This is to certify that the

dissertation entitled

AN INTERFACE FOR
HUMAN-COMPUTER INTERACTION
BASED ON FACE FEATURE TRACKING
IN 2D

presented by

VERA BAKIC

has been accepted towards fulfillment
of the requirements for

PhD degree in Computer Science
& Engineering

G C Stockman
Major professor

Date 28 Sep 2000

DATE DUE	DATE DUE	DATE DUE
OCT 1 5 2003		

AN INTERFAC

BASED O

in

Departm

An Interface for Human-Computer Interaction Based on Face Feature Tracking in 2D

By

Vera Bakić

A Dissertation

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Department of Computer Science and Engineering

2000

AN INTERFACE FO

F

[illegible]

Abstract

An Interface for Human-Computer Interaction Based on Face Feature Tracking in 2D

By

Vera Bakić

In this dissertation, a new approach to control a computer using head movements, gaze direction and face expressions is discussed. A workstation user is observed in a non-intrusive fashion. Algorithms to find a user's face and where the user is looking at the workstation display are described. The user can issue a command to the computer by making a certain facial expression. The following modules are described: *(i) face detection* that is based on a skin-color model and motion, *(ii) face feature location* that is based on the knowledge of the face geometry, *(iii) gaze location detection* that is based on an Artificial Neural Network mapping of face feature location to display location and a mapping of mouse movement to display movement, and *(iv) selection* that is done through face expressions. In a computer implementation, real-time rates are achieved, and between 10 and 30 frames per second can be processed.

Essentially, a handless Human-Computer Interaction (HCI) is provided that can have various applications. Disabled people who cannot use their hands can greatly

[illegible] from such as H-1

[illegible] and mouse. Fi[illegible]

[illegible] improving and the

[illegible] visual perc[illegible]

[illegible]

The results of a [illegible]

reported in this [illegible]

respect to the new [illegible]

[illegible] grams. A[illegible]

[illegible] several W[illegible]

[illegible] of time.

benefit from such an HCI. Other users can use the interface as a replacement for a conventional mouse. Finally, the interface can be adapted to monitor where the user is looking in time, and the results could be used for various analysis, e.g., in cognitive science for visual perception analysis, in marketing for display exploration analysis, etc.

The results of a study that evaluated the usability of the head-eye input interface are presented in the final Chapter. These results show that the users were able to adjust to the new interface and perform a range of tasks common for current application programs. Among other tasks, the users successfully controlled Netscape: they browsed several WWW pages with an average of 5 erroneous selections and 8:35 minutes of time.

To My Family

[illegible]

Dr. George St... an...
of PhD research. I a...
[illegible] supervision [illegible]

Dr. Frank B...
[illegible] and [illegible]
[illegible] and Dr. St...
[illegible] suggest...

Dr. Charles O...
[illegible] Depart...

Dr. Richard E...
[illegible] for g...
[illegible] and S...
[illegible] M...

ACKNOWLEDGMENTS

This thesis would not be possible without the help and support of many people:

Dr. George Stockman offered excellent guidance, advice and support throughout my PhD research. I am thankful to him for the inspiration to pursue the degree in the computer vision field.

Dr. Frank Biocca from Telecommunication Department served in my guidance committee and gave many valuable advices for my work. Dr. Anil Jain, Dr. John Weng and Dr. Sridhar Mahadevan served in my guidance committee and made many valuable suggestions and comments.

Dr. Charles Owen helped with his comments. Dr. John Henderson from the Psychology Department helped with his expertise in the eye-tracking research.

Dr. Richard Enbody was my advisor during the Masters degree program and I thank him for guidance and support. I would like to thank Dr. Alvin Smucker from the Plant and Soil Science Department and Dr. Theresa Bernardo from the School of Veterinary Medicine for their support during various research projects.

I would like to thank

[illegible] and [illegible]

Lisa Moore in partic

I would like to than

[illegible] for their he

[illegible] lab for all their

[illegible] Paul All[illegible]

[illegible]

[illegible]

Special thanks

[illegible] and

Finally I would

[illegible] in part

[illegible]

I would like to thank all the other professors at the CSE department for excelent teaching and providing research facilities. I would like to thank all the CSE staff, Linda Moore in particular, for all their help.

I would like to thank CSE undergraduate students Kristina Johnson and Brian Langner for their help during CSE Open House demos; fellow students from the PRIP lab for all their help: Aditya Vailaya, Salil Prabhaka, Wey Shuan Huang, Arun Ross, Paul Albee, Scott Connell, Nicolae Duta, Friederike Griess, Karissa Miller, Dan Gutchess, Erin McGarrity, and many others. Jay Newell and Jerry Roll from the Telecommunication Department helped with the "eyes-on-screen" experiments.

Special thanks to all the people who participated in the experiments. Without their help and cooperation this work would not be possible.

Finally, I would like to thank my family for their continuing support and help, my mother in particular for helping out in taking care of my daughter while I was finishing the dissertation.

Table of Contents

List of Tables

LIST OF FIGURES

[illegible] Task T2, subj

each button.

[illegible] Task T2, subj

each button.

[illegible] Task T2, subj

each button.

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, subj

each butt

[illegible] Task T2, sub

each butt

[illegible] Task T2, sub

each butt

[illegible] Task T2, su

each but

[illegible] Task T2, su

each but

[illegible] Task T2, s

each but

[illegible] Task T3, s

each tar

[illegible] Task T3, s

each tar

[illegible] Task T3, s

each tar

[illegible] Task T3. su

each tar

[illegible] Task T3. su

each tar

[illegible] Task T3. su

each tar

[illegible] Task T3. su

each tar

[illegible] Task T3. su

each tar

[illegible] Task T3. su

each tar

[illegible] Task T3. s

each tar

[illegible] Task T3. s

each tar

[illegible] Task T3. s

each tar

[illegible] Task T3. s

each tar

[illegible] Task T3. s

each tar

[illegible] Task T3. s

each tar

[illegible] Task T3. s

each tar

[illegible] Task T3. s

each tar

[illegible] Task T3. s

each tar

[illegible] Task T3. s

each tar

Chapter 1

Introduct

In this thesis, a [illegible]

[illegible] is presented. [illegible]

[illegible] in 2D and [illegible]

[illegible]-error [illegible]

[illegible] a human [illegible]

[illegible]

[illegible] of an ex[illegible]

In the proposed [illegible]

the user is move[illegible]

[illegible] position, by [illegible]

[illegible] all novel, [illegible]

[illegible]

[illegible] to perform [illegible]

Chapter 1

Introduction

In this thesis, a computer vision based interface for Human-Computer Interaction (HCI) is presented. The interface is based on the tracking of a human face and facial features in 2D and offers handless, non-intrusive and inexpensive HCI in a moderately controlled environment. We present the computer vision algorithms used to locate and track a human face and facial features, we propose a gaze determination scheme used to control the cursor and HCI based on the interface, and, finally, we present the results of an evaluation study of the usability and performance of the interface.

In the proposed interface, the human face is tracked and based on the face motion the cursor is moved. We do not determine the cursor position based on the absolute head position, but rather the user is *driving* the cursor position with head movements. As with all novel interfaces, the use of our interface needs to be learned since most users do not think of their head and eyes as an input interface, and the users need to learn to perform controlled head movements to achieve the desired cursor movements.

1.1 Applica

Computers are bec [illegible]

[illegible] common and [illegible]

[illegible] studies [illegible]

[illegible]

[illegible] import [illegible]

[illegible] growth the [illegible]

[illegible]

[illegible]

[illegible]

Recently, m [illegible]

[illegible]

[illegible]

[illegible]

[illegible] a com [illegible]

1.2 Prol

Problem Stat

[illegible]

[illegible] means of

[illegible]

[illegible]

1.1 Applications of Computer Vision

Computers are becoming part of our daily life, and interaction with them has become a common and natural task for people. The field of Human-Computer Interaction *(HCI)* studies issues that have emerged with the use of computers, and how to make the interaction as seamless as possible. Mainly, HCI deals with how to make better mechanical input devices, how to organize displays, how to organize information and dialogs with the user, and similar issues. Some effort has been made in using natural means of communication (e.g., vision and/or speech) to enhance HCI. This was used mainly for interfaces for handicapped users, and these input devices were cumbersome and hard to use.

Recently, more attention has been devoted to the use of vision in the interaction with computers. The goal is to be able to understand the user better and to offer a more friendly and natural working environment. The use of vision offers the ability to see the user's face, understand the user's mood, enable the user to look somewhere and issue a command using a facial expression, etc.

1.2 Problem Statement and Motivation

Problem Statement: The goal of this research has been to develop a vision-based HCI that observes the user and understands the user's gaze and facial expressions as a means of communication with the computer. The computer vision side of the work involves the general capability to detect and track a human face as it moves in a 3D workspace. The use of such a system as an HCI would be twofold: (i) to

provide an input into
workspace and an
documents.

Imagine the follo
the import
the user looks at
system starts up.

This scenario has
that there is no
this would be
such systems is th
the user.

1.2.1 Huma

In the early days
systems. They w
and laboratory
with the machin
the level
[illegible]
[illegible]

provide an input interface, and (ii) to enable an observation of user's movements in a 3D workspace and an evaluation of how humans explore computer displays or virtual environments.

Imagine the following scenario: user enters a room, walks to a computer display, the computer recognizes who the user is and automatically opens the user's workspace, the user looks at an icon, signals with a facial expression to open it and the desired program starts up.

This scenario has all the actions from today's state-of-the-art systems, with a twist that there is no username/password typing, and no mouse pointing and clicking. All this would be done through the use of computer vision systems. The key point in such systems is that they should be non-intrusive and not require a special set-up for the user.

1.2.1 Human-Computer Interaction

In the early days of computing, computers were "magical" machines, "don't touch" systems. They were operated by a few trained people in white coats and kept in a strict laboratory environment. There was not much room for any friendly interaction with the machines, and these issues were not on the agenda at all. Even though people were developing artificial intelligence programs, these were not used to enhance the interaction with the machine, but rather to create a machine that will reason and understand scenes as humans do.

The increased use
and the machines we
by trained exp
be made simpler a
because certain ty
size of the [illegible]
put of data and th
interpretation of
were used to se

Engineers using
environment. H
Computers are stil
But there is no ef

Apart from util
of voice recogn
be used as an inp
a replacement f
[illegible]. Many s
[illegible] 19, 6, 3.
Use of these syst
be cumbersome a
There are health

The increased use of personal computers (PCs) brought more personal interaction and the machines were more accessible to the individual user. Since the users were not only trained experts, but also non-experts and even little childern, the interfaces had to be made simpler and easier to use. Still, the users had to spend time in learning how to use certain input devices. People had a lot of problems in coordinating mouse usage at the beginning. Moreover, in a typical computer, two devices are needed for input of data and the results are displayed on the third device: keyboard and mouse for input, monitor for display. This leads to a divided attention of the user, and a constant need to switch from one to another.

Ubiquitous computing [88, 87] tries to make computers unnoticeable and part of the environment. However, the input is still done through writing on various pads. Computers are smaller, and the communication with host computers is unnoticeable. Still, there is no effort to observe the user using computer vision techniques.

Apart from ubiquitous computing, new input interfaces have been developed. The use of voice recognition for input [85] is now commonplace. Gesture recognition is also used as an input interface [64, 29, 20]. Finally, head and eye movements are used as a replacement for the conventional mouse [48, 91, 75, 79, 2, 100, 73, 46, 81, 49, 6, 3, 4, 1]. Many such systems were developed to be used primarily by handicapped users [40, 49, 6, 3, 4, 1]. The advantage of handless HCI for such users is obvious. Most of those systems require special set-up and/or special hardware, which can be both cumbersome and expensive. The use of infrared cameras is typical. However, there are health concerns with their usage, such as whether infrared light beamed at the eyes damages

likely the user's

Recently, mo

gained popularit

both mode

for the interac

in Section E

the number

interface w

easily m

in the keyboard

methods whic

an HCI system.

All the ab

[illegible] for anal

[illegible]

of HCI [58,

1.2.2 Cog

[illegible]

[illegible]

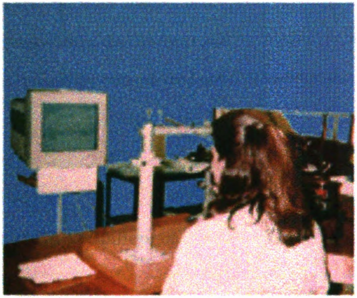
the eyes damages eye tissue. The use of various helmets might be rather unpleasant. Mostly, the user's movements are constrained to a very limited space.

Recently, more emphasis has been placed on developing a handless HCI for a general population [75, 79, 2]. This work is directed toward developing a more user-friendly interface that would involve interaction with computers that is very different from the interaction we know today, and very similar to the scenario at the beginning of the Section. Extensive use of computer vision algorithms would lead to healthier, less cumbersome and cheaper systems that could have many practical uses. This kind of interface would offer more freedom to the user—there would be no need to constantly move one hand from keyboard to the mouse, thus all the attention is devoted to the keyboard. Additionally, we would exploit the user's natural behaviour—the user looks where he or she wants to work and that should be naturally captured by an HCI system.

All the above systems rely heavily on the use of various computer vision algorithms for analysis of faces. Individual applications like face recognition [82, 83, 78], gesture recognition [26, 92], face-spotting in images [69, 59, 42, 96], gaze detection and HCI [58, 10, 8], are all used to create the new HCI.

1.2.2 Cognitive Aspects

In cognitive science the field of visual perception has been studied extensively. Visual perception is considered to have a key role in our cognition. By studying eye



[Image is presented in color.]

Figure 1.1: Purkinje pupil tracker, an illustration (Courtesy of John M. Henderson, Michigan State University Eyetracking Laboratory, `http://eyelab.msu.edu/`)

movements, we can conclude which parts of a scene are viewed first, to which parts we devote more attention, what we notice and what we do not notice [38, 37].

To measure a subject's eye movement, a "classic" Purkinje pupil tracker is used. This device is very accurate and allows excellent measurements to be taken. The problem with it is that the measuring device is intrusive and very hard to use, and the user does not behave naturally. During measurements, the user's head is fixed within a wire frame and the user needs a bite bar to reduce head movements (see Figure 1.1 for illustration). Calibration of the system is needed before each measurement starts, and often also during the measurement session.

A great advantage would be to have a completely non-intrusive eye tracker that would allow measurements to be taken while the user is naturally working in a 3D space. The head-eye

as this. More important

would behave natur

way.

1.2.3 Computing (

From the computing po

many challenges. Th

the problem need to be

covered. Finally, all th

ments.

Segmentation of a fa

task for humans. F

such as skin color,

complicating the process

same fixed scale and o

the face from a grayscal

binary image try to

image. Some algorithms

[36, 59, 42]. Pro

the face orientatio

workspace. The head-eye input interface described in this dissertation would facilitate this. More importantly, the interface could be "hidden" from the user and the user would behave naturally, without the burden of knowing that an experiment is underway.

1.2.3 Computing Challenges

From the computing point of view, the task of detecting and tracking a human face has many challenges. The aim is not to analyze a general scene, but many aspects of that problem need to be covered. Many forms and variations in human faces need to be covered. Finally, all the algorithms need to work fast to keep pace with the user's movements.

Segmentation of a face: Distinguishing a face in an arbitrary scene is such an easy task for humans. For a computer vision algorithm, though, this is not trivial. Issues such as skin color, face orientation, and scale play a very important role in complicating the process of face segmentation. Algorithms for face recognition usually assume fixed scale and orientation, and they deal with the problem of recognizing the face from a grayscale image only [82, 62]. Algorithms for face spotting in an arbitrary image try to deal with the above problems, and to locate a face in a single image. Some algorithms use grayscale input images [69], while others work with color images [96, 59, 42]. Problems that persist are finding dark faces and covering all the possible face orientations. In all cases, the execution time of these algorithms is not en close to real-time

suitable for analysis of live

To overcome this pro

example used is some

assumption here is that th

Finding selected face fe

relatively straightforward

templates for desired feat

template matching and

approach is that it is very

defined templates that

features. Additionally, a

By using a slightly d

to finding these four m

significantly. In this work

overcome the slow p

Determining user's

age. Even infants can

us. For computer vis

to calculate the gaze

methods to calculate

and extensive calibra

features' positions. A

even close to real-time (e.g., 30 processed frames per second), thus making them not suitable for analysis of live video.

To overcome this problem for analysis of live video streams, the most common technique used is segmentation using a *skin color model* [59, 42, 96, 8]. A significant assumption here is that there *is* a face in the input image.

Finding selected face features: Locating the eyes, nose, and mouth in a face is relatively straightforward to achieve once the face has been extracted. We can use templates for desired features (e.g., average eye, nose and mouth), perform brute-force template matching and obtain our feature locations. The major problem with this approach is that it is very time-consuming, it would assume fixed-scale input images, and fixed templates that might not cover all the variance in the appearances of the features. Additionally, a change in lighting conditions could result in poor results.

By using a slightly different approach—not looking for each feature in isolation, but finding these features together since they have fixed natural relationships would help greatly. In this work, the geometry of the face and various heuristics are exploited to overcome the slow template-matching approach.

Determining user's gaze: When we see a face, we can easily determine the person's gaze. Even infants can distinguish whether someone is looking them in the eyes or not. For computer vision algorithms, many assumptions need to be fulfilled in order to calculate the gaze. Pupil trackers based on infrared lighting use the Purkinje reflections to calculate the gaze (Figure 2.2). As was discussed in Section 1.2.2, they need extensive calibration. Another possibility is to calculate the gaze from face features' positions. Algorithms like Pose-From-Three-Points (P3P) [58, 10] proposed

mathematical solution.
characteristics. This con
forced to ask different w
A solution to this problem
Network in conjunction w

Respecting fo

humans. We express our f
personal interaction. F
would be impossible since
an artificial system, w
problem. The recogni
nearest-neighbours cl
feature finding method

Fast operation: An
similar fast updates. Th
at a rate of at least 1
image for processing
and feature finding.
classify each pixel and
pixels. Feature findi
methods. A solut
task but also the a
algorithm sparsely.

a mathematical solution, however, we need to know the user's dimensions and camera characteristics. This could lead to problems with the portability of the system, and the need to ask different users to enter their dimensions prior to the use of the system. A solution to this problem is to use a non-linear approach based on an Artificial Neural Network in conjunction with an approach used in the conventional mouse.

Recognizing face expressions: A wide variety of facial expressions is present in humans. We express our feelings through them and they are key attributes in human-to-human interaction. For a computer system to recognize all possible expressions would be impossible since it is hard to enumerate all possibilities and variations. Thus, in an artificial system, we need to concentrate on a few expressions and work only with them. The recognition can be based on neural networks [7], Markov models [59], or nearest-neighbours classification. In this work, an approach derived from our facial features finding method is used, as will be described in Chapter 7.

Fast execution: An important issue in designing an input interface is ability to have fast updates. The algorithms described above need to analyze a video stream, at a rate of at least 15 frames per second. That leaves about 66 milliseconds on average for processing of one image. Necessary operations are face segmentation and facial feature finding. A segmentation algorithm is time consuming due to the need to classify each pixel and to use a connected components algorithm for the grouping of pixels. Feature finding using template matching would be too expensive for the real-time needs. A solution is to use either smaller images, thus reducing the necessary work, but also the accuracy of feature detection, or to use the most time consuming algorithm sparsely.

1.3 Propose

In the remaining Cha

termination. HCI

is based on the

diagram of the syst

Then the face is lo

and further analys

eyebrows and nose

whether the mouth

again and their m

just as we will use

The gaze coordin

the screen and the

the cursor location

underlying windows

1.4 Organiz

The remainder of th

Chapter 2 review

estimation and fa

and its applications

work are discussed

1.3 Proposed Gaze Tracking System Organization

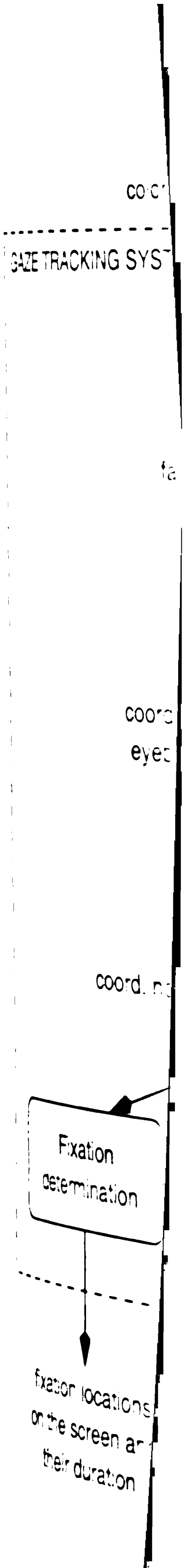
In the remaining Chapters, the methods for face tracking, facial feature location, gaze determination, HCI building and evaluation will be presented. They are all parts of or based on the gaze tracking system we developed. Figure 1.2 depicts the block diagram of the system. The input is the color image stream from a video camera. Then, the face is located, and face coordinates are used to extract the face sub-image and further analyze it and locate facial features. Currently located features are eyes, eyebrows and nose. In addition, based on the eyes and nose location, we determine whether the mouth is opened or closed. Finally, we transform the facial features location and their movement into the gaze point coordinates on the screen. *Gaze point*, as we will use this term, is defined in Chapter 5.

The gaze coordinates can be used in several ways: determining fixation points on the screen and their duration, measuring user attention to the display, controlling the cursor location and using the mouth state to send selection signals (clicks) to the underlying windows interface.

1.4 Organization of the Dissertation

The remainder of this dissertation is organized as follows.

Chapter 2 reviews background information and related work. The work on face segmentation and facial feature location is reviewed. The Kalman filter is introduced and its applications are briefly discussed. Several approaches to gaze direction detection are discussed. The Purkinje pupil tracker is introduced and its application


co-or
GAZE TRACKING SYS
fa
coord
eye
coord. in
Fixation
determination
fixation locations
on the screen a
their duration

F

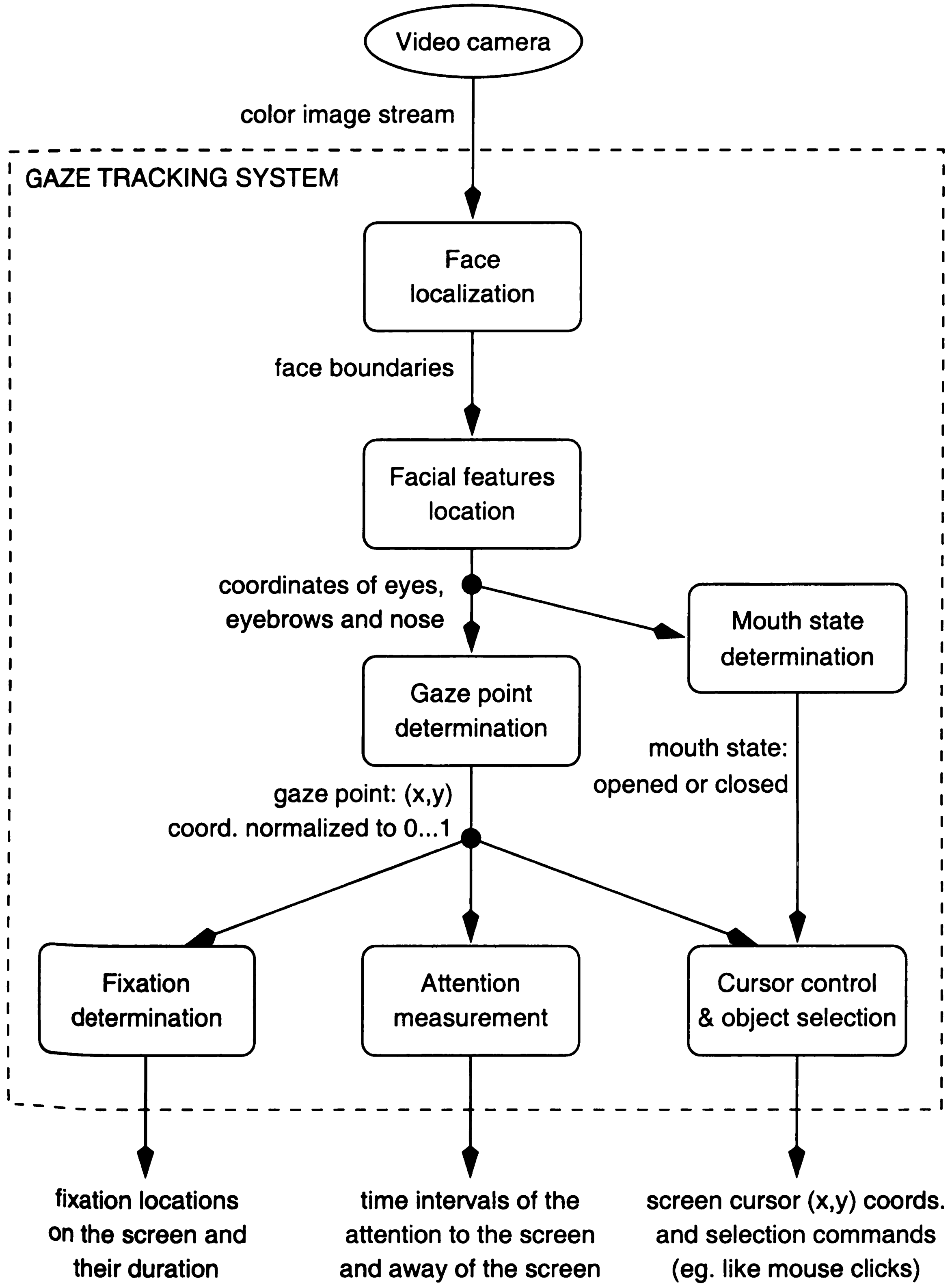

Video camera
color image stream
GAZE TRACKING SYSTEM
Face localization
face boundaries
Facial features location
coordinates of eyes, eyebrows and nose
Mouth state determination
Gaze point determination
mouth state: opened or closed
gaze point: (x,y) coord. normalized to 0...1
Fixation determination
Attention measurement
Cursor control & object selection
fixation locations on the screen and their duration
time intervals of the attention to the screen and away of the screen
screen cursor (x,y) coords. and selection commands (eg. like mouse clicks)

Figure 1.2: Gaze tracking system block diagram

gaze direction detectio

viewed and discussed. I

re-introduced, and m

ons are discussed.

Chapter 3 describes a s

facial features. Its advan

face segmentations is pres

segmentation cases. A n

s that the skin color m

ay to overcome the dr

locating eyes, eyebrows a

knowledge of the geome

evaluation of eyes-nose

of the feature location a

Chapter 4 describes

promising results. Th

certain parts of the o

algorithm is presented

problems in the trackin

the tracking. Several re

Chapter 5 presents

coordinates of eyes and

feature location to scre

to gaze direction detection are discussed. Several commercial eye-mouse systems are reviewed and discussed. Finally, HCI-related issues in the creation of user interfaces are introduced, and findings of several studies that compare conventional and eye mouse are discussed.

Chapter 3 describes a skin color model for face detection and a method for locating facial features. Its advantages and drawbacks are discussed. A number of successful face segmentations is presented, as well as a number of unsuccessful and problematic segmentation cases. A method for detecting a dark face and enhancing the colors so that the skin color model can be used is presented. Finally, some suggestions on how to overcome the drawbacks are made. Once the face is located, a method for locating eyes, eyebrows and nose on a face is presented. The method is based on the knowledge of the geometry of the face and uses several heuristics. The parameters for evaluation of eyes-nose matches are introduced. Finally, evaluations of the accuracy of the feature location are presented.

Chapter 4 describes how the features are tracked in time and discusses the accuracy and timing results. The system state diagram is presented and it is discussed when certain parts of the overall algorithm are executed. The usage of a motion detection algorithm is presented and it is discussed how it can be used to overcome some problems in the tracking. It is described how the Kalman filter is used to smooth out the tracking. Several results of successful smoothing using the filter are presented.

Chapter 5 presents a method for detecting user's gaze direction based on the coordinates of eyes and nose in the image. A neural network used to map user's facial feature location to screen coordinates is described, as well as a method used for its acking. The mapping

ace is described. The

a the results of smooth

velopment. The ease

easurement. The results

Chapter 6 presents de

ace tracking. Ways

sues related to GUI de

ased interface is propos

Chapter 7 discusses

nd presents the results

he input interface. Tas

election, dragging of

all were tested. The

marks on the GUI de

Chapter 8 conclud

nd future directions

training. The mapping of motion of features in the image to the motion of the screen cursor is described. The joint use of the above two mappings is discussed, as well as the results of smoothing of the output, which provides a framework for further development. The gaze tracking is applied to fixation determination and attention measurement. The results of preliminary studies are presented.

Chapter 6 presents development of an HCI based on the head-eye input interface and gaze tracking. Ways to make a handless selection are discussed along with some issues related to GUI design. Finally, the evaluation procedure for a head-eye input based interface is proposed.

Chapter 7 discusses several applications of the system described in this dissertation and presents the results of an evaluation and usability study of our proposed head-eye input interface. Tasks such as moving the cursor along a defined path, button selection, dragging of icons, and the use of general applications (e.g., Netscape and mail) were tested. The results for two user sessions are compared and some concluding remarks on the GUI design and interface usability and learnability are made.

Chapter 8 concludes the dissertation with a summary of the major contributions and future directions towards the improvement of the proposed interface.

Chapter 2

Backgroun

In this Chapter, vari
view the topics of fa
ation of facial express
Section 2.5 introduces
In particular, faces and
relevant to this work a
perception research is

2.1 Face Loca

Finding faces in an arbi
This is due to the compl
possible locations and siz
single approach is very

Chapter 2

Background and Related Work

In this Chapter, various background materials are discussed. The first three Sections review the topics of face localization in images, locating facial features, and discrimination of facial expressions. In Section 2.4, gaze direction determination is discussed. Section 2.5 introduces the Kalman filter and reviews the topic of tracking of objects, in particular, faces and facial features. Human-Computer Interaction (HCI) issues relevant to this work are presented in Section 2.6. Finally, in Section 2.7, visual perception research is discussed.

2.1 Face Localization

Finding faces in an arbitrary image has been studied extensively. The problem is very difficult due to the complexity of the human face, within-class variations in faces, and possible rotations and size of the face. To cover all races, skin colors and shapes in a single approach is very hard. In this Section we review three paths that can be [illegible] clustering. F[illegible]

[illegible] approach and sh[illegible]

2.1.1 Clustering.

In this Section we descri[illegible]

[illegible] and AI te[illegible]

[illegible]-based systems

Sung and P[illegible] [7[illegible]

the first level, they de[illegible]

the subwindows into [illegible]

[illegible] to each cluster

[illegible] significant ei[illegible]

[illegible] in the 77[illegible]

[illegible] to classify poin[illegible]

[illegible] more negative th[illegible]

[illegible] reducing the exp[illegible]

Sakai et al. [72, [illegible]

[illegible] templates. [illegible]

[illegible] the scaling of [illegible]

Govindaraju et al. [illegible]

[illegible] and semantic[illegible]

[illegible] Hough [illegible]

followed: clustering-, PCA-, and rules-based approaches, artificial neural network-based approach and skin color-based approach.

2.1.1 Clustering, PCA and Rule Based Approaches

In this Section we describe several face location systems based on "traditional" pattern recognition and AI techniques like clustering, principal components analysis (PCA), and rule-based systems.

Sung and Poggio [77] designed a face location system based on clustering. In the first level, they determine whether a face exists in each subwindow by classifying the subwindows into one of six face or six non-face clusters. They use two distance metrics to each cluster: "partial" distance between test pattern and the cluster's 75 most significant eigenvectors, and Euclidean distance between the test pattern and its projection in the 75-dimensional subspace. In the second level, a neural network is used to classify points using the two distances to each of the clusters. They used 10 times more negative than positive face examples, collected in a "bootstrap" manner, thus reducing the explicit number of negative examples.

Sakai *et al.* [72] proposed a system based on line extraction and matching lines to predefined templates. They had templates for the face, eyes, nose and mouth, and allowed the scaling of templates to be able to locate faces of unknown sizes.

Govindaraju *et al.* [33] proposed a method for face location based on face boundary detection and semantic networks. From an edge-image they locate curves using the generalized Hough transform. Curves that are likely to be face sides, hair line, or ain line are grouped us

al. [8] applied a similar

train candidate curvat

the weight of the connec

between features. The fa

Yang and Huang [34] m

In the first phase, simple

to find candidate regions

lower-resolution windows

classify a full-resolution

2.1.2 Artificial Ne

The introduction of Ros

research that tried to mim

ons (ANN). However, M

ceptrons could solve only

the mid 1980's, the backpr

in pattern recognition, an

Burel and Carel [15] d

background. The inpu

classification result. S

chin line are grouped using weighted voting. In a follow-up work, Govindaraju *et al.* [34] applied a similar method. They found the edge-image and used thinning to obtain candidate curvature lines. The lines were grouped in a sparse graph, where the weight of the connections between the nodes represented the degree of similarity between features. The face was found as the biggest clique in the graph.

Yang and Huang [93] manually coded rules and feature detectors for face detection. In the first phase, simple rules are applied to the high-resolution windows in order to find candidate regions. In the second phase, more complex rules are used in the lower-resolution windows. Finally, in the third phase, edge-based features are used to classify a full-resolution window as face or non-face.

2.1.2 Artificial Neural Network Based Approach

The introduction of Rosenblatt's perceptron [68] in late 1950's resulted in a lot of research that tried to mimic human visual perception through Artificial Neural Networks (ANN). However, Minsky and Papert's proof [52], in the late 1960's, that perceptrons could solve only limited problems, resulted in a pause in the use of ANNs. In the mid 1980's, the backpropagation training algorithm [71] revived the use of ANNs in pattern recognition and artificial intelligence (AI) problems.

Burel and Carel [15] designed a 2-layer neural network that classified input as face or background. The input to the network was a 15×20 window, and the output was a classification result. Subwindows of an input image were presented at various scales recognize faces of d

ole recording of a c

Vaillant et al. [

age produces a resp

face images only, th

the system is that it

number of non-face

Important resul

[illegible] accuracy in

a 20 × 20 window a

network at the ori

the network is to a

the size of the tran

are added at will. T

The execution time

about 24 seconds p

designed a rotation

is a router network

The router network

input window is a f

rotated, the network

the face is in the upr

face or non-face. T

to recognize faces of different sizes. Training and testing data were obtained from the video recording of a conference room.

Vaillant *et al.* [84] designed a system that used two layers of networks. The first layer produces a response for face images, while the second layer reacts to centered face images only, thus locating the face more precisely. One possible problem with the system is that it was trained mainly with positive face examples, and only a small number of non-face images was used.

Important results were obtained by Rowley *et al.* [69] at CMU. They achieved 80–90% accuracy in face location using ANNs. Their network was trained to classify a 20 × 20 window as face or non-face. Each 20 × 20 subwindow is presented to the network, at the original scale and at reduced scales. A key point in the training of the network is to add non-face examples as the training progresses [77]. This reduces the size of the training data set, which can grow very quickly if negative examples are added at will. The network is trained to accept a certain degree of head rotation. The execution time of the program is not suitable for real-time applications since it is about 24 seconds per 320×240 input image. In their follow-up work, Rowley *et al.* [70] designed a rotation-invariant face detection system. The first stage of their system is a router network that detects whether the input window represents a rotated face. The router network assumes that the candidate window *is* a face, and whether the input window is a face or not is resolved at a later stage of processing. If the face is rotated, the network detects the rotation angle, and the window is de-rotated so that the face is in the upright position, and the network from [69] is used to classify it as face or non-face. The accuracy of the new system is 79.6%, and the processing time for an 160 × 120 input in

with an 174 MHz R

Osuna et al. [

Rowley and achieved

polynomial neural net

2.1.3 Skin Colo

The main problem w

the time. All the al

the face locations. In

the input. In the case

added—the skin colo

mainly using a skin c

In 1996, Yang an

skin color model. H

can be modeled by

only fits in one cl

pixels, grouping the

regions are likely to

the gaze tracking sy

et al. [42] for tracki

for an 160×120 input image is about 6 seconds, measured on an SGI O2 workstation with an 174 MHz R10000 processor.

Osuna *et al.* [60] trained a support vector machine (SVM) in a similar fashion as Rowley, and achieved comparable results. Their system provides new way of training polynomial, neural network or Radial Basis classifiers in a much faster manner.

2.1.3 Skin Color Based Approach

The main problem with approaches described in Sections 2.1.1 and 2.1.2 is the execution time. All the algorithms need to inspect many subimages in order to determine the face locations. In addition, these algorithms all work with grayscale images as the input. In the case of color images of faces, one important piece of information is added—the skin color. In the late 1990's, tracking of faces in video streams is done mainly using a skin color model.

In 1996, Yang and Waibel [96] presented a system for face tracking based on a skin color model. Human skin can be clustered into easily discriminable clusters, that can be modeled by one or more Gaussian distributions. A wide variety of skin colors easily fits in one cluster. The method for tracking is based on the classification of pixels, grouping them into connected components or regions, and determining which regions are likely to be face. These results are further used by Stiefelhagen *et al.* [75] for a gaze tracking system, Oliver *et al.* [59] for tracking and facial expressions, Jebara *et al.* [42] for tracking and modeling, and Bala *et al.* [9] for face and eye tracking, to mention just a few. In

computers in many ca

Most skin color mo

the case of RGB, the

is $G_{norm} = \frac{G}{R+G+B}$

are used. The model

issued this issue

A potential problem

by the model. This

of different chrominan

problem with dark skin

their relative values, w

Another problem w

colors similar to the fa

of Caucasian skin, or

In such cases, a face

face and the adjacent

regions

2.2 Facial Fe

Locating facial features

of methods have be

mention just a few. In all these systems, real-time rates are achieved using low-end computers, in many cases PCs.

Most skin color models are based on either RGB or HSI chrominance models. In the case of RGB, the plot of normalized red and green components ($R_{norm} = \frac{R}{R+G+B}$ vs. $G_{norm} = \frac{G}{R+G+B}$) is used, and in the HSI model, hue and saturation components are used. The model can be adapted to a particular skin, and Yang and Waibel [94] discussed this issue.

A potential problem with a skin color model is that dark skin is not well covered by the model. This was noted in Terrillon and Akamatsu [80], who did a study of different chrominance models applied to the skin color clustering problem. The problem with dark skin is that low values of R, G and B lead to large fluctuations in their relative values, which results in significant noise.

Another problem with all skin color models is that background pixels might be of color similar to the face color. For example, the walls could be beige, which is similar to Caucasian skin, or the subject could be wearing clothes that are of skin-like color. In such cases, a face detection algorithm may not be able to discriminate easily the face and the adjacent regions, and more processing may be needed to analyze the regions.

2.2 Facial Feature Finding

Locating facial features in the face images has been studied extensively. A wide variety of methods have been applied, and all of them have advantages and disadvantages mainly the execution

algorithms like templa

ng using geometrical

application-specific ap

or detecting blinking

data.

Yuille et al. [

They defined an eye

regions of the eyelids

stage of their system

image. They then de

Pentland et al. [

features and compari

nose-mouth and were

matching with eigen-

mouth locations had

values more than 5

% with 6% false al

Jalmi and Liu [

features, and did s

matching to find th

camera to zoom in or

Kwon and da Vi

(mainly the execution times were rather high for some real-time applications). The algorithms like template matching, applying an eigen-feature approach, or matching using geometrical constraints, could be applied on typical still images. More application-specific approaches are based on detecting infrared reflections of the eyes, or detecting blinking in video stream, or in modeling mouth color in video stream data.

Yuille *et al.* [99] find eyes and nose in a face image using deformable templates. They defined an eye template as a combination of a dark circle for iris, two parabolic sections of the eyelids for the boundary, and white regions of eyes. The preprocessing stage of their system includes morphological filters to find peaks and valleys in the image. They then deform the templates to minimize the energy function.

Pentland *et al.* [62] proposed a method for face recognition based on finding facial features and comparing them to the database. They defined eigeneyes, eigennose and eigenmouth and were able to successfully locate eyes, nose and mouth using template matching with eigen-features. The locations of eyes were easily found, while nose and mouth locations had higher variation. Overall, the erroneous matches were defined as ones more than 5 pixels away from the true location, and the recognition rate was 94% with 6% false alarms rate.

Talmi and Liu [79] used a PCA-based approach to find eyes. They created eigeneyes, and did subwindow search for the eyes. Additionally, they used stereo matching to find the precise eye position in 3D space, so that they could focus a camera to zoom in on the eye.

Kwon and da Vitoria Lobo [50] use snakelets to locate eyes, nose and mouth.

Stiefelhagen et

mouth corners. S

into an eye-region

they use an iterati

integral projection

thresholding for

Baia et al. [

ultimately black

of successive im

luminance-a lapo

Oliver et al. [

and track the m

Jebara et al.

motion compo

a line from a d

disease estima

radial gradient

Ballard and

method uses d

speed up the co

the pupils and

tracks the user

Stiefelhagen *et al.* [75] use geometrical constraints to locate eyes, nostrils and mouth corners. Starting from a region that is likely to be a face, they separate it into an eye-region and a mouth-region. Assuming a near-frontal view of the face, they use an iterative thresholding in the eye region to find eye pupils; horizontal integral projection in the mouth region to find mouth location; and finally, iterative thresholding for nostrils.

Bala *et al.* [9] find the eyes in a video stream based on blinking. Humans involuntarily blink to moisten the eyes, and the authors analyze luminance differences of successive images, which indicate motion. The eyes are located by tracking a luminance-adapted block matching technique.

Oliver *et al.* [59] model mouth color similarly to the skin color modeling, and find and track the mouth based on a classification into mouth color and connectivity.

Jebara *et al.* [42] find eyes in a face image by computing dark symmetry transformation, computed from a phase and edge map by wave propagation. Mouth detection is done from a dark axial symmetry map, where the longest limb is selected as mouth. A coarse estimate of nose's vertical location is done by searching for the strongest vertical gradient in an intensity image bounded by the eyes and mouth.

Ballard and Stockman [10] used two methods to find eyes and nose. The first method uses deformable templates for the eye, similarly to ones described in [99]. To speed up the computation, simple ellipsoidal templates are used. The second method finds pupils and nose from light reflection. In their experiment, light is projected towards the user's face, and pupils can be easily found by finding bright reflections sing thresholding.

ne harsh for the us

Hutchinson et al

lluminate the eye

amera they can eas

2.3 Facial I

etermining facial e
stream. A number
the case of still ima
skin regions, an
video stream analy
based on them. In
determined (e.g., th
image is analyzed a

Yacoob and Da
rigid and non-rigid
which is based on
expressions: disgus
ranged from 80 t

Kimura and Ya
faces. Potential

using thresholding. A problem with their system was that the lighting conditions were harsh for the user.

Hutchinson *et al.* [40] and Morimoto *et al.* [54] used an infrared light source to illuminate the eye. Using a simple thresholding, in images obtained from an IR camera they can easily find a pupil and light reflection off the cornea and retina.

2.3 Facial Expression Discrimination

Determining facial expressions could be done either from a still image, or from video stream. A number of different algorithms have been proposed for both cases. In the case of still images, algorithms like shape analysis, PCA-based classification of salient regions, and neural networks-based classification are used. In the case of video stream analysis, motion vectors are computed, and expressions are determined based on them. In fact, in the later case, typically the *change* in facial expression is determined (e.g., change from neutral to smile), while in the former case, the static image is analyzed and classified into an expression category.

Yacoob and Davis [92] use optical flow computation to identify the direction of rigid and non-rigid motion of the face muscles when changing expressions. Their model is based on linguistic and psychological considerations. They distinguish six expressions: disgust, sadness, happiness, fear, anger and surprise. The success rate ranged from 80% for sadness to 94% for surprise.

Kimura and Yachida [47] used the PCA analysis to classify face expression and its degree. Potential nets were calculated from the motion vectors of the face and four expressions were de

that the degree of a

with other three e

individual who was

and when an unkn

Cover et al. [

mouth smile and

is characterized b

four extrema poin

the final classifica

Baird and M

expressions as (

expression was e

network had the n

Colmenarez e

expressions were

They define regi

nose, mouth cor

regions. The fea

corners and the t

classification base

model is retrieved

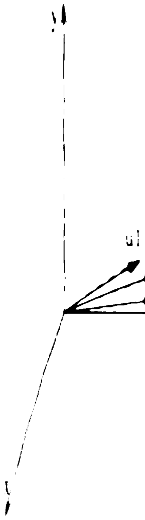
and using both th

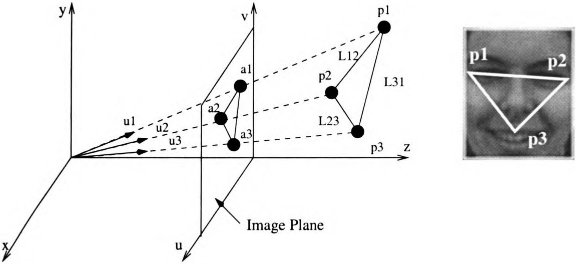
expressions were determined: neutral, anger, happiness and surprise. They showed that the degree of an expression can be estimated as the person changes from neutral to the other three expressions. An evaluation of their system was done on a single individual who was used for training also. The performance of the system was not as good when an unknown test person used it.

Oliver *et al.* [59] used mouth shape to discriminate four expressions: neutral, open mouth, smile, and open mouth smile. Their primary feature is the mouth shape that is characterized by the XY eigenvalues of the mouth region and the XY position of four extrema points. They used these features in a Hidden Markov Model (HMM) for final classification, and achieved 96% accuracy.

Bakić and Miller [7] used an artificial neural network to discriminate the same expressions as Oliver. The overall accuracy achieved was 84%. The open mouth expression was easier for discrimination than neutral, smile and open smile, and the network had the most confusion between the latter two expressions.

Colmenarez *et al.* [21] proposed a classification based on several features. Six expressions were considered: neutral, sadness, happiness, surprise, disgust and anger. They define regions for facial features and use the position of extreme points (e.g., eye or mouth corners) within the regions for classification, as well as the images of regions. The features they use are the end points of the eyebrows, eye and mouth corners and the tip of the nose. For each user they design a separate facial expression classification based on the PCA, and using their face recognition algorithm, the user's model is retrieved and used. Using the feature points only, the error rates are 19–46%, and using both the feature points and feature images, the error rates are 6–20%.





Coordinate systems and perspective transformation and three feature points on face.

Figure 2.1: The illustration of the system described in Ohmura *et al.* [58]

2.4 Gaze Direction Detection

In this Section, several methods for gaze estimation are presented: based on mathematical equations, based on the detection of pupil reflections, based on limbus tracking, based on teaching an artificial neural network to map eye image to gaze direction, and based on electro-oculography.

2.4.1 Mathematical Solution

In 1988 Ohmura *et al.* [58] presented an algorithm for calculation of gaze direction from the image coordinates of three points, distances of the three points on the face, and camera focal length. The gaze was defined as perpendicular to the plane that passes through the three face points. They used three cosmetic marks near eyes and nose with

[illegible] their

Ballard and

They applied th

Gee and Cip

features of fac

sequence [57]. U

approaches, an

eye and mouth

was proposed

symmetry [57]

generated fa

that the calcula

Stiefelhagen

tation problem

lters [24]. In

Artificial Neural

2.4.2 Pupi

Hutchinson et a

illuminated the

relative position

and nose, which were tracked in real-time using specialized hardware. Figure 2.4.1 illustrates their system.

Ballard and Stockman [10] used the algorithm from [58] to calculate user's gaze. They applied their results to the HCI task of menu selection.

Gee and Cipolla [31] presented a solution based on the knowledge of relative distances of facial features, and assumption that the imaging process is weak perspective [67]. Using simple calculations based on the geometry of the face and imaging process, and knowing the distances between eyes, between eye line and mouth, nose and mouth, and nose depth, they can estimate the face normal. Two methods were proposed—one based on the 3D geometry, and another based on the skew-symmetry [55] of the facial plane. They calculated the accuracy of their model using a generated face model, which was rotated in all possible directions. They showed that the calculated normal is within 6^o of the true facial normal.

Stiefelhagen *et al.* [75] posed the gaze determination problem as the pose estimation problem, and used iterative POSIT algorithm developed by DeMenthon and Davis [24]. In addition to head-pose estimation, they estimate eye-gaze using an Artificial Neural Network in the same manner as described in [11].

2.4.2 Pupil Reflections

Hutchinson *et al.* [40] designed an eye-controlled HCI for disabled persons. They illuminated the user's eye with infrared light, and the gaze was inferred from the relative position of the *bright-eye* (the reflection off the retina) and the *glint* (the

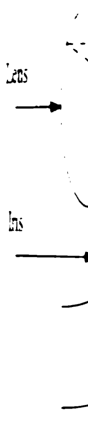


Figure 2.2: T
in the b

schema fr

Purkinje ima

calibration

eye exten

Taking

eye pos

from a light

[illegible]

eye tra

the time lag

[illegible]

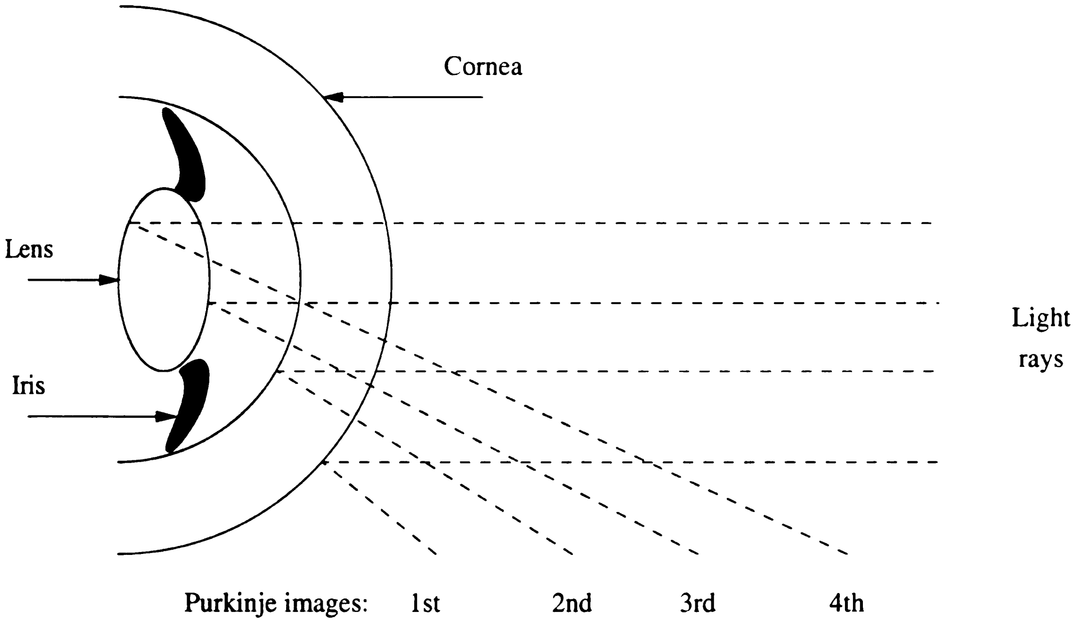


Figure 2.2: The four Purkinje images are formed as reflections of incoming light rays on the boundaries of the lens and cornea

reflection from the cornea, the first Purkinje image). Figure 2.2 illustrates the four Purkinje images. The system needs to be calibrated before each session. For the calibration to be valid, the user's head should be still throughout the session. In [48] they extended the system to allow some head movements.

Talmi and Liu [79] obtained a zoomed eye image by focusing a camera to the current eye position in 3D space. They found the pupil center and the bright reflection from a light source, and used their relative position to determine user's gaze. The system needs to be calibrated before each session, and is sensitive to head motion. Using stereo eye tracking, they were able to compensate for some head motion. However, the time lag needed to re-calibrate and re-focus the eye camera upon head motion was one second, which may be problematic in a real-time application.

2.4.3

[illegible]

2.4.4

[illegible]

2.4.5

[illegible]

2.4.3 Limbus Tracking

The limbus tracking method tracks the boundary between the dark iris and white sclera (the white of the eye). By measuring the proportion of sclera left and right of the iris, we could determine horizontal eye motion and transform that into gaze. This method, however, is not suitable for vertical motion detection. Similar to the systems described in Section 2.4.2, this system would require calibration before each session. Applied Science Laboratories [1] has developed a commercial eye-tracker based on this method, that requires the user to wear glasses with a tracking equipment.

2.4.4 Artificial Neural Network Estimation

Baluja and Pomerleau [11] trained an Artificial Neural Network to estimate gaze direction. The input to the ANN was an image of an eye, and the output was a grid of 50×50 units organized as X and Y coordinates of the gaze. The network was trained to have Gaussian output representation, similarly to what was used in the ALVINN system [63]. For each user a separate ANN has been trained. Achieved accuracy was 1.5^o. However, it is not clear whether significant head motion was allowed, and what the testing procedure was.

2.4.5 Electro-Oculography

Kaufman *et al.* [44] presented a gaze tracking system based on electro-oculography. They used neurophysiological methods for eye movement measurement, which place electrodes close to the eyes, and measure the potentials. These signals could then be ped as an in

essin and l

2.5 Ka

The problem

points. In m

1960 Kalma

problem. Th

empirical m

the Kalman

Kalman filte

The goal

process. The

equation

$$x_{t+1} = A$$

The measure

$$z_t = H_t$$

The matr

B relates th

measurement

Ise. They are

with normal

used as an indication of user's gaze. Their system needs to be calibrated before each session, and head movement is not tolerated by the system.

2.5 Kalman Filter

The problem of parameter estimation can easily be solved if we know all the past data points. In many practical applications, saving all the data points is not feasible. In 1960, Kalman [43] proposed a recursive solution to the discrete-data linear filtering problem. The Kalman filter is a set of mathematical equations that provide an efficient computational (recursive) solution of the least-squares method. An introduction to the Kalman filter is available in [89]. In this Section, just a brief description of the Kalman filter equations is presented.

The goal of the process is to estimate the state $x \in \Re^n$ of a discrete-time controlled process. The variable x to be estimated is described by the linear stochastic difference equation:

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k.$$

The measurement $\mathbf{z} \in \Re^m$ is used to estimate the value of $\mathbf{x}$:

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k.$$

The matrix $\mathbf{A}$ relates the state $\mathbf{x}$ at time step k to the state at step $k+1$, Matrix $\mathbf{B}$ relates the control input $\mathbf{u} \in \Re^l$ to the state $\mathbf{x}$, matrix $\mathbf{H}$ relates the state $\mathbf{x}$ to the measurement $\mathbf{z}$. Random variables $\mathbf{w}$ and $\mathbf{v}$ represent the process and measurement noise, respectively. They are assumed to be independent of each other, white and with normal probability distribution.

The ...

- *Meas*

matrix

- *Time U*

matrix

The step

1. Initial

2. Take m

3. Perform

4. Go to

The equa

Time up

Measure

with the foll

$\hat{x}_k^-$ $\hat{x}_k$

P_k^-, P_k

K_k

R_k

Q_k

$z_k - H_k\hat{x}$

The estimation algorithm has two steps:

- *Measurement Update*, or "Correct" step, updates the values of Kalman gain matrix $\mathbf{K}$, estimate $\mathbf{x}$ with measurement $\mathbf{z}$, and error covariance matrix $\mathbf{P}$.
- *Time Update*, or "Predict" step, projects ahead the state $\mathbf{x}$ and error covariance matrix $\mathbf{P}$.

The steps are combined as follows:

1. Initial estimation of $\mathbf{x}$ and $\mathbf{P}$;
2. Take measurement $\mathbf{z}$ from the world, and perform *Measurement Update* step;
3. Perform *Time Update* step and use the state estimate $\mathbf{x}$ as needed;
4. Go to step 2.

The equations for the update steps are as follows:

$$\boxed{\text{Time update:}\quad \begin{aligned} \hat{\mathbf{x}}_{k+1}^{-} &= \mathbf{A}_k\hat{\mathbf{x}}_k + \mathbf{B}_k\mathbf{u}_k \\ \mathbf{P}_{k+1}^{-} &= \mathbf{A}_k\mathbf{P}_k\mathbf{A}_k^T + \mathbf{Q}_k \end{aligned}}$$

$$\boxed{\text{Measurement update:}\quad \begin{aligned} \mathbf{K}_k &= \mathbf{P}_k^{-}\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^{-}\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^{-}) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^{-} \end{aligned}}$$

with the following definitions:

$\hat{\mathbf{x}}_k^{-}, \hat{\mathbf{x}}_k$	*a priori*, *a posteriori* state estimate
$\mathbf{P}_k^{-}, \mathbf{P}_k$	*a priori*, *a posteriori* error covariance estimate
$\mathbf{K}_k$	Kalman *gain* or *blending factor*, minimize the a posteriori error covariance
$\mathbf{R}_k$	measurement error covariance matrix
$\mathbf{Q}_k$	process noise
$\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^{-}$	measurement *innovation* or the *residual*

2.6

2.6.1

	CRT	LCD	ELD
resolution	1280 × 1024 or 1024 × 800	up to 800 × 1000	70 addressable dots per inch
size	15 to 21 inches diameter	up to 14 inches diameter	6 by 8 inches up to 12 by 16 inches

Table 2.1: Resolution and size characteristics of common display devices

2.6 HCI-Related Issues

This Section discusses some Human-Computer Interaction (HCI) issues that are used to evaluate input devices. First, basic input device terminology and mental models in the HCI are presented. Fitts' law, used to express the time to move the pointer from one location to another, is presented. A study of eye tracker as an input device is presented, and, finally, several existing eye- and head-controlled interfaces are presented.

2.6.1 Basic 2D Input Devices

In this section, basics of the input device terminology and interaction tasks are presented. For details, see [35, 28].

The most common display technologies used nowadays are Cathode-Ray Tube (CRT) displays, Liquid-Crystal Displays (LCD) and electroluminescent displays (ELD). Table 2.1 shows basic resolution and size characteristics of the above three input devices.

Conventional input devices that are used in conjunction with the above display devices are touch screen, light pens, graphic tablets, mouse, trackballs, and joysticks.

Design detail

resolution al

Touch Scre

the fin

only

from 2

wire t

needs

e.g.

is the

limit

it uses

stylus

may c

display

finger

User

of th

Design details are described very briefly for each of them, and the emphasis is on the resolution and usability of each device.

Touch Screen produces an input signal as a response to a touch or movement of the finger on the display. Touch screens can be based on several technologies: conductive, capacitive, cross-wire, acoustic or infrared. The resolution can range from 25 × 40 for infrared, through 256 × 256 for capacitive, to 1000 × 1000 to 4000 × 4000 for conductive discrete touch points. Studies showed that cross-wire technology had the best tradeoff between display resolution, touch screen resolution and user preference. The issue with touch screens is that the user needs to press the target area, and a user's fingers are not too good for high-resolution target selection. Thus, typical selection area sizes are in centimeters (e.g., $1 \times 2cm$ with spacing of $1cm$). The advantage of touch screen device is that the input device is output device at the same time, thus the hand-eye coordination is natural. The disadvantage of the device is that the resolution is limited by the finger size, regardless of possible device resolution, thus making it unsuitable for small object picking. This can be somewhat resolved by using a stylus, however, the hand movements are not natural in that case, and the hand may obscure the display. Another disadvantage is the parallax problem: the display and the touch-sensitive screen are separated, thus the user places the finger slightly above the target on the display. This can be resolved by asking user to place fingers perpendicular to the screen, however, then the naturalness of the device is decreased.

Light Pen i

play scr

of the r

possib

to the

user ch

Graphic Ta

of a th

a matr

technol

is p

are us

play c

is to

the to

and

Mouse is a

and th

have

moveg

Specia

[illegible]

Light Pen is a stylus that generates position information when pointed to the display screen. The pen senses the electron beam of the CRT and based on timing of the refresh signal, the position of the pen can be calculated. The highest possible resolution obtained is 1/4 of a pixel on a 1000 × 1000 display. Similar to the touch screens, the usability of the light pen is more dependent on the user characteristics and pointing capabilities than on the technology itself.

Graphic Tablets have a flat panel placed on a table near the display. Movement of a finger or a stylus provides position information. The tablets are based on a matrix-encode, voltage-gradient, acoustic, electro-acoustic or touch-sensitive technology. The cursor control can be absolute (cursor position on the tablet is returned), or relative (the cursor movement and previous cursor position are used to calculate the new cursor position). Another issue is what the display/control gain level is. One study showed that for a 12.5in display, a gain of 0.8 to 1.0 worked the best. Tablets have a resolution problem similar to that of the touch screens, and while the display is not obscured by the hand movement and there is no parallax problem, hand-eye coordination is needed.

Mouse is a small hand-held box that measures its movement on the pad surface, and the movement is translated into the graphical cursor movement. Mice have one to three buttons that are used for various selection tasks. Mouse movement can be detected mechanically or optically. In the former case, no special pad is required, while in the later case, a special pad must be used and mouse movements are restricted to it. Similar to graphic tablets, display/control gain o

The a

enabli

mode

be ad

Howev

is on t

Trackball

rotati

on g

gain r

Joystick i

move

move

appli

gain

The ab

speeds ar

speed and

selection a

be used

trackballs

gain can be adjusted to translate mouse movements into display movements. The gain can be larger for rapid movements than for slower movements, thus enabling fast and accurate positioning. The mouse can provide only relative mode and hand-eye coordination is required. Up to a display pixel accuracy can be achieved with the mouse, and it can be easily used to point to small targets. However, some difficulty might arise in selecting them since the selection button is on the mouse and pressing it might cause mouse movement.

Trackball is a ball in a fixed housing that can be rotated in any direction. The rotation is translated into cursor movement. The underlying technology is based on optical or shaft encoders. Similar to the tablets and mice, display/control gain must be specified and the trackball works in relative mode only.

Joystick is a lever mounted vertically in a fixed base. Potentiometers sense the movement of the lever and the output signal can be transformed into cursor movements. A force or isometric joystic has a rigid lever, and the amount of force applied results in cursor movements. Similar to the mice, the display/control gain factor needs to be appropriately adjusted.

The above devices are used for various tasks based on their specifications. Touch screens are best suited for menu selection tasks for data already displayed on the screen and are widely used in information kiosks. Light pens are used for menu selection and locating and moving symbols on the display. The graphic tablets are best used for drawing purposes and can be used to select menu items. The mice and trackballs are best suited for pointing and selection tasks. They could be used for training, how

continuous tra

The abo

relative dir

- All

  are cal

  the m

  the pr

  arbitr

  of th

  that

  prog

  Th

  conf

  are

- Wi

  ind

  in

  an

  Th

  tra

drawing, however, the tablets are more useful in such cases. The joystics are used for continuous tracking task and pointing that does not require high precision.

The above devices are all used as *locator* devices, and they can be absolute or relative, direct or indirect, and discrete or continuous.

- *Absolute* devices have a frame of reference, and the origin and all cursor positions are calculated with respect to the reference frame. *Relative* devices report only the movement of the device, and the display position changes with respect to the previous position. Thus, with the relative devices, the user can specify an arbitrarily large change in position, while with the absolute devices, the range of the position change is restricted. Another advantage of relative devices is that the cursor could be positioned anywhere on the screen from an application program.

  The touch screen and light pen are absolute devices. Graphic tablets can be configured as both absolute and relative device. Mouse, trackball and joystick are all relative devices.

- With *direct* devices, the user points directly with the finger or stylus, while with *indirect* devices the user moves a device that is not on the screen, which results in screen cursor motion. For indirect devices, hand-eye coordination is needed, and that typically requires some time to learn.

  The touch screen and light pen are direct devices. The graphic tablets, mouse, trackball and joystick are all indirect devices.

- Cont

meth

curs

is ch

scali

confi

the ra

Exam

2.6.2 I

The two ba

specifyin

method is

election ra

object clas

representa

Often u

ment and

large a q

Among a

[illegible]

3, 8, an

- *Continuous* devices generate smooth cursor motion as a result of smooth hand motion, while *discrete* devices like cursor-control keys do not provide smooth cursor movement. In case of continuous devices, the speed of cursor positioning is defined by the *control-to-display ratio*, or C/D ratio. The ratio defines the scaling of hand movement change to screen cursor movement change. It can be configured such that for rapid motion the ratio is large while for slow motion the ratio is small, thus allowing accurate positioning.

  Examples of continuous devices are the graphic tablets, joystick and mouse.

2.6.2 Interaction Tasks

The two basic interaction tasks are positioning and selection tasks. A positioning task is specifying an (x, y) position to the application program. The interaction technique involved is moving a display cursor to the desired location and pushing a button. The selection task is choosing an element from a choice set (commands, attribute values, object classes or object instances). This task can be performed by pointing at a visual representation of a set element or pressing a function key for a set element.

Often used interaction tasks for comparison of input devices are target acquisition, menu and text selection, text entering and editing, and continuous tracking. The target acquisition involves positioning of a cursor at or inside the target position and pressing a button to indicate selection. Experiments often have varying target area (e.g., 0.13 to 2.14 cm^2), and target distance from the initial cursor position (e.g., 2, 4, 6, 8, and 16 cm). The menu selection task involves selection of a target menu tem in a va

their spatia

ine via ke

pen mouse

position wr

2.6.3 M

What does

user's know

developer.

model a use

be better.

In this Sec

HCI is rev

What t

actions to

tions and

how to us

[illegible]

the action

In the

needed for

item in a variable length menu. Varying items are the number of menu choices and their spatial and logical organization. The text entering and editing task is typically done via keyboard, but other forms of input have been explored (e.g., using light pen, mouse, etc.) The continuous tracking task consists of the user following a cursor position with an input device.

2.6.3 Mental Models in Human-Computer Interaction

What does a user know about a certain HCI system? How can a developer model user's knowledge? These are very important questions from the standpoint of a developer, and it is very hard to give exact answers. If a software developer can model a user's reasoning about the system, the system's usability and efficiency will be better. From the user's point of view, learning to use the system would be easier. In this Section, Carroll and Reitman Olson's [18] Chapter in Hellander's *Handbook of HCI* is reviewed, and basic terms and models from it are introduced.

What the user knows about a software system includes (i) rules that prescribe actions to be applied (simple sequences), (ii) general methods that fit general situations and (iii) a "mental model"—user's perception of how the system works and how to use it (this should include the knowledge about system components, their interconnections, and ability to construct reasonable actions and explanations why the actions are appropriate).

In the simple sequences representation, the user rote memorizes the sequences needed for certain tasks. For example, the user would memorize a command to be

typed and r
that can be

At the s
that can be
Card. Mon
Operators.
primary a
schools...
by which s
showed th
changing

Anoth
Comman
Task Ac
perform
represent
structur
pick a t

Fina
kinds o
Surroge
report a
the op

typed, and might have no knowledge about the underlying system or general rules that can be applied.

At the second level of representation, we can model the knowledge of methods that can be used. This is often done by modeling tasks and goals to be achieved. Card, Moran and Newell [16] proposed the GOMS model, which stands for Goals, Operators, Methods, and Selection rules. In this model, the user recognizes that a primary goal can be broken into set of subgoals, which can be further broken into subgoals... until a subgoal matches a method in the system. The user has some rules by which subgoals are created and methods chosen. A number of empirical studies showed that this model can be applied to a variety of tasks, sometimes without even changing time parameters.

Another model of user's knowledge of methods is based on command grammars: Command Language Grammar (CLG) [53], Backus Naur Form (BNF) [65, 66], and Task Action Grammars [61]. These grammars are sets of rules that can be used to perform actions in a system, and "sentences" that are acceptable in the grammar represent correct actions that the user can apply. The grammar rules have similar structure as GOMS subgoals, but show more compactly alternative ways to accomplish a task.

Finally, we can model the most complex level—"mental model". There are several kinds of models: surrogates, metaphors, glass box machines and network models. Surrogates [98] is a conceptual analysis that mimics perfectly the target systems's input and output. It is not assumed that the surrogate and the real system produce the output in the same way, thus real causal basis of input-output cannot be provided.

A morph

functions. L

choices (

until the

me

Network

the actions

Generaliz

system dy

2.6.4 F

A metaphor model [19] compares directly the target system with a system already known to the user (e.g., text editor and typewriter). The user easily learns known functions, however, new functions are harder to comprehend and are a constant source of errors. Glass box models [25] are a mixture of surrogates and metaphors. They mimic the target system, and provide some semantic explanations for the internal components. They are more used in a prescriptive context rather than in a descriptive one.

Network representations of the system have states the system can be in, and the actions that the user can take to change states [51]. Kieras and Polson [45] "Generalized Transition Network" (GTN) contains detailed description of what the system does. The states of the network represent visible states of the system (e.g., screen display), and the arcs represent commands and menu choices that can be taken from each state. In simple terms, GTN represents what can be expected of the system when the user takes some action, and that knowledge can be useful to the user while learning and recovering from errors.

2.6.4 Fitts' Law

How fast can a human position a cursor from point A to point B on a computer display? The time needed can be estimated by using Fitts' law equation [16].

The model is based on a *Model Human Processor.* It consists of three processors: *(i) perceptual system*—sensors and associated buffer memories such as visual and auditory image storage; *(ii) cognitive system*—based on sensory information from

Figure 2.3

working m

is made o

The ta

The poin

from the

time ze

process

the total

T_{p}

T_{p}

For

Fig

$T_{}$

$T_{}$

wh

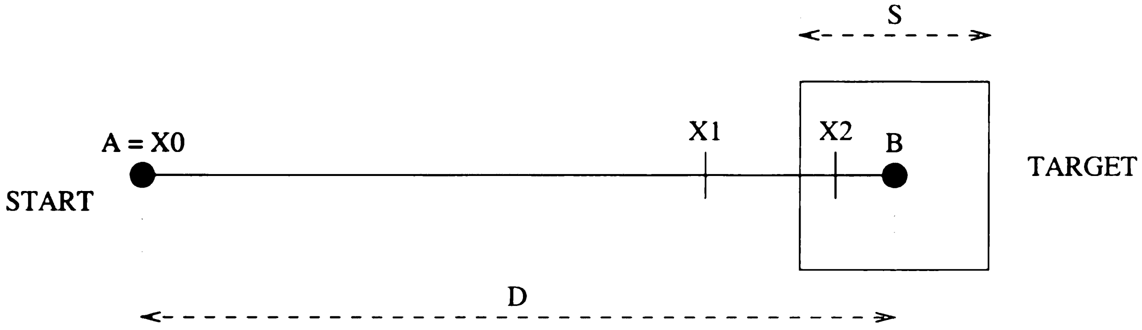


Figure 2.3: Fitts' law: analysis of the movement of user's hand (from [16], page 52)

working memory and previously stored information from long term memory, a decision is made on how to respond; *(iii) motor system*—carries out the response.

The task of moving a pointer from point A to point B is illustrated in Figure 2.3. The points are D units apart, and the goal is to move the pointer within $S/2$ units from the target. In each cycle, the user's perceptual processor observes the hand (time needed τ_P), cognitive processor decides on corrections (time τ_C), and motor processor performs the correction (time τ_M). The number of such cycles is n, thus the total time needed is

$$T_{pos} = n(\tau_P + \tau_C + \tau_M)$$

$$T_{pos} = I_M \log_2(\tfrac{2D}{S})$$

where $I_M = -\frac{(\tau_P+\tau_C+\tau_M)}{\log_2 \epsilon}$ [in msec/bit], and ϵ is a constant.

For low values of $\log_2(\frac{D}{S})$, the equation does not fit the data well, and Welford [90] proposed a correction that better fits the data:

$$T_{pos} = I_M \log_2(\tfrac{D}{S} + \tfrac{1}{2}).$$

The value of I_M is calculated empirically, and ranges $I_M = 27 \sim 122$ msec/bit, and is usually about $I_M = 100$ msec/bit.

2.6.5

Fitting data into Fitts' law equation provides an easy comparison between different input devices' performance on the same task. For example, the data for the conventional mouse and for an eye-mouse could be compared, as will be discussed in the following Section.

2.6.5 Evaluation of Eye-Tracker as Input Device

Ware and Mikaelian [86] studied eye tracking data and eye-movement usability as a computer input device. They used eye-tracking equipment based on infrared corneal reflection, and the system required the user's head to be still during the experiment.

In the first experiment, the user's task was to select one of seven vertically arranged buttons on the screen. To indicate the location that the eye-tracker "thinks" the user is looking at, they showed the current cursor location, as well as outlined the button that was fixated. Three selection mechanisms were tested: 1) *dwell time button*—the object that is fixated for more than some interval gets selected; 2) screen button—a large rectangular area on the screen is set aside for a selection button, and when the user fixates the selection button, the object that was last looked at is selected; 3) hardware button—the user presses a physical button while fixating the item to be selected. Each trial proceeded as follows: the user would fixate the central screen point, and initiate a trial, then the target button was indicated and the user would try to select it. The selection times were all below 1 second, and the dwell button was the fastest selection mechanism, followed by the hardware and screen buttons. The errors in

human r

In th

and the

[illegible]

were 3

[illegible]

the sele

[illegible]

for large

2.6.6

[illegible]

[illegible]

is name

[illegible]

and bas

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

errors in the selection were 12%, 22% and 8.5% for the dwell, screen and hardware button, respectively.

In the second experiment, the buttons were arranged into a 4 × 4 square matrix, and the user was driven to selection of targets by the computer. The sizes of screen buttons varied from 48mm to 7.2mm (at the viewing distance of 90cm, the sizes were 3^o–0.45^o), and the dwell time was set to 0.4 seconds. For smaller button sizes (0.45^o and 0.75^o), selection times were significantly higher, while for 1.5^o–3^o sizes, the selection time was at or below 1 second. The hardware button was faster for the selection. In terms of errors made, for smaller buttons, the errors were 20–50%, and for larger buttons, the errors were below 10%.

2.6.6 Eye-Controlled Systems

Hutchinson *et al.* [40] designed an eye-controlled HCI for disabled persons—the quadriplegic population who retain some motor control of their eyes. The system is named *Erica* for eye-gaze-response interface computer aid. As mentioned in Sections 2.2 and 2.4.2, they use an infrared light source to locate glint and bright eye, and based on their relative position and calibration they can determine the user's gaze. The user's head must remain stationary for the system to work properly. In their system, there is a 3 × 3 matrix with 9 menu buttons. To select an option, the user needs to stare at its menu box, and after some fixed time interval (2 or 3 seconds, but it can be altered) a tone sounds and the cursor appears in the menu box. If the user continues to stare at the same box, the tone will sound again and that option will beside

communi

games a

two pow

part of

making

for con

Fro

the to

letters

puts a

further

for som

By

to exper

Pro

to be

larger

In a

Gener

step

To be

have

be selected. A number of applications are available for Erica: control of appliances, communications programs such as word-processor and voice synthesizer, computer games, and text reading. In the case of the word processing program, only the upper two rows are used for control and the lower row is used for text display. To enter one page of text, it can take 85 minutes for an experienced user. In the case of the text reading program, the upper two rows are used for display and the lower row is used for controls.

Frey *et al.* [30] improved Erica's original word processor. They used a menu-tree to organize letters based on their frequency, and re-organized the menu options (letters) based on the probability of the next letter in a word. The average time to pick a character is 1 second, which leads to 80 minutes per page. White *et al.* [48] further improved the system by enabling spatially dynamic calibration, thus allowing for some head movement during the session.

Buquet *et al.* [14] installed a slide viewer in a museum in Paris, that was based on eye-tracking and gaze as input. Their results were quite promising (on 153 test persons, the success rate was 83%). However, as noted in [32], the participants were not described, which leads to a conclusion that they were a highly motivated chosen target group.

In a museum in Denmark, an eye-controlled exhibition system has been installed. Glenstrup and Eugell-Nielsen [32] attended an exhibition and tested the system extensively. The system, called *EyeCatcher*, introduces a new on-screen button—*EyeCon*. The button changes from open eye to closed eye to indicate to the user that it has been selected. The selection is made by fixating the button for a time period. The system neede

watching a fil

were able to t

and were no

keep the hea

Starker a

terface. Th

Exupery. T

interest lev

draw inf

for it is hig

Jacob

ments, th

reflections

improvem

and that

locations.

area and s

slightly

to mistak

Porn is fa

is that so

Every eye

system needed a calibration phase, and offered text reading, looking at pictures and watching a film clip. The overall conclusion was that the interested users (e.g., adults) were able to use it, despite the boring calibration phase. Children were very impatient and were not able to use the system. Among other problems, they noted the need to keep the head still.

Starker and Bolt [74] tried to design a non-command [57] eye-tracking based interface. The application is based on "The Little Prince" novel of Antoine de Saint Exupery. The user's interest in objects is measured by the length of fixation and the interest levels "age" if they are not refreshed by fixations. The user will automatically obtain information about an object if the system "thinks" that user's interest level for it is high.

Jacob [41] conducted extensive research on eye-tracking based HCI. In his experiments, the Applied Science Laboratory [1] eye-tracking equipment based on infrared reflections was used. In such systems, the user's head must remain stationary. Several improvements of the use of raw eye-tracking data were proposed. First, he discovered that calibration was not uniform across the screen, e.g., more imprecise at some locations. To correct this, the user could move the mouse pointer to the problematic area and stare it for a while while the system re-calibrates. Second, if the user fixates slightly off-target, the system accepts that as target fixation. To ensure that there is no mistake in the selection, the system will accept off-target fixation only if the fixated point is far away from other targets. Finally, the problem with raw eye-tracking data is that some noise could be present. The noise can come either from unconscious, jittery eye-movement, or blinking, or some system measurement error. To avoid the noise. data tokeniz

msec interval. the m

tions instead of st

Kaufman *et al*

oculography describ

walking. They test

accuracy was 73 %. I

LC Technologies. I

programs similar to

Recently, several

been developed in r

commercial yet. B

degree accuracy, co

neither specify the r

Stiefelhagen *et al.*

1-5 degrees. They

use their system (

so that the signal

2.6.7 Probl

In this Section, s

detection system

noise, data tokenization is used: instead of reporting all the tracking data, after 100 msec interval, the mean gaze position is reported. The resulting data is a string of *tokens* instead of stream of eye-tracker measurements.

Kaufman *et al.* [44] designed an HCI based on eye-tracking using electro-oculography (described in Section 2.4.5). The selection mechanism is blinking or winking. They tested menu selection using 3 × 2 menu buttons, and the achieved accuracy was 73%. In the case of selecting corners only, the accuracy was 90%.

LC Techologies, Inc. [4] offers commercial systems that use technology and offer programs similar to those of Hutchinson *et al.* [40].

Recently, several systems based on Artificial Neural Network gaze detection have been developed in research laboratories. The applications of such systems are not commercial yet. Baluja and Pomerleau [11] reported that they could achieve 1.5 degree accuracy, compared to 0.75 degrees in the commercial trackers. However, they neither specify the methodology for evaluations, nor the number of subjects. Similarly, Stiefelhagen *et al.* [75] reported errors in rotation angle around x, y, and z axis of 1–5 degrees. They use their system for panoramic image viewing. Additionally, they use their system [95] to locate head and perform lip reading, and to locate speakers so that the signal coming from a microphone can be enhanced.

2.6.7 Problems with Eye-Tracking Systems

In this Section, several problems related to the state-of-the-art eye-tracking and gaze detection systems are discussed.

Non-Intrusivenes

"The ey
the user
a conve
the eye tr
red light,
and, mos
results in
one the e

This observat
trackers. Even
required to b
use of infrare

Calibration: In m
is required to
in an annoyin
with calibrat
tracking, the
not require c
would be tru

Non-Intrusiveness: Jacob wrote in [41]:

> "The eye tracker is, strictly speaking, non-intrusive and does not touch the user in any way. Our setting is almost identical to that for a user of a conventional office computer. Nevertheless, we find it difficult to ignore the eye tracker. It is noisy; the dimmed room lighting is unusual; the dull red light, while not annoying, is a constant reminder of the equipment; and, most significantly, the action of the servo-controlled mirror, which results in the red light following the slightest motion of user's head gives one the eerie feeling of being watched."

This observation illustrates the non-intrusiveness problem related to most eye-trackers. Even though the equipment does not touch the user, the user is often required to be completely still. Additionally, some question the health of the use of infrared lighting.

Calibration: In most eye-tracking systems, the calibration step is a must. The user is required to calibrate the system before each session, and this might result in an annoying situation if the system is to be widely used. Another problem with calibration is that if the user moves out of focus of the camera that is tracking, the system would need to be re-calibrated. Thus, systems that would not require calibration before use, and that would enable free head motion, would be truly non-intrusive and user-friendly ones.

Midas Touch: T

Often the t

the user re

point with

eyes when

wherever v

Midas To

user's inte

a better F

command

2.6.8 Hea

Heuvelmans

head moveme

transducers.

a head-mount

differing in

the screen.

for a prede

used a typ

In their se

application

Midas Touch: The problem of the selection mechanism in eye-trackers is persistent. Often, the target is selected after long fixation. However, how do we know that the user really wants the target to be selected? The user might just stare at a point without the intention to select an item. Since we cannot "turn off" our eyes when we don't want to control the system, we could enter a state in which wherever we look we activate some command, and the interface would have the Midas Touch problem. An eye-tracking system should be able to distinguish user's intent to select an item, from the state of observance. In our aim to create a better HCI, we are faced with a need to enable the user to seamlessly activate a command without the need to perform some complicated and unnatural action.

2.6.8 Head-Controlled Systems

Heuvelmans *et al.* [39] designed a handless interface for disabled persons based on head movements. The system has a head-borne mouse replacement unit based on transducers. Ultrasonic sound waves are emitted by a control unit, and are received by a head-mounted transducer that converts the sound information into electronic signals differing in phase, which are further used to calculate the position of the pointer on the screen. The selection signal is a "dwell button", that is, the user fixates a point for a predefined period of time. The system was used by two disabled persons, who used a typewriting application and achieved a typing speed of 60 characters/minute. In their setup, the smallest button was 34 × 20 pixels or 11.1 × 6.5 mm and all the alphanumeric keys and space bar are displayed in the lower portion of the screen.

Additionally, the

middle of the scr

Beardsley [1

on matching a

head pose. In

the image and r

generated, wh

driver is looking

model segment

can say approxim

Bradski [13]

and graphics. T

extension of th

Based on an HS

is determined.

information. Th

and up down m

2.7 Visu

In studies on h

important. Co

pattern of exp

Additionally, the screen has several control buttons arranged in a top row and in the middle of the screen is a text display area.

Beardsley [12] designed a system for monitoring of drivers. The system is based on matching against synthetic images from a 3D model to determine an approximate head pose. In the initialization step, the 3D model of a head is projected onto the image and manually adjusted. The model is rotated and template images are generated, which are used for later comparison. To better discriminate whether the driver is looking up or down, the state of the eyelid is determined using a skin color model segmentation of eye area. Based on the knowledge of the interior of the car, they can say approximately where the driver is looking (e.g., rear mirror or dashboard).

Bradski [13] one used head motion as an input for applications like computer games and graphics. The system is based on the mean shift algorithm, and they proposed an extension of the algorithm called CAMSHIFT (Continuously Adaptive Mean Shift). Based on an HSV color histogram, the probability distribution of the head location is determined. Based on zeroth, first and second moments they determine head orientation. The controls for the computer game are based on left/right, back/forth and up/down motion.

2.7 Visual Perception Studies

In studies on human visual perception, eye movement during scene exploration is very important. Cognitive scientists present various scenes to subjects, and observe the pattern of exploration. The subject might be asked to explore the scene for some clues or to just examine

during the explora

are explained, then

discussed.

2.7.1 Types

When closely exa

types of eye move

portion of the eye

and it covers abo

the periphera and

that covers abo

can perceive in

Saccades are

of a visua

min mak

it takes a

it (depen

of 40 d

once it

movem

or to just examine it freely. Yarbus [97] illustrates seven records of eye movements during the exploration of one picture. In this section, first, the types of eye movement are explained, then, the equipment used in cognitive science studies is described and discussed.

2.7.1 Types of Eye Movement

When closely examined, the patterns of eye movement show that there are several types of eye movements. The structure of a human eye is such that only one small portion of the eye has densely packed receptive cells. That area is called the *fovea*, and it covers about 2 degrees of viewing angle. The area next to the fovea is called the *parafovea* and covers about 5 degrees, and everything else falls into the *extrafovea*, that covers about 60 degrees and is perceived blurry by a human. The area that we can perceive in the fovea is equivalent to a word in a page at normal reading distance.

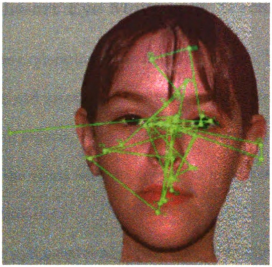
Saccades are the principal method of moving the fovea to focus on different portions of a visual scene. There can be 250 saccades per minute, error, since 250 per min makes sense in some cases even 900! When the visual stimuli is presented, it takes about 100–300 msec to initiate a saccade, and 30–120 msec to complete it (depending on the angle traversed, ranging from minimum of 1 to maximum of 40 degrees, most typically 15–20 degrees). A saccade could be voluntary, but once it starts, it cannot be suppressed, nor can its path be changed. During the movement, vision is suppressed. Some studies show that the suppression is not

Figure 2.4: Example
tesy of John M. H
http://eyelab.msu.

complete. Once
200 600 msec, a

Pursuit motion is a
object, so that i
be induced by in

Nystagmus occurs
so that the pur
from the field o
the opposite d



[Image is presented in color.]

Figure 2.4: Example of eye movement during face learning and recognition. (Courtesy of John M. Henderson, Michigan State University Eyetracking Laboratory, `http://eyelab.msu.edu/`)

complete. Once the fovea focuses on the object, the examination lasts about 200–600 msec, and this period is called *fixation.*

Pursuit motion is a smoother, slower movement of the fovea. It follows a moving object, so that it is foveated. This kind of movement is not voluntary, but can be induced by introducing a moving object in the visual field.

Nystagmus occurs as a response to moving one's head. The fovea is slowly moved so that the pursued object's image is followed. If the followed object disappears from the field of view and appears on the other side, the fovea rapidly moves in the opposite direction. In this way we can follow repetitive patterns.

2.7.2 St

Figure 2.4 sh

the subject

to eyes, nos

of image is

How is

The sta

23] and is

forehead re

that the h

calibratio

that the v

calibrated

that the c

Numer

to it. Ho

movement

similar sy

The a

l are, an

based sys

2.7.2 State-of-the-art Eye Movement Tracking Technology

Figure 2.4 illustrates the exploration of a sample face image. The lines indicate where the subject was looking, and we can see clearly that most of the attention was devoted to eyes, nose and mouth regions, while background remained unexplored. This kind of image is typical for the visual perception and eye movement studies [38, 37].

How is this data obtained?

The state-of-the-art equipment used is called the "Purkinje Image Eyetracker" [22, 23], and is illustrated in Figure 1.1. The subject's head is placed within a frame, the forehead rests on the forehead rest, the bite-bar is in subject's mouth. All this ensures that the head stays perfectly still during an experiment. Before a session begins, a calibration procedure is done: the subject looks at several points on the display, so that the vectors defined by the Purkinje images (see Section 2.4.2) can be properly calibrated. It is common that the system gets uncalibrated during the session, so that the calibration procedure must be repeated.

Numerous studies have been done using this procedure and subjects can get used to it. However, the equipment is highly intrusive and the user has no freedom of movement as he or she would have in a natural setting. A system that would provide similar eye-tracking data, but in a more natural setting, would be of great benefit.

The accuracy of the above eye-tracking equipment is high, about 0.75 degrees or 1 arc, and the sampling rate is about 1000 Hz. The challenge of a computer vision based system would be to achieve similar accuracy and performance rates.

Chapter

Face anc

This Chapter desc
for location of fac
or normalized val
hue, saturation an
...e. Some experi
subjects, and pos
features like the
presented. The p
method is evalu
dark, and result

The method
does not overla
a beard or mus

Chapter 3

Face and Facial Feature Detection

This Chapter describes a method for face localization in a color image and a method for location of facial features. For the face localization, a skin color model based on normalized values of red and green components is used. A classification using the hue, saturation and intensity color space is also presented, and compared with a RGB one. Some experimental results are presented, on both light-colored and dark-colored subjects, and possible improvements are discussed. A novel method for locating facial features like the eyes, eyebrows and nose based on the knowledge of face geometry is presented. The parameters for evaluation of eyes-nose matches are introduced. The method is evaluated on a number of test images of various subjects, both light and dark, and results are presented.

The method works under an assumption that there is one face in the image that does not overlap with other skin-color-like objects. If the person wears glasses or has a beard or mustache, the method sometimes does not work well.

3.1 Fa

This Secti

face using

applied to

3.1.1 S

c

The locati

skin color

HSI Hue.

a discussi

shows skin

Caucasian

red and gr

R_{norm} =

Figure

column. Y

represent

The de

be broken

as Face. S

a connected

3.1 Face Detection

This Section describes the development of the skin color model and detection of a face using a connected components algorithm. Problems with the skin color model applied to dark faces are addressed.

3.1.1 Skin Color Model in RGB Color Space and Face Localization

The location of the face in an image is based on the skin color model. Human skin color can be represented as a cluster in either the RGB (Red, Green, Blue) or HSI (Hue, Saturation, Intensity) space. In this work, the RGB space is used, and a discussion on the use of the HSI space is presented in Section 3.1.2. Figure 3.1 shows skin color clusters obtained from the images of 4 subjects of different skin color (Caucasian, Asian, and Indian). The clusters are based on the 2D plot of normalized red and green components values:

$R_{norm} = \frac{R}{R+G+B}$, and $G_{norm} = \frac{G}{R+G+B}$

Figure 3.2 depicts the classification results for the sample images in the middle column. Yellow color represents the primary skin color, and blue and pink colors represent the shadow areas.

The detection of face region boundaries in an input image (Figure 3.2 (a)) can be broken into three steps. The first step labels all the pixels in the input image as `Face`, `Shadow_1`, `Shadow_2` or `Other` (Figure 3.2 (b)). The second step applies a connected components algorithm to find the components of the first three labeled

Figure 3.1: Skin co
G_{norm}. Out of six
primary skin color.
beard and shadows

classes. A row-by-r

too small or too big

to about 100. the m

biggest object of th

obtains a compact

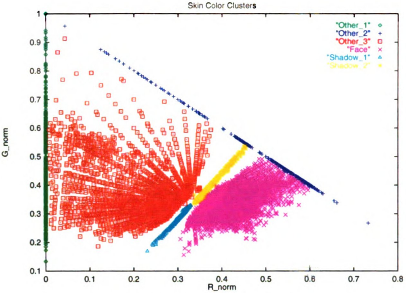
in the case whe

cheeks and beard

necessary in ord

the mask around the

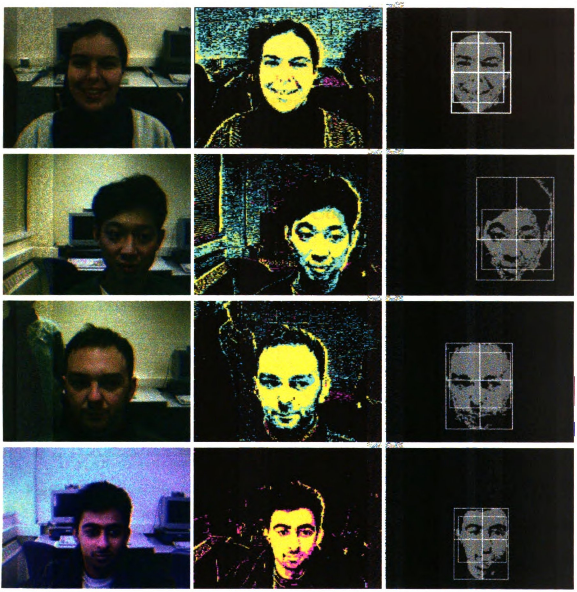
samples to obtai



[Image is presented in color.]

Figure 3.1: Skin color clusters: the horizontal axis is R_{norm} and the vertical axis is G_{norm}. Out of six clusters, three represent skin color. Cluster labeled **Face** is the primary skin color, and clusters labeled **Shadow_1** and **Shadow_2** are areas of shaved beard and shadows around eyes.

classes. A row-by-row algorithm is used for computational efficiency. Objects that are too small or too big are discarded, thus reducing the number of objects from 500–700 to about 100, the majority of which are in the shadow classes. The third step finds the biggest object of the class **Face**, and merges it with objects that border it. This step obtains a compact face region even in the case when the subject's face is in shadow or in the case when the subject has a beard. In some cases, the subject's forehead, cheeks and beard will be separate components. Thus, merging of these components is necessary in order to obtain the correct face region. The resulting bounding box is shrunk around the first moment lines using heuristics learned from processing many examples to obtain the final face region bounding box (Figure 3.2 (c)). The outer



(a) Original Image (b) Skin Color Classification (c) Face Region Boundaries

[Image is presented in color.] , (Pictures are part of the author's private database.)

Figure 3.2: Isolated face regions on sample images

box is the bou

area determin

This thre

ciding two o

For a very sm

skin model w

lighting was s

3.1.2 Ski

The skin color

color space.

equations:

$I = R$

$S = 1.0$

$H =$

Figure 3.3

identified, sin

labeled as Fa

and Shadow_2

Figure 3.4

and HSI mod

colors represe

color with sim

box is the bounding box for the original face object, and the inner box is the target area determined from the first moments lines.

This three-step approach has been tested on many images of many persons, including two open-house sessions when dozens of unfamiliar people tested the program. For a very small number of subjects, the skin color model did not work well. The skin model worked well even for subjects with very dark skin, provided that ambient lighting was strong.

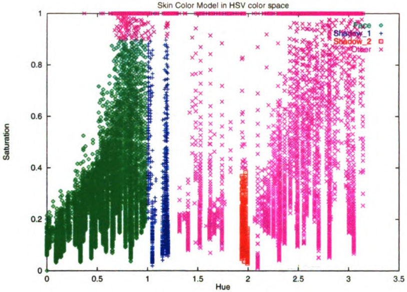
3.1.2 Skin Color Model in HSI Color Space

The skin color model can be developed using the Hue, Saturation and Intensity (HSI) color space. Translation from the RGB to HSI space is done using the following equations:

$$\begin{aligned} I =& \; R+G+B \\ S =& \; 1.0 - \frac{3\min(R,G,B)}{I} \\ H =& \; \frac{\arccos(2R-G-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)})} \end{aligned}$$

Figure 3.3 shows the plot of hue vs. saturation. Three skin color clusters are identified, similar to the clusters used in the RGB space: the primary face color (labeled as **Face**) and areas of beard and shadows around eyes (labeled as **Shadow_1** and **Shadow_2**).

Figure 3.4 compares the classification results for the sample images using the RGB and HSI models. Yellow color represents the primary skin color, and blue and pink colors represent the shadow areas. As can be seen, both models are classifying skin color with similar success. The RGB model is classifying darker skin color a bit better



[Image is presented in color.]

Figure 3.3: Skin color clusters in HSI color space: the horizontal axis is *hue* and the vertical axis is *saturation*. Skin color is represented with three clusters. Cluster labeled **Face** is the primary skin color, and clusters labeled **Shadow_1** and **Shadow_2** are the areas of shaved beard and shadows around eyes.

than HSI model. Another advantage of the RGB model is that the values are available directly from the camera interface, while for the HSI model, the values need to be calculated for each pixel, and the computation could be time consuming.

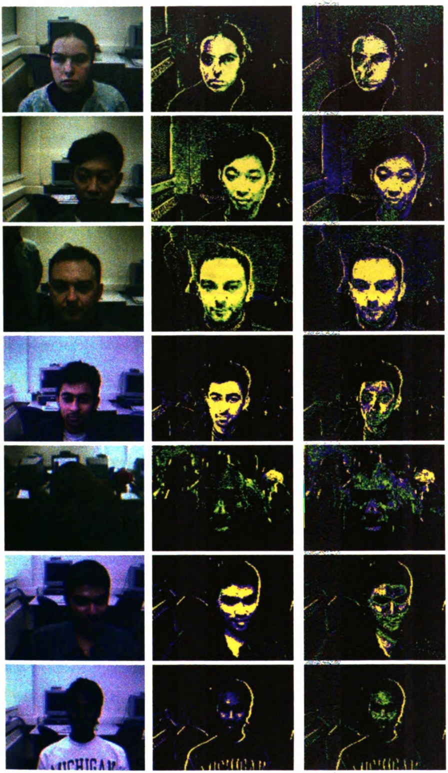
3.1.3 Problems with Skin Color Model

The processing using the skin color model gives erroneous results in some cases. The most common errors occur when the face object is close to some object of similar color, especially the subject's clothes. Another object can be recognized as a face, or will create one component with the face. Figure 3.5 illustrates this problem. If another object and the face are recognized as one, the algorithm has some chance of locating the eyes and nose. However, if the unwanted object is recognized as the face

a

Image is

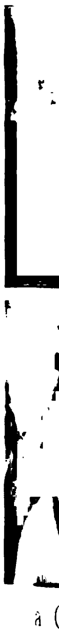
Figure 3.



(a) Original Image (b) RGB Classification (c) HSI Classification

[Image is presented in color.] , (Pictures are part of the author's private database.)

Figure 3.4: Comparison of RGB and HSI models for skin color classification



a (

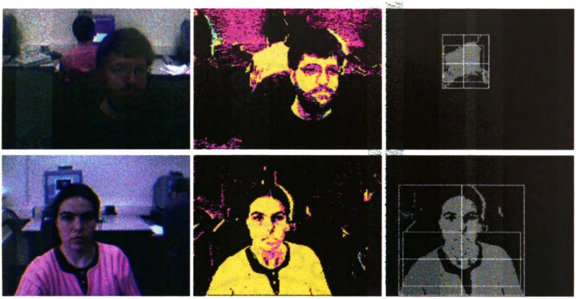
Image

Figure

3.1.4

of dark sh

However,



(a) Original Image (b) Skin Color Classification (c) Face Region Boundaries

[Image is presented in color.] , (Pictures are part of the author's private database.)

Figure 3.5: Erroneous face regions due to similar background and face colors.

object, the real face is lost. Some control of the environment greatly reduces these problems and maintains real-time speed.

3.1.4 Problems with Dark Skin Color

The skin color model described above works well for light colored faces. In the case of dark skin color, if the room is well lit, the model will be able to segment the skin. However, if the lighting is not strong enough, or if the subject's skin is very dark, the model will not cover that skin type. As an illustration, Figure 3.6 shows images taken under the same settings as the other test images, where the skin color model classification fails in two cases (first two rows) and succeeds in one case (third row). As can be seen, the first two subjects' skin color was classified as mainly shadow area, thus the connected components algorithm could not find a big enough component

(a

Imag

Figure 3
classifica

of the fa

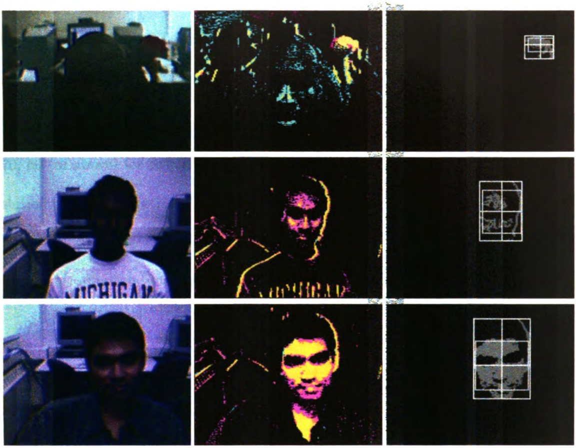
program

arbitrary

3.2 I

The main

d i



(a) Original Image (b) Skin Color Classification(c) Face Region Boundaries

[Image is presented in color.] , (Pictures are part of the author's private database.)

Figure 3.6: Localization of dark skin color subjects: two failed and one successfull classification

of the face color. Big connected components of shadow class are discarded by the program as possible background. In the third, successful case, the combination of the primary skin color and one shadow class enabled successful face localization.

3.2 Improvement of Dark Skin Color Detection

The main problem in detecting dark skin color is that the skin color is not well clustered in $R_{norm} \times G_{norm}$ space. The skin points are mainly spread across the area of cluster labeled

blue intensities h

illustrates the his

As can be see

70, 60) or above

shifted to have t

image in such a

The transforma

$I_{altered}(i, j, c$

where, $I(i, j, c$

$hist\_peak_c$ is d

$a1_c =$

$a2_c =$

$b2_c = 25$

and $const_R =$

The dark

$hist\_peak_c$ va

$[const_R, \frac{3}{4}co$

be a dark sk

If the ab

see the imag

the skin colo

of cluster labeled "Other_2" in Figure 3.1. However, histograms of red, green and blue intensities have the same shape for both dark and light skin tones. Figure 3.7 illustrates the histograms for five windows.

As can be seen, the peaks of the histograms for the light skin tone are around 80, 70, 60 (or above) for red, green and blue, respectively. If the dark skin histogram is shifted to have the peak at the same value as the light skin one, we can transform the image in such a way that the skin color then fits into the proposed skin color model. The transformation function used is the following:

$$I_{altered}(i,j,c) = \begin{cases} a1_c * I_{orig}(i,j,c), & \text{if } I_{orig}(i,j,c) < hist\_peak_c; \\ a2_c * I_{orig}(i,j,c) + b2_c, & \text{elsewhere,} \end{cases}$$

where, $I(i,j,c)$ is image intensity at row i, column j, and of color $c \in \{R,G,B\}$, $hist\_peak_c$ is determined from the corresponding color histogram, and

$$a1_c = \frac{const_c}{hist\_peak_c}$$

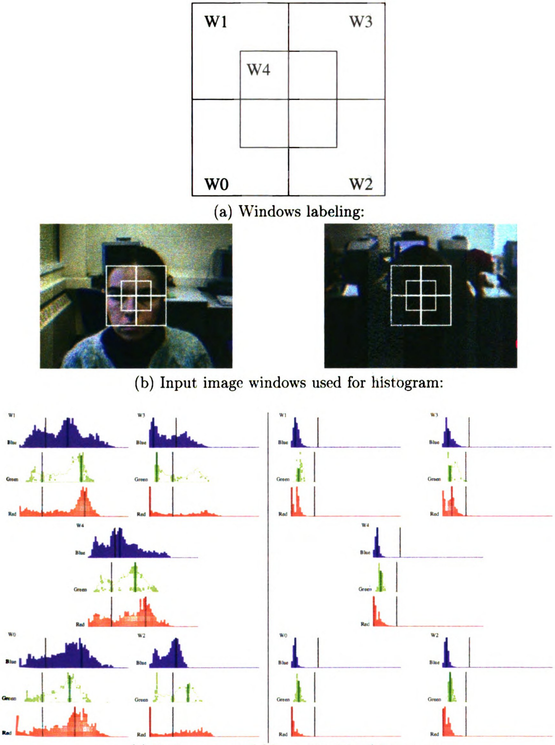
$$a2_c = \frac{255-const_c}{255-hist\_peak_c}$$

$$b2_c = 255\frac{const_c-hist\_peak_c}{255-hist\_peak_c},$$

and $const_R = 80$, $const_G = 70$ and $const_B = 60$.

The darkness of each window can be checked automatically, by comparing the $hist\_peak_c$ value with boundary values for R,G,B, that are empirically found to be $\frac{1}{3}const_R$, $\frac{3}{4}const_G$ and $const_B$, respectively. Then, the window can be considered to be a dark skin area if at least the three leftmost or rightmost windows are dark.

If the above formulae are applied to the first two images from Figure 3.6 (a), we get the images where the skin is easily classified. Resulting altered input images and the skin color classification results are shown in Figure 3.8.



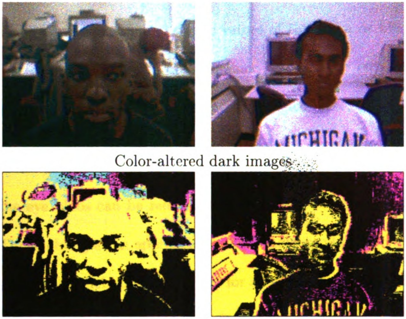
(a) Windows labeling:

(b) Input image windows used for histogram:

(c) Red, green and blue intensities histograms:
Black line is dark-light skin boundary, dark red, green, blue bars are peak values

[Image is presented in color.] , (Pictures are part of the author's private database.)

Figure 3.7: Red, green and blue intensity histograms for dark and light skin tones



Color-altered dark images

Skin color classification of altered images

[Image is presented in color.] , (Pictures are part of the author's private database.)

Figure 3.8: Color-altered dark images, and resulting skin-color classification

3.3 Eyes, Eyebrows and Nose Location

Once the face target area is found, the search for the eyes, eyebrows and nose is limited to its bounding box. A greyscale image is used for this, specifically, the smoothed red component of the input color image. Finding the eyes, eyebrows and nose is based on the knowledge of the face geometry. In this Section, the algorithm and heuristics used are described in detail.

3.3.1 Detecting Eyes

The eye pupils are the darkest objects on a typical human face, regardless of the eye color. If the face and eyes are brighter, the pupils will be brighter, but still the

darkest overall.
eye pupils will b
Figure 3.9. first
components of a
in the red image

Finding the e
face image. Thres
problem is that th
is applied until tw
of the red. green a
seen. in the red c
component they t
are not visible b
often of bluish or
pupil area. and w

For each thre
blobs. Blobs tha
sized are rejecte
image. The thre
is found. For fa
eye pupils or for
to be below thre

darkest overall. The red component of a color image absorbs the most light, thus the eye pupils will be darkest, compared to the values of the green or blue component. Figure 3.9, first column, illustrates the difference between the red, green and blue components of a color image. Since our skin is of reddish color, skin is rather bright in the red image compared to the green one.

Finding the eye blobs can be reduced to finding the two darkest objects in the face image. Thresholding at an appropriate value would produce two eye blobs. The problem is that the threshold is not the same for all eyes. Thus, gradual thresholding is applied until two dark blobs are found. Figure 3.9 shows results of thresholding of the red, green and blue component at gray levels $1, 5, 10, 15, ..., 35, 40$. As can be seen, in the red component image the pupils are easily identified, while in the green component they become visible at a higher threshol, and in the blue component they are not visible below the threshold level of 40. Additionally, since human eyes are often of bluish or greenish color, these components would have high intensity in the pupil area, and would not be the darkest area in a face image.

For each thresholded image, a connected components algorithm is run to find dark blobs. Blobs that are at the edges of the target area, too small, too big, or sparsely filled are rejected. Figure 3.10 depicts the results of gradual thresholding for a sample image. The threshold is increased from 0 until a successful detection of eyes and nose is found. For faces with dark eyes, smaller thresholds result. For faces with bright eye pupils or for very bright images, higher thresholds are needed to get the eye blobs to be below threshold.

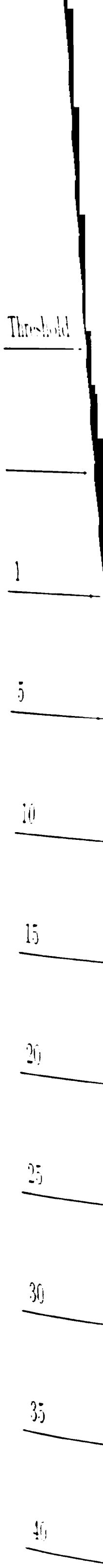
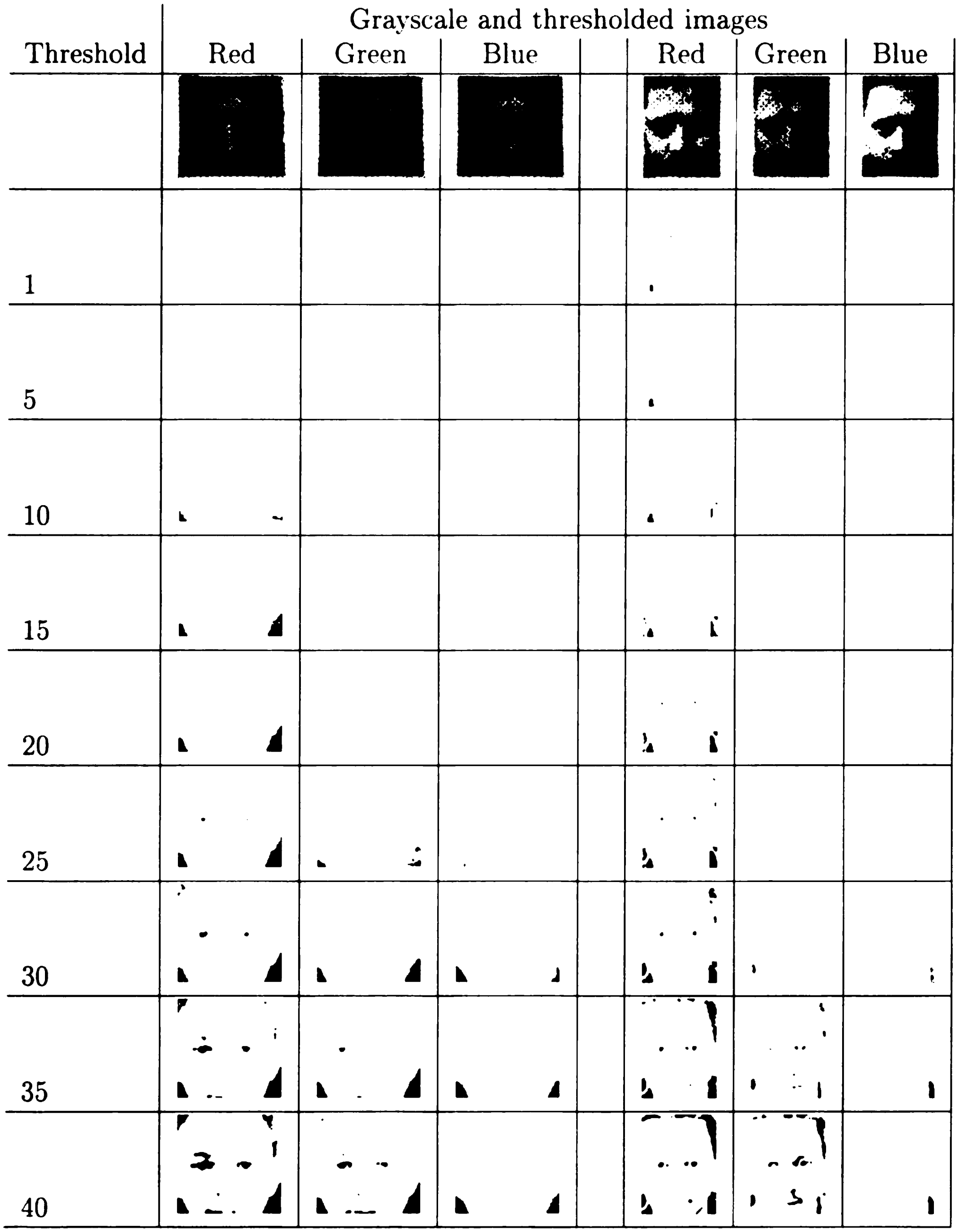


Figure 3.9: Re
search for eye



(Pictures are part of the author's private database.)

Figure 3.9: Red, green and blue components of two images: gradual thresholding in search for eye blobs

Wh

thresh

value t

prior k

algorit

conditi

3.3.2

As can

two dat

ere it is

the eyeb

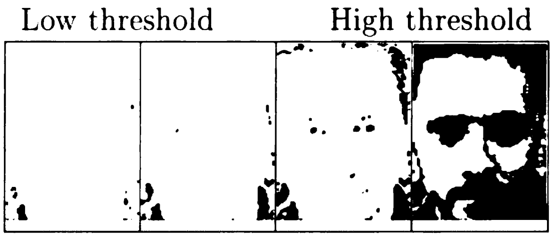
20-30 pix

eyebrow a

All the

to find the

intensity-th



(Pictures are part of the author's private database.)

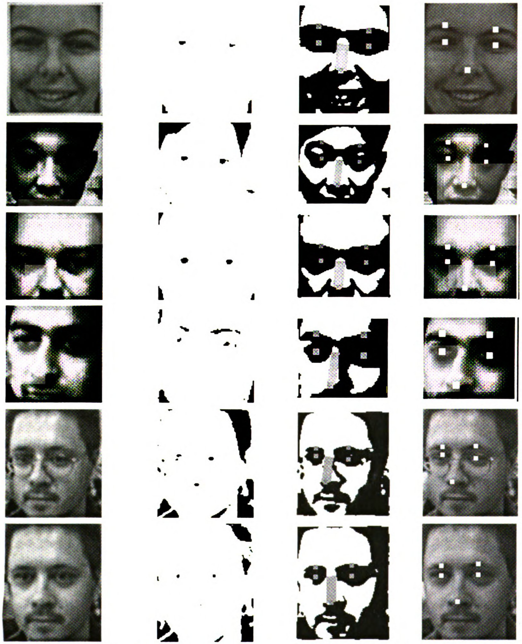
Figure 3.10: Gradual thresholding in search for eye pupils

When only one image is processed and there is no prior knowledge of the suitable threshold, it is necessary to try thresholds starting from the lowest value up to the value that yields a satisfactory match. However, in the case of continuous processing, prior knowledge of the threshold for the previous frame can be used. Thus, the algorithm can adapt itself to the current subject's face and to the current lighting conditions.

3.3.2 Matching Eyes with Eyebrows and Nose

As can be seen in Figure 3.11, at the best threshold we typically do not get only two dark blobs, but often more than two. For each blob that is a candidate for an eye, it is first checked whether an eyebrow is present above the eye. The existence of the eyebrow is verified by looking for an edge in the average threshold image that is 10–30 pixels away from the eye. The blobs that do not have the edge representing an eyebrow are discarded and not considered candidates for the eyes.

All the remaining blobs are assumed to be candidates for the eyes, and are matched to find the nose. The image that is used to find the nose is the red component, intensity-thresholded at the average intensity. When finding the nose, it is assumed



(a) Original image (b) Best thresholding (c) Nose line (d) Feature points

(Pictures are part of the author's private database.)

Figure 3.11: Matching eye blobs with nose

that most of the lighting on the face is from above. It is also assumed that the nose line is perpendicular to the line between the eye pupils, and that it intersects the line between the eye pupils roughly at the midpoint of that line. The precise location of the beginning of the nose line is the mid-point of the widest above-threshold region on the eye line. The estimated nose width is equal to the width of that above-threshold region. The beginning point of the nose line is on the eye pupils line and the nose line ends at the tip of the nose. Then, the nose line is pursued downwards, using the estimated width, until a significant below-threshold area is found. That point is considered the tip of the nose. The shaded areas in the Figure 3.11(c) represent the traced nose line. Each eyes-nose match is evaluated based on several heuristics. The match with the highest score is selected as the eyes-nose match.

Scoring rules are based on the comparison with the current face region dimensions. Thus, the user's face does not have to be at a fixed distance from the camera, since the parameters that are used for match scoring will automatically get re-adjusted when the user's face moves. The only constants used are the approximate relationships for eyes and nose distances, and they were determined empirically.

Each match is evaluated using the following rules:

1. If the Euclidean distance between the eyes is smaller than 25% of the target area width, or greater than 70% of the target area width, the match is rejected.

2. If the Euclidean distance between the eyes differs more and 9% of the expected eye difference, the match is rejected.

3. Add $100 \cdot (1 - \mathrm{abs}(\frac{\mathrm{eye1\_y} - \mathrm{eye2\_y}}{\mathrm{eye1\_x} - \mathrm{eye2\_x}}))$ to punish if the slope is too big. It is assumed that the subject will not turn the head for more than 45 degrees. If the slope is more than 45 degrees, the match is automatically rejected.

4. If one of the eyebrows has not been found, but estimated, the whole match value is multiplied by 0.9.

   Add percentage of fullness of the forehead line with respect to the skin clustered points. The forehead line is the line between two eyebrows.

   Add $100 \cdot (1 - \frac{\mathrm{abs}(\mathrm{eyebrow1}_{dist} - \mathrm{eyebrow2}_{dist})}{\max(\mathrm{eyebrow1}_{dist}, \mathrm{eyebrow2}_{dist})})$ to reward symetric eyebrows. $\mathrm{eyebrow}_{dist}$ is defined as the shortest distance between eyebrow and eyeline, and is calculated as the number of points from eyebrow to eyeline along the eyeline normal.

5. If calculated_nose_width < min_nose_width, subtract 10 to punish the absence of above-threshold area on eyeline, and min_nose_width = 0.1eye_distance.

6. If calculated_nose_width > max_nose_width, add $30 \cdot (3 - \frac{\mathrm{calculated\_nose\_width}}{\mathrm{perfect\_nose\_width}})$ to punish wide above-threshold area on eyeline, and perfect_nose_width = 0.3eye_distance, and max_nose_width = 0.5eye_distance.

7. Add $100 \cdot (1 - \frac{\mathrm{num\_black\_points}}{\mathrm{total\_points\_on\_noseline}})$ to reward above-threshold areas on the noseline.

   Add one half of the average gray value on the nose line, to reward bright areas, and punish dark areas.

The problems with the above algorithm occurred when the lighting conditions were such that one side of the face was in shadow and the other side was directly illum

The

extre

tha

the

as t

an

3.4

The a

light s

a way

with a pla

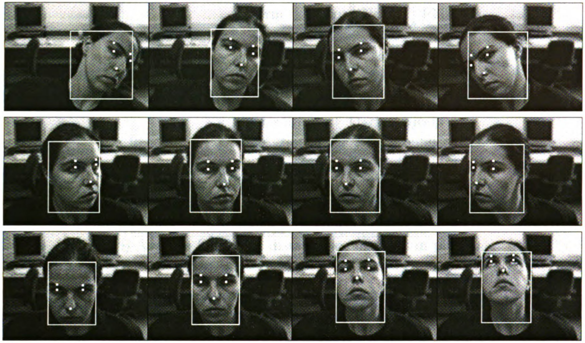
illuminated. In such cases, both eyes could not be found using the same threshold. The solution for this problem would be that the subject does not work under such extreme conditions. It is likely that any subject who is working in front of a workstation would not like to be directly illuminated, since a lot of reflection is present and the screen cannot be seen clearly.

Another solution to this problem is to remember which blobs are growing in size, as the threshold increases, and to use their locations in a higher threshold. In that way, we can compensate for the disproportional lighting on the faces sides.

Figure 3.12 shows examples of the feature detection for various pan, tilt and roll angle rotations, some of which are extreme. The program imposes a limitation of up to 45 degrees of roll angle. The eye features are lost if the head is panned completely to the left or right. As roll and pan angles increase, so does the error in detecting the tip of the nose.

3.4 Accuracy of Feature Detection

The accuracy of the feature point detection was tested on four subjects (two with light skin color and two with dark skin color). The continuous stream of images was recorded while the subject was moving, e.g., rotating the head or moving toward or away from the camera. Two sets of images were recorded: one with a normal background (e.g., white walls, workstations in the background, etc), and another one with a plain green background. Figure 3.13 illustrates the input images used.



(Pictures are part of the author's private database.)

Figure 3.12: Results of eyes-nose detection for various roll, pan and tilt angles. The white rectangle shows face boundaries, and the white dots show the locations of the eyes, eyebrows and tip of the nose.

[Image is presented in color.] , (Pictures are part of the author's private database.)

Figure 3.13: Sample images used for accuracy of feature detection calculation

Table
Euclid
tive re

Th

150 fra

tance b

Points

Tab

correct

of eucli

For t

the plai

normal b

was 0-1.6

with the

	Light subjects Background		Dark subjects Background	
	Norm.	Plain	Norm.	Plain
Matches(%)	93.00	99.00	58.33	80.00
Outliers(%)	5.00	0.00	5.00	1.67
Not found(%)	2.00	1.00	36.67	18.33

(a) Percentage of matches found, matches not found, and outliers.

	Light subjects Background		Dark subjects Background	
	Norm.	Plain	Norm.	Plain
Left Eye	1.04	1.09	1.59	1.59
Right Eye	1.52	1.52	2.78	1.58
Nose	4.05	3.77	3.53	2.63

(b) Average Euclidean distance for matches in pixels.

Table 3.1: Accuracy of the feature location. Percentage of found matches and average Euclidean distance of program-found features from hand-labeled location. Comparative results for normal environment vs. controlled (plain green) background are given.

Then, a random sample of (50 or 30) video frames was selected (from 300 or 150 frames). The eyes and nose locations were hand-labeled and the Euclidean distance between the labeled features and the program-detected features was calculated. Points that had Euclidean distance of more than 10 were considered outliers.

Table 3.1 shows the average Euclidean distance in the set of images when the correct match was found. Also shown: the percentage of correct matches, percentage of outliers and percentage of times when the match was not found.

For the normal background, matches were found for 58–93% of the frames; for the plain green background the success was 80–99%. The percentage of outliers for normal background was 5%, while the percentage of outliers for plain background was 0–1.67%. For the dark-skin subjects, the percentage of missed matches was 36% with the normal background, and 20% with the plain green background. Clearly, the plain green background enhanced the performance. The error measured as the average Euclidean distance between the hand-labeled and the program generated feature points was 1–4 pixels. The error for the eyes was very small (1–2 pixel off), while the error for the tip of the nose was larger (2–4 pixels). This discrepancy is due to the lighting conditions and our assumption that the line between the eyes and down along the nose make a 90^o angle. When the face is rotated significantly the angle is smaller than 90^o. However, the error made is still small, and the amount of computation is reduced since only one angle is considered. The eye blobs of the dark-skin subjects were found at very low thresholds (0 or 1 in most cases), and the threshold increment had to be modified to 1 instead of 5, which was used for light-skin subjects. This problem can be overcome with increased ambient lighting or more gradual increments in the thresholding based on the knowledge that a dark-skin subject is using the system.

3.5 Summary

In this Chapter, a method for face localization in a color image was presented. The method is based on a skin color model that covers a wide range of faces, from light to dark ones. Both models using RGB and HSI color space are presented and compared. Some problems with the skin color model due to background similar to the face color were presented, as well as problems in the detection of dark skin. A method of color alteration for improving dark skin detection was presented. A novel method for locating facial features based on the knowledge of the face geometry was presented.

The method finds the eyes, eyebrows and nose of a subject. The evaluation of the accuracy of the feature detection showed 1–2 pixel accuracy for detecing the eyes and 2–4 pixel accuracy for detecting the tip of the nose.

C

T

In C

The

sub

the

amo

pres

varia

of the

the dis

algorithm

represent

Present

Chapter 4

Tracking the Features

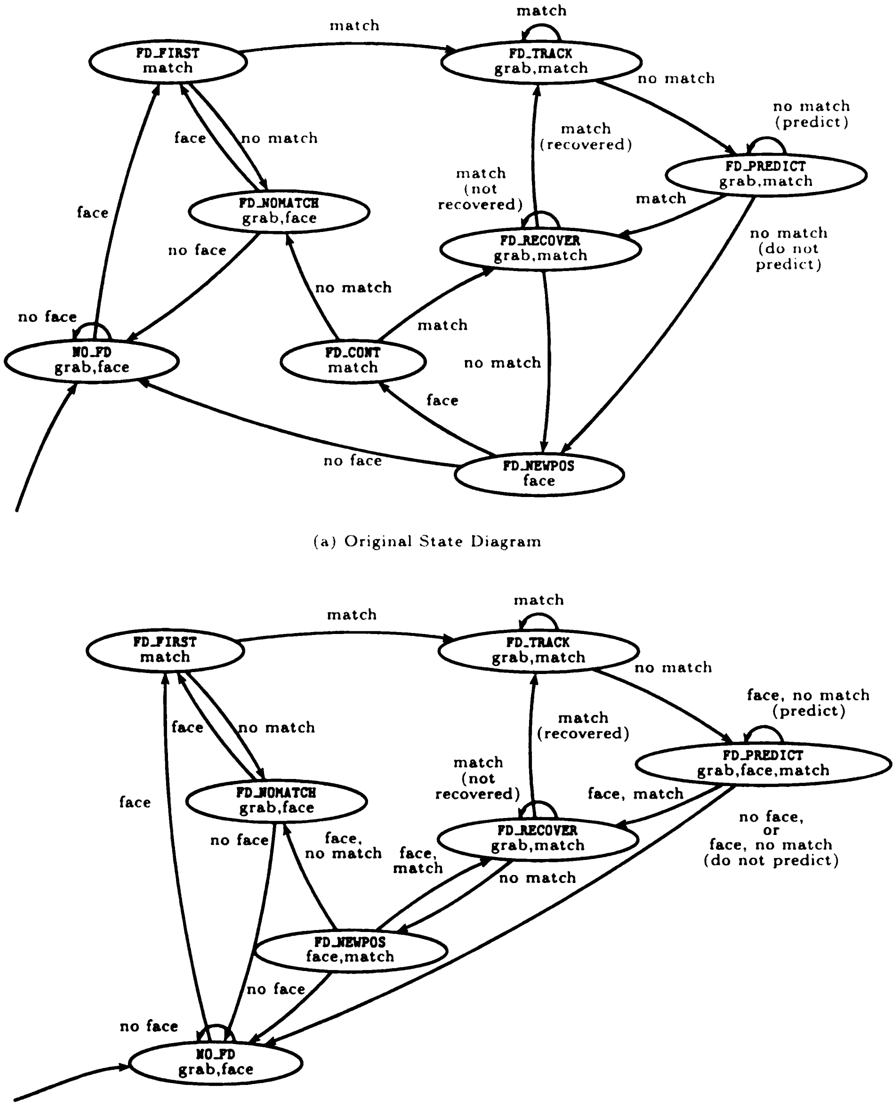
In Chapter 3, the algorithm for face location and facial feature location was described. The algorithm is designed to work on individual images. In this Chapter, it is described how the algorithm from Chapter 3 has been integrated in the processing of video stream data. The input to the base program is a live video stream from the camera attached at the top of the workstation monitor, and the output is the list of eyes-eyebrows-nose coordinates. The frames are processed in sequence. To take advantage of movement history, a Kalman filter is used to estimate the motion vectors of the tracked feature points and future position of tracked feature points [43]. The state diagram of the system is presented that defines how the individual steps of the algorithm are combined to achieve real-time processing rates. Time-space diagrams representing the feature coordinates in time are analyzed. Finally, numerical data are presented that show that the system can run at real-time rates with a good accuracy.

4.

4.1 Tracking System State Diagram

The system state diagram is depicted in Figure 4.1 (a). The tracking starts in state NO_FD (no face detected). Once the face object has been located (state FD_FIRST) in the input video stream, feature points are tracked using the approach described in Chapter 3 (state FD_TRACK). First, an attempt is made to find tracked features in the location of the previous frame. If that does not succeed, a new face position is determined. Thus, time is saved by not unnecessarily running the face detection algorithm, which is the most time-consuming segment. Since the subject's motion is typically smooth, once the face region has been locked, the tracked features can be located in the same face bounding box for a number of video frames.

If tracked features cannot be found, a prediction based on the Kalman filter is used to fill in the gap (state FD_PREDICT). The prediction is done for a limited number of frames, namely at most three. If the tracked features are found while in the prediction state, a switch is made to the recovering state **FD_RECOVER** in which only measurements are taken from the environment and no prediction is made. The system stays in the recovering state as long as it was in the prediction state. After the recovery period, if tracked features have been found, a switch is made to the tracking state. If the tracked features have not been found while in the prediction or recovering state, the face finding algorithm is started again to find a new face location (state FD_NEWPOS), since the assumption is that the state of the environment has changed too much during the prediction period. If the face has not been found, a switch is made to the initial state, whereas if the face has been found, a switch is made to the intermediate



(a) Original State Diagram

(b) Improved State Diagram

Figure 4.1: Tracking System State Diagram

continuation state FD_CONT. In this state, an attempt is made to find the tracked features in the new location. If the features have been found, a switch is made to the recover state. If the features have not been found, a switch is made to the no-match state FD_NOMATCH. In this state, the face object is located in the image, however, no face features are found.

If the face starts to move, the diagram in Figure 4.1 (a) will not respond promptly to the movement, and the tracking parameters will not be adjusted well. Thus, an improved scheme, depicted in Figure 4.1 (b) is introduced. The only difference is that in the state FD_PREDICT, an attempt is made to find the face first, then to try to find the features in the new locations. In this way, if the face moves a bit, the new face boundaries will be found that will fit the data better, and the tracking parameters will be adjusted faster.

The motion of six variables is estimated: the X and Y coordinates of both eyes and nose. In the Kalman filter equations, using previous and current values of the variables, we manipulate 2×2 matrices, and matrix operations such as the inversion are not time consuming. The time update ("predict") equations for the projected state of tracked point $\widehat{\mathbf{x}}_{k+1}^{-}$ and the error covariance matrix $\mathbf{P}_{k+1}^{-}$ are:

$$\begin{aligned}\widehat{\mathbf{x}}_{k+1}^{-} &= \mathbf{A}_k \widehat{\mathbf{x}}_k = \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f^k \\ f^{k-1} \end{pmatrix} \\ \mathbf{P}_{k+1}^{-} &= \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{Q}_k\end{aligned}$$

where f^k is the estimate at time k, and f^{k-1} is the estimat at time $k-1$. In our case, $\widehat{x}_k$ is the X or Y coordinate of the tracked point.

Th

estima

$K_t =$

$\hat{x}_t =$

$P_t =$

where

matrix

Th

found

smoo

each m

is val

match

match

would

ries.

4.2

It sum

skin co

backgr

The measurement update equations for the Kalman gain $\mathbf{K}_k$, the update of the estimate with the measurement $\mathbf{z}_k$, and the update of error covariance matrix are [43]:

$$
\begin{aligned}
\mathbf{K}_k = &\ \mathbf{P}_k^-(\mathbf{P}_k^- + \mathbf{R}_k)^{-1} \\
\hat{\mathbf{x}}_k = &\ \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) \\
\mathbf{P}_k = &\ (\mathbf{I} - \mathbf{K}_k)\mathbf{P}_k^-
\end{aligned}
$$

where we set $\mathbf{H}_k = \mathbf{I}$, and empirically determine the measurement error covariance matrix $\mathbf{R}_k$ and the process noise $\mathbf{Q}_k$.

The predicted coordinates are used in two ways: (i) to verify the position of newly found features and (ii) if the tracked features are lost to predict their location and thus smooth out the tracking and avoid losing the tracked features. In the former case, each match is checked against the predicted location. The simple Euclidean distance is calculated for each of the coordinates, and if the overall distance is too large, that match is discarded. It is assumed that such a match is rather far away from the real match, e.g., the eyes and ears might be matched. In the later case, the predicted coordinates are used as the output of the program, instead of the program-calculated ones, as described earlier.

4.2 Using Motion Detection to Improve Tracking Results

In some cases when the user's clothes or background are of color similar to the user's skin color (e.g., reddish, pinkish, orange colors), the segmentation of the face vs. background is not very good. To overcome that problem, we use motion detection

to
s
ex
th
th
eff

th
as
30
in
of th
mo
mo
the p

O
that a
for th
achieve

By
When
that are
about th

to facilitate face-background segmentation. The area where there is some motion is likely to contain the user's face only, and the areas where there is no motion is assumed to be the background. We need to focus the search for the face only in the area where there is motion, and the rest of the viewing field is ignored. Thus, if the background includes the colors that are similar to the skin color, they would not affect the search for the face.

The motion detection algorithm used is rather simple: for each frame at time t, the difference between the frames t and $t-1$ is calculated. All pixels are flagged as changed, if the pixel value difference is larger than 10, or same otherwise. Every 30 frames, a check is made to determine which pixels have changed in the previous interval, and the motion area is calculated, by finding connected components out of the moving pixels. Components that are out of range of the previously detected motion object are discarded, to reduce the noise. At times of significant motion, the motion area is updated at the global level, and when there is just slight or no motion, the previously found global level motion is used.

Once we have determined the area that is moving, we restrict search for the face to that area. In this way, we can speed up the face detection algorithm, and compensate for the time spent in calculating the motion. Real-time processing rates are still achieved.

By detecting the motion area, we can also easily check whether a dark face is present using the algorithm described in Section 3.2. We need only check the windows that are centered around the motion area center. If we didn't have the information about the possible face area, we would need to check all possible sub-windows for

Euclidean distance from hand-labeled	No motion detection no KF	 use KF	Motion & dark face no KF	 use KF
Eye 1	2.84	3.31	1.75	2.29
Eye 2	8.77	8.83	4.46	4.25
Nose	5.11	5.47	8.17	8.88
# Matches	102	117	130	116
# Outliers	6	27	11	50
# Misses	227	191	194	160
# No Matches	25	25	25	25
# Detected No Matches	24	25	25	19

Table 4.1: Accuracy of the motion detection and dark face detection given as the Euclidean distance between feature points and hand-labeled locations. Euclidean distance is given in pixel units, and the number of matches is in frame units.

dark areas, or to restrict the face location to some specific image area (e.g., center of the image). The latter approach would significantly constrain the users.

The above approach was tested on a 359-frame movie of a subject with dark-skin. Table 4.1 compares the accuracy of the detection when motion detection is and is not used, and when the Kalman filter is and is not used. The Euclidean distance between program-detected and hand-labeled feature points in each image frame is calculated in pixel units. The table also includes the number of video frames processed: the number of correct eyes-nose matches, number of outliers, number of frames where no eyes-nose match was found, number of no-matches in the video stream, and number of detected no-matches in the video stream. As can be seen in the table, the number of outliers increased, while the number of missed matches decreased when the motion and dark-face detection were turned on. Also, the accuracy of finding both eyes increased. However, the nose location detection error increased. This is due to the altering of the input image, and in that process some information about the precise m

m

Th

of

th

fo

4.3

Fig

com

of th

Kal

is ap

used

Th

is an

experi

Table 4

dark-sh

Typically

away wi

nose location was lost. The use of the Kalman filter resulted in fewer no-matches and more outlier matches, in both the case of motion detection and no motion detection. The increased number of outlier matches is due to the fact that the use of the Kalman filter prediction might reject a correct match if it is locked onto an incorrect match.

The most important issue is that the face area was detected more accurately in the case when motion detection and image alteration was used, which leaves room for improvement of the feature finding algorithm.

4.3 Results Using Kalman Filter

Figure 4.2 shows results of the feature tracking on a sample movie file: the X and Y coordinates of eyes and nose points are plotted over time. We compare the results of the tracking without smoothing and with smoothing and prediction based on the Kalman filter. As can be seen, the number of peaks in the case when the smoothing is applied is significantly reduced compared to the case when the smoothing is not used.

The **accuracy of the feature point detection** is measured as the Euclidean distance in pixels between program-detected and hand-labeled coordinates. In our experiments, the camera had square pixels so the X and Y scales are the same. Table 4.2 (a) shows the accuracy measured on three movies (two light-skin and one dark-skin subjects, with the total length of 835 frames). The program detection was typically 1–3 pixels away from the real locations. The location of eyes was 1–3 pixels away, while the location of nose was 2–3 pixels away from the hand-labeled location.

Fig
(b

The

Tabl

of o

are s

lower

in th

in Se

Th

achiev

camera

subjec

64 × 64

we can a

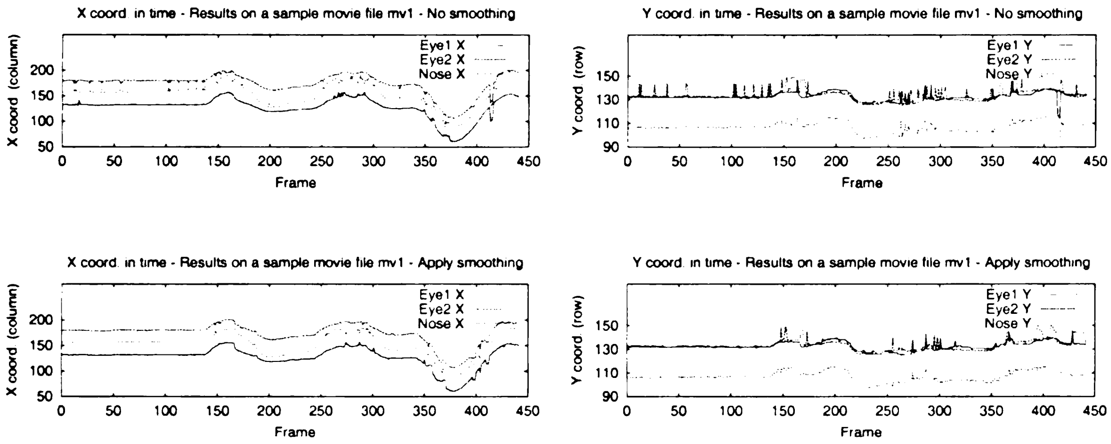


Figure 4.2: Comparison of the X and Y coordinate tracking without (top) and with (bottom) the use of Kalman filter for smoothing and prediction.

These results were obtained after the outlier matches were removed from the statistics. Table 4.2 (b) shows for each subject the number of correct eyes-nose matches, number of outliers, and number of frames where no eyes-nose match was found. The errors are similar in both cases. However, the number of outlier and missed matches are lower if smoothing is applied. This ensures smooth changes of the feature coordinates in time, which is important for gaze direction and menu selection, as will be discussed in Section 5.2 and Chapter 7.

The **execution time** of the program is such that real-time tracking rates are achieved. An image size of 320 × 240 allows the user to move freely in front of the camera and to sit far from the camera (and display). In the current set-up, if the subject sits about 50*cm* away from the camera/display, the subject's face is about 64 × 64 pixels, which allows reasonable matching results. For the above image size, we can achieve a frame rate of 10–30 Hz on a SGI Indy 2 workstation.

	No smoothing			Apply smoothing		
	Eye 1	Eye 2	Nose	Eye 1	Eye 2	Nose
S1	1.44	1.20	3.14	1.60	1.38	3.54
S2	1.30	3.34	3.21	1.18	3.11	3.22
S3	1.21	1.23	2.33	1.29	1.34	2.19

(a) Euclidean distance for the eyes and nose in pixels between hand-labeled eyes and nose locations, and computed locations.

	No smoothing			Apply smoothing		
	Matches	Outliers	Not Found	Matches	Outliers	Not Found
S1	240	3	1	242	0	2
S2	332	0	7	334	2	3
S3	252	0	0	251	0	1

(b) Number of video frames processed: number of correct eyes-nose matches, number of outliers, and number of frames where no eyes-nose match was found.

Table 4.2: Accuracy of the tracked feature location

Table 4.3 shows the execution times for six sample runs of two subjects. Three different motion patterns were tested: no significant motion, smooth motion and sudden movement (subject moving extremely fast). All measurements were done for 1000 frames of video on an SGI Indy 2 workstation with the input image size of 320 × 240 pixels. The first row shows the total number of matches, and the second row shows the number of frames spent in the tracking state. The execution time of the face detection algorithm (row FD_NEWPOS) is 80–90 msec, while the time needed to locate the tracked features in the face (row FD_TRACK or FD_CONT) is 10–30 msec. The overhead induced by grabbing and displaying the image is not significant, and the overall frame rate that can be achieved for smooth motion is 20–30 Hz. In the case of sudden movements, the subject's face position changes in every frame, and the face detection algorithm dominates the execution times, so that the frame rate

	No Motion		Smooth Motion		Sudden Moves	
	R1	R2	R3	R4	R5	R6
Match	999	1000	998	999	958	799
FD_TRACK	992	998	941	924	623	431
Average Execution Times in milliseconds						
FD_TRACK	14	12	29	21	54	96
FD_NEWPOS	94	125	89	90	90	86
FD_CONT	24	16	37	15	60	53
Analysis	15	12	37	29	111	175
Grab	12	15	5	8	11	10
Display	6	5	7	5	6	5
Total	34	33	49	42	129	191
Frame Rates in Hz						
Analysis	69	81	27	35	9	6
Overall	30	30	20	24	8	5

Table 4.3: Execution times for six runs (two subjects). The program was run for 1000 frames of video, on an SGI Indy 2 workstation with the input image size 320 × 240.

achieved is low. If we used smaller images, the frame rate would be higher; however, that would significantly constrain the subject's movements, if we are to track eyes and nose accurately.

Similar execution times were achieved on a 200 MHz Pentium PC with a Matrox video board, running Windows NT and using Vision SDK as the interface for the camera. In this case, the major bottleneck was the Vision SDK interface, while the processing itself took the same time as on the SGI workstation. Thus, the maximum frame rate achievable was 10 Hz.

The tracking program was tested on numerous occasions, including several open-house sessions. Both light-, brown- and dark-skin subjects were tracked with good results. In the case of dark-skin subjects, in some occasions extra lighting was used to increase the ambient lighting of the room. When measured formally using recorded movies, the subject's face and face features were successfully tracked in the video stream for 99% of the frames. We did, however, have some difficulty with the system performance in another room in another location during a demo session.

4.3.1 How far from the camera can the user be?

In majority of the experiments, the users were sitting at about 2 ft distance from the camera, which was located on top of the workstation monitor. In the early stages of the development, we used a low-quality generic SGI eye-camera for our experiments. In the later stages of the development, we used a high-quality pan-tilt-zoom camera. The tracking program worked well using both cameras. The users were not required to be at a fixed sitting distance from the camera during experiments. They could move closer or further, and the facial features were tracked without problems. Our facial features finding heuristics do not require any user-specific information. As long as the facial features are spaced enough for the described geometry to work, the features could be located.

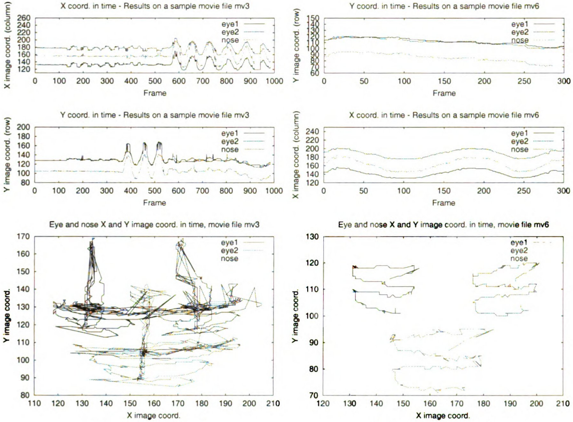
In an experiment using a pan-tilt-zoom camera, we measured that the smallest face area in which facial features could be determined was 50×70 pixels (using a 320×240 pixel image). In such a case, the distance between the eyes translated into 20 pixels, and eyes-nose distance was 10 pixels. Similar results were achieved using a generic SGI eye-camera. If the zoom feature of the camera is used, the maximum face distance from the camera depends on the zoom factor. In our experiment, at the

maximum zoom factor, the user was 7 yards (6.5 meters) away from the camera and the facial features were tracked by our algorithm. All the above experiments were conducted in a controlled environment: a room with fluorescent ceiling lighting and no windows. No additional light source has been used.

4.4 Analysis of Movement Data

What can the time-space statistics tell us about the subject's movement? Figure 4.3 shows the X and Y coordinates in time for two sample movie files. Figure 4.3(a) shows a sequence when the subject was first still, then moved the head up and down three times, then moved head left-right six times. The last graph shows the X and Y image coordinates, and we can clearly see the motion indicated by the plots of the coordinates in time. In Figure 4.3(b), we see a sequence when the subject was moving the head left and right, and down, which indicates spiral moves. The X vs. Y plot shows the spiral moves for each tracked feature point.

The plots from Figure 4.3 resemble the plots from Yarbus [97], and we can analyze them in a similar way. Additionally, the data could be used to monitor the user's movements, and certain movement could trigger certain actions, as will be discussed in Chapter 7.



(a) Movie file mv3

(b) Movie file mv6

Figure 4.3: Sample time-space statistics for two movie files: X and Y coordinates of feature points in time, and X vs. Y image coordinate

4.5 Summary

This Chapter presented an integration of the feature finding algorithm into a real-time tracking application. The system state diagram was defined, and the use of Kalman filtering was described. Kalman filter estimate is used in conjunction with the feature finding algorithm to eliminate erroneous matches. In addition, the filter estimate is used as the output of the system when the features could not be found by the algorithm, which resulted in smoother tracking. The use of a motion detection algorithm was presented. Motion detection was used to overcome the problems when the face-background segmentation and to facilitate dark face location. Accuracy results and execution time data were presented. Finally, plots of the tracked feature coordinates in time were presented and analyzed. An analogy to the eye-movement pattern analysis in Cognitive Science was made.

Chapter 5

Gaze Direction Detection

This Chapter presents a method for detecting the users gaze direction based on the coordinates of the eyes and nose in the image. A neural network used to map the user's facial feature location to screen coordinates is described, as well as a method used for its training. The mapping of the motion of features in the image to the motion of the screen cursor is described. The joint use of the above two mappings is discussed. Methods to smooth out the gaze direction are presented. Finally, results of the gaze tracking are presented in a form suitable for analysis of eye-movement patterns, e.g., in visual cognition studies, in attention measurement, or in the cursor control by the gaze.

5.1 Introduction and Terminology

Once the coordinates of three points on the face are known, we can determine the subject's gaze direction. Several methods for gaze direction determination are discussed in Section 2.4, that range from the exact mathematical solutions to an approximation using Artificial Neural Networks. In this work, the main goal is to avoid the calibration of the system, and to avoid the explicit knowledge of the user's dimensions or camera characteristics. This would enable high transparency and portability of the system.

Currently, the program tracks only the user's head and eyes and nose position within the face, unlike systems that track the user's eye movements. The head pose is determined by the position of the eyes and nose points in 2D, and the user must turn the whole head for a gaze point to change in our system. In the eye-only-tracking systems, the user's head must be stationary, and only the eyes would move.

Thus, in our work we are not able to track the saccadic eye movements. The experiment setting is such that we do not have enough information about the eye area to determine the saccadic movements: they would translate into just 1–2 pixels change of the eye pupil position in the image plane. Figure 4.3(a) illustrates this point: in the first 100 frames of the movie (about 6 seconds) the user's head was still. Due to the nature of human eye movements, the user's eyes *must have moved* in that time interval. However, the resulting changes in the eye coordinates are only one or two pixels. In the following 200 frames (about 12 seconds), the subject moved the eyes. These moves resulted in about ±5 pixels change in the eye pupil coordinate.

To determine saccadic eye movement we must use a high resolution camera that would zoom in on one eye, and the lighting conditions must be carefully controlled. Alternatively, an infra-red light source could be used to track the precise eye movements, as discussed in Section 2.4. In both cases, the user's head must be stationary (or move just slightly), since the tracking systems are sensitive to the head motion.

In the later text of this document, we will use the term *"gaze direction"* in the following context:

△ *Gaze direction:*

The point in the display plane determined through a transformation of the 2D image plane coordinates of the eyes and tip of the nose.

The transformation we use does not necessarily result in the point that coincides with the point that the user foveates. However, in the case when the user is in front of the display and explores it, the gaze point determined by our algorithm is a good approximation of the point which the user foveates.

5.2 Gaze Direction Determination Using ANN

One intuitive solution to the gaze determination problem is to determine the left-right motion by measuring the distance between the eyes and nose in the horizontal direction. This is easy to achieve and provides accurate results. However, applying a similar logic to up-down motion does not produce good results, since the eyes-nose relations are similar for a face looking up and down.

Another option is to fit the features' image position data for various gaze directions into a curve or to classify them in some way. However, using pure image coordinates would require a vast amount of training data that would cover all gaze directions from all image positions.

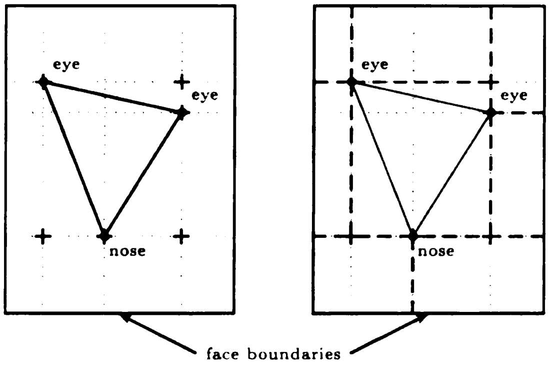


Figure 5.1: Relations between the eyes and nose used as the input to the neural network to determine the gaze direction. Dotted lines show the relations we use as the ANN input, dashed lines show the relations that had high weight in the resulting ANN.

Finally, after an analysis of the eyes and nose position within the face frame, it was concluded that some relations matter more than others, and that the gaze direction could be determined from the relative positions of the eyes and nose within a face. The solution to the function approximation was to use an Artificial Neural Network. The input to the network are the eyes and nose coordinates and their relations, and the output is normalized screen coordinates of where the subject is looking. Figure 5.1 depicts the eyes-nose relations used as the input to the network. All inputs to the ANN are normalized to values from 0.0 to 1.0. The relations used (depicted in dashed lines) are:

- X and Y coordinates of eyes and nose (total of 6 inputs),
- distance between two eyes, and each eye and nose in X and Y coordinates (total of 6 inputs),
- distance of eyes and nose to X and Y face boundary (total of 12 inputs).

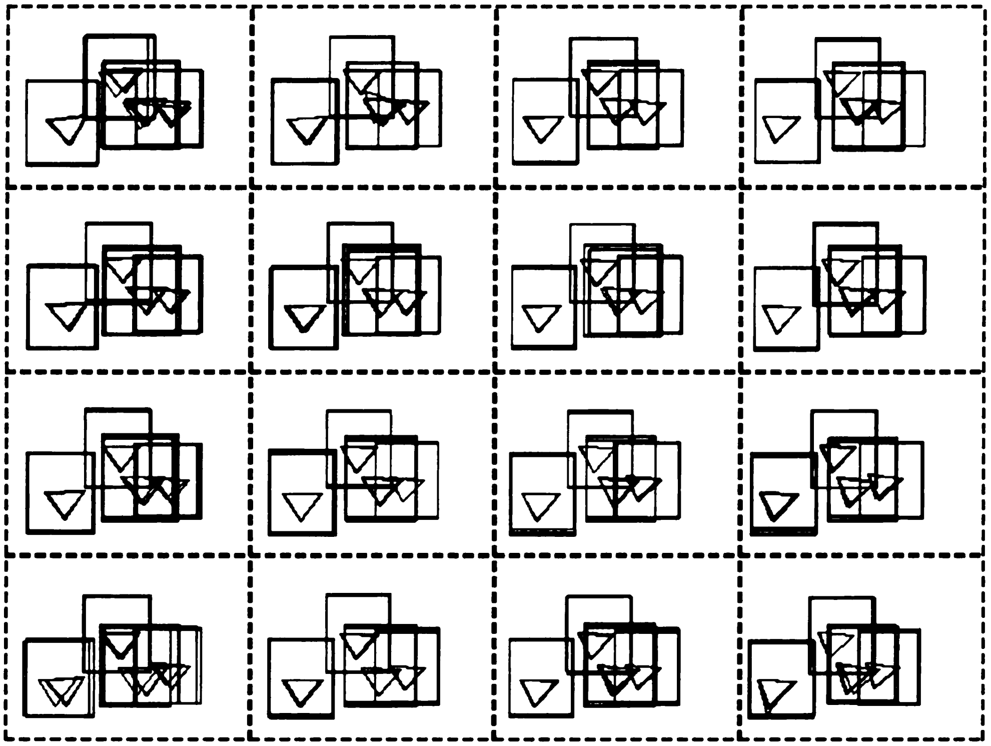


Figure 5.2: Sample input data for ANN trained to determine gaze direction.Dashed lines show the target grid points, rectangles represent face boundaries and triangles represent the corresponding eyes and nose locations.

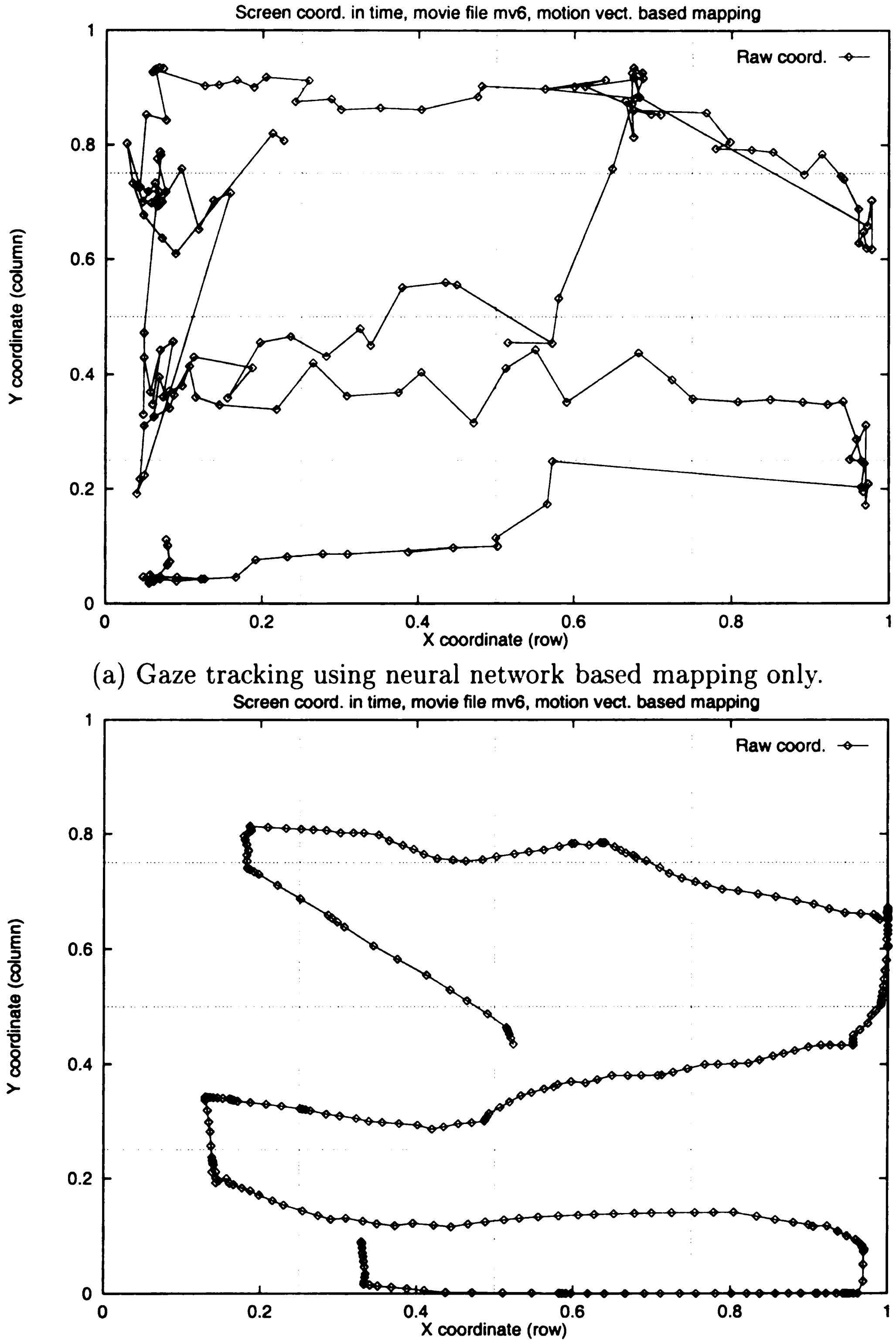
The ANN has a total of 24 inputs, one hidden layer, and 2 outputs (normalized X and Y screen coordinates). Different numbers of hidden units were tested, and the best results were achieved for 6 hidden units.

Training samples were obtained by having users look at buttons in a 4 × 4 grid on the workstation monitor while their pictures were captured. For each captured image the coordinates of the face boundaries and found features were recorded along with the screen coordinate of where the subject was looking. Figure 5.2 shows the sample input triangles and face boundaries for the 4 × 4 grid. Then, the neural network was trained using the QuickProp program [27].

The results using only the neural network-based mapping were not satisfactory. The main reason was that no movement history was used, and thus, for small changes in the input, the mapped screen coordinate changes could be quite significant. Figure 5.3(a) shows the screen coordinates in time for a sample movie file. The subject was instructed to look from the upper left to the lower right corner in spiral moves. For each frame, the ANN was run to output the screen coordinates. As can be seen, the movements of the imaginary cursor are not smooth at all. The reason for this problem is that the ANN's output could change significantly for small changes in the input data.

5.3 Gaze Direction Determination Using Movement Vector Scaling

To improve the performance of the gaze tracking, a combined approach that takes advantage of smooth subject motion is used. After an initial estimate of the screen coordinates using the neural network, we scale the subject's movements in the image into the display cursor movements. We can achieve smooth movements of the cursor as long as the subject's head is moving smoothly. The logic is similar to that used for an ordinary mouse, where the display/control gain factor is used. In the long run, this approach enables us to adapt to individual users needs, where each user would decide how much he or she wants to move the head in order to move the cursor. Also, an adaptable gain factor could be used so that for faster motion a greater gain value would be used, while for slower motion a smaller gain value would be used.



(a) Gaze tracking using neural network based mapping only.

(b) Gaze tracking using scaling of movements in picture to display movements.

Figure 5.3: Comparison of gaze tracking results using the neural network only and using the scaling of movements in the picture to display movements.

This could enable fast coarse positioning and slower and smaller motion for finer positioning. These issues will be further discussed in Chapter 6.

In the experiments described below, the display/control gain factor was determined based on the screen resolution and the head movement range in the X and Y coordinates. For the screen resolution of 946×910 pixels, and the maximum expected head movement of 50 pixels in the X and 30 pixels in the Y direction, the gain factor was 18.88 in the X and 30.33 in the Y direction. The gain factor was constant during the experiment.

To calculate the gaze point on the screen, the following rules are used:

- if the current frame is the first frame with the face detected, the ANN is used to estimate gaze point;

- if the gaze was estimated in the previous frame, the new gaze point is calculated as the previous gaze point plus the velocity vector calculated in the Kalman filter equations;

- if the gaze point coordinates are out of range (e.g., in the case when the display/control gain is not well adjusted to the subject's movements), the out-of-range coordinate is set to the maximum or minimum coordinate;

- if the tracked features are temporarily lost, the gaze point is kept at its previous location.

The above rules ensure that the gaze point is always in the display units range. A problem might arise if the gain factor is not appropriate, and that problem could be solved by individual calibration or on-line adjustment.

Results of the gaze tracking for a 4 × 4 grid are depicted in Figure 5.3. The subject was instructed to look in spiral moves from the upper left corner to the lower left corner and to traverse all grid points. As can be seen in Figure 4.3(b), the X and Y coordinates of the tracked points are smooth in time. The graphs in Figure 5.3(a,b) show the X and Y display coordinates normalized to values between 0.0 and 1.0. The raw coordinates obtained by the mapping were normalized to the grid coordinates. Only the grid coordinates were shown to the subject. Using the neural network-based mapping gives very unstable screen coordinates, as shown in Figure 5.3(a). On the other hand, if the movement scaling is used, smooth movements of the screen coordinates are achieved, as shown in Figure 5.3(b).

5.4 Smoothing of Gaze Point

The approach described in the previous Section will work well when the tracking is highly accurate, and when the user moves smoothly. However, in many practical cases these conditions will not be met. As a result, the detected gaze point will not be very stable and will look more like the output in Figure 5.3 (a).

To overcome this problem, we use two levels of smoothing of the gaze point produced by movement scaling. First, we estimate gaze point using the Kalman Filter. The input data to the filter are gaze coordinates produced by the movement scaling.

Then, a weighted averaging of estimated gaze point is done by using 90% of the previous value and 10% of the new value. In this way, we ensure that all the changes in the gaze point will be smooth.

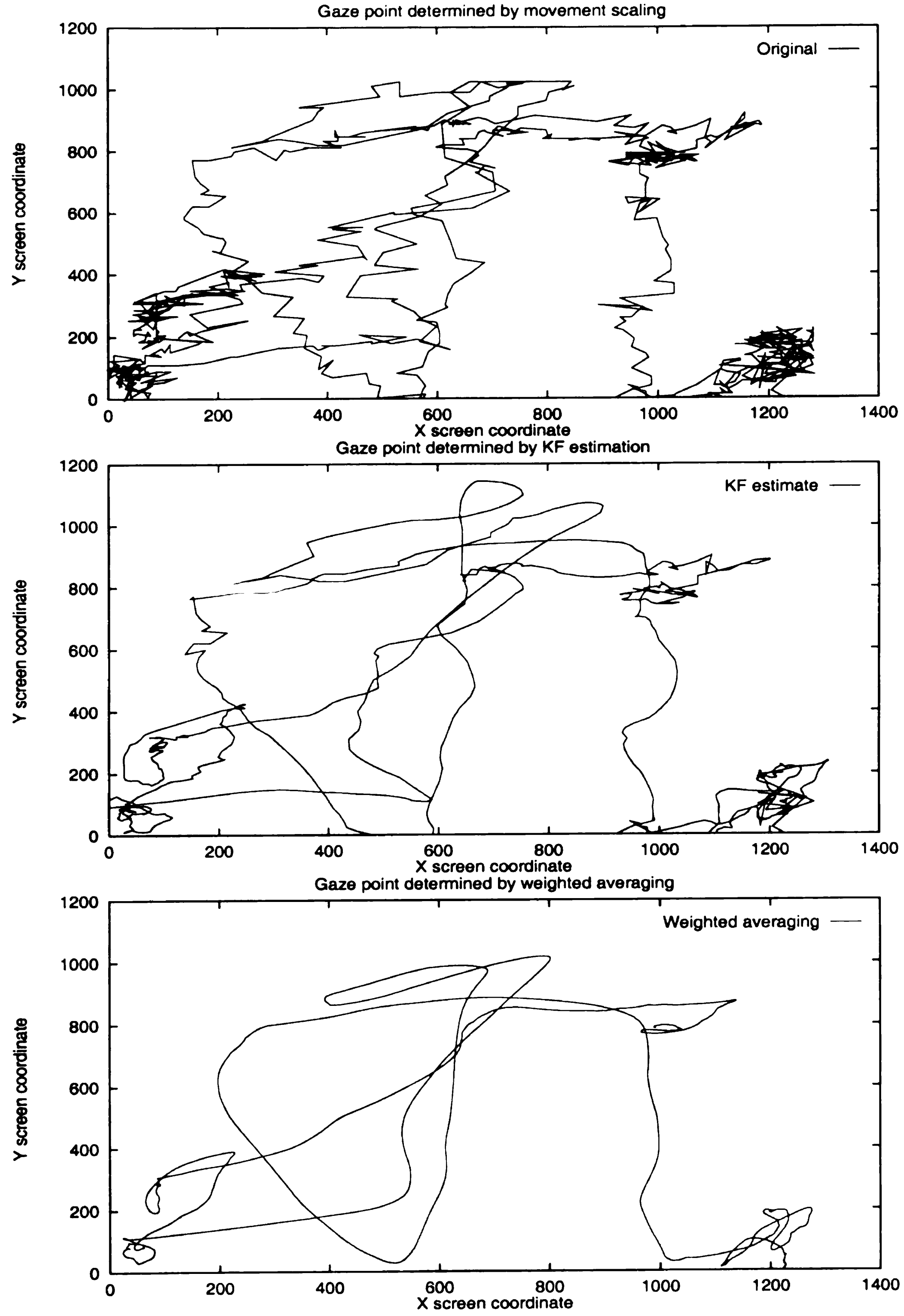
Comparisons of the three different ways to determine gaze point are shown in Figure 5.4. Figure (a) compares the movement scaling and the use of the estimate from the Kalman filter. Clearly, the filtered output is much smoother. Figure (b) compares the estimate from the Kalman filter and weighted averaging. In this case the differences are a bit more subtle, and they are visible only when the user is still, and staring at one point. In that case, small vibrations of the head will be compensated by the weighted averaging. The results and applications of this method will be discussed in more detail in Chapter 7.

5.5 Results

The algorithms described in the previous Sections can be applied in many ways. In this Section, it is described how the user's gaze can be monitored and used for applications such as visual perception studies, and how the user can control the cursor by moving his/her head.

5.5.1 Fixations Determination

As discussed in Section 1.2.2, determining the precise gaze point is of great interest in visual perception studies. However, measurement equipment is intrusive and users are not observing the displays in a natural setting. In our work, we applied our



(a) movement scaling

(b) KF estimation

(c) weighted averaging

Figure 5.4: Comparison of gaze point determined by movement scaling, Kalman filter estimation, and weighted averaging

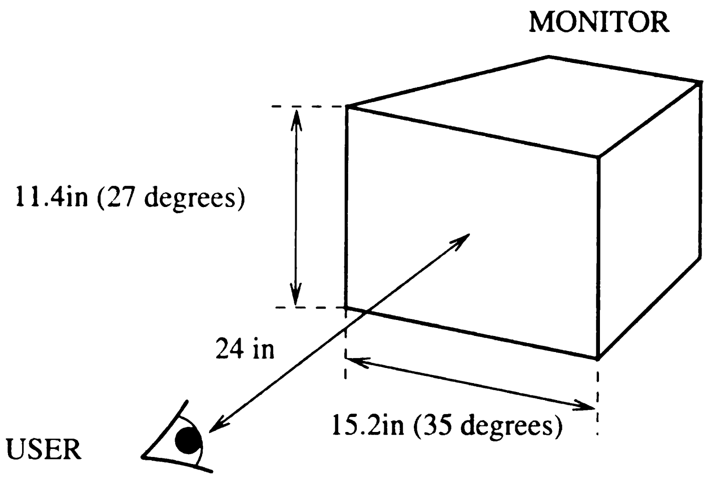


Figure 5.5: Fixations measurement experiment setting and viewing angles

gaze direction measurement method to measuring the fixation points while the users observed an image. The setting of the experiment consisted of a workstation monitor (19" diagonal) and users comfortably sat in a chair at about an arm's length from the monitor (about 2 ft away) (see Figure 5.5). The image was displayed at 1200 × 800 pixels resolution and covered the whole screen. The span in the horizontal direction was about 40 degrees of viewing angle and in the vertical direction it was about 26 degrees. These angles are much more than 15 degrees, which is the limit that can be viewed by just eye movement, so that the users *had to* move their head a bit in order to view the whole displayed image. Thus, by measuring the gaze as proposed above, we could reconstruct the user's viewing patterns.

The output of the gaze direction detection algorithm is a sequence of the screen coordinates with a timestamp. The measurements are not output in constant intervals, and the time interval depends on the image analysis time in each frame, and ranges from 20 msec to 800 msec (when a log file is written). In addition, since the user is not able to keep the head perfectly still, the gaze point is not perfectly still when the user's head appears to be still, but might oscillate slightly. To be able to calculate the fixation points, we apply the algorithm in Figure 5.6, which has another level of smoothing in it. Simply, results for each frame are accumulated for at least $100msec$. The averaged gaze location is computed and Euclidean distance is calculated from the gaze location from the previous time interval. The computed distance determines whether the subject's head is moving or whether the head is still. Fixation is defined as the period of time while the subject does not move significantly.

In addition, the minimum and maximum value of the screen coordinates are found and all the values are scaled to be in the range 0.0 to 1.0. This is needed since no calibration is performed and the user might move the head for just a fraction of the full range, so it is important to re-scale the gaze point. The results of the gaze path and fixations for a sample image are shown in Figure 5.7. The subject was shown the image for 30 seconds and was told to examine the image and remember as many details as possible. A total of 23 fixations were recorded, with durations ranging from 123 to 1690 milliseconds.

These results are not equivalent to the results of the Purkinje eye-tracker, where on the same image the number of saccades was about 40 and subjects viewed the image for 15 seconds. In our setting, we are not recording the actual saccades, but rather the head movements during the exploration of an image. In one fixation, there is probably a number of saccadic eye movement that might be recorded with a camera that would zoom in on the subject's eye.

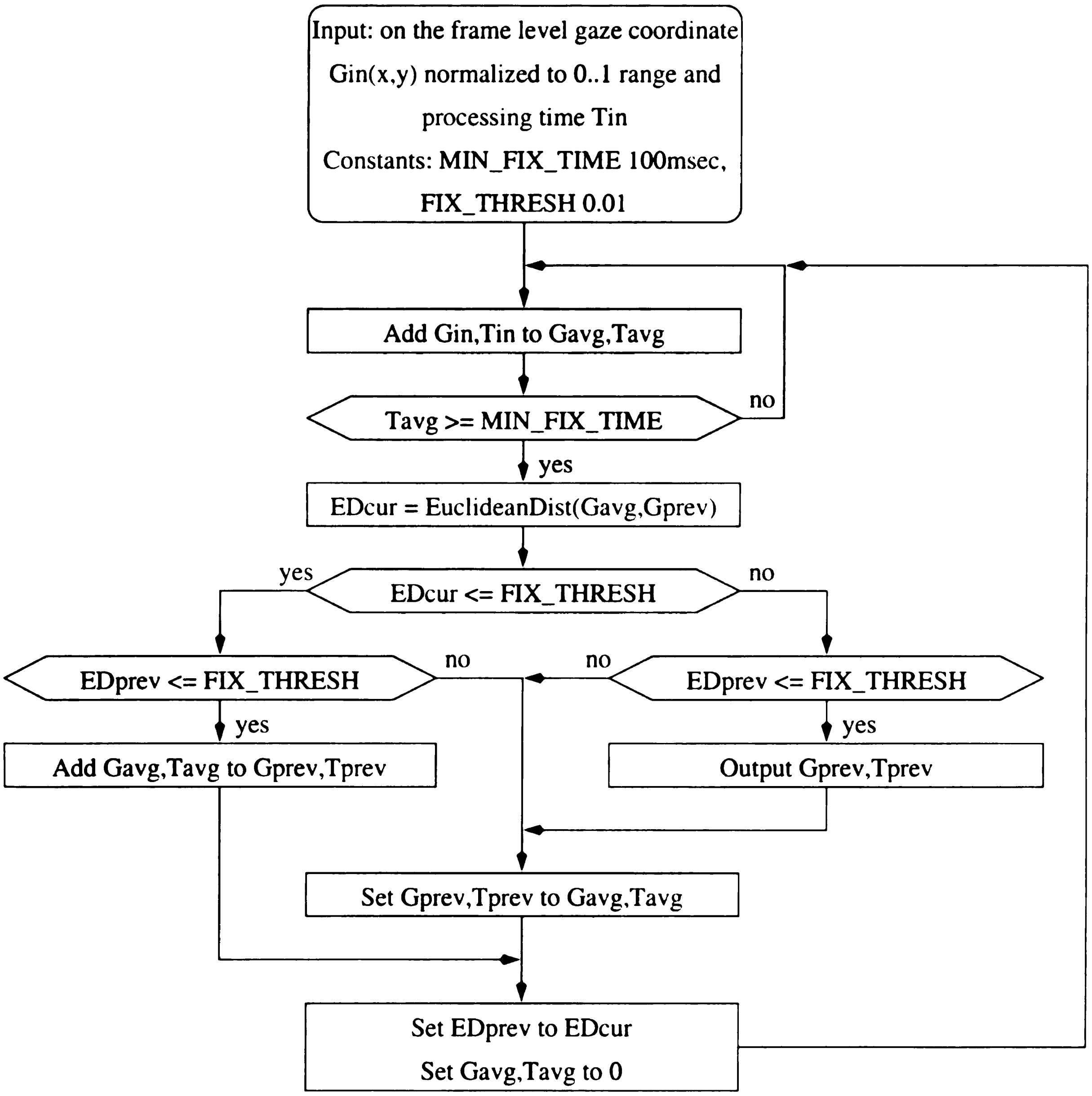


Figure 5.6: Fixation determination algorithm

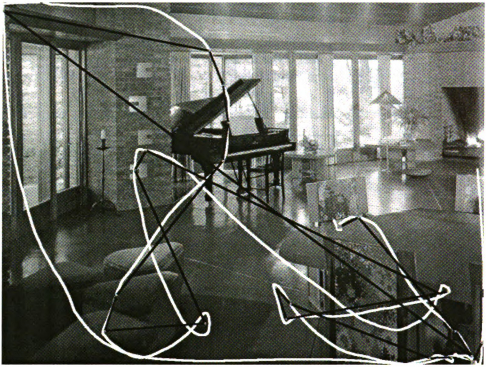


Photo from "50 Favorite Rooms by Frank Lloyd Wright" by Diane Maddex

Figure 5.7: Gaze path and fixations for a sample image. White line indicates the gaze path. Gray dots are fixation locations, the dot size indicates fixation duration. Black line connects consecutive fixation locations.

The complete results of the fixations determination experiment are presented in Appendix A. The experiment was conducted adjunct to the evaluation of the head-eye input device described in Chapter 7. A total of 18 subjects viewed three different images for 30 seconds each, and the gaze path and fixation locations were determined automatically using the algorithm in Figure 5.6.

5.5.2 Attention Measurement

Another aspect where automatic gaze determination can be used is measuring user's attention (for example to television, computer,...). Currently used methods rely on

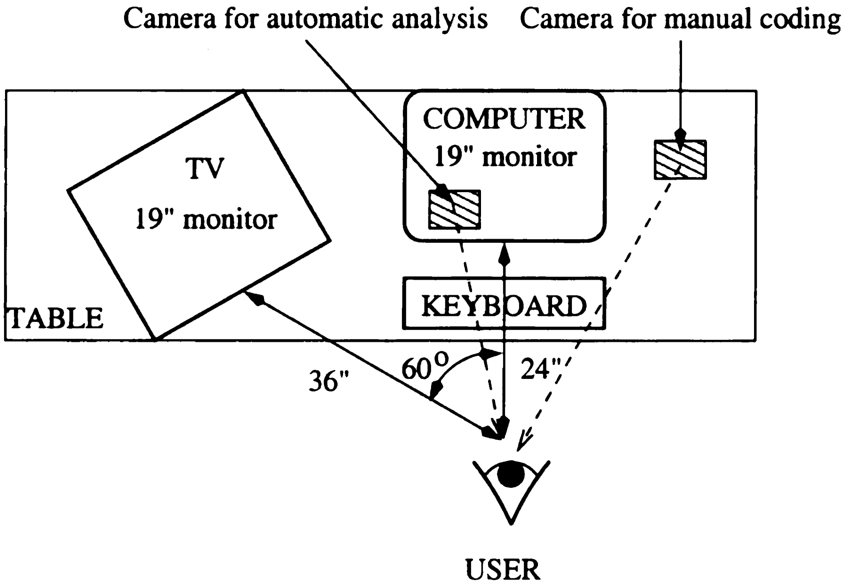


Figure 5.8: Attention measurement experiment setting and viewing angles

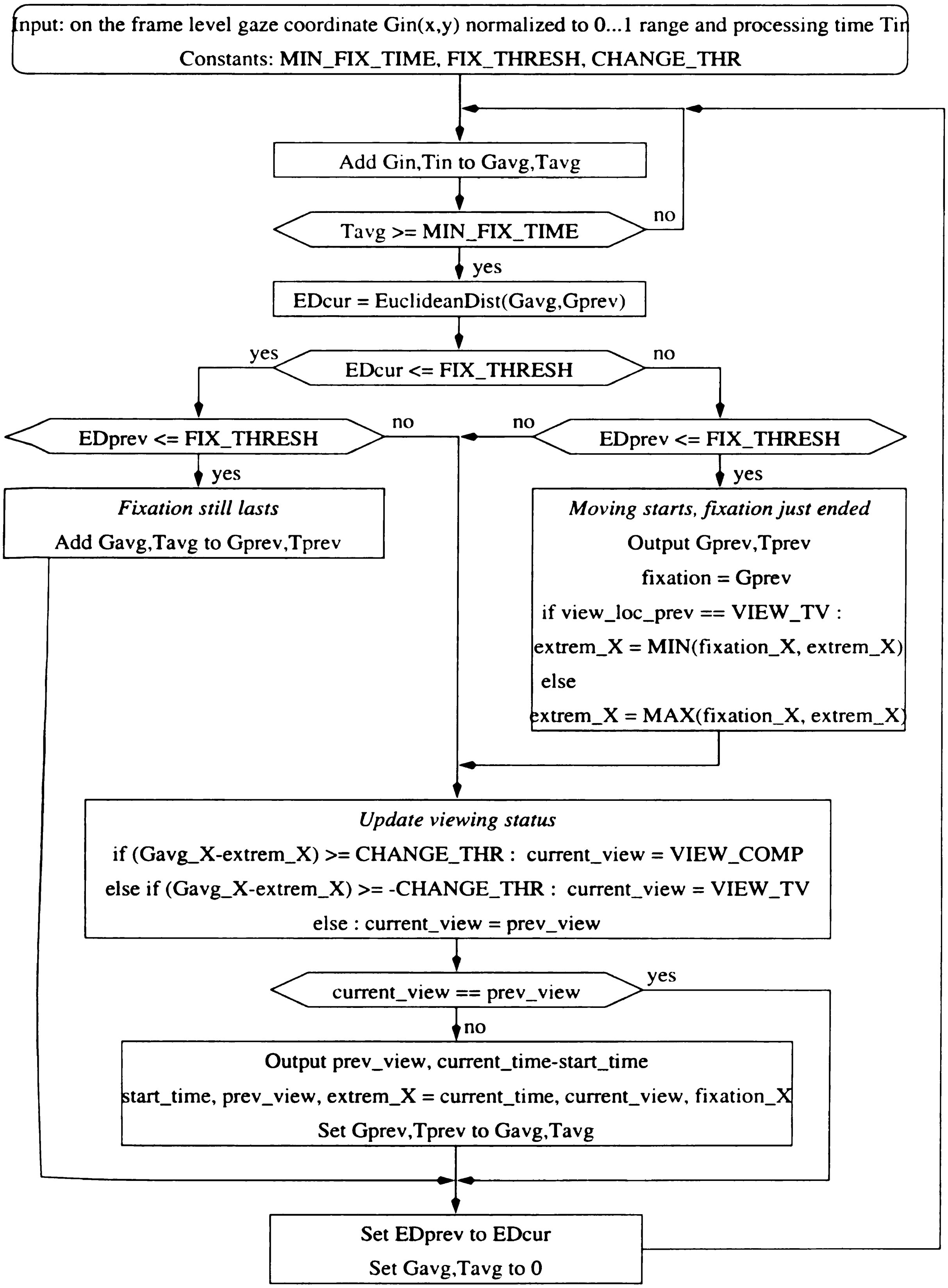
the manual coding of video tapes that record the subject (e.g., while watching TV). These methods are tedious for the coders, and even with a highly automated system, the human coder gets tired quickly and is prone to erroneous coding. The gaze determination system we developed could be used in such an application and would be completely automatic. The only human intervention is to set-up the recording at the beginning of the experiment and the measurement can be done for an indefinite time.

We conducted an experiment to verify the usability of our system in such an automatic measurement. This was a pilot project that would give us an insight in how well the automatic measurement would work. The experiment was conducted with Jay Newell, PhD student in the Telecommunication Department at Michigan State University [56]. The manual coding has been done in the MIND lab at the

Telecommunication Department at MSU by Jerry Roll, an intern in the lab, and Jay Newell.

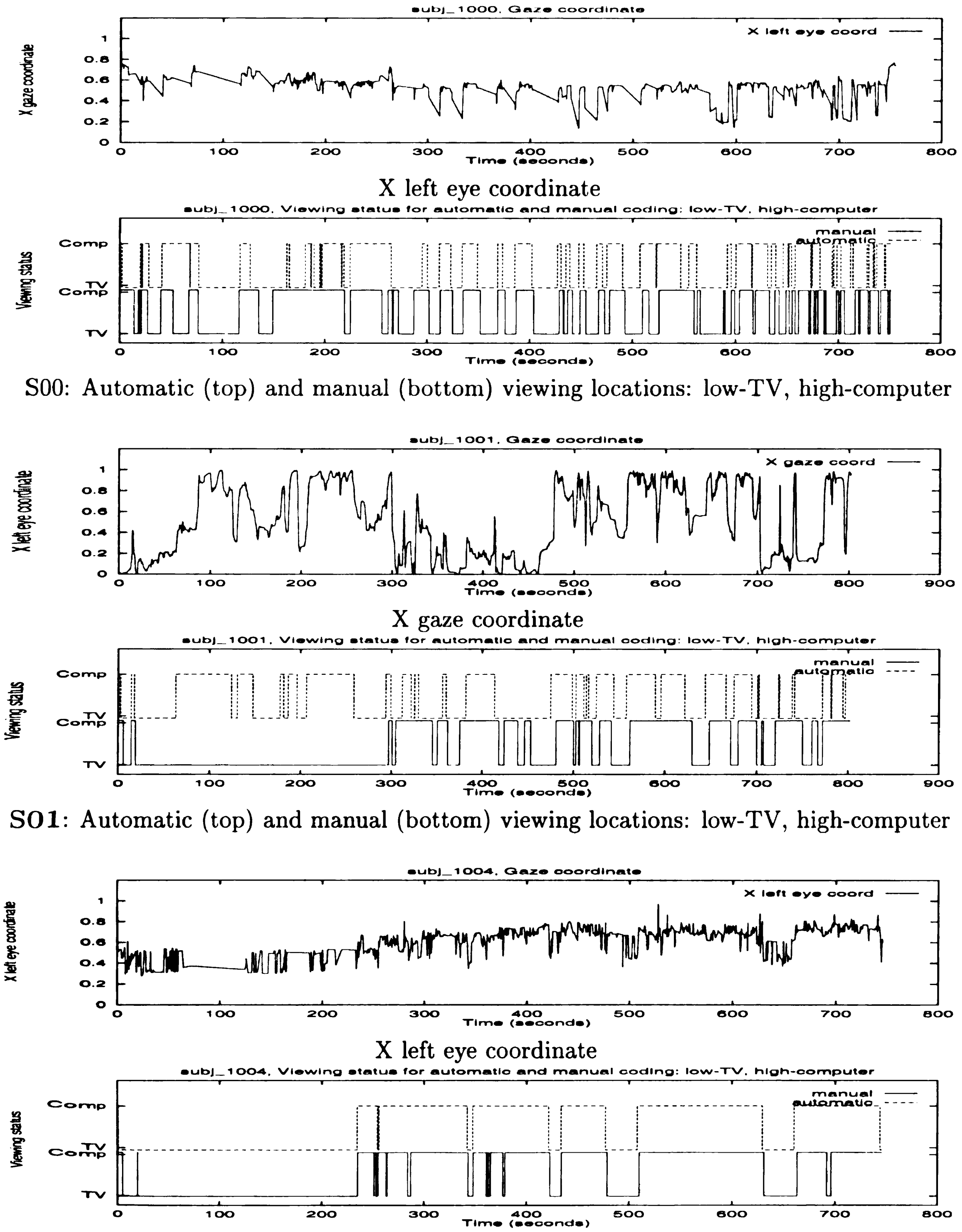
The subjects were asked to watch a video tape about 13 minutes long of a newscast and advertisements. Throughout the video a WWW address would be shown, either in the newscast part or as part of the advertisment. The subjects were instructed to enter the WWW address into the Netscape program whenever they would notice one on the TV. The Netscape program was already running on the computer. The first four minutes of the video tape contained mostly the newscast and in the nine remaining minutes contained both the advertisements and the newscast. The purpose of the task was to have the users look back and forth from the computer to the TV monitor. A total of 10 subjects participated in the experiment, two of them were the experimenters themselves. A black background was used to avoid any erroneous detection of faces in a pink carpeting that was in the room. Two subjects, who had glasses, had three blue squares (above eyebrows and on the tip of the nose), and the blue squares were tracked. That was necessary due to the current problems with tracking when a person wears glasses.

We assumed the set-up of the experiment as depicted in Figure 5.8. Subjects were seated about 24" away from a computer monitor, and about 36" away from a TV monitor. The viewing angle between the TV and the computer was 60^0. The algorithm used to analyze the data is similar to the one used for fixation determination (Figure 5.6) and is given in Figure 5.9. What is basically done is to check whether the subject moved left or right with respect to the last extrema point. The extrema point is calculated dynamically and in the horizontal direction only and depends on the



Input: on the frame level gaze coordinate Gin(x,y) normalized to 0...1 range and processing time Tin
Constants: MIN_FIX_TIME, FIX_THRESH, CHANGE_THR
Add Gin,Tin to Gavg,Tavg
Tavg >= MIN_FIX_TIME
no
yes
EDcur = EuclideanDist(Gavg,Gprev)
EDcur <= FIX_THRESH
yes
no
EDprev <= FIX_THRESH
no
yes
Fixation still lasts
Add Gavg,Tavg to Gprev,Tprev
EDprev <= FIX_THRESH
no
yes
Moving starts, fixation just ended
Output Gprev,Tprev
fixation = Gprev
if view_loc_prev == VIEW_TV :
extrem_X = MIN(fixation_X, extrem_X)
else
extrem_X = MAX(fixation_X, extrem_X)
Update viewing status
if (Gavg_X-extrem_X) >= CHANGE_THR : current_view = VIEW_COMP
else if (Gavg_X-extrem_X) >= -CHANGE_THR : current_view = VIEW_TV
else : current_view = prev_view
current_view == prev_view
yes
no
Output prev_view, current_time-start_time
start_time, prev_view, extrem_X = current_time, current_view, fixation_X
Set Gprev,Tprev to Gavg,Tavg
Set EDprev to EDcur
Set Gavg,Tavg to 0

Figure 5.9: Attention measurement calculation algorithm



X left eye coordinate

S00: Automatic (top) and manual (bottom) viewing locations: low-TV, high-computer

X gaze coordinate

S01: Automatic (top) and manual (bottom) viewing locations: low-TV, high-computer

X left eye coordinate

S04: Automatic (top) and manual (bottom) viewing locations: low-TV, high-computer

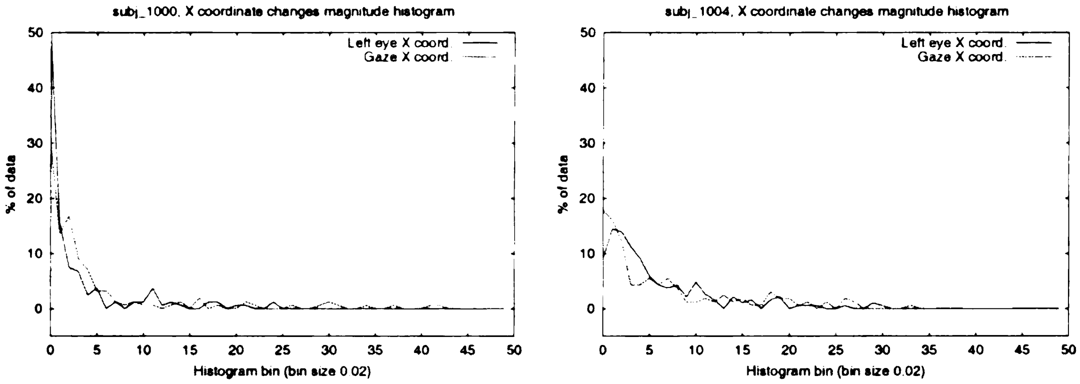
Figure 5.10: Gaze coordinate and viewing positions (TV vs. computer) over time for three subjects S00, S01, S04

current viewing location. For TV attention, it is the minimal point in that viewing interval, and for the computer attention, it is the maximal point in that viewing interval.

Figure 5.10 depicts the X gaze coordinate on the screen, left eye X image coordinate, and the viewing position determined automatically (upper line) and through manual coding (lower line) for the duration of the experiment. We can see that there is some agreement in the two codings and that in some occasions the two codings do not agree. The first two plots are for a subject who had three blue cosmetic dots that were tracked. That subject wears glasses and the tracking of the facial features could not be done very accurately with glasses on. Thus, blue dots were tracked. The later four plots are for the subjects whose facial features have been tracked.

The constant value in the algorithm is the threshold used to determine whether the subject moved enough to signal that the viewing location has changed. The threshold was determined empirically and for some subjects it worked well, while for some it didn't work well. In addition, we observed that for a number of subjects where there was almost no agreement in the viewing location, the input signal was the source of the problem. We then tested two input signals: gaze point on the screen, and raw image left eye coordinate. There would be a difference between the two signals, since the eye coordinate is used to determine gaze coordinate, and in the transformation process some information might be lost.

To determine the magnitude of the change of the signal (x coordinate over time only), the histograms of the absolute value of the changes magnitude were plotted for both input signals. All the data points were scaled to the 0.0–1.0 interval and



(a) Narrow histograms

(b) Wide histograms

Figure 5.11: Gaze and eye coordinate changes magnitudes histograms for two subjects

bin size was 0.02, with a total of 50 bins. The histograms should have Gaussian shape with a peek at zero. If the changes magnitudes vary, the Gaussian curve will be wider. Figure 5.11 depicts two sets of histograms for the gaze and eye coordinats. The common patterns in the histograms were that both histograms were narrow (Figure 5.11(a)), one histogram was narrow and the other was wide, or both were wide (Figure 5.11(b)). The narrow histogram generally gives a better estimate of the viewing change threshold. Thus, we select input signal type based on the histograms: the one that has 70% or more of the data points in the first five bins was selected over the one that had less than 70% in the first five bins. If both histograms were narrow, the one that had 95% of the data in a smaller numbered bin was selected. If both histograms were wide, the one that was narrower was selected.

Once the input type is selected, the viewing change threshold is selected based on the percentage of data points in the first five bins (N_5). The bin up to which $N_5 + (100 - N_5)/2$ percent of the data lies is selected as a threshold.

	Automatic selection					Manual best-threshold sel.				
				Kappa with					Kappa with	
			%	tolerance				%	tolerance	
Subj.	Input	thr.	agree	0s	±1s	Input	thr.	agree	0s	±1s
1000	EYE	0.14	57.58	0.18	0.31	EYE	0.08	68.26	0.37	0.58
1001	SCR	0.28	57.80	0.15	0.26	SCR	0.30	60.28	0.20	0.30
1002	EYE	0.14	59.18	0.21	0.51	EYE	0.14	59.18	0.21	0.51
1003	EYE	0.14	62.08	0.26	0.44	EYE	0.32	76.96	0.52	0.55
1004	EYE	0.14	66.18	0.30	0.58	EYE	0.28	95.57	0.91	0.94
1005	SCR	0.24	66.36	0.11	0.14	SCR	0.28	69.67	0.12	0.16
1006	SCR	0.20	63.19	0.26	0.31	SCR	0.34	63.32	0.27	0.30
1007	EYE	0.14	64.80	0.30	0.37	EYE	0.22	74.56	0.49	0.52
1008	SCR	0.32	68.06	0.28	0.40	SCR	0.12	70.57	0.32	0.55
1009	EYE	0.14	47.31	-0.08	0.15	EYE	0.08	53.55	0.03	0.33
Avg.			61.25	0.20	0.35			69.19	0.34	0.47

Table 5.1: Percentage of agreement in viewing positions and Kappa statistics for all subjects

We assumed that the manual coding is correct. We compared the percentage of agreement of the two codings, in every millisecond, and the corresponding Kappa [76, 17] statistic both with no tolerance and with a ±1 second tolerance. The Kappa statistic is commonly used to compare different coding schemes. Table 5.1 (left five columns) shows the input type and the threshold selected for each subject, the percentage of agreement and both Kappa statistics. For most subjects, the automatic coding was correct in at least 60% of the measurement interval. Kappa statistics with no tolerance were on average 0.2, and with the one second tolerance they were on average 0.35. For one subject, 1009, the automatic coding did not produce a satisfactory result. The right columns in Table 5.1 give the best manually selected results for each subject. The thresholds were selected for each subject independently by the author. For all but one subject, both the percentages of agreement and the Kappa scores are higher than in the case of the automatic selection. In the case of subject

	Human coder						Fully automatic						Manual threshold sel.					
	TV			Comp.			TV			Comp.			TV			Comp.		
	#	time		#	time		#	time		#	time		#	time		#	time	
Subj.	ob.	me.	std	ob.	me.	std	ob.	me.	std	ob.	me.	std	ob.	me.	std	ob.	me.	std
1000	42	8	2	43	10	2	22	14	5	22	9	2	40	9	2	40	7	1
1001	20	22	2	21	17	4	25	17	5	25	14	4	25	17	5	25	14	4
1002	42	10	1	42	8	2	76	4	1	76	6	1	76	4	1	76	6	1
1003	45	7	3	46	10	2	45	7	2	45	4	1	6	39	25	6	20	14
1004	15	22	3	16	26	5	62	4	1	63	7	2	6	52	40	7	16	10
1005	6	117	-	7	6	2	9	53	30	9	26	13	8	58	33	9	27	13
1006	43	9	1	44	8	2	11	23	9	11	15	7	8	31	23	8	12	5
1007	41	9	2	42	8	2	16	18	7	17	12	5	5	50	39	5	5	2
1008	28	17	2	29	9	3	22	21	8	22	8	5	34	15	5	35	5	1
1009	45	10	2	46	8	2	44	9	2	44	4	1	62	7	1	62	3	1

Table 5.2: Number of observations, their mean and standard deviation in seconds for manual coding, and automatic coding

1004, the agreement with the human coder was almost perfect (the last two plots in Figure 5.10).

Table 5.2 shows the number of observations of changes in the viewing locations and mean durations for the human coder, fully automatic selection and automatic selection with manual threshold setting. Both the number of observations and the mean duration of the attention interval in the case of automatic coding differs significantly in many cases compared to the manual coding results. This means that the automatic coding either generates too many or too few viewing location changes and that it misses some quick and short glances to either TV or computer monitor.

The above results show that the method could be adapted and improved. Firstly, the tracking of facial features must be very accurate. In our case, for some subjects the tracking failed to detect the correct features and detected erroneous features. As a consequence, the automatic detection would detect more changes in the viewing location. In the case of subjects 1000 and 1005, the tracking worked reasonably well, since blue dots were tracked instead of the facial features. For subject 1000, the achieved percentage of agreement and Kappa scores were high, and the number of observations and their mean duration were comparable. Subject 1005 was mainly looking at the TV monitor and made just a few glances to the computer. Thus, in his case, it was not easy to determine the appropriate thresholds through the histograms. The automatic coding only error was that it detected that the subject looked at the computer toward the end of the recording and never switched towards the TV, which resulted in an error.

Secondly, the position of the camera should be carefully selected. One solution would be to position the camera so that the subjects have to turn their head completely to view the TV, and in that case the face would not be visible at all. An easy attention determination would then be to simply check whether the face is visible or not. The problem in that set-up is that, in some cases, people make just a glance to a TV and the face is still visible. In addition, such a situation does not resemble the real-world set-up, where the people often have the computer and TV arranged as in our experiment, and glances quick and small in magnitude are frequent.

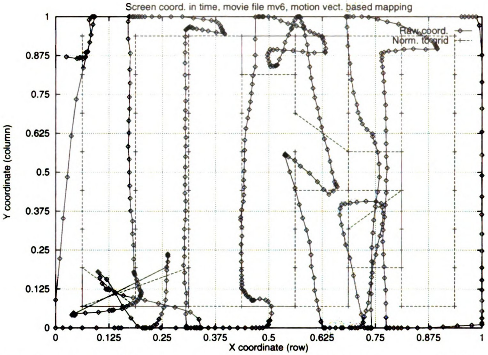
5.5.3 Cursor Movement

In the previous section, a passive observance system was described. The user has no feedback in terms of where the system thinks the user is looking. If the user is given that feedback and sees the actual gaze point on the screen, he or she can respond and interact with the system. The user can willingly move the head to produce cursor motion. As the system gives the feedback on the cursor position the user can learn to move in the right way so that the cursor can be moved along the desired path.

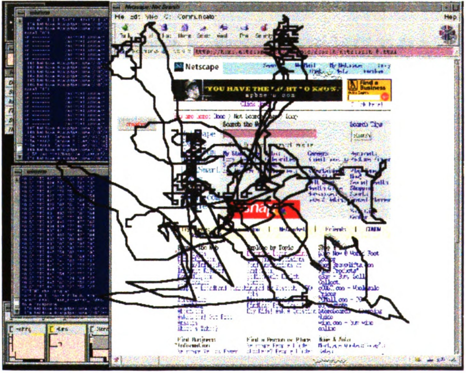
No calibration is needed at the beginning of the usage, rather, the system calibrates itself while it is being used. The Kalman filter estimation will pick up the motion trend as the user moves. If the tracking is temporarily lost, or if the user tilts too much, the re-calibration procedure is rather simple: the user should just look in the left-right, top-down, or vice versa, direction, and the filter parameters will get re-adjusted and the user will be able to control the cursor again.

In practical cases, novice users have some difficulty using the system in the first few minutes until they figure out what needs to be done. When they familiarize themselves with the system, they are able to move the cursor rather well. They are also able learn when to perform the re-calibration procedure and resume cursor control. That means that calibration is not the key issue in the system. The key issue is the concept of the system: the user learns how to move the head in order to control the cursor.

Figure 5.12 displays sample results of the cursor control. Figure 5.12 (a) shows a spiral path along an 8×8 grid: the subject was instructed to look from the upper left to the upper right corner in a spiral path and to attempt to traverse all the grid points. The solid line shows the cursor path and the dashed line shows selected buttons (equivalent to the solid line normalized to the grid points). The path was rather straight, and there were a few out-of-path moves that were corrected. Figure 5.12 (b) shows a cursor path during a net-surfing session. The snapshot of the screen shows the Netscape window and the black line shows the cursor path. The subject was selecting



(a) Interactive data and 8 × 8 grid, subject looked in spiral moves. Solid line shows the cursor path, dashed line shows the selected grid buttons.



(a) Net surfing using the head-eye control. Black line shows the cursor path. [Image is presented in color.]

Figure 5.12: Sample results of cursor control using gaze tracking

"Back", "Forward", "Stop" buttons and search engines just right of the "Snapshot" icon. Results of an evaluation of the above method will be presented in Chapter 7.

5.6 Summary

In this Chapter, a method for gaze direction detection was presented. The method is based on the use of an Artificial Neural Network to determine the initial gaze direction and the display vs. image coordinates gain factor. In this way, movements of the tracked feature coordinates are scaled into display cursor movements. A method for smoothing of the gaze path based on the Kalman filter estimation and weighted averaging was presented. The described algorithms were applied to a fixations measurement and results were presented. Results are presented of a pilot study that compared the automatic attention measurement system based on gaze tracking with manual coding. It was described how the gaze direction could be used to control the cursor motion and preliminary results for an 8×8 grid and free-form button arrangement were presented. It was also shown that the cursor can be moved along a predefined path using the head-eye input interface, e.g., spiral moves along all grid points, or to a specific point on the screen.

Chapter 6

HCI Based on the Head-Eye Input

When we interact with a computer, we need to watch the display. In most state-of-the-art computers pointing on the screen with a mouse is a must for most of the tasks. While we are moving the mouse with our hand, we need to observe the results on the screen with our eyes. We have to learn how to move the mouse and then we need to learn to coordinate the mouse movements with our eye movements. This hand-eye coordination, however, is a barrier for some people, and some people just cannot learn how to move the mouse well. If we were able to capture the gaze and enable the user to effortlessly point with head and eyes, we will be able to overcome the hand-eye coordination problem and will provide a more natural way to control a computer.

Another important issue is the level of intrusiveness of the gaze tracking equipment. Many systems are available that are very intrusive and do not allow natural motion, as discussed in Chapter 2. Other systems are not intrusive but require specialized and/or expensive hardware or require some make up that will be tracked.

The system described in Chapter 5 offers a relatively inexpensive and easy-to-use alternative to other eye-tracking systems. This Chapter describes how a human-computer interface for the head-eye input interface has been developed. Issues such as selection mechanisms, appropriate button sizes and feedback level are discussed and suggestions are made on what is applicable. The discussion is based on the observations made while various users used the proposed head-eye input interface for typical computer interaction tasks.

6.1 Selection Mechanism

With hardware pointing devices, selection is typically made by depressing some button or key that is attached to the device. In a handless interface, such a selection mechanism would not be appropriate, since it would involve the use of hands. Many eye-tracking systems opted to use a "dwell button" selection mechanism: a button would get selected if the user fixates it for some predefined time interval. If the duration of the interval is small, problems like "Midas Touch" would arise, as discussed in Section 2.6.7. If the duration of the interval is long, the user would easily get tired while using the system.

One natural solution is to use the face as a selection mechanism. Either facial expression or certain head movement could be used as a signal to select a button. Using predefined head movements is similar to the "on-screen selection button" concept discussed in [86] and 2.6.5. In [86], Ware and Mikaelian showed that an on-screen selection button had the worst performance. To make a selection, the user would need to fixate the desired target button for a predefined period and then to fixate the selection button for another period of time, and finally, the desired target button would be selected. In the eye-tracking system they tested, the head was still during the experiment.

6.1.1 Selection by Head Motion

If we are to apply such a logic to our proposed head-eye input, the user would need to double the amount of head motion for each selection, which might be uncomfortable. Some other head motion (e.g., move head up-down for yes, or left-right for no) can be an alternative selection mechanism. This kind of selection would be rather long, and very confusing, and could lead to the Midas Touch problem. What if the user wants to just look somewhere up or down, and not to make a selection? How would the system distinguish that from the real selection? One option is to have a large number of false alarms, which would lead to the user's dissatisfaction with the system. Another option is to have a custom-designed application that would incorporate such moves in the user interface.

We designed an experiment where the users would fixate one screen button, then they would get some feedback from the system on what the system thinks they fixated. Then, they would reply with a yes or no signal, which was nodding or shaking of the head. The user had no feedback on the program-calculated cursor position. Figure 6.1 illustrates the state diagram for yes/no signals: NO, SO, WE, EA stand for moving North, South, West, East, respectively. The motion is checked at every processed

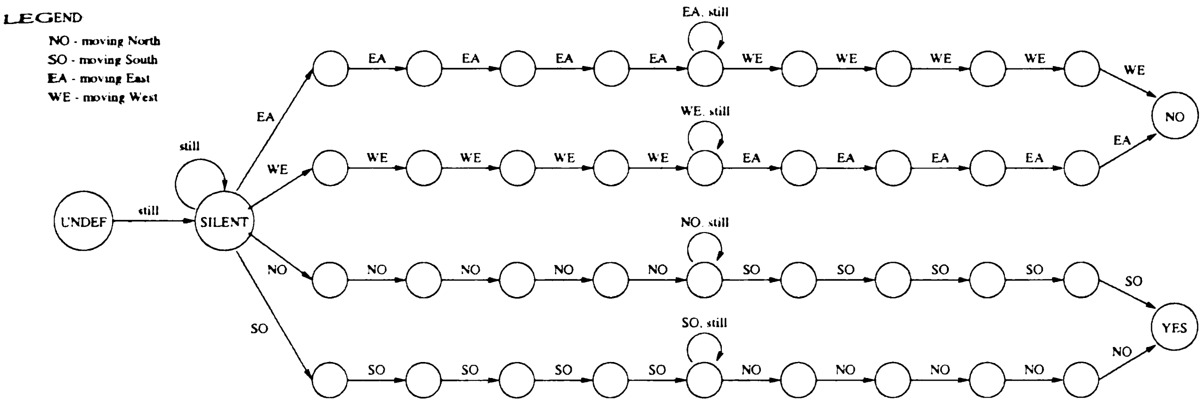


Figure 6.1: Selection by head motion—state diagram for "yes" and "no" motion detection

frame. As an underlying GUI for the experiment, we used a card game: the user would fixate one of 15 cards (arranged in 3 rows and 5 columns), and when the program detects that the user fixated a card the program would change the card that it thinks the user fixated. If the change was correct, the user would reply with "yes" motion, and if the change was incorrect, the user would reply with "no" motion. If the user doesn't notice the change in a card, the program would start flickering the changed card after a timeout period.

The order in which the user viewed the cards is depicted in Figure 6.2. The paths to the selected location (dashed line), and yes or no response paths (solid line) are depicted in Figure 6.3. In some cases the user had to move a lot until the system recognized the motion, and in some cases the response was not detected at all. Out of 15 selections, user's response has been correctly recognized 10 times, "no" response was not recognized in 3 cases, and in 2 cases the user didn't notice any change and didn't attempt any response.

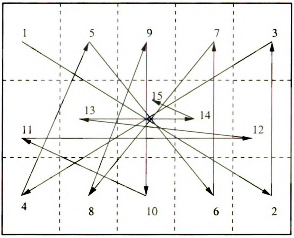


Figure 6.2: Predefined order of viewing the cards

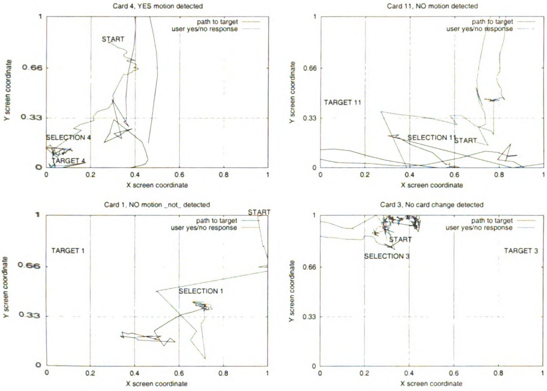


Figure 6.3: Paths to selected locations and YES/NO response paths

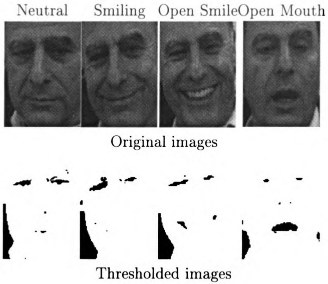
Preliminary results of this experiment showed that it would be possible to design an interface using motion response. However, this type of selection would not be suitable in a real-world application due to long selection times.

6.1.2 Selection by Facial Expression

As discussed in the Section 6.1.1, head motion would not be suitable for selection. The remaining option is making a facial expression as a selection mechanism. This should be easy to do for the user; after fixating a desired button, the user would make the required facial expression and thus make a selection. The expression to be used must be easy to do and should not involve unnatural movements.

In this work, an "open mouth" expression is used as the selection signal. The reason we use a mouth expression instead of, say, eye blinking, is that mouth movements are done voluntarily. Eye blinking is not always done consciously: humans blink frequently to moisten their eyes. Thus, that is not the best way to control the computer.

To determine whether the mouth is opened or not, we look for a big, dark ellipsoidal blob just below the nose. The best thresholded image from Figure 3.11 is used to find the mouth. Figure 6.4 shows examples of blobs for several face expressions: neutral, smiling (with mouth closed), open smiling (with mouth opened), and open mouth. For these four basic expressions, it can be clearly seen that the open mouth expression is distinct from the other three because of the dark blob.

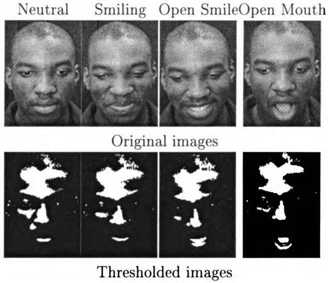


Original images

Thresholded images

(Pictures are part of the author's private database.)

Figure 6.4: Sample mouth expression



Original images

Thresholded images

(Pictures are part of the author's private database.)

Figure 6.5: Mouth expressions for a dark-skin person

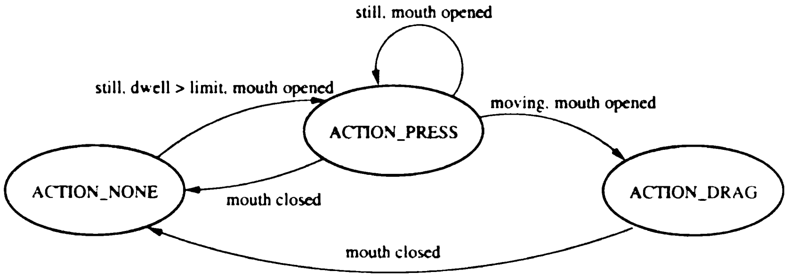


Figure 6.6: Mouth states transition diagram

The selection is done only when the subject is still for a predefined time. "Being still" is defined by the difference between the previous and current position of the gaze point being not more than 1% of the total screen width and height. Figure 6.6 depicts the transition diagram for mouth states. When a still state is discovered, it is checked for the state of the mouth. If the mouth is open, the transition to **MOUTH_ACTION_PRESS** is made. If the subject starts moving with the open mouth, we define that state as **MOUTH_ACTION_DRAG**. Finally, when the mouth is closed, we return to the initial state **MOUTH_ACTION_NONE**.

An alternative way to check for the open mouth state would be to learn the intensities of the mouth blob for each user. For example, in the case of dark-skin persons, the mouth blob was lighter than the skin area when they open the mouth in our experiment. Thus, in that particular case, we need to look for an above-threshold area. Figure 6.5 illustrates four facial expressions in the case of a dark-skin person.

6.1.3 Other Ways to Make a Selection

Sections 6.1.1 and 6.1.2 discussed two ways of making a selection that would involve the interaction using the head only. However, for some users that might not be possible or practical. Alternative selection mechanisms are hardware button, voice commands, or tongue-keyboard [5]. The users would then point the cursor with their head, then depress some hardware button (e.g., a key on the keyboard), or say a command (e.g., "yes", "no", "go",...).

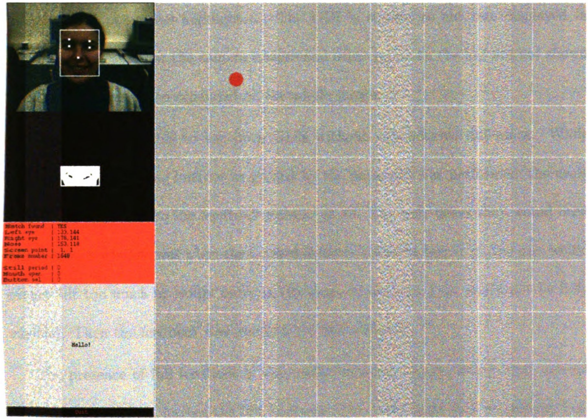
6.2 GUI Design Issues

Figure 6.7 shows the GUI of the button selection program. In the upper left corner is the subject's image as recorded by the camera. The tracked feature points are highlighted with the red crosses. Right below it we show a thresholded image of the mouth region that is used to recognize the mouth state. On the right is a grid where the program indicates gaze direction and selected button. A dark point indicates screen coordinates where the head-eye input interface is pointing, and when a selection is made, its color is changed to blue.

If the users are to use this kind of interface, what is the right way to display the information and what is the level of feedback that should be given to the user? When someone is controlling an application (e.g., Netscape), they do not need to know whether the program finds their face or not. However, in the learning phase or during the troubleshooting, they need to have the feedback.

6.2.1 Level of Feedback

In the setting of Figure 6.7, the user has full feedback on whether the tracked features are found and what the mouth state is as well as where the gaze point is. If there is



```
Match found  | YES
Left eye     | 133,144
Right eye    | 178,141
Nose         | 153,118
Screen point |  1, 1
Frame number | 1648

Still period | 0
Mouth open   | 0
Button sel   | 0
```

Hello!

[Image is presented in color.]

Figure 6.7: Snapshot of the face tracking demo GUI. Red dot shows where on the screen the user is looking

some error in tracking, e.g., the user tilts left or right and the tracked features are not found any more, the user can notice that on the display and initiate the re-calibration procedure (Section 5.5.3) or move in the more appropriate position. Also, the user can learn how to open the mouth so that the open mouth state is detected.

In our experiments we designed a GUI that had no feedback, except that the selected grid point was highlighted. The GUI is similar to the one displayed in Figure 6.7, except that the camera images and other items on the left are not shown and the grid points are expanded on the whole screen.

The users were able to use both GUIs without any notable difference. While the task (e.g., selecting buttons as guided by the program) was performed, the users didn't pay attention to the feedback window at all. The only times they would need the feedback window was when the tracked features would be lost (e.g., the user would either tilt too much or would move in the chair so that the face would not be fully visible). Then the feedback was essential for fast recovery.

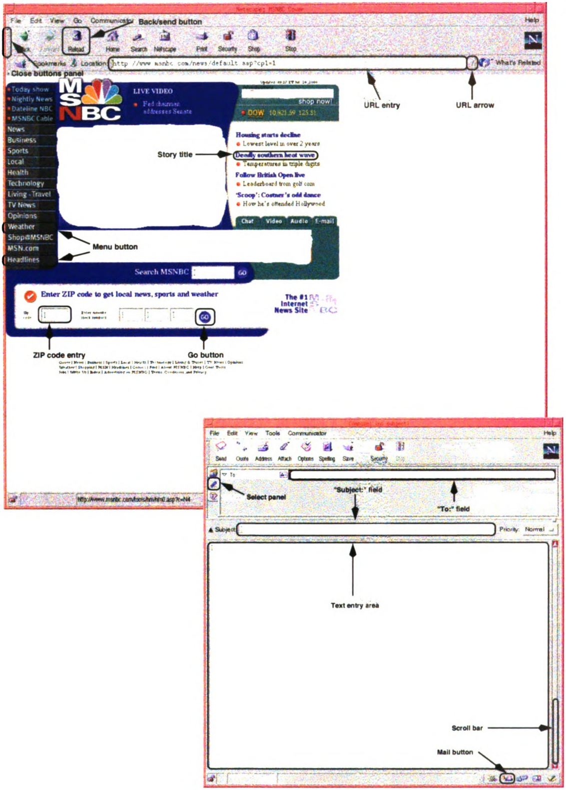
The presence of full feedback is also useful for novice users. While they are not used to the interface, having the full feedback helps them learn how to perform best (e.g., how to open the mouth, and how much to move).

6.2.2 Button Sizes

What is the appropriate button size that can be selected by the head-eye input interface? In terms of the updating accuracy of the cursor coordinates, we can update up to one pixel value. The users just need to learn to move very slowly to achieve that kind of update. We experimented with several different button arrangements: in an 8 × 8 grid, two 100 × 100 pixel icons, and a Netscape browser and mail windows. In all experiments, the setting was the same as shown in Figure 5.5. The display diameter was 19", or 15.2" × 11.4" with the resolution of 1266 × 1010 pixels.

In the case of the 8 × 8 grid, the grid button size was 2.075" × 1.425" inches (in horizontal vs. vertical direction), or $5^o \times 3.25^o$ of viewing angle. The size of a 100 × 100 pixels icon was 1.2" × 1.16", or $2.75^o \times 2.75^o$ of viewing angle. In the latter case, we could arrange 12 × 10 = 120 icons on the display, which gives a lot of room for various application needs. These large-sized selection areas would be suitable for custom-designed applications, e.g., a text editor or text display where a few buttons are needed and the remaining area would just display the text. However, most application programs have a large number of variable-sized buttons. Netscape is a typical example of such an application.

Figure 6.8 shows a typical Netscape browser and mail window. The buttons and text entry fields that have been used in the experiments are highlighted. A typical news WWW page is also included and the sizes of its menu items and story titles are highlighted. Figure 6.9 gives the Netscape button sizes relative to the display size. Table E.1 in Appendix E gives the sizes of the buttons in pixels, inches and degrees of viewing angle. As can be seen, the button sizes vary greatly. The navigation buttons in the Netscape window itself could be configured to be larger if needed. However, the buttons on a specific WWW site cannot be re-configured. Most of the state-of-the-art WWW sites often have a menu of options and each menu button is rather small in size. For a sample WWW site we used, the buttons are only $2.5^o \times 0.5^o$ of


File Edit View Go Communicator Help
Back/send button
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://www.msnbc.com/news/default.asp?cp1=1
What's Related
Close buttons panel
URL entry
URL arrow
Today show
Nightly News
Dateline NBC
MSNBC Cable
News
Business
Sports
Local
Health
Technology
Living - Travel
TV News
Opinions
Weather
Shop@MSNBC
MSN.com
Headlines
LIVE VIDEO
Fed chairman addresses Senate
shop now!
DOW 10,821.59 125.31
Story title
Housing starts decline
Lowest level in over 2 years
Deadly southern heat wave
Temperatures in triple digits
Follow British Open live
Leaderboard from golf.com
'Scoop': Costner's odd dance
How he's offended Hollywood
Chat Video Audio E-mail
Menu button
Search MSNBC GO
Enter ZIP code to get local news, sports and weather
The #1 Internet News Site
ZIP code entry
Go button
http://www.msnbc.com/tools/nm/nm0.asp?c=N4
File Edit View Tools Communicator Help
Send Quote Address Attach Options Spelling Save Security Stop
To:
Select panel
"Subject:" field
"To:" field
Subject: Priority: Normal
Text entry area
Scroll bar
Mail button

[Image is presented in color.]

Figure 6.8: Netscape browser and mail windows with selected buttons highlighted

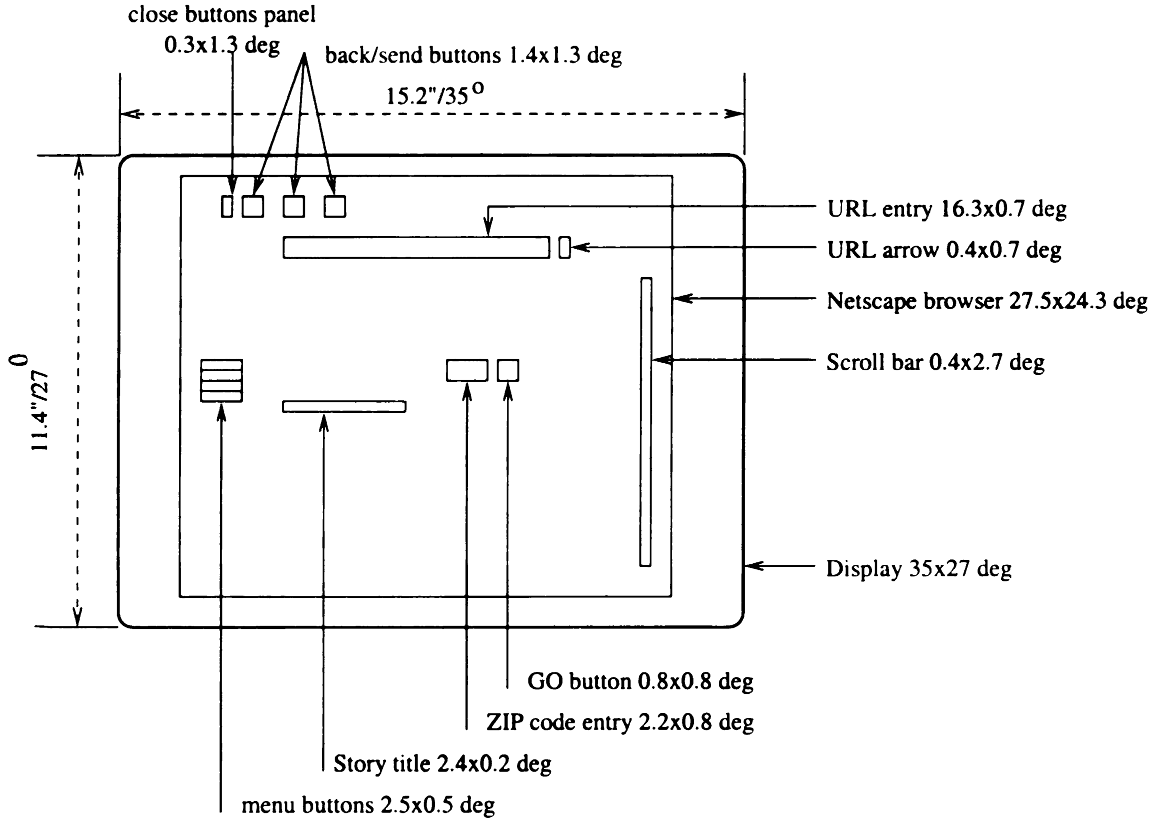


Figure 6.9: Netscape button sizes relative to the display size

viewing angle arranged in an array with no spacing in between. Thus, if the users are to successfully use the head-eye input interface, the accuracy of the system must be up to a few degrees of viewing angle.

If we are to use this interface on a different display size, the pixels and inches values would differ based on the display size. In case of a smaller display (e.g., 14" diameter or less monitors) the user would, generally, sit closer to the display. Thus, for small values of the viewing angles, the pixel count would decrease significantly comparing to the medium size display (e.g., 19"). As the result, the selection areas would need to be larger, in order to achieve a reasonable accuracy. On the other hand, in case of a large size display (e.g., Immersadesk of 4 × 5 ft or 77" diameter), the small viewing angles would result in a larger pixel count than in the case of the medium size display. This would result in the possibility to achieve a higher selectable resolution on a large size display.

6.3 Advantages and Disadvantages of the Head-Eye Input Interface

Each new input interface must have some advantages over the existing ones to be widely accepted and used. In the case of the mouse, it offered an easy pointing mechanism, and was widely accepted. Devices like the trackball are widely used in the notebook-style computers because of their size. Touch-screens are a widely used and convenient device for many public information systems.

	Head-eye	Eye-tracking	Mouse	Trackball	Touch-screen
Needs hand-eye coordination	No	No	Yes	Yes	No
Needs and occupies hands	No	No	Yes	Yes	Yes
Tolerates head movements	Yes	No	Yes	Yes	Yes
Display resolution	Medium	Medium	Fine	Fine	Coarse
For small size displays (ATM machines)	Yes	Yes	Yes	Yes	Yes
For medium size displays (Workstatations)	Yes	Yes	Yes	Yes	Yes
For large size displays (Immersadesk)	Yes	No	No	No	No
Selection times	3 sec	1 sec	1 sec	1 sec	1 sec

Table 6.1: Features and requirements for the head-eye input, eye-tracking based input, mouse, trackball and touch-screen

On the other hand, eye-based input has been used mainly by handicapped users, often as their only way to control the computer. Due to constraints on head movement and no easy way to make selections, this interface has not been widely accepted. Head-based input has been used in some military applications and by handicapped users. Usually, a device is mounted on the head and tracked. Similar to the eye-based input, this interface is not widely used.

The head- and eye-based input interfaces however offer a more natural interaction with the computer. Table 6.1 gives some features and requirements for the head-eye input interface, eye-tracking based input, mouse, trackball and touch-screen. In all hand-based interfaces, the users must check with their eyes the results of the input device movements. Thus, hand-eye coordination must be learned. For some users (e.g., elderly and children) this is not easy to achieve. All the hand-based interfaces, naturally, occupy the users hands. In tasks such as blind typing, if the user wishes to select another window, he/she must first move one hand off the keyboard, then position it on the mouse, trackball or screen, make the selection, and finally put it back on the keyboard and resume typing. Even for experienced users, this task can cause a number of errors and slow-downs. Errors such as missing the mouse position or wrong re-positioning of the hand on the keyboard are common. In terms of head motion, no hand-based devices constrain it. A head-eye input interface does not require the head to be still, the user must move the head in order to control the cursor. In the case of eye-based input, in most systems the user's head must be stationary or some small head motion is allowed.

In terms of the display resolution and size, head-eye and eye-tracking input can be used for fine resolution selections, but require some skill from the user, and are prone to errors. This is due to the nature of our head and eye movements: it is very hard for the user to keep the head perfectly still and it is impossible to totally control eye movements. A touch-screen cannot be used for fine resolution selections. When used on small size displays (e.g., ATM machines), a head-eye input interface would be suitable since the number of options on such screens is not high and the buttons are typically large. In terms of medium size displays (e.g., workstation monitors), all interfaces are suitable. However, in the case of a large display such as the Immersadesk, "conventional" input devices like the mouse, trackball, or touch-screen are not suitable. Due to the screen size the user would need to move the head significantly in order to locate the cursor, or would need to move the whole body and hands to reach the buttons on the display. Eye-based input, similarly, would not be suitable in such an environment: the user would need to stand far from the display in order to be able to have the whole display in the 15° viewing range. In such an application, head-eye based input is a natural solution: the users need to move their head naturally to view various points on the display.

Finally, in terms of selection times, when measured on the conventional workstation monitor, the selection times for eye-input, mouse, trackball and touch-screen are about 1 second per selection. In the case of the head-eye input interface, the selection time is about 3 seconds per selection. This is significantly higher than for other input devices. Even the expert user (e.g., the author of the interface) was not able to make selections faster.

We must note, however, that the 3 seconds time includes the time needed for the user to position to the target location, dwell time before the mouth state is checked, and the time needed to open the mouth by the user. If the users would be making selections by, for example, depressing a hardware button, the selection time would decrease. In some preliminary experiments, the selection times were about 2 seconds.

6.4 Summary

In this Chapter, it was described how a gaze tracking system was integrated into a human-computer interface. Several selection mechanisms were discussed: response by nodding and shaking the head, and selection based on opening the mouth. GUI design issues such as level of feedback and button sizes were discussed and preliminary results were presented.

Chapter 7

Evaluation of the Head-Eye Input Interface

Eye-tracking systems have been used in many ways. Mainly, their use has been limited to handicapped users and as a research tool in visual perception studies. Recently, the development of handless interfaces is gaining momentum and new head and eye tracking systems are being developed for the wide range of users.

In Spring 1998 the author conducted market research for an independent study course. The research was done with Kang Hun Lee, an MBA student. In the course of the study, two focus group interviews were conducted and the participants suggested the following applications: "mouse/trackball/touchscreen replacement, aid/train/monitor handicapped persons, control head moves on an avatar in VR applications, as an advertisement aid, for choosing TV channels, as a teleconferencing aid, in focus of attention measurement, in a virtual museum, for monitoring high-risk working environments, as an identification system aid, for opening the door when one's hands are full, for typing using gaze direction, for screen savers, to count the number of people in a crowd, in hospitals to aid/monitor patients, for an automatic door, for security in banks to aid clerks during a robbery, as an aid for children to use computers, in car theft prevention, in opthalmology to fit glasses, to zoom the screen, and for selecting from a menu in a fast-food restaurant". As our participants suggested, applications are numerous, ranging from support systems for the disabled, through replacement for a pointing device, to the aid in cognition and market research.

In order to assess the usability, performance and possibilities of the head-eye input interface, we conducted a comprehensive evaluation of the interface. In this final Chapter the results are presented. In the evaluation we attempted to incorporate most of the tasks needed for typical applications. With these results we hope to shed more light on the requirements of a head-eye input interface, as well as on the pace of learning to use a novel interface.

7.1 The Goals of the Study

In order to verify the assumption that subjects improve performance with the head-eye input interface and to assess the range of tasks that can be performed we designed an experiment with the following objectives:

- Introduce novice user to the interface and let him/her learn basics of the usage.

- Perform a set of common computer tasks and assess the performance levels. The tasks should be arranged in increasing level of complexity. The tasks that are performed first should serve more as the learning tasks. The final task should incorporate the learned material from the previous tasks.

- Repeat the sessions after some time interval and compare the performance. The number of repeats is arbitrary and depends on the circumstances of the experiments. The more the users repeat, the more skilled they should be.

  In the repeated session, the subjects would perform *exactly* the same tasks and under the same conditions. That way, the only variable from session to session is the subjects and we can assess a subject's learning pace.

In our case we had to constrain the session time to one hour and we were able to have only one repeated session. These constraints were due to subjects' availability. Thus, the range of tasks had to be narrowed to be able to fit everything in the prescribed time interval.

7.2 Subject Pool

Eighteen subjects gave informed consent for participating and were paid $10 for each session. The subjects of various complexions participated and all of them repeated the experiment. Table 7.1 details variations in complexions of the faces. As can be seen, many variations in the human population were presented in the subject pool. Most of the subjects were students or professors at Michigan State University, and two of the subjects were 10-year old children. A number of subjects normally wore eyeglasses and they had to take them off during the first session. During the second

Race:	Caucasian - 9, Asian - 6, Indian/African - 3
Hair color/style:	black - 10, brown - 4, blonde 2, bald - 2
Gender:	male - 14, female - 4
Glasses:	no - 12, yes - 6

Table 7.1: Subject pool: details of the variations in facial complexities

session, 4 out of 6 subjects with glasses and 2 children had blue rectangular spots that were tracked instead of the facial features.

The set-up of the experiment was the same for all the subjects: we used a black background to avoid any erroneous detection of faces in the pink carpeting we had in the room. For one dark-skin subject we used an additional light source that enabled successful tracking.

Along with the results of our subjects, the results for the author herself are presented. The author did not participate in the study since she can be considered an experienced and expert user. The author's results (as subject 20) are presented just for reference and are not included in any of the calculated statistics.

7.3 Evaluation Procedure

In each session, subjects performed five tasks. In the later text, we will use names task T1, T2, T3, T4 and T5. The tasks were the following:

T1: Subjects had to move the cursor along a curve for 60 seconds. The video image with highlighted tracked feature point locations was displayed in the upper left corner of the display. The curve spanned the whole display area. A total of three curves were used:

- *curve 0* $\begin{cases} x = \cos(t) \cdot \cos(t), \\ y = 0.5 + 0.2 \cdot \sin(4t) \end{cases} t = [-\pi ... \pi]$
  (shape of figure 8)
- *curve 1* $\begin{cases} x = 0.5 + 0.2 \cdot \sin(4t), \\ y = \cos(t) \cdot \cos(t) \end{cases} t = [-\pi ... \pi]$
  (shape of figure ∞)
- *curve 2* $\begin{cases} x = 0.5 + 0.5 \cdot t, \\ y = 0.5 + 0.5 \cdot \cos(4 \cdot \cdot \pi) \end{cases} t = [0...1]$
  (wave form spanning from the left to the firth edge of the display)

The curves were presented to the subjects in the following order: curve 0, curve 1, curve 2. The measure of the performance was squared error of each cursor location with respect to the closest curve point averaged over the number of cursor update points.

T2: Guided button selection on an 8×8 grid. In each trial, the target positions were randomly selected and there were 10 targets. The random number generator had a unique seed value for each trial, thus, all the subjects repeated the same sequence of the target buttons. A target button would be highlighted in the blue color by the program and the subject would have to align the red dot that represented the selected grid button with the blue target grid button. When the points were aligned, the subject would open the mouth to make the final selection. If a button was not selected after three attempts, the next target button would be introduced. The GUI consisted of the grid buttons arranged to cover the whole display area and the only feedback was on the tracking success was the selected grid point. Three trials were done. After the head-eye input control and selection, the subjects performed the same tasks using the conventional mouse so that the results obtained with the two input interfaces could be compared.

The measure of the performance was the number of buttons correctly selected, and in which attempt, as well as the time needed for each selection. The time was measured with respect to the distance between buttons, by Fitts' law (see Section 2.6.4). The conventional mouse data was plotted as one linear regression line, the data for the first trial as the second line, and the data for second and third trials were plotted as the third line. The reason for this latter breakup is to compare the performance in the initial (training) trial, and in the two later (testing) trials. Since we assume that all the subjects are experienced mouse users, we analyze all the three mouse trials together. In addition, the cursor path for each button selection was plotted.

T3: Guided dragging of 100 × 100 pixel icons. The initial and target positions were randomly selected, and there were 10 draggings in one trial. The random numbers generator had a unique seed value for each trial, thus, all the subjects repeated the same sequence of the target location pairs. The initial position was labeled with a basketball image, and the target position was labeled with a basket image. The subject had to correctly place each basketball icon, and only then the new icon position would be displayed. The selection was done by opening the mouth, and the mouth was kept open while the icon was dragged to the destination. When the mouth would be closed, the icon was released. There were three trials as in the task T2. Upon completing the task with head-eye input, the subject would repeat it using the conventional mouse. The measure of the performance was the same as for the task T2, with the addition of measuring the number of steps needed to move the icon from the initial to the target position, where a *step* is the number of select-move-release tasks a subject performed.

T4: Automatic determination of the gaze path and fixation points. The users examined three pictures for 30 seconds each. The results of this task were presented in Section 5.5.1 and are given in full in Appendix A.

T5: Surfing the internet and writing an email message using head-eye input for moving the cursor and selection, and keyboard for typing. All the subjects performed the same task that included typical browsing and writing tasks. The Netscape program was used. The layout of the windows with selected buttons is depicted in Figure 6.8. Initially, the Netscape window was centered on the display and the mail program was minimized and its icon was located just left of the upper left corner of the Netscape window. The subjects did the following tasks:

1. Click on the URL entry field, using backspace/delete keys delete the current contents, type in "www.msnbc.com", and hit return key. The news page would load.

2. Click on the headlines menu button (the last button among the menu options). The intermediate page with an advertisement would load and the subject would need to click on the sentence "Click here to go to headlines", or simply wait for a few seconds, and the headlines page would load. If the subject would click on the advertisement image, s/he would need to click to the "Back" button to return to the headlines page.

3. Scroll to the very bottom of the page by clicking on the area just above the scroll arrow. The number of clicks was about 5, depending on the amount of information in the headlines page.

4. Click on the first weather story (located at the bottom of the headlines page). The page with the menu options on the left and the weather story on the right would load.

5. Click on the weather menu button. There was the advertisement page also, as for the headlines button.

6. Click on the ZIP code entry field, enter local ZIP code using the keyboard, and click on the "Go" button located just right of the ZIP code entry area. Weather report would load.

7. Click on the mail icon. The icon would open up in the upper left corner of the display.

8. Click on the "New msg" button.

9. Type in an email message to the author about the weather report. The typing tasks included clicking on the "To:" field to enter the author's

email address, clicking on the "Subject:" field to enter the subject of the message, and clicking on the text entry area and typing the body of the message.

10. Click on the "Send" button.

The experiment procedure was the following:

- The experiments were conducted in the PRIP lab in the Computer Science and Engineering Department at MSU. The experiment set-up consisted of: an SGI Octane workstation with a camera mounted on the top of the monitor. The subject would sit in a chair without any constraints as if s/he were working at the computer in the usual circumstances. The procedure was completely non-invasive. The experimenter was the author of the program.

- During the experiments the video data was not recorded, but processed by the program and discarded. The only data saved were the coordinates of the tracked points on the face, screen pointer coordinates, and statistics relevant to the experiment (e.g., time to select a button, whether correct button was selected, etc). The session of task T5 (Net-surfing) was taped (the display and the subject's side profile) so that the task could be later evaluated.

- The training session consisted of a general demo of the system by the experimenter, then the subject tried the system (s/he would see the picture from a video camera, with eyes, eyebrows and nose marked by the program, and s/he would be able to move the cursor on a grid and practice the selection mechanism). The training session lasted 5–10 minutes. No data was collected during the training session.

- The testing sessions would test tasks T1, T2, T3, T4, and T5, in that order. Before the data collection would begin, the experimenter would briefly describe the steps in the current task. Then, the task would be performed and the data would be collected. The projected times needed for each task were: T1: 5 minutes, T2: 5-10 minutes, T3: 5 minutes, T4: 2 minutes, T5: 15 minutes. Upon conducting all the tasks, it turned out that T2 was completed in about 5 minutes, and T3 in about 10 minutes. All other projected times were correct in practice.

- Finally, the subject would fill-out a short follow-up questionnaire.

The second session was repeated about one week after the first session and all the tasks were the same. The only differences were that T4 was not conducted for the second time and there was no questionnaire.

7.4 Results

In this Section, the results of the study are presented. Each task is presented in a separate section for clarity.

7.4.1 Task T1: Moving the Cursor Along a Path on the Display

In the Task T1, the subjects had to move the cursor along a curve for 60 seconds. The video image with highlighted tracked feature point locations was displayed in the upper left corner of the display. The curve spanned the whole display area. A total of three curves were used: c0 - ∞, c1 - 8, and c2 - wave form. The performance for each subject was measured as the average squared error with respect to the target curve. All the coordinates were normalized to the 0...1 range. The data for the first and second session for each subject were compared. Squared error for the individual curves ranged from 0.03 to 0.15 in the first session and the sum of squared errors for all three curves ranged from 0.14 to 0.31. In the second session, the error for the individual curves ranged from 0.02 to 0.13, and the sum of squared errors for all three curves ranged from 0.11 to 0.28. Figure 7.1 gives the average squared error in the two sessions as well as the errors achieved by the author. The overall performance increased, both the minimal and maximal value of errors decreased in the second session. In terms of percentage of improvement, the average error for each curve improved, ranging from 8% to 15%, and for the overall error, the improvement was 12%.

As for the improvement for the individual subjects, for curve 0 61% improved their performance, for curve 1 55% improved, and for curve 2 72% improved. In terms of the overall improvement (sum of the squared errors for all three curves), 72% improved the performance.

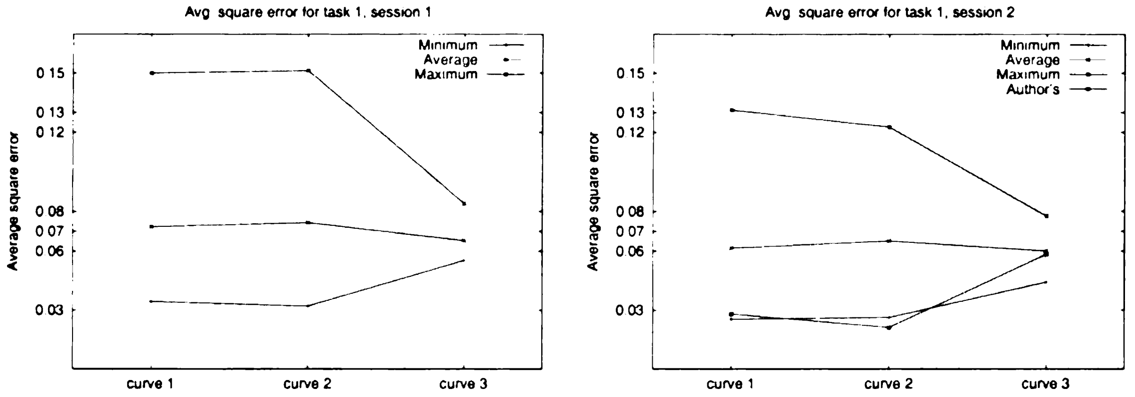
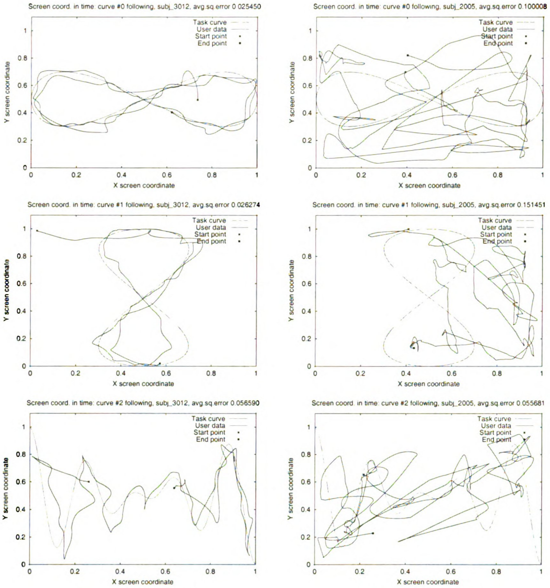


Figure 7.1: Task T1: average squared error

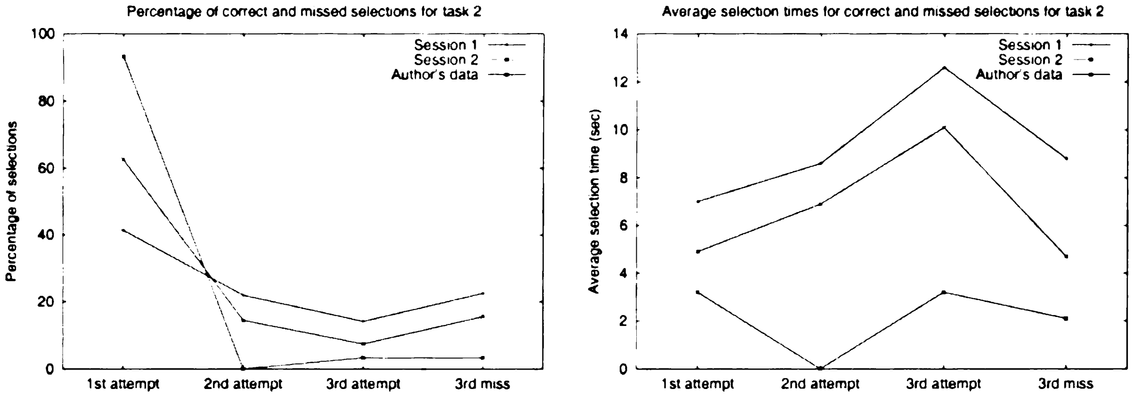
To illustrate the shape of the cursor paths, we plotted the target curve and the cursor path that the user produced. Figure 7.2 gives the cursor path for the best and worst overall subject performances. All the three curves for the same subject are given. The subject who achieved the best performance was able to move the cursor along the target path rather well and the errors were minor and the general shape of the movement was in the shape of the target curve. In the case of the worst performing subject, it is very hard to determine the movement pattern, and clearly, the subject struggled to move the cursor even close to the target curve. In the case of the second curve, the cursor was completely off the curve most of the time. Complete comparative results for each subject are presented in Appendix B and the improvements shown in the squared error comparisons are clearly visible with the bare eye.



subject 12, session 2
best overall performance

subject 05, session 1
worst overall performance

Figure 7.2: Task T1: cursor path for the best and worst squared error results in the first session



(a) Selection accuracy statistics

(b) Selection times (seconds)

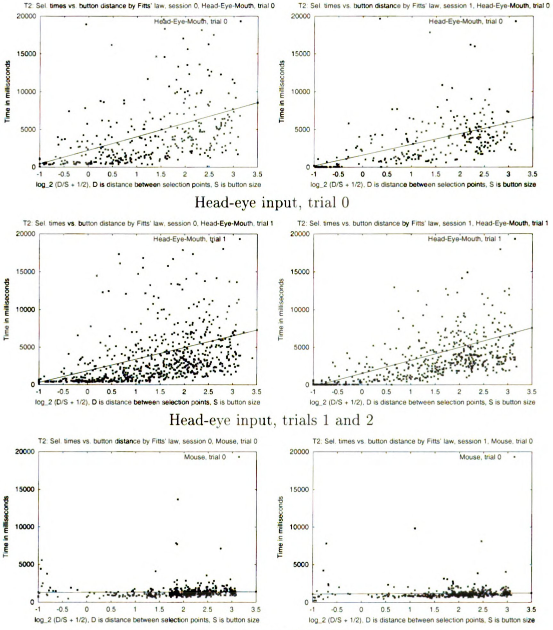
Figure 7.3: Task T2: button selection accuracy and selection times

7.4.2 Task T2: Guided Button Selection

In the task T2, the subjects had to select buttons on an 8 × 8 grid, as guided by the program. A random target button would be highlighted and the subject would have to select it. After three unsuccessful attempts a new target would be introduced. There were three trials, with ten randomly selected targets. The performance for each subject was measured based on the number of correctly selected buttons and the total time spent for button selection. Also, the performance using the head-eye input was compared with the mouse-based selection. The results were analyzed for all the three trials of the head-eye or mouse input, or for the first head-eye trial (practice trial), the second and third head-eye trials (test trials), and all three mouse trials.

Figure 7.3(a) compares the button selection accuracy in the first and second session. Subjects were able to make more accurate selections in the first attempt in the second session, on average 63% in the second vs. 41% selections in the first session. In addition, the number of buttons selected in the second and third attempt decreased

Session 1 Session 2



Head-eye input, trial 0

Head-eye input, trials 1 and 2

Mouse input, trials 0–2

Figure 7.4: Task T2: selection times vs. button distance by Fitts' law

from 22% and 14% in the first session to 14% and 7% in the second session. The number of incorrectly selected buttons decreased from 23% in the first session to 16% in the second session. This means that the users were generally more accurate in the second session. The overall number of correctly selected buttons increased from 77% in the first session to 84% in the second session. These results are comparable with those of the Ware and Mikaelian [86] eye-mouse study.

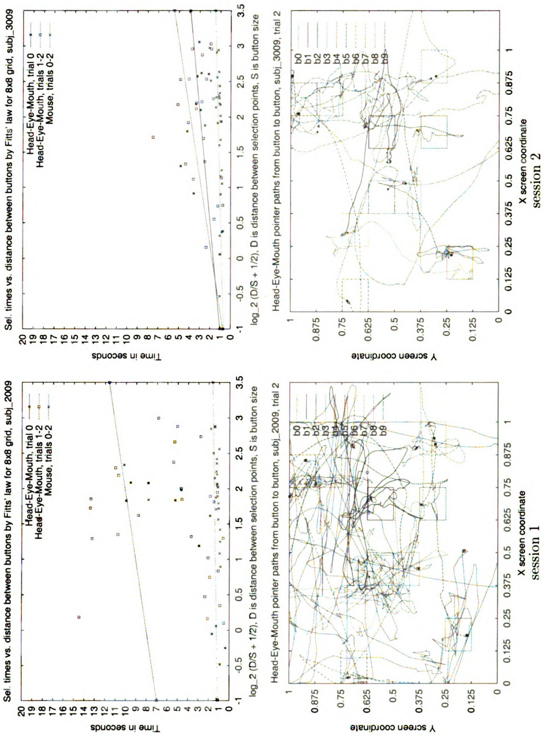
For every correct selection in the second and third attempt, the subject made one and two erroneous selections. In the first session, for each correct selection subjects made 1.5 erroneous selections. In the second session, for each correct selection subjects made 0.7 erroneous selections.

Figure 7.3(b) compares the selection times in the first and second session. The selection time increases with the number of attempts, which is expected, since the user needs to make more than one selection. The selection time in case of the incorrect selection after three attempts is close to the time for the correct selection in the first attempt. This means that the users were making erroneous selections unwillingly, e.g., either they were reacting slowly to the stimulus of the new target or the open-mouth detection algorithm produced false alarms. The time needed to make the first correct selection decreased in the second session, from 7 to 5 seconds. In addition, the decreased selection times resulted in the overall decrease of 31% in the time needed to complete the experiment: from 238 seconds in the first session to 164 seconds in the second session.

As for selection with the mouse, out of 1050 selections, only 9 (0.8%) were wrong, and all were correctly selected in the second attempt. The selection times using the mouse were around 1 second for all subjects.

Figure 7.4 plots selection times vs. button distance by Fitts' law. The left column shows the data for the first session and the right column shows the data for the second session. The first row gives the data for the first head-eye input trial, the second row for the second and third head-eye input trial, and the third row gives the data for the mouse trials. The linear regression line is also plotted for each data set. As can be seen, the data for the head-eye input do not fit well into the Fitts' law framework, when analyzed for all the subject. The data for the individual subjects who performed the task well fit into the Fitts' law framework better.

Figure 7.5 depicts the selection time vs. the distance, by Fitts' law, and cursor paths from button to button for a subject whose performance on the task increased significantly in the second session. As can be seen, in the first session, the subject's data is very hard to fit in the Fitts' law framework and the time needed to select each button was very high. The cursor paths from button to button also show that trend: there is a lot of motion just to move the cursor to the desired location and a lot of motion around the target location. In the second session, however, we see that the selection times for each button were below 5 seconds and in some cases close to the mouse selection times. The cursor paths were much more direct and the subject clearly didn't make a lot of erroneous motion. Complete results of the Task T2 are presented in Appendix C, where the data for each subject are compared along the two sessions.



session 1

session 2

Figure 7.5: Task T2: selection times vs. distance by Fitts' law, and cursor paths for trial 2 for a subject whose performance increased in the second session

When we add the result that the number of errors decreased in the second session, we can define overall subject performance in a session using the following formula:

$$T2\_session\_performance = (1 + number\_of\_misses) \times total\_elapsed\_time$$

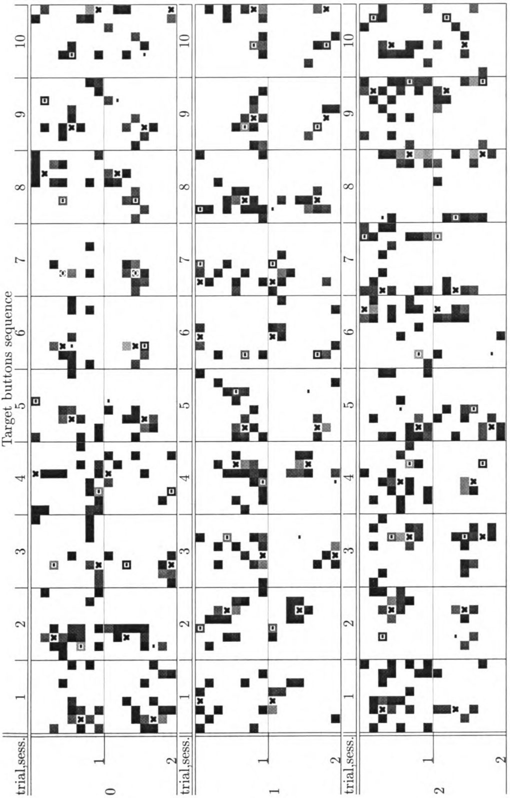
The lower the score, the better performance, since we want to minimize both the number of misses and the elapsed time. When compared using the above formula, we get more balanced results, and can compare the scores of different subjects.

In terms of the improvement for the individual subjects in the second session, 83% of the subjects decreased the elapsed time for the task, 67% of the subjects decreased the number of misses. Overall task performance improvement was observed in 83% of subjects.

Sources of Erroneous Selections

Why did the subjects make erroneous selections? Did they improve in time? It is important to discuss these issues so that we can identify possible problems and correct them. Figure 7.6 shows which buttons the subjects selected for each target button in the first and second session, respectively. For ease of reading, only the non-zero entries are shown. The number in each entry represents the number of times that button was selected while selecting the target button displayed in bold. We can observe three things:

1. We can see that often subjects selected buttons adjacent to the target button. That is understandable, the reason for this error is that they either didn't position the cursor properly or moved slightly while opening the mouth.



The two adjacent rows represent the data for the same trial in the first and second session. The columns represent the error distribution for each target button. In each table entry, the cross represents the target button, the dash represents the previous target button, and gray shades represent the percentage of the erroneous selections at the non-target button locations (dark shades of gray represent more errors, and bright shades represent less errors).

Figure 7.6: Task T2: Distribution of the erroneous selections for each target button

2. The number of erroneous selections at the coordinates of the previous target button is high in most cases. The reason for this error is that subjects kept their mouth open (or kept opening and closing it) when the new target was initiated and they simply didn't move to the new target and stayed still fixating the previous target. In some cases the subjects would move very slowly from one target to another while keeping their mouth open most of the time (in spite of the experimenter suggestion to close the mouth and open it only when they want to make a selection). In other cases the algorithm for detecting the open mouth state was simply finding that the mouth was open when it was closed, which would cause an erroneous selection. To overcome this problem, we could repeat the same experiment using a different selection mechanism, e.g., hardware button that would be less prone to errors.

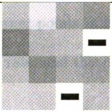
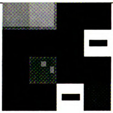
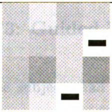
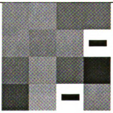
3. In the second session, the number of erroneous selections at the coordinates of the previous target button decreased. It slightly increased in the last trial of the second session towards the end of the experiment, which can be connected with the fatigue of the subjects or their decreased attention. The general trend in the second session was that the erroneous selections were not widespread across the grid, which means that the subjects got more skilled using the interface.

Were There Some "Hard Positions"?

We noticed that the number of errors decreased in the second session. We also have to ask whether the errors were uniformly distributed across the display positions?

	trial 0										trial 1										trial 2										
Button	2	3	4	5	6	7	8	9	10	∑	2	3	4	5	6	7	8	9	10	∑	2	3	4	5	6	7	8	9	10	∑	∑
Sess. 1	y	y	y	n	y	n	y	n	n	5	y	y	y	y	y	y	n	n	n	6	n	y	y	n	y	y	n	y	y	6	17
Sess. 2	n	n	n	n	n	n	y	n	n	1	y	n	n	n	y	n	n	n	n	2	n	n	n	y	n	y	y	y	n	4	7

Table 7.2: Number of erroneous selections at the previous target location: "y" entries mean that there were more than three selections at the previous target location, "n" entries mean that there were less than or equal to three selections at the previous target.

Session 1		
Session 2		
	percentage of correct selections: bright shades represent higher selection accuracy and darker shades represent lower selection accuracy	*number of erroneous selections per each correct selection:* dark shades represent more erroneous selections per one correct selection for a target button, and bright shades represent less erroneous selections per one correct selection

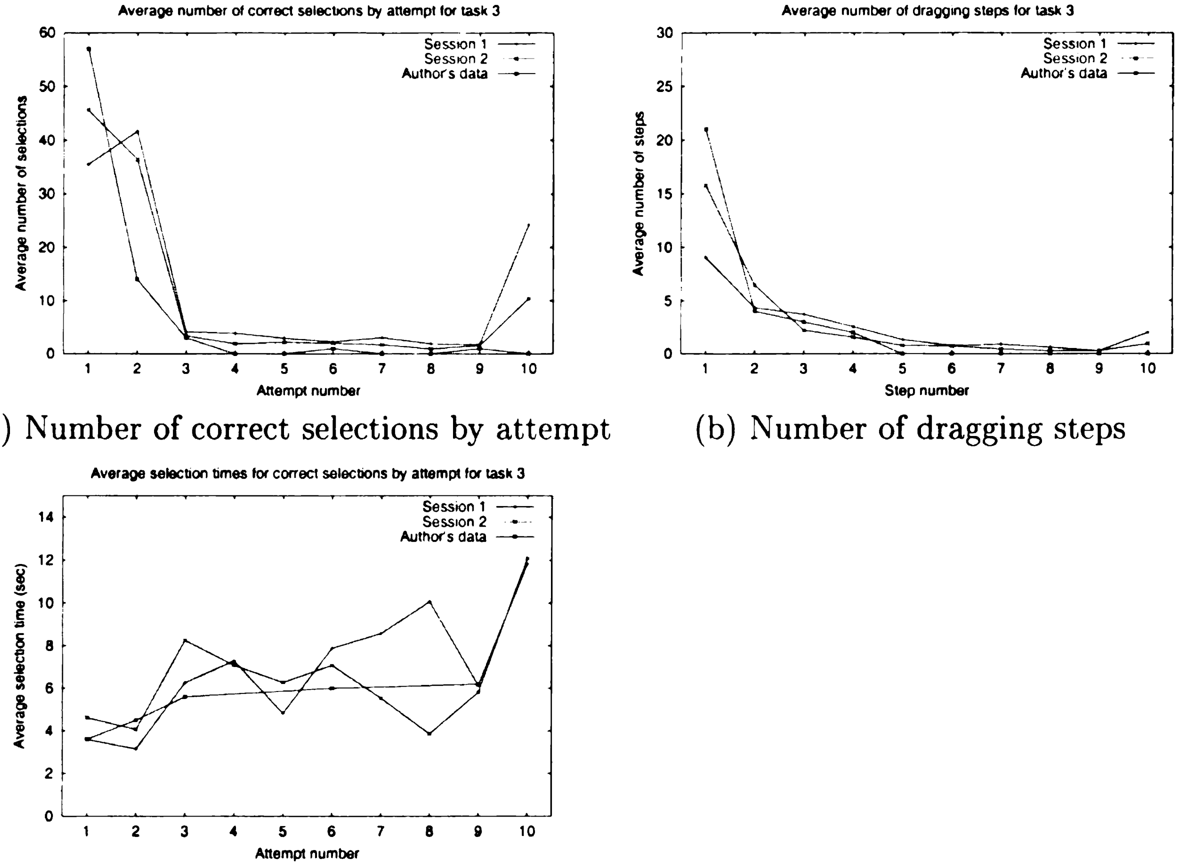
(Entries with a dash were not the target.)

Table 7.3: Percentage of correct selections and the number of erroneous selections per each correct target selection: data from the 8 × 8 grid is collapsed into a 4 × 4 grid.

Table 7.3 shows the percentage of subjects who selected each target button correctly and the number of erroneous selections made per one target button selection. The data from the 8×8 grid is collapsed into a 4×4 grid for clarity. In the first session, the percentage of the correct selections for each target button ranged from 56% to 89% with 73% average, while in the second session it ranged from 67% to 93% with 82% average. We also calculated the number of erroneous selections subjects had to make for each correct target selection. For different display locations, in the first session the number ranged from 0.6 to 1.4 with 1.1 average, and in the second session it ranged from 0.5 to 1.3 with 0.8 average. The errors were distributed rather uniformly across all the grid locations. There seems to be a slight increase in the number of erroneous selections in the corner points in the second session. However, not all corner points had this high error rate, and this does not hold for the first session.

7.4.3 Task T3: Guided Dragging of Icons

In the task T3, the subjects had to drag 100 × 100 pixel icons, as guided by the program. The initial and target positions were randomly selected and there were 10 draggings in one trial. The initial position was labeled with a basketball image and the target position was labeled with a basket image. The subject had to correctly place each basketball icon on the basket icon, and only then the new icon positions would be displayed. The selection was done by opening the mouth, and the mouth was kept open while the basketball icon was dragged to the basket destination. When the mouth would be closed, the icon was released. There were three trials as in the



(a) Number of correct selections by attempt

(b) Number of dragging steps

(c) Selection times (sec) by attempt

Figure 7.7: Task T3: number of correct selections by attempt and their timing, and number of dragging steps

task T2. The performance for each subject was measured based on the number of steps of dragging needed to drag an icon to the target location and the total time spent for dragging. As in the Task T2, for reference the performance using the head-eye input was compared with the mouse-based dragging. The results are analyzed for all the three trials of the head-eye input or mouse, or for the first head-eye trial, the second and third head-eye trial and all three mouse trials.

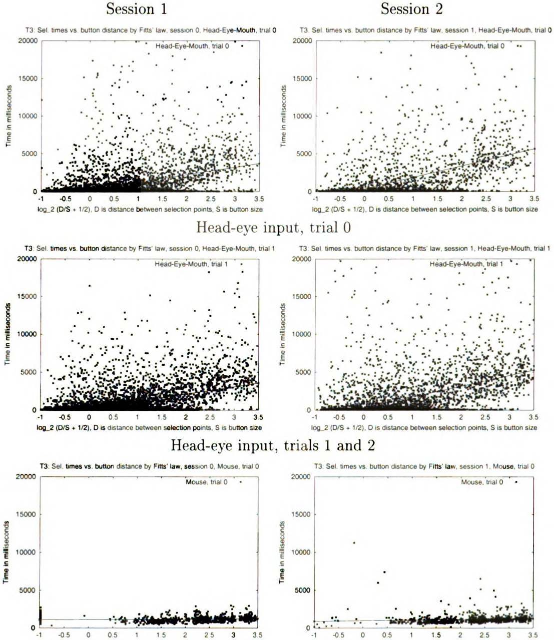
In this case, the subjects had to complete each dragging task; the new target would not be initiated unless the previous target was placed correctly. Thus, we measured the number of steps needed to perform a dragging, where a *step* is the number of select-move-release tasks the subject performed for each target. Ideally, this should be one, that is, the subject should select an icon, move it to the target, and release it. However, in practice, the subjects would place the icon incorrectly, then they would have to select it again and move it to the target. This last step has been repeated many times for some subjects. In our reporting of results, we also give the number of erroneous clicks needed to make a correct selection. In some cases, subjects needed more than 10 steps and/or made erroneous selections more than 10 times. Thus, we analyzed the data when the number of steps/errors was more than 10 as if it were 10 items.

Figure 7.7(a) gives the statistics of the number of correct selections in the i^{th} attempt ($i = 1...10$). In both the first and second session most of the correct selections were made in the first or second attempt. In the second session the number of times the correct selection was made at or after the 10^{th} attempt decreased 58% (from 24 to 10). The author made mostly correct selections in the first attempt. Figure 7.7(b) gives the selection times for each correct selection in the i^{th} attempt. The selection times for both the first and second correct selection were about 4 seconds. In the first session, the time needed in higher attempt number, generally, increased, while in the sêcond session, the time needed in higher attempt number, generally, decreased. In both sessions, selection time was around 12 seconds at or after the 10^{th} attempt. The author achieved times similar to other subjects for in the first two attempts. However, for higher attempts, the author needed more than 3 attempts only twice (3% of selections).

Figure 7.7(c) gives the average number of times the dragging had to be performed in i steps (i = 1...10). In the second session, most subjects were able to complete most draggings in one or two steps, unlike the first session, where the number of steps varied greatly. The author, for example, never needed more than four steps.

As for the dragging with the mouse, out of 2100 selections, only 16 (0.8%) were wrong, but all of those were correctly selected in the second attempt. The selection times using the mouse were around 1 second for all subjects.

Figure 7.8 plots selection times vs. distance by Fitts' law. The left column shows the data for the first session, and the right column shows the data for the second session. The first row gives the data for the first head-eye input trial, the second row for the second and third head-eye input trial, and the third row gives the data for the mouse trials. The linear regression line is also plotted for each data set. As can be seen, the data for the head-eye input do not fit well into the Fitts' law framework when analyzed for all the subjects. The data for the individual subjects who performed the task fit better into the Fitts' law framework better.



Head-eye input, trial 0

Head-eye input, trials 1 and 2

Mouse input, trials 0–2

Figure 7.8: Task T3: selection times vs. distance by Fitts' law

Figure 7.9 depicts the selection time vs. the distance, by Fitts' law and cursor paths from button to button for a subject whose performance on the task increased significantly in the second session. As can be seen, in the first session the subject's data is very hard to fit in the Fitts' law framework and the time needed to select each button was very high. Such a huge number of selections is the result of frequent losses of the mouth tracking (or the user didn't keep the mouth open consistently), and that resulted in many short paths of the target icon. In the second session, the number of such short paths decreased significantly.

The cursor paths from target to target also show that trend: there is a lot of motion just to move the cursor to the desired location and a lot of motion around the target location. In the second session, however, we see that the selection times for each target were much more consistent with the Fitts' law curve, and range 5–8 seconds. The cursor paths are much more clear and the subject clearly didn't make a lot of erroneous motion. Complete results of the Task T3 are presented in Appendix D, where the data for each subject are compared across the two sessions.

In order to calculate the overall performance, we have to incorporate the number of steps needed for each dragging and the total time. The number of steps can be calculated using the formula:

$total\_steps = \sum_{i=1...10}(i \cdot number\_of\_steps_i)$.

The last row of Table D.1 shows the total number of steps averaged over subjects. The minimal value was lower in the second session (31 vs. 35), and, the average value was also smaller in the second session (67 vs. 84). The value of 67 steps is the number of selections of target icon, and since there was total of 30 icon positions, each dragging

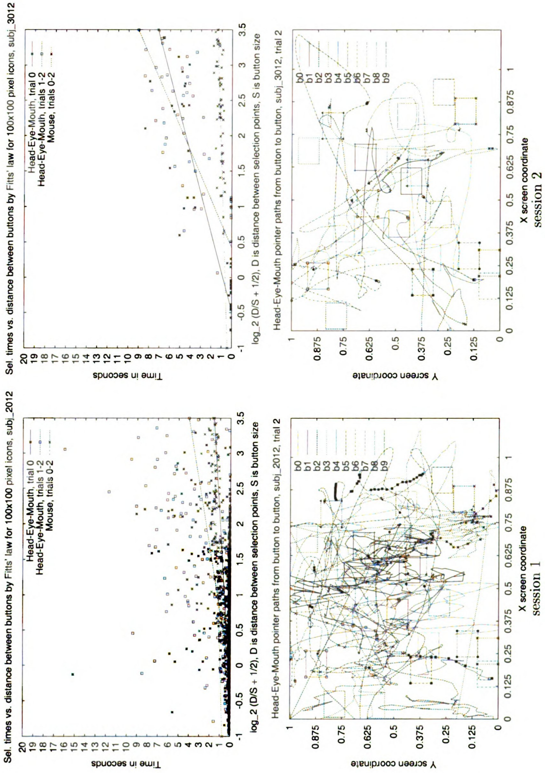


Figure 7.9: Task T3: selection times vs. distance by Fitts' law and cursor paths for trial 2 for a subject whose performance increased in the second session

was performed in 2.2 steps. In the first session, a total of 84 steps resulted in 2.8 dragging steps. Then the overall performance can be calculated using the formula:

$$T3\_session\_performance = total\_steps \times total\_elapsed\_time$$

The lower the score, the better performance, since we want to minimize both the number of steps and the elapsed time. When using the above formula, we get more balanced results and can compare the scores of different subjects.

In terms of the improvement for the individual subjects in the second session, 55% of the subjects decreased the elapsed time for the task, 78% of the subjects decreased the number of steps needed for one dragging. The overall task performance improvement was noted in 67% of subjects.

Spatial Distribution of Errors

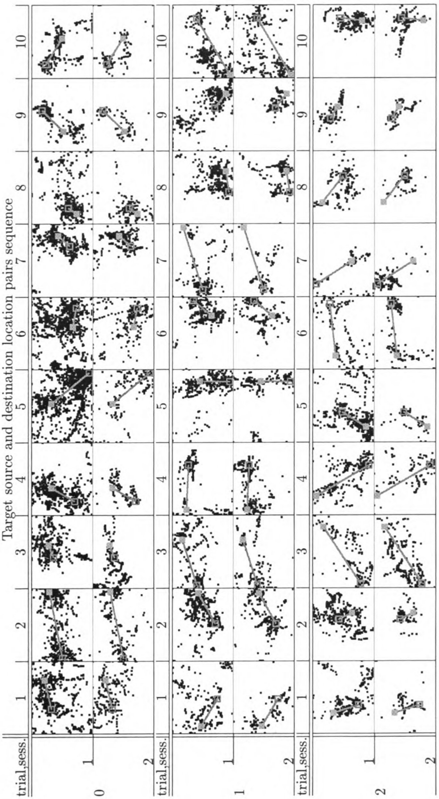
Figure 7.10 gives the spatial distribution of the erroneous selections for each target locations pair for all subjects in the first and second session. The filled gray rectangle is the dragging destination, and an empty rectangle is the dragging source position. The two adjacent rows plot the data for the same trial in two sessions. The general trend is that the number of errors decreases as the session progresses, and that the number of errors in the second session is smaller than in the first session. In addition, in the second session, the errors are, generaly, more clustered around the line that connects the source and destination locations.

Figure 7.11 gives the spatial distribution of the correct selections for each target location pair for all subjects in the first and second session. Plotting format is the same as in Figure 7.10. Ideally, the points should be clustered around the starting and ending dragging positions. However, since the dragging was done in more than one step many times, the selected points were spread across the display. The selections are distributed on or close to the line connecting the two locations, which means that the subjects were attempting to drag the icons along the shortest path to the destination.

Figure 7.12 shows the spatial distribution of the erroneous selections for each target locations pair. The darker the line that connects the target location the more errors were made in that dragging. Clearly, in the second session, the number of errors decreased. In the first session, there was a lot of variation in the number of errors for different target pairs. However, in the second session, the amount of the variation decreased. This leads us to conclude that no particular location on the display caused problems for the users. We were especially interested in the corner points, since the experimenter's and subjects' impression during the experiment was that many subjects had a hard time selecting corner positions.

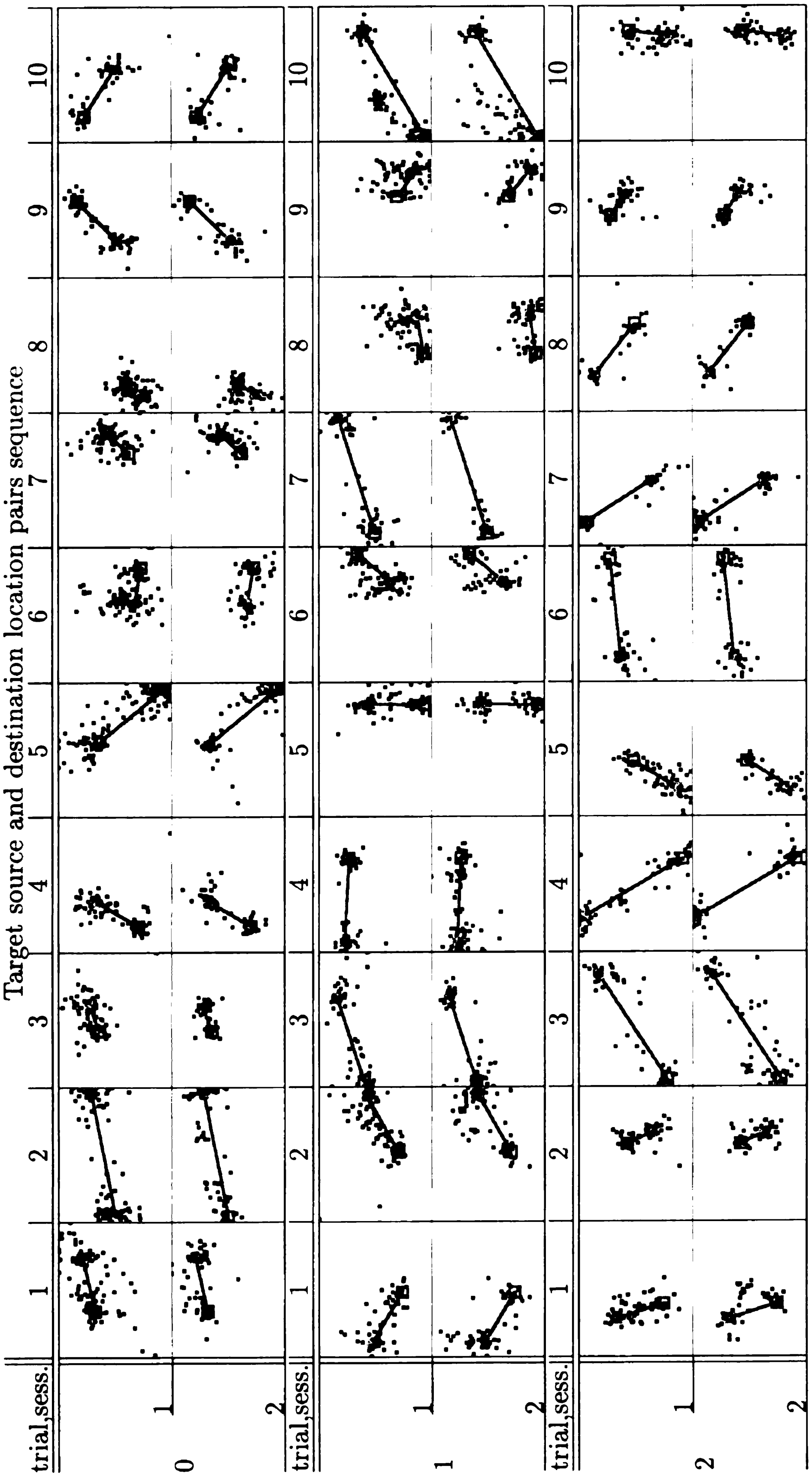
7.4.4 Task T5: Application Control—Net-surfing and E-mail Writing

In the Task T5, the subjects had to integrate all the skills learned in the tasks T1-T3 to perform a real-world task: surfing the internet and writing an e-mail. The details of the task were described in Section 7.3. The performance of each subject was evaluated by watching the video tapes that recorded the session, and the experimenter recorded the number of erroneous clicks, the number of times the experimenter had to turn



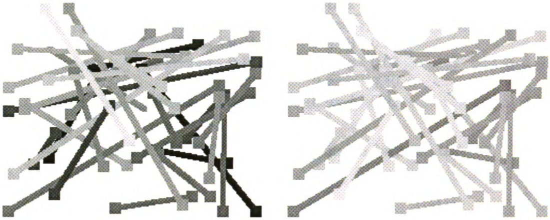
The two adjacent rows represent the data for the same trial in the first and second session. The columns represent the erroneous selections location distribution for each target location pair. In each table entry, the filled square is the dragging destination and the empty square is the dragging source location. The line connects the source and destination in the shortest path. Black dots represent the locations where the selections were made by the subjects.

Figure 7.10: Task T3: Distribution of the erroneous selections for each target locations pair



The two adjacent rows represent the data for the same trial in the first and second session. The columns represent the correct selections location distribution for each target location pair. In each table entry, the filled square is the dragging destination and the empty square is the dragging source location. The line connects the source and destination in the shortest path. Black dots represent the locations where the selections were made by the subjects.

Figure 7.11: Task T3: Distribution of the correct selections for each target locations pair



Session 1

Session 2

Figure 7.12: Spatial distribution of the erroneous selections for each target locations pair: darker lines represent more errors for that target locations pair.

off the camera (in order to assume the control of the pointer) and intervene with the mouse, the number of times the experimenter had to type in text for the user, the total time the subject spent in completing the task, and which parts of the task were skipped. The nature of the task was such that not all the erroneous clicks resulted in some unwanted action. The subject might click on the Netscape window, but that area would not be associated with a button, and there would be no change in the displayed WWW page. Thus, as *erroneous clicks* we counted only those clicks that resulted in some unwanted action such as loading a wrong WWW page or opening an unwanted window (e.g., security information or similar). In such actions, the subject would need to recover from the error by clicking on the Netscape's "Back" button or to click on the "Cancel" button on the popped-up window. All those actions would add extra burden to the task and would require the subject to spend more time in the completion of the task.

sub-task	weight	sub-task	weight
1:url	0.20	6:zip	0.14
2:headlines	0.04	7:mail	0.02
3:scroll	0.10	8:newmsg	0.02
4:title	0.02	9:type	0.40
5:weather	0.04	10:send	0.02

Table 7.4: Task T5: Assigned weights for each sub-task

	Session		Author	
Average over all subjects:	1	2	Head-eyes	Mouse
Number of erroneous selections	8.6	5.2	0	0
Number of camera turn-offs	0.4	0.2	0	0
Experimenter types	0.3	0.2	0	0
Elapsed time (mm:ss)	9:12	8:35	2:40	1.15
Percent of the task completed	85%	96%	100%	100%

Table 7.5: Task T5: average number of errors, elapsed time and percentage of the task completed

For each subject the percentage of completed tasks was calculated based on the assigned weight of each sub-task. The sub-task weights are shown in Table 7.4, and were assigned based on the number of selections and text entries involved in each sub-task.

Some subjects' initial task did not include all the sub-tasks due to time constraints or subject's age. For one child we changed the tasks a bit and adapted to his age. The change preserved most of the sub-task actions but used different WWW pages.

Table 7.5 shows the average error and timing statistics over all subjects in the two sessions. In the first session the number of errors ranged from none to 19, and several subjects were not able to complete the first session. The experimenter had to turn off the camera a total of 8 times and to enter text for the subjects 5 times. From the first to the second session the number of erroneous clicks decreased 40% (from total of 154 to 93), as well as the number of times the experimenter had to intervene (4 interventions with the mouse and 3 with typing). Also, the total time needed to complete the task decreased 7% (from 552 to 515 seconds). Only 5% of subjects (one out of 18) were not able to complete the task, compared to 44% (8 subjects) in the first session.

For comparison the author completed the task in 160 seconds using the head-eye input interface, and 75 seconds using the conventinal mouse. Based on the results of the previous tasks using the mouse we can assume that the author's performance with the mouse is comparable with other subjects' performance. Then we can conclude that the average performance of our subjects in the second session was 6.8 times slower than had the task been peformed with the conventional mouse.

In order to calculate the overall performance of the subject, we calculated the total error using the following formula:

$$total\_errors = \#\_erroneous\_clicks + 2 \cdot (\#\_turnoffs + \#\_experimenter\_typing)$$

The elapsed time is also adjusted for the percentage of the task completed, so the overall time was calculated as $\frac{elapsed\_time}{percentage\_completed}$. Finally, the overall performance was calculated as:

$$T5\_session\_performance = total\_errors \cdot overall\_time$$

The lower the score the better and if the score in the second session was lower than in the first session, we assumed that the subject's performance improved.

In terms of the improvement for the individual subjects in the second session, the elapsed time for the task decreased for 61% of the subjects, while the number of erroneous selections decreased for 67% of the subjects. Overall task performance improvement was noted in 61% of subjects.

Which Sub-Tasks Were Hard?

Were there any differences among the sub-tasks in hardness? Based on the skipped tasks statistics, zip code entry (sub-task 6) was skipped the most times (10), then clicking on the weather menu (3), and all the other tasks were skipped twice or once. It is not fair to say that sub-task 6 was the hardest, since the experimenter decided to skip it often and to go to the mail writing part due to the time constraints and when the session progressed with a lot of errors.

Table 7.6 shows the distribution of errors across the sub-tasks for both sessions. It is clear that the tasks of clicking on a menu button (e.g., headlines or weather) was hard in both sessions. The first click in the task on the menu button (headlines) had more errors than the second click in sequence (weather). The erroneous clicks were always on the adjacent menu buttons. Another problematic sub-task was entering the URL. The errors in this case were mainly made in selecting some buttons that were located just above the URL entry field. Entering the ZIP code was also hard, and the errors were mainly in selecting buttons adjacent to the ZIP entry field. Finally, two other subtasks that had high error rates were clicking on the "New Msg" and "Send" button. These buttons are of the same size as the "Back" button, but were located in the upper left corner of the display. The errors were also clicks on the adjacent buttons, some as small as 0.27^o and 0.53^o of viewing angle. These errors resulted in

	Number of errors			Number of errors	
sub-task	Session 1	Session 2	sub-task	Session 1	Session 2
1:url	24	12	6:zip	23	12
2:headlines	42	22	7:mail	0	3
3:scroll	12	4	8:newmsg	12	12
4:title	8	1	9:type	2	1
5:weather	26	13	10:send	5	11

Table 7.6: Distribution of errors across the sub-tasks

the closure of the panel with main buttons or with the closure of the "To:" field in the mail window.

The scrolling task did not result in a high number of errors, but was hard for a number of subjects. That task took a lot of time and it was problematic due to the small target area size of only 0.36^o of viewing angle in the horizontal direction. In scrolling, there were a lot of erroneous clicks just around the scroll bar, however, they did not result in some action that required the subject to recover from the error.

The typing task also caused some problems to the subjects who were not typing blindly. They had to look at the keyboard when typing and the pointer would move out of the mail window and then the keyboard input would not be accepted by the mail window. This problem could be solved by changing the user interface to recognize the beginning of typing and to freeze the pointer position. In this experiment, however, that was not done. An alternative solution to this problem would be to introduce a "Stop-the-head-eye-input" command that would be issued through some facial expression (e.g., smiling) or by depressing a special key.

7.4.5 Summary of Questionnaires

After the first session all the subjects filled out a questionnaire from which we wanted to capture their impressions of the interface. The following questions were asked:

1. *Do you find the idea of head-eye input device interesting? YES/NO, explain:*

   All subjects answered YES. The predominant explanation was that it would be "useful for handicapped users" (55% subjects) and 39% subjects said that it would be "nice to integrate the interface as the general input", either replacing or working with the conventional mouse.

2. *Did you like the head-eye interface? YES/NO, explain:*

   In this case, 83% subjects said that they liked the interface, and 17% said that they did not like it. The subjects' explanation was that they didn't like it "at its present stage", since they had difficulties in the interaction.

3. *Did you have any problems using the interface (e.g., not able to control the pointer or click, neck pain or discomfort, or other)? YES/NO, explain:*

   Not all subjects specified problems: 89% said that they had problems, and 11% said that they had no problem. Out of the ones that specified them, 44% had problems with cursor control, 37% had problems with clicking, and 25% experienced some neck pain or discomfor. Some subjects reported more than one problem. One subject commented that "it felt a bit strange to them to open the mouth at the beginning", but in time the subject got used to that fact and learned how to open the mouth for the system to recognize that. One subject commen

to movin

move his

4. *Do you h*

Subjects

integratin

have alte

their mo

compute

be impro

to type a

The results

itive. The prob

menter. and son

the second sessi

change in the m

Since nothing ha

perception of the

while performing t

The above sugg

problem: their own

commented that he had problems with the cursor control since he "was not used to moving his head while working with the computer and that he would rather move his eyes only".

4. *Do you have any other comments or suggestions for head-eye interface?*

   Subjects suggested an alternative way of selection, e.g., through blinking, or integrating with voice recognition. Two subjects said that it would be nice to have alternative selection mechanisms, since the current one made them "keep their mouth closed" and that they wanted to be able to talk while using the computer. Most subjects suggested that the general control of the cursor should be improved as well as the interface itself, e.g., to recognize the user's intention to type and not to move the cursor while the user is typing.

The results of the above exit survey showed that the general impression was positive. The problems such as adjustment to the interface were expected by the experimenter, and some of them were not present in the second session. For example, after the second session, a number of subjects asked the experimenter "Did something change in the meantime?", since they found the interface easier the second time. Since nothing had been changed in the interface we can conclude that the subject's perception of the interface changed: they got used to it, and didn't feel that strange while performing the tasks.

The above suggestions indicate that the subjects experienced the "Midas Touch" problem: their own body was the interface and in some cases they were simply not

able to adjus
the two sessi

7.5 Dis

The overall
ment in ti
improved
of the nu
or 4 tas
improv
the su

Th
interf
was 2
4.8 se
Subje
respec
noted
than w

In t
were tra

able to adjust their behavior and movements to incorporate the interface needs. Over the two sessions, most were able to adjust, however some could not adjust.

7.5 Discussion of T1, T2, T3, T5 Results

The overall performance in the individual tasks showed a trend of subject improvement in time. In task 1, 72% improved; in task 2, 83% improved; in task 3, 67% improved and in task 5, 61% improved. Overall improvement was measured in terms of the number of tasks in which the subject improved: when a subject improved in 3 or 4 tasks, by definition the overall performance improved. A total of 61% of subjects improved (28% in all the tasks, and 33% in 3 out of 4 tasks). The remaining 33% of the subjects improved in two tasks only, and 6% (1 subject) improved in only 1 task.

The total time the subject spent in performing the tasks with the head-eye input interface in the first session was on average 27.8 minutes and in the second session it was 24.6 minutes. The time needed to make the first correct selection was 4.6 and 4.8 seconds, for the tasks T2 and T3. The selection times were similar in Task T5. Subjects needed about 5, 10, and 6 times more time to perform tasks T2, T3, and T5, respectively, using the head-eye input interface compared to the mouse. It should be noted that a few subjects were able to perform Task T5 only about 2–3 times slower than with the conventional mouse.

In the second session, for subjects 01, 02, 07, 10, 12, and 13, three blue dots were tracked instead of the facial features. The tracking in the first session was not successful (f

cases, the pe

It should b

input interfac

and they ofte

move out of

task was b

experimen

focus mov

The

applicat

window

If we a

it to

conclu

not al

glitche

we wo

- It

ea

hig

successful (for children) or the subjects could not see well without glasses. In all six cases, the performance improved in the second session.

It should be noted that our child subjects had some difficulty in using the head-eye input interface. The problems were due to their impatient and rapid head movements, and they often moved a lot in the chair during the session. Thus, they would often move out of the focus of the camera and would have to re-position. The net-surfing task was boring for them and instead of asking them to perform the regular tasks the experimenter instructed them to visit appropriate WWW sites for their age and to focus more on the writing of e-mail messages to their parents.

The most important conclusion is that it was possible to control a real-world application without any changes and adjustments to the interface. The Netscape window we used, and the news WWW pages, had some rather small selection areas. If we are to make a custom-built application suited to the interface, we can adjust it to the special needs of the head-eye input interface. We believe that the same conclusions could be made for an eye-based interface, since our head and eyes are not always perfectly still, and due to our natural movements the interface is prone to glitches. To the future designers of applications based on the head-eye input interface, we would suggest the following:

- It is possible to select out of 10 × 12 icons on a 19" display. Selection time for each icon is about 3–5 seconds or less for an experienced user. This timing is higher than 1 second per selection, reported in many eye-mouse or head-mouse based st

users pr

- The reas

and simi

tons an

tions. a

"Back

subj

wer

"B

- It

t

-

c

s

- T

cu

wh

en

based studies, but we believe that the selection time could be reduced as the users practice more.

- The reasonable button size is about 1.4^o of viewing angle (e.g. Netscape "Back" and similar buttons). Our subjects had very little problem to select these buttons and they had to use them often. For example, for many erroneous selections, an unwanted WWW page would load and subjects had to click on the "Back" button, or a pop-up window with an error message would open and subjects had to click on an "OK" or similar button. In both cases, the buttons were of similar size. The number of erroneous selections while selecting the "Back" button was very low, compared to the selections of other buttons.

- It is possible to select smaller buttons (0.5^o), however, the error rates increase as the size decreases (e.g., adjacent menu buttons on a WWW page). When small targets are isolated, it is possible to select them with ease and small number of errors (e.g., the title of a story on a WWW page).

- There should be a blank space between the adjacent target buttons that are crucial for the control of the application. That would minimize the erroneous selections.

- The users must have a way to signal to the interface to temporarily stop the cursor control and have an easy way to re-start it. This option would be used when the user wishes to type some text and the cursor should stay on the text entry area regardless of where the user is actually looking.

- Selecti
  face ex
  subject
  keyboa

- If the i
  patter
  chair.
  cam
  the

7.5.1

As wit
The id
the us
to the
head
well, t
be abl

In c
the mo
the inte
explain e

- Selection with a face expression is possible but can cause some problems if the face expression determination algorithm is not working perfectly. When the subject's condition would allow, an alternative selection mechanism, such as keyboard or other hardware button, or voice signal, should be used.

- If the interface is to be used by children, it should incorporate their movement pattern while working with the computer: children may not sit straight in the chair, and might often lean towards the display, or to the chair sides. Thus, the camera should not be focused on their face only, but on the wide area around them. The interface should not be sensitive to their sudden moves.

7.5.1 Learning curve

As with all new interfaces, the use of the head-eye input interface must be learned. The idea of using one's head as an interface is not very familiar to the average user, and the user must constantly think about where and how to move the head. This is similar to the hand-eye coordination in the case of the conventional mouse. We can call this *"head-mind coordination"*. If the tracking and update of the screen coordinates works well, the users should be able to learn this coordination fast. Moreover, they should be able to adjust to the interface so that they do not think about the coordination.

In our interaction with novice users we often compared the new interface with the mouse. This metaphor helped the user to familiarize with the general idea of the interface. A positive aspect of this comparison was that we did not have to explain extensively what is the potential use of the interface, and how to use it, since most peopl

the perform

However, th

the same lev

mouse. The i

to use the m

with the w

after the

the need

the seco

that m

mouse

of the

not n

lab v

H

users

1. M

T

th

to

most people are nowadays familiar with the mouse. Also, we could easily compare the performance of the new interface and with the mouse on some common tasks. However, the first drawback of the comparison was that the users implicitly expected the same level of performance with the new interface that they were achieving with the mouse. The important fact most users were forgetting was that they had to *learn* how to use the mouse, and that there was some adjustment period before they were skilled with the mouse. Thus, some users had a number of complaints about the interface after the first usage, and most complaints addressed the issue that they do not see the need for the new interface when they can do much better with the mouse. After the second usage of the new interface, when the users got more skilled, there were not that many complaints and comparisons with the mouse. The second drawback of the mouse metaphor was that the users would not think about other potential applications of the interface, which extended well beyond the mouse capabilities. However, we did not notice that the mouse metaphor blocked other ideas, and many novice users and lab visitors would quickly point out other potential applications.

How well do the users learn the interface in practice? We observed a number of users while using the interface, and there is a common pattern for everyone.

1. *Novice level:* In the first few minutes of use all users perform rather poorly. They make rapid head movements and observe how the cursor would move. As the time passes, they get more skilled and learn how much they need to move to produce the desired cursor movement.

2. After

and th

have a

(a) E

wi

t

(b)

2. After the initial "warm-up" period, the users either get used to the interface and their performance improves, or they still struggle with the interface and have a hard time performing the tasks:

    (a) *Expert level:* If the users switch to this level, they can use the interface without any problems and they perform rather well, and we can compare their performance with the use of the conventional mouse.

    (b) *Intermediate level:* The users who stay at this level don't get used to the interface due to several reasons that range from the problems with the underlying computer vision algorithms to the problems related to the adjustment to the interface:

    - For some, the computer-vision-based tracking did not work well, and the users were not able to get good response from the interface. For example, users with glasses could not use them at present since the tracking would fail. Thus, when they would need to read some detailed information on the display they would lean closer or put on the glasses, which would result in erroneous updates of the cursor position. Then, they would need to re-position themselves in the chair and to re-calibrate, which would result in several seconds (or tens of seconds) of time lag, leading to user dissatisfaction. For such users, if the tracking problem is solved, they can quickly adjust to the interface and switch to the "expert level". For example, to overcome the glasses problem, we tracked three blue squares glued just above the eyebrows

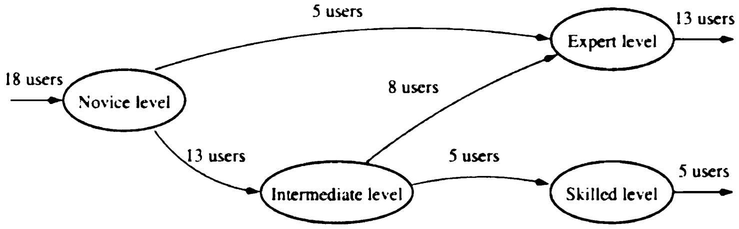


Figure 7.13: Learning levels for 18 subjects

and on the tip of the nose. The movements of the blue spots were equivalent to the eyes-nose movements and the users had the feedback as if the tracking worked well.

- *Skilled level:* The tracking works, but they are not able to learn to move slowly and to re-calibrate when needed. Those users would have good cursor control at one point, but would often tilt too much, and an eye would be occluded, or the mouth would not be visible and the interface would not respond. They would not realize what is happening, and when instructed to move slowly and re-calibrate, they would start to rapidly move their head in all directions and would think that they did re-calibrate, but in reality they would just cause more confusion to the interface. Either they did not understand the instructions to move slowly or did not want to follow the instructions. In both cases, the results would be that they would not be able to control the cursor movement at all. In time, they would gradually start moving slower and be able to control the cursor. However, they would never get even close to the performance of the expert users.

The obs

puter input

pick it up ri

the head-ey

two one-hour

tasks using

in skill leve

level and

based on

7.6

In th

wide

curso

writi

one w

we co

repeat

were al

to the

to comp

The observed learning pattern is similar to the learning pattern of any new computer input device. Some users will simply never adjust to the interface and some will pick it up right away. In our experiments, we attempted to test the learning curve for the head-eye input interface. Our subjects attended an experiment that consisted of two one-hour sessions, approximately one week apart, where they performed various tasks using the head-eye input device. Figure 7.13 shows the breakup of the subjects in skill levels. A total of 18 subjects participated in the experiment; 13 reached expert level and 5 reached skilled level. The breakup in the skill levels is done by the author based on the observance of the experiments and the performance results.

7.6 Summary

In this Chapter we presented the results of a study with 18 subjects who performed a wide range of tasks using our head-eye input interface. The tasks involved moving the cursor along a path, selecting buttons, dragging icons, and surfing the internet and writing an e-mail message. All subjects participated in two sessions approximately one week apart. The performance on each task was measured and based on that we concluded that 11 out of 18 subjects (61%) improved their performance in the repeated usage of the interface. The most important fact was that most subjects were able to control the real-world application such as Netscape without any change to the application, and that in the second session only one subject was not able to complete the task. We believe that if our subjects persisted using the interface, they would

comparable

they would get to be skilled users and their performance with the interface could be comparable with performance using other input devices (e.g., the mouse).

Chap

Concl

In this dissert
sented. The i
user's head an
completely no
sults of featur
that the featu
pixel image.
tificial Neural
movements w
ranging from
head moveme
experiments w

The gaze d
(open vs. close

Chapter 8

Conclusion and Future Work

In this dissertation, an interface for handless human-computer interaction was presented. The interface is based on the use of computer vision algorithms to track a user's head and facial features such as the eyes, nose and eyebrows. The interface is completely non-intrusive, and requires only an camera attached to the computer. Results of feature tracking were presented and accuracy was evaluated, and it was shown that the features could be tracked on average with 2 pixels accuracy in a 320 × 240 pixel image. A framework for gaze direction detection based on the use of an Artificial Neural Network and feature image coordinates movement scaling to display movements was presented. The framework can be used in a number of applications ranging from measuring user's focus of attention to controlling the computer using head movements. Preliminary results of attention measurement and cursor control experiments were presented.

The gaze detection algorithm was used along with the face expression detection (open vs. closed mouth) to develop a handless human-computer interaction interface.

The interfa
week apart.
button *num*
that the use
was selectab
number of a
movement p
selecting butt
Our subjects
They spent in
showed impro

8.1 Re

- *A facial*
  geometr
  using ge
  compone
  successful
  If the per
  does not w
- An algorit
  a dark-skin

The interface was evaluated by 18 novice users in two sessions approximately one week apart. In terms of the usability of the interface, the appropriate selectable button number, size and configuration were investigated. Our experiments showed that the users can select buttons of 0.5^o with a lot of errors, while button size of 1.3^o was selectable with reasonable error rates. The selection time was 3–5 seconds. A number of applications of the interface were evaluated: monitoring user's gaze and movement pattern, moving the cursor along a predetermined path on the display, selecting buttons, dragging icons, and controlling a real-world application (Netscape). Our subjects learned how to successfully use the interface after moderate training. They spent in total about 1 hour in active use of the interface and most of the subjects showed improvment in the performance.

8.1 Research Contributions

- *A facial feature location* method was presented based on the knowledge of the geometry of the face. The method searches for the eyes, eyebrows and nose using geometrical constraints, and the knowledge of the face colors. The red component of the color image is used to search for eyes. The method was successfully tested on a number of faces in both still images and video sequences. If the person wears glasses or has beard or mustaches, the method sometimes does not work well.

- An algorithm for *detecting dark skin* and a method for *adapting the image of a dark-skin person* so that the skin color detection algorithm can find the dark skin

moti

- *Gaze*

initial

play/c

plied t

flexibil

head m

- *Monito*

studies

offer n

presen

- *Evalua*

user le

18 subj

can get

are har

selected

rather c

was 3–5 s

skin was presented. The algorithm works automatically in conjunction with the motion detection, and automatically finds the adjustment coefficients.

- *Gaze direction detection* that requires no calibration was presented. The user's initial gaze is detected using an Artificial Neural Network, and then the display/control gain factor logic, used for pointing devices like the mouse, is applied for the gaze calculation. This method offers adjustable gain factor, and flexibility for individual users. The user could set-up the interface so that the head movements are rather small.

- *Monitoring user's gaze* in time enables applications such as visual perception studies, attention measurement, or marketing research. This interface could offer non-intrusive monitoring in a more natural setting than what is done at present.

- *Evaluation of the interface* in terms of the usability, achievable resolution, and user learning pace and performance was presented. The results of a study with 18 subjects, who performed a wide range of tasks, show how well novice subjects can get used to the interface, which tasks can be performed with ease and which are hard. We showed that a target as small as 0.36^o of viewing angle can be selected with the interface, and that a target size of 1.4^o of viewing angle was rather comfortable for the users to select. The time needed to make a selection was 3–5 seconds.

- *Succe*
withc
and m
to lea
point
an e-m
averag
showe
author
comple
and in
achieve

8.2 Fu

The interface
this work, we
directions in t
will one day be

- Improven
the use of
would allo

- *Successful control of a real-world application* using the head-eye input interface without any changes of the application itself was achieved. Netscape navigator and mail programs were used as the test programs and our subjects were able to learn how to use the system and to surf the network using their head as a pointer. They were able to browse several WWW sites successfully and to write an e-mail message, after moderate training with the interface of, 28 minutes average in the first session, and 16 minutes in the second session. The results showed that they would be able to become very skilled in the interface. The author of the interface, who can be considered an expert user, for example, completed the netscape surfing task in 2:40 minutes using the head-eye input, and in 1:15 minutes using the conventional mouse. Five other subjects (28%) achieved performance approaching the author's.

8.2 Future Work

The interface presented in this dissertation can be used in various applications. In this work, we investigated some basic tasks. In this section we list the possible future directions in the research of the head-eye input interface. We hope that the interface will one day become widely used and will improve human interaction with computers.

- **Improvement** of the feature detecting algorithm in terms of the robustness to the use of glasses, and presence of beard or mustaches. Such an improvement would allow wide range of users to use the interface without any constraints.

- Use o
featur
way *t*
monit
Altern
and an
Applic

- Improv
on optic
user co
fashion

- Evalua
of some
users w
benefit

- Improve
such as
conditio
any envi

- Use of more than one video camera, that would all attempt to find the tracked features. The best results would be used as the output of the system. That way the user would be able to move freely in the workspace and the computer monitor would not need to be the only device that would be controlled.

  Alternatively, an omnidirectional camera could locate the faces and track them, and another pan-tilt-zoom camera could zoom in to get a better face picture. Applications such as teleconferencing would benefit greatly.

- Improvement of the UI to have an easily selectable turn-off and turn-back-on option, that would be used when the user is just typing or reading. The user could then use the interface along with other input interfaces in an easier fashion.

- Evaluation of the interface for physically challenged users, and customization of some procedures, depending on the special user needs. We believe that the users whose only way to control the computer is through head motion would benefit greatly.

- Improvement of the algorithms in terms of the robustness to the environment, such as presence of skin-color tones in the background or changing lighting conditions. If a fully robust algorithm is devised the interface could be used in any environment, e.g., airports, museums, train stations, etc.

APPENDICES

Ap

Res

Exp

In this
perimer
of the
3 differe
been de
graph us
the rema

Table
duration
tion point
the image

Appendix A

Results of the Gaze Monitoring Experiment

In this Appendix, the results of the gaze monitoring and fixations determination experiment are included. The experiment has been conducted adjunct to the evaluation of the head-eye input device described in Chapter 7. A total of 18 subjects viewed 3 different images for 30 seconds each, and the gaze path and fixation locations have been determined automatically using the algorithm in Figure 5.6. The first photograph used is from *"50 Favorite Rooms by Frank Lloyd Wright" by Diane Maddex*, and the remaining two photographs are from the WWW site `http://www.photo.net/`.

Table A.1 includes number of fixations, average, minimum and maximum fixation duration for each of 18 subjects. Figures A.1 through A.9 give gaze path and fixation points for each of three images. Gaze path and fixations are superimposed on the image that the subject viewed. The white line represents the gaze path. Gray

subj ID
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2015
2016
2017
2018

Table A.
algorithm

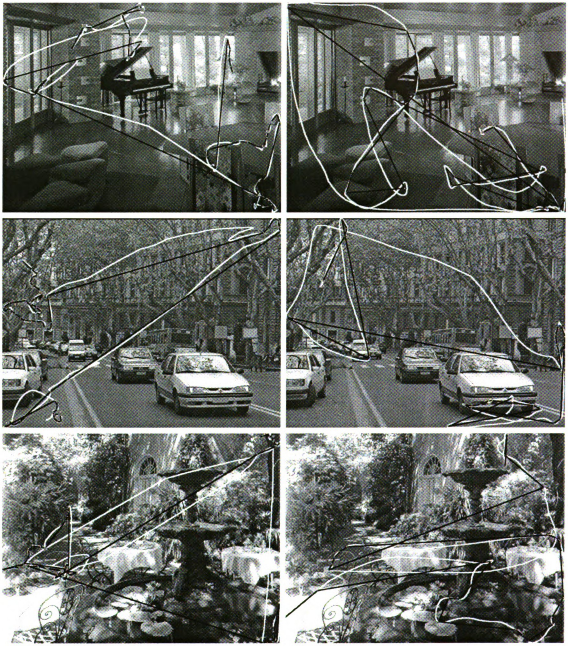
dots show

connects

subject	# of fixations, avg., min. & max. duration (msec)											
ID	image 0				image 1				image 2			
2000	53	864	112	3800	57	734	115	2401	54	789	140	2991
2001	23	558	123	1690	27	447	115	1255	26	310	102	851
2002	23	361	117	1544	16	580	117	2669	17	341	107	939
2003	31	547	116	2394	27	300	121	623	31	316	115	839
2004	31	398	109	2097	44	433	108	1438	40	411	111	1306
2005	18	218	100	438	24	284	104	1083	31	252	30	761
2006	30	441	102	1386	11	239	109	565	31	291	105	882
2007	34	455	79	2022	31	378	117	1150	13	268	116	876
2008	39	490	102	1729	36	286	101	662	38	430	126	1426
2009	37	322	101	963	33	345	100	1181	27	238	97	595
2010	19	964	236	3118	28	783	220	4073	20	571	123	1535
2011	46	476	109	1251	45	544	110	1504	50	413	72	1727
2012	19	294	110	885	33	234	41	573	23	250	100	660
2013	18	364	117	816	35	311	104	1042	27	339	102	922
2015	41	510	102	1529	24	390	122	1015	23	356	125	1471
2016	40	509	100	1376	31	354	115	1167	35	469	117	1905
2017	24	257	110	835	13	226	104	593	41	312	101	943
2018	32	761	108	2711	43	461	105	1305	35	465	102	1670

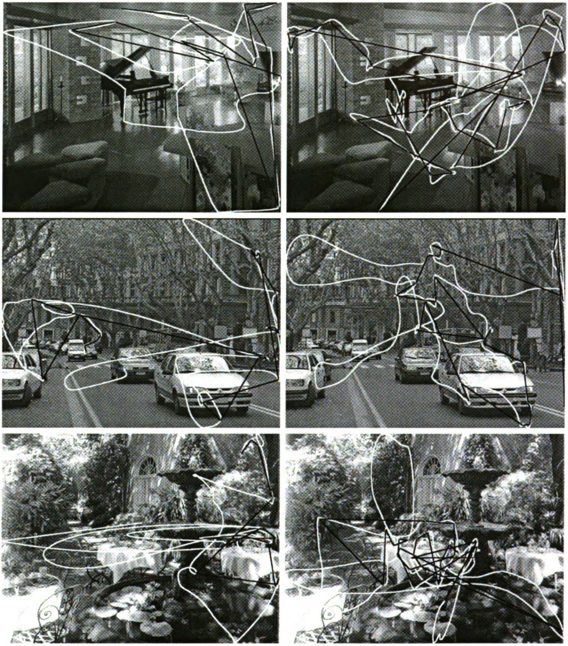
Table A.1: Number and durations of fixations measured by the gaze-determination algorithm

dots show fixation locations, and the dot size indicates fixation duration. Black line connects consecutive fixation locations.



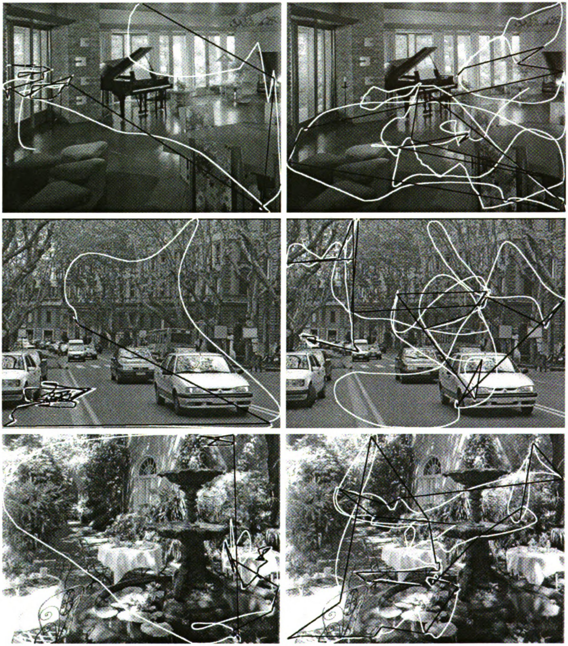
subj-2000 subj-2001

Figure A.1: Gaze path and fixation points, subj-2000 and subj-2001, images 0, 1, 2



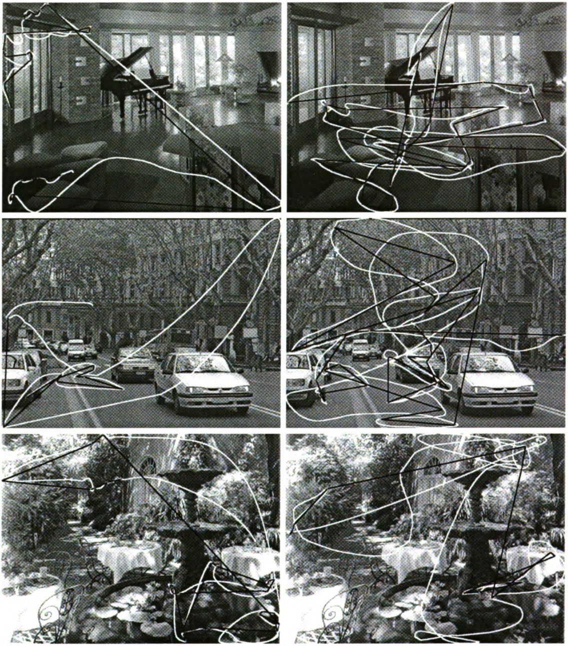
subj-2002 subj-2003

Figure A.2: Gaze path and fixation points, subj-2002 and subj-2003, images 0, 1, 2



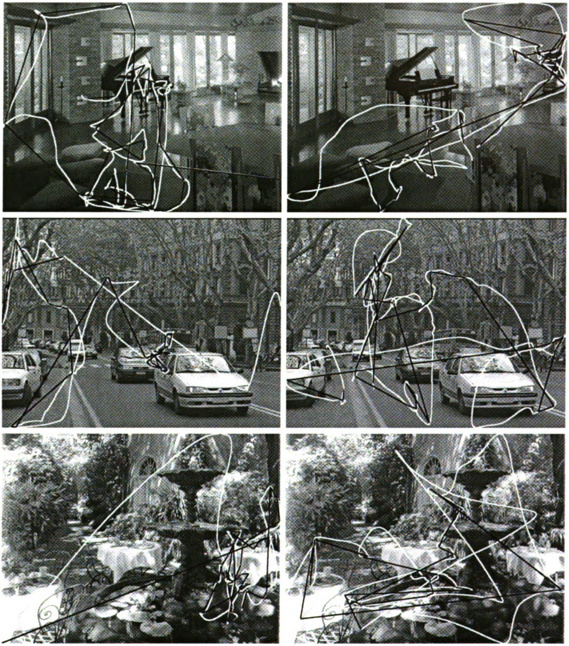
subj-2004 subj-2005

Figure A.3: Gaze path and fixation points, subj-2004 and subj-2005, images 0, 1, 2



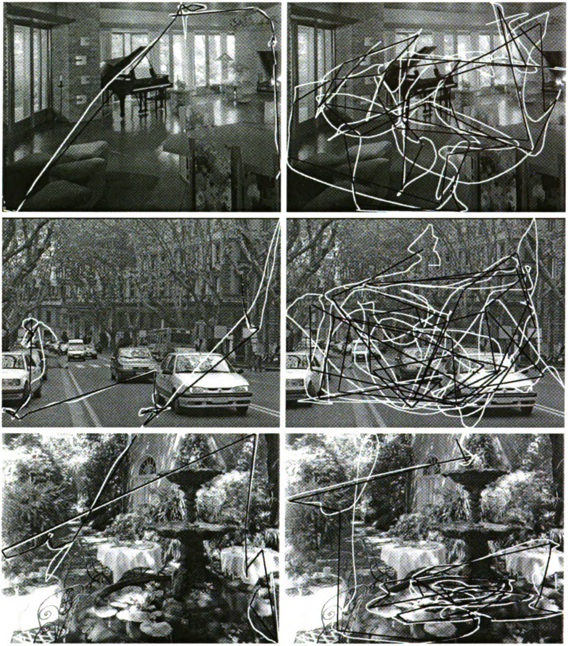
subj-2006 subj-2007

Figure A.4: Gaze path and fixation points, subj-2006 and subj-2007, images 0, 1, 2



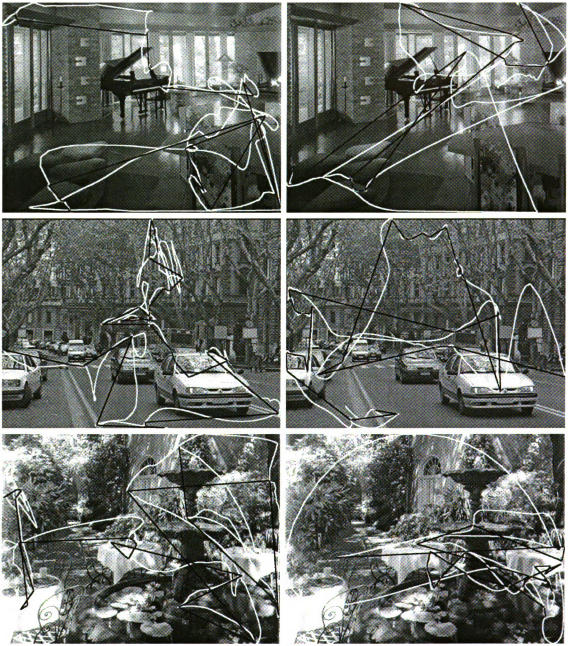
subj-2008 subj-2009

Figure A.5: Gaze path and fixation points, subj-2008 and subj-2009, images 0, 1, 2



subj-2010 subj-2011

Figure A.6: Gaze path and fixation points, subj-2010 and subj-2011, images 0, 1, 2



subj-2012 subj-2013

Figure A.7: Gaze path and fixation points, subj-2012 and subj-2013, images 0, 1, 2

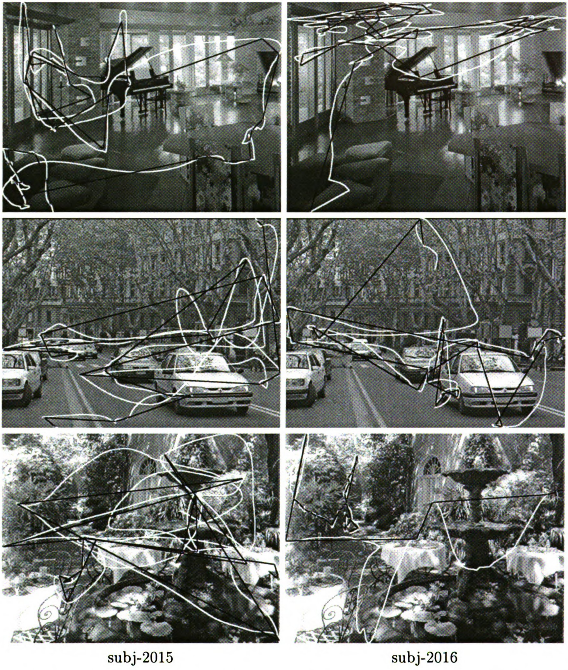
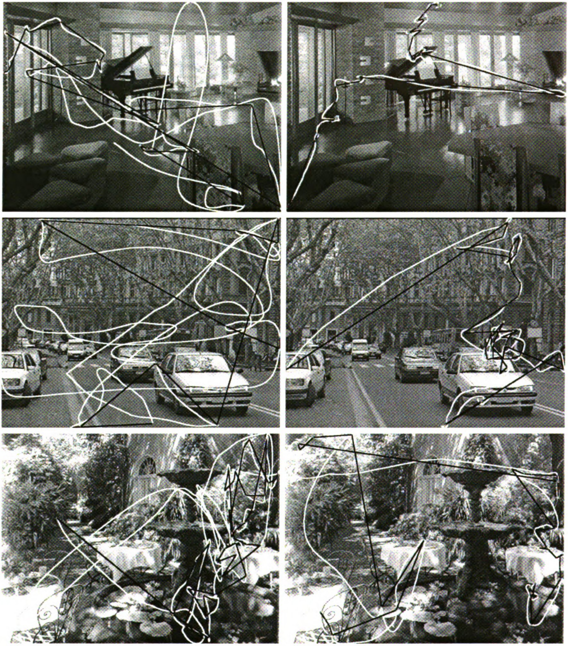


Figure A.8: Gaze path and fixation points, subj-2015 and subj-2016, images 0, 1, 2



subj-2017 subj-2018

Figure A.9: Gaze path and fixation points, subj-2017 and subj-2018, mages 0, 1, 2

Appendix B

Task T1 Complete Results

Table B.1 gives the squared error from the target curve for each subject and curve, for the two sessions done one week appart. The screen coordinates were normalized to 0–1 range. Figures B.1 through B.19 give the comparison of the target curve and the cursor path shapes in the two sessions, for each subject.

Subj.	Session 1, averaged squared error					Session 2, averaged squared error				
ID	c0	c1	c2	sum	#pnt	c0	c1	c2	sum	#pnt
00	0.0787	0.0447	0.0578	0.1812	3563	0.0325	0.0266	0.0569	0.1160	4139
01	0.0579	0.0349	0.0594	0.1522	3213	0.0360	0.0456	0.0588	0.1404	2778
02	0.1053	0.1248	0.0708	0.3009	2986	0.0886	0.0767	0.0776	0.2429	3257
03	0.0620	0.0577	0.0596	0.1793	4279	0.0781	0.0622	0.0541	0.1944	3420
04	0.0491	0.0321	0.0770	0.1582	3571	0.0705	0.0587	0.0639	0.1931	2884
05	0.1000	0.1514	0.0557	0.3071‡	3064	0.1081	0.0783	0.0603	0.2467	2197
06	0.0627	0.0493	0.0840	0.1960	3809	0.0414	0.0766	0.0575	0.1755	2644
07	0.0695	0.1337	0.0679	0.2729	3205	0.0679	0.1226	0.0603	0.2508	1546
08	0.0778	0.0691	0.0795	0.2264	3314	0.0616	0.0574	0.0637	0.1827	3786
09	0.0344	0.0467	0.0553	0.1364†	4356	0.0345	0.0311	0.0443	0.1099	3411
10	0.1501	0.0831	0.0683	0.3015	2674	0.0872	0.0689	0.0588	0.2149	2722
11	0.0718	0.0818	0.0690	0.2226	3204	0.0495	0.0422	0.0601	0.1518	3980
12	0.0902	0.1366	0.0553	0.2821	3098	0.0254	0.0263	0.0566	0.1083†	3200
13	0.0612	0.0477	0.0565	0.1654	3924	0.0670	0.1054	0.0724	0.2448	2444
15	0.0363	0.0377	0.0661	0.1401	3520	0.0420	0.0412	0.0474	0.1306	2346
16	0.0563	0.0439	0.0705	0.1707	4908	0.1313	0.0825	0.0719	0.2857‡	2615
17	0.0672	0.0712	0.0622	0.2006	4159	0.0485	0.0994	0.0574	0.2053	2755
18	0.0715	0.0926	0.0596	0.2237	3242	0.0377	0.0700	0.0586	0.1663	3073
20	subject is the author					0.0281	0.0213	0.0584	0.1078	3807
Sum	1.3020	1.3390	1.1745	3.8173	64089	1.1078	1.1717	1.0806	3.3601	53197
Avg.	0.0723	0.0744	0.0653	0.2121	3560	0.0615	0.0651	0.0600	0.1867	2955
Percentage of improvement:						15%	12%	8%	12%	

† - minimal overall squared error value in each session

‡ - maximal overall squared error value in each session

Table B.1: Task T1: average squared error for each subject

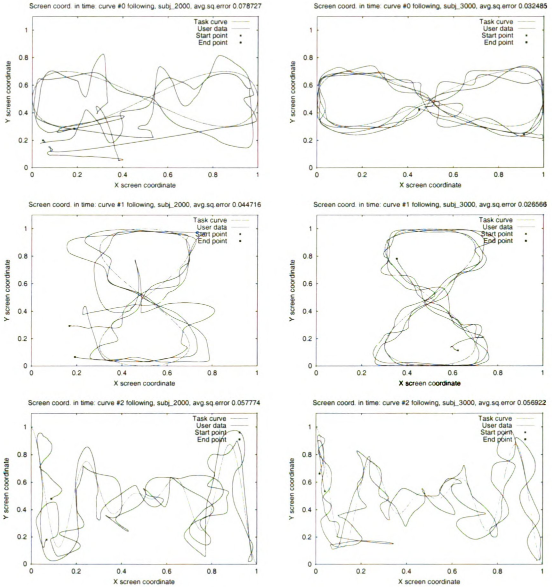


Figure B.1: Task T1, subject 00, cursor paths for all three curves

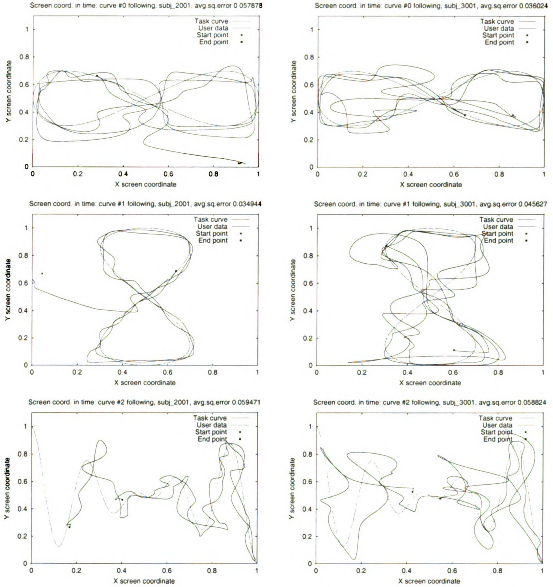


Figure B.2: Task T1, subject 01, cursor paths for all three curves

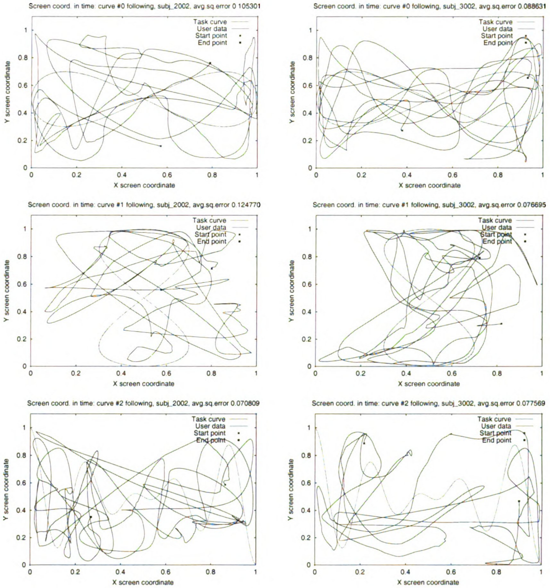


Figure B.3: Task T1, subject 02, cursor paths for all three curves

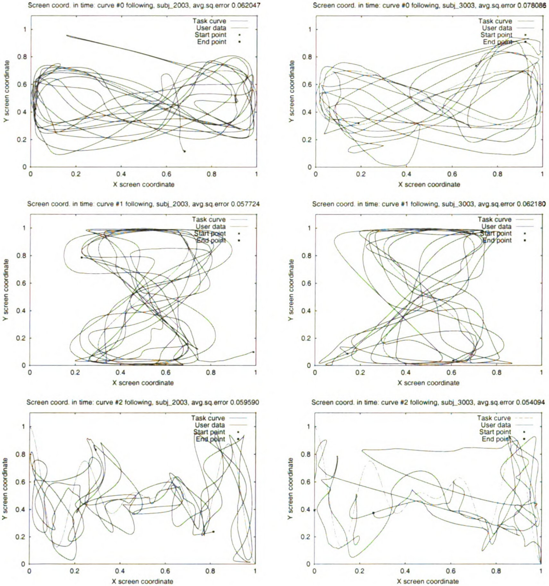


Figure B.4: Task T1, subject 03, cursor paths for all three curves

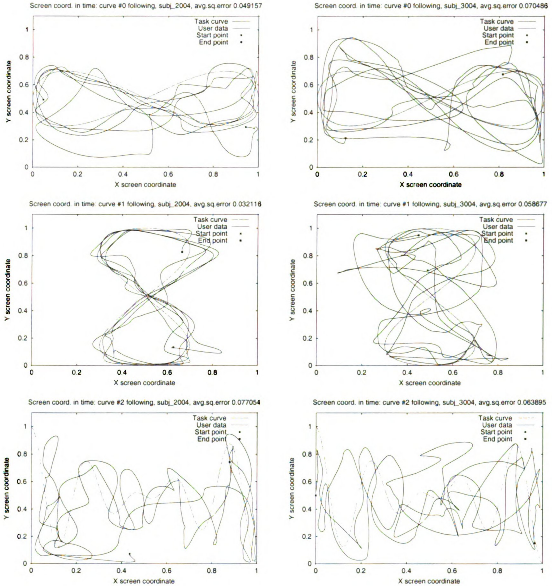


Figure B.5: Task T1, subject 04, cursor paths for all three curves

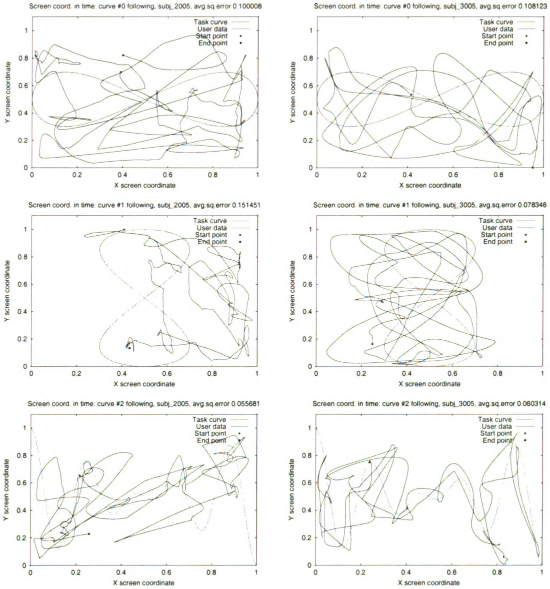
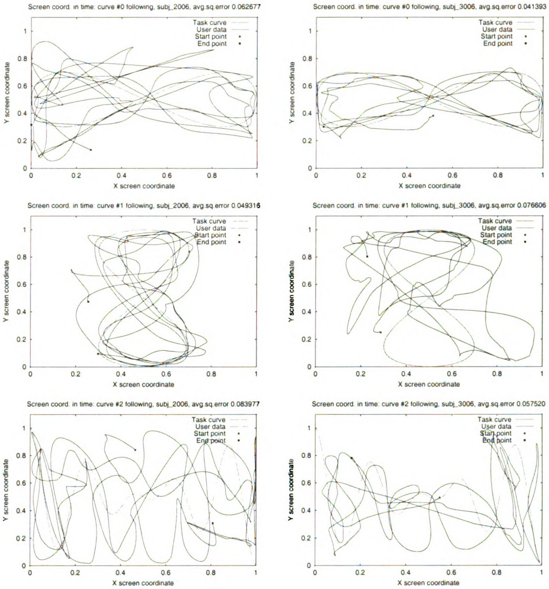


Figure B.6: Task T1, subject 05, cursor paths for all three curves


Screen coord. in time: curve #0 following, subj_2006, avg.sq.error 0.062677

Screen coord. in time: curve #0 following, subj_3006, avg.sq.error 0.041393

Screen coord. in time: curve #1 following, subj_2006, avg.sq.error 0.049316

Screen coord. in time: curve #1 following, subj_3006, avg.sq.error 0.076606

Screen coord. in time: curve #2 following, subj_2006, avg.sq.error 0.083977

Screen coord. in time: curve #2 following, subj_3006, avg.sq.error 0.057520

Figure B.7: Task T1, subject 06, cursor paths for all three curves

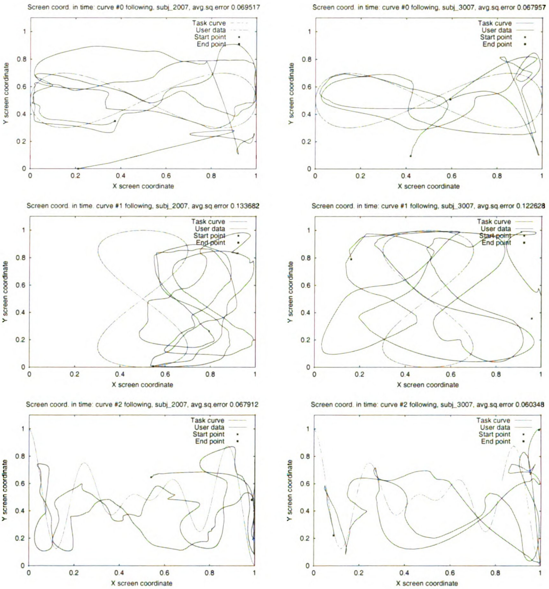


Figure B.8: Task T1, subject 07, cursor paths for all three curves

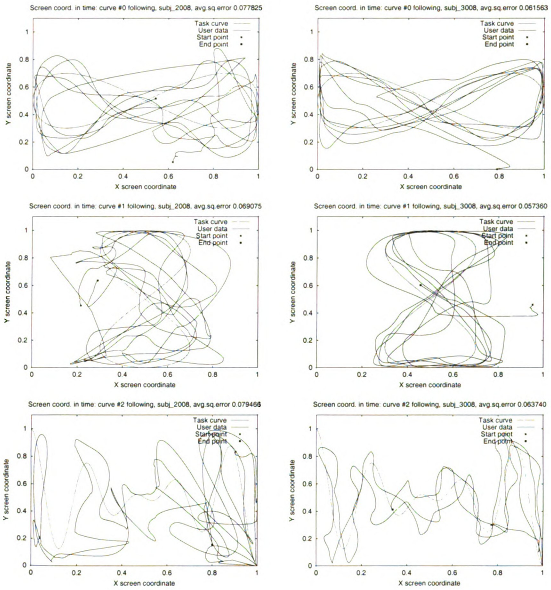


Figure B.9: Task T1, subject 08, cursor paths for all three curves

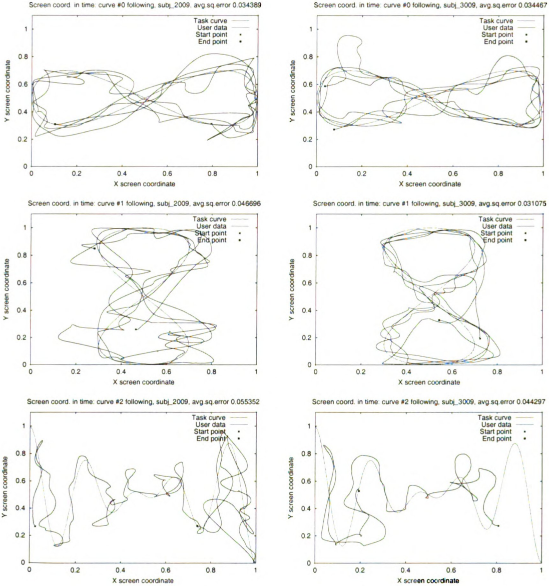


Figure B.10: Task T1, subject 09, cursor paths for all three curves

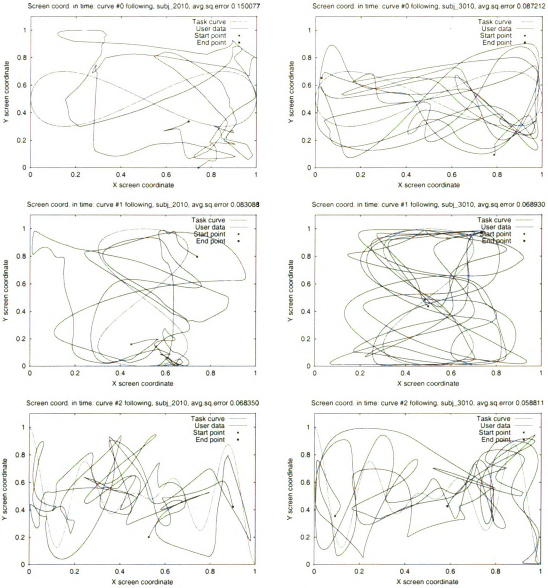


Figure B.11: Task T1, subject 10, cursor paths for all three curves

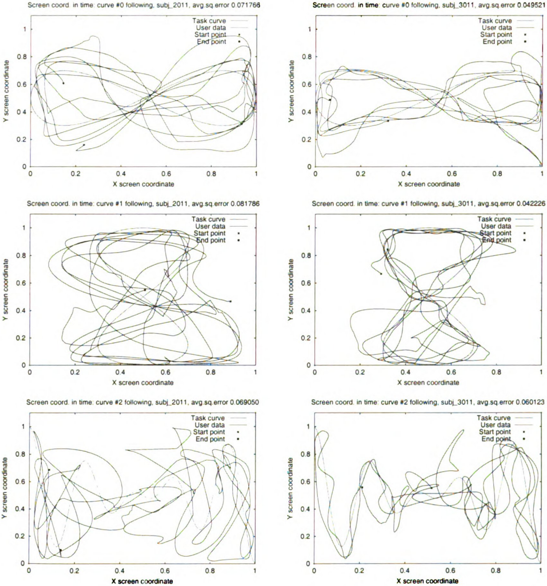


Figure B.12: Task T1, subject 11, cursor paths for all three curves

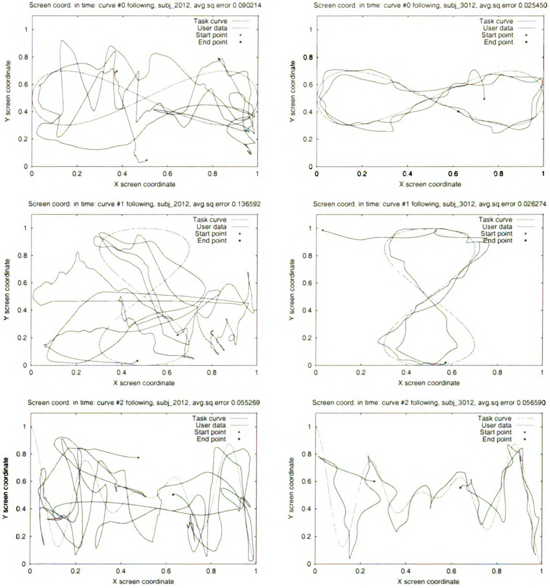
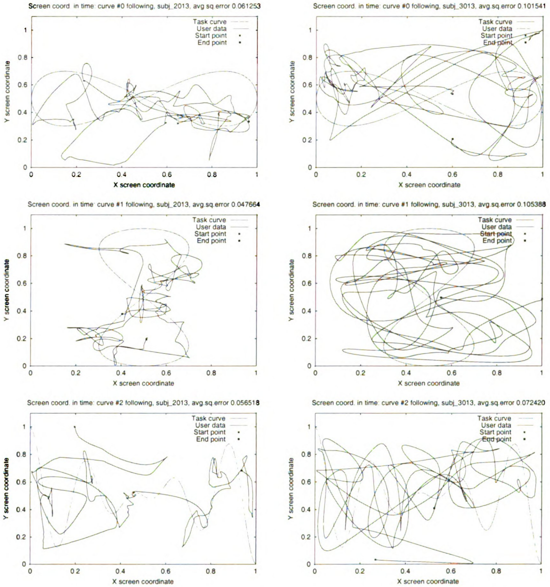
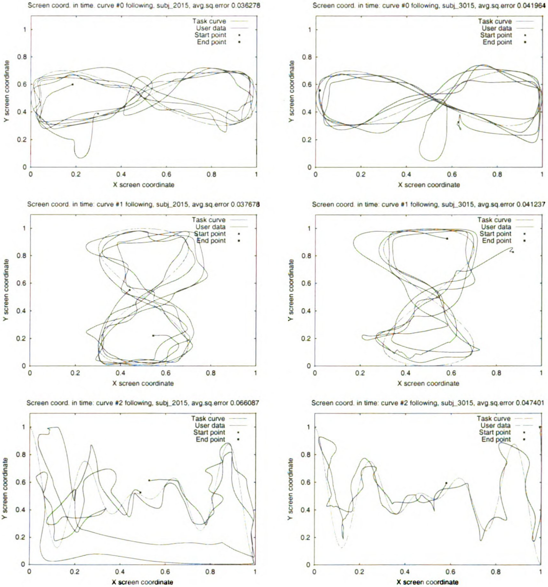


Figure B.13: Task T1, subject 12, cursor paths for all three curves





Screen coord. in time: curve #0 following, subj_2015, avg.sq.error 0.036278

Screen coord. in time: curve #0 following, subj_3015, avg.sq.error 0.041964

Screen coord. in time: curve #1 following, subj_2015, avg.sq.error 0.037678

Screen coord. in time: curve #1 following, subj_3015, avg.sq.error 0.041237

Screen coord. in time: curve #2 following, subj_2015, avg.sq.error 0.066087

Screen coord. in time: curve #2 following, subj_3015, avg.sq.error 0.047401

Figure B.15: Task T1, subject 15, cursor paths for all three curves

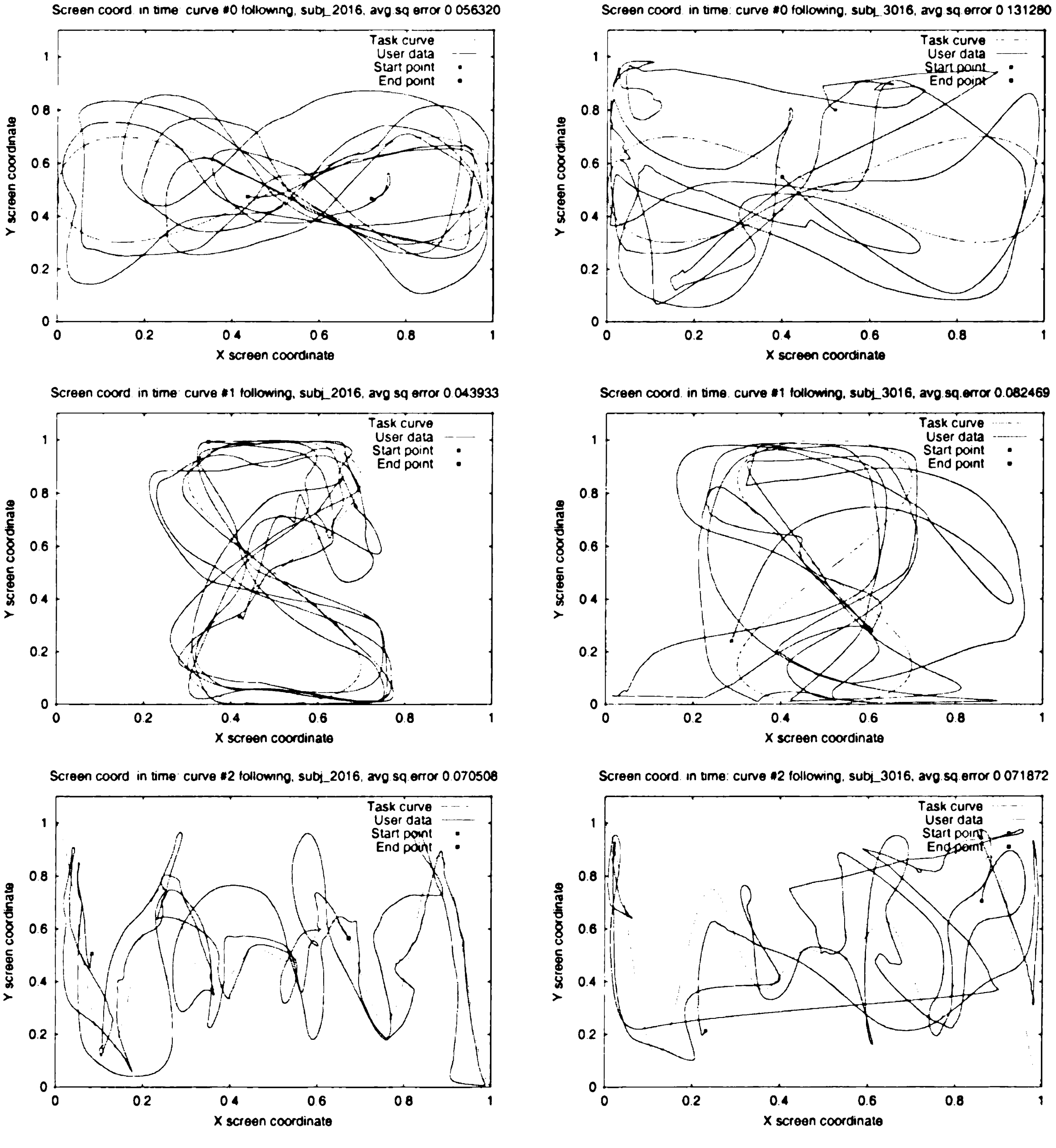


Figure B.16: Task T1, subject 16, cursor paths for all three curves

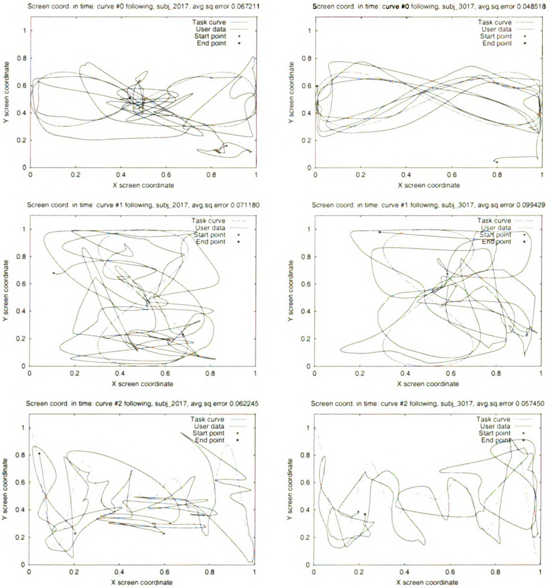


Figure B.17: Task T1, subject 17, cursor paths for all three curves

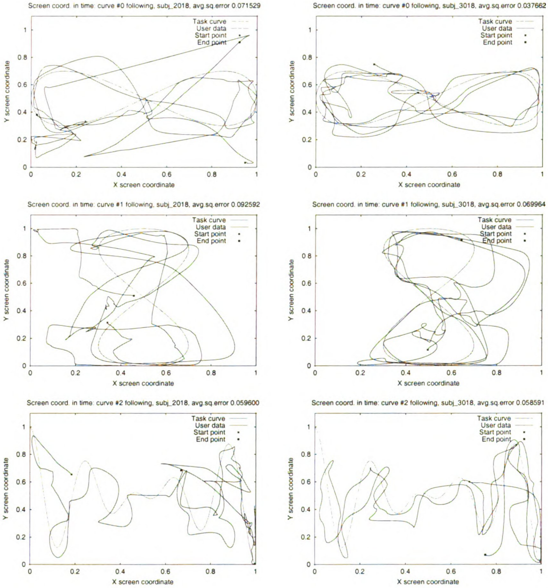
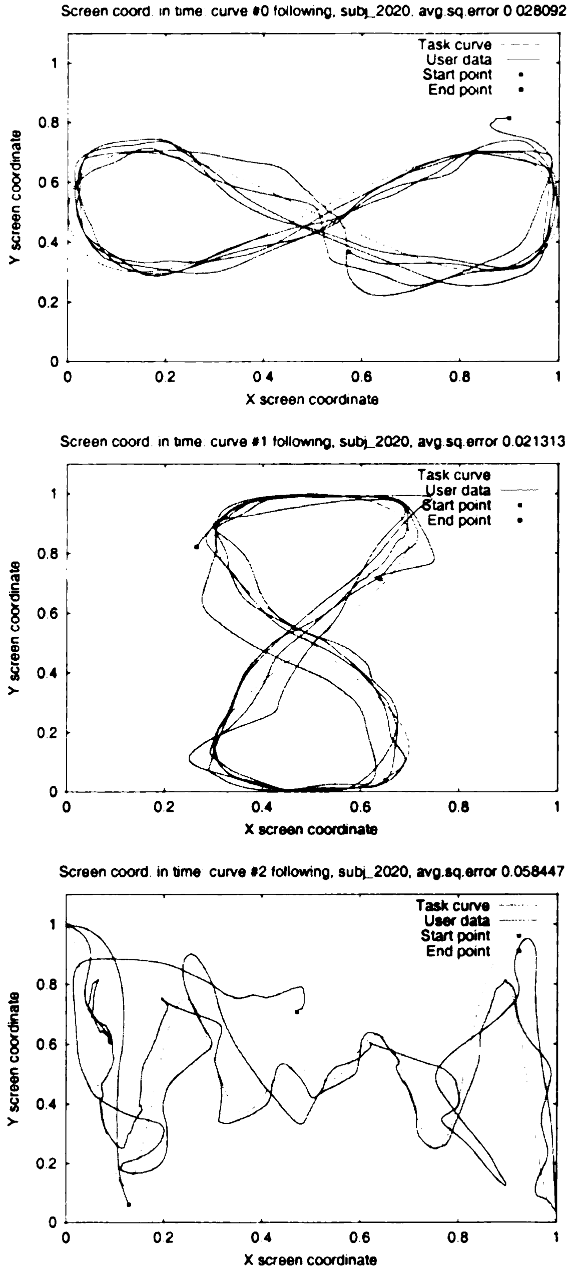


Figure B.18: Task T1, subject 18, cursor paths for all three curves



Screen coord. in time: curve #0 following, subj_2020, avg.sq.error 0 028092

Screen coord. in time: curve #1 following, subj_2020, avg.sq.error 0.021313

Screen coord. in time: curve #2 following, subj_2020, avg.sq.error 0.058447

Figure B.19: Task T1, subject 20, cursor paths for all three curves

Appendix C

Task T2 Complete Results

Table C.1 gives the button selection accuracy. For each session, the first three columns correspond to the first, second and third correct selection. The fourth column gives the number of incorrectly selected buttons in the third attempt. The remaining three columns give the average squared error from the linear regression line by Fitts' law.

Table C.2 gives the selection time statistics. The first four columns for each session give the average total time needed to select a button in the first, second, and third attempt, and the time when after the third attempt the selection was incorrect. The average time generally increases with the number of attempts, which is expected, since the user needs to make more than one selection. Also shown in the Table are total elapsed times for each trial and for all the three trials. As the trials progress, the elapsed times didn't decrease. However, when compared across the sessions, both the individual trial times and the total time decreased significantly.

Figures C.1 through C.19 give the selection times vs. button distance by Fitts' law and the cursor paths for each target button for all 18 subjects and the author.

	Session 1							Session 2						
	Selection				Avg.sq.err of lin.			Selection				Avg.sq.err of lin.		
Subj	correctness				regression line			correctness				regression line		
ID	1	2	3	not	E0	E1–2	M	1	2	3	not	E0	T1–2	M
00	4	3	2	6	1035	951	-	23	4	3	0	521	340	32
01	18	6	3	3	628	661	57	23	6	0	1	391	296	33
02	7	6	2	15	1108	574	36	23	1	1	5	825	216	49
03	20	7	2	1	408	443	105	27	2	1	0	1292	1474	318
04	13	5	5	7	812	558	39	19	2	2	7	315	335	37
05	23	1	4	2	1460	561	110	8	5	5	12	798	518	38
06	9	5	6	10	318	794	216	18	7	4	1	1527	452	47
07	8	12	5	5	364	477	468	18	6	4	2	363	364	108
08	8	12	5	5	1095	588	31	22	3	4	1	192	875	25
09	18	4	4	3	2738	1916	244	24	6	0	0	314	784	68
10	9	6	5	10	384	353	79	15	10	3	2	1036	1794	60
11	21	7	1	1	235	353	21	24	5	1	0	563	262	17
12	2	7	8	13	397	490	219	25	3	2	0	562	220	80
13	2	5	3	20	850	1089	52	4	5	1	20	170	711	112
15	20	6	2	2	666	303	27	23	1	1	5	148	1462	68
16	7	5	8	10	190	383	53	15	6	4	5	370	324	55
17	9	11	6	4	797	200	41	7	4	2	17	97	1206	302
18	19	7	3	1	400	1277	36	20	2	2	6	1001	217	31
20	subject is the author							28	0	1	1	128	435	34
Avg	12.1	6.4	4.1	6.6				18.8	4.3	2.2	4.7			
%	41	22	14	23				63	14	7	16			
	77% correct, 23% wrong							84% correct, 16% wrong						

Table C.1: Task T2: button selection accuracy statistics

	Session 1								Session 2							
Subj	Avg. time for selection (sec)				Trial time (sec)			Tot. time (sec)	Avg. time for selection (sec)				Trial time (sec)			Tot. time (sec)
ID	1	2	3	not	T0	T1	T2		1	2	3	not	T0	T1	T2	
00	6.6	12.1	14.9	14.3	76	68	33	178	4.4	8.0	4.4	-	52	54	39	145
01	6.3	8.0	6.6	5.2	71	79	46	197	3.4	6.5	-	5.9	45	36	42	124
02	8.5	14.4	11.1	11.9	111	142	92	346	4.5	11.0	4.1	1.5	58	33	33	125
03	4.6	9.3	14.3	6.3	63	63	64	190	6.4	5.6	5.7	-	45	56	86	189
04	7.5	11.1	11.1	7.9	85	73	104	264	5.8	7.5	8.9	2.6	41	60	59	161
05	7.1	10.7	11.7	8.2	102	46	87	236	3.7	5.5	10.8	10.0	76	110	44	231
06	4.1	7.3	13.7	8.6	40	87	114	242	5.8	11.9	16.2	8.0	116	79	64	260
07	5.0	7.3	7.8	5.7	58	72	65	195	4.6	6.8	9.1	9.9	59	42	77	180
08	4.1	4.9	8.9	23.7	118	87	49	254	4.7	14.8	11.3	1.5	40	56	96	194
09	13.6	17.8	42.6	6.4	196	169	139	505	4.1	4.2	-	-	32	35	55	123
10	3.3	5.7	5.3	9.1	53	43	83	181	5.7	7.4	41.8	30.3	80	174	91	346
11	2.9	5.9	5.7	5.1	31	35	46	112	3.6	8.1	7.7	-	53	47	34	135
12	3.5	5.7	12.5	6.0	80	88	56	225	3.8	6.2	4.9	-	51	34	38	124
13	30.4	13.2	31.7	11.8	176	224	55	457	6.1	7.0	39.5	2.3	28	95	20	145
15	4.4	5.4	6.3	11.4	65	44	45	155	7.3	4.6	2.2	3.4	32	98	60	190
16	3.2	4.1	6.1	7.2	33	43	86	163	4.4	3.8	6.2	3.0	39	58	30	128
17	4.2	5.3	5.7	6.3	79	35	40	155	5.1	3.1	5.5	5.0	20	20	103	144
18	7.4	7.3	10.6	3.3	46	83	97	227	5.4	3.1	4.4	2.1	53	47	34	135
20	subject is the author								3.2	0.0	3.2	2.1	22	37	34	94
Avg	7.0	8.6	12.6	8.8	82	82	72	238	4.9	6.9	10.1	4.7	51	59	58	164

Table C.2: Task T2: button selection timing statistics

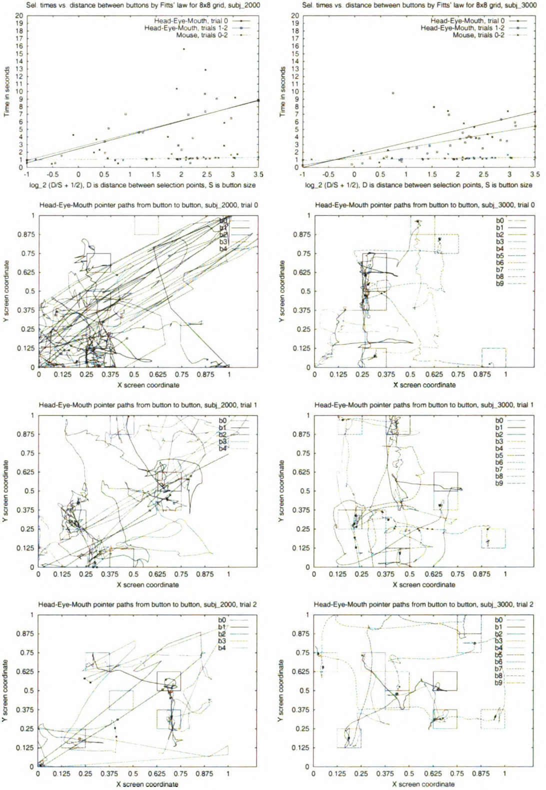


Figure C.1: Task T2, subject 00, Fitts' law time-distance plots and cursor paths for each button

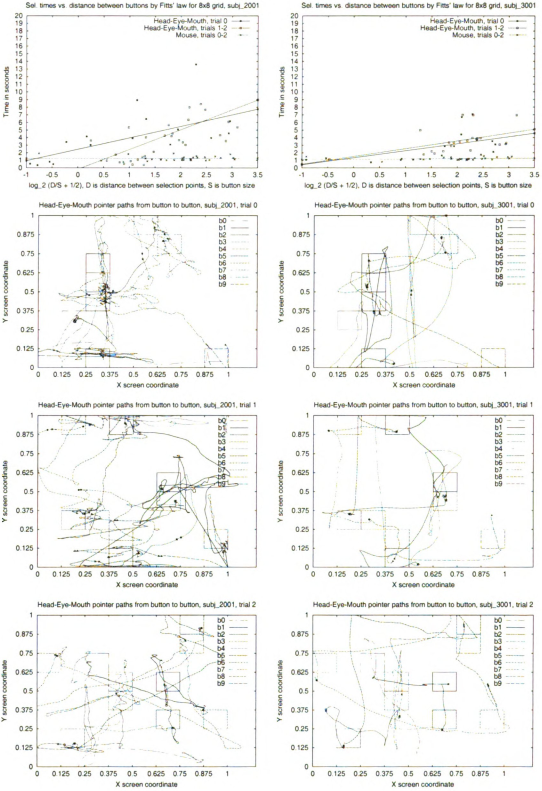


Figure C.2: Task T2, subject 01, Fitts' law time-distance plots and cursor paths for each button

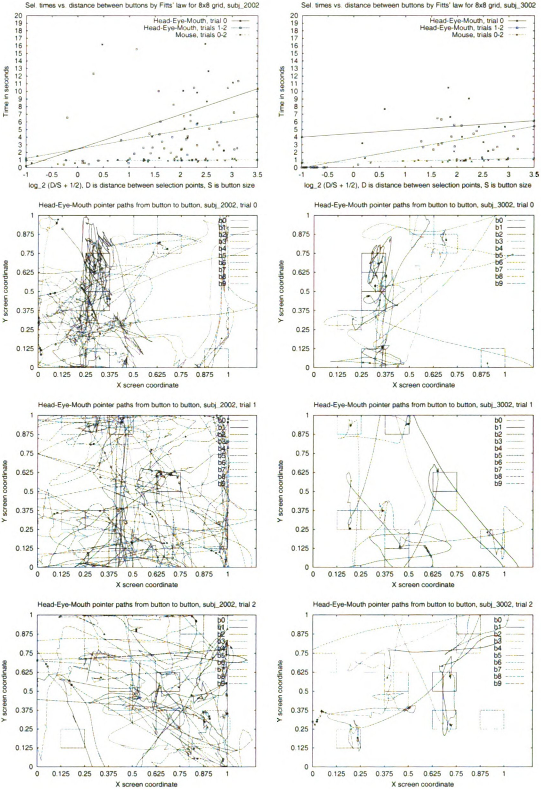
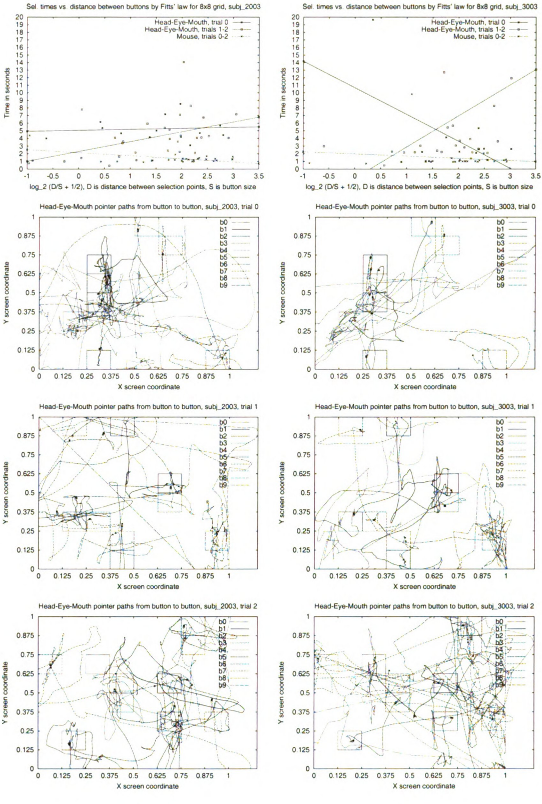


Figure C.3: Task T2, subject 02, Fitts' law time-distance plots and cursor paths for each button


Sel. times vs. distance between buttons by Fitts' law for 8x8 grid, subj_2003

Sel. times vs. distance between buttons by Fitts' law for 8x8 grid, subj_3003

Head-Eye-Mouth pointer paths from button to button, subj_2003, trial 0

Head-Eye-Mouth pointer paths from button to button, subj_3003, trial 0

Head-Eye-Mouth pointer paths from button to button, subj_2003, trial 1

Head-Eye-Mouth pointer paths from button to button, subj_3003, trial 1

Head-Eye-Mouth pointer paths from button to button, subj_2003, trial 2

Head-Eye-Mouth pointer paths from button to button, subj_3003, trial 2

Figure C.4: Task T2, subject 03, Fitts' law time-distance plots and cursor paths for each button

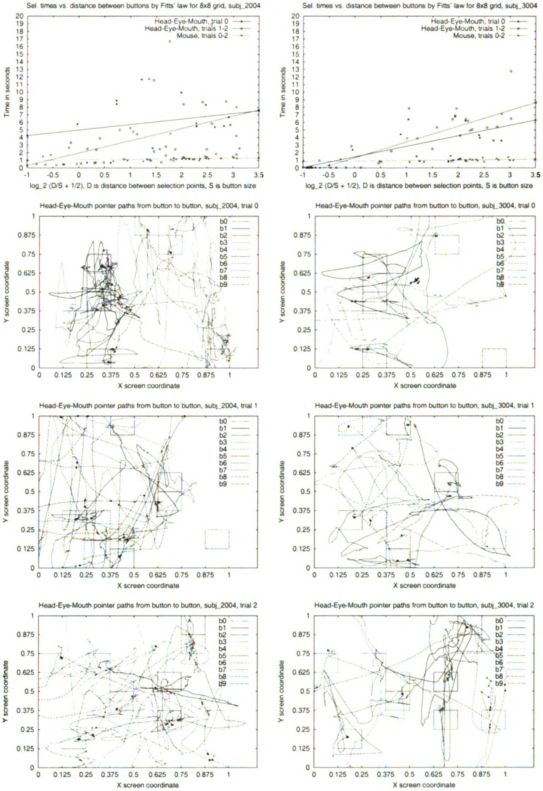


Figure C.5: Task T2, subject 04, Fitts' law time-distance plots and cursor paths for each button

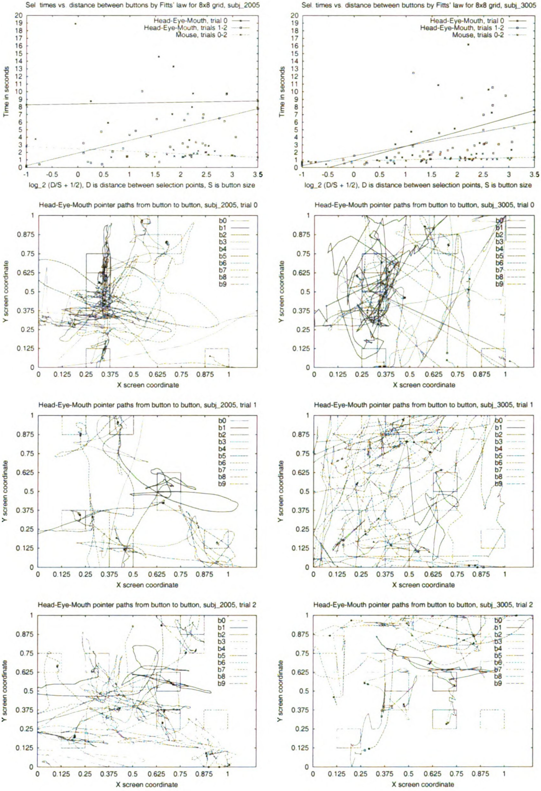


Figure C.6: Task T2, subject 05, Fitts' law time-distance plots and cursor paths for each button

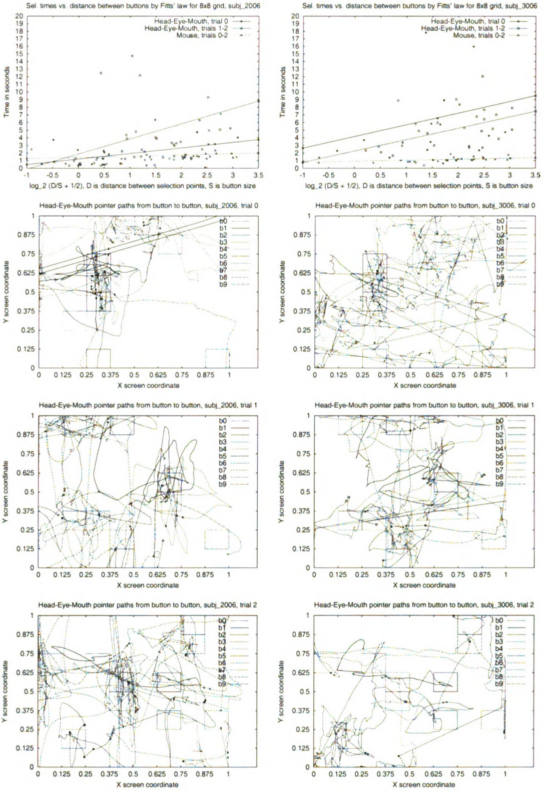


Figure C.7: Task T2, subject 06, Fitts' law time-distance plots and cursor paths for each button

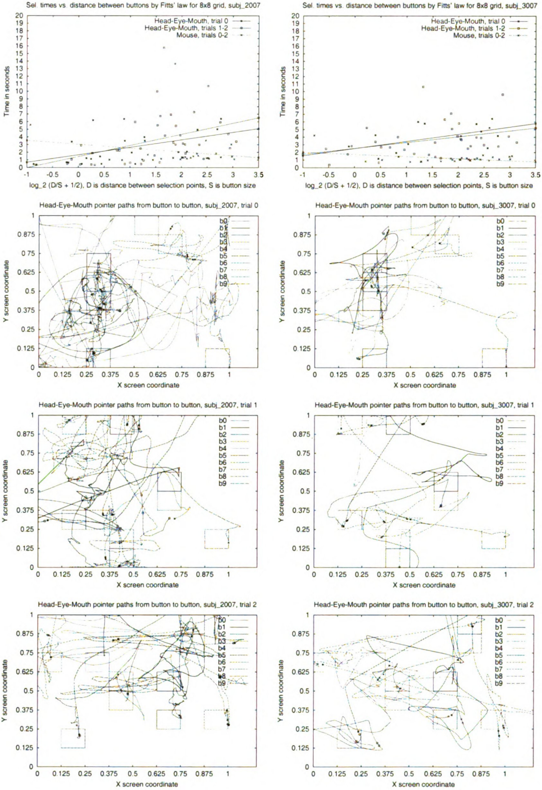


Figure C.8: Task T2, subject 07, Fitts' law time-distance plots and cursor paths for each button

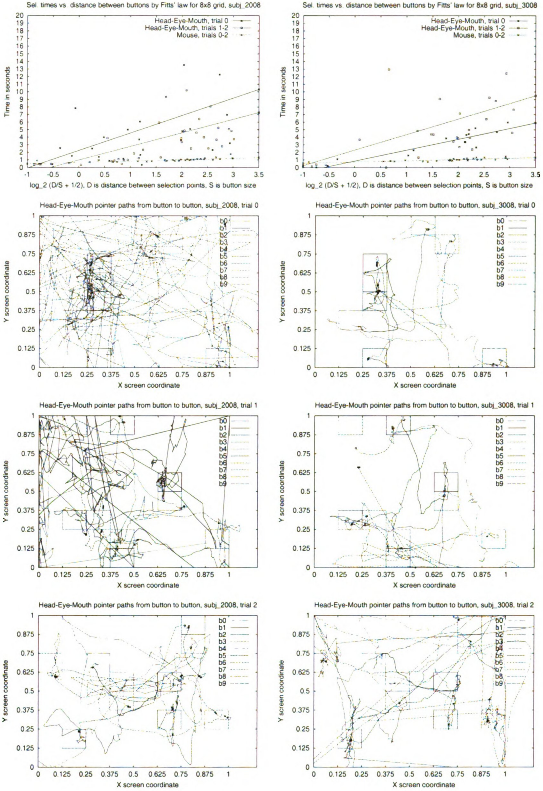


Figure C.9: Task T2, subject 08, Fitts' law time-distance plots and cursor paths for each button

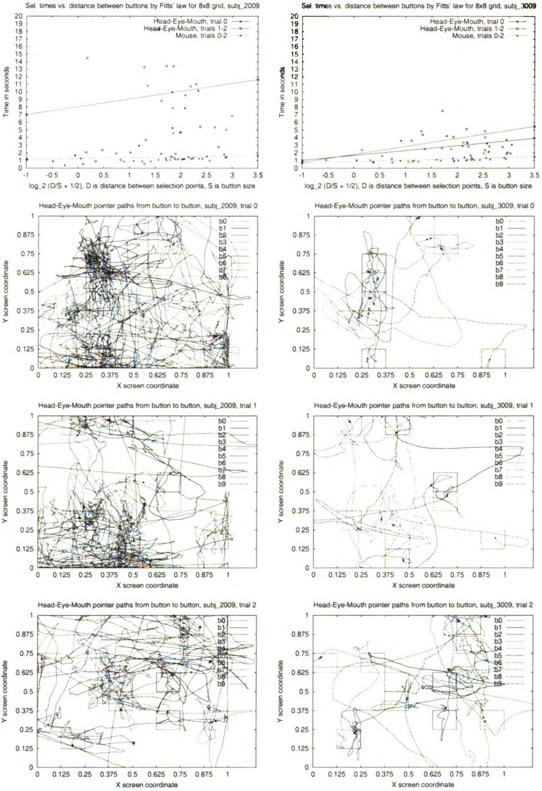


Figure C.10: Task T2, subject 09, Fitts' law time-distance plots and cursor paths for each button

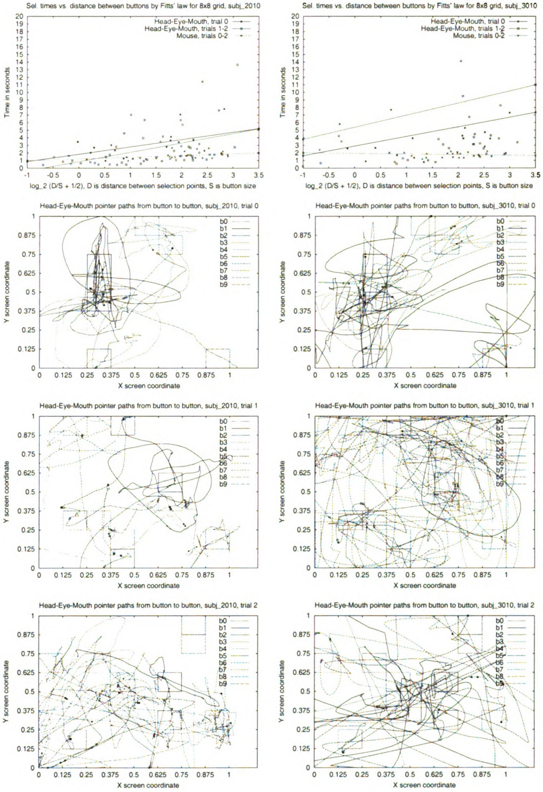


Figure C.11: Task T2, subject 10, Fitts' law time-distance plots and cursor paths for each button

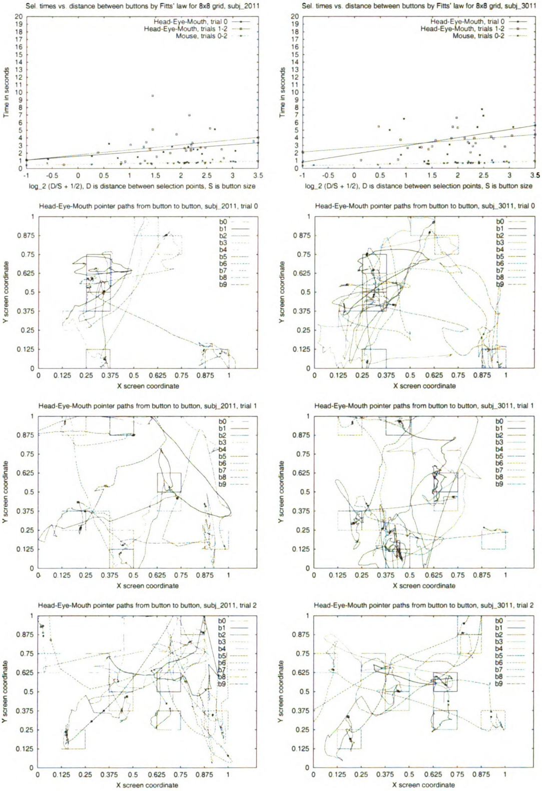


Figure C.12: Task T2, subject 11, Fitts' law time-distance plots and cursor paths for each button

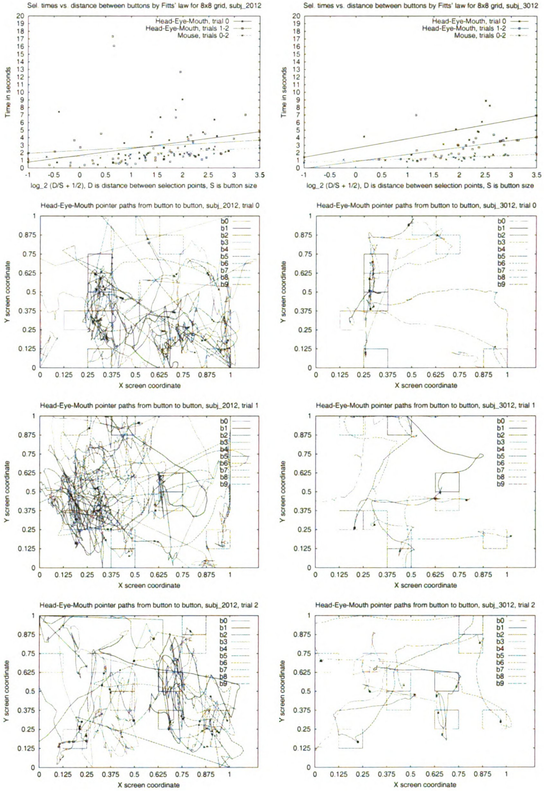


Figure C.13: Task T2, subject 12, Fitts' law time-distance plots and cursor paths for each button

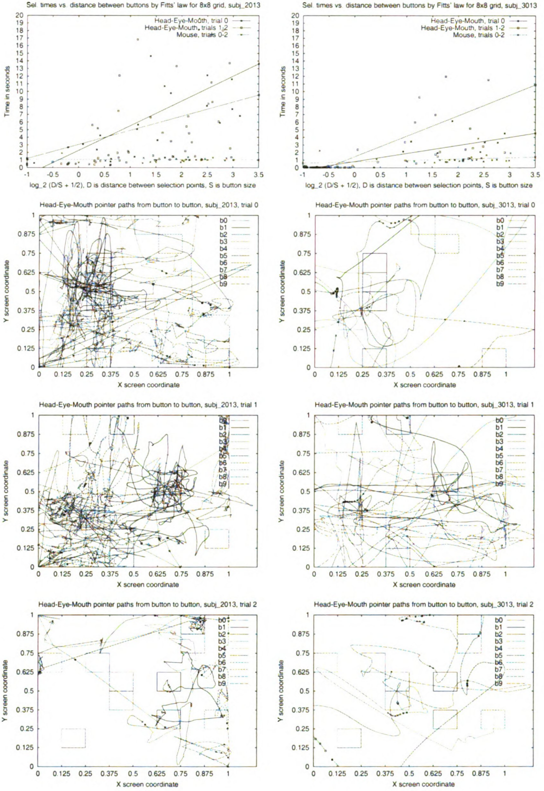


Figure C.14: Task T2, subject 13, Fitts' law time-distance plots and cursor paths for each button

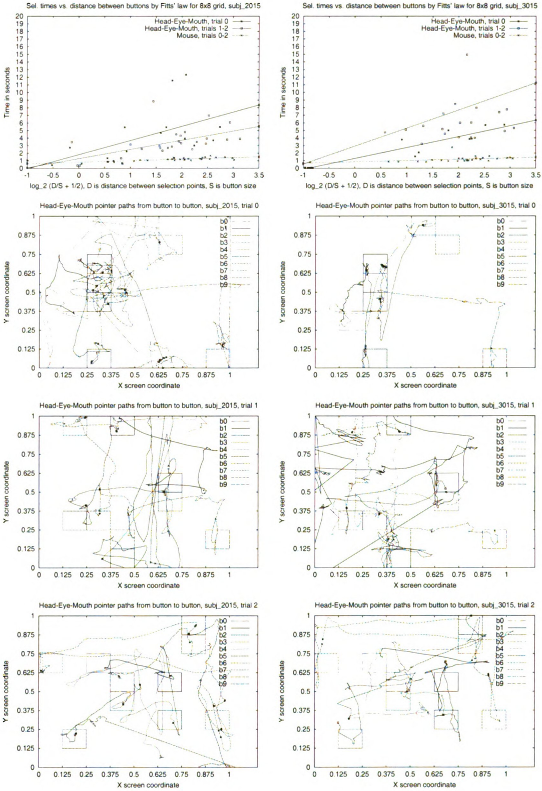


Figure C.15: Task T2, subject 15, Fitts' law time-distance plots and cursor paths for each button

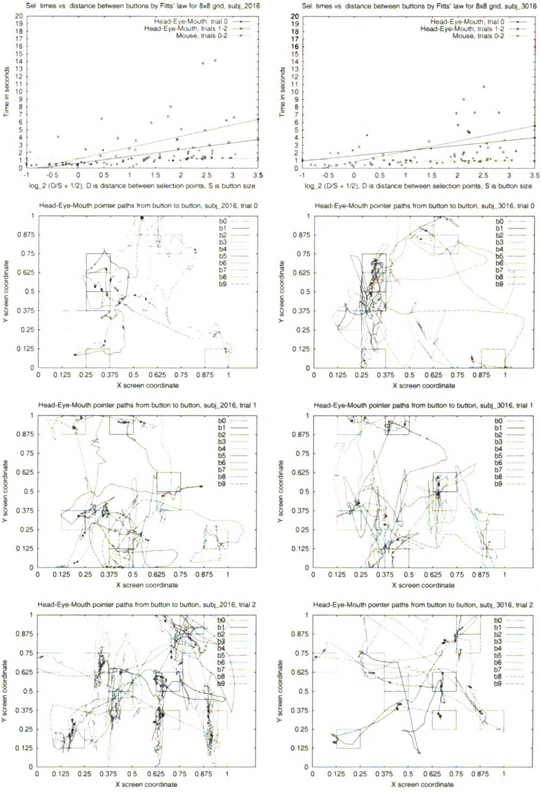


Figure C.16: Task T2, subject 16, Fitts' law time-distance plots and cursor paths for each button

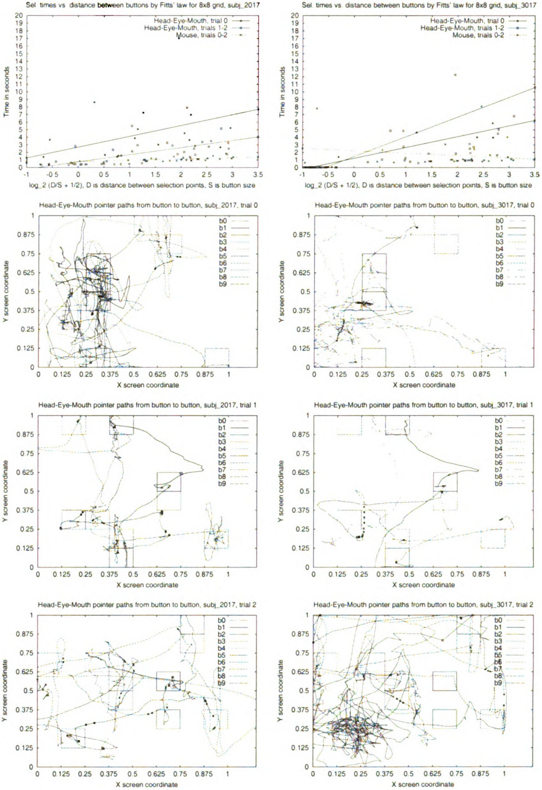


Figure C.17: Task T2, subject 17, Fitts' law time-distance plots and cursor paths for each button

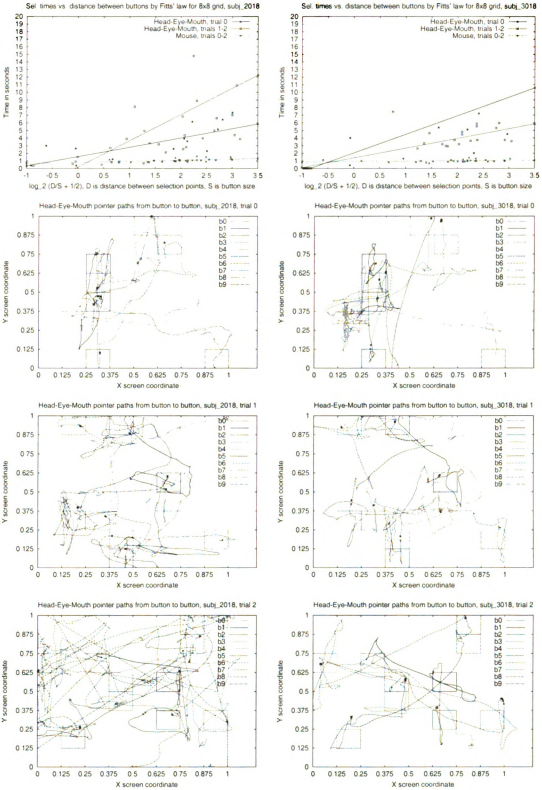


Figure C.18: Task T2, subject 18, Fitts' law time-distance plots and cursor paths for each button

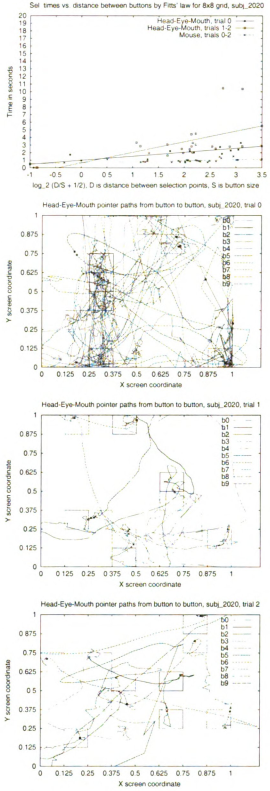


Figure C.19: Task T2, subject 20, Fitts' law time-distance plots and cursor paths for each button

Appendix D

Task T3 Complete Results

Table D.1 gives the statistics of the button selection accuracy and the average squared error from the linear regression line of the data plotted by Fitts' law. The first ten columns give the attempt number when the correct selection was made. The general trend from the first to the second session was that the selections were made mainly in the first two attempts and that the number of more-than-ten attempts is lower. Also, in the second session, most subjects were able to complete most draggings in one or two steps, unlike the first session, where the number of steps varied greatly.

Table D.2 shows the selection time statistics. The first ten columns for each session give the average total time needed to make a correct selection in the first, second,...,tenth attempt. The average time generally increases with the number of attempts, which is expected, since the user needs to make more than one selection. Also shown in the Table are total elapsed times for each trial, and for all the three trials. As the trials progress, the elapsed times didn't decrease. However, when compared across the sessions, both the individual trial times and the total time decreased significantly. This means that the users were able to make faster selections in the second session.

Figures D.1 through D.19 give the selection times vs. button distance by Fitts' law and the cursor paths for each two dragging locations, for all 18 subjects and the author.

		Selection correctness										Number of dragging steps										REG_LINE†		
Subj	Sess	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	E0	E1–2	M
00	1	25	14	0	2	0	0	0	0	0	3	7	4	2	2	0	0	0	0	0	0	526	139	-
	2	50	13	2	1	2	0	0	1	0	3	22	5	2	1	0	0	0	0	0	0	238	255	14
01	1	45	15	3	0	1	0	2	2	1	7	17	10	3	0	0	0	0	0	0	0	480	134	23
	2	43	93	2	1	3	2	3	3	0	2	23	6	1	0	0	0	0	0	0	0	353	249	23
02	1	11	6	1	1	2	0	2	1	0	21	3	1	4	0	0	0	0	1	0	1	69	-	53
	2	30	21	2	1	2	1	0	0	1	19	9	4	2	1	0	1	2	0	0	1	294	231	48
03	1	32	56	4	4	4	2	2	4	1	11	6	3	2	2	1	1	1	1	0	3	64	13	100
	2	55	86	8	10	5	3	3	0	2	6	5	10	4	0	2	1	0	3	0	5	176	166	43
04	1	33	80	9	3	4	0	3	4	2	47	4	6	6	3	1	2	4	0	0	4	69	33	39
	2	38	36	2	2	2	3	4	0	4	10	17	6	1	2	2	0	1	0	0	1	433	109	18
05	1	25	41	3	3	2	1	6	1	2	18	5	2	6	4	0	0	0	0	1	2	101	100	136
	2	44	44	4	2	5	4	2	2	2	19	10	5	4	5	0	2	2	0	1	1	138	124	37
06	1	41	36	4	2	5	5	3	1	1	25	11	6	6	2	2	0	0	1	0	2	78	105	11
	2	48	34	4	1	1	2	4	0	1	4	13	8	4	1	2	0	2	0	0	0	322	248	29
07	1	34	54	5	7	6	3	6	1	0	41	4	3	7	5	2	4	2	2	0	1	13	27	18
	2	37	98	8	3	2	5	4	2	6	24	4	9	1	4	2	2	0	1	2	5	85	96	12
08	1	47	52	4	4	3	4	2	3	4	8	9	8	3	3	4	0	1	0	1	1	218	117	14
	2	51	20	1	3	1	3	0	0	1	4	18	8	2	1	0	0	0	0	0	1	251	250	2
09	1	46	33	5	4	3	2	0	1	2	14	13	3	5	4	1	3	1	0	0	0	221	118	17
	2	49	30	2	1	3	1	2	1	4	7	17	6	2	2	1	0	0	0	0	2	189	131	16
10	1	34	29	0	7	3	5	1	2	2	60	9	2	3	6	7	1	0	0	0	2	5	18	40
	2	48	12	5	0	2	0	1	0	1	3	20	8	2	0	0	0	0	0	0	0	107	58	54
11	1	48	82	6	4	0	6	2	2	3	18	10	8	4	3	0	0	1	0	1	3	225	42	11
	2	56	17	1	2	1	0	1	0	0	1	17	9	2	2	0	0	0	0	0	0	358	193	7
12	1	37	37	7	5	0	5	6	1	3	43	14	5	2	1	3	1	0	1	0	3	18	40	18
	2	56	23	0	1	0	0	0	0	0	2	29	1	0	0	0	0	0	0	0	0	495	494	6
13	1	6	43	5	3	5	2	5	1	2	35	0	0	0	0	0	0	1	1	0	4	23	-	156
	2	38	10	4	1	3	2	3	2	1	16	16	9	4	1	0	0	0	0	0	0	211	121	140
15	1	55	15	0	0	0	2	0	0	0	2	24	4	0	1	0	0	0	1	0	0	619	248	19
	2	58	8	1	0	0	0	0	0	0	3	26	3	0	0	0	0	0	1	0	0	212	106	391
16	1	31	66	6	6	3	1	7	4	5	40	9	3	3	3	1	1	2	2	1	4	38	34	18
	2	30	41	2	2	0	1	2	4	3	47	4	9	5	4	3	3	1	0	1	0	56	19	21
17	1	39	41	6	6	6	2	7	4	2	35	9	4	6	4	1	0	1	1	0	4	70	84	37
	2	38	16	2	0	2	5	1	0	1	11	21	4	3	2	0	0	0	0	0	0	146	117	-
18	1	50	50	7	9	6	1	1	2	2	9	9	6	5	3	1	1	2	0	1	2	126	130	15
	2	52	53	11	3	5	4	1	2	1	6	13	6	1	2	2	4	0	0	1	1	145	175	11
20	2	57	14	3	0	0	1	0	0	1	0	21	4	3	2	0	0	0	0	0	0	476	283	32
Avg	1	35	42	4	4	3	2	3	1	2	24	9	4	4	3	1	1	1	1	0	2	165	86	43
	2	46	36	3	2	2	2	2	1	2	10	16	6	2	2	1	1	0	0	0	1	234	175	51
	1	Across subject avg. number of steps: min 35, max 131, avg 84																						
	2	Across subject avg. number of steps: min 31, max 139, avg 67																						

† - Average squared error of the regression line by Fitts' law

Table D.1: Task T3: button selection accuracy statistics

		Avg. time for										Trial			Total
		selection (sec)										time (sec)			time
Subj	Sess	1	2	3	4	5	6	7	8	9	10	T0	T1	T2	(sec)
00	1	5.4	4.9	0.0	15.1	0.0	0.0	0.0	0.0	0.0	23.7	110	100	91	303
	2	4.3	2.8	4.3	3.8	5.7	0.0	0.0	2.9	0.0	7.9	88	126	88	303
01	1	3.5	4.6	4.4	0.0	8.9	0.0	4.4	8.6	4.0	5.8	105	101	111	318
	2	4.2	4.0	4.8	4.5	15.5	6.9	8.6	4.4	0.0	6.6	102	120	120	344
02	1	1.7	1.8	7.5	7.4	4.2	0.0	7.2	4.3	0.0	11.9	321	-	-	321
	2	7.9	4.7	11.3	12.7	6.3	11.5	0.0	0.0	5.4	18.5	290	462	-	753
03	1	3.3	4.0	6.1	10.8	7.9	11.0	32.8	8.9	14.5	6.6	314	323	-	637
	2	3.8	4.6	4.8	15.7	9.7	5.1	6.7	0.0	5.6	7.9	295	510	133	939
04	1	3.5	1.7	5.6	14.4	4.2	0.0	7.3	3.5	4.5	9.4	183	480	181	845
	2	4.8	2.9	9.0	12.9	5.6	8,7	5.2	0.0	8.1	9.7	133	234	148	515
05	1	3.9	1.9	11.4	7.8	5.0	13.1	7.6	34.8	4.6	16.0	355	281	-	636
	2	5.3	4.3	13.9	10.1	13.8	23.7	8.7	4.4	11.3	15.9	439	417	156	1014
06	1	3.4	4.4	3.6	5.5	5.9	5.4	9.2	72.4	13.5	12.8	467	156	185	809
	2	3.6	4.9	12.9	5.5	3.7	22.9	13.7	0.0	3.9	12.9	125	171	262	558
07	1	3.9	2.7	4.6	3.1	4.4	5.6	6.0	3.5	0.0	8.8	331	218	218	769
	2	4.7	4.2	9.5	15.2	13.4	4.4	12.3	10.7	13.5	29.5	885	336	391	1613
08	1	3.6	3.0	10.2	10.5	4.5	5.0	8.5	3.7	5.1	5.3	250	162	119	533
	2	3.9	3.7	5.3	5.0	5.4	4.7	0.0	0.0	7.3	9.5	142	107	106	356
09	1	3.3	3.9	5.1	2.7	2.9	19.3	0.0	1.1	15.4	25.7	343	136	275	755
	2	4.1	4.3	4.9	8.7	7.9	12.8	3.2	3.9	8.6	10.0	172	230	98	501
10	1	2.6	2.7	0.0	4.6	3.8	2.8	8.4	4.0	4.9	9.3	297	178	332	807
	2	3.8	3.1	8.0	0.0	7.2	0.0	3.7	0.0	3.3	8.9	113	91	103	308
11	1	3.6	1.1	6.2	10.8	0.0	6.9	4.0	5.6	8.9	8.0	190	299	87	577
	2	3.6	3.4	2.2	3.4	1.7	0.0	4.2	0.0	0.0	19.1	79	127	86	293
12	1	5.1	2.3	5.4	4.1	0.0	7.5	6.1	3.7	8.8	10.0	421	252	190	864
	2	5.7	2.5	0.0	5.0	0.0	0.0	0.0	0.0	0.0	6.4	105	140	98	344
13	1	1.1	4.4	16.6	16.6	16.4	5.7	18.5	1.9	5.5	27.0	1470	-	-	1470
	2	4.4	6.2	11.4	4.0	8.9	12.7	11.6	10.6	16.7	8.9	184	182	178	545
15	1	4.3	3.2	0.0	0.0	0.0	5.9	0.0	0.0	0.0	7.0	108	117	84	309
	2	4.7	3.1	19.8	0.0	0.0	0.0	0.0	0.0	0.0	6.0	108	127	104	340
16	1	3.0	2.6	6.4	5.1	4.8	5.4	6.4	4.4	5.2	8.4	247	249	277	774
	2	3.5	3.1	11.9	8.4	0.0	4.9	5.1	5.0	5.9	13.7	512	206	253	971
17	1	4.9	3.8	10.1	5.2	8.0	43.2	16.0	13.7	8.1	12.7	537	458	201	1198
	2	6.6	8.1	7.2	0.0	2.0	6.1	12.7	0.0	6.4	18.1	252	225	167	645
18	1	4.6	3.7	9.5	7.5	5.9	5.1	12.1	7.1	7.6	4.8	160	287	224	673
	2	4.3	3.2	7.4	12.5	6.0	4.6	4.0	27.6	8.5	7.5	151	324	197	673
20	-	3.6	4.5	5.6	0.0	0.0	6.0	0.0	0.0	6.2	0.0	102	98	95	295
Avg	1	3.6	3.1	6.3	7.3	4.8	7.9	8.6	10.1	6.1	11.8	345	237	184	700
	2	4.6	4.1	8.3	7.1	6.3	7.1	5.5	3.9	5.8	12.1	232	230	158	612

Table D.2: Task T3: button selection timing statistics

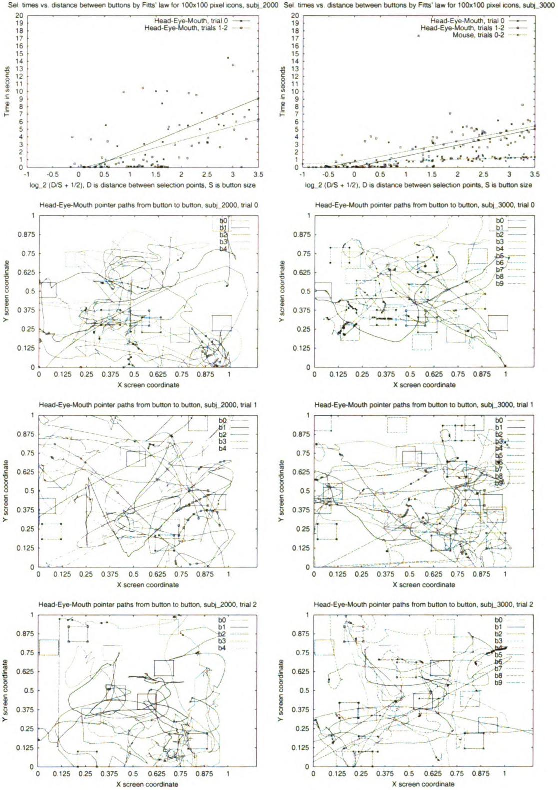


Figure D.1: Task T3, subject 00, Fitts' law time-distance plots and cursor paths for each target

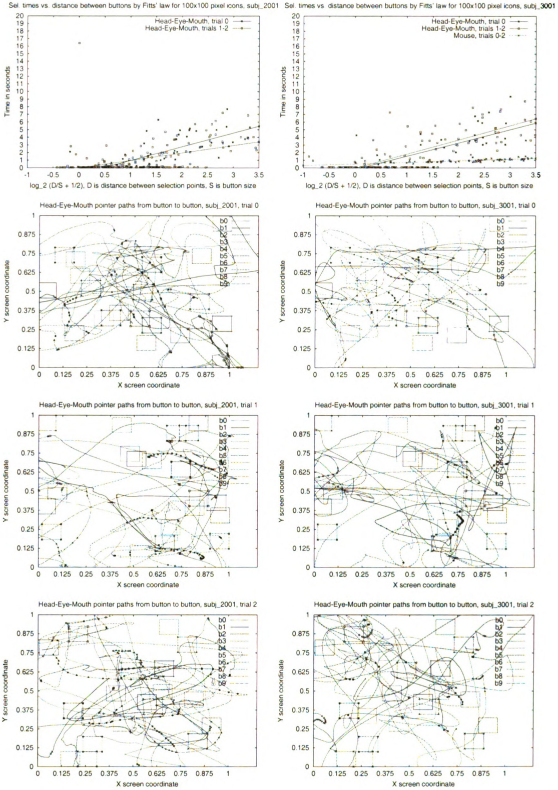
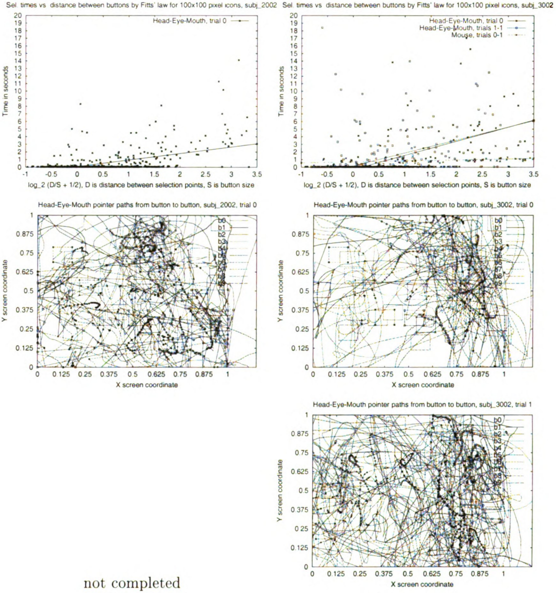


Figure D.2: Task T3, subject 01, Fitts' law time-distance plots and cursor paths for each target



not completed
not completed

not completed

Figure D.3: Task T3, subject 02, Fitts' law time-distance plots and cursor paths for each target

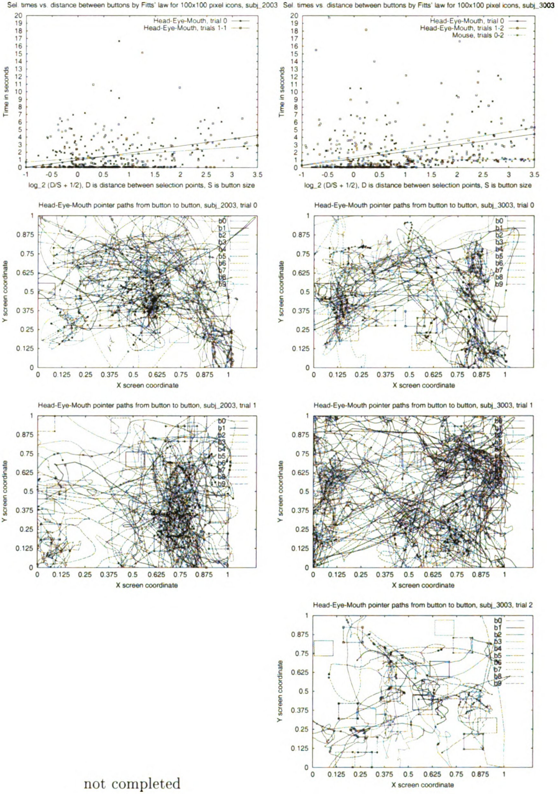


Figure D.4: Task T3, subject 03, Fitts' law time-distance plots and cursor paths for each target

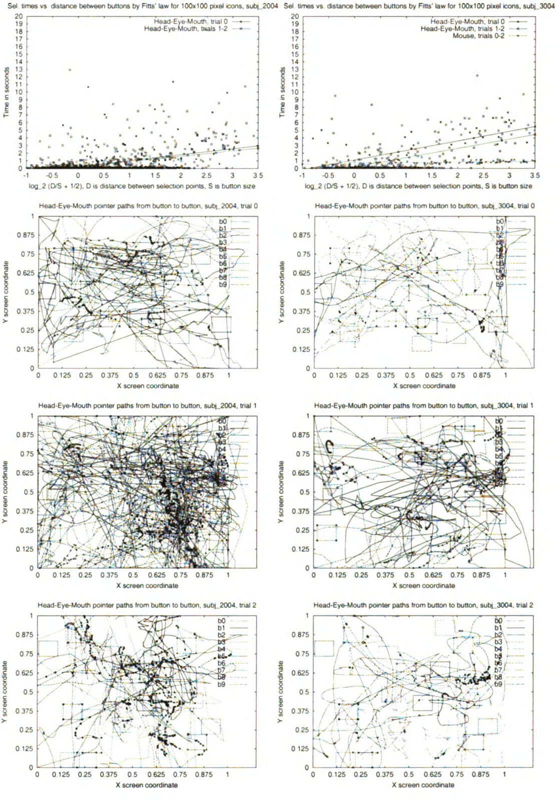
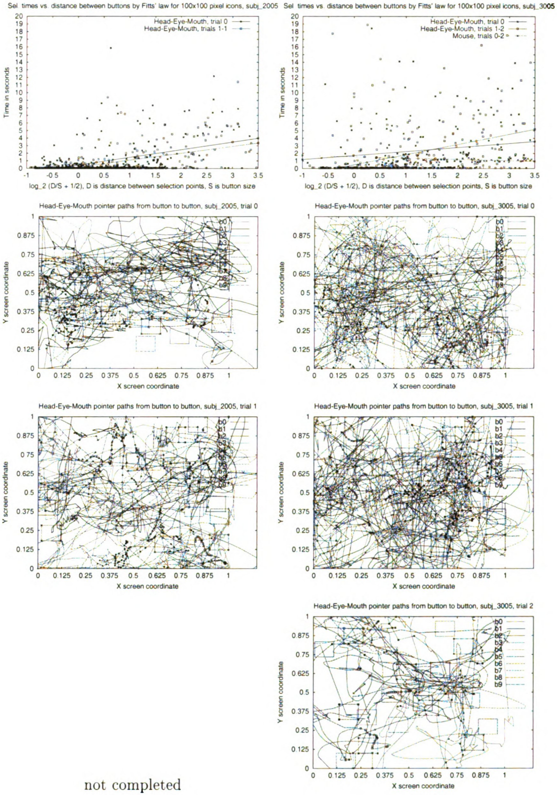


Figure D.5: Task T3, subject 04, Fitts' law time-distance plots and cursor paths for each target



not completed

Figure D.6: Task T3, subject 05, Fitts' law time-distance plots and cursor paths for each target

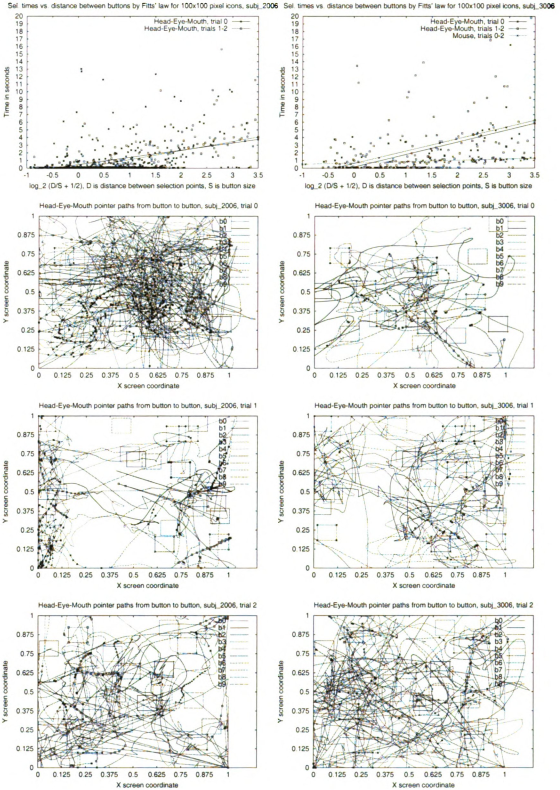


Figure D.7: Task T3, subject 06, Fitts' law time-distance plots and cursor paths for each target

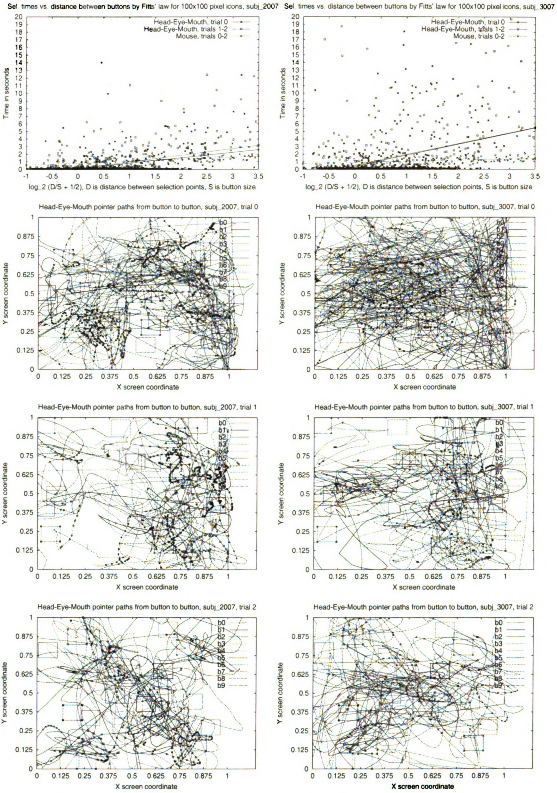
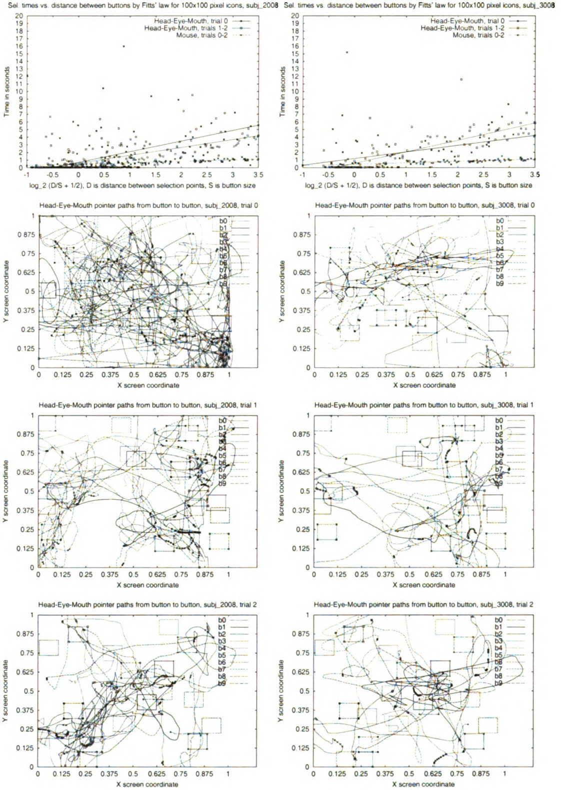


Figure D.8: Task T3, subject 07, Fitts' law time-distance plots and cursor paths for each target



Sel. times vs. distance between buttons by Fitts' law for 100x100 pixel icons, subj_2008

Sel. times vs. distance between buttons by Fitts' law for 100x100 pixel icons, subj_3008

Head-Eye-Mouth pointer paths from button to button, subj_2008, trial 0

Head-Eye-Mouth pointer paths from button to button, subj_3008, trial 0

Head-Eye-Mouth pointer paths from button to button, subj_2008, trial 1

Head-Eye-Mouth pointer paths from button to button, subj_3008, trial 1

Head-Eye-Mouth pointer paths from button to button, subj_2008, trial 2

Head-Eye-Mouth pointer paths from button to button, subj_3008, trial 2

Figure D.9: Task T3, subject 08, Fitts' law time-distance plots and cursor paths for each target

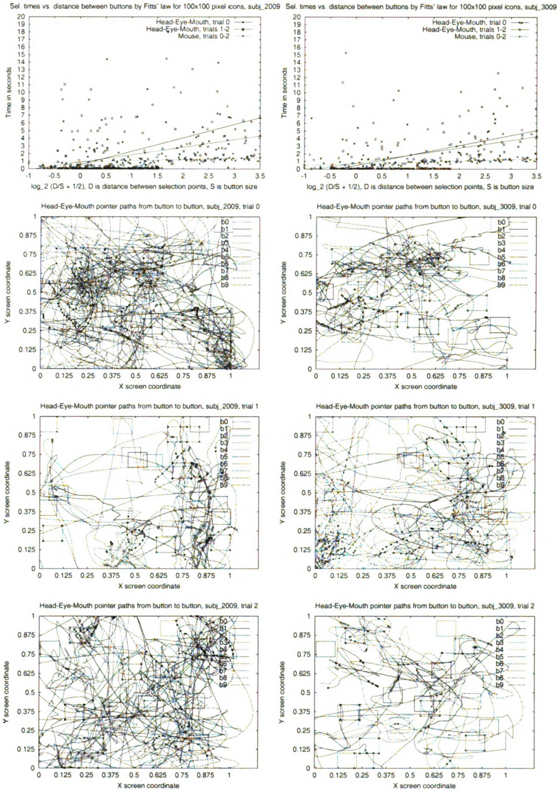


Figure D.10: Task T3, subject 09, Fitts' law time-distance plots and cursor paths for each target

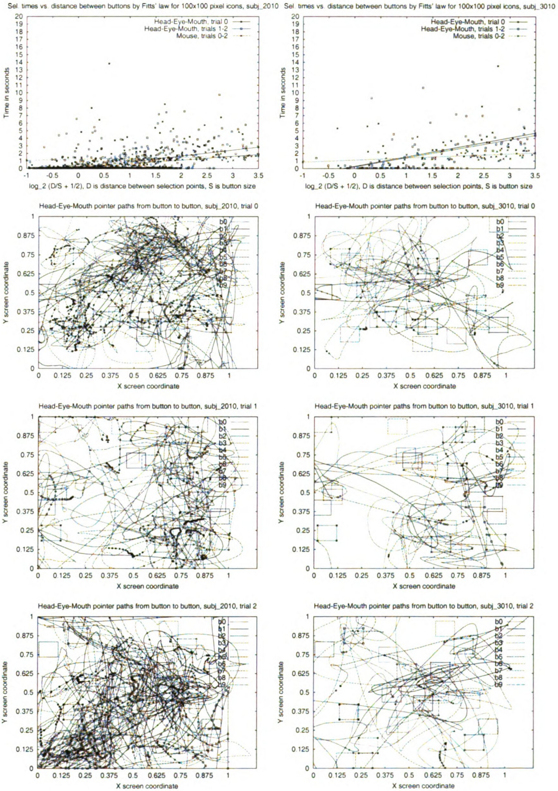
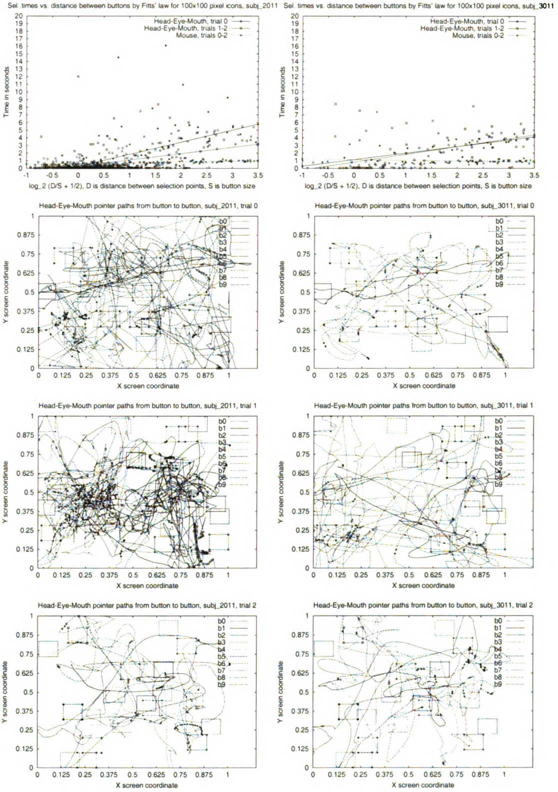


Figure D.11: Task T3, subject 10, Fitts' law time-distance plots and cursor paths for each target


Sel. times vs. distance between buttons by Fitts' law for 100x100 pixel icons, subj_2011

Sel. times vs. distance between buttons by Fitts' law for 100x100 pixel icons, subj_3011

Head-Eye-Mouth pointer paths from button to button, subj_2011, trial 0

Head-Eye-Mouth pointer paths from button to button, subj_3011, trial 0

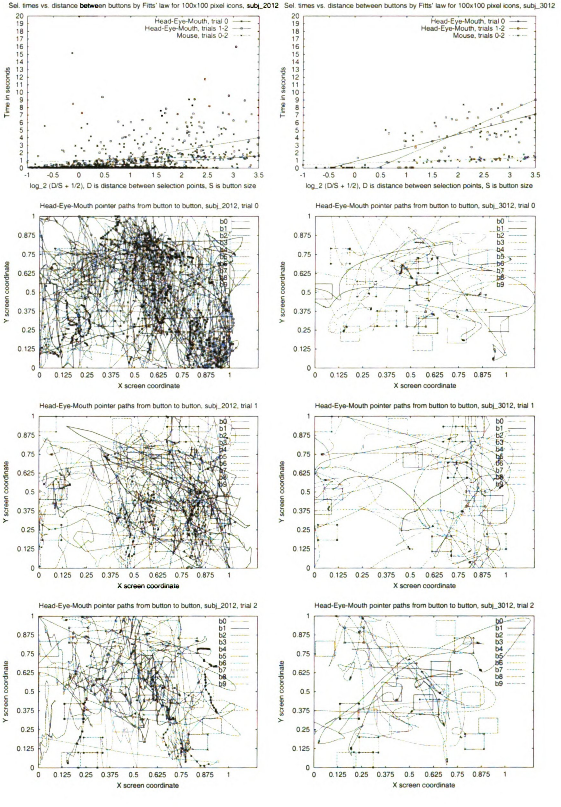
Head-Eye-Mouth pointer paths from button to button, subj_2011, trial 1

Head-Eye-Mouth pointer paths from button to button, subj_3011, trial 1

Head-Eye-Mouth pointer paths from button to button, subj_2011, trial 2

Head-Eye-Mouth pointer paths from button to button, subj_3011, trial 2

Figure D.12: Task T3, subject 11, Fitts' law time-distance plots and cursor paths for each target



Sel. times vs. distance between buttons by Fitts' law for 100x100 pixel icons, subj_2012

Sel. times vs. distance between buttons by Fitts' law for 100x100 pixel icons, subj_3012

Head-Eye-Mouth pointer paths from button to button, subj_2012, trial 0

Head-Eye-Mouth pointer paths from button to button, subj_3012, trial 0

Head-Eye-Mouth pointer paths from button to button, subj_2012, trial 1

Head-Eye-Mouth pointer paths from button to button, subj_3012, trial 1

Head-Eye-Mouth pointer paths from button to button, subj_2012, trial 2

Head-Eye-Mouth pointer paths from button to button, subj_3012, trial 2

Figure D.13: Task T3, subject 12, Fitts' law time-distance plots and cursor paths for each target

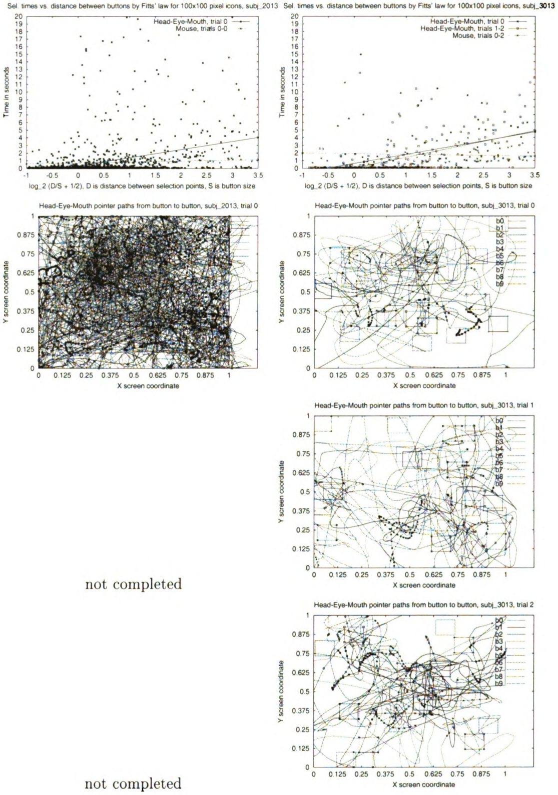


Figure D.14: Task T3, subject 13, Fitts' law time-distance plots and cursor paths for each target

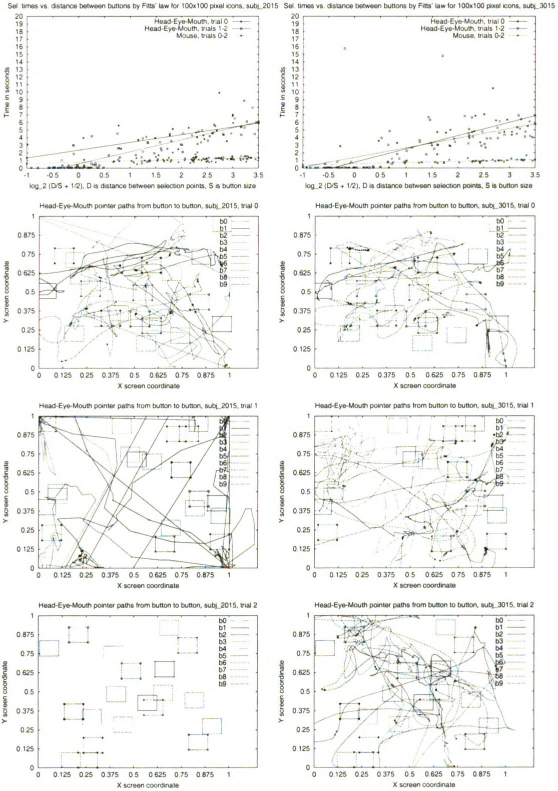


Figure D.15: Task T3, subject 15, Fitts' law time-distance plots and cursor paths for each target

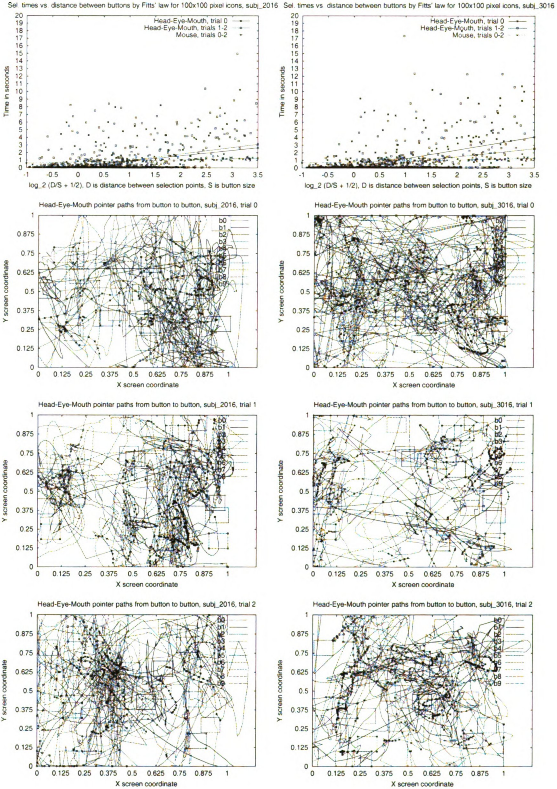


Figure D.16: Task T3, subject 16, Fitts' law time-distance plots and cursor paths for each target

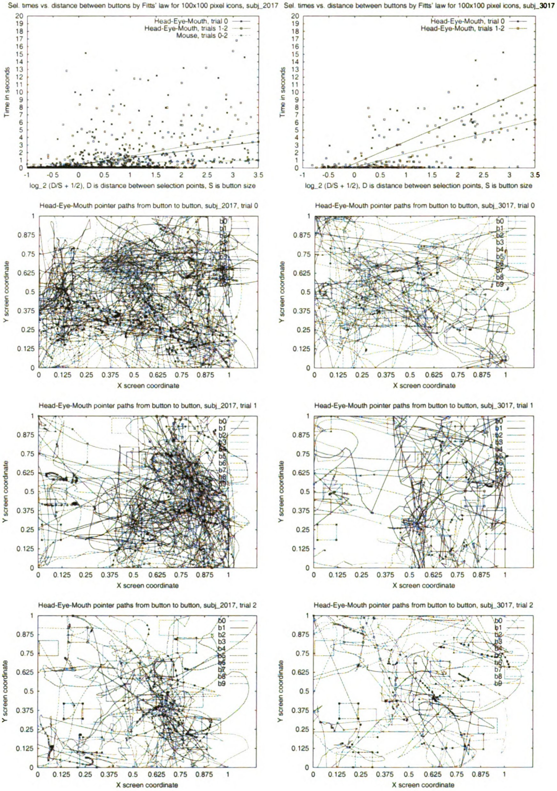


Figure D.17: Task T3, subject 17, Fitts' law time-distance plots and cursor paths for each target

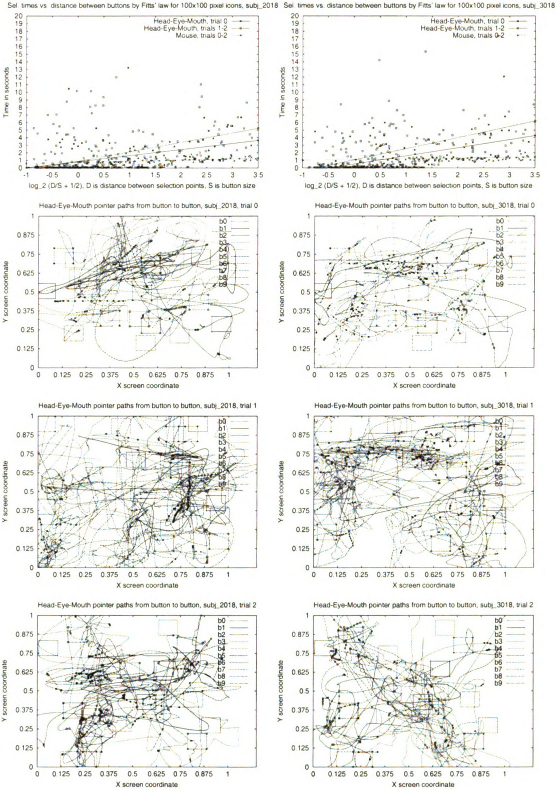


Figure D.18: Task T3, subject 18, Fitts' law time-distance plots and cursor paths for each target

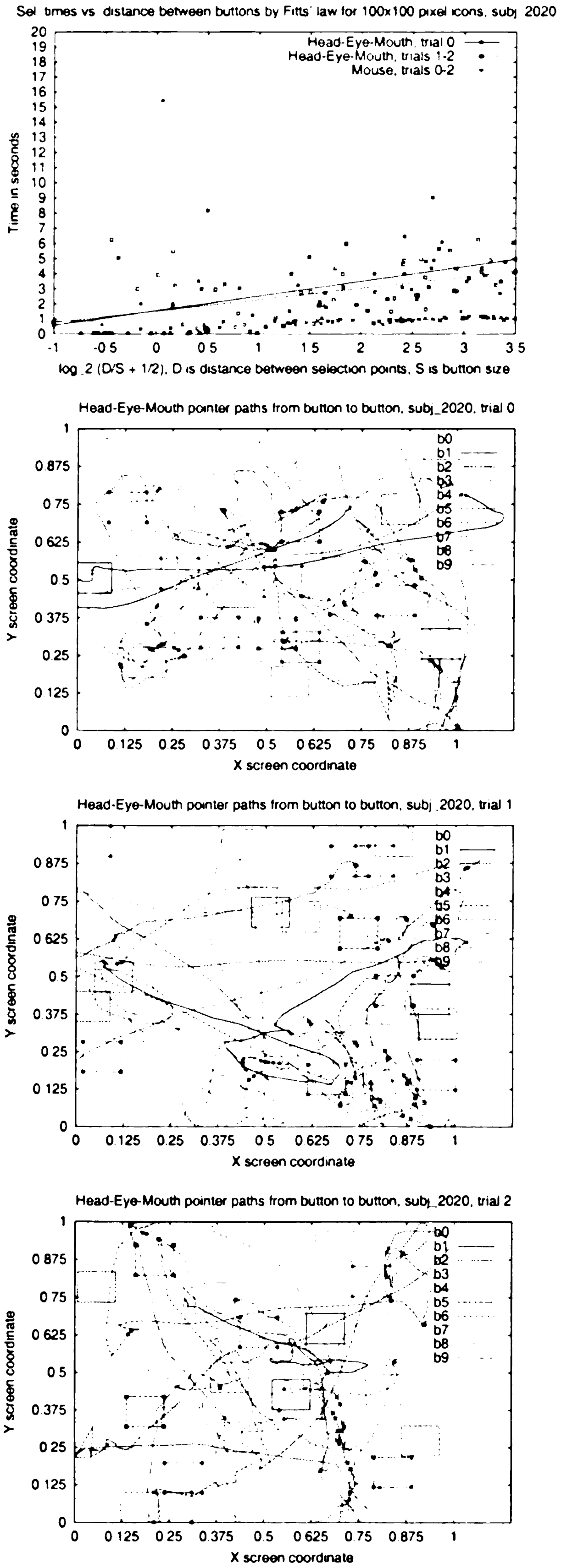


Figure D.19: Task T3, subject 20, Fitts' law time-distance plots and cursor paths for each target

Appendix E

Task T5 Complete Results

Table E.1 gives the sizes of the buttons used in the experiment. The sizes are given in pixel units, inches and visual angle.

Table E.2 gives these data:

- number of erroneous selections (column 3),
- the number of times the experimenter had to turn off the camera (column 4),
- the number of times the experimenter had to type (column 5),
- time needed to complete the task (column 6),
- percentage of the task completed (column 7), and
- the sub-tasks that were skipped (column 8).

Buttons	$x \times y$ measures in:			minimal
identifications	pixels	inches	degrees	angle size
Netscape browser	994 × 910	11.93 × 10.27	27.48 × 24.33	24.33
URL entry	590 × 25	7.08 × 0.28	16.31 × 0.67	0.67
URL arrow	15 × 25	0.18 × 0.28	0.41 × 0.67	0.41
Back/send buttons	50 × 50	0.60 × 0.56	1.38 × 1.34	1.34
Close buttons panel	10 × 50	0.12 × 0.56	0.27 × 1.34	0.27
Scroll bar	13 × 100^{+}	0.16 × 1.13	0.36 × 2.67	0.36
Mail button	30 × 25	0.36 × 0.28	0.83 × 0.67	0.67
MSNBC menu button	90 × 20	1.08 × 0.22	2.49 × 0.53	0.53
Story title	200^{+} × 20	2.40 × 0.22	5.53 × 0.53	0.53
ZIP code entry	80 × 30	0.96 × 0.34	2.21 × 0.80	0.80
GO button	30 × 30	0.36 × 0.34	0.83 × 0.80	0.80
Mail window	740 × 740	8.88 × 8.35	20.46 × 19.78	19.78
"To:" field	560 × 20	6.72 × 0.22	15.48 × 0.53	0.53
"Subject:" field	480 × 30	5.76 × 0.34	13.27 × 0.80	0.80
text entry area	700 × 460	8.40 × 5.19	19.35 × 12.30	12.30
Select panel	20 × 20	0.24 × 0.22	0.54 × 0.53	0.53

Table E.1: Sizes of typical buttons in the Netscape browser and mail windows, and a typical news WWW page

Subj	Sess	num. err	num. turn off	exper. types	time mm:ss	% finish	skipped sub-tasks
00	1	1	0	0	7:00	80	not done 2,3,4,5†
	2	0	0	0	3:34	100	
01	1	2	0	0	2:50	84	not done 2,3,4†
	2	4	0	0	3:24	100	
02	1	10	0	1	13:52	100	
	2	2	0	0	7.25	90	not done 2,4,5†
03	1	9	0	1	12:39	100	
	2	4	0	0	8:47	100	
04	1	10	0	0	7:23	100	
	2	14	1	0	9:02	100	
05	1	11	2	0	7.36	82	skip 5,6
	2	5	0	1	12:11	70	skip 3,4,5,6
06	1	18	2	0	12:27	82	skip 2,6
	2	10	0	0	9:50	100	
07	1	10	1	1	15:27	88	skip 3,4
	2	3	0	0	11:10	100	
08	1	2	0	0	9:34	100	
	2	6	0	0	5:24	100	
09	1	7	0	0	7:46	100	
	2	1	0	0	5:20	100	
10	1	19	0	1	12:16	86	skip 6
	2	4	0	0	9:49	100	
11	1	1	0	0	3:28	100	
	2	4	0	0	4:41	100	
12	1	18	0	1	13:00	40	skip 6,7,8,10
	2	5	1	0	7:58	100	
13	1	12	1	0	10:16	20	skip 1,6,7,8,9,10
	2	8	1	1	17:45	86	skip 6‡
15	1	0	0	0	3.58	100	
	2	2	0	0	8:46	86	skip 6‡
16	1	11	1	0	11:03	82	skip 5,6
	2	3	1	0	6:54	100	
17	1	12	1	0	10:33	86	skip 6
	2	18	0	1	13:08	100	
18	1	1	0	0	4:26	100	
	2	0	0	0	9:28	100	
20		0	0	0	2:40	100	
		with the mouse			1:15	100	
Sum	1	154	8	5			
	2	93	4	3			
Avg	1	8.6	0.4	0.3	9:12	85	
	2	5.2	0.2	0.2	8:35	96	

Percentage improvement: # of errors 40.6%, elapsed time 6.6%

† - these subjects' task did not include some sub-tasks due to time constraints, but they completed 100% of their current task.

‡ - these subjects skipped the task due to network problems, but would otherwise attempted it.

Table E.2: Task T5: number of errors and timing statistics

BIBLIOGRAPHY

Bibliography

[1] Applied science laboratories. [Online] Available `http://www.a-s-l.com/`.

[2] Blue eyes project. [Online] Available `http://www.almaden.ibm.com/cs/blueeyes/`.

[3] Iscan incorporated. [Online] Available `http://www.iscaninc.com/`.

[4] Lc technologies, inc. [Online] Available `http://www.eyegaze.com/`.

[5] Newabilities system inc. [Online] Available `http://members.aol.com/UCS1000/home.htm`.

[6] Tracker 2000. [Online] Available `http://www.madentec.com/`.

[7] V. Bakić and K. Miller. Using neural networks to recognize facial expressions. Technical Report MSU-CPS-98-5, Department of Computer Science, Michigan State University, 1998.

[8] V. Bakić and G. Stockman. Menu selection by facial aspect. *Proceedings of Vision Interface*, pages 203–209, 1999. [Online] Available `http://www.cse.msu.edu/~bakicvel/faces/`.

[9] L.-P. Bala, K. Talmi, and J. Liu. Automatic detection and tracking of faces and facial features in video sequences. *Proceedings of Picture Coding Symposium*, September 1997, Berlin, Germany. [Online] Available `http://atwww.hhi.de:80/~blick/Papers/papers.html`.

[10] P. Ballard and G. Stockman. Controlling a computer via facial aspect. *IEEE Transactions on System, Man and Cybernetics, Vol. 25, No. 4*, pages 669–677, April 1995.

[11] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. Technical Report CMU-CS-94-102, Carnegie Mellon University, School of Computer Science, 1994.

[12] P. A. Beardsley. A qualitative approach to classifying head and eye pose. *Proceedings of Workshop on Application of Computer Vision*, pages 208–213, 1998.

[13] G. R. Bradski. Real time face and object tracking as a component of a perceptual user interface. *Proceedings of Workshop on Application of Computer Vision*, pages 214–219, 1998.

[14] C. Buquet, J. R. Charlier, and V. Paris. Museum application of an eye tracker. *Medical and Biomedical Engineering and Computing, Vol. 26*, pages 277–281, 1988.

[15] G. Burel and D. Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters, Vol. 15*, pages 963–967, 1994.

[16] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

[17] J. Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics, Vol. 22, No. 2*, pages 249–254, 1996. [Online] Available `http://www.iccs.informatics.ed.ac.uk/~jeanc/squib.ps`.

[18] J. M. Carroll and J. Reitman Olson. *Chapter 2: Mental Models in Human-Computer Interaction*, pages 45–65. In Helander [36], 1988.

[19] J. M. Carroll and J. C. Thomas. Metaphor and the cognitive representation of computing system. *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 12*, pages 107–116, 1982.

[20] R. Cipolla and A. Pentland, editors. *Computer Vision for Human-Machine Interaction.* Cambridge University Press, 1998.

[21] A. Colmenarez, B. Frey, and T. S. Huang. A probabilistic framework for embedded face and facial expression recognition. *Proceedings of Computer Vision and Pattern Recognition, Vol. I*, pages 592–597, 1999.

[22] H. D. Crane. The purkinje image eyetracker, image stabilization, and related forms of stimulus manipulation. In D. H. Kelley, editor, *Visual science and engineering: Models and applications*, pages 15–89. Macel Dekker, New York, 1994.

[23] H. D. Crane and C. M. Steele. Generation-v dual-purkinje-image eyetrcker. *Applied Optics, Vol. 24*, pages 527–537.

[24] D. F. DeMenthon and L. S. Davis. Model based object pose in 25 lines of code. *International Journal of Computer Vision, Vol. 15*, pages 123–141, June 1995. [Online] Available `http://www.cfar.umd.edu/~daniel/`.

[25] B. DuBoulay, T. O'Shea, and J. Monk. The black box inside the glass box: Presenting computing to novices. *International Journal of Man-Machine Studies, Vol. 14*, pages 237–249, 1981.

[26] I. Essa and A. Pentland. A vision system for observing and extracting facial action parameters. *Proceedings of Computer Vision and Pattern Recognition*, pages 76–83, 1994.

[27] Scott E. Fahlman. An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, Carnegie Mellon University, School of Computer Science, 1988.

[28] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, and R. L. Phillips. *Introduction to Computer Graphics.* Addison-Wesley Publishing Company, 1993.

[29] W. T. Freeman, D. B. Anderson, P. A. Beardsley, C. N. Dodge, M. Roth, C. D. Weissman, W. S. Yerazunis, H. Kage, K. Kyuma, Y. Miyake, and K.-I. Tanaka. Computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications, Vol. 18, No. 3*, pages 42–53, May-June 1998.

[30] L. A. Frey, JR. K. P. White, and T. E. Hutchinson. Eye-gaze word processing. *IEEE Transactions on System, Man and Cybernetics, Vol. 20, No. 4*, pages 944–950, July/August 1990.

[31] A. Gee and R. Cipolla. Non-intrusive gaze tracking for human-computer interaction. *Proc. Mechatronics and Machine Vision in Practice*, pages 112–117, 1994. [Online] Available `http://www.diku.dk/~panic/eyegaze/`.

[32] A. J. Glenstrup and T. Eugell-Nielsen. Eye controlled media: Present and future state. June 1995. [Online] Available `http://www.dicu.dk/~panic/eyegaze/`.

[33] V. Govindaraju, D. Sher, and R. Srihari. Locating human faces in newspaper photographs. *Proceedings of Computer Vision and Pattern Recognition*, pages 549–554, 1989.

[34] V. Govindaraju, R. Srihari, and D. Sher. A computational model for face location. *Proceedings of* 3^{rd} *International Conference on Computer Vision*, pages 718–721, 1990.

[35] J. S. Greenstein and L. Y. Arnaut. *Chapter 22: Input Devices*, pages 495–519. In Helander [36], 1988.

[36] M. Helander, editor. *Handbook of Human-Computer Interaction.* Elsevier Science Publishers, North-Holland, 1988.

[37] J. M. Henderson and A. Hollingworth. Eye movements during scene viewing: An overview. In G. Underwood, editor, *Eye Guidance while Reading and While Watching Dynamic Scenes*, pages 269–293. Elsevier Science Publishers, Oxford, 1998.

[38] J. M. Henderson, Jr. P. A. Weeks, and A. Hollingworth. Effects of semantic consistency on eye movements during scene viewing. *Journal of Experimental Psychology: Human Perception and Performance, Vol. 25*, pages 210–228, 1999.

[39] A. M. F. Heuvelmans, H. E. M. Mélotte, and J. J. Neve. A typewriting system operated by head movements, based on home-computer equipment. *Applied Ergonomics, Vol. 21, No. 2*, pages 115–120, June 1990.

[40] T.E. Hutchinson, Jr. K.P. White, W.N. Martin, K.C. Reichert, and L.A. Frey. Human-computer interaction using eye-gaze input. *IEEE Transactions on System, Man and Cybernetics, Vol. 19, No. 6*, pages 1527–1534, November/December 1989.

[41] R. J. K. Jacob. Eye movement-based human-computer interaction techniques: Toward non-command interfaces. *Advances in Human-Computer Interaction, Vol. 4*, pages 151–190, 1993. [Online] Available `http://http://www.cs.tufts.edu/~jacob/papers.html`.

[42] T. Jebara and A. Pentland. Parameterized structure from motion for 3D adaptive feedback tracking of faces. *Proceedings of Computer Vision and Pattern Recognition*, pages 144–150, 1997.

[43] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, pages 35–45, March 1960.

[44] A. E. Kaufman, A. Bandopadhay, and B. D. Shaviv. An eye tracking computer user interface. *Research Frontiers in VR Workshop Proceedings*, pages 120–121, October 1993. [Online] Available `http://www.cs.sunysb.edu/~vislab/projects/eye/`.

[45] D. E. Kieras and P. G. Polson. An approach to the formal analyss of user complexity. *International Journal of Man-Machine Studies, Vol. 22*, pages 365–394, 1985.

[46] K.-N. Kim and R.S. Ramakrishna. Vision-based eye-gaze tracking for human computer interaction. *IEEE International Conference on System, Man and Cybernetics, Vol. 2*, pages 324–329, 1999.

[47] S. Kimura and M. Yachida. Facial expression recognition and its degree estimation. *Proceedings of Computer Vision and Pattern Recognition*, pages 295–300, 1997.

[48] Jr. K.P. White, T.E. Hutchinson, and J.M. Carley. Spatially dynamic calibration of an eye-tracking system. *IEEE Transactions on System, Man and Cybernetics, Vol. 23, No. 4*, pages 1162–1168, July/August 1993.

[49] Y. Kuno, T. Ishiyama, S. Nakanishi, and Y. Shirai. Combining observations of intentional and unintentional behaviors for human-computer interaction. *Proceedings of CHI—Human Factors in Computing Systems*, pages 238–245, 1999.

[50] Y. H. Kwon and N. da Vitoria Lobo. Age classification from facial images. *Proceedings of Computer Vision and Pattern Recognition*, pages 762–767, 1994.

[51] R. A. Miller. A systems approach to modeling discrete control performance. In W. B. Rouse, editor, *Advances in Man-Machine Systems Research, Vol. 2.* JAI Press, Greenwich, Connecticut, 1985.

[52] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, MA, 1969.

[53] T. P. Moran. The command language grammar: A representation for the user interface of interaction computer systems. *International Journal of Man-Machine Studies, Vol. 15*, pages 3–50, 1981.

[54] C. Morimoto, D. Koons, A. Amir, and M. Flickner. Pupil detection and tracking using multiple light sources. [Online] Available `http://www.almaden.ibm.com/cs/blueeyes/`.

[55] D. P. Mukherjee, A. Zisserman, and M. Brady. Shape from symmetry—detecting and exploiting symmetry in affine images. Technical Report OUEL 1988/93, Oxford University Department of Engineering Science, June 1993.

[56] J. Newell. The division of attention: A social learning approach to the interaction of internet and television use. Technical report, Michigan State University, Telecommunication Department, December 1999. (unpublished).

[57] J. Nielsen. Noncommand user interfaces. *Communications of the ACM, Vol. 36, No. 4*, pages 83–99, 1993.

[58] K. Ohmura, A. Tomono, and Y. Kobayashi. Method of detecting face direction using image processing for human interface. *SPIE Visual Communication and Image Processing, Vol. 1001*, pages 625–632, 1988.

[59] N. Oliver, A. Pentland, and F. Bérard. LAFTER: lips and face real time tracker. *Proceedings of Computer Vision and Pattern Recognition*, pages 123–129, 1997.

[60] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. *Proceedings of Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[61] S. J. Payne and T. R. G. Green. The user's perception of the interaction language: A two-level model. *Proceedings of CHI-Human Factors in Computing Systems*, 1983.

[62] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. *Proceedings of Computer Vision and Pattern Recognition*, pages 84–91, 1994.

[63] D. A. Pomerleau. *Neural Network Perception for Mobile Robot Guidance.* Kluwer Academic Publishing, 1993.

[64] F. K. H. Quek. Unencumbered gestural interaction. *IEEE MultiMedia*, pages 36–47, Winter 1996.

[65] P. Reisner. Formal grammar and human factors design of an interactive graphics system. *IEEE Transactions of Software Engineering, Vol. SE-7*, pages 229–240, 1981.

[66] P. Reisner. Formal grammar as a tool for analyzing ease of use: Some fundamental concepts. *Proceedings of CHI-Human Factors in Computing Systems*, page 53, 1984.

[67] L.G. Roberts. Machine perception of three-dimensional solids. In J.T. Tippet, editor, *Optical and Electro-Optical Information Processing*, Cambridge, MA, 1995. MIT Press.

[68] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review, Vol. 65*, pages 386–408, 1959.

[69] H. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. Technical Report CMU-CS-95-158R, Carnegie Mellon University, School of Computer Science, 1995.

[70] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. *Proceedings of Computer Vision and Pattern Recognition*, pages 38–44, 1998.

[71] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, Cambridge, MA, and London, England, 1986.

[72] T. Sakai, M. Nagao, and S. Fujibayashi. Line extraction and pattern detection in a photograph. *Pattern Recognition*, pages 233–248, 1969.

[73] D.D. Salucci. Inferring intent in eye-based interfaces: Tracing eye movements with process models. *Proceedings of CHI—Human Factors in Computing Systems*, pages 254–261, 1999.

[74] I. Starker and R. A. Bolt. A gaze-responsive self-disclosing display. *Proceedings of CHI—Human Factors in Computing Systems*, pages 3–9, 1990.

[75] R. Stiefelhagen, J. Yang, and A. Waibel. A model-based gaze tracking system. *IEEE International Joint Symposia on Intelligence and Systems: Image, Speech and Natural Language Systems*, pages 304–310, 1996. [Online] Available `http://werner.ira.uka.de/ISL.publications.html`.

[76] H.K. Suen and D. Ary. *Analyzing Quantitative Data.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.

[77] K. Sung and T. Poggio. Example-based learning for view-based human face detection. Technical Report 1521, MIT A.I. Lab, December 1994.

[78] D. Swets and J. Weng. Using discriminant eigenvectors for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 8*, pages 831–836, August 1996.

[79] K. Talmi and J. Liu. Eye and gaze tracking for visually controlled interactive steroscopic displays. *Image Communication*, 1998. [Online] Available `http://atwww.hhi.de:80/~blick/Papers/papers.html`.

[80] J.-C. Terrillon and S. Akamatsu. Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images. *Proceedings of Vision Interface*, pages 180–187, 1999.

[81] K. Toyama. Head parallax tracking for control of a virtual space: a comparison of algorithms. *IEEE International Conference on System, Man and Cybernetics, Vol. 6*, pages 1–6, 1999.

[82] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience, Vol. 3, No. 1*, pages 71–86, 1991.

[83] R. Uhl and N. da Vitoria Lobo. A framework for recognizing a facial image from a police sketch. *Proceedings of Computer Vision and Pattern Recognition*, 1996.

[84] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localization of objects in images. *IEEE Proceedings on Vision, Image and Signal Processing* 141 (4), August 1994.

[85] M. T. Vo and A. Waibel. Multimodal human-computer interaction. *Proceedings of ISSD*, 1993. [Online] Available `http://werner.ira.uka.de/ISL.publications.html`.

[86] C. Ware and H. H. Mikaelian. An evaluation of an eye tracker as a device for computer input. *Proceedings of CHI—Human Factors in Computing Systems*, pages 183–188, 1987.

[87] M. Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM, Vol. 36, No 7*, pages 74–83, July 1993. In Special Issue, Computer-Augmented Environments.

[88] M. Weiser. Hot topic: Ubiquitous computing. *IEEE Computer*, pages 71–72, October 1993.

[89] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 1995. [Online] Available `http://www.cs.unc.edu/~welch/`.

[90] A. T. Welford. *Fundamentals of Skills*. Methuen, London, 1968.

[91] X. Xie, R. Sudhakar, and H. Zhuang. A cascaded scheme for eye tracking and head movement compensation. *IEEE Transactions on System, Man and Cybernetics, Vol. 28, No. 4*, pages 487–490, July 1998.

[92] Y. Yacoob and L. Davis. Computing spatio-temporal representations of human faces. *Proceedings of Computer Vision and Pattern Recognition*, pages 70–75, 1994.

[93] G. Yang and T.S. Huang. Human face detection in a complex background. *Pattern Recognition, Vol. 27, No. 1*, pages 53–63, 1994.

[94] J. Yang, W. Lu, and A. Waibel. Skin-color modeling and adaptation. *Proceedings of Asian Conference on Computer Vision, Vol. II*, pages 687–694, 1998. [Online] Available `http://werner.ira.uka.de/ISL.publications.html`.

[95] J. Yang, R. Stiefelhagen, U. Meier, and A. Waibel. Visual tracking for multimodal human computer interaction. *Proceedings of CHI—Human Factors in Computing Systems*, pages 140–147, 1998.

[96] J. Yang and A. Waibel. A real-time face tracker. *Proceedings of WACV*, pages 142–147, 1996. [Online] Available `http://werner.ira.uka.de/ISL.publications.html`.

[97] A. L. Yarbus. Eye movements during perception of complex objects. In L. A. Riggs, editor, *Eye Movements and Vision*, pages 171–196. Plenum Press, New York, 1967.

[98] R. M. Young. Surrogates and mappings: Two kinds of conceptual models for interactive devices. In D. Gentner and A. Stevens, editors, *Mental Models.* Erlbaum, Hillsdale, New Jersey, 1983.

[99] A. L. Yuille, D. S. Cohen, and P. W. Hallinan. Feature extraction from faces using deformable templates. *Proceedings of Computer Vision and Pattern Recognition*, pages 104–109, 1989.

[100] S. Zhai, C. Morimoto, and S. Ihde. Manual and gaze input cascaded (magic) pointing. *Proceedings of CHI—Human Factors in Computing Systems*, pages 246–253, 1999.