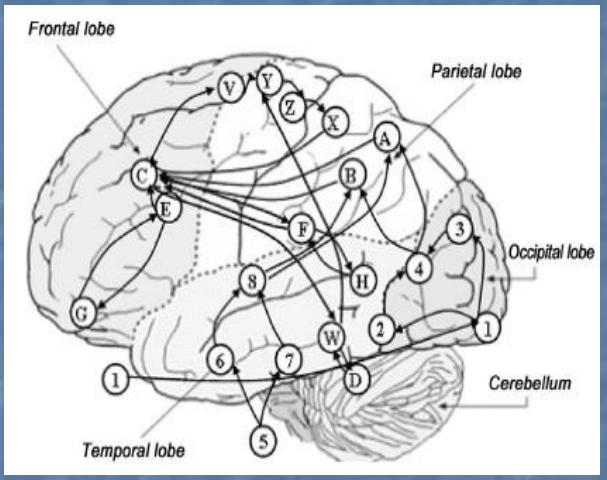
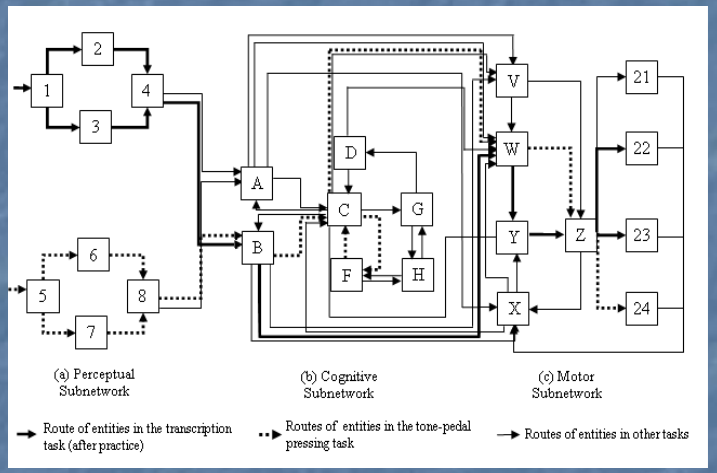
Integrative Modeling and Simulation of Human Behavior and Human-Machine Systems with Queuing Network (QN) Architecture





Queueing Network of Mental Architecture

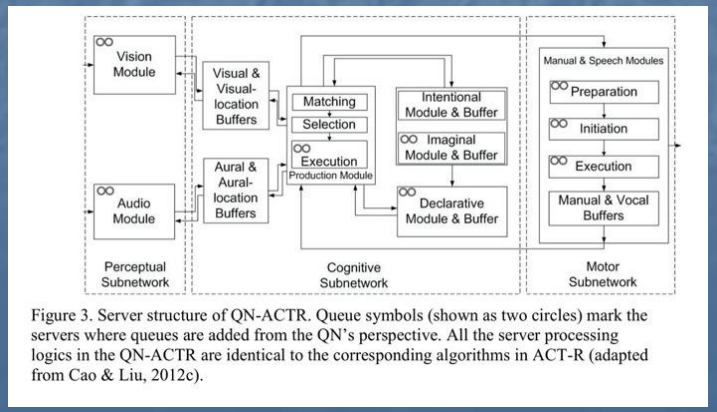


Figure 3. Server structure of QN-ACTR. Queue symbols (shown as two circles) mark the servers where queues are added from the QN's perspective. All the server processing logics in the QN-ACTR are identical to the corresponding algorithms in ACT-R (adapted from Cao & Liu, 2012c).

Queueing Network (QN)
Models of Human Behavior (MHB)
QN-MHB

1. RT: Reaction Time (QN-RT) and Mental Structure
2. RT and Accuracy (QN-RMD) (Mental Structure vs State of Mind)
3. Procedural Tasks (QN-MHP or QN-MHP-BE)
4. Complex Cognition Tasks (QN-ACTR)
5. Visual Attention tasks (QN-NSEEV) (QN-RLEM)
6. Manual or Continuous Control tasks (QN-Control: Classical/Modern)
7. Basic Body Motion tasks (QN-MTM)
8. Mind-Body System (QN-MBS)
9. Neural level (QN-Neural: Indexes and Neural Networks)
10. Nervous and Endocrine Systems (QN-NES)
11. Multi-Person Multi-Machine QN (QN-HMN)
12. Engineering Applications in various domains

Note: relations with **Task Network** Methods/Tools such as Micro Saint #, IMPRINT)

Carnegie Mellon University

Contact | Sign up | Submit Publication | Mailing List | Search

HOME ABOUT PEOPLE PUBLICATIONS & MODELS SOFTWARE WORKSHOPS LINKS

Please Note: All the ACT-R slides are from or largely based on ACT-R website

About

ACT-R is a **cognitive architecture**: a theory about how human cognition works. On the exterior, ACT-R looks like a programming language; however, its constructs reflect assumptions about human cognition. These assumptions are based on numerous facts derived from psychology experiments.

Like a programming language, ACT-R is a framework: for different tasks (e.g., Tower of Hanoi, memory for text or for list of words, language comprehension, communication, aircraft controlling), researchers create **models** (aka programs) that are written in ACT-R and that, beside incorporating the ACT-R's view of cognition, add their own assumptions about the particular task. These assumptions can be tested by comparing the results of the model with the results of people doing the same tasks. By "results" we mean the traditional measures of cognitive psychology:

- time to perform the task,
- accuracy in the task, and,
- (more recently) neurological data such as those obtained from FMRI.

One important feature of ACT-R that distinguishes it from other theories in the field is that it allows researchers to collect quantitative measures that can be directly compared with the quantitative measures obtained from human participants.

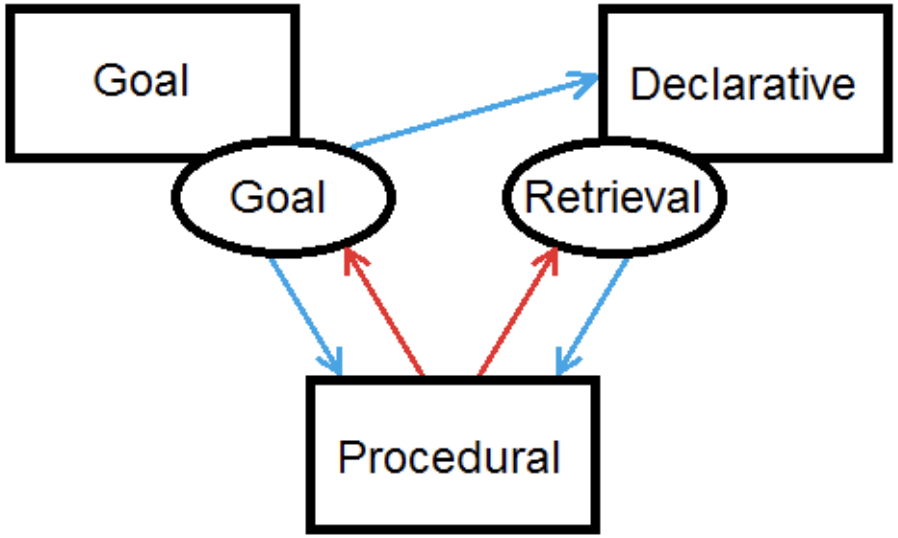
Related Info

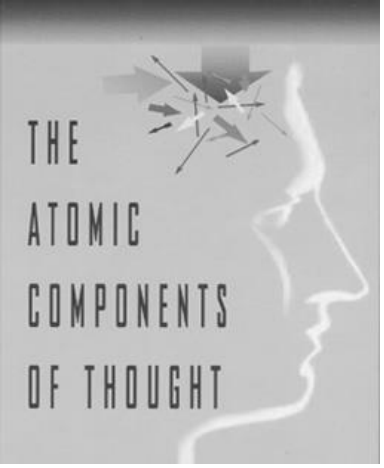
PUBLICATIONS

"An Integrated Theory of the Mind" (2004).

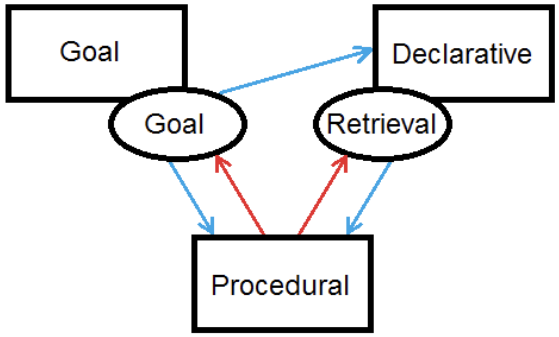
"How Can the Human Mind Occur in the Physical Universe?" (2007).

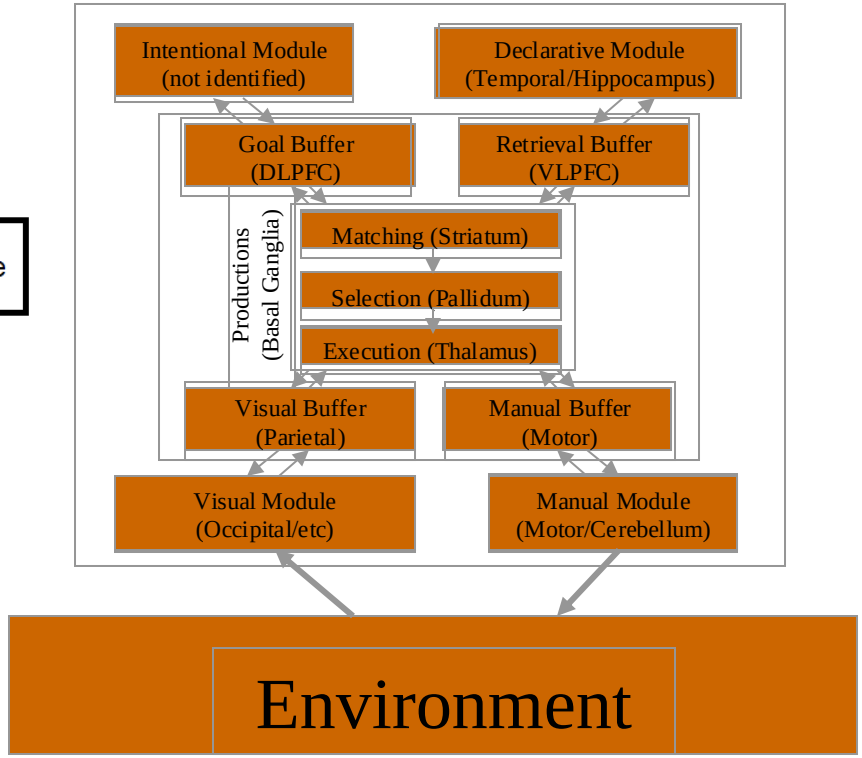
Core Structure of ACT-R

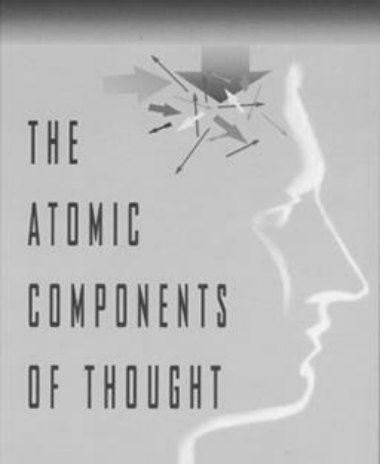



THE
ATOMIC
COMPONENTS
OF THOUGHT

ACT-R


Goal
Declarative
Goal
Retrieval
Procedural

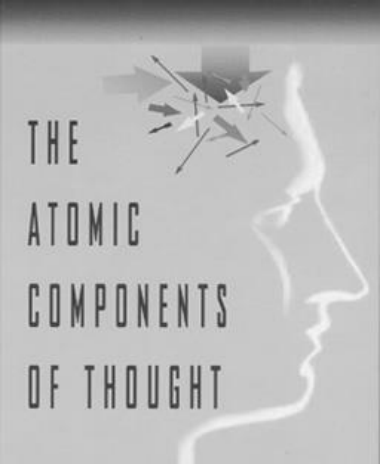

Intentional Module
(not identified)
Declarative Module
(Temporal/Hippocampus)
Goal Buffer
(DLPFC)
Retrieval Buffer
(VLPFC)
Productions
(Basal Ganglia)
Matching (Striatum)
Selection (Pallidum)
Execution (Thalamus)
Visual Buffer
(Parietal)
Manual Buffer
(Motor)
Visual Module
(Occipital/etc)
Manual Module
(Motor/Cerebellum)
Environment



Chunks: Example

(CHUNK-TYPE NAME SLOT1 SLOT2 SLOTN)

(FACT3+4
  isa ADDITION-FACT
  ADDEND1 THREE
  ADDEND2 FOUR
  SUM SEVEN)



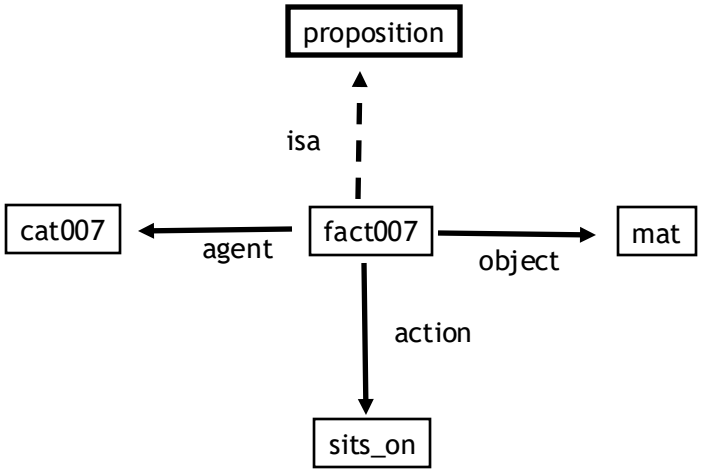
Chunks: Exercise I

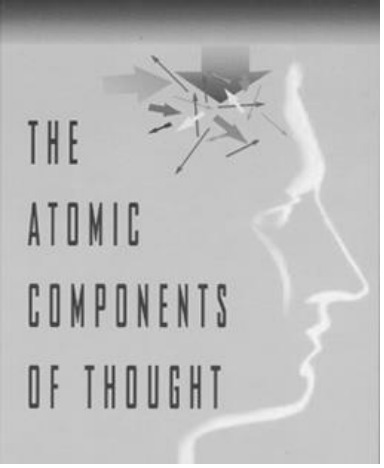
Fact: **The cat sits on the mat.**

Encoding:

(Chunk-Type proposition agent action object)

```
(Add-DM
   (fact007
      isa proposition
      agent cat007
      action sits_on
      object mat)
  )
```





Chunks: Exercise II

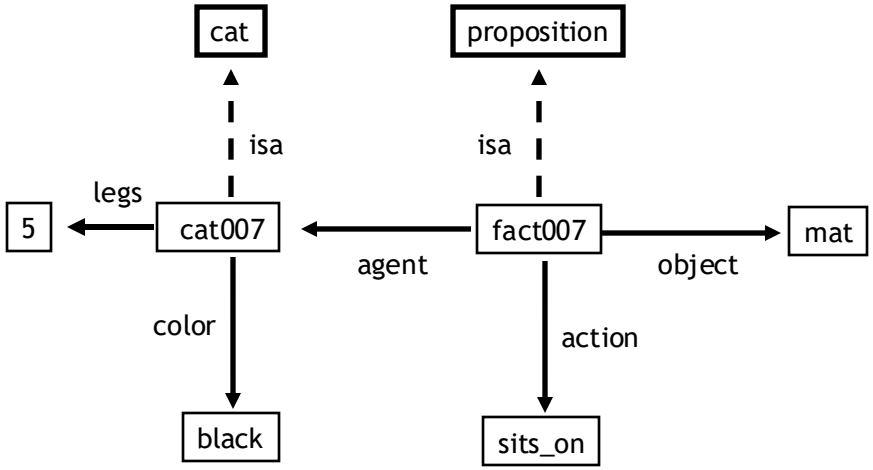
Fact — The black cat with 5 legs sits on the mat.

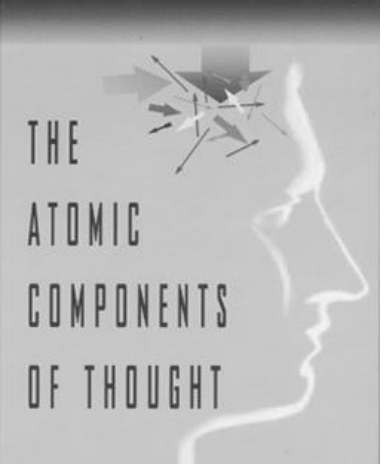
Chunks

(Chunk-Type proposition agent action object)
(Chunk-Type cat legs color)

(Add-DM
(fact007 isa proposition
agent cat007
action sits_on
object mat)

(cat007 isa cat
legs 5
color black)
)





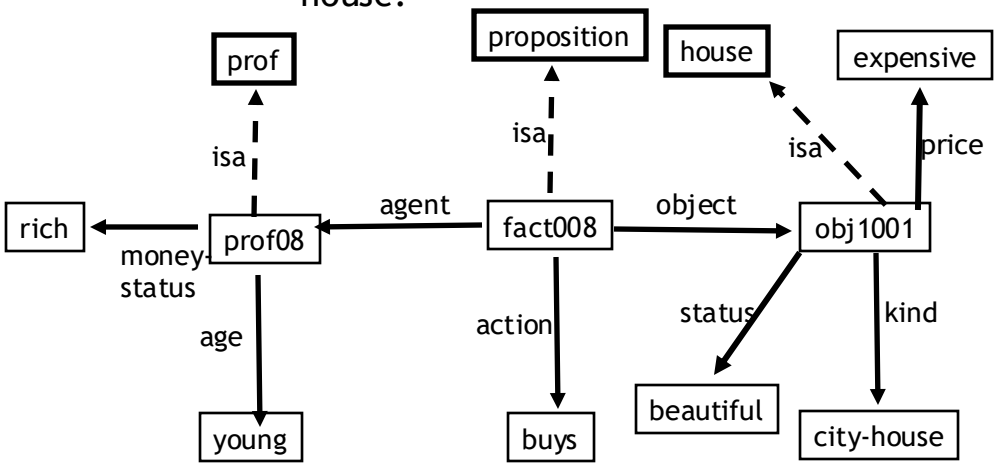
Chunks: Exercise III

```
(Chunk-Type proposition agent action object)
(Chunk-Type prof money-status age)
(Chunk-Type house kind price status)
```

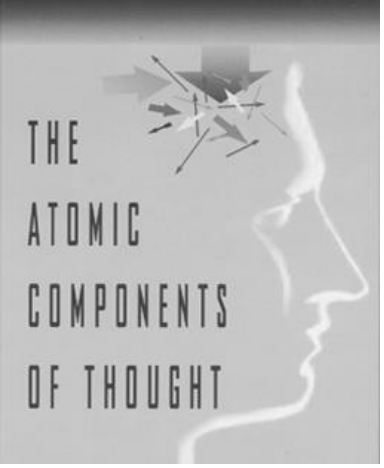
Fact

The rich young professor buys a beautiful and expensive city house.

Chunk




```
(Add-DM
   (fact008 isa proposition
            agent prof08
            action buys
            object house1001
   )
   (prof08 isa prof
            money-status rich
            age young
   )
   (obj1001 isa house
            kind city-house
            price expensive
            status beautiful
   )
)
```



A Production is

1. The greatest idea in cognitive science.

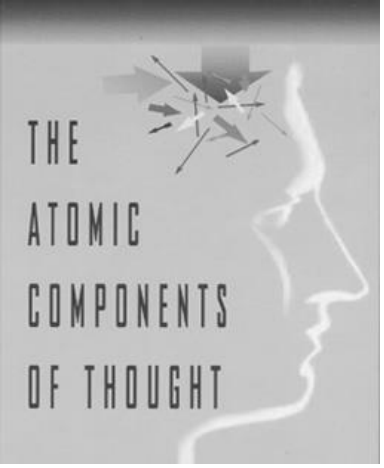
2. The least appreciated construct in cognitive science.

3. A 50 millisecond step of cognition.

4. The source of the serial bottleneck in otherwise parallel system.

5. A condition-action data structure with “variables”.

6. A formal specification of the flow of information from cortex to basal ganglia and back again.



Productions

Key Properties →

- modularity
- abstraction
- goal/buffer factoring
- conditional asymmetry

Structure of productions

(p name

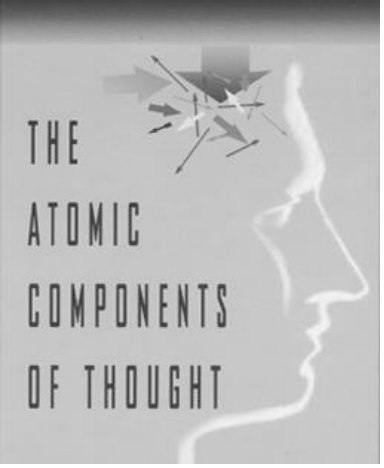
Specification of
Buffer Tests

==>

Specification of
Buffer Transformations

)

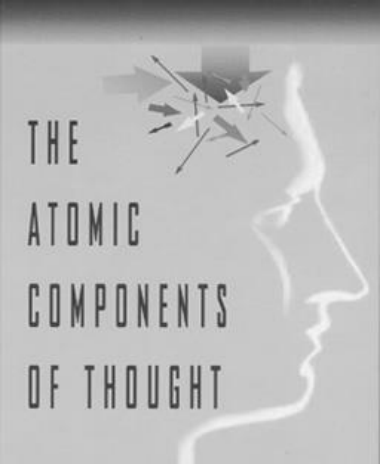
"modification" or "module request"



Attending to a Word in Two Productions

```
(P find-next-word
  =goal>                                          ← no word currently being processed.
     ISA          comprehend-sentence
     word         nil
==>
  +visual-location>                               ← find left-most unattended location
     ISA          visual-location
     screen-x     lowest
     attended     nil
  =goal>                                          ← update state
     word         looking
)

(P attend-next-word
  =goal>                                          ← looking for a word
     ISA          comprehend-sentence
     word         looking
  =visual-location>                               ← visual location has been identified
     ISA          visual-location
==>
  =goal>                                          ← update state
     word         attending
  +visual>                                        ← attend to object in that location
     ISA          visual-object
     screen-pos   =visual-location
)
```



Subsymbolic Level

The subsymbolic level reflects an analytic characterization of connectionist computations. These computations have been implemented in ACT-RN (Lebiere & Anderson, 1993) but this is not a practical modeling system.

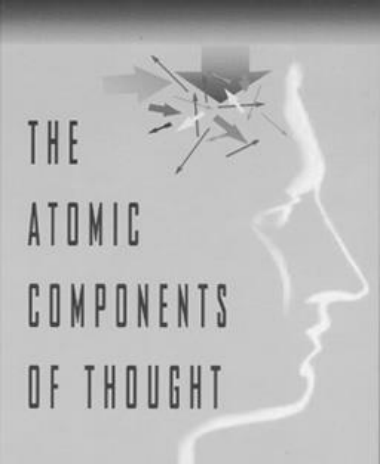
1. Production Utilities are responsible for determining which productions get selected when there is a conflict.

2. Production Utilities have been considerably simplified in ACT-R 5.0 over ACT-R 4.0.

3. Chunk Activations are responsible for determining which (if any chunks) get retrieved and how long it takes to retrieve them.

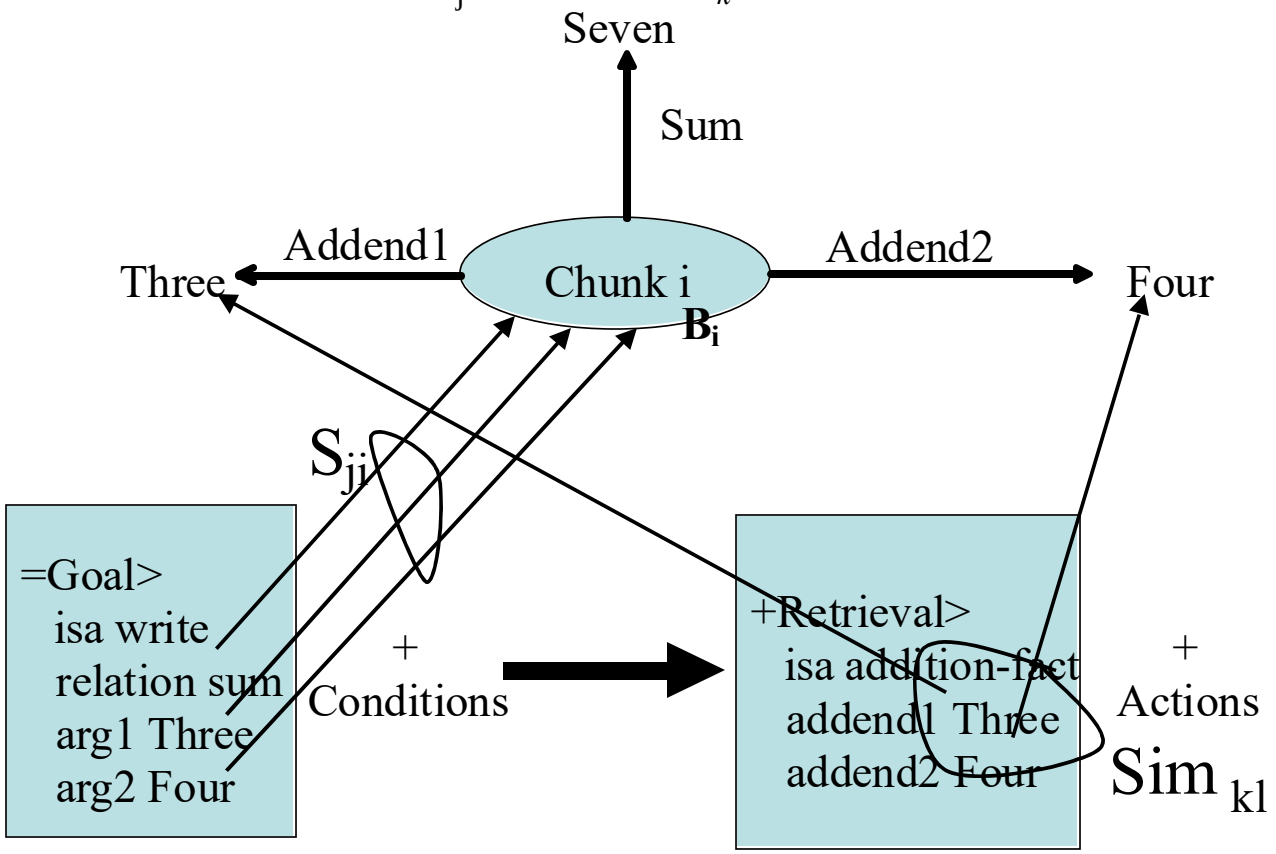
4. Chunk Activations have been simplified in ACT-R 5.0 and a major step has been taken towards the goal of parameter-free predictions by fixing a number of the parameters.

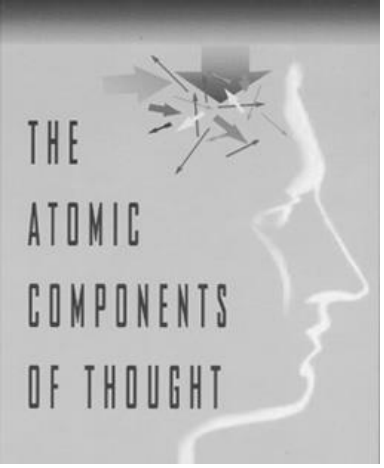
As with the symbolic level, the subsymbolic level is not a static level, but is changing in the light of experience. Subsymbolic learning allows the system to adapt to the statistical structure of the environment.



Activation

$$A_i = B_i + \sum_j W_j \times S_{ji} + \sum_k MP_k \times Sim_{kl} + N(0, s)$$





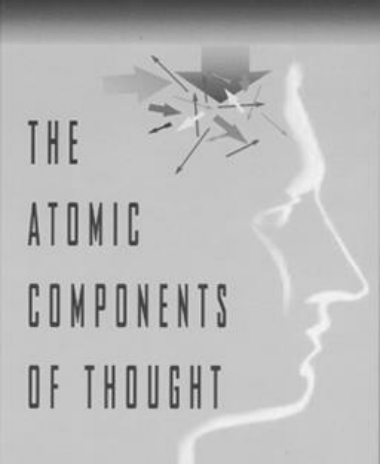
Chunk Activation

activation = base activation + (source activation * associative strength) + (mismatch penalty * similarity value) + noise

$$A_i = B_i + \sum_j W_j \times S_{ji} + \sum_k MP_k \times Sim_{kl} + N(0, s)$$

Activation makes chunks available to the degree that past experiences indicate that they will be useful at the particular moment:

- Base-level: general past usefulness
- Associative Activation: relevance to the general context
- Matching Penalty: relevance to the specific match required
- Noise: stochastic is useful to avoid getting stuck in local minima



Activation, Latency and Probability

- Retrieval time for a chunk is a negative exponential function of its activation:

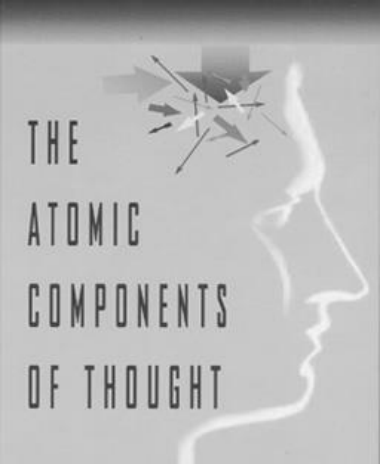
$$Time_i = F \times e^{-A_i}$$

- Probability of retrieval of a chunk follows the Boltzmann (softmax) distribution:

$$t = \sqrt{2} \times s = \frac{\sqrt{6} \times s}{p}$$

$$P_i = \frac{e^{A_i / t}}{\sum_j e^{A_j / t}}$$

- The chunk with the highest activation is retrieved provided that it reaches the retrieval threshold τ
- For purposes of latency and probability, the threshold can be considered as a virtual chunk



Base-level Activation

activation = base activation

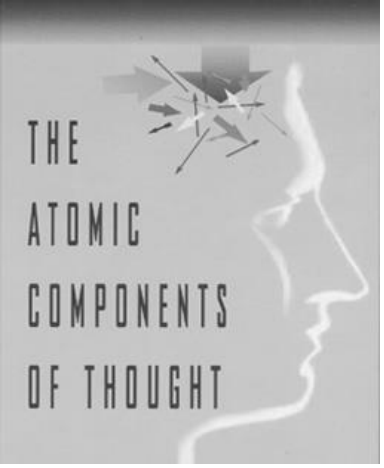
$$A_i = B_i$$

The base level activation B_i of chunk C_i reflects a context-independent estimation of how likely C_i is to match a production, i.e. B_i is an estimate of the log odds that C_i will be used.

Two factors determine B_i:

- frequency of using C_i
- recency with which C_i was used

$$B_i = \ln\left(\frac{P(C_i)}{P(\overline{C_i})}\right)$$



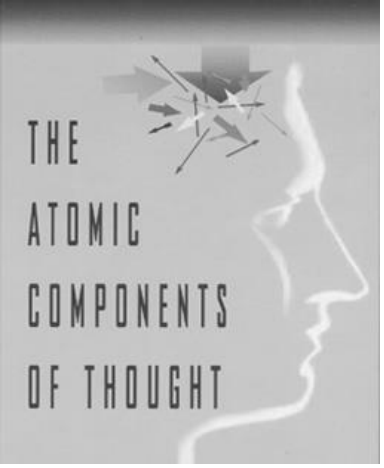
Source Activation

$$+ \left(\text{source activation} * \text{associative strength} \right)$$

$$+ \sum_j W_j * S_{ji}$$

The source activations W_j reflect the amount of *attention* given to elements, i.e. fillers, of the current goal. ACT-R assumes a *fixed capacity* for source activation

$W = \Sigma W_j$ reflects an individual difference parameter.



Associative Strengths

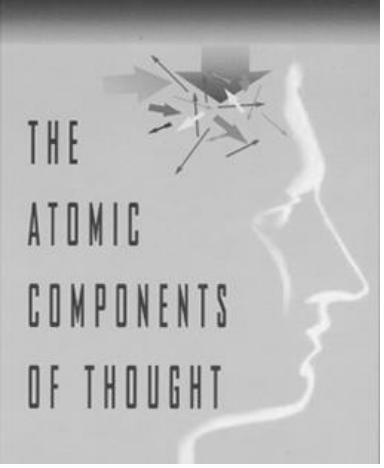
$$+ \left(\text{source activation} * \text{associative strength} \right)$$

$$+ \sum W_j * S_{ji}$$

The association strength S_{ji} between chunks C_j and C_i is a measure of how often C_i was needed (retrieved) when C_j was element of the goal, i.e. S_{ji} estimates the log likelihood ratio of C_j being a source of activation if C_i was retrieved.

$$S_{ji} = \ln \left(\frac{P(N_i|C_j)}{P(N_i)} \right)$$

$$= S - \ln(P(N_i|C_j))$$



Production Utility

Making Choices: Conflict Resolution

Expected Gain = E = PG-C

P is expected probability of success
G is value of goal
C is expected cost

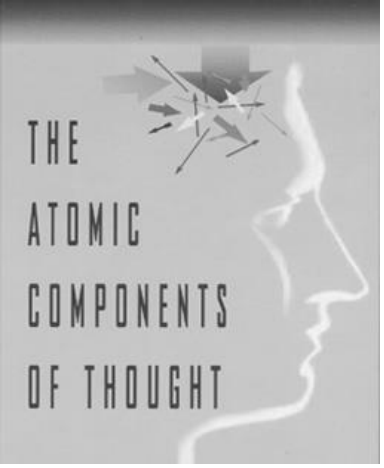
Probability of choosing i = $\frac{e^{E_i/t}}{\sum_j e^{E_j/t}}$

t reflects noise in evaluation and is like temperature in the Bolztman equation

$$P = \frac{\text{Successes}}{\text{Successes} + \text{Failures}}$$

Successes = a + m
Failures = b + n

α is prior successes
m is experienced successes
β is prior failures
n is experienced failures

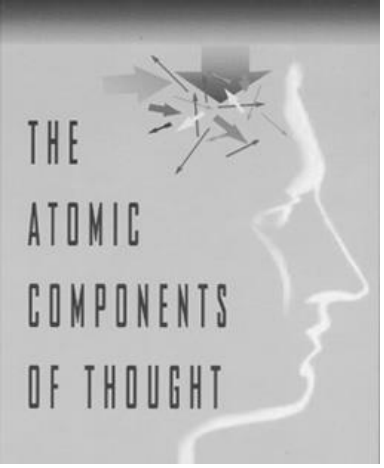


Production Compilation: The Basic Idea

```
(p read-stimulus
   =goal>
      isa goal
      step attending
      state test
   =visual>
      isa text
      value =val
==>
   +retrieval>
      isa goal
      relation associate
      arg1 =val
      arg2 =ans
   =goal>
      relation associate
      arg1 =val
      step testing)

(p recall
   =goal>
      isa goal
      relation associate
      arg1 =val
      step testing
   =retrieval>
      isa goal
      relation associate
      arg1 =val
      arg2 =ans
==>
   +manual>
      isa press-key
      key =ans
   =goal>
      step waiting)

(p recall-vanilla
   =goal>
      isa goal
      step attending
      state test
   =visual>
      isa text
      value "vanilla
==>
   +manual>
      isa press-key
      key "7"
   =goal>
      relation associate
      arg1 "vanilla"
      step waiting)
```



Production Compilation: The Principles

1. **Perceptual-Motor Buffers:** Avoid compositions that will result in jamming when one tries to build two operations on the same buffer into the same production.

2. **Retrieval Buffer:** Except for failure tests proceduralize out and build more specific productions.

3. **Goal Buffers:** Complex Rules describing merging.

4. **Safe Productions:** Production will not produce any result that the original productions did not produce.

5. **Parameter Setting:**
Successes = P***initial-experience***
Failures = (1-P) ***initial-experience***
Efforts = (Successes + Efforts)(C + *cost-penalty*)

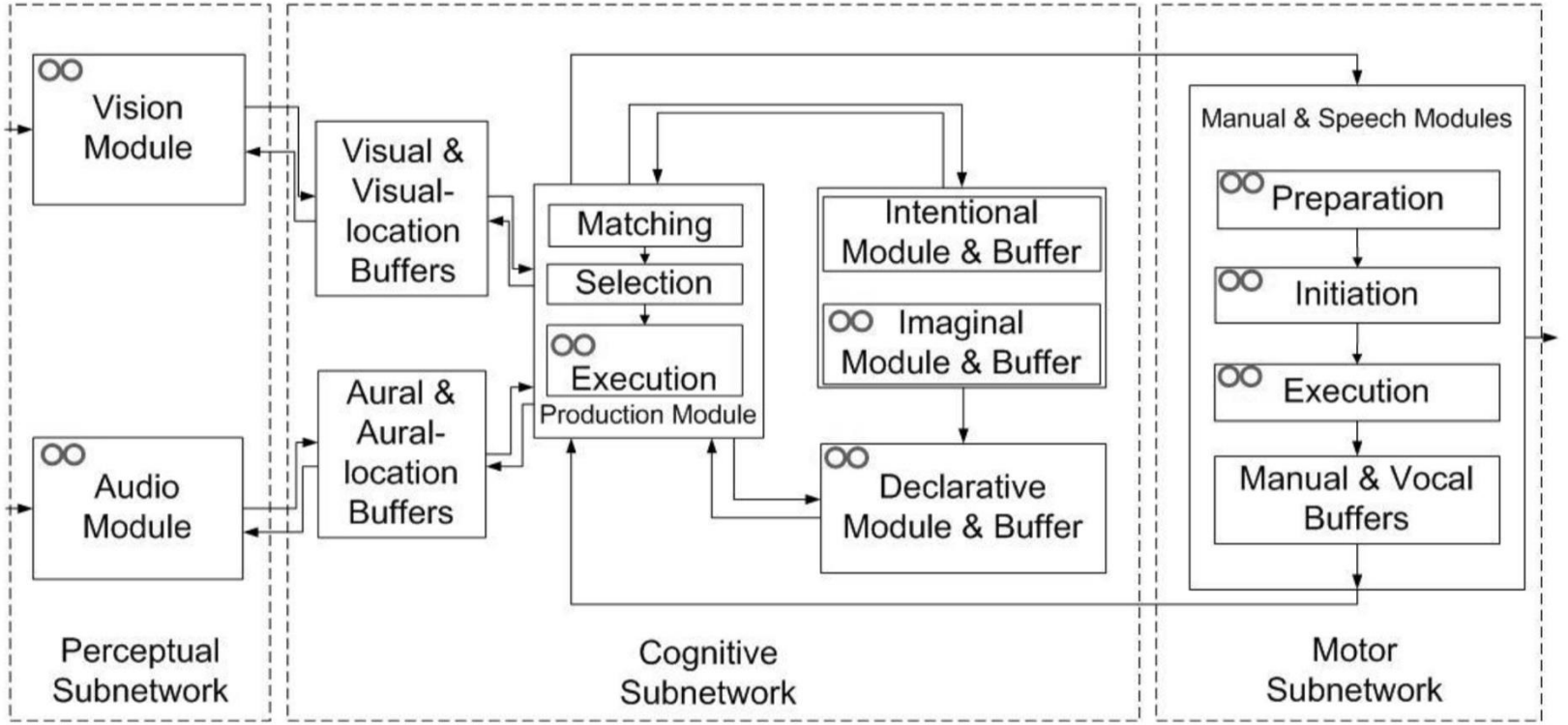


Figure 3. Server structure of QN-ACTR. Queue symbols (shown as two circles) mark the servers where queues are added from the QN's perspective. All the server processing logics in the QN-ACTR are identical to the corresponding algorithms in ACT-R (adapted from Cao & Liu, 2012c).

Harper, 2007). This scenario used the same DISALT shooting testbed and provided behavioural data to model the shooting task. Production rules were defined to model the shooting-only task as the process of visual searching, manual aiming, and pulling the trigger.

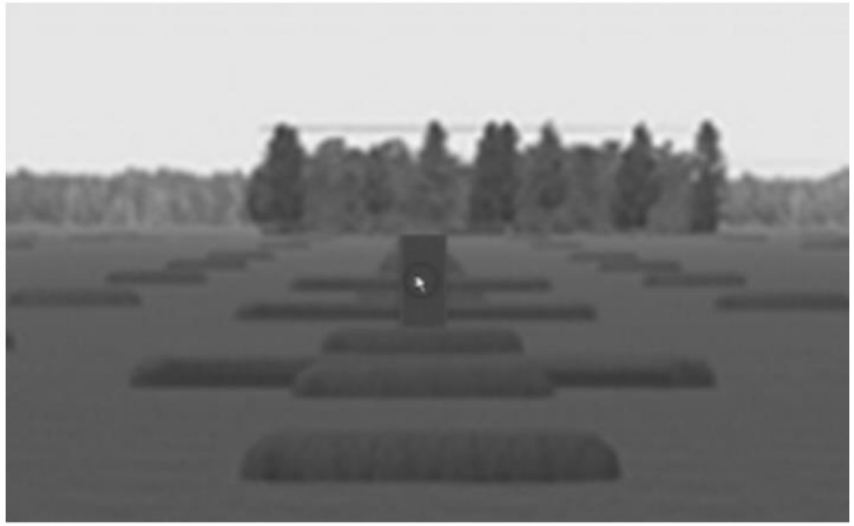


Figure 4 Visualization of the task environment with an enemy-target in QN-ACTR. The cursor represents the iron sight aiming. Background picture from (Scribner et al., 2007).

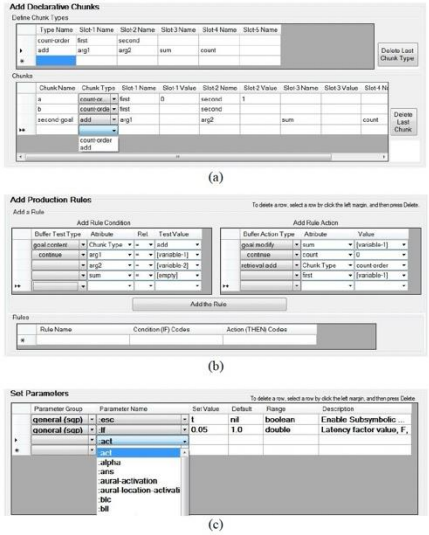


Figure 7. Screenshot of defining the mind and the parameter parts of a model using Model Setup Assistant, including (a) chunks, (b) production rules, and (c) parameters.

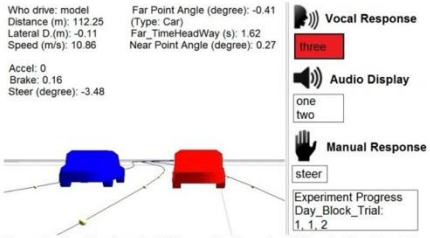


Figure 8. Visualization of a driving and arithmetic addition dual-task in QN-ACTR.

Integrated human experiment interface

The same task interface with which the model interacts can also serve as the interface for human participants to complete the same tasks. We have developed a human driving interface in QN-ACTR that supports simulated driving experiments with steering wheels and pedals. This feature allows the model and the human to perform and be compared in the same tasks with identical interfaces, with no need to replicate the real world experiment system in the modeling

Using the same experiment platform avoids any discrepancy between human and model tests due to the experiment setup.

FINDINGS

The usability development of QN-ACTR is evaluated using Nielsen's ten heuristics for user interface design (1994).

Visibility of system status. MSA always shows the stage of model development at the top-left corner. The visualization of the mind and the task keeps users informed about what is going on in the model during the simulation. Buttons in MSA are dimmed and disabled when their actions cannot be performed in some cases. Program responses and feedbacks are immediate with no delay.

Match between system and the real world. All the column headers in MSA tables and the items in menus use self-explanatory phrases without abbreviation. The steps of modeling in MSA follow the logical order shown in Figure 2. Full names and detailed descriptions are shown for each abbreviated ACT-R parameter name (Figure 7c).

User control and freedom. MSA supports undo (e.g., change the road name, delete a chunk, and reset a table) and redo (e.g., go back to the previous stage, and then go next again). A cancel button is provided at each stage to exit the setup at any time, and then users can restart MSA if needed.

Consistency and standards. Definitions and names are used consistently throughout all modeling steps. Tables and menus follow similar layouts and styles. Button position is the same between templates and stages.

Error prevention. The use of menu selection in MSA tables prevents the input of invalid items. Table cells automatically perform validation check, and users are notified when an input is of an invalid type or out of the valid range. Duplicated names assigned by users (e.g., chunk names) are automatically revised to prevent run-time errors. Syntax errors are also reported before the simulation starts.

Recognition rather than recall. MSA provides menus for users to select their options and tables to fill in. Model developing knowledge is provided to users in the interface. For example, users do not have to learn any modeling syntax. Instead, they can describe the model in natural language and fill in blanks or select items (Figure 7a, b). The default value, valid range, and description of model parameters are displayed for the users (Figure 7c).

Flexibility and efficiency of use. The syntax method and MSA cater to both inexperienced and experienced users. Experienced user can speed up the modeling work by directly copying and editing syntaxes. Syntaxes for the mind and the parameters can also be directly copied from ACT-R codes.

Aesthetic and minimalist design. MSA tables and menus are organized and aligned in groups. Introductions and explanations are concise.

Help users recognize, diagnose, and recover from errors. Error messages are expressed in plain language (no codes) and precisely indicate the problem. For example, "Error! Set General Parameters needs para_name: :lf to be a double rather than nil."

Comparison of Model Implementations

- ACT-R: LISP (mainly and originally), Python (for some models)
  It has an 8-UNIT Tutorial

- QN-ACT-R: (Micro Saint # and Java: Please contact Shi Cao at Uof Waterloo)

- pyactr: python, by Jakub Dotlacil
  Implements all 8 units in python

- **QN-ACT-R: python**
  Uses **pyactr codes**, PLUS **QN codes**, for:

  1. **QN-visualization**

  2. cross-tutorial-unit **QN-ACT-R instructional work**

  3. **QN-ACT-R multi-tasking research work**

QN-ACT-R: python

- Uses **pyactr codes**, PLUS **QN codes**, for:

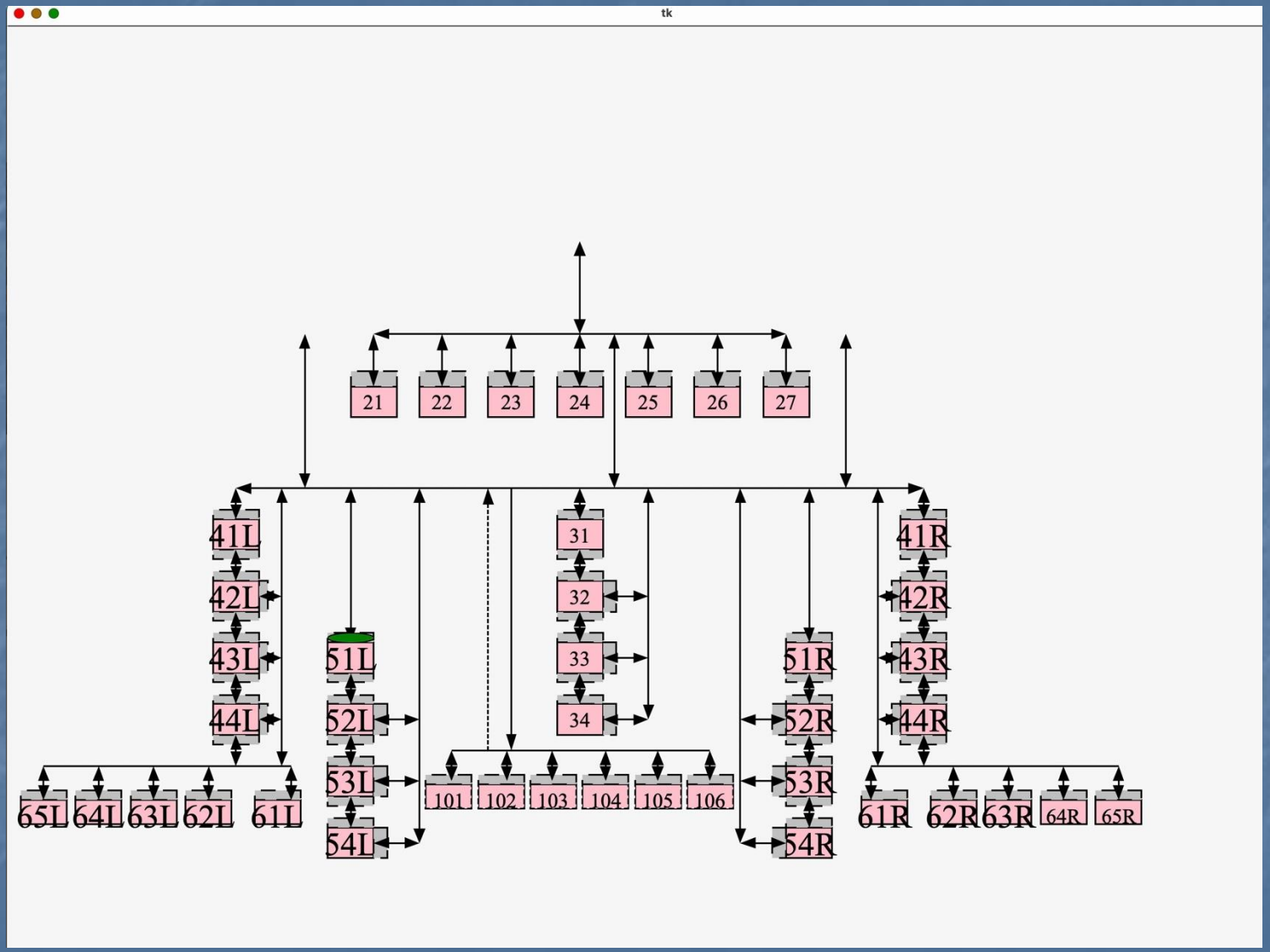
    1. **QN-visualization**

    2. cross-tutorial-unit **QN-ACT-R instructional work**

    3. **QN-ACT-R multi-tasking research work**

Software Demo

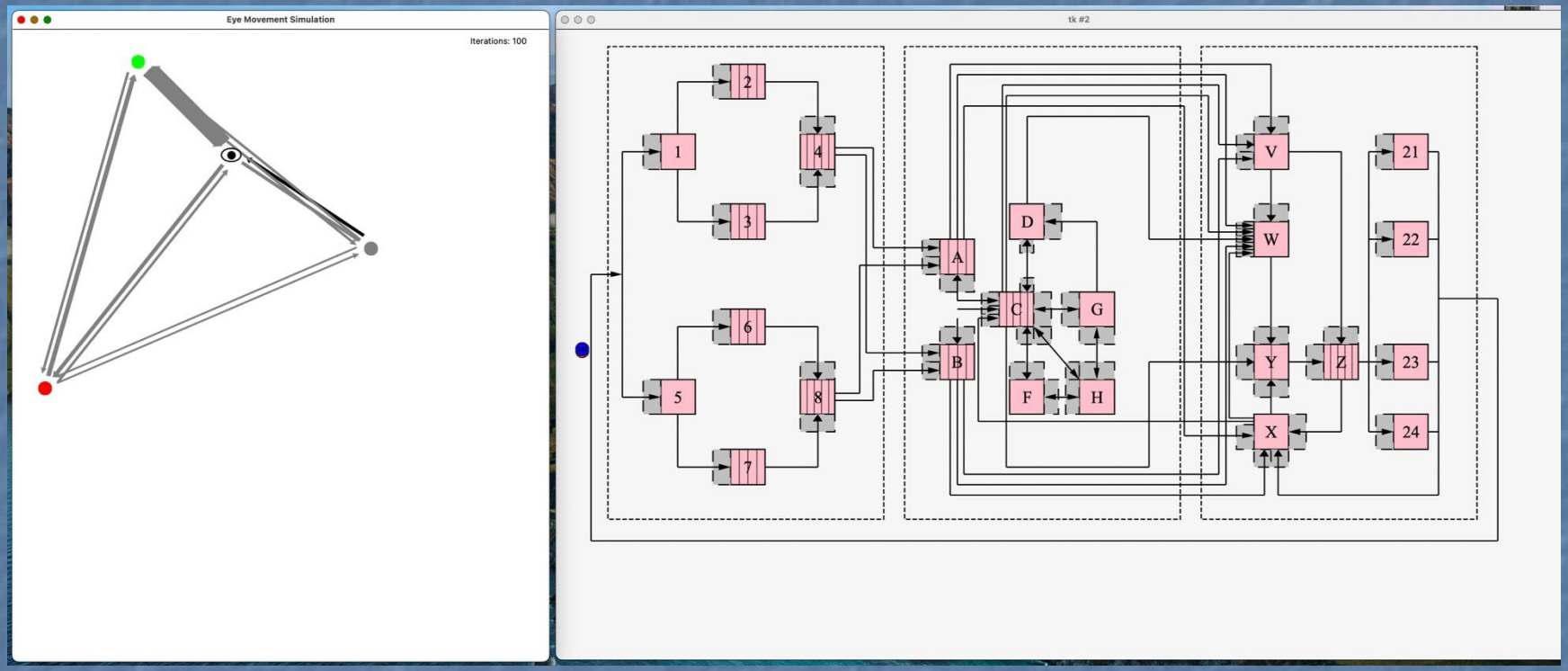
QN-BDS


tk
21
22
23
24
25
26
27
41L
42L
43L
44L
51L
52L
53L
54L
31
32
33
34
101
102
103
104
105
106
51R
52R
53R
54R
41R
42R
43R
44R
65L
64L
63L
62L
61L
61R
62R
63R
64R
65R

QN-BDS (QN-MTM)

Software Demo

QN-SEEV


Eye Movement Simulation
Iterations: 100
tk #2
1
2
3
4
5
6
7
8
A
B
C
D
F
G
H
V
W
X
Y
Z
21
22
23
24

QN-SEEV

Software Demo

QN-Reinforcement Learning-Eye Movement

Investigation of Driver Performance With Night-Vision and Pedestrian-Detection Systems—Part 2: Queuing Network Human Performance Modeling

Ji Hyoun Lim, Yili Liu, *Member, IEEE*, and Omer Tsimhoni, *Member, IEEE*

***Abstract*—This paper introduces a queueing network-based computational model to explain driver performance in a pedestrian-detection task assisted with night-vision-enhancement systems. The computational cognitive model simulated the pedestrian-detection task using images displayed by two night-vision systems as input stimuli. The system equipped with a far-infrared (FIR) sensor generated less-cluttered images than the system equipped with a near-infrared (NIR) sensor. Using a reinforcement learning process, the model developed eye-movement strategies for each night-vision system. The differences in eye-movement strategies generated different eye-movement behaviors, in accord with the empirical findings.**

***Index Terms*—Cognitive model, human performance modeling, night vision, pedestrian detection, queueing network.**

I. Queueing Network Model of Pedestrian Detection and Driving

DRIVER performance is a critical factor in examining the effectiveness and efficiency of an intelligent transportation system (ITS) [1], [2]. Computational models of driver performance can help study driver behavior and assist system design. This study introduces an example of using a computational model to assess driver performance using ITSs (two night-vision-enhancement systems designed to assist in pedestrian-detection tasks). In this study, the near-infrared (NIR) night-vision-enhancement system and the far-infrared (FIR) system were investigated. The NIR system actively illuminates a scene in the NIR spectrum and captures the reflected radiation, whereas the FIR system generates images by passively detecting thermal emissions from objects in a scene of interest.

Manuscript received October 23, 2008; revised April 18, 2010; accepted April 27, 2010. Date of publication June 7, 2010; date of current version December 3, 2010. The Associate Editor for this paper was N. B. Sarter.

J. H. Lim was with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA. He is now with the Mobile Communication Division, Samsung Electronics, Seoul, 137-857 Korea (e-mail: smilelim@gmail.com).

Y. Liu is with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109-2117 USA.

O. Tsimhoni was with the University of Michigan Transportation Research Institute and Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA. He is now with the General Motors Advanced Technical Center-Israel, Herzliya 46725, Israel (e-mail: omer.tsimhoni@gm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2010.2049844

The two night-vision systems generate distinctively different images of the same scene, which may lead to different glance behaviors, such as different glance frequencies and search times to detect pedestrians during night driving. In this study, pedestrian detection with a concurrent driving task was simulated with a computational cognitive model, based on the queueing network architecture, to investigate the different glance behaviors associated with the different night vision systems.

As shown in Fig. 1, the cognitive agent (driver) processes information from the images displayed by the night-vision-enhancement systems and then generates glance behaviors (scans the scene for pedestrians). The relationship between the characteristics of a visual scene (input stimulus) and the glance behaviors has been examined in empirical studies [3]–[6]. The computational modeling approach allows us to investigate the possible underlying cognitive mechanisms of the glance behaviors.

Along the line of research on developing comprehensive human performance models with a unifying cognitive architecture, we have been making steady progress in developing a queuing network (QN) architecture for human performance modeling. Mathematical models based on QNs have successfully integrated a large number of models in response time [7] and in multitask performance [8] as special cases of QNs. A computational model based on the QN mental architecture called the queueing network—model human processor (QN-MHP) [9] has been developed to simulate and generate human performance. This architecture represents a human psychological system in the form of a queueing network with multiple servers that represent functional units in a human brain. Entities represent pieces of information to be processed by the servers. An entity travels on routes, which represent the flow of information in the system. The QN-MHP has successfully been applied to model various task domains including simple and choice reaction tasks [10]–[12], visual search tasks [10], [13], driving [14], [15], driving with a concurrent map reading task [7], and driver workload [16]. For a detailed description of the QN-MHP and its applications methodology, see [9].

Adopting the QN-MHP methodology, a QN menu search model was developed in our earlier study [13], which was significantly extended in the present study to establish a QN model of pedestrian detection. There are nine effective servers for the menu search model used in the previous study [13], which were kept in this pedestrian-detection model. A

Queuing Network Modeling of Driver EEG Signals-Based Steering Control

Luzheng Bi, Yun Lu, XinAn Fan, Jinling Lian, and Yili Liu

***Abstract*— Directly using brain signals rather than limbs to steer a vehicle may not only help disabled people to control an assistive vehicle, but also provide a complementary means of control for a wider driving community. In this paper, to simulate and predict driver performance in steering a vehicle with brain signals, we propose a driver brain-controlled steering model by combining an extended queuing network-based driver model with a brain–computer interface (BCI) performance model. Experimental results suggest that the proposed driver brain-controlled steering model has performance close to that of real drivers with good performance in brain-controlled driving. The brain-controlled steering model has potential values in helping develop a brain-controlled assistive vehicle. Furthermore, this study provides some insights into the simulation and prediction of the performance of using BCI systems to control other external devices (e.g., mobile robots).**

***Index Terms*— Assistive technology, brain-controlled vehicles, electroencephalography (EEG), queuing network modeling.**

I. Introduction

HEALTHY drivers use limbs to operate a vehicle. Driver models representing this type of driver-vehicle interactions have been widely investigated [1]. Researchers can use these models to simulate and predict driving performance and thus help reduce the time and effort on experimental design and implementation in testing the performance of driver assistance systems [2]. Furthermore, these models can be used to develop adaptive driver assistance systems to better assist driving [3], [4]. Thus, driver models are valuable in helping develop driver assistant systems for improving driving performance and safety.

Some disabled people, however, cannot drive a vehicle with limbs. To help these disabled individuals control vehicles to increase their mobility, brain-controlled vehicles (BCVs) have been proposed. A brain-controlled vehicle (BCV) is a vehicle that is directly controlled by the human "mind" through interpreting brain activity signals into motion commands rather than limbs.

BCVs can be considered as one of the applications of brain–computer interfaces (BCIs). Brain signals used to develop BCIs can be measured and recorded with methods such as functional near-infrared imaging (fNIR), recording electrical or magnetic fields, functional magnetic resonance imaging (fMRI), and positron emission tomography (PET) [5], [6]. The electrical fields can be recorded at the scalp (electroencephalography (EEG) signals), on the cortex (electrocorticographic signals), or within the cortex (neuronal action potentials). Currently, compared to other methods, EEG recording is less expensive, simpler, and non-invasive, and thus more suitable for use in practice. EEG signals have been widely applied to develop various brain-controlled systems, such as brain-controlled cursors [7], [8], brain-controlled prosthesis [9], [10], and brain-controlled wheelchairs [11]–[13]. In contrast to brain-controlled wheelchairs, BCVs are more complicated in dynamic characteristics, and they travel faster in a more complicated environment.

Recently, some researchers have started to explore how to develop a BCV with EEG signals because of the advantages of EEG recordings mentioned above. Gohring *et al.* [14] explored how to employ a commercial BCI product from Emotiv to control an intelligent vehicle. If the intelligent vehicle is about to leave the predefined free driving zone or to collide instantly, it immediately stops itself and resets the BCI system. They tested the system by requiring one participant to finish driving on a closed airfield. The experimental results indicated that it was feasible to use EEG signals in conjunction with an autonomous control system to control a vehicle. In our previous work [15], we have developed a new steady-state visually evoked potential (SSVEP) BCI, whose visual stimuli were presented on a windshield with a head-up display, and used such BCI along with the alpha rhythm of EEG to operate the simulated vehicle supported by a 14-degree of freedom (DOF) vehicle dynamics model. We tested the BCV by asking four participants to finish a driving task online. The experimental results showed the feasibility of only using EEG to continuously steer vehicles. The major difference between [15] and [14] is that we developed a novel BCI and applied it to completely and continuously steer a simulated vehicle without any stops during the whole testing process, whereas [14] applied a BCI product (without reporting the performance of the BCI) to intermittently control an intelligent vehicle, which can stop itself and reset the BCI.

Manuscript received February 25, 2016; revised August 21, 2016; accepted September 24, 2016. Date of publication September 28, 2016; date of current version August 7, 2017. This work was supported by the National Natural Science Foundation of China under Grant 61374192 and Grant 51575048 and the Beijing Natural Science Foundation under Grant 4162055.

L. Bi, Y. Lu, and J. Lian are with the School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081 China. (e-mail: bhxblz@bit.edu.cn; luyun_bit@126.com; lianjinling@bit.edu.cn).

X. A. Fan is with the Beijing Institute of Mechanical Equipment, Beijing 100854 China. (e-mail: anmengxiang@126.com)

Y. Liu is with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: yililiu@umich.edu).

Digital Object Identifier 10.1109/TNSRE.2016.2614003

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2023.3342151

Modeling Car-Following Behavior of Brain-Control Driving with Queuing Network Architecture

Yun Lu, *Member, IEEE,* Rong Su, *Senior Member, IEEE,* and Yili Liu

***Abstract*—Directly using brain signals to drive vehicles is called brain-control driving, which has the potential to help people with disabilities to acquire driving ability. Developing driver models of the brain-control driving can help understand and simulate the driver behaviors, which contributes to the development of the brain-control driving. In this paper, to simulate the car-following behavior of the brain-control driving, we propose two brain-control car-following models, i.e., CF-BCB and CF-BCI models. The two models are built by integrating a car-following decision model with a brain-control behavior model and a brain-computer interface (BCI) performance model in the queuing network (QN) cognitive architecture, respectively. The manual- and brain-control experiments are conducted to collect the car-following data of the ideal and real brain-control driving, respectively. Simulations with the proposed models are performed to mimic the ideal and real brain-control driving of different subjects. We compare the performance of the car-following decision model and two decision models extended from the intelligent driver model (IDM) and the full velocity difference model (FVDM), respectively. The comparison results show that the car-following decision model performs better than the two decision models in simulating the ideal brain-control driving. Besides, we prove the effectiveness of the proposed models in simulating the car-following behaviors of the brain-control drivers by comparing the simulation and experimental results.**

***Index Terms*—Brain-control driving, car-following behavior, queuing network, driver model.**

I. Introduction

DIRECTLY using brain signals to drive vehicles is called brain-control driving. Unlike the manual-control driving, the brain-control driving does not rely on muscles or peripheral nerves of human drivers. Such driving method, for one thing, has the potential to help people with disabilities to acquire driving ability. For another thing, it can free the limbs of healthy people by offering a new driving approach. The brain-control driving is performed by using brain-computer interfaces (BCIs), which can directly interpret the brain signals of humans into control commands [1]. Electroencephalography (EEG) has become the most popular technique to develop brain-computer interface (BCI) systems due to its low cost, high temporal resolution, and ease of use [2].

Several researches have explored to employ EEG-based BCIs to perform the brain-control driving. Göhring *et al.* [3] provided subjects with a BCI product from Emotive to drive an autonomous vehicle. Users could perform the speed and steering control when the vehicle was in a safe zone. In [4], a motor imagery-based BCI was designed for drivers to control the steering, movement, and engine of a vehicle at the velocity of about 5 km/h. Bi *et al.* [5] have investigated how to use a steady state visually evoked potential (SSVEP) BCI to control the steering of a vehicle with the speed of 2–3 m/s. In [6], we have developed a longitudinal driving system equipped with an SSVEP BCI for drivers to conduct the longitudinal control of a vehicle with the speed being no higher than 10 m/s. Lu and Bi [7] have explored the feasibility of applying an SSVEP-based BCI to perform the combined longitudinal and lateral control of a vehicle.

Owing to the performance limitations (e.g., accuracy limitation) of BCIs, the driving performance of the vehicles relying on BCIs is poor. A few studies have explored to build assistive control methods to improve the driving performance. Our previous works [8] and [9] have designed two shared control methods for the brain-controlled vehicles built in [5] and [6] to improve their lateral and longitudinal driving performance, respectively. Shi *et al.* [10] have built a control strategy by using the model predictive control (MPC) and fuzzy logic to improve the lateral performance of the brain-controlled vehicles. However, even with the assistance of the shared control techniques, the performance of the brain-control driving is not good enough. To further improve its performance, it is significant for vehicle designers and control engineers to understand and simulate the behavior of the brain-control drivers.

Developing driver models can help understand, compute, predict, and simulate the related driver behaviors [11], [12]. In practice, driver models can help increase the development efficiency of vehicle systems by decreasing the requirement to perform driver-in-the-loop experiments [13], [14]. They also contribute to the development of advanced driver assistance systems [15], [16]. Numerous researches have been carries out

Manuscript received January 10, 2023; revised May 20, 2023, August 30, 2023; accepted December 8, 2023. This work was supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-GC-2023-007). *(Corresponding author: Yun Lu)*

Y. Lu and R. Su are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798. (e-mail: yun.lu@ntu.edu.sg; rsu@ntu.edu.sg).

Y. Liu is with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA. (e-mail: yililiu@umich.edu).

(a) Tuning radio using the knob

(b) A zoom-in view of the physical panel

Figure 31. Task Procedure using the knob: (1) press the power button, (2) press the AM/FM button, (3) turn the knob to decrease or increase the frequency shown on the display ("590" as in the picture)

(a) Tuning radio on the touch screen

(b) Click the "Entertainment" button

(c) Click the "FM" then "Direct Tune" button

(d) Enter the radio frequency

Figure 32. Task procedure using the virtual buttons

Queuing Network Modeling of Driver Lateral Control With or Without a Cognitive Distraction Task

Luzheng Bi, *Member, IEEE*, Guodong Gan, Junxing Shang, and Yili Liu, *Member, IEEE*

***Abstract*—In this paper, we propose a computational model of driver lateral control based on the queuing network cognitive architecture and the driver preview model about driver lateral control activities. This computational model was applied to model the dual tasks of driving with a cognitive distraction task. The comparison between human driver data and model simulation data shows that this computational model can perform vehicle lateral control well, and its performance is consistent with that of drivers under single- and dual-task driving conditions. Furthermore, we examine the effectiveness of some parameters of the model in representing different styles of driving and discuss the value of this computational model in facilitating the evaluation of vehicle dynamics and driver assistant systems and providing new insights into research on unmanned vehicle control techniques.**

***Index Terms*—Cognitive distraction, driver lateral control, multitask modeling of driving, queuing network (QN).**

I. Introduction

COMPUTATIONAL models of driver behavior cannot only provide scientific understanding of driver behavior but help compute, simulate, and predict a variety of aspects of driver behavior as well [1], [2]. These models can help improve the development process of vehicles and driver assistant systems by reducing or eliminating the need for conducting driver experiments in the early stage of the development [3]–[6]. In addition, they play a major role in evaluating complex traffic situations for traffic engineers [7]. Furthermore, these driver models can provide insights into the research and development of intelligent driver assistant systems or unmanned vehicle control techniques [8]–[10].

Since the mid-1950s, researchers have proposed various driver models, including longitudinal control driver models and lateral control/combined control driver models. One representative category of driver lateral control models is driver preview (preview/predictive) models [11], which were developed by mimicking driver preview and predictive behavior based on findings that drivers steer a car using visual information from the road ahead [12], [13]. Generally, these driver preview models preview the desired path for a preview interval to obtain information of the desired path and predict vehicle response within the preview interval by using an internal model. The steering angle is finally decided based on the lateral position error between the desired and predicted trajectories. More details about driver preview models can be seen in [11] and [14]. Early literature reviews of the broad field of driver models can be seen in [15] and [16]. Two recent comprehensive reviews include understanding and modeling the human driver by Macadam [17] and driver models in automobile dynamics application by Plochl and Edelmann [18]. These reviews provide comprehensive and excellent comments and insights regarding driver models. Although some of these models can represent some aspects of human psychological limitations (such as Macadams' nonlinear model [19], an errorable car-following driver model proposed by Peng *et al.* [20], and a hybrid driver model proposed by Kiencke *et al.* [21]), they lack a unified architecture to capture a broad range of human capabilities and constraints. In particular, the models previously mentioned do not support multitask (driving and secondary tasks) modeling and thus cannot account for and simulate cognitive bottlenecks or distractions from secondary tasks, thus limiting their value in supporting the development of driver assistant systems.

To address this challenge, some researchers have started to develop driver models based on cognitive architectures, which can better account for and simulate driver behavior because cognitive architectures embody human abilities and constraints. Aasman presented a model of driving behavior using the State Operator And Result (SOAR [22]) cognitive architecture [23]. Salvucci *et al.* developed models of driver behavior with the Adaptive Control of Thought-Rational (ACT-R) cognitive architecture [2], [24], which is a symbolic architecture based on chunks of declarative knowledge and condition–action production rules [25]. Liu *et al.* developed a basic model of driver performance based on the queuing network (QN) cognitive architecture [26], [27].

These driver models based on cognitive architectures differ mainly in the characteristics of different cognitive architectures. Both ACT-R and SOAR are symbolic architectures/models. They lack mathematical frameworks for representing their overall "architecture." In contrast, QN cognitive architecture is a mathematical model and particularly suited for real-time generation of concurrent activities in a concurrent manner. A comprehensive comparison among these cognitive architectures can be seen in [28]. Recently, the QN cognitive architecture has been successfully applied to model various task domains

Manuscript received October 27, 2011; revised February 6, 2012, May 25, 2012 and June 5, 2012; accepted June 7, 2012. Date of publication June 28, 2012; date of current version November 27, 2012. This work was supported by the National Natural Science Foundation of China under Grant 61004114 and Grant 90920304 and in part by the Fundamental Research Funds for the Central Universities under Grant 2012CX01018. The Associate Editor for this paper was C. Wu.

L. Bi, G. Gan, and J. Shang are with the School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China (e-mail: bhxblz@bit.edu.cn; gangd8701@163.com; 20903041@bit.edu.cn).

Y. Liu is with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: yililiu@umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2012.2204255

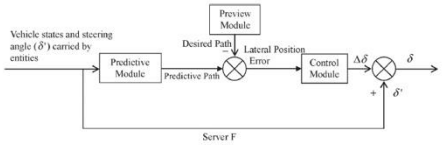


Fig. 2. Schematic of driver lateral control implemented in Server F.

The preview module previews the desired path for a preview interval to obtain information of the desired path. This paper uses a predefined desired path since it is not the purpose and is beyond the scope of this paper to determine arbitrary desired path in real time. The preview time can affect the performance of driver lateral control and likely show different driving styles, which will be discussed in the section on model validation.

The prediction module predicts the vehicle response within the preview interval by using an internal model with vehicle states and steering angle as its inputs, which can be a simplified dynamic system of the vehicle, representing driver learning and adaptation to the controlled vehicle. In this paper, a 3-degree-of-freedom (DOF) vehicle dynamics model, including longitudinal, lateral, and yaw movement, was constructed as the internal model according to [34].

The control module computes the desired control input (i.e., the increment of steering angle $\Delta\delta$) to make the vehicle track the desired path. The calculation proceeds as follows: The desired acceleration a_y (assuming it is a constant in the preview time) is first calculated with the following equation:

$$a_y = \frac{2 \cdot (E - v \cdot t_p)}{t_p^2} \tag{1}$$

where E is the error between the desired lateral position gained with the predefined desired path and predictive lateral position computed with the internal vehicle dynamics model, ν is the current lateral velocity, and t_p is the preview time.

The increment of steering angle $\Delta\delta$ is then calculated with a simple proportional derivative controller of acceleration, as follows:

$$\Delta\delta = k_p \cdot a_y + k_d \cdot a'_y \tag{2}$$

where k_p and k_d are the coefficients of the proportional–derivative (PD) controller, which represent the knowledge and style of driving, and a'_y is the first derivative of the acceleration. The driver control model can compensate the lateral position and yaw angle errors (see the Appendix for the inference) to control the vehicle to track the desired path. Furthermore, the simple form of the model is good for exploring the relationship between the parameters of the model and the style of driving.

Finally, a new steering angle δ is computed with the following equation and is carried by entities, which are transmitted to the motor subnetwork and implemented by the hand server:

$$\delta = \delta' + \Delta\delta \tag{3}$$

where the δ' stands for the steering angle of the last cycle.

TABLE I
BASIC PARAMETERS OF THE QN ARCHITECTURE

server	processing time	capacity
1	E (0.042,0.025)	4
2	E (0.042,0.025)	4
3	E (0.042,0.025)	4
4	E (0.042,0.025)	5
A	E (0.018,0.006)	4
C	E (0.018,0.006)	3
F	E (0.018,0.006)	1

4) Parameter Settings: The parameters used in the proposed model can be classified into two sets: One set mainly consists of the psychological parameters (i.e., the standard and basic parameters of the QN architecture [29]). It mainly includes server processing time and server capacity. The processing time was assumed to be stochastic and exponentially distributed with mean X and an axis shift of Y [$\sim$ E(X, Y)], as shown in Table I. The second set of parameters is specifically related to driving, including parameters of the preview, prediction, and control modules (i.e., preview time, k_p, and k_d). These parameters depend on specific driving situations and, thus, were estimated by testing their values with experimental data under the driving conditions.

5) Model Output: The driver model outputs a steering angle, which is associated with the hand position as the hands move the steering wheel.

C. Driving and Secondary Dual-Task Modeling

Secondary tasks are tasks directly unrelated to driving that drivers voluntarily or involuntarily engage in [35]. Driving includes three concurrent activities, i.e., perceptual, cognitive, and motor activities. Accordingly, secondary tasks can influence the three activities of driving. For example, looking at an in-vehicle navigation system can affect the perceptual activity of driving; counting numbers and recollecting words can influence the cognitive activity of driving, whereas drinking can affect the motor activity of driving.

To demonstrate how the proposed driver model can be extended to implement modeling of concurrent driving and secondary tasks, we conducted multitask modeling of driving and a cognitive secondary task (counting backward in a certain step), which has been applied to evaluate driver distraction by researchers and practitioners in automobile industry [36], as an example.

To model concurrent tasks beyond the single task of lateral control, three servers from the QN architecture were employed and added to the eight-server structure for the driver lateral control model described in Section II-B. The three servers are the goal selection server, the long-term procedural memory server, and the mouth server. Fig. 3 shows the 11 effective servers and their functions in modeling the driving and secondary tasks. Since Server F, representing complex cognitive

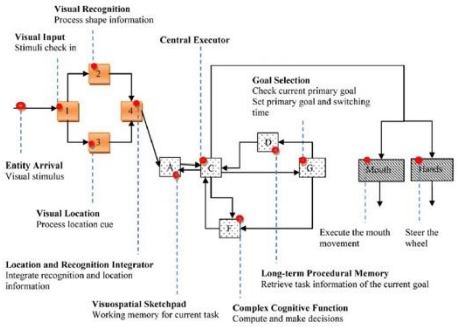


Fig. 3. QN architecture of the driving and secondary dual-task model.

function, can only process one task at once, the two tasks have to be processed in turn through task switching when they both compete for Server F. The QN model manages task switching by using a scheduled goal prioritization. Dual-task modeling has the following three components in common: 1) modeling the first task; 2) modeling the second task; and 3) a method to manage the two tasks [33]. The driver model has been described in Section II-B. The other two components are introduced as follows:

1) Modeling the Secondary Task: Like modeling the primary task of driving with the QN, modeling the secondary task with the QN needs to specify the QN architecture. Generally, the servers whose functions are associated with the target task are included in the architecture.

In this paper, the secondary task modeled was requiring subjects to count backward from a particular initial number in a specified step and to give answers in an oral expression. Since subjects count in their heads without the need to perceive any new information, the servers in the perceptual subnetwork were not included in the architecture for the counting task. The to-be-counted numbers carried by entities are first transmitted from Server A, which is the working memory server, to Server F, which is the complex cognitive function server that performs the number counting, by Server C, which is the central executor server. Then, entities carrying the processed numbers are sent back to both Server A for updating the to-be-counted numbers and the mouth server to voice the answer, via Server C.

2) Task Switching by Goal Prioritization: In the QN model, the two concurrent tasks are encoded as two streams of entities traversing the network. To model task switching between the two streams representing driving and secondary tasks, respectively, two assumptions were made: First, the driving and secondary tasks were well learned. Second, task switching was scheduled with certain time intervals.

In the model, task switching for the driving and secondary tasks was implemented by using the goal selection server (see Server G in Fig. 3), which manages task priority. Upon an entity's arrival at the goal selection server by Server C, the goal selection server first checks the current primary goal. If the current primary goal is the driving task of changing the lane, no task switching happens for safe driving, which means that the secondary task is completely ignored during the process of lane changing. If the vehicle is running outside the road boundaries, the goal selection server instantly switches to the driving task, given that the current goal is the secondary task. For other situations, two tasks are scheduled every p seconds in the goal selection server. The goal selection server checks the current primary goal. The entity is then transmitted to the long-term procedural memory server (see Server D in Fig. 3), which retrieves a task information that is related to the current primary goal. After that, the entity can be processed according to the associated task model (i.e., the driver lateral control model or the secondary task model).

III. Model Validation

To validate the developed driver model, a comparison between the model simulation data and the real driver data under the single- and dual-task situations of driving is required. The driving behavior was measured by lateral displacement, yaw angle, lateral acceleration, and steering wheel angle, whereas the performance of the secondary task was measured by the amount of counted numbers. (The reason for not using error number as a measure was that the secondary task was very easy, and all participants made few mistakes, as shown in the following experiment result.) We assumed that the desired trajectory was predetermined as the center line of the lane and the vehicle speed maintained constant in the whole process of each experiment or simulation at the speeds of 50 and 80 km/h, representing a relatively low speed and a relatively high speed, respectively. (80 km/h or so is widely used to subjectively evaluate vehicle dynamics under the ISO severe double-lane-changing scenario used in this paper [37], and the higher speed than 80 km/h is quite difficult for subjects, if not impossible, to perform the driving task.) All the measures used in the following were the means of all participants. In other words, we did not fit the behavior of individual drivers, in this paper. However, we simulated and discussed the effects of some parameters of this model on driving and how they can somewhat represent some different styles of driving.

A. Human and Model Simulation Data

1) Human Driver Data: Sixteen male drivers (aged 20–26 with a mean age of 23) with 1–3-year driving experience attended two experiments (including single- and dual-task driving with a secondary task) in a driving simulator with a 14-DOF vehicle dynamics model from the CarSim software produced by the Mechanical Simulation Corporation. Furthermore, we used a within-participant design. All participants were physically healthy and were randomly divided into four groups to balance the order of single task and dual tasks and the order of the speeds of 50 and 80 km/h.

We applied an ISO severe double-lane-changing scenario [37], as shown in Fig. 4(a), to validate the proposed driver lateral control model, because this scenario needs a high lateral acceleration, which is quite important to driving tasks like evasive maneuvers, as mentioned in Section I, and is well

form vehicle lateral control well, and its performance is consistent with that of drivers under the dual-task situation. This result lends support to the assumption of the QN architecture that multiple tasks can be concurrently performed as long as they do not compete for service at Server F at the same time. Server F can only process one task at a time and will cause performance degradation of a human driver when two tasks com-

APPENDIX

The specific inference procedure is given as follows:
According to (1), we have

$$a'_y = \frac{2}{t_p^2}(E - v \cdot t_p)' = \frac{2}{t_p^2}(E' - a_y \cdot t_p). \tag{4}$$

In terms of [11], we have

$$\boldsymbol{E'} = \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{\alpha} \tag{5}$$

where u is the longitudinal speed, and α represents the yaw angle error.

By combining (4) and (5), we have

$$\begin{aligned} a'_y &= \frac{2}{t_p^2}(E - v \cdot t_p)' \\ &= \frac{2}{t_p^2}(E' - a_y \cdot t_p) \\ &= \frac{2}{t_p^2}(v + u \cdot \alpha - a_y \cdot t_p). \end{aligned} \tag{6}$$

Furthermore, by combining (2) and (6), we have

$$\begin{aligned} \Delta\delta &= k_p \cdot a_y + k_d \cdot a'_y \\ &= k_p \cdot a_y + k_d \cdot \left(\frac{2}{t_p^2}(v + u \cdot \alpha - a_y \cdot t_p)\right) \\ &= \left(k_p - k_d \cdot \frac{2}{t_p}\right) \cdot a_y + k_d \cdot \frac{2}{t_p^2}(v + u \cdot \alpha) \\ &= \left(k_p - k_d \cdot \frac{2}{t_p}\right) \cdot \frac{2 \cdot (E - v \cdot t_p)}{t_p^2} + k_d \cdot \frac{2}{t_p^2}(v + u \cdot \alpha) \\ &= \frac{2}{t_p^2} \cdot \left(\left(k_p - k_d \cdot \frac{2}{t_p}\right) \cdot E + k_d \cdot u \cdot \alpha \right. \\ &\qquad \left. + (3 \cdot k_d - t_p \cdot k_p) \cdot v\right). \end{aligned} \tag{7}$$

REFERENCES

[1] C. Wu and Y. Liu, "Queuing network modeling of driver workload and performance," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 3, pp. 528–537, Sep. 2007.

[2] D. D. Salvucci, "Modeling driver behavior in a cognitive architecture," *Hum. Factors*, vol. 48, no. 2, pp. 362–380, Summer 2006.

[3] L. Bi, O. Tsimhoni, and Y. Liu, "Using image-based metrics to model pedestrian detection performance with night-vision systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 1, pp. 155–164, Mar. 2009.

[4] L. Bi, O. Tsimhoni, and Y. Liu, "Using the support vector regression approach to model human performance," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 41, no. 3, pp. 410–417, May 2011.

[5] P. Angkititrakul, R. Terashima, and T. Wakita, "On the use of stochastic driver behavior model in lane departure warning," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 1, pp. 174–183, Mar. 2011.

[6] G. Xu, L. Liu, Y. Ou, and Z. Song, "Dynamic modeling of driver control strategy of lane-change behavior and trajectory planning for collision Prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1138–

[10] B. Lin and C. Wu, "Mathematical modeling of the human cognitive system in two serial processing stages with its applications in adaptive workload-management systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 1, pp. 221–231, Mar. 2011.

[11] A. Y. Ungoren and H. Peng, "An adaptive lateral preview driver model," *Veh. Syst. Dyn.*, vol. 43, no. 4, pp. 245–259, Apr. 2005.

[12] J. P. Wann and D. Swapp, "Why you should look where you are going," *Nat. Neurosci.*, vol. 3, no. 7, pp. 647–648, Jul. 2000.

[13] M. F. Land and D. N. Lee, "Where we look when we steer," *Nature*, vol. 369, pp. 742–744, 1994.

[14] H.-S. Tan and J. Huang, "Experimental development of a new target and control driver steering model based on DLC test data," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 375–384, Mar. 2012.

[15] E. R. Hoffmann, "Human control of road vehicles," *Veh. Syst. Dyn.*, vol. 5, no. 1/2, pp. 105–126, 1976.

[16] L. D. Reid, "A survey of recent driver steering behavior models suited to accident studies," *Accident Anal. Prevention*, vol. 15, no. 1, pp. 23–40, Feb. 1983.

[17] C. C. Macadam, "Understanding and modeling the human driver," *Veh. Syst. Dyn.*, vol. 40, no. 1–3, pp. 101–134, 2003.

[18] M. Plochl and J. Edelmann, "Driver models in automobile dynamics application," *Veh. Syst. Dyn.*, vol. 45, no. 7/8, pp. 699–741, Jul./Aug. 2007.

[19] C. C. Macadam, "Development of a driver model for near/at-limit vehicle handling," Gen. Motors Corp., Detroit, MI, UMTRI-2001-43, 2001.

[20] H.-H. Yang and H. Peng, "Development of an errorable car-following driver model," *Veh. Syst. Dyn.*, vol. 48, no. 6, pp. 751–773, Oct. 2009.

[21] U. Kiencke, R. Majjad, and S. Kramer, "Modeling and performance analysis of a hybrid driver model," *Control Eng. Pract.*, vol. 7, no. 8, pp. 985–991, Aug. 1999.

[22] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An architecture for general intelligence," *Artif. Intell.*, vol. 33, no. 1, pp. 1–64, Sep. 1987.

[23] J. Aasman, *Modelling Driver Behaviour in Soar*. Leidschendam, Netherlands: KPN, 1995.

[24] D. D. Salvucci and N. A. Taatgen, "Threaded cognition: An integrated theory of concurrent multitasking," *Psychol. Rev.*, vol. 115, no. 1, pp. 101–130, 2008.

[25] J. R. Anderson, D. Bothell, M. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind," *Psychol. Rev.*, vol. 111, no. 4, pp. 1036–1060, 2004.

[26] O. Tsimhoni and Y. Liu, "Modeling steering using the queuing network-model human processor (QN-MHP)," in *Proc. 47th Annu. Meeting Hum. Factors Ergon. Soc.*, Santa Monica, CA, 2003, pp. 1875–1879.

[27] Y. Liu, R. Feyen, and O. Tsimhoni, "Queueing network-model human processor (QN-MHP): A computational architecture for multi task performance in human-machine systems," *ACM Trans. Comput.-Hum. Interact.*, vol. 13, no. 1, pp. 37–70, 2006.

[28] Y. Liu, "QN-ACES: Integrating Queuing Network and ACT-R, CAPS, EPIC, and Soar architectures for cognitive modeling," *Int. J. Hum. Comput. Interact.*, vol. 25, no. 6, pp. 554–581, 2009.

[29] R. Feyen, "Modeling human performance using the queuing network model human processor (QN-MHP)," Ph.D. dissertation, Dept. Ind. Oper. Eng., Univ. Michigan, Ann Arbor, MI, 2002.

[30] C. Wu, Y. Liu, and C. Walsh, "Queuing network modeling of a real-time psychophysiological index of mental workload—P300 in event-related potential (ERP)," *IEEE Trans. Syst., Man, Cybern. A, Sys., Humans*, vol. 38, no. 5, pp. 1068–1084, Sep. 2008.

[31] C. Wu and Y. Liu, "Queueing network modeling of transcription typing," *ACM Trans. Comput.-Hum. Interact.*, vol. 15, no. 1, p. 6, May 2008.

[32] C. Wu and Y. Liu, "Queuing network modeling of the psychological refractory period (PRP)," *Psychol. Rev.*, vol. 115, no. 4, pp. 913–954, Oct. 2008.

[33] J. H. Lim, Y. Liu, and O. Tsimhoni, "Investigation of driver performance with night vision and pedestrian detection systems—Part 2. Queuing network human performance modeling," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 4, pp. 765–772, Dec. 2010.

[34] H. Pacejka, *Tyre and Vehicle Dynamics*. Oxford, U.K.: Elsevier,

Development of a Driver Lateral Control Model by Integrating Neuromuscular Dynamics Into the Queuing Network-Based Driver Model

Luzheng Bi, *Member, IEEE*, Mingtao Wang, Cuie Wang, and Yili Liu, *Member, IEEE*

***Abstract*—This paper describes the development of a novel driver lateral control model by integrating the driver's neuromuscular dynamics into the queuing network (QN)-based driver lateral control model. Experimental results from 16 participants in a driving simulator show that, compared to the QN-based model, the proposed model performs better, and its performance is closer to that of drivers when a vehicle runs at a relatively high speed. The proposed model not only has the advantages of the models based on a cognitive architecture but also captures the dynamic interaction between the vehicular steering system and the driver's neuromuscular system. Thus, it can better represent driver lateral control and has greater value in supporting the development of driver assistance systems.**

***Index Terms*—Driver assistance system, driver lateral control, neuromuscular dynamics, queuing network.**

I. Introduction

DRIVER models can help us to understand, compute, and simulate driver behavior. They have important values in assisting the development of vehicles and driver assistance systems and traffic flow simulations [1]–[3]. One main group of driver models is driver lateral control models.

A variety of driver lateral control models have been proposed. One important kind of driver lateral control models is the driver preview models [4], [5]. These models were built by imitating driver preview and predictive behaviors. Furthermore, researchers have developed some driver models with neuromuscular dynamics for driver steering control to capture the dynamic interaction between the vehicular steering system and neuromuscular system [6]–[8]. However, these models mentioned above lack a unified architecture to capture a broad range of human capabilities and constraints. Particularly, they do not support dual-task (driving and secondary tasks) modeling, and thus cannot account for and simulate distractions from secondary tasks [1].

Manuscript received July 11, 2014; revised September 28, 2014 and November 30, 2014; accepted February 27, 2015. Date of publication March 18, 2015; date of current version September 25, 2015. This work was supported by the National Natural Science Foundation of China under Grants 61004114 and 61374192. The Associate Editor for this paper was F.-Y. Wang.

L. Bi, M. Wang, and C. Wang are with the School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China (e-mail: bhxblz@bit.edu.cn; 490355347@bit.edu.cn; wangcui_e@126.com).

Y. Liu is with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: yililiu@umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2015.2409115

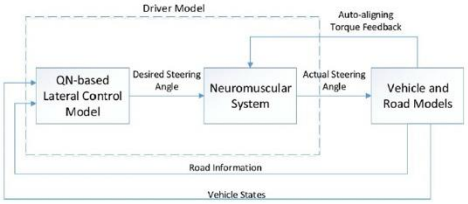


Fig. 1. The QN-based driver lateral control model with neuromuscular system.

To deal with this issue, researchers have started to develop driver models based on cognitive architectures because cognitive architectures have a unified architecture to capture human capabilities and limitations in performing the various tasks. Aasman applied the State Operator And Result (SOAR) cognitive architecture to develop a driver model [9]. Liu *et al.* developed a preliminary driver model based on the Queuing Network (QN) cognitive architecture [10]. Salvucci *et al.* [11] proposed a driver model by using the Adaptive Control of Thought-Rational (ACT-R) cognitive architecture. Bi *et al.* [1] have developed a driver lateral control model using the QN cognitive architecture in conjunction with a driver preview/predictive model and modeled the dual-task driving with a cognitive distraction task based on this lateral control model. Furthermore, Bi *et al.* have employed the QN-based driver lateral control model and logistic regression to develop a dual-task model of driving with a visual distraction task [12].

However, these existing models based on cognitive architectures do not include the driver's neuromuscular system. Thus, they cannot capture the dynamic interaction between the vehicular steering system and the driver's neuromuscular system, which is important to steering control, particularly when a vehicle runs at a relatively high speed (such as, higher than 80 km/h). The weakness limits the value of these models in supporting the development of driver assistance systems.

In this paper, to better understand and model driver lateral control behavior, we develop a novel driver lateral control model by integrating the driver's neuromuscular dynamics into the QN-based driver model previously developed by us. The contribution of this paper is that this is the first study in which the neuromuscular dynamics is integrated into a driver model based on a cognitive architecture. By integrating the driver

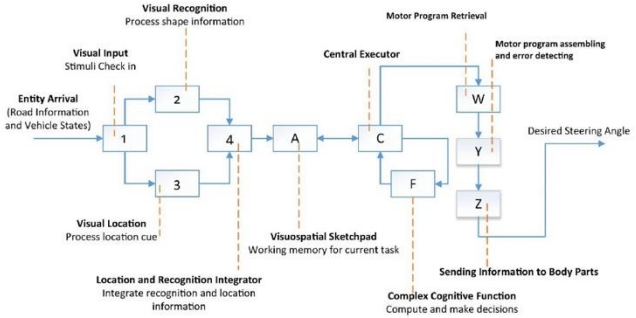


Fig. 2. The QN architecture of the proposed driver lateral control model.

neuromuscular dynamics into the QN-based driver model, the integrated model can not only improve the performance of the QN-based cognitive model by capturing the dynamic interaction between the vehicular steering system and the driver's neuromuscular system, but it also can support dual-task modeling of driving with a motor distraction task, and thus can account for and simulate distractions from motor secondary tasks compared to other driver models with the driver neuromuscular dynamics.

The remainder of this paper is organized as follows. In Section II, we describe how the novel driver lateral control model was developed by integrating the driver's neuromuscular dynamics into the QN-based driver model. The experimental validation and analysis of the proposed model are given in Section III. Our conclusion and a discussion of our future work are presented in Section IV.

II. Driver Model With Neuromuscular Dynamics

A. General Architecture of the Proposed Model

As shown in Fig. 1, the proposed driver model includes two components: the QN-based lateral control model and the neuromuscular system. The inputs of the entire model consist of road information (i.e., the location of the centerline of the road and the boundaries of the road) and vehicle states (i.e., the position, acceleration, and velocity of the vehicle). The output is the actual steering angle acting on the vehicle dynamics model.

The work procedure of the proposed model is as follows. The QN-based driver model first computes a desired steering angle by using the inputs of the whole model. These inputs can be directly obtained from the vehicle and road models without the need for actual image processing of a road scene, but the estimated time for such visual processing was considered in the corresponding visual servers. Then, the desired steering angle is used as the reference input of the neuromuscular system subject to a torque feedback. After that, the neuromuscular system generates the actual steering angle applied to the vehicle model.

B. Queuing Network-Based Driver Model

The queuing network mental architecture represents human information processing as a queuing network system on the basis of neuroscience and psychological findings [10]. It consists of three major parts: servers, entities, and routes. A variety of servers denote different functional units in the human brain that can process entities. Entities represent pieces of to-be-processed information, which travel on routes that connect corresponding servers. More details of the structure, functions, and assumptions of the QN architecture can be found in [10], [13], [14].

In our previous work [1], we have modeled driver lateral control by integrating the preview/predictive model into the QN architecture. The considerations for such combination are as follows. On the one hand, the QN cognitive architecture, as the general framework of the QN-based model, can capture human capacities and limitations. However, the QN cognitive architecture itself does not contain information about how certain driver lateral control activities are implemented, whereas driver preview models can provide the information. On the other hand, driver preview models have no knowledge about what human perceptual and cognitive resources are needed and what the human capabilities and limitations are in performing the various tasks. By combining the QN cognitive architecture and the driver preview models, the QN-based model benefits from both.

However, in the original QN-based driver model, we did not consider the motor sub-network. Thus, the servers related to the motor subnetwork (i.e., Servers W, Y, and Z) of the QN architecture were not included in the architecture of the driver model. Note that Servers W, Y, and Z represented motor program retrieval, assembled, and sent to body parts, respectively.

In this work, to develop the QN-based driver model including neuromuscular dynamics, we expanded the QN architecture of the original QN-based driver lateral control model by including the three motor servers (i.e., Servers W, Y, and Z). Fig. 2 shows the QN architecture of the proposed model with neuromuscular dynamics, which includes ten effective servers and their functions.

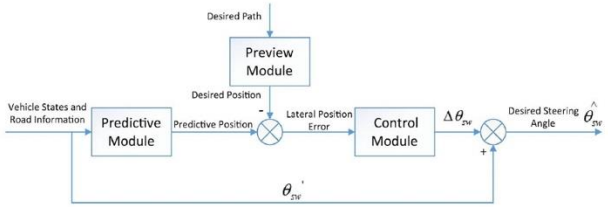


Fig. 3. Block diagram of driver preview model implemented in Server F.

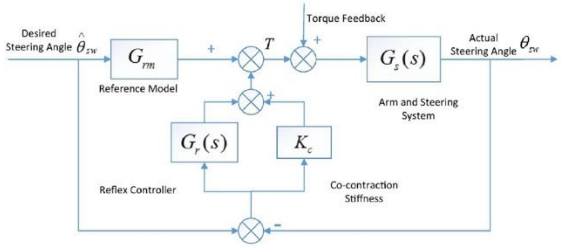


Fig. 4. The structure of the neuromuscular system.

As shown in Fig. 2, entities carrying the inputs of the entire model first enter the visual perceptual subnetwork (Servers 1 (visual input) $\rightarrow$ 2/3 (visual recognition/visual location) $\rightarrow$ 4 (perceptual integrator)). Via Server 4 (Perceptual-Integrator server), the entities are routed to the cognitive subnetwork, including Servers A (visuospatial sketchpad), C (central executor), and F (complex cognitive function), where the desired steering angle is computed according to the preview model, as described in Fig. 3. Then entities carrying the desired steering angle $\hat{\theta}_{sw}$ travel to the motor subnetwork via Server C, where the desired steering angle carried by entities is transported into the neuromuscular system via Server Z.

As shown in Fig. 3, the driver preview model implemented at Server F of the QN architecture contains three main modules: preview module, predictive module, and control module. The preview module previews the desired path for a preview time to obtain the information of the desired path (note that the desired path was predetermined). The predictive module predicts the vehicle response within the preview time by using an internal model (i.e., 3 degree-of-freedom (DOF) vehicle dynamics model, including longitudinal, lateral, and yaw movement).

The control module computes the desired control input (i.e., the increment of steering angle $\Delta\theta_{sw}$) to make a vehicle track the desired path. The calculation proceeds as follows [1]. The desired acceleration a_y (assuming that it is a constant in the preview time) is first calculated by

$$a_y = \frac{2 \times (\Delta E - v \times t_p)}{t_p^2}, \tag{1}$$

where ΔE is the error between the desired lateral position obtained with the desired path and predictive lateral position computed with the internal vehicle dynamics model, v is the current velocity, and t_p is the preview time.

The changed steering angle $\Delta\theta_{sw}$ is then calculated with a proportional derivative (PD) controller of acceleration:

$$\Delta\theta_{sw} = k_p \times a_y + k_d \times a'_y, \tag{2}$$

where k_p and k_d are the coefficients of the PD controller, and a'_y is the first derivative of the acceleration.

Finally, a new steering angle $\hat{\theta}_{sw}$ is computed by

$$\hat{\theta}_{sw} = \Delta\theta_{sw} + \theta'_{sw}, \tag{3}$$

where θ'_{sw} is the steering angle of the last cycle. More details on the QN-based driver lateral control can be found in our previous work [1].

C. Neuromuscular Dynamics

The neuromuscular system (NMS) was developed according to [6] and implemented outside of the QN architecture. The difference between the neuromuscular system in this paper and that of [6] was in the parameter values. The model was used to produce an actual steering angle given the reference input (i.e., the desired steering angle). It can reject any disturbance torque on the steering wheel, to which an external disturbance leads. The structure of NMS used is shown in Fig. 4 [6].

The proposed model includes feedforward module G_{rm}, the co-contraction stiffness K_c, the stretch reflex controller $G_r(S)$, and the arm and steering system $G_s(s)$. G_{rm} represents the angle-torque stiffness, which can provide a steering torque proportional to the desired angle $\hat{\theta}_{sw}$. K_c denotes the increased

TABLE I
BASIC PARAMETERS OF THE QN ARCHITECTURE

Server	Processing time	Capacity
1	E（0.042, 0.025）	4
2	E（0.042, 0.025）	4
3	E（0.042, 0.025）	4
4	E（0.042, 0.025）	5
A	E（0.018, 0.006）	4
C	E（0.018, 0.006）	3
F	E（0.018, 0.006）	1
W	E（0.024, 0.01）	1
Y	E（0.024 0.01）	2
Z	E（0.024, 0.01）	2

intrinsic stiffness of the muscle resulted from muscle activation or co-contraction. The reflex controller $G_r(s)$ is a first-order filter with time delay [6] and can be expressed as

$$G_r = \frac{w_c(B_r s + K_r)}{s + w_c} e^{-\tau s}, \tag{4}$$

where K_r and B_r mean the reflex stiffness and damping, respectively, and depend on the degree of muscle activation and the nature of the control task. w_c represents the cut-off frequency for the first-order filter. The time delay τ is the transport lag in conveying information to and from the motor neurons in the spine. The arm and steering system $G_s(s)$ is assumed to be a linear two-order mass-spring-damper system [6] and can be written as

$$G_s(s) = \frac{1}{Js^2 + Bs + K} \tag{5}$$

where J, B, and K represents the inertia, damping, and stiffness of the arm and steering system, respectively.

D. Parameter Settings

1) Parameters of the QN-Based Lateral Control: Currently, the parameters of the QN framework include server processing time and server capacity, which were set to values from [10], as listed in Table I. The parameters directly related to lateral control (including t_p, k_p, and k_d) were taken values from [1]: $t_p = 1.5$ s, $k_p = 0.008$, and $k_d = -0.02$.

2) Parameters of the NMS: The time delay τ was set to be 0.04 s according to [15] and the cut-off frequency w_c was set to be 30 rad/s according to [6]. The angle-torque stiffness G_{rm} was set to be 0.1 Nm/rad (i.e., the steady-state value of the vehicle dynamics). The reflex stiffness K_r and damping B_r, and the co-contraction stiffness K_c were obtained by using the trial-and-error method based on the experimental data. The

TABLE II
PARAMETERS OF NMS

Steering column inertia	J_{sc}	0.2 kg m^2
Steering column damping	B_{sc}	2.8 N ms/rad
System inertia	J_s	0.01 kg m^2
System damping	B_s	0.2 N ms/rad
Torque bar stiffness	K_t	2 Nm/rad

parameters (i.e., J, B, and K) of the arm and steering system $G_s(s)$ were calculated, respectively, by

$$J = J_{sc} + J_s + J_{arm} \tag{6}$$

$$B = B_{sc} + B_s + B_{arm} \tag{7}$$

$$K = K_t + K_{arm}, \tag{8}$$

where J_{sc} represents steering column inertia, which includes inertias of the steering wheel and rotating column components, J_s is system inertia, B_{sc} is steering column damping, which includes dampings of the steering gear and linkage components, B_s means system damping, K_t is torque bar stiffness, and $J_{arm}, B_{arm}, K_{arm}$ denotes inertia, damping, and stiffness of the arm dynamics, respectively. The parameters (i.e., J_{sc}, J_s, B_{sc}, B_s, and K_t) related to steering system were from Carsim, as listed in Table II. The parameters of the arm dynamic $(J_{arm}, B_{arm}, K_{arm})$ were obtained from [6]: $J_{arm} = 0.064$ kgm^2, $B_{arm} = 0.56$ Nms/rad, and $K_{arm} = 3.8$ Nm/rad. According to (6)–(8), we obtained $J = 0.274$ kgm^2, $B = 3.56$ Nms/rad, and $K = 5.8$ Nm/rad, respectively.

III. MODEL EVALUATION

To validate the proposed driver model, we compared model simulation with the real driver data. The model simulation data included simulation data of the previous QN-based model and the proposed model. The driving behavior was measured by lateral displacement and steering angle. In the whole process of the experiment or simulation, the vehicle speed was kept constant (including 50 km/h or 80 km/h, representing a relatively low speed and a relatively high speed, respectively). Note that 80 km/h or so has been widely used to evaluate vehicle dynamics under the ISO double-lane-changing scenario [16]. All measures used in the following were the average values of all participants. That is to say, we did not investigate the behavior of individual drivers in this paper.

A. Testing Scenario

The scenario we applied is shown in Fig. 5. The width of the lane was 3.0 m and the whole length of the path was 400 m, in which the lane-changing process accounted for 80 m. We assumed that the desired trajectory was predetermined as the centerline of the lane.

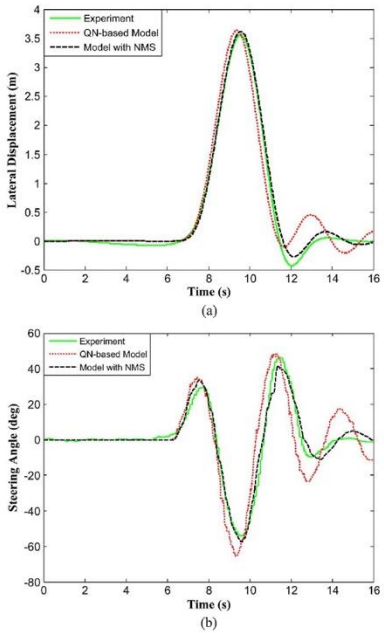


Fig. 7. Comparisons of (a) lateral displacement and (b) steering angle between experiment and the two models simulation at 80 km/h.

D. Performance Analysis of the NMS in Frequency Domain

To analyze the performance of the NMS, a transfer function of the NMS was computed according to Fig. 4, and is written as (9), shown at the bottom of the page.

In order to analyze the frequency response of the NMS system by using the Bode command of Matlab, the reflex delay $e^{-\tau s}$ needs to be simplified as a rational function. Here, we simplified it as a first order system by using Pade approximation

$$e^{-\tau s} = \frac{1 - \frac{\tau}{2}s}{1 + \frac{\tau}{2}s}. \tag{10}$$

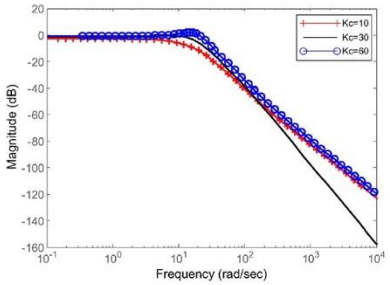


Fig. 8. The Magnitude Bode diagrams for the system with different values of co-contraction stiffness.

By substituting (10) into (9), we obtain the simplified transfer function of the NMS, (11), shown at the bottom of the page.

The magnitude Bode diagrams for the system with various levels of co-contraction stiffness and reflex gain are plotted with the Bode command. As shown in Fig. 8, with the increase of the co-contraction stiffness, the bandwidth of the system rises, making the steering wheel angle θ_{sw} to closely follow the desired angle $\hat{\theta}_{sw}$. As shown in Fig. 9, with the increase of the reflex gain, there is a slight increase in the bandwidth of the systems, producing a slight improvement in the tracking performance.

IV. Discussion and Conclusion

In this paper, we developed a novel driver lateral control model by combining the driver's neuromuscular dynamics with the QN-based driver model. Experimental results from sixteen subjects show that compared to the QN-based model, the proposed model performs better, and its performance is closer to that of drivers when a vehicle runs at a relatively high speed.

Compared to other driver models based on a cognitive architecture, the proposed model can capture the dynamic interaction between the vehicular steering system and the driver's neuromuscular system. In contrast to other driver models with the neuromuscular dynamics, such as [6], the integrated model is based on the QN cognitive architecture. Thus, it has advantages of the QN architecture. The QN cognitive architecture is a mathematical model and particularly suited for real-time

$$G(s) = \frac{(G_r + K_c)s + (G_r + K_c)\omega_c + \omega_c(sB_r + K_r)e^{-\tau s}}{Js^3 + (J\omega_c + B)s^2 + (K + B\omega_c + K_c)s + \omega_c(K + K_c) + \omega_c(sB_r + K_r)e^{-\tau s}}. \tag{9}$$

$$G(s) = \frac{\frac{\tau}{2}(G_r + K_c\text{-}\omega_c B_r)s^2 + [\frac{\tau}{2}\omega_c(G_r + K_c - K_r) + G_r + K_c + \omega_c B_r]s + \omega_c(K_r + G_r + K_c)}{\frac{\tau}{2}Js^4 + [J + \frac{\tau}{2}(J\omega_c + B)]s^3 + [\frac{\tau}{2}(K + B\omega_c + K_c\text{-}\omega_c B_r) + J\omega_c + B]s^2} + \left[\frac{\tau}{2}\omega_c(K + K_c - K_r) + K + B\omega_c + K_c + \omega_c B_r\right]s + \omega_c(K + K_c + K_r). \tag{11}$$

Queuing Network Modeling of Driver EEG Signals-Based Steering Control

Luzheng Bi, Yun Lu, XinAn Fan, Jinling Lian, and Yili Liu

***Abstract*— Directly using brain signals rather than limbs to steer a vehicle may not only help disabled people to control an assistive vehicle, but also provide a complementary means of control for a wider driving community. In this paper, to simulate and predict driver performance in steering a vehicle with brain signals, we propose a driver brain-controlled steering model by combining an extended queuing network-based driver model with a brain–computer interface (BCI) performance model. Experimental results suggest that the proposed driver brain-controlled steering model has performance close to that of real drivers with good performance in brain-controlled driving. The brain-controlled steering model has potential values in helping develop a brain-controlled assistive vehicle. Furthermore, this study provides some insights into the simulation and prediction of the performance of using BCI systems to control other external devices (e.g., mobile robots).**

***Index Terms*— Assistive technology, brain-controlled vehicles, electroencephalography (EEG), queuing network modeling.**

I. Introduction

HEALTHY drivers use limbs to operate a vehicle. Driver models representing this type of driver-vehicle interactions have been widely investigated [1]. Researchers can use these models to simulate and predict driving performance and thus help reduce the time and effort on experimental design and implementation in testing the performance of driver assistance systems [2]. Furthermore, these models can be used to develop adaptive driver assistance systems to better assist driving [3], [4]. Thus, driver models are valuable in helping develop driver assistant systems for improving driving performance and safety.

Some disabled people, however, cannot drive a vehicle with limbs. To help these disabled individuals control vehicles to increase their mobility, brain-controlled vehicles (BCVs) have been proposed. A brain-controlled vehicle (BCV) is a vehicle that is directly controlled by the human "mind" through interpreting brain activity signals into motion commands rather than limbs.

BCVs can be considered as one of the applications of brain–computer interfaces (BCIs). Brain signals used to develop BCIs can be measured and recorded with methods such as functional near-infrared imaging (fNIR), recording electrical or magnetic fields, functional magnetic resonance imaging (fMRI), and positron emission tomography (PET) [5], [6]. The electrical fields can be recorded at the scalp (electroencephalography (EEG) signals), on the cortex (electrocorticographic signals), or within the cortex (neuronal action potentials). Currently, compared to other methods, EEG recording is less expensive, simpler, and non-invasive, and thus more suitable for use in practice. EEG signals have been widely applied to develop various brain-controlled systems, such as brain-controlled cursors [7], [8], brain-controlled prosthesis [9], [10], and brain-controlled wheelchairs [11]–[13]. In contrast to brain-controlled wheelchairs, BCVs are more complicated in dynamic characteristics, and they travel faster in a more complicated environment.

Recently, some researchers have started to explore how to develop a BCV with EEG signals because of the advantages of EEG recordings mentioned above. Gohring *et al.* [14] explored how to employ a commercial BCI product from Emotiv to control an intelligent vehicle. If the intelligent vehicle is about to leave the predefined free driving zone or to collide instantly, it immediately stops itself and resets the BCI system. They tested the system by requiring one participant to finish driving on a closed airfield. The experimental results indicated that it was feasible to use EEG signals in conjunction with an autonomous control system to control a vehicle. In our previous work [15], we have developed a new steady-state visually evoked potential (SSVEP) BCI, whose visual stimuli were presented on a windshield with a head-up display, and used such BCI along with the alpha rhythm of EEG to operate the simulated vehicle supported by a 14-degree of freedom (DOF) vehicle dynamics model. We tested the BCV by asking four participants to finish a driving task online. The experimental results showed the feasibility of only using EEG to continuously steer vehicles. The major difference between [15] and [14] is that we developed a novel BCI and applied it to completely and continuously steer a simulated vehicle without any stops during the whole testing process, whereas [14] applied a BCI product (without reporting the performance of the BCI) to intermittently control an intelligent vehicle, which can stop itself and reset the BCI.

Manuscript received February 25, 2016; revised August 21, 2016; accepted September 24, 2016. Date of publication September 28, 2016; date of current version August 7, 2017. This work was supported by the National Natural Science Foundation of China under Grant 61374192 and Grant 51575048 and the Beijing Natural Science Foundation under Grant 4162055.

L. Bi, Y. Lu, and J. Lian are with the School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081 China. (e-mail: bhxblz@bit.edu.cn; luyun_bit@126.com; lianjinling@bit.edu.cn).

X. A. Fan is with the Beijing Institute of Mechanical Equipment, Beijing 100854 China. (e-mail: anmengxiang@126.com)

Y. Liu is with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: yililiu@umich.edu).

Digital Object Identifier 10.1109/TNSRE.2016.2614003

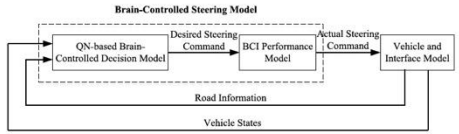


Fig. 1. Brain-controlled steering model.

Although the two studies reported in [14] and [15] indicate the feasibility of brain-controlled driving to some degree, the performance of brain-controlled driving is not good enough. One essential and important method for improving the performance and safety of brain-controlled driving is to raise the performance of BCIs (including increasing the accuracy and number of commands that the BCI can output and decreasing the response time [5]). For example, developing hybrid BCIs is an important avenue to improve the performance of BCIs [7], [16]–[18]. Furthermore, opinions and assessments of people with disability can help better develop BCI systems [19], [20]. Another method is to develop driver assistance systems for brain-controlled driving. As driver models representing driving with limbs have values of importance in helping develop driver assistance systems, modeling driver controlling a vehicle with brain signals (i.e., driver brain-controlled models) have important values in helping develop assistance systems for brain-controlled driving.

However, to the best of our knowledge, no study has explored how to build driver brain-controlled models. Since it is currently only possible for a brain-controlled assistive vehicle to run at a low speed, steering control is more common and important compared to the speed control. In this paper, we explore how to build a driver brain-controlled model in steering control. The main contributions of this paper are twofold: 1) it is the first study to explore how to develop a brain-controlled driving model to simulate and predict the performance of using a BCI to steer a vehicle; 2) it provides some insights into the simulation and prediction of the performance of using BCIs to control other external devices (such as mobile robots).

II. Method

A. General Architecture of the Proposed Model

The basic idea of this paper is to develop the driver brain-controlled model in steering control by combining an extended queuing network (QN)-based driver model with a BCI performance model capturing the performance of BCIs, which can issue three types of steering commands (i.e., going forward and turning right and left). Fig. 1 shows the brain-controlled driving model in steering control, which consists of two models: the QN-based brain-controlled decision and BCI performance models. The inputs to the whole model include vehicle states (i.e., the acceleration, velocity, and position of the vehicle) and road information (i.e., the centerline and boundaries of the road). The output is the actual steering commands. The working process of the proposed model is as follows. First, the QN-based brain-controlled decision model makes a decision of the desired steering command according to vehicle states and road information. Then, the BCI performance model issues the actual steering command given the desired steering command as the input. Finally, the actual steering command acts on the vehicle model through an interface model transforming the actual steering command into the crisp control signal (i.e., steering wheel angle). It should be noted that vehicle states and road information can be directly obtained from the simulated environment with no need for actual image processing of a road scene, but the estimated time of such visual processing was considered in the corresponding visual servers of the QN cognitive architecture.

B. Queuing Network-Based Brain-Controlled Decision Model

The QN-based brain-controlled decision model was developed by extending the QN-based driver model in [21], which represents driver steering control with limbs and was built by fusing the preview model into the QN architecture.

The driver preview model implemented at Server F (for description, see Fig. 3) of the QN architecture includes three modules: the preview, predictive, and control modules, as shown in Fig. 2. The preview module previews the desired path for a preview horizon to determine the desired position. The predictive module calculates the vehicle response within the horizon by using an internal vehicle dynamics model.

The control module calculates the desired steering wheel angle change $\Delta\theta_{SW}$ according to the following procedure [21]. Under the assumption that the desired acceleration a_y is kept constant in the preview horizon, the desired acceleration is first calculated by

$$a_y = \frac{2 \times (\Delta E - v \times t_p)}{t_p^2} \tag{1}$$

where v means the current velocity, t_p denotes the preview horizon, and ΔE stands for the error between the desired lateral position computed with the desired path and predicted lateral position obtained via the internal model.

The steering wheel angle change $\Delta\theta_{SW}$ is then computed with

$$\Delta\theta_{SW} = k_p \times a_y + k_d \times \dot{a}_y \tag{2}$$

where k_p and k_d denotes the coefficients, and $\dot{a}_y$ means the first derivative of the acceleration.

However, the QN-based driver model, which represents the steering control with limbs, cannot be directly applied to the steering control with BCIs, because, compared to the steering control with limbs, the steering control with BCIs has three differences. The first difference is that, currently, the brain-controlled steering can only issue limited number of discrete commands due to the performance constraints of the BCIs. The second one is that the brain-controlled steering does not use limbs. The third one is that for the steering control with BCIs, the parameters, which are directly related to steering control (i.e., t_p, k_p, and k_d), should be different from those of driver models representing driving with limbs.

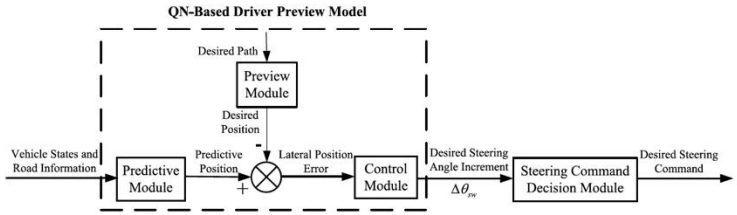


Fig. 2. Block diagram of brain-controlled driver preview model implemented in Server F.

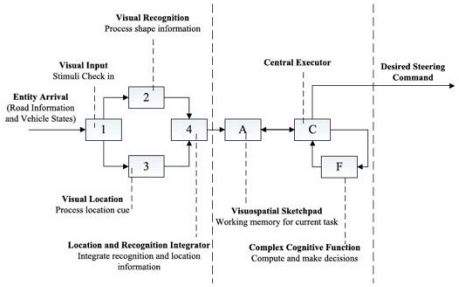


Fig. 3. QN architecture of the proposed brain-controlled decision model.

To extend the QN-based driver model representing steering control with limbs to represent the steering control with BCIs, according to the differences mentioned above, we made the following adjustments in the driver model representing driving with limbs.

First, a steering command decision model, which transforms continuous commands into discrete commands, was added into the QN-based driver model, as shown in Fig. 2. The decision model converts the desired steering wheel angle change $\Delta\theta_{SW}$ into a desired steering command. In this paper, the proposed steering command decision model is as follows:

$$l = \begin{cases} 1, & \Delta\theta_{SW} \in [\theta_{Th}, \infty) \\ 0, & \Delta\theta_{SW} \in (-\theta_{Th}, \theta_{Th}) \\ -1, & \Delta\theta_{SW} \in (-\infty, -\theta_{Th}], \end{cases} \tag{3}$$

where l is the output command of the steering command decision model, with $l = -1$ denoting turning right, $l = 1$ denoting turning left, and $l = 0$ denoting going forward, and θ_{Th} is the threshold which is a positive constant angle. According to (3), if the change of steering wheel angle is less than the threshold θ_{Th}, the output command is non-control (i.e., going forward); otherwise if $\Delta\theta_{SW}$ is greater than the threshold θ_{Th}, the output command is turning left; otherwise, the output is turning right.

Second, we removed the motor subnetwork, which can capture human characteristics in limbs movement, from the QN-based driver model, to obtain the QN architecture of the proposed brain-controlled decision model, as shown in Fig. 3. Entities carrying the inputs of the entire model first enter the visual perceptual subnetwork (Servers 1 (visual input) → 2/3 (visual recognition/visual location) → 4 (perceptual integrator)). Via Server 4 (Perceptual-Integrator server), the entities are then routed to the cognitive subnetwork, including Servers A (visuospatial sketchpad), C (central executor), and F (complex cognitive function). In Server F, the desired change of the steering angle $\Delta\theta_{SW}$ is then computed according to (2) and converted into the desired steering command according to (3). Finally, the desired steering command enters the BCI performance model via Server C. More details about the QN-based driver model representing control with limbs can be seen in our paper [21].

Third, the parameters directly related to lateral control (including the preview time t_p, k_p, and k_d) were set by the trial-and-error method based on the brain-controlled driving data.

C. BCI Performance Model

We aimed at building a model to represent the performance of BCIs that issue three types of commands. The reasons for modeling the performance but not the internal process of the BCI system are followed. First, developing a model to represent the process of the BCI system needs to model the relationship between EEG signals and BCI stimuli or cognitive tasks, which is rather difficult and task-dependent or stimulus-dependent. Second, the BCI performance model can be applied to represent the performance of using other similar interfaces to steer a vehicle.

The BCI performance model, in this paper, was built under the following simplifications: 1) the hit rates (accuracies) of the three kinds of commands of BCIs are equal and constant during the whole process; 2) the probabilities of each class of commands being falsely determined to be other classes are constant and equal. Given such simplifications, the BCI performance model can be expressed as

$$A_i = C \tag{4}$$

$$P_{ij} = \frac{1-C}{N-1}, \quad i \neq j \qquad (i, j \in \{1, 2, 3\}, N = 3, 0 \leq C \leq 1) \tag{5}$$

where A_i is the accuracy of the BCI in ith kind of commands, C is a constant, P_{ij} is the probability of ith class being falsely

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2023.3342151

Modeling Car-Following Behavior of Brain-Control Driving with Queuing Network Architecture

Yun Lu, *Member, IEEE,* Rong Su, *Senior Member, IEEE,* and Yili Liu

***Abstract*—Directly using brain signals to drive vehicles is called brain-control driving, which has the potential to help people with disabilities to acquire driving ability. Developing driver models of the brain-control driving can help understand and simulate the driver behaviors, which contributes to the development of the brain-control driving. In this paper, to simulate the car-following behavior of the brain-control driving, we propose two brain-control car-following models, i.e., CF-BCB and CF-BCI models. The two models are built by integrating a car-following decision model with a brain-control behavior model and a brain-computer interface (BCI) performance model in the queuing network (QN) cognitive architecture, respectively. The manual- and brain-control experiments are conducted to collect the car-following data of the ideal and real brain-control driving, respectively. Simulations with the proposed models are performed to mimic the ideal and real brain-control driving of different subjects. We compare the performance of the car-following decision model and two decision models extended from the intelligent driver model (IDM) and the full velocity difference model (FVDM), respectively. The comparison results show that the car-following decision model performs better than the two decision models in simulating the ideal brain-control driving. Besides, we prove the effectiveness of the proposed models in simulating the car-following behaviors of the brain-control drivers by comparing the simulation and experimental results.**

***Index Terms*—Brain-control driving, car-following behavior, queuing network, driver model.**

I. Introduction

DIRECTLY using brain signals to drive vehicles is called brain-control driving. Unlike the manual-control driving, the brain-control driving does not rely on muscles or peripheral nerves of human drivers. Such driving method, for one thing, has the potential to help people with disabilities to acquire driving ability. For another thing, it can free the limbs of healthy people by offering a new driving approach. The brain-control driving is performed by using brain-computer interfaces (BCIs), which can directly interpret the brain signals of humans into control commands [1]. Electroencephalography (EEG) has become the most popular technique to develop brain-computer interface (BCI) systems due to its low cost, high temporal resolution, and ease of use [2].

Several researches have explored to employ EEG-based BCIs to perform the brain-control driving. Göhring *et al.* [3] provided subjects with a BCI product from Emotive to drive an autonomous vehicle. Users could perform the speed and steering control when the vehicle was in a safe zone. In [4], a motor imagery-based BCI was designed for drivers to control the steering, movement, and engine of a vehicle at the velocity of about 5 km/h. Bi *et al.* [5] have investigated how to use a steady state visually evoked potential (SSVEP) BCI to control the steering of a vehicle with the speed of 2–3 m/s. In [6], we have developed a longitudinal driving system equipped with an SSVEP BCI for drivers to conduct the longitudinal control of a vehicle with the speed being no higher than 10 m/s. Lu and Bi [7] have explored the feasibility of applying an SSVEP-based BCI to perform the combined longitudinal and lateral control of a vehicle.

Owing to the performance limitations (e.g., accuracy limitation) of BCIs, the driving performance of the vehicles relying on BCIs is poor. A few studies have explored to build assistive control methods to improve the driving performance. Our previous works [8] and [9] have designed two shared control methods for the brain-controlled vehicles built in [5] and [6] to improve their lateral and longitudinal driving performance, respectively. Shi *et al.* [10] have built a control strategy by using the model predictive control (MPC) and fuzzy logic to improve the lateral performance of the brain-controlled vehicles. However, even with the assistance of the shared control techniques, the performance of the brain-control driving is not good enough. To further improve its performance, it is significant for vehicle designers and control engineers to understand and simulate the behavior of the brain-control drivers.

Developing driver models can help understand, compute, predict, and simulate the related driver behaviors [11], [12]. In practice, driver models can help increase the development efficiency of vehicle systems by decreasing the requirement to perform driver-in-the-loop experiments [13], [14]. They also contribute to the development of advanced driver assistance systems [15], [16]. Numerous researches have been carries out

Manuscript received January 10, 2023; revised May 20, 2023, August 30, 2023; accepted December 8, 2023. This work was supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-GC-2023-007). *(Corresponding author: Yun Lu)*

Y. Lu and R. Su are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798. (e-mail: yun.lu@ntu.edu.sg; rsu@ntu.edu.sg).

Y. Liu is with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA. (e-mail: yililiu@umich.edu).

output discrete driving commands with limited numbers in the brain-control driving, while they send continuous driving signals in the manual-control driving; 2) to execute their driving intentions, drivers perform brain-control operation via brain signals in the brain-control driving, while they conduct limb-control operation via muscles and peripheral nerves in the manual-control driving.

Since the car-following behavior of the brain-control driving consists of the car-following decision and brain-control operation sub-behaviors, we model it by fusing the models of the two sub-behaviors. As shown in Fig. 2, we first design a car-following decision model to simulate the car-following decision of the brain-control drivers. Then, the brain-control behavior and BCI performance models are applied to simulate the brain-control operation of the users. After that, we build two brain-control car-following models (i.e., CF-BCB and CF-BCI models) by combining the car-following decision model with the brain-control behavior and BCI performance models in the QN architecture, respectively. The QN architecture, including perceptual and cognitive subnetworks, is employed to denote human information processing in performing the brain-control driving.

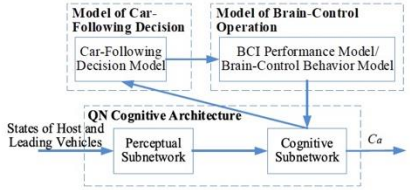


Fig. 2. Architecture of the brain-control car-following models.

Remark 2: The proposed brain-control car-following models work as follows. At first, visual entities of the states of the host and leading vehicles enter the visual perceptual subnetwork and get processed. Then, the processed entities enter the cognitive subnetwork. The car-following decision model and the BCI performance model/the brain-control behavior model conduct calculation so that the entities carry information of the actual acceleration commands. See the following sections for details of the working process.

Remark 3: The considerations for fusing the models of the car-following decision and the brain-control operation into the QN architecture are as follows. The decision and operation models can provide information about how the car-following tasks are performed in the brain-control driving; the QN architecture has knowledge about human limitations and capabilities, along with human cognitive and perceptual resources in performing various tasks. The combined models are expected to simulate the car-following process of the brain-control drivers, while capturing their characteristics in information processing.

B. Car-Following Decision Model

The car-following decision model is developed based on the MPC method, which performs well in the modeling of driving behaviors [13], [17], [18]. In the MPC algorithm, we use the longitudinal models of the host and leading vehicles as prediction models. They enable the car-following decision model to predict the future states of the two vehicles. The cost function incorporates a car-following and a smooth control objective.

1) Vehicle Model

The host vehicle is expected to imperfectly track its desired acceleration. Then, we can describe its longitudinal model with a first-order lag as

$$\tau \dot{a} + a = u \tag{1}$$

where τ denotes the time lag [33], and a and u represent the actual and desired accelerations of the host vehicle, respectively.

Integrating forward difference approximations, the dynamic model in (1) can be rewritten as

$$x(k+1) = Ax(k) + Bu(k),$$
$$A = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & T \\ 0 & 0 & 1-T/\tau \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 0 \\ T/\tau \end{bmatrix}. \tag{2}$$

where T denotes the sampling time and $x(k) = [s(k), v(k), a(k)]^T$ represents the state variable of the host vehicle. s and v are the absolute position and longitudinal velocity of the host vehicle, respectively. The users were allowed to control the host vehicle within a certain velocity range. The velocity constraint can be expressed as

$$v_{\min} \le v \le v_{\max} \tag{3}$$

where $v_{\min}$ and $v_{\max}$ are the limits of the longitudinal velocity.

Assuming the leading vehicle has a constant acceleration in the prediction horizon at each time step [34], we can describe the longitudinal model of the leading vehicle as

$$x_p(k+1) = A_p x_p(k),$$
$$A_p = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

where $x_p = [s_p, v_p, a_p]^T$ represents the state variable of the leading vehicle. s_p, v_p, and a_p are its position, longitudinal velocity, and longitudinal acceleration, respectively.

2) Driving Objective

In the same spirit of [23], the car-following objective is achieved by introducing penalties on the relative velocity between the host and leading vehicles and the deviation from a desired spacing in the cost function. The penalties are formulated as

$$J_C = \sum_{k=0}^{N-1} (Q(S(k) - S_d(k))^2 + R(v_r(k))^2) \tag{5}$$

where $S(k)$ and $S_d(k)$ are the inter-vehicle and desired spacing at time step k, respectively, $v_r(k)$ denotes the relative velocity, Q

output discrete driving commands with limited numbers in the brain-control driving, while they send continuous driving signals in the manual-control driving; 2) to execute their driving intentions, drivers perform brain-control operation via brain signals in the brain-control driving, while they conduct limb-control operation via muscles and peripheral nerves in the manual-control driving.

Since the car-following behavior of the brain-control driving consists of the car-following decision and brain-control operation sub-behaviors, we model it by fusing the models of the two sub-behaviors. As shown in Fig. 2, we first design a car-following decision model to simulate the car-following decision of the brain-control drivers. Then, the brain-control behavior and BCI performance models are applied to simulate the brain-control operation of the users. After that, we build two brain-control car-following models (i.e., CF-BCB and CF-BCI models) by combining the car-following decision model with the brain-control behavior and BCI performance models in the QN architecture, respectively. The QN architecture, including perceptual and cognitive subnetworks, is employed to denote human information processing in performing the brain-control driving.

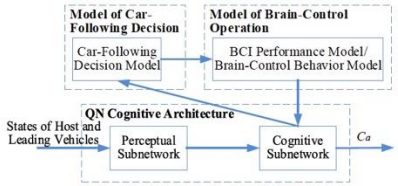


Fig. 2. Architecture of the brain-control car-following models.

Remark 2: The proposed brain-control car-following models work as follows. At first, visual entities of the states of the host and leading vehicles enter the visual perceptual subnetwork and get processed. Then, the processed entities enter the cognitive subnetwork. The car-following decision model and the BCI performance model/the brain-control behavior model conduct calculation so that the entities carry information of the actual acceleration commands. See the following sections for details of the working process.

Remark 3: The considerations for fusing the models of the car-following decision and the brain-control operation into the QN architecture are as follows. The decision and operation models can provide information about how the car-following tasks are performed in the brain-control driving; the QN architecture has knowledge about human limitations and capabilities, along with human cognitive and perceptual resources in performing various tasks. The combined models are expected to simulate the car-following process of the brain-control drivers, while capturing their characteristics in information processing.

B. Car-Following Decision Model

The car-following decision model is developed based on the MPC method, which performs well in the modeling of driving behaviors [13], [17], [18]. In the MPC algorithm, we use the longitudinal models of the host and leading vehicles as prediction models. They enable the car-following decision model to predict the future states of the two vehicles. The cost function incorporates a car-following and a smooth control objective.

1) Vehicle Model

The host vehicle is expected to imperfectly track its desired acceleration. Then, we can describe its longitudinal model with a first-order lag as

$$\tau \dot{a} + a = u \tag{1}$$

where τ denotes the time lag [33], and a and u represent the actual and desired accelerations of the host vehicle, respectively.

Integrating forward difference approximations, the dynamic model in (1) can be rewritten as

$$x(k+1) = Ax(k) + Bu(k),$$
$$A = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & T \\ 0 & 0 & 1-T/\tau \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 0 \\ T/\tau \end{bmatrix}. \tag{2}$$

where T denotes the sampling time and $x(k) = [s(k), v(k), a(k)]^T$ represents the state variable of the host vehicle. s and v are the absolute position and longitudinal velocity of the host vehicle, respectively. The users were allowed to control the host vehicle within a certain velocity range. The velocity constraint can be expressed as

$$v_{\min} \le v \le v_{\max} \tag{3}$$

where $v_{\min}$ and $v_{\max}$ are the limits of the longitudinal velocity.

Assuming the leading vehicle has a constant acceleration in the prediction horizon at each time step [34], we can describe the longitudinal model of the leading vehicle as

$$x_p(k+1) = A_p x_p(k),$$
$$A_p = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

where $x_p = [s_p, v_p, a_p]^T$ represents the state variable of the leading vehicle. s_p, v_p, and a_p are its position, longitudinal velocity, and longitudinal acceleration, respectively.

2) Driving Objective

In the same spirit of [23], the car-following objective is achieved by introducing penalties on the relative velocity between the host and leading vehicles and the deviation from a desired spacing in the cost function. The penalties are formulated as

$$J_C = \sum_{k=0}^{N-1} (Q(S(k) - S_d(k))^2 + R(v_r(k))^2) \tag{5}$$

where $S(k)$ and $S_d(k)$ are the inter-vehicle and desired spacing at time step k, respectively, $v_r(k)$ denotes the relative velocity, Q

and R represent the weights penalizing the spacing deviation and relative velocity, respectively, and N is the control horizon as well as the prediction horizon of the MPC algorithm. Assuming the desired spacing is a linear function of the speed of the host vehicle [23], we can derive

$$S_d(k) = \alpha v(k) + \gamma \tag{6}$$

where α and γ are positive constants.

The smooth control objective is realized by including a penalty on the desired acceleration of the drivers in the cost function, which is described as

$$J_S = \sum_{k=0}^{N-1} P(u(k))^2 \tag{7}$$

where P is the weight penalizing the acceleration.

3) Optimization Problem

Based on the above mentioned descriptions, the optimization algorithm is formulated as

$$\min_{u} \sum_{k=0}^{N-1} (Q(S(k) - \alpha v(k) - \gamma)^2 + R(v_r(k))^2 + P(u(k))^2) \tag{8a}$$

$$\text{s.t. } x(k+1) = Ax(k) + Bu(k) \tag{8b}$$

$$x_p(k+1) = A_p x_p(k) \tag{8c}$$

$$v_{\min} \le v(k) \le v_{\max} \tag{8d}$$

$$k = 0, \cdots, N-1$$

$$x(0) = x_0 \tag{8e}$$

$$x_p(0) = x_{p0} \tag{8f}$$

where x_0 and x_{p0} are the current states of the host and leading vehicles, respectively. In the cost function (8a), the first two terms are applied to implement the car-following objective; the third term is used to realize the smooth control objective. Constraints (8b) and (8c) are the vehicle models. Constraint (8d) limits the longitudinal velocity of the host vehicle. Constraints (8e) and (8f) enable that the vehicle states are initialized as the measured values at the current time instant.

The algorithm (8) can be converted to a convex quadratic programming problem as

$$\min_{z} \ q(z) = \frac{1}{2} z^T H z + z^T c \tag{9a}$$

$$\text{s.t. } Ez \le d \tag{9b}$$

where $z = [u(0), u(1), \ldots, u(N\text{-}1)]$ denotes the optimal variable, $H \in \mathbb{R}^{N \times N}$ is a positive semidefinite matrix, $c \in \mathbb{R}^N$ and $d \in \mathbb{R}^M$ are real vectors, $E \in \mathbb{R}^{M \times N}$ is a real matrix [35]. M is the number of linear inequality constraints, which equals to $2N$ according to the optimization problem (8). The first signal of the solution $u(0)$ is employed to calculate the desired command by using a command decision model, which can be formulated as

$$C_d = \begin{cases} [1, 0, 0]^T, & u(0) \in (a_{Th}, \infty) \\ [0, 1, 0]^T, & u(0) \in [-a_{Th}, a_{Th}] \\ [0, 0, 1]^T, & u(0) \in (-\infty, -a_{Th}) \end{cases} \tag{10}$$

where a_{Th} represents the acceleration threshold, which is a positive constant, and C_d denotes the desired command with $C_d = [0, 0, 1]^T$, $[1, 0, 0]^T$, and $[0, 1, 0]^T$ representing commands of slowing down, speeding up, and maintaining a constant speed, respectively.

Remark 4: Given an initial velocity satisfying the velocity limit, there always exists a longitudinal acceleration sequence enabling the future velocities to be within the velocity limit. That is, the constraints can always be satisfied. Thus, the optimization problem is always solvable.

C. Models of the Brain-Control Operation

The brain-control behavior and BCI performance models are used to simulate the brain-control operation of the drivers. The brain-control behavior model is a process-oriented model, which can help understand the detailed process of the brain-control operation. The BCI performance model is a performance-oriented model, which could better simulate the performance of the brain-control operation.

1) Brain-Control Behavior Model

The brain-control behavior model is developed from the process perspective of the human brain-control operation. Particularly, the brain-control behavior model is made up of an encoding model, a signal acquisition model, a preprocessing model, a feature extraction model, and a classification model [32]. The encoding model is developed to model human activity in generating discriminative EEG signals, which is formulated as

$$F^{(t)} = B_e C_d^{(t)} + \varepsilon_e^{(t)} \tag{11}$$

where $F^{(t)} \in \mathbb{R}^D$ denotes the generated EEG signals, $B_e \in \mathbb{R}^{D \times K}$ is the model parameter, and $\varepsilon_e^{(t)} \sim N(0, e)$ denotes the assumed physiological artifacts, e.g., eye movements. K and D represent the number of the desired commands and the simulated EEG channels, respectively.

The signal acquisition model, modeling real BCIs to collect EEG data with noise signals, is expressed as

$$\hat{F}^{(t)} = F^{(t)} + \varepsilon_s^{(t)} \tag{12}$$

where $\hat{F}^{(t)} \in \mathbb{R}^D$ stands for the measured EEG signals and $\varepsilon_s^{(t)} \sim N(0, s)$ represents the non-biological noise.

Simulating artifact removal process of BCIs for decreasing the noise from EEG signals, the preprocessing model is formulated as

$$\hat{F}_p^{(t)} = B_e C_d^{(t)} + \frac{1}{l}\varepsilon_e^{(t)} + \frac{1}{m}\varepsilon_s^{(t)}, l, m \ge 1 \tag{13}$$

where $\hat{F}_p^{(t)} \in \mathbb{R}^D$ represents the preprocessed EEG signals, and l and m are the model parameters.

The feature extraction model is built to simulate BCIs for extracting features from EEG signals in certain time window, which is expressed as $F_c^{(t)} = [\hat{F}_p^{(t)}, \hat{F}_p^{(t-1)}, \ldots, \hat{F}_p^{(t-n)}]$. We formulate the model as

$$V^{(t)} = F_c^{(t)} B_f \tag{14}$$

Methods-time measurement (MTM)

- Purpose
  - Predict task time

    (overcome a limitation of time study)

- How?
  - Decompose tasks into elemental motions
  - Use pre-determined timing data of the elemental motions to estimate total task time

Pre-determined motion-time data

- E.g. MTM tables

- Time measurement unit (TMU)

  1 / 100,000 of an hour (or 0.036 s)

TABLE I—REACH—R

Distance Moved Inches	Time TMU				Hand in Motion	
	A	B	C or D	E	A	B
¾ or less	2.0	2.0	2.0	2.0	1.6	1.6
1	2.5	2.5	3.6	2.4	2.3	2.3
2	4.0	4.0	5.9	3.8	3.5	2.7
3	5.3	5.3	7.3	5.3	4.5	3.6
4	6.1	6.4	8.4	6.8	4.9	4.3
5	6.5	7.8	9.4	7.4	5.3	5.0
6	7.0	8.6	10.1	8.0	5.7	5.7
7	7.4	9.3	10.8	8.7	6.1	6.5
8	7.9	10.1	11.5	9.3	6.5	7.2
9	8.3	10.8	12.2	9.9	6.9	7.9
10	8.7	11.5	12.9	10.5	7.3	8.6
12	9.6	12.9	14.2	11.8	8.1	10.1
14	10.5	14.4	15.6	13.0	8.9	11.5
16	11.4	15.8	17.0	14.2	9.7	12.9
18	12.3	17.2	18.4	15.5	10.5	14.4
20	13.1	18.6	19.8	16.7	11.3	15.8
22	14.0	20.1	21.2	18.0	12.1	17.3
24	14.9	21.5	22.5	19.2	12.9	18.8
26	15.8	22.9	23.9	20.4	13.7	20.2
28	16.7	24.4	25.3	21.7	14.6	21.7
30	17.5	25.8	26.7	22.9	15.3	23.2

CASE AND DESCRIPTION
A Reach to object in fixed location, or to object in other hand or on which other hand rests.
B Reach to single object in location which may vary slightly from cycle to cycle.
C Reach to object jumbled with other objects in a group so that search and select occur.
D Reach to a very small object or where accurate grasp is required.
E Reach to indefinite location to get hand in position for body balance or next motion or out of way.

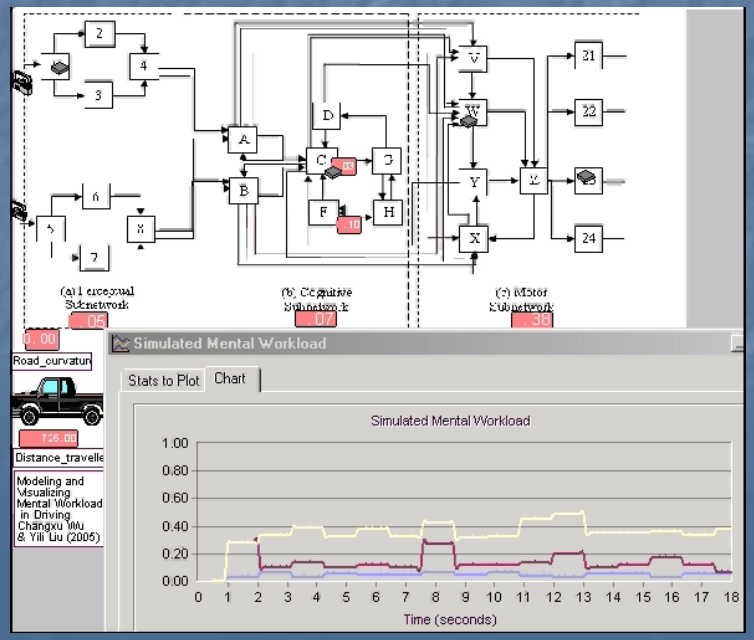
Integration of Cognitive Model with Biomechanical model (QN and HUMOSIM-Jack)

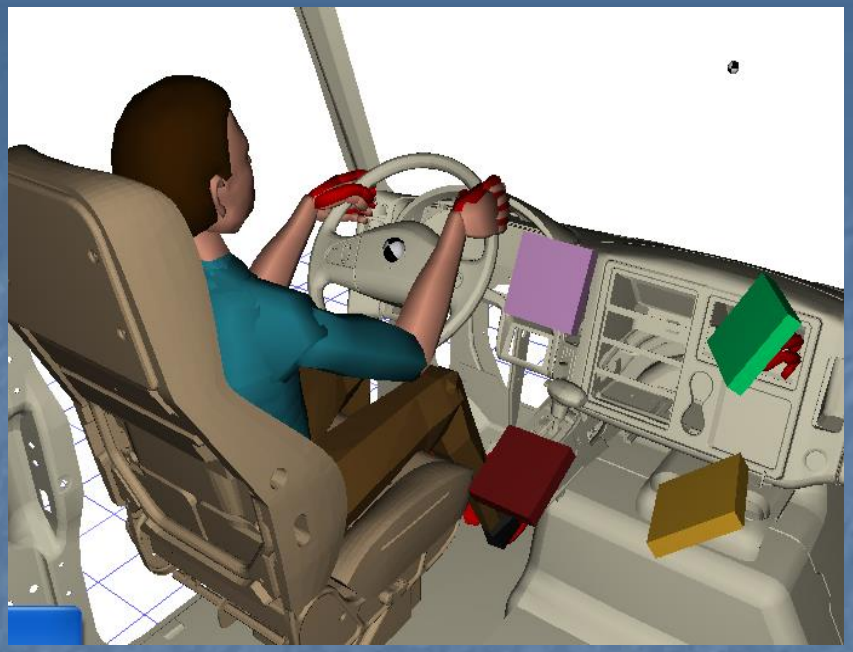
Helen Fuller, Matt Reed, Yili Liu

Supported by Automotive Research Center

Integrated Cognitive-Physical Model of Driving

- Task appears on screen
- Press button to start task
- Perform visual search for matching icons
- Press button to complete task





Near High

Far High

Near Low

Far Low

NeuroImage 59 (2012) 109–116

Contents lists available at ScienceDirect

NeuroImage

journal homepage: www.elsevier.com/locate/ynimg

Computational neuroergonomics

Yili Liu [a,*], Changxu Wu [b], Marc G. Berman [c]

[a] Department of Industrial and Operations Engineering, University of Michigan, USA
[b] Department of Industrial and Systems Engineering, State University of New York at Buffalo, USA
[c] Department of Psychology and Cognitive Neuroscience Lab, University of Michigan, USA

ARTICLE INFO

Article history:
Received 20 February 2011
Revised 4 May 2011
Accepted 8 May 2011
Available online 18 May 2011

ABSTRACT

Neuroergonomics merges neuroscience and ergonomics for the study of brain and behavior in natural and naturalistic settings. Together with the rapid development of neuroergonomics concepts, technologies, and related data, there is an urgent need to develop computational models of neuroergonomics that can help integrate and interpret empirical findings and make predictions for scientific research and practical application. This article discusses the relationship between computational neuroscience and computational neuroergonomics, and describes a queuing network based computational neuroergonomic architecture and its applications. These discussions illustrate the mission and challenges of computational neuroergonomics and future research needs.

Introduction

Neuroergonomics merges neuroscience and ergonomics for the study of brain and behavior at work in relation to every day environments, technologies and settings in the real world (Parasuraman and Rizzo, 2008; Parasuraman and Wilson, 2008). As a rapidly developing field, it is accumulating a large amount of empirical data, developing a fast growing body of concepts and technologies, and opening up new frontiers of theoretical inquiries and practical applications. As in all fields of science and applications, researchers also see the importance and necessity to develop computational models of neuroergonomics in addition to data collection, concept formation, and technology development.

Computational modeling can help researchers and practitioners organize and summarize empirical data, provide unique perspectives to examine and interpret the data, guide and generate hypotheses for new experiments to obtain data in a systematic fashion, and make extrapolations and predictions to situations in which empirical data collection is not feasible or undesirable. Scientifically, computational modeling help advance the understanding of the brain and behavior; practically, it may help guide system design by comparing different design alternatives and predicting human behavior in operational contexts that cannot be easily covered by experimental or simulation studies.

One may ask "don't we have many computational models in neuroscience?" and "can't we simply use them in neuroergonomics?" The answer is that we do have many outstanding computational models in neuroscience, which can be used to inform and inspire computational neuroergonomics models; but in the same sense that neuroscience is not synonymous with neuroergonomics, computational neuroscience is not synonymous with computational neuroergonomics. In the same sense that neuroscience provides a rich soil for neuroergonomics to grow and also benefit from it, computational neuroscience models will support and hopefully benefit from computational neuroergonomics models.

In the following section (A brief summary of some related computational neuroscience models), we first summarize some of the major existing models of computational neuroscience, with a focus on models of neuroimaging data. This summary shows the strengths of these models and their limitations for direct application in neuroergonomics. In the section "A queuing network architecture of computational neuroergonomics", we discuss the unique challenges faced by computational neuroergonomics models, and describe one example of a computational neuroergonomic architecture called the queuing network architecture. In Neuroergonomic applications of the QN architecture section, some example applications of the queuing network architecture in neuroergonomics are described. The last section concludes this article by emphasizing the unique mission of computational neuroergonomics and its relation to computational neuroscience.

A brief summary of some related computational neuroscience models

Computational neuroscience is an extremely active field of research, and many outstanding computational models in neuroscience have been developed, from neural networks models, to statistical regression models, to causal network models, among others. It is neither feasible nor the intention of the present paper to offer a

* Corresponding author at: Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109–2117, USA.
E-mail address: yililiu@umich.edu (Y. Liu).

1053-8119/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.neuroimage.2011.05.027

task, the failed component will return to normal operation. This human-computer-machine system can be modeled as the simple queueing network model shown in Figure 6. For the 3-node (machine, computer, human) closed-series network, the state of the queueing system at any time instant t is a vector p(n1, n2, n3), representing the number of customers at node ni (i=1, 2, 3) at time t. For the system described above, p(n1, n2, n3) means that there are n1 machine components in normal operation, n2 being serviced by the computer, and n3 by the human. The total number of customers (denoted by S) should be a known quantity (e.g., S=k could mean that there is a total of k engines).

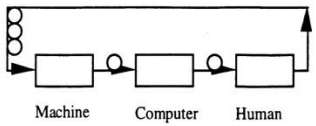


Fig.6. A closed queueing network model of human-computer interaction in a failure management system described in the text

p(n1, n2, n3) can be computed easily with the following set of equations, derived from the results of Jackson (1963) [34]:

$p(n1, n2, n3|S=k) = \omega^*(n1, n2, n3)/T^*(S=k)$;

$\omega^*(n1, n2, n3) = \prod^{3}_{i=1}\prod^{ki}_{j=1}(1/\mu ij)$;

$T^*(S=k) = \sum\omega^*(n1, n2, n3)$, summed over (n1, n2, n3) with S=k;

where μij is the mean service rate of node i when there are j customers at node i. Apparently, the "service rate" of the machine (node 1) is the rate at which it causes machine components to fail. The values for uij are usually obtainable from measurements, specifications or historical data.

The above set of equations allow us to predict a number of interesting performance features of the system. For example, it is easy to compute the proportion of time during which the human operator will have at least one machine component to repair ($\sum p(n1, n2, n3)$, summed over (n1, n2, n3) with n3>0 and S=k), or the proportion of time during which the machine will have at least two components working normally (e.g., at least two engines are running) ($\sum p(n1, n2, n3)$, summed over (n1, n2, n3) with n1>1 and S=k).

We have extended the work to modeling more complicated systems involving more than one humans and more than one computers--a human-computer network. A specific example is a failure management system in which there are two type of machine component failures, each type is handled by a computer and then by a human operator. A possible scenario is that two computers and two human operators work cooperatively in a manner illustrated in Figure 7, where the "copilot" completes his/her task alone with a probability of p, but need to forward the problem to the "pilot" with a probability of (1-p), before the component is returned for normal operation.

In order to compute the queue length distributions, we need the routing probability of the customers--pij, the probability that a machine component will immediately visit node j after departing from node i, which is specified by the task structure. In Figure 7, we have,

p12 = q (the probability that a failure is of type 1),

p13 = 1-q (the probability that a failure is of type 2),

p54 = p (the probability that human operator 2 needs help from human operator 1),

p56 =1 - p (the probability that human operator 2 can complete his/her job alone)

p24 = p35 = p46 = p60 = p01 = 1

pij = 0, for all other i and j's.

The expected value of the number of appearances of node i on a routing is computed with the following recursive equation,

$e_i = p0i + \sum^{i}_{m=1}(e_m\, Pmi)$

With a total of k machine components in the queueing system, p(n1, n2, n3, n4, n5) can be computed easily with the following set of equations, derived from the results of Jackson (1963):

$p(n1, n2, n3, n4, n5|S=k) = \omega^*(n1, n2, n3)/T^*(S=k)$;

$\omega^*(n1, n2, n3, n4, n5) = \prod^{5}_{i=1}\prod^{ki}_{j=1}(e_i/\mu_{ij})$;

$T^*(S=k) = \sum\omega^*(n1, n2, n3, n4, n5)$, summed over (n1, n2, n3, n4, n5) with S=k;

A number of question can be answered with the computed queue length distributional values. The type of questions include the relative workload of operator 1 versus operator 2, the proportion of time during which the machine has at least c components operating normally, and the effects of changing network configuration or service rates.

Although the models are presented in the context of a network of human and computer agents interacting with each other toward a common goal, an area known as computer-supported cooperative work (CSCW). The same methodology can be applied to the broader area of human-computer networks, which also includes situations in which competitive or confrontive agents may compete with each other for

limited network resources and cause delays in servicing other agents' processing needs.

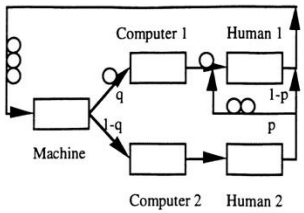


Fig. 7. A queueing network model of a human-computer network in the failure management system described in the text

Although a multitude of human-computer networking tools and CSCW applications have been developed, there is a substantial lack of predictive models and theories. As Schneiderman (1992) pointed out, this is a "vast uncharted territory: theories are sparse, measurement is informal, data analysis is overwhelming, and predictive models are nonexistent" ([35], p.391). The model presented in this section illustrates that queueing network methods could serve as a useful tool for establishing performance theories and predictive models of human-computer networks and for establishing theory-guided, systematic ways of performance measurement and analysis, particularly the issues of concern involve timing, scheduling and resource allocation.

The models presented in Figures 6 and 7 are currently being evaluated with lab experiments using a simulated failure management system and human subjects. We are also in the process of preparing experiments to validate a model of human-computer network with competing agents.

We hope that this article has illustrated the potential power of queueing network methods in establishing new models of human cognition, human performance and human-computer interaction on various analysis levels, and in establishing an integrated, computational framework for unifying some currently isolated models.

REFERENCES

[1] R. Disney and D. Konig, "Queueing networks: A survey of their random processes," SIAM Review, vol-27, 335-403, 1985.

[2] L. Kleinrock, "Queueing Systems," New York: Wiley, 1975.

[3] Y. Liu, "Visual scanning, memory scanning, and computational human performance modeling," Proc. of the Human Factors Society 37th Annual Meeting, 1993.

[4] Y. Liu, "A queueing network model of human multi-task performance," Tech. Rep. Univ. of Michigan, Dept. of IOE. 1993.

[5] J. Jackson, "Networks of waiting lines," Oper. Res., vol-5, pp.518-521, 1957.

[6] O. Boxma and H. Daduna, "Sojourn times in queueing networks," In H. Takagi (Ed.), Stochastic Analysis of Computer and Communications Systems, p.401-450, 1990, North Holland.

[7] J. Buzen, "Fundamental operational laws of computer system performance," Acta Informatica, vol-7, pp.167-182, 1976.

[8] P. Denning and J. Buzen, "The operational analysis of queueing network models," Computing Surveys, vol-10, pp.225-261, 1978.

[9] S. Sternberg, "The discovery of processing stages: Extensions of Donders's method," Acta Psychologica, 30, p.276-235, 1969.

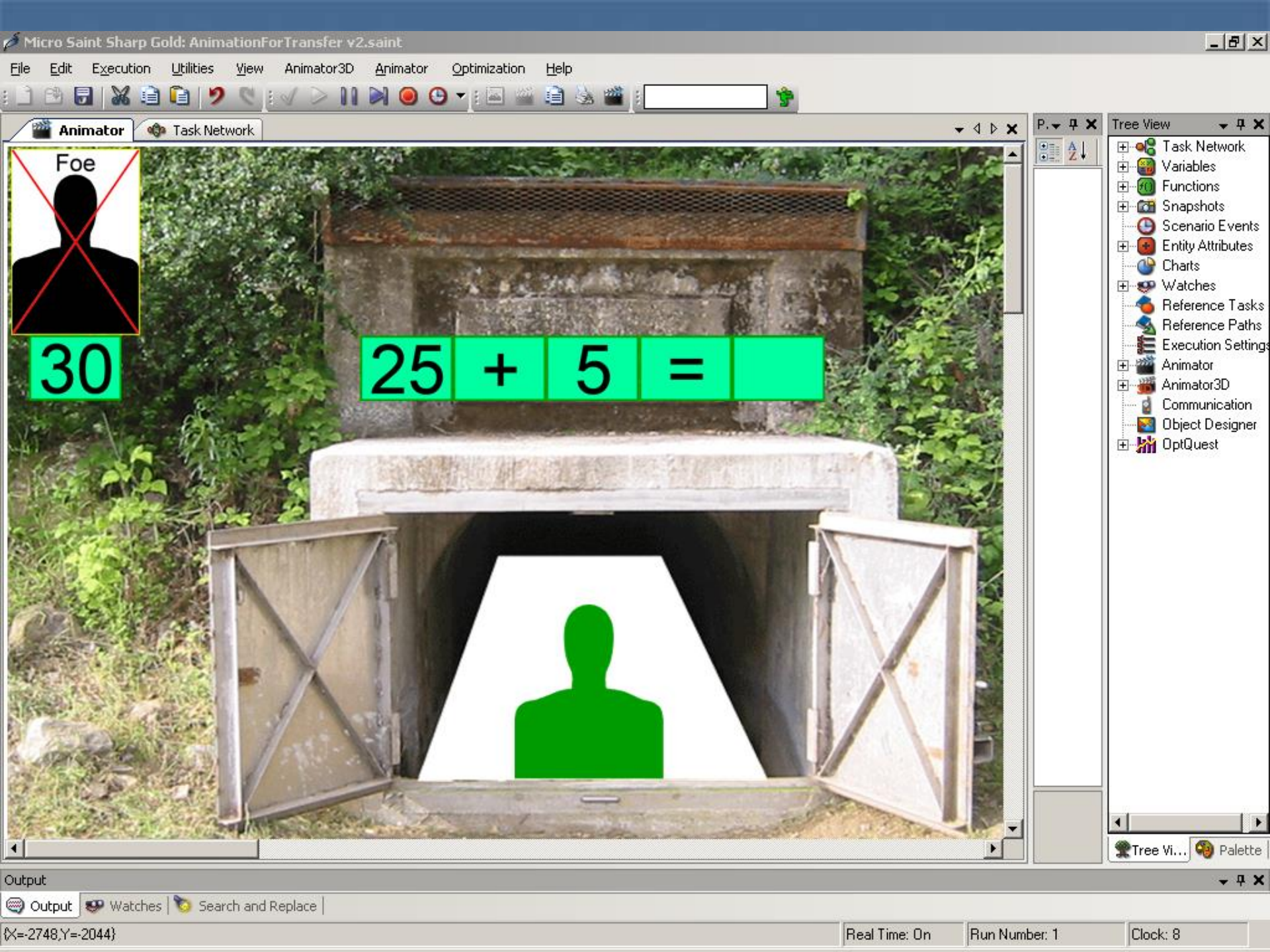
[10] R. Pachella, "The interpretation of reaction time in information processing research," In B. Kantowitz (Ed.), Human information processing: Tutorials in Performance and Cognition, p.41-82, 1974, Hillsdale, N.J.: Erlbaum.

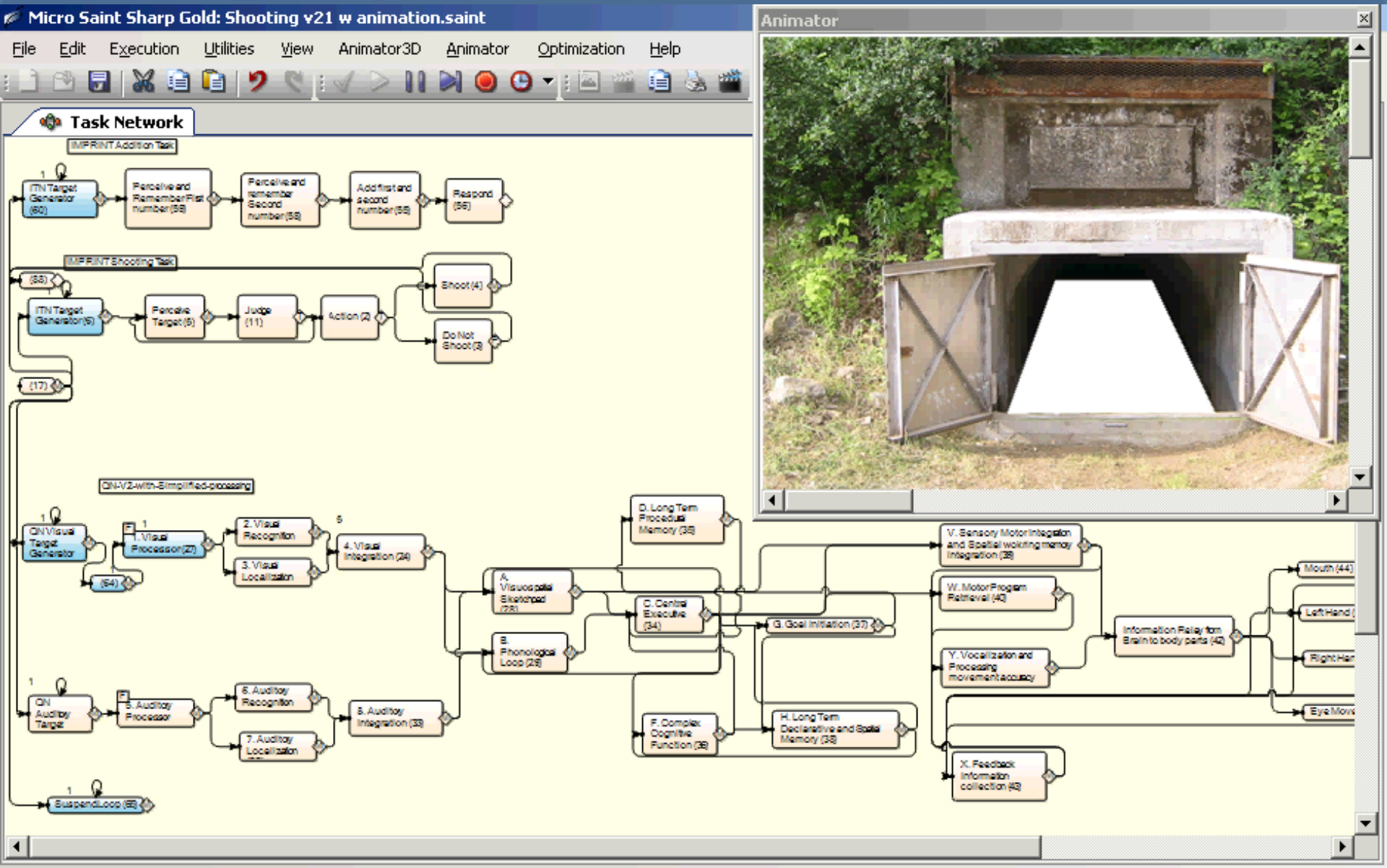
[11] W. McGill and J. Gibbon, "The general-gamma distribution and reaction times," J. of Math. Psych., 2, 1-18, 1965.

[12] J. McClelland, "On the time relations of mental processes: An examination of systems of processes in cascade," Psychological Review, 86, 287-330.

[13] J. Miller, "A queue-series model for reaction time, with discrete-stage and continuous-flow models as special cases," Psychological Review, 100, 702-715, 1993.

[14] J. Townsend and F. Ashby, "The Stochastic Modeling of Elementary Psychological Processes," Cambridge: Cambridge Univ. Press, 1983.


Micro Saint Sharp Gold: AnimationForTransfer v2.saint
File
Edit
Execution
Utilities
View
Animator3D
Animator
Optimization
Help
Animator
Task Network
Foe
30
25 + 5 =
Tree View
Task Network
Variables
Functions
Snapshots
Scenario Events
Entity Attributes
Charts
Watches
Reference Tasks
Reference Paths
Execution Settings
Animator
Animator3D
Communication
Object Designer
OptQuest
Tree Vi...
Palette
Output
Output
Watches
Search and Replace
{X=-2748,Y=-2044}
Real Time: On
Run Number: 1
Clock: 8


Micro Saint Sharp Gold: Shooting v21 w animation.saint
File
Edit
Execution
Utilities
View
Animator3D
Animator
Optimization
Help
Task Network
IMPRINT Addition Task
ITN Target Generator (60)
Perceive and Remember First number (58)
Perceive and remember Second number (58)
Add first and second number (58)
Respond (56)
IMPRINT Shooting Task
ITN Target Generator (5)
Perceive Target (5)
Judge (11)
Action (2)
Shoot (4)
Do Not Shoot (3)
CN-V2-with-Simplified-processing
CN Visual Target Generator
1. Visual Processor (27)
2. Visual Recognition
3. Visual Localization
4. Visual Integration (24)
CN Auditory Target
5. Auditory Processor
6. Auditory Recognition
7. Auditory Localization
8. Auditory Integration (33)
A. Visuospatial Sketchpad (28)
B. Phonological Loop (29)
C. Central Executive (34)
D. Long Term Procedural Memory (35)
F. Complex Cognitive Function (36)
G. Goal Initiation (37)
H. Long Term Declarative and Spatial Memory (38)
V. Sensory Motor Integration and Spatial working memory Integration (39)
W. Motor Program Retrieval (40)
Y. Vocalization and Processing movement accuracy
X. Feedback Information collection (43)
Information Relay fom Brain to body parts (42)
Mouth (44)
Left Hand
Right Han
Eye Move
SuspendLoop (88)
Animator

(a) Tuning radio using the knob

(b) A zoom-in view of the physical panel

Figure 31. Task Procedure using the knob: (1) press the power button, (2) press the AM/FM button, (3) turn the knob to decrease or increase the frequency shown on the display ("590" as in the picture)

(a) Tuning radio on the touch screen

(b) Click the "Entertainment" button

(c) Click the "FM" then "Direct Tune" button

(d) Enter the radio frequency

Figure 32. Task procedure using the virtual buttons

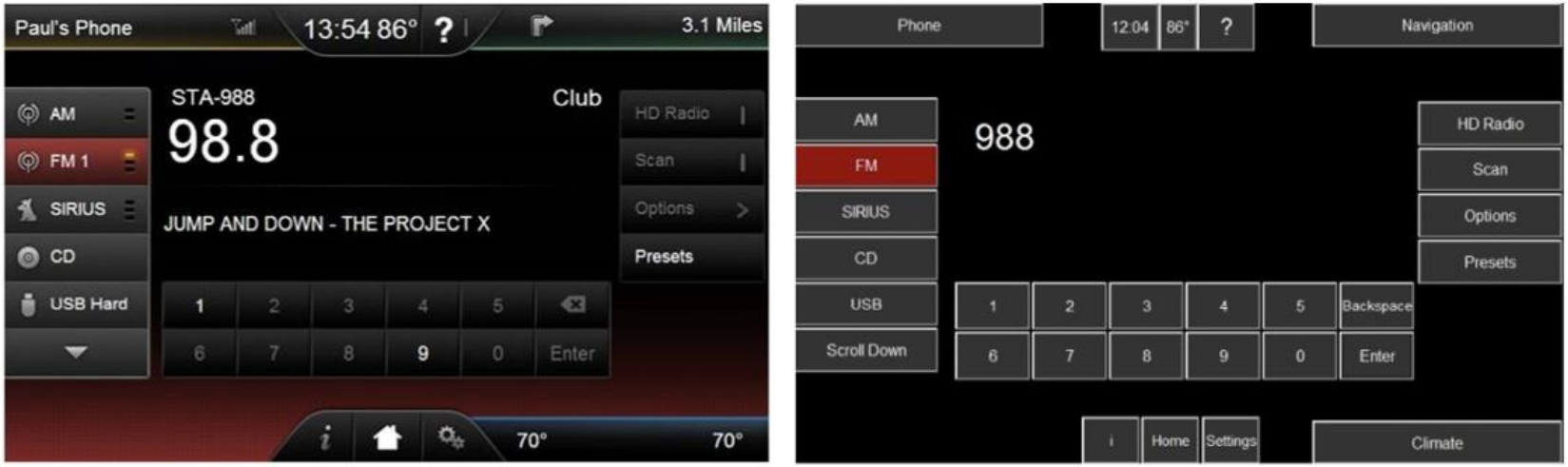


Fig. 9. Touch screen UI used in the experiment (left) and its digital mockup (right).

Fig. 10. Physical panel used in the experiment (left) and its digital mockup (right).

of each session the participants were asked to rate their workload in a scale from 0 to 10 using the standard NASA-Task Load Index (NASA-TLX, Hart & Staveland, 1988). A full factorial within-subject design was used in this experiment. Two independent variables (with 2 and 3 levels respectively) and 3 replications for each combination give a total of 18 ($=2\times3\times3$) trials for each participant.

2.2.3. Procedures

Once the participants arrived at the laboratory and completed the consent form, they were given an introduction using slides to ensure that they understood the tasks they were about to perform. This was followed by a practice session for both the driving and radio-tuning tasks. Then the data collection started. Counterbalancing was used for controlling the order effects of the within-subject design. 10 (50%) participants started with the parked condition, while 10 (50%) started with the driving condition. 10 (50%) participants started with using the touch screen, while 10 (50%) participants started with the physical panel (five of which started with the method of using the physical buttons. And the other five started with the method of using the knob).

2.3. Task modeling

2.3.1. Radio-tuning task

Digital mockups of the touch screen and the physical panel that were used in the experiment were firstly created using MATLAB GUIDE. Figs. 9 and 10 show the real devices used in the experiment and the digital mockups that would be used in the software simulation.

To simulate the radio-tuning task, A NGOMSL-style task analysis was conducted. Fig. 11 shows the NGOMSL-style task analysis of tuning the radio using the physical buttons.

The task analyses of tuning the radio using the knob and touch screen follow a similar format. However, there are differences on the sequential dependency of the task components (TCs) due to the inherent characteristics of the devices. For example, clicking a series of virtual buttons on a touch screen without tactile feedback requires the driver's visual attention to confirm that the button is successfully clicked (by using visual cues of, for example, a change of the button's color) before he/she can start searching for the next button. And in some cases, the next button may not appear until the previous button is clicked (e.g., the 'FM' button in the entertainment screen appears only after the 'Entertainment' button is clicked). For the physical buttons on the other hand, searching for the next button could start as soon as the previous button is reached by the hand, assuming that clicking on a physical button could be confirmed by the tactile feedback without visual attention. We do assume that the driver needs the visual guidance for reaching the button (i.e., no 'touch typing'), so searching for the next button could start as early as the button is reached by hand, but not earlier. Fig. 12 illustrates the sequential dependency of the radio tuning task using the physical buttons and a touch screen. It can be observed that the task using the physical buttons can be performed in a more parallel manner (top figure) compared with the task using a touch screen (bottom figure).

2.3.2. Multitasking modeling

Many visual-manual tasks (e.g., tuning the radio stations) are composed of multiple steps that altogether take more than a few seconds to complete even as a single task (i.e., without driving).

Fig. 9. Touch screen UI used in the experiment (left) and its digital mockup (right).

Fig. 10. Physical panel used in the experiment (left) and its digital mockup (right).

Harper, 2007). This scenario used the same DISALT shooting testbed and provided behavioural data to model the shooting task. Production rules were defined to model the shooting-only task as the process of visual searching, manual aiming, and pulling the trigger.

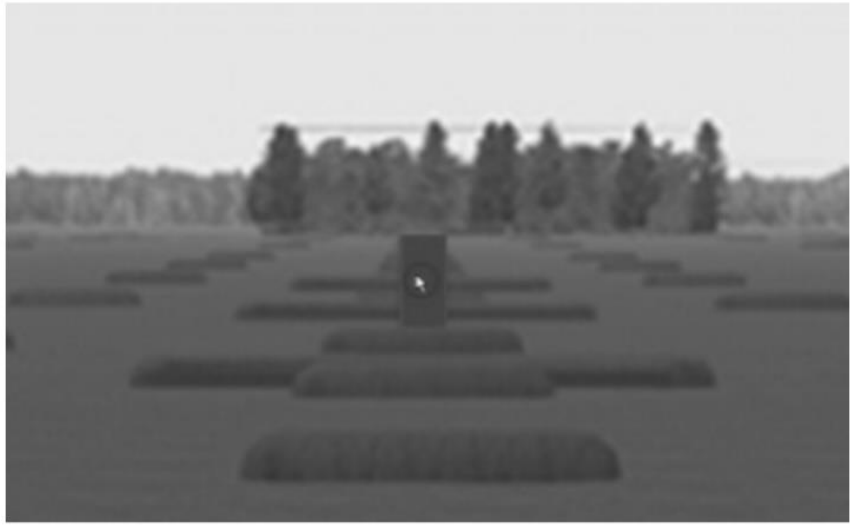


Figure 4 Visualization of the task environment with an enemy-target in QN-ACTR. The cursor represents the iron sight aiming. Background picture from (Scribner et al., 2007).

QN-Model of Human Behavior (QN-MHB)
Journal Publications
by Yili Liu and his collaborators

For a list of about 30 Journal articles, please visit:

https://websites.umich.edu/~yililiu/pubs.html

Thank you!