with his or her preferences for the trade-offs between customer satisfaction, monetary return, and preferences for risk aversion, the decisionmaker can evaluate the alternatives included in the Pareto optimal set. This information provides support and justification for why the chosen solution is the optimal one.

The second advantage is the value of incorporating probability distributions and extreme events into business analysis problems. Since there exist so many unknowns concerning potential costs and revenues, using distributions and considering what will happen in extreme cases allows the problem to become manageable and solvable (i.e., large potential losses).

Appendix

The following tables present the Pareto optimal solutions for other scenarios considered. Tables A-I and A-II present the Pareto optimal solutions for a rural area where the telephone company can provide video services in two years and eight years, respectively. Tables A-III–A-V present the Pareto optimal solutions for an average area where the telephone company can provide video services in five years, two years, and eight years, respectively. Finally, Tables A-IV–A-VIII present the Pareto optimal solutions for an urban area where the telephone company can provide video services in five years, two years, and eight years, respectively.

Acknowledgment

The authors greatly appreciate the contributions of E. Dudley of GTE to this study.

References

[1] E. Asbeck and Y. Y. Haimes, "The partitioned multiobjective risk method," *Large Scale Systems*, vol. 6, pp. 13–38, 1984.
[2] V. Chankong and Y. Y. Haimes, *Multiobjective Decision Making: Theory and Methodology*. New York: North-Holland, 1983.
[3] J. L. Corner and C. W. Kirkwood, "Decision analysis applications in operations research literature, 1970–1989," *Oper. Res.*, vol. 39, pp. 206–219, 1991.
[4] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*. Kensington, CA: Cole, 1987.
[5] R. Dillon, "The application of risk analysis and multiobjective decision trees to policy decisions for GTE of Virginia," Master's Thesis, University of Virginia, Charlottesville, 1993.
[6] Y. Y. Haimes, notes from a presentation titled, "Quantification of risk of cost overrun, time delay, and not meeting performance specifications," presented in Washington, DC, Sept. 22–24, 1992.
[7] ——, "Total risk management," *Risk Analysis*, vol. 11, No. 2, pp. 169–171, 1991.
[8] Y. Y. Haimes, J. Lambert, and D. Li, "Risk of extreme events in a multiobjective framework," *Water Resources Bulletin*, vol. 38, no. 1, pp. 201–209, Feb. 1992.
[9] Y. Y. Haimes, D. Li, and V. Tulsiani, "Multiobjective decision-tree analysis," *Risk Analysis*, vol. 10, No. 1, pp. 111–129, 1990.
[10] D. L. Keefer and S. E. Bodliy, "Three-point approximation for continuous random varibles," *Mgmt. Sci.*, vol. 29, pp. 595–609, 1983.
[11] R. L. Keeney and D. von Winterfeldt, "Eliciting probabilites from experts in complex techincal problems," *IEEE Trans. Eng. Manag.*, vol. 38, pp. 191–201, 1991.
[12] R. L. Keeney and H. Raiffa, *Decisions With Multiple Objectives*. New York: Wiley, 1976.
[13] A. Law, "Statistical analysis of simulation output data," *Oper. Res.*, vol. 31, pp. 983–1029, 1983.
[14] A. Law and W. D. Kelton, *Simulation Modeling and Analysis*. New York: McGraw-Hill, 1991.
[15] S. Martin, *Industrial Economics*. New York: Macmillan, 1988.
[16] M. W. Merkhofer, "Quantifying judgmental uncertainty: Methodology, experiences, and insights," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, pp. 741–752, 1987.
[17] T. Pratt and B. Charles, *Satellite Communications*. New York: Wiley, 1986.
[18] P. D. Reed, *Residential Fiberoptic Networks*. Boston: Artech, 1992.
[19] C. S. Spetzler and C. A. S. Staël von Holstien, "Probability encoding in decision analysis," *Mgmt. Sci.*, vol. 22, pp. 340–358, 1975.
[20] D. von Winterfeldt and W. Edwards, *Decision Analysis and Behavioral Research*. Cambridge, UK: Cambridge Univ. Press, 1986.

A Metric for Quantifying Response Time in a Browser Application

D. J. Dailey and J. F. Brinkley

***Abstract*—This paper presents to a performance quantification methodology for browser applications that operate in a distributed computing environment and are used in a nondeterministic way. It asserts that usability is intimately tied to response time, and that a single metric to quantify response time can be developed using probabilistic arguments. This metric incorporates both the delays inherent in the operation of a distributed application, and the observed frequency of each delay in order to create a single value that represents the responsiveness of the application.**

I. Introduction

Interest in applications that access information over the Internet is growing, as demonstrated by the many browser-like implementations such as Wais, Internet Relay Chat, Xmosaic, medical applications in the areas of telemedicine and teleradiology, as well as distance learning programs [2]. One of the factors that can diminish the usefulness of these network based programs is the delay introduced when clients access information servers over the Internet. Usability analysis of distributed computing applications, such as distance learning, imply some unique considerations with respect to response time. Previous modeling work has focused on models of the users and the user interaction with the application [5], [6], this paper models the responsiveness of applications that access data over a network.

In this paper we assert that system response is intimately tied to usability, and that the system response of a distributed application depends upon several variables. The primary purpose of this paper is to present a tool that quantifies response time considerations for distributed computing applications. This tool provides a metric that, in conjunction with qualitative analysis, can be used in the arena of usability testing to provide a quantification of the quality responsiveness in the framework of a distributed application that uses information resources through a network.

In addition this paper demonstrates this tool by applying it to quantify delays in a distributed application used for teaching at the University of Washington medical school. This application is a real world example of client-server computing for distance learning.

Manuscript received July 5, 1994; revised March 18, 1995.
D. J. Dailey is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: dailey@ee.washington.edu).
J. F. Brinkley is with the Department of Biological Structure, University of Washington, Seattle, WA 98195 USA.
Publisher Item Identifier S 1083-4427(96)01401-4.

To develop the performance metric tool we begin by asserting several postulates.

Postulate #1: To study the usability of a particular application in local verses network modes of operation, it is necessary to quantify both 1) the usage pattern, and 2) the delays introduced by the computing hardware and the network access to data.

Postulate #2: A measure that quantifies the response time of an application that is used nondeterministically must include a notion of how the application is used. For example, users may be willing to wait quite a while for certain infrequent actions such as initialization but will be quite frustrated if an action that is frequently repeated introduces a significant delay.

Postulate #3: There are two basic types of delays in distributed applications that affect responsiveness: 1) those that arise from accessing the network for information (labeled *Network* for the purposes of this paper) and 2) those that depend principally on the speed of the computing equipment in use (labeled CPU for the purposes of this paper).

These postulates in conjunction with the application's operational behavior provide the framework in which our metric is developed. This framework is described in the next section.

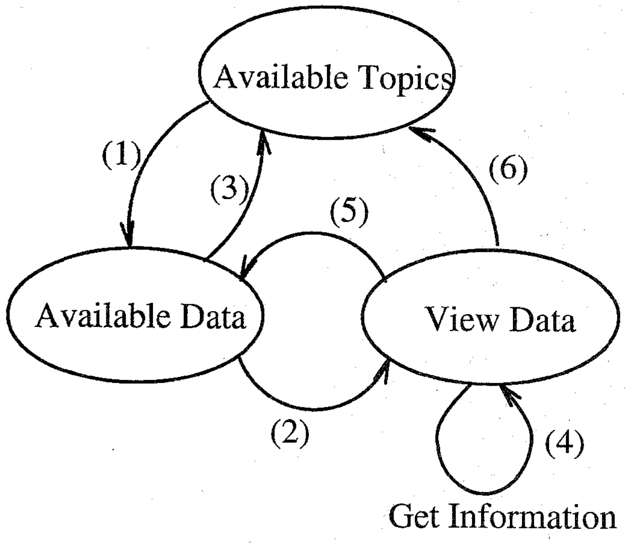

Fig. 1. Application usage.

II. Application Operation

The development of a performance metric requires that the operational behavior of an application be specified. We focus on applications where the user is presented with a screen of information and has the immediate options of: 1) making a transition to a new screen/state, or 2) requesting additional information about the present screen.

Examples of applications that are appropriate for our metric are the numerous tools that fall into the category of a "browser." Such tools are often the interface for databases of information on a variety of topics [7]. In this manuscript we consider browser-like applications that retrieve images or large data sets and display this data graphically. The user then requests additional information about the data presented on the screen (often by using a pointing device to select the area of the screen on which additional information is desired). The interaction of the user with the browser takes place in several steps, each of which causes a change in state of the application.

The overall operational taxonomy combines a set of user actions with the notion of state (see Fig. 1). There are six user actions: (1) Select a topic, (2) Select an image/data set, (3) Return to subject list, (4) Get more information about the present image or data set, (5) Return to data selection screen, (6) Return to topic selection screen.

There are three states: (1) a screen presenting a list of topics available from which the user can select, often textual and indicated by **Available Topics** in Fig. 1, (2) a screen allowing the selection of a specific list of data or images available for retrieval, often composed of text, icons or reduced images, represented by **Available Data** in Fig. 1, and (3) a screen displaying the selected image or data set, often graphical with hot spots or regions which allow the user to get more information about the material displayed, indicated by **View Data** in Fig. 1. The user actions induce the state transitions indicated by the numbered arrows in Fig. 1.

In a session (the duration that a single user is operating the application), the user selects from the available subject list some number of times; having done so he/she select the particular image/data to be retrieved from that subject group. The process is repeated for various data items over the course of the session. Thus for a group of students using such an application, the usage pattern will vary from student to student and from session to session.

III. Performance Metric

The goal of our performance evaluation methodology is to develop a single number (or metric) that takes into account not only the mean delay time for various possible user actions, but also the probability that a user will choose each action. Since the particular sequence of actions taken by the user cannot be predicted in advance, and since network load will cause delay times that vary from one session to the next, the metric is presented as a probabilistic measurement over multiple sessions.

Our general approach is to estimate mean values for each of the time delay components (T_j) associated with each possible user action and from these estimate a mean total delay $(\hat{T})$ which is our performance metric. The performance metric is estimated using,

$$\hat{T} = \sum_{j=1}^{M} p(a_j)T_j \tag{1}$$

where $p(a_j)$ is the probability of action j (of M possible actions), and (T_j) is the mean value for the jth delay.

The probabilities are used as weights to reflect the assertion that the overall metric should not be unduly influenced by actions that are performed only rarely, even if those actions have a long delay. The next sections describe the methodology for estimating the components of this metric.

A. Probabilities of Actions

As suggested above, the usage pattern of a browser application, in terms of the sequence and frequency of actions taken, will vary by user. Our quantification of browser performance is designed to represent some "typical usage". This typical usage is a probabilistic representation of the usage patterns and would not necessarily duplicate any one person's particular experience. The particular statistics needed for our performance metric are: 1) the probabilities for the actions taken by the user of the browser and 2) the probabilities associated with the likelihood of selecting particular images and textual information. These probabilities can be obtained by observing users over and extended period of time and then be used to weight the delays as in (1).

During usage, the probability for each of the state transitions is based on the frequency of occurrence of each action. The frequency interpretation of the probability for the occurrence of action (a_i) of the N available actions each having been observed m_i times is,

$$p(a_i) = \frac{m_i}{\sum_{i=1}^{N} m_i}. \tag{2}$$

While the probabilities for state transition are in reality conditioned upon being in the present state, the delay introduced depends only on that state transition, and hence we model the probability weights as independent.

The probabilities just developed are used as weights for the observed delays.[1] This balances the effect of the absolute delays against the frequency of such delays. Since the delays are observed random variables we represent the absolute delays as mean values calculated from observing the application operation. The next sections discuss our approach to quantifying the delays, both CPU and *Network*.

B. Mean Delays

Our overall performance metric combines the probabilities developed in the last section with measured quantities. To quantify responsiveness we need to define observed rates and delays.

One of the principle delays in a distributed application is the transfer of data or large files across the network[2]. In the case of file transfer across the network, the observables are the size of the file (f_i) and the rate at which data is transferred across the network (r^f) in bytes per second. However depending on the usage pattern, different files will be transferred. We use both the absolute file size in bytes and the probability associated with requesting the each file to arrive at a "typical file size". The "typical file size" is calculated,[3]

$$\bar{f} = \sum_{i=1}^{N} p(f_i) f_i \tag{3}$$

where $p(f_i)$ is the probability of selecting the ith file, and the mean rate is established by observing the sample mean of a number of data transfers,

$$\bar{r}_f = \frac{1}{N} \sum_{i=1}^{N} r_i^f. \tag{4}$$

These values $\bar{f}$ and $\bar{r}_f$ are combined to get a mean delay time,

$$t_f = \frac{\bar{f}}{\bar{r}^f}. \tag{5}$$

This presumes a linear relationship between the size of the file and the time it takes to transfer the file. Testing over a range of file sizes supports the validity of this hypothesis [4].

C. CPU and Network Delays

The time delay penalty can be divided into two major categories: CPU delays $(\hat{T}^c)$ and *Network* delays $(\hat{T}^n)$

$$\hat{T} = \hat{T}^c + \hat{T}^n. \tag{6}$$

The delays that are heavily dependent on the network performance are assigned to the *Network* category, and those that depend primarily on local computing system performance are assigned to the CPU category. In reality each *Network* operation must have some CPU overhead in addition to the network activity but we are assuming that *Network* delays are dominated by the network performance. Using this notion with (1) the total delay is composed of the probability weighted CPU and *Network* components,

$$\hat{T} = \sum_{i=1}^{M} p(a_i) T_i^c + \sum_{j=1}^{N} p(a_j) T_j^n. \tag{7}$$

The two terms on the right hand side of (7) quantify the CPU and *Network* components of responsiveness. The notion of the CPU and *Network* delays allows a quantitative comparison of the effect of network distance[4] and computing power. Since we assert that the responsiveness aspect of usability is proportional to the available computing power and inversely proportional to the network distance the trade off between these two is important in overall application usability.

The next section illustrates the methodology suggested using a distributed application.

IV. Demonstration

To demonstrate our methodology we apply it to an application developed at the University of Washington (UW). This application is a browser for neuroanatomy education and is called "The Digital Anatomist Browser" [3]. The Browser is currently used as an image-based reference atlas for neuroanatomy. Students pick from a list of subjects, each of which consists of a series of image frames, usually depicting a set of serial sections through an anatomical region of the brain. Associated with each image is a set of contours depicting active areas on the image. When the user clicks on an active area the computer displays the name of the object, as well as any textual descriptions about the object. The Browser can also quiz the student, asking him or her to point on the screen to named objects. All information utilized by the Browser is stored on the server, and is sent over the network to the client, which for this study was a Macintosh program written in Supercard.

The performance results in this paper were obtained using local clients at the UW, and a remote client at the National Library of Medicine (NLM), Bethesda, MD.

At the University of Washington the users were students in a neuroanatomy class during one 10 week quarter. Users at NLM were staff who were demonstrating the Browser at the NLM Teaching Learning Center. Because of the difference in user population the usage probabilities were obtained from the UW students and applied to the delays measured both locally and at NLM, in order to more realistically simulate expected usage in a class situation at both sites.

The actions taken by students using the Browser are shown in Table I. The mix of the three actions enumerated in Table I have probabilities for each action: 1) Choose subject $p(a_1)$, 2) Choose frame $p(a_2)$, and 3) Choose structure $p(a_3)$. These quantify the probability of students taking each of the three actions identified in Table I during the course of a typical session. Values for these probabilities were obtained from a data set that included 5058 accesses of images, each of which retrieved one of the 105 images available. The action probabilities are used with the file sizes and contour sizes to produce mean file and contour size values as shown in (3).

The delays inherent in the Browser are CPU and *Network* in nature. The CPU time delay is composed of:

1) Operating system overhead on contour retrieval (C_o) whose average value is obtained in performance tests.
2) Operating system overhead on frame acquisition (F_o) whose average value is obtained in performance tests.

[1] See (1).

[2] We use the term file to mean either files or large data sets for the remainder of this paper.

[3] We approximate the mean value by $\hat{x} = \sum_{i=1}^{N} x_i p(x_i) \approx \int x p(x)\, dx$ [1].

[4] Related work by the authors has shown that network delay and network distance have a linear relationship see [4].

TABLE I
ACTIONS (ACT), VARIABLES (VAR) AND ASSOCIATED DELAYS

Act	Description	Var	Type
a_1	**Choose subject** - This is the selection of the subject area to be considered. This is principally a dynamic type of action that has an observable network delay.	S_c	*Net*
a_2	**Choose frame** - This is the selection of the specific image to be viewed. This action contains both static and dynamic delays delays associated with:		
	Retrieval of image information.		
	Retrieve image size and filename.	I_s	*Net*
	Retrieve contour structure names.	t_n	*Net*
	Retrieval of image:		
	File transfer	t_f	*Net*
	Image retrieval overhead	I_o	*CPU*
	Retrieval of contours:		
	Transfer shapes (xy coordinates)	t_s	*Net*
	Contour retrieval overhead	C_o	*CPU*
	Frame overhead	F_o	*CPU*
a_3	**Choose structure** - This is the selection of individual structures on the anatomical slide presented. A delay is introduced by the retrieval of information about the selected structure.	G_d	*Net*

3) Image overhead (I_o) whose average value is obtained in performance tests.

These delays are weighted by the probability of the "choose frame" action that initiates these functions so that,

$$\hat{T}^c = (C_o + F_o + I_o)p(a_2) \tag{8}$$

is the CPU delay metric.

The *Network* delay is composed of:

1) The delay due to the retrieval of structure names (t_n).
2) The delay associated with the retrieval of contour outlines (t_s).
3) The delay due to image file transfers (t_f).
4) The delay in retrieving image size and filename (I_s).
5) The time to get descriptive information for the material in the current frame (G_d).
6) The time delay (S_c) introduced by the "choose subject" activity.

The *Network* delays above are weighted with the probability for the action that precipitates the delay and then summed to calculate the overall *Network* delay,

$$\hat{T}^n = (t_n + t_s + t_f + I_s)p(a_2) + S_c p(a_1) + G_d p(a_3). \tag{9}$$

The sum of the CPU delays (8) and *Network* delays (9) is a metric for responsiveness. Separating the delay into CPU and *Network* components allows the relative effect of network delays to be quantified.

To demonstrate the metric just presented, we instrumented the Browser and recorded the time for each of the values in Table I for three different situations: 1) the Browser operating locally on a Macintosh Quadra, 2) the Browser operating at the NLM using a Macintosh Quadra, and 3) the Browser operating locally using an older Macintosh IIX. These three simple cases allow the trade off between CPU power and network distance to be discussed in a quantitative manner.

The numerical results are presented in Table II. The first column identifies the type and name for each delay. The second column indicates the action with which the delay is associated. The third column lists the values for the probability of the actions $p(a_i)$ that precipitate the delays. The next three columns show the measured mean delays. The values for the measured mean delays are in seconds and are unweighted by the probabilities. The CPU ($\hat{T}^c$), *Network* ($\hat{T}^n$), and overall ($\hat{T}$) delay metric for the two sites and two CPU's are shown in bold in Table II. These overall values reflect the combination of delay and frequency of occurrence.

TABLE II
DELAY RESULTS

				Location/Computer		
DelayType		a_i	$p(a_i)$	UW/Quadra	NLM/Quadra	UW/Mac IIX
Static Delays						
Contour Overhead	C_o	a_2	0.14	1.42	1.38	6.32
Frame Overhead	F_o	a_2	0.14	1.09	1.18	2.94
Image Overhead	I_o	a_2	0.14	0.46	0.80	5.45
Static Delay	$\hat{T}^S$			**0.416**	**0.475**	**2.06**
Dynamic Delays						
Get Descriptions	G_d	a_3	0.83	0.74	1.50	1.69
Choose Subject	S_c	a_1	0.025	0.45	1.07	1.12
File Transfer	t_f	a_2	0.14	1.37	29.81	6.9
Get Names	t_n	a_2	0.14	0.34	0.48	0.48
Get Shapes	t_s	a_2	0.14	1.27	2.94	2.06
Image Setup	I_s	a_2	0.14	0.45	1.21	1.50
Dynamic Delay	$\hat{T}^D$			**1.10**	**6.08**	**2.96**
Delay Metric	$\hat{T}$			**1.52**	**6.55**	**4.96**

For two Macintosh Quadra CPU's of essentially the same speed located at different sites on the Internet the CPU delays ($\hat{T}^c$) are nearly equal but the *Network* delays ($\hat{T}^n$) are vastly different. The CPU delays on the slower Macintosh IIX computer are a factor of four larger than those on the faster CPU. However, the *Network* delays are also larger for the slower computer at the UW site. This behavior is expected since it is difficult, in the case of the *Network* delays, to separate the CPU speed effect from the network performance.

V. DISCUSSION

The value of our overall metric is that it provides a quantitative comparison of the responsiveness of the Browser when operating on different platforms and at different locations. The values from Table II suggest that the overall responsiveness on a slower local machine is similar to a faster CPU that uses the Internet to obtain the structural information from the other side of the country.

A questionnaire we are developing will ask students to qualitatively rate the acceptability of the perceived delay. Correlations between these qualitative assessments and the measured metric will allow us to determine an acceptable threshold value for Browser responsiveness.

VI. CONCLUSION

This paper has presented a tool that quantifies the responsiveness of applications operating in a client server mode and transferring data over the Internet. The quantification accounts for the nondeterministic use of distributed applications by weighting delays probabilistically. This metric is suitable for examining the tradeoff between CPU speed and network delay inherent in retrieving data. It is also appropriate as a quantitative foundation on which qualitative usability analysis can be built.

REFERENCES

[1] J. S. Bendat and A. G. Piersol, *Random Data: Analysis and Measurement Procedures*, 2d ed. New York: Wiley, 1986.
[2] E. Braun, *The Internet Directory*, 1st ed. New York: Fawcett Columbine, 1994.
[3] J. F. Brinkley, K. Eno, and J. W. Sundsten, "Knowledge-based client-server approach to structural information retrieval: The digital anatomist browser," *Computer Methods and Programs in Biomedicine*, vol. 40, pp. 131–145, 1993.
[4] D. J. Dailey, K. E. Eno, G. L. Zick, and J. F. Brinkley, "A network model for wide area access to structural information," in *17th Symposium on Computer Applications in Medical Care*, SCAMC, 1993, pp. 497–501.
[5] L. Gugerty, "The use of analytical models in human-computer-interface design," *Int. J. Man-Machine Studies*, vol. 38, pp. 625–660, 1993.

[6] R. Holcomb and A. L. Tharp, "Users, a software usability model and product evaluation," *Interacting with Computers*, vol. 3, pp. 155–166, 1991.
[7] B. Kahle, H. Morris, F. Davis, K. Tiene, C. Hart, and R. Palmer, "Wide area information servers: An executive information system for unstructured files," *Electron. Networking*, vol. 2, pp. 59–68, 1992.

A Model of the Human Smooth Pursuit System Based on an Unsupervised Adaptive Controller

Phil W. Koken, Harm J. J. Jonker, and Casper J. Erkelens

***Abstract*—A first attempt was made, on the basis of Pavel's proposal [18], to integrate a predictive mechanism into a model of the human smooth pursuit system. The predictive mechanism contained an adaptive filter based on the LMS algorithm. Smooth pursuit simulations were made of a large variety of target movements. The model provides a fairly good qualitative and mostly also a fairly good quantitative description of human tracking of the various stimuli. However, when the model was applied to the tracking of sinusoidal target movements with frequencies higher than about 1 Hz, it performed even better than the human smooth pursuit system.**

I. Introduction

It has been known for a long time that the human smooth pursuit system is able to predict periodic target motion. In [8] it is noted that sometimes the eye movements changed direction before the target did and the investigators referred to the phenomenon as "anticipatory reversals". Other early investigators incorporated a predictive mechanism in their models, e.g., [7], [25], [29].

These and most of the other existing models simulate human smooth pursuit eye movements quite well, but only for single sinusoidal target movements. All models have in common that they include two parallel channels: one containing a predictive (feedforward) mechanism and another via which purely retinal information is processed (feedback). If the predictive mechanism does not perform adequately, it will switch itself off, whereupon the eyes move purely on the basis of retinal information. However, it is not at all clear how the predictive channel is switched on and off. Switching between two channels that proces retinal information continuously or discretely, respectively, [4] remains unclear too; the 'recognition' of the type of stimulus motion by an adaptive controller [1], [2] remains obscure, which makes this type of models rather unsatisfactory.

Several studies [15], [16] have made it clear that such models cannot really describe human pursuit; a predictive mechanism operates all the time, irrespective of whether the output is adequate or not. Therefore the human pursuit system can better be simulated by a single control loop containing a predictive mechanism [12], [14], [24]. It is not yet clear what the principle is on which the predictive mechanism is based.

Manuscript received July 20, 1994; revised March 11, 1995. This work was supported by the Netherlands Organization for Scientific Research (NWO) through the Foundation for Biophysics.

The authors are with the Vakgroep Fysica van de Mens, Helmholtz Instituut, Universiteit Utrecht, 3584 CC Utrecht, The Netherlands (e-mail: koken@fys.ruu.nl).

Publisher Item Identifier S 1083-4427(96)01402-6.

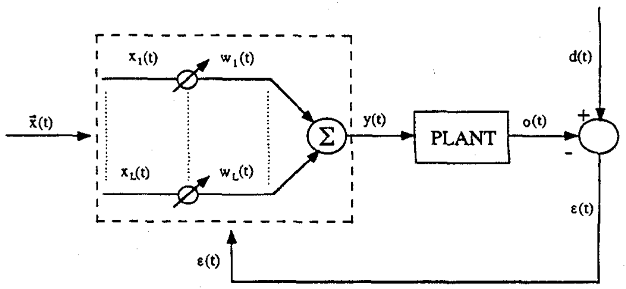

Fig. 1. Model A of human smooth pursuit. An adaptive filter learns via the LMS-algorithm with inputs $\vec{x}(t)$ and error $\epsilon(t)$. The output $y(t)$ of the filter is fed to the plant which consists of a second-order low-pass filter and a pure time-delay. The output of the plant $o(t)$ is substracted from the desired signal $d(t)$.

In [18] it is proposed that an adaptive filter based on the Least Mean Square (LMS) algorithm as introduced in [27], could serve as the predictive mechanism. One advantage of such a mechanism is that the output of a filter based on this algorithm can change very fast. Rapid adaptation to changed conditions is essential to describe the results obtained from various transition experiments in which a sinusoidal target motion unexpectedly changed into another sinusoidal target motion (see [12]).

This study is a first attempt in working out the consequences of Pavel's neural network model [18] regarding the modeling of human smooth pursuit. We examined the model performance for a variety of stimulus motions:

- tracking of single sinusoidal target movements,
- tracking of pseudo-random target movements,
- tracking of single sinusoidal target movements during which a transition from one sinusoidal motion to another one occurs unexpectedly,
- and tracking of pseudo-random target movements during which foveal stabilization occurs unexpectedly, i.e., retinal error and slip become zero.

Smooth pursuit simulations are compared to human data. We also discuss the model with respect to physiological properties of the human smooth pursuit system.

II. Model of Human Smooth Pursuit

A. General Definitions

Fig. 1 shows the proposed model of the human smooth pursuit system, including the adaptive filter which serves as the predictive mechanism and is based on the LMS-algorithm. Such an algorithm minimizes the mean-square error [27]. The error signal ϵ is defined as the difference between the desired output signal d and the actual output signal o of the system (all signals are time-dependent). The plant consists of a pure time delay $\tau = 150$ ms, due to the processing time in the smooth pursuit system, and includes the dynamics of the eyeball and its musculature, which can be described by a second-order low-pass filter [26]. The transfer function of the plant in the Laplace domain (s represents the Laplace operator) is given by

$$G(s) = \frac{e^{-s\tau}}{(s\tau_1 + 1)(s\tau_2 + 1)}.$$