensional cursor control. J Neurosci Methods 209(2):209–307
2. Geng T, Gan JQ, Hu H (2010) A self-paced online BCI for mobile robot control. Int J Adv Mechatron Syst 2(1–2):28–35
3. Liu T, Yang P, Peng X, Huang Y, Yao D (2009) Real-time brain-computer interface system based on motor imagery. J Electron Sci Technol Chin 7(1)
4. Yu X (2012) Real time brain-computer interface based on alpha rhythms in electroencephalography. J Chongqing Univ Technol (Nat Sci) 26(7):89–93, 2012
5. Henderson A (2010) A design for a middleware communications layer between an industrial robotic arm and the BCI2000 software package. In: Proceedings of Florida conference on recent advances in robotics
6. Chin Z, Ang K, Wang C, Guan C, Zhang H (2009) Multi-class filter bank common spatial pattern for four-class motor imagery BCI. In: Engineering in medicine and biology society, annual international conference of the IEEE, vol 1, pp 571–574
7. Yuan P, Gao X, Allison B et al (2013) A study of the existing problems of estimating the information transfer rate in online brain–computer interfaces. J Neural Eng 10(2) (Article ID 026014)
8. Leite D, Ballini R, Costa P, Gomide F (2012) Evolving fuzzy granular modeling from nonstationary fuzzy data streams. Evolving Syst 3(2):65–79
9. Katz G, Peled D (2008) Genetic programming and model checking: synthesizing new mutual exclusion algorithms, automated technology for verification and analysis. Springer, Berlin, pp 33–47
10. Han L, Kong Q, Yang F, Li W (2012) Visualization of multi-beam bathymetric data based on GDI. Sci Surveying Mapp 4:051
11. Li Q, Hai T (2010) The study on GDI/GDI + rendering function defects and how to avoid them. In: 2010 2nd international conference on Information engineering and computer science. IEEE, pp 1–5
12. Kutter O, Shams R, Navab N (2009) Visualization and GPU-accelerated simulation of medical ultrasound from CT images. Comput Methods Programs Biomed 94(3):250–266

13. Long Z, Liang X, Yang L (2010) Some approximation properties of adaptive fuzzy systems with variable universe of discourse. Inf Sci 180(16):2991–3005
14. Gebhart M, Johnson DR, Tarjan D et al (2012) A hierarchical thread scheduler and register file for energy-efficient throughput processors. ACM Trans Comput Syst 30(2):8
15. Zhuravlev S, Blagodurov S, Fedorova A (2010) Addressing shared resource contention in multicore processors via scheduling. ACM SIGARCH Comput Archit News 38(1):129–142
16. Carribault P, Pérache M, Jourdren H (2011) Thread-local storage extension to support thread-based MPI/OpenMP applications, OpenMP in the Petascale Era. Springer, Berlin pp 80–93