ELSEVIER

CrossMark

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 28 (2014) 54 – 61

Procedia
Computer Science

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

Experimental Trials Based on a Neocortex-Based Adaptive System Pattern

Brian J. Phillips[a]*, Mark Blackburn[a]

[a]*Stevens Institute of Technology, Castle Point on Hudson, Hoboken NJ,03070*

Abstract

This paper proposes a general design pattern for building adaptive systems. The Neocortex Adaptive System Pattern (NASP) architecture is an adaptive decision-making architecture. It is derived from the physical architecture observed within the neocortex of a primate brain. This architectural pattern is used as a basis to provide necessary functions to adaptive systems, allowing different adaptive system components with different methodologies and techniques to coexist and cooperate within a single system. Properties of the NASP are illustrated using an agent-based simulation experiment framework composed of simulated tank vs. tank game. This study supplies experimental results that compare adaptive decisions based on accuracy and timeliness. It shows that a more accurate decision may in fact be the less optimal one due to time constraints. The experimentation results suggest that multi-system adaptation can increase system performance, and learned information can identify time frames when an adaptation can increase system performance. The practice of designing and building agent based systems shares many principles and approaches with the NASP. An agent-based architecture has a common environment that is utilized to share the state of the system with member agents. It contains autonomous entities that communicate with each other in order to perform their designed functions. A unique contribution of the NASP approach over other research is to add the ability for different agents to create alternative courses of action and controls such as rule-based, neural, or Bayesian that are used to choose from those alternatives based on their latest information. While counter intuitive, the findings suggest that increased

* Corresponding author. Tel.: 1-703-459-8864; fax: +0-000-000-0000 .
E-mail address: bphilli1@stevens.edu

1877-0509 © 2014 The Authors. Published by Elsevier B.V. Open access under CC BY-NC-ND license.
Selection and peer-review under responsibility of the University of Southern California.
doi:10.1016/j.procs.2014.03.008

performance in this combatant domain suggest that earlier adaptations, using less information, improve the performance of the adaptive system. The paper provides a literature review of relevant neuroscience literature that describes the parallels between the architecture of the neocortex and NASP. The paper discusses the simulation experiments and associated results that illustrate how tradeoffs between information completeness and timeliness affect system performance within a NASP-based system.

Keywords: neuroscience;neocortex;adaptive;systems;design;pattern;

1. Introduction

Examples of successful adaptive systems exist throughout nature. They are the very building blocks of nature itself. Biological adaptive systems are found in microbial single cell organisms, plant life, and larger animals. Adaptive properties extend beyond the individual agents within nature. Complex adaptive systems are found in nature by examining collections of organisms as populations of an ecosystem.

Previous work has identified a design pattern based on the mammalian neocortex called the Neocortex Adaptive System Pattern (NASP)[1]. This previous work reviewed key literature from the fields of neuroscience and cognitive science. It presented a list of properties that a general adaptive systems pattern would possess in order to conform to the observations found in the literature sources. It presented an architectural model that conforms to those properties, and posited useful applications of that architecture to real world adaptive systems.

This paper describes a proposed NASP approach that defines an architectural pattern for adaptive systems based upon scientific observations of the mammalian neocortex. Experiments were performed to demonstrate the utility of the NASP to design adaptive systems using 11 8000 match tournaments of two competing agents. One agent is a Bayesian learning-agent that uses a NASP approach to choose its behaviors based on observations of the opposing agent. The other opposing agent uses a variety of rule-based approaches to derive and execute a plan to win matches. The Bayesian agent uses learning methods to identify its opponent, and choose one of 80 behaviors to combat it. This experiment set tested a hypothesis that the two-stage adaptive system build from a NASP basis outperforms single stage systems.

This research uses the RRobots[2] environment to create a series of matches where one simulated tank agent competes with another within a simulated tank-combat game. RRobots is an open-source environment, written in Ruby and developed by Simon Kröger for simulating virtual robot tank combat. The environment allows robot tank agents to move within a square arena while shooting at each other. The objective for these robots is survivability. The match ends when all robots, except for the winner, have been depleted of resources normally related to receiving too much damage. Robots can dynamically change offensive and defensive strategies. The enhanced environment uses Ruby on Rails to integrate data collection and measurement. New robot capabilities along with operational and environmental measures help provide an understanding on how self-adaptation can better leverage resources to increase survivability.

2. RRobots as a Research Tool

This experimental framework provides a simulation-based environment that allows the robot agents to be aware of their context both within the environment, and in relationship to other robots. A robot agent within the simulation can self-configure to adapt to specific opponent capabilities. It can observe opposing agents and change its behavior according to what it perceives.

The environment is continuously enhanced by the researchers to better investigate different levels of autonomy, such as those specified by Salehie and Tahvildari[3], as shown in Fig. 1. Self-awareness and context-awareness provide a basis for all autonomous behaviors. The agents use this information to self-configure, optimize, heal, or adopt protective behaviors. The combination of each of these abilities forms the basis for self-adaptive behaviors. The concept of health is designed into the RRobots simulation and directly relates to the resources such as energy or firing power, available to the robots. The current adaptation strategies investigate how adaptive agents can discover strategies to self-protect and avoid damage, or to become more aggressive in order to quickly defeat an opponent. The primary concern of this research is self-adaptation of behaviors that are dynamic and extend beyond parametric adaptation.

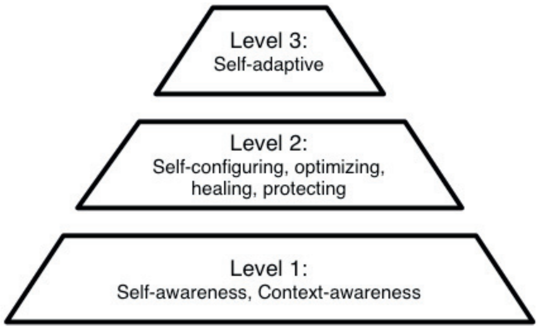
Level 3:
Self-adaptive
Level 2:
Self-configuring, optimizing, healing, protecting
Level 1:
Self-awareness, Context-awareness

Fig. 1 Levels of Autonomy

2.1. Controls and Measurements

The overall control structure follows the typical feedback processes with four key activities: collect, analyze, *decide*, and *act*[4], sometimes discussed in terms of *monitor-analyze-plan-execute*[5]. This section describes base behaviors and the current set of rules that govern the robot matches. These relate primarily to the *act* function of the control loop. The *collect* function of the control loop is supported within the environment referred to as "RailsRobots." RailsRobots is a web server that contains tools for designing experiments. Experiments are composed of a number of RRobots matches. The server collects performance data concurrently during execution. Different numbers of matches can be set up between any of the robots to be executed through the server. For example, Robot1 can oppose Robot2 one hundred times while the server collects statistics such as number of wins, energy remaining, and shots fired.

All experiments operate under the following rules. Matches last a given number of processing steps, currently set to 20,000. The arena is set to be 800 pixels by 800 pixels. Robots are 16 pixels long by 16 pixels wide. Each robot agent begins the match from a random location with 100 health points. Robots shoot using firepower settings from 0.1 to 3.0, which directly relate to the amount of damage delivered. Damage delivered reduces health of the opponent. The firing rate is controlled to be inversely proportional to the firing power setting used; this is controlled in game through the gun heat variable. The behaviors use a set of methods analogous to a domain-specific language for robot agent commands.

The unique contribution of this paper is to demonstrate an architectural design pattern that unifies different decision-making methodologies within a single adaptive system. It uses a simulation environment to show an example of an implemented NASP-based adaptive system. It presents data collected from executing this adaptive system within a simulated combat between the adapting system and opposing agents. This effort demonstrates the tiered effects of decision timing and describes tradeoffs between making quicker vs. better decisions. It reports data and findings based on thousands of matches between opposing agents.

3. NASP Design Pattern

The NASP design pattern seeks to provide a decision framework for adaptive systems based on the observed physical architecture of the neocortex. A survey of relevant literature has found a set of architectural properties that the neocortex exhibits. These properties form the basis of an adaptive system design pattern that permits different adaptation approaches and algorithms to exist within the same system without conflict.

The architectural properties are listed below, along with relevant citations.

- Brain architectures are constructed using sets, called columns[4], of hierarchical pattern matching neurons. Each column is composed of child-nodes, named sub-columns by Hubel, which perform pattern matching. Higher-

level columns use an evaluation to choose from the results of sub-columns to inform the top-level cognitive process. This reflects a hierarchy of decision points within architecture. Top-level decision points select results based on the outputs of lower level decision points.
- Columns and sub-columns communicate between each other using networked signals. This allows one pattern matching function within the neocortex to affect another, even when they are not directly related. Signal channels exist between different columns and sub-columns that process input patterns. They influence other sections of their column and external columns. This creates a number of communication channels between different independent hierarchies. Each hierarchy of parent columns and child sub-columns will process a distinct set of signals[7].
- Brain architectures are sensitive over different time scales. Some brain systems are optimized to store short-term patterns; others collect patterns over a longer period of time[8]. The physical architecture of these two-systems is very similar, but pattern matching is different between them[9].
- When a brain selects a pattern to use, it uses a two-system method. It may perform a quick instinctive selection with minimal information. It may also use a more determined method that uses additional time and requires additional information[9].
- System builders may utilize Bayesian statistical analysis to model cognitive selection processes[10]. The neocortex and its associated neurological structure are vastly more complicated than a Bayesian solution or a computing system. The operation of that same logic can be modeled using a scalable Bayesian solution.
- A system can use Monte Carlo methods[11] to create candidate behavioral choices. It can solve many possible patterns, guesses, or solutions and select best fitting answers from those candidate solutions to arrive at a best option. A brain architecture that contains a greater number of solutions seems to be present in higher performing organisms. In essence, the neocortex pre-computes many courses of action, then down selects from those candidates when faced with situations where those courses of action are applicable.
- There are feedback loops in the cognitive process. Feedback loops act to reinforce or weaken selected options[12]. It is even theorized that consciousness is a result of the constant reassessment between observed reality and mentally constructed forecasts.

3.1. NASP-Based Agent vs. Rule-based Agent example

This paper documents an example NASP based adaptive system that adapts based on a fixed set of behaviors to oppose a non-adaptive agent within a simulation-based tank versus a tank game environment. Each match within the simulation features a single adaptive tank agent, and a single rule-based tank agent. There are 80 different types of rule-based tank agents. Each type of tank agent has a probability of defeating every other type of tank agent. The rule-based tank has a set of parametric behaviors that it uses to combat the adaptive tank. These rules generated using a combinatory exploration of different movement, firing, and sensing strategies. Both agent types use the same set of behaviors to combat each other. The adaptive agent under study has a method to change its behavior selection at run time. This change capability is the critical feature under study.

A NASP-based system is organized into a hierarchy of behavior selection nodes. The NASP system uses a selection mechanism to choose between candidate behaviors. Candidate behaviors can exist at many levels within the hierarchy. Behavior selection nodes within the hierarchy share information between each other across multiple levels within that hierarchy by using a shared information bus. In this example, the tank agent system uses multiple Bayesian classifiers to determine its opponent's identity. It also uses a layered selection method to choose an appropriate behavior when faced with an opponent.

The adapting tank agent, shown in Fig. 2, is constructed using a NASP-based hierarchy of behavior selectors and opponent classifiers. An opponent classifier is simply another behavior selector that is specialized in identifying the opponent agent. The top layer of this hierarchy is composed of this opponent classifier and a behavior selection node. The opponent classifier system is composed of a fast opponent classifier that uses minimal information and minimal time to determine the type of its opponent. The opponent classifier also has a deliberate opponent classifier that requires more information and time to classify its opponent tank agent. In practice, the deliberate opponent classifier identifies an opponent with a high degree of accuracy; the fast opponent classifier identifies the opponent quicker, with a lower degree of accuracy.

This difference in timing is a key finding of this research effort. Results indicate that early detection with shorter learning cycles and lower accuracy outperforms accurate but less-timely identifications in many cases. The NASP-based system executes both of these identification strategies in order to arrive at an initial low-confidence identity, and then improve the opponent identification as more information arrives. This two-system method results in better overall performance than either system alone. The behavior selection node contains a set of behaviors that mirror every behavior possessed by any of the 80 types of opposing rule-based agents. This node also contains a Bayesian-based probability correlation that associates opponent identities and selected behaviors. This association is a probability of one behavior defeating a specific rule-based tank agent. The behavior selection node must assume an opponent type in order to select a behavior to adopt. It has no capability to determine this information on its own.

The NASP design pattern specifies a shared information bus. This acts as an information conduit between the opponent classification node and the behavior selection node. The opponent classification node assesses information provided by child nodes to determine an assumption of opponent type based on the binning values. This information is placed into the shared information bus. The behavior selection node uses this information to inform its choice of behavior. The behavior selection node contains a reference to each of the 80 behaviors exhibited by each of the opposing tank agent types. The adaptive agent has the same behaviors as the rule-based agents. It has the ability to choose a behavior to fit a specific opponent during run time. A high-level view of this system is shown in Fig. 2.

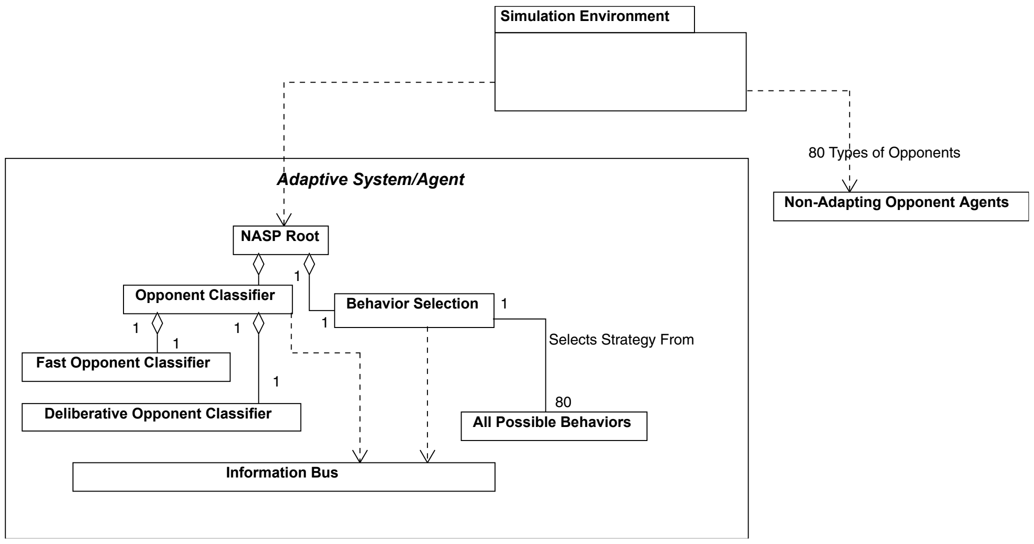

Fig. 2 NASP Behavioural Selections

4. Experiments using the NASP Design Pattern

The RRobots simulation provides a useful research tool for the investigation of the NASP design pattern. This tool was used to simulate a match between an adaptive NASP-based agent, and a more traditional rule-based agent. Each of the 80 types of rule-based agents is matched against a NASP-adaptive agent 100 times for a total of 8000 simulation executions within a tournament of matches. The NASP had an information bus that contained training data in the form of a statistical correlation of adaptive behavior performance when faced with a distinct opponent.

The adaptive agent demonstrated the ability to win these matches more than 93% of the time. This win was dependent on two factors. It depended on the timeliness and accuracy of the adaptive systems ability to identify its opponent and then select an appropriate behavior. When behavior selection is random, the agent wins only 25% of the matches.

These results extend previous research that investigated adaptive agents that could instantly recognize their opponent and change their behavior accordingly. Adaptive agents waited for a fixed time delay prior to changing their behavior. The change was based on the ability to recognize an opponent, and associate a set of past behaviors with the current opponent. The adaptive agent possessed an error rate that simulated incorrectly identifying an opponent. The overall win rate of the adaptive agent in the previous experiments was 84%.

This study adds to this effort by building a two-stage opponent recognition system. The two stages are points in time associated with a probability of identifying the opponent. The adaptive agent performs an initial adaptation. If the adapting agent is losing its match, then it adapts again using the extra information that it had accumulated between those two adaptations. Each stage has a probability of erroneously identifying an opponent formed from its training data.

4.1. Simulation Actions

This paper adds the classification and timeliness capabilities into the NASP-based adaptive agent. Each of the matches between an adaptive agent and a rule-based agent results in a series of step-by-step movement, aiming, and firing decisions. The adaptive system continuously analyzes information collected about its opponent. At any point, the adaptive system can evaluate its latest set of data and choose to change its behavior accordingly. As more information accumulates over time, the accuracy of the classification process improves in a decreasingly monotonic function. The collection of more information was not found to be a perfect predictor of system performance though. The additional information created a larger sample size for the binning algorithm. As the binning included more data, the data began to represent a more uniform and less situational situation.

One-system and two-system adaptive agents are compared against each other using this same data. One-system agents simply have one adaptation time, while two-system agents have two adaptation times.

The NASP adaptive system evaluates the perceived state of its opponent every time-step. If the adaptive system simply used every piece of information it had, then eventually enough information accumulates so that its accuracy is decreased due to the conflicts among different information sets and their correlated dependencies. The NASP adaptive system under study avoided this problem by utilizing a distinct segment of time, called a bin size, which it would evaluate much like a histogram data set. All segments are evaluated independently of the others. Experimental data shows that smaller windows can result in a large difference in system performance. Fig. 3 shows the difference between a 100 and 500 time-step bin sizes within the same 80000-match tournament.

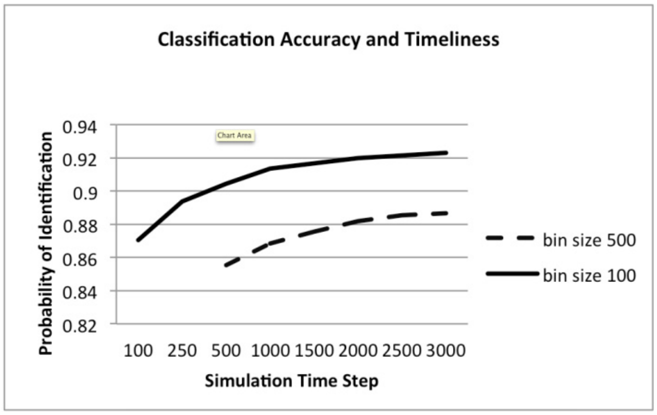

Fig. 3 Effect of timeliness on accuracy of opponent classification

Two bin sizes are shown in Fig. 3. The 100-step bin size is composed of selecting only information from the last 100 time-steps, and re-evaluating that selection every 100 steps as well. The 500-step bin size executes the same evaluation only using a 500-step time interval. This indicates that additional statistical information over time may interfere with an adaptive agent's ability to perceive the difference between opponent types.

After the bin based identity information was derived, then the data from the second tournament is used to measure that accuracy of the selection process. A post processor reads the second set of test data. The post processor derives its vector quantization values and uses this to randomly select a behavior from the training data after each record. The post processor assumes a proportional probability among all candidate identities. The post processor randomly selects an identity and compares it to the actual identity for every record in the testing data set. The accuracy of the prediction is computed for each rule-based agent type by dividing the total number of correct classifications by the total number of all classifications. Figure 3 shows average of all classification probabilities for every agent type.

An identification training set is computed by executing two independent tournaments. The first tournament sorts into lists based on their time value. The specific time-correlated list is referred to as its bin position. Adaptive systems in each tournament evaluate their perception of their opponent using either 500 or 100 time-step intervals. At the end of each tournament, the simulation writes a file containing one line of data for each time-step, for each match, for each agent. The collected bin data is used to create a list of every opponent data value organized a using vector quantization method referred to as *binning*. The research correlated each of these values to an actual opponent identity, and the probability of selecting any of the possible opponents based on that vector-quantized data. The research uses this data as training data, deriving how the system selects its behaviors at run time based on available perception information.

Each of the non-adapting agents possesses a unique strategy that defines how they move, aim, fire, and expend energy. These bins represent the adapting agent's perception of those factors. The adapting agent does not have perfect perception. It relies on assigning widely scoped bin values to behaviors, and later, inspecting those same observed behaviors for close matches. The closest matching bin is assumed to be the identity of the non adapting opponent.

4.2. Tournament Results

The results of this tournament are documented in Table 1. The research used the information shown in Fig. 3 to create a two-system opponent classification process. Each of these systems acts independently over time. These two systems are the *fast* and the *deliberate* opponent classification systems. The accuracy of that process is dependent on sample sizes and data content, shown in Fig. 2. The NASP Fast Opponent Classifier used only the adaptive systems initial perception data within the first time span. After that time-step, the adaptive system changed behaviors to compete with its opponent. The NASP Deliberative classifier agent activates further along in time. It uses information from previous time bins to update the adaptive agent's behavior process.

The adaptive agents compete against rule-based agents in the same 80000-match tournament structure. This research compares the maximum entropy case where behaviors are decided randomly, against both the 100 time-step and the 500 time-step case.

The binning strategies were used from both the 100 time-step and the 500 time-step case. The RRobots tournaments were first executed using completely random adaptation within the adaptive agent. This yielded a 25.8% win rate for the adaptive agent. Rule-based agents outperformed the random adapting agent in a majority of cases. Adapting agents are compared against each other using binning strategies and adaptation times.

The first informed point of adaptation is when the first set of bins was populated. This required either 100 or 500 time-steps depending on the bin accumulation size in question. The 100 time-step bin size outperformed the 500 time-step bin size. A 100 time-step adaptation period left an additional 400 time-steps to achieve victory after the adaptation was finished. As shown in Fig. 3, the 500-step bin size did not result in a larger identification probability.

Those results changed as the time of adaptation increased. At time-step 1000, adaptation was the most effective for the 500 time-step binning mechanism. Both of the adaptation timing strategies became equivalent when the time of adaptation became 3000 time-steps, achieving victory in 55% of the matches.

The NASP based two-system approach was tested in the final two cases. The first case showed a large improvement in performance when an initial adaptation was executed at the first binning point. This adaptation occurred at either 100 or 500 time-steps. Then another adaptation was executed at a later time point. The 100 time-step binning size outperformed all other options when the second adaptation was at 1000 time-steps, and when it was at 3000 time-steps. The earlier second adaptation option of 1000 time-steps resulted in the best performance improvement overall, achieving a 93% victory rate.

Table 1. Adaptive System Performance.

Tournament	100 time-step win probability	500 time-step win probability
Adapt at time = 0	25.8%	25.8%
Adapt only at first time = 100/500	84% (at 100)	76.85% (at 500)
Adapt only at time = 1000	63.5%	69.91%
Adapt only at time=3000	55.4%	54.81%
Adapt at time 100/500 and then 3000	84%	76.94%
Adapt at time 100/500 and then 1000	**93.4%**	**77%**

These results indicate an advantage to using the two-system adaptation method. The adaptation strategy was informed by thousands of previous observations collected into a statistical representation of enemy behaviors. The adaptation strategy relies on a Bayesian probability that correlated opponent identities with effective behaviors. The NASP pattern-based adaptive system outperformed the random case, and all of the single adaptation patterns tested.

Future Work

This study demonstrates the NASP within the context of a simulation where two agents oppose each other within a simulated game battle. The addition of multiple agents into the battle scenario would create another layer of decisions where adaptive agents must decide on behaviors, where those behaviors are influenced by an overall team strategy. The introduction of resources into the scenario would add another layer of complexity as well. Such a multi-agent resource dependent scenario would demonstrate that the NASP could be used outside of the simulation environment. The experimental results identified types of information collected during runtime. Future research intends to investigate how this type of information can support runtime verification for Bayesian-based adaptation strategies.

References

1. Phillips B, Blackburn M, Towards a Design Pattern for Adaptive Systems Inspired by the Physical Architecture of the Neocortex. Manuscript Submitted, 2013.
2. Kröger S. RRobots. http://rrobots.rubyforge.org/ (accessed Sep 2013).
3. Salehie M, Tahvildari L. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems 2009; 4(2):1-42.
4. Dobson S, Zambonelli F, Denazis S, Fernández A, Gaïti D, Gelenbe E, et al.. A survey of autonomic communications. ACM Transactions on Autonomous and Adaptive Systems 2006; 1(2): 223-259.
5. IBM Corporation. An architectural blueprint for autonomic computing. http://www-03.ibm.com/autonomic/pdfs/ AC%20Blueprint%20White %20Paper%20V7.pdf (accessed Sep 2013).
6. Hubel D. Eye, Brain, and Vision, 2 ed. New York: Scientific American Library : Distributed by W.H. Freeman; 1988.
7. Mountcastle V. The Mindful Brain: Cortical Organization and the Group-Selective Theory of Higher Brain Function. 1st ed. Cambridge: MIT Press; 1985. An Organizing Principle for Cerebral Function: The Unit Module and the Distributed System; p. 7-50.
8. Hasson U, Yang E, Vallines I, Heeger DJ, Rubin N. A Hierarchy of Temporal Receptive Windows in Human Cortex. Journal of Neuroscience 2008; 28(10): 2539-2550.
9. Kahneman D. Maps of Bounded Rationality: A Perspective on Intuitive Judgement and Choice. http://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2002/kahnemann-lecture.pdf (accessed 2013).
10. Lee TS, Mumford D. Hierarchical Bayesian inference in the Visual Cortex. Journal of the Optical Society of America A 2003; 20(7): 1434-1448.
11. Doucet A, de Freitas N, Gordon N. Statistics for Engineering and Information Science New York: Springer; 2003. An introduction to Sequential Monte Carlo Methods; p. 3-14.
12. Felleman DJ, Van Essen DC. Distributed Hierarchical Processing in the Primate Cerebral Cortex. Cerebral Cortex 1991; 1(1): 1-47.