6)

Perceptive Agents with Attentive Interfaces: learning and vision for man-machine systems

by

Trevor Jackson Darrell

B.S.E, University of Pennsylvania (1988)
S.M., Massachusetts Institute of Technology (1991)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Author .....
Program in Media Arts and Sciences
May 3, 1996

Certified by .....
Alex P. Pentland
Associate Professor of Computers, Communication, and Design Technology
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by .....
Stephen A. Benton
Chairman, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Perceptive Agents with Attentive Interfaces: learning and vision for man-machine systems

by

Trevor Jackson Darrell

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on May 3, 1996, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis address the issue of how to integrate action and perception in changing, complex environments which include people. We consider the development of perceptive agents, or more generally perceptual computer interfaces, which can directly respond to a user's state, including body pose and hand or face gestures. This domain presents challenging machine vision and learning problems, which must be solved in real-time to be useful as an interface. We present methods for tracking an unconstrained user moving about a cluttered office environment, and methods for recognition and interpolation of spatio-temporal hand and face gestures using images obtained from an active, moving camera. To guide the active camera, we adopt a model of visual attention based on the partially observable Markov decision process (POMDP), which formalizes the notion of action selection given perceptual input, i.e, hidden state. We implement an attention system based on hidden state reinforcement learning, which solves an active recognition task formulated as a POMDP. This system has been integrated into our interactive office environment, and can selectively track a user's head, hands, or other salient features as they wander about.

Thesis Supervisor: Alex P. Pentland
Title: Associate Professor of Computers, Communication, and Design Technology
Program in Media Arts and Sciences

Doctoral Committee

Thesis Advisor .....

Alex P. Pentland
Associate Professor of Computers, Communication, and Design Technology
Program in Media Arts and Sciences

Thesis Reader .....

Aaron Bobick
Assistant Professor of Computational Vision
Program in Media Arts and Sciences

Thesis Reader .....

Rodney Brooks
Professor of Computer Science and Engineering

Thesis Reader .....

Pattie Maes
Associate Professor of Media Technology
Program in Media Arts and Sciences

Acknowledgements

I first and foremost want to thank, and dedicate this thesis, to my parents. Their love and support are the essential part of all my successes, past, present, and future. To my advisor, Sandy Pentland, I owe the greatest thanks for the thesis itself. His years of adroitly guiding me to interesting problems and solutions have been incredibly valuable, productive, exciting, and fun. I count him as a true mentor, fellow "hacker", and friend, both before this thesis and after.

A special thanks to Jody Pringle, who gave me both warmth and wit during the completion of this work, and showed me with her example that it is possible to finally finish! To my closest friends in Cambridge and elsewhere, thanks for putting up with me during the final months, and for providing years of distractions and entertaining repartee that kept me thinking about topics beyond computer vision!

I want to thank my colleagues in the vision group at the Media Lab, past and present (including Marty!), who created an intellectual and computational infrastructure that made this such an amazing place to do research. Also thanks to those who put up with me in close quarters, my officemates, especially Stan Sclaroff who put up with years of close-the-door gripe sessions and disproportionate consumption of office supples, with always a laugh if not a challenging reply.

Finally I'd like to thank Aaron Bobick for his insight and advice during the last year, and to my first mentor in the field, Ruzena Bajcsy, who introduced me to the quirky pleasures (and frustrations!) of research in vision, and who first brought my attention to the promise of active vision methods, a topic that I continue researching on today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As found in nature, vision is not a passive process. It is both *active*, in that visual sensors are actuated to change the imaged scene, and *reactive*, since the control signals which govern those actuators are based on environmental stimuli. When the environment changes quickly and itself has an active component, a model which captures the dynamic interplay between the environmental stimuli and the perceptual system is needed. We seek a model of this type of *interactive vision*.

Many practical vision problems require the use of active methods in rapidly changing environments. The well-studied domain of automated navigation and vehicle control, where automobiles are autonomously or semi-autonomously guided in real-time, is a good example of an environment which can be neither absolutely nor statically described, given the time constraints of performance. We will focus on a domain which similarly requires real-time, context-dependent processing, the domain of interacting with people.

A system for interaction with people calls for active behaviors due to the unpredictable nature of the domain. People are quite difficult to predict or model exactly, and so visual methods for perceiving people need to be adapted and selected according to the current context. Depending on the state of the user a different suite of visual routines or models

may be appropriate for processing, and a perceptual system should thus adapt processing based on prior experience with the user.

Most significantly, the spatial scale over which human interaction takes place is large – a wide field of view is needed to first distinguish a human form and recognize overall pose, but a narrow field of view is needed to discriminate fine gestures or expressions performed by a hand or face. Since it is prohibitive (in either silicon or neuronal terms) to have a single vision sensor which has high-resolution over a wide field of view, a foveated vision sensor and associated active vision/attention mechanism for governing eye movements is needed.

Developing a computational framework for this type of visual attention – a mechanism to guide what to look for, and when – is an important problem. Attention is a critical component of both the human visual system, which can create a unified experience from a series of saccadic eye movements and disjoint foveated views, and for contemporary models of machine vision, which must direct limited computational resources in order to achieve real-time performance. In addition to an explicit spatial "window of attention", attention is manifest in other dimensions as well: one can attend to particular spatial frequencies, chromatic characteristics, or even spatial arrangements. Broadly speaking, we define attentive behaviors are those which affect the current context of visual processing, either explicitly as through a active camera, or implicitly by changing the parameters of filtering or model selection performed in early stages of visual processing.

Our goal in this thesis is to implement a mechanism which can perform attentive behaviors in the context of interacting with people. Practically, we wish to find a mechanism which can offer high-resolution, foveated images of parts of the body that are important for gesture understanding, such as hands and faces. This mechanism should also ultimately support perception for interactive dialog, guiding visual processing of a human user interacting with a conversational agent.

A rich model of attention needs to balance sensor/data-driven and goal-based processing, provide context for evaluation of situated routines, and offer an action-selection method

for executing active visual processes. An attention system needs to mediate between pure perception and pure action, and to do so contingent on the current changing environmental context. The key algorithmic aspect to this type of attentive, and interactive, vision is the arbitration, selection, and scheduling of a set of resource-limited visual routines based on observations that are both noisy and incomplete. In this thesis, we will discuss how this can be captured in formal models of behavior and motor control, and describe their application in the domain of interactive systems. In particular, we will develop a model of action selection for attentive visual behaviors based on a model of Markovian decision processes with hidden state. As we shall see, this offers a balance between approaches which assume significant prior knowledge but can handle partial observability, such as the Kalman filter [40], and model-free approaches which learn a model of the world (and by extension, of the interaction with the user) but assume an absolute state representation.

1.1 Models of Dynamic Vision

To model the dynamic aspects of attention, it is natural to look to models traditionally used for motor control and planning. However, it is important to choose a level of representation which is appropriate for the task or system we are attempting to characterize [15]. Since attentive behavior requires models which can adapt quickly to changing conditions, this can be problematic.

Control-theoretic approaches in the computer vision literature have been a topic of increasing recent interest for guiding low-level active vision [12, 37]. When the task is complicated, and involves rapidly changing state, representations, and goals – as is the case with attention – conventional approaches to control often break down. To accommodate changing representation or mode, multiple control models or regimens are needed in the model of attention. But when there are many different models that may potentially be present, it is important to have some way of predicting which are most likely to be

transitioned into next. Otherwise, they will all have to be evaluated, at a potentially large computational cost. If the probabilities of inter-model transfer are known, they can greatly lower the computational cost of evaluating all the likely models to be seen at the next time step. This is the essence of attentive processing.

This approach can be formalized with the notion of a Markovian state model, where multiple models correspond to multiple states, and the likelihood of state transitions are represented explicitly. In domains where a state-based representation is appropriate, Markov Models can provide the contextual structure needed for multiple model recognition.

For active vision and attention, the Markovian approach can be used to model what actions should optimally be taken according to both observations of external conditions and the current representation of internal state. This type of *Markov Decision Process* provides a model for action as well as observation, and has been applied extensively in the Operations Research and Robot planning literature [71]. As a model of perceptual processing, however, strict Markovian Models are limited by the assumption of observable state. Perception problems are characterized almost by definition as being "ill-posed", in that the underlying state of the world is not directly recoverable without further assumption. In the context of Markov Models, this is termed a hidden state problem, since the true state of the world is hidden from the model. For passive perception problems, the Hidden Markov Model (HMM) addresses this, and has been used successfully for speech, handwriting, and vision recognition tasks [64, 68]. To model active perception, Markov Decision Processes need to be extended to accommodate hidden state.

A Markov Decision Process without direct access to the state of the environment is called a *Partially Observable Markov Decision Process* (POMDP). POMDPs were developed in the Operations Research literature [72, 84, 48], and have recently been introduced to the field of Artificial Intelligence for active planning[53, 46, 21]. The POMDP approach holds promise as a model for reactive behaviors in a perceptually realistic environment.

A related approach to modeling visual attention is the use of Bayes networks with

decision theoretic criteria. These methods have much to offer from a representational perspective, but the problem of learning their structure from experience is unsolved [69]. POMDP models provide a general framework for modeling action in the context of perception with hidden state, and combine the three essential components of a model of attention and/or interactive vision: they have a model of action, they have a model of perception (state is hidden), and they have a method for learning. In addition, they are built on a well-established literature on statistical models of Markovian Decision Processes. Because of these qualities, we feel POMDPs will be very useful as a model of attentive behaviors within an environment for interacting with virtual agents.

1.2 Perception and Interaction

Previously we developed an interface to an interactive environment called the ALIVE system, an acronym for "Artificial Life Interactive Video Environment" [30]. The system presents a simulated mirror in the form of a large video screen, in which the user sees him/herself immersed in a graphical environment inhabited by virtual creatures. The image of the user is captured by a video camera, combined with the output from a computer graphics workstation, and displayed on the screen in front of the user. Computer vision techniques analyze the form of the user and compute his/her location in 3-D so as to afford proper depth compositing, as well as the location of the user's head, hands, and other body parts in order to determine whether the user is performing a known gesture.

The previous ALIVE system employed a stationary camera, and little attentional or active vision.[1] In this thesis we add active and attentional modes of processing to the ALIVE system, using POMDP models to control which visual routines/models are selected

[1]The vision architecture as originally implemented supports a primitive attention mechanism, in that the various processing routines could be switched on or off, but the higher-level control routines did not take advantage of that capability.

and when they are executed. Foveated perception is explicitly provided, through the addition of a motorized camera with a narrow-field of view lens.

Active, foveated vision has two main implications for the ALIVE system. First, it allows more detailed processing to be performed on regions of a users image for which resolution is critical: hands, faces, etc. The current resolution of a users face in the ALIVE system is too coarse for all but the simplest face recognition methods, and is insufficient for detailed tracking of gestures or expressions. With a foveated sensor, view-based models of hand and face gesture recognition, as well as other recognition techniques that require high-resolution, foveated images, have been integrated into the ALIVE environment.

Second, and perhaps more important, the availability of foveated vision creates a limited resource within the vision system that necessitates attentional modes of behavior. Previously in the ALIVE system there was an option for attentional behavior, but since all vision routines could be run simultaneously at little or no time penalty, there was no incentive to actually use attentive behaviors. However a foveated sensor is obviously a limited resource, and no universal (non-attentional) mode of driving it is feasible. (E.g. it would be nonsensical to use a foveated camera as a raster scan input device.) Creating a system which can perform resource allocation in a timely and productive manner, e.g. deciding what to look for when, yields insights into the combination of behavior with perception, and provides a system which can offer practical advances in the kinds of interaction with people it can provide.

The major application of attentive processing developed in this thesis is a system for active gesture recognition. Gestures are typically performed at multiple spatial scales and are dependent on temporal context. An example of the former is recognition of a grasping gesture where analysis on the scale of body parts is first needed to track an extended arm, followed by a finer scale analysis of hand pose to determine when (and if) a user clasps his or her hand. The latter occurs whenever a gesture is not defined by a set of static spatial features, but requires transitioning through a particular temporal sequence of spatial states.

The combination of these two characteristics calls for modeling with POMDPs. If temporal context was the only confounding factor a HMM would be adequate to perform recognition. Indeed HMM's have been applied to considerable success in recognizing signals with temporal structure, such as speech, and recently have been used for recognizing visual patterns at fixed resolution. But the presence of spatial context calls for active methods, since we need to shift the window of processing across different scales and/or spatial locations. This can be either a overt or covert process, the former being implemented via a set of actuators on a moving camera and the later being implemented virtually as a selection and subsampling from a single very high-resolution image.

This thesis formulates a model of attentive behavior in the interactive domain based on partially observable Markov models. The utility of this approach is demonstrated through the implementation of a system for multi-scale foveated gesture recognition using a reinforcement-based learning algorithm. By utilizing a learning method to compute a perceptually driven action selection policy, we show that attentive behaviors useful for interactive systems can be found.

In Chapter 2, we will describe in detail our interactive interface domain, and discuss the implications it places on the construction of a set of person tracking routines. Chapter 3 will then describe a method for hand and face gesture analysis which yields accurate results in real-time, but which requires high-resolution imagery. Chapter 4 will present a method to perform these methods in the interactive domain, using an active camera to obtain foveated images of the user's hand or face. Finally Chapter 5 will present our attention framework for deciding which feature of the user to look at, using a model of action selection based on POMDPs and an active recognition task.

Chapter 2

Interactive Interfaces

Vision has numerous uses in the natural world. It is used by many organisms in navigation and object recognition tasks, for finding resources or avoiding predators. Often overlooked in computational models of vision, however, and particularly relevant for humans, is the use of vision for communication and interaction between individuals. In these domains visual perception serves as an important modality either in addition to language or in conditions where language cannot be used. In most settings, people place considerable weight on visual signals from another individual, such as facial expression, hand gestures, and body language.

Models of machine perception have, in general, been designed with the goal of performing recognition and navigation tasks. While these are important and challenging problems, we feel that the domain of interactive interfaces can offer new insights into models of machine perception. In particular, we will argue that this domain has constraints which call for models of visual perception that are both active and adaptive, since conventional models cannot capture the dynamic and contextual aspects of people's visual language.

Vision as an interface modality offers a range of potential new applications which are of increasing demand given the current climate of expanding sources of information

Figure 2-1: Binary silhouettes of users after figure-ground processing.

without a similar expansion in the nature of the interface to that information (i.e., click and type interaction.) For many users and many applications, keyboards, mice, or even wired gloves or goggles are inappropriate tools, for either physical or sociological reasons. In this chapter we will discuss the interactive interface domain as it relates to models of visual perception, present the basic algorithms we use for constructing an interactive vision system, and discuss three applications for human-computer interaction.

2.1 Implications

The domain of interacting with people provides several challenges to the designer of a vision system. People are dynamic, have intentions and motivations, and have articulated body kinematics as well as non-rigid motions in facial expressions. All of these are difficult issues for conventional computer vision methods. We will discuss three of these issues: the difficultly in modeling human forms, semantic context in interactive systems, and temporal constraints on interactive performance.

The forms and motion of human bodies are complicated and can be hard to characterize with precision. This means that methods which process static images of humans have to deal with a wide range of potential shapes or patterns. For example, Figure 2-1 shows a set of bitmap patterns of users of our system. A descriptive model of these forms would require an articulated body model that included a non-rigid model of skin and clothing dynamics.

Common methods for recovering these models use motion information and often assume motion obeys an affine flow model or a rigid-body constraint, neither of which seem to hold in this case. For both static and dynamic processing, the people-watching domain is a challenge in that it calls for finding methods which work robustly in the absence of a strict model.

The fact that people are not simply objects, and have intentions and communicate via semantically-laden signs, has interesting implications for a vision system. Traditional domains such as navigation and object recognition have the property that problems within the domain can be well described as the estimation of objective properties of the world for which partial information is available. In these domains engineering estimation principles which require an objective "ground-truth" can be straightforwardly be applied. Unfortunately, when vision is considered in the domain of communication and interaction there is no concrete notion of ground-truth to use as a problem formulation. When people use vision to look at other people, the most important features they extract are *semantic*, in that they have a meaningful or emotional content. People can use vision in interpersonal communication to explicitly attend to and remember 3-D shape or location of the other person's body, but that is not the norm, nor is it necessarily of great utility. To develop interfaces which allow people to interact with computers as easily as they interact with other people requires placing the emphasis on semantics rather than shape.

Communication implies context, so a purely descriptive approach to vision is clearly inappropriate – the goal of vision routines for an interactive people-watching system should not be to perfectly estimate and represent the three-dimensional shape of the body and face. Vision routines for an interactive people-watching system should recover and provide some *meaningful* signal in the context of the current interaction. What exactly is meaningful will change over time, so no static description would suffice. Rather than attempt to fully describe the 3-D world before interpretation, the task-based or "purposive" vision paradigm [2] sidesteps the recovery of a 3-D representation, and advocates computation

which directly achieves some goal given the visual input. They and others argue that the process of recovering a full 3-D representation is a computationally inefficient and often unnecessary stage, when the larger task is properly considered [4, 5].

Finally, the presence of people in an interactive system provides a strong pressure to achieve real-time performance. Quite simply, if the system does not react in real-time, or perhaps what is more appropriately called "interactive-time", the user will get bored and leave. Unlike many static domains in which the agent can stop momentarily if necessary to make a decision, the visual routines and agent models used in an interactive man-machine system must be both robust and fast.

For these reasons, successful vision methods for interactive interfaces must be robust, semantically-situated, and active. To achieve these goals, we have developed a specific interaction paradigm, described in the following section, that allows simplified processing of the human form and yet offers a wide range of interaction. We have focused on the implementation of a set of real-time visual routines which make robust estimates of the user's position, pose, and/or gestures, and allow interaction with a set of virtual agents.

2.2 The Magic Mirror

We have chosen to explore an interaction domain in which the user faces a wall-size (but fixed) display which contains cameras that observe the user. Computer-generated graphics and video images are presented on the display, along with a graphical representation of the user. The cameras are connected to a vision system which analyze in real time the state and location of the user, and update the virtual representation accordingly. This representation may consist of the user's digitized image, a 3-D model of a person, a model of a graphical cartoon character, or a combination of all of these. Objects (or agents) in the virtual world can use the vision system to "see" the user, who can in turn see the graphical representation of the objects on the display.

This "magic-mirror" paradigm is attractive because it provides a set of domain constraints which are restrictive enough to allow simple vision routines to succeed, but is sufficiently unencumbered that can be used by real people without training or a special apparatus. These constraints derive from the fact that we can arrange the imaging geometry of the camera such that the user is almost always in a frontal pose. In our system, the same camera used for vision processing is also used for acquiring the image of the user which is composited into the graphics display. For the "magic mirror" effect to work, the image of the user used for the display must come from a camera position which is located approximately at the position of the screen. Since in this paradigm the user will be watching the screen almost continuously, we can assume with some degree of confidence that they will face the screen and thus their body will be oriented parallel to the screen much of the time. When this is true, we can make relatively strong inferences about the position of the user's head and hands.

We implement the magic-mirror model using a single CCD camera to obtain a color image of the scene. The image of the user is separated from the background, composited into the 3-D graphical world, and projected onto a large screen which faces the user. (The polarity of projection is reversed so that the image appears as it would in a mirror.) Vision routines are run on the image to allow the user to interact with the virtual world. The entire system, including vision, animation, and rendering, occurs in real-time (10Hz) so that an interactive experience is preserved.

2.3 Related Work on Interfaces to Virtual Environments

A wireless sensor, such as vision, has several additional advantages over tethered goggles-and-gloves interfaces to virtual environments. It provides a safer solution because the user can still see where he or she is moving, and thus can avoid bumping into things, or tripping over wires. The also user enjoys greater behavioral and expressive freedom. We

observed that users of the Magic Mirror system feel very uninhibited (we have have seen users doing cartwheels, jumping jacks, etc). Finally, the user ends up concentrating more on the environment itself, rather than on the complex and unfamiliar equipment being used to interact with that environment.

Other systems, such as the Visual Portal [32] and the CAVE [24] system have solved many of the limitations of traditional goggle-based environments through the use of wireless batons and other sensors, thus avoiding both the problems of a tethered display and viewpoint estimation (head angle). Our system has the advantage that it is completely unencumbered, and works on users with no special tools or marks. We also adopt a mirror paradigm, where the user explicitly sees a representation of him/herself and his/her relationship to other objects in the world.

The novel vision-based interface presented here was inspired by the pioneering work of Myron Krueger's Videoplace system [45]. The Magic Mirror and Videoplace differ primarily in three respects. The first is that Videoplace focuses on 2-D rather than 3-D worlds and interaction. A second difference is our emphasis on modeling agents. Most of Krueger's worlds allow users to interact with other users, a notable exception being the "critter", a 2-D animated sprite. Finally, the vision system is able to recognize hand and body gestures as patterns in space and time.

Another system that bears similarities to the Magic Mirror is the Mandala system [80] which composites the user's color image with a virtual world that is sometimes video-based and sometimes computer animated. The Mandala system only supports 2-D and requires a chromakey background or specially-colored manipulation objects; it does not attempt to recognize parts of the user's figure nor does it do any gesture recognition. Other systems have been developed for vision-based interactive graphics but have generally been restricted to off-line analysis of either face or limb motion [35, 87, 75]. (But see [28] for a real-time facial analysis system.)

2.4 Vision Routines for Person Tracking

We have developed a set of vision routines for perceiving body actions and gestures performed by a human participant in an interactive system.[1] Vision routines acquire the image of the user, compute a figure/ground segmentation, and find the location of head, hands, and other salient body features (Figure 2-2). We use only a single, calibrated, wide field-of-view camera to determine the 3-D position of these features. We do assume that the background is fixed, although it can be arbitrarily complex, and that the person is normally facing the camera/screen. The integration of the person and and localization of his/her head or hand features in the world are performed using the following modules: figure-ground processing, scene projection, hand tracking, and gesture interpretation.

2.4.1 Figure-ground processing

To detect appropriate hand/face features and composite the user's image onto the magic mirror, the vision system must isolate the figure of the user from the background (and from other users, if present). This is accomplished by use of spatially-local pattern recognition techniques to characterize changes in the scene, followed by connected-components and morphological analysis to extract objects.

We assume the background to be an arbitrary, but static, pattern. Mean and variance information about the background pattern are computed from an initial sequence of images with no person present, and these statistics are used to determine space-variant criteria for pixel class membership. In general, we use a hierarchical color classification is used to compute figure/ground segmentation, using a Gaussian model of each background pixel's color and an n-class adaptive model of foreground (person) colors. The classification takes care to identify possible shadow regions, and to normalize these region's brightness before

[1] The first version of these routines was implemented by the author [30]; more recent versions have been implemented in collaboration with Chris Wren and Ali Azarbayejani [88].

the figure/ground classification. The classification also makes use of Markov neighborhood statistics in setting the priors for each pixel's classification [88].

2.4.2 Scene projections and calibration

Once each pixel has been identified as most likely belonging to the user, we use connected components and morphological analysis to delineate the foreground region. This analysis begins with a seed point at the centroid location of the person in the previous frame; if this fails to grow a sufficiently large region, random seed points are selected until a stable region is found. Finally, we compute the contour of the extracted region by chain-coding the connected foreground region.

When the figure of the user has been isolated from the background, we compute an estimate of its 3-D location in the world. If we assume the user is indeed sitting or standing on the ground plane, and we know the calibration of the camera, then we can compute the location of the bounding box in 3-D. Establishing the calibration of a camera is a well-studied problem, and several classical techniques are available to solve it in certain broad cases [6, 41]. Typically these methods model the camera optics as a pinhole perspective optical system, and establish its parameters by matching known 3-D points with their 2-D projection.

Knowledge of the camera geometry allows us to project a ray from the camera through the 2-D projection of the bottom of the bounding box of the user. Since the user is on the ground plane, the intersection of the projected ray and the ground plane will establish the 3-D location of the user's feet. The 2-D dimensions of the user's bounding box and its base location in 3-D constitute the low-level information about the user that is continuously computed and made available to all agents in the computer graphics world. The contour is projected from 2-D screen coordinates into 3-D world coordinates, based on the computed depth location of the person. This is then used to perform video compositing and depth

clipping to combine the user's video image with computer graphics imagery.

2.4.3 Hand tracking and gesture interpretation

One of the most salient cues in an interactive interface is the location of the user's hands. We have implemented feature localization heuristics that determines hand locations by searching within a window along the side of the contour for extremal horizontal and vertical points. If the highest point in the window is above the shoulder of the user, we label that the hand, otherwise the horizontal extremal point is used. The highest point within a window of the contour located above the centroid of the foreground region is labeled the head. These feature localization algorithms are not infallible, but we have found they work well in a wide range of conditions, especially if combined with color space classification to identify the location of flesh tones[88].

Our system improves on earlier systems in which only the 2-D position of the user's hand was used to determine activation of objects such as virtual buttons. The improvements avoid inadvertent manipulation of objects, such as unintended activation of buttons. The system uses combination of clues including 2-D position of the hands, Z position of the user's body, and gesture information to make sure that the user's intention is to actually manipulate an object. For example, in order for the 3-D button to be pushed, the user has to perform a "pointing gesture", have the hand over the button in 2-D and the user's feet have to be placed in the correct Z-plane.

Both the absolute position of hands, and whether they are performing characteristic gesture patterns, are relevant to the agents in the virtual world. We use pattern recognition strategies to detect and classify these characteristic gesture patterns. Static gestures, such as pointing, are computed directly from the hand feature location. To recognize dynamic gestures, we use a high-resolution, active camera to provide a foveated image of the hands (or face) of the user. The camera is guided by the computed feature location, and provides

images which can be used successfully in a spatio-temporal gesture recognition method. This framework for real-time gesture processing and active camera control is described in detail in the following chapters.

2.5 Applications of the Magic Mirror

Several applications have been implemented which use the full-body, interactive vision metaphor described above. Here we describe three applications: interacting with autonomous virtual agents, browsing a multimedia database using natural gestures, and interactive game experiences.

2.5.1 Perceptually situated intelligent agents

The first implementation of an interactive vision environment was a system designed to allow a user to interact with an immersive visual environment of artificial life agents, without using any physical apparatus. Our system, ALIVE, or "Artificial Life Interactive Video Environment" uses the vision routines described above, together with behavior-based animation and agent modeling systems.[2] With few exceptions (e.g. [45]), to experience these environments previously required the use of gloves, goggles, and/or a helmet, and most likely a wired tether to a computer graphics workstation [67].

The initial ALIVE system [30] contained of two virtual worlds, which the user could switch between by pressing a virtual 3-D button. One world was inhabited by a Puppet and the other world by a Hamster and a Predator. The Puppet had behaviors to follow the user around, try to hold the user's hand, and imitate some of the actions of the user (sitting down, jumping, etc). It would be sent away when the user pointed away and come back when the

[2]The behavior system for the Hamster and Dog characters in ALIVE was implemented by Bruce Blumberg [13, 14], and the Puppet chacter by Jeremey Brown, under the supervision of Pattie Maes.

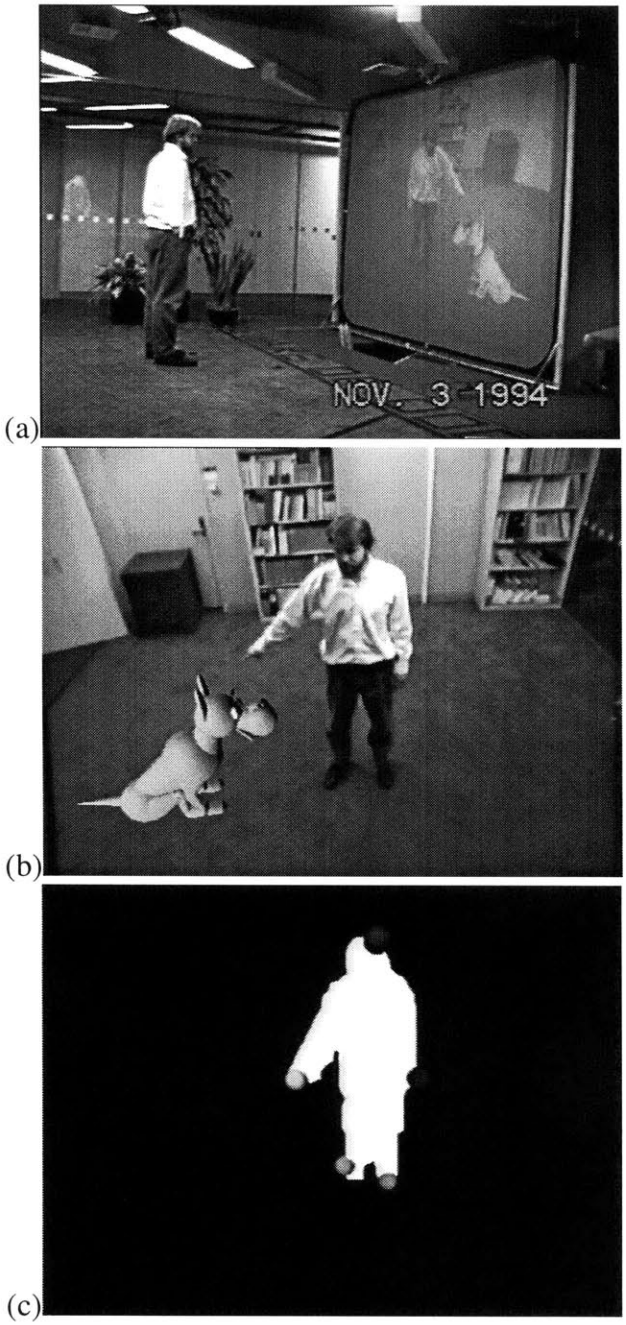


Figure 2-2: Person tracking in a system for vision-based interaction with a virtual environment. (a,b) A user sees him/herself in a "magic mirror", composited in a virtual environment. Computer vision routines analyze the image of the person to allow him/her to effect the virtual world through direct manipulation and/or gestural commands. (c) Results of feature tracking routine; head, hands, and feet are marked with color-coded balls.

Figure 2-3: Gestures are interpreted by the agents based on the context. Here, the Puppet walks away in the direction the user is pointing.

user waved. The puppet employed facial expressions to convey some of its internal state. For example, it would pout when the user sent it away and smile when the user motioned it to come back. It giggled when the user would touch its belly.

Similarly the Hamster had behaviors to avoid objects, follow the user, and to beg for food. The user was able to feed the Hamster by picking up food from a virtual table and putting it on the floor. The user could open an adjoining cage and release a Predator, which would then chase the Hamster (but avoid the user).

The user could interact with the agent using certain hand gestures, which were interpreted in the context of the particular situation. For example, when the user points away (Figure 2-3) and thereby sends the puppet away, the puppet will go to a different place depending on where the user is standing. If the user waves or comes towards the puppet after it has been sent away, this gesture is interpreted to mean that the user no longer wants the puppet to go away, and so the puppet will smile and return to the user. In this manner, the gestures employed by the user can have rich meaning which varies on the previous history, the agents internal needs and the current situation.

More recent ALIVE worlds have focused on artifical agents of increasing complexity.

In the ALIVE system presently demonstrated at MIT, users interact with an autonomous dog agent who would follow, beg, mimic, and play fetch with the user, as well as other behaviors. The dog agent was comprised of an ethologically based action selection mechanism that choose behaviors for execution based on both perceptual input provided by the vision system and internal state/goals [13, 14].

2.5.2 Multimedia Navigation

Given the trend that dramatically increasing amounts of information are available through publically accessible computer networks, it is important to develop methods to better access and manipulate that information. In cases where traditional keyboard interfaces are inadequate due to public access constraints or naive users, an interactive vision-based interface has several desirable properties: it is passive, non-intrusive, and responds to a person's natural gestures.

The vision-based magic mirror interface can be used in an interactive multimedia browsing task. Using the same physical installation and vision routines as above, Wren, Sparacino, and colleagues [89] have constructed a system using the Magic Mirror interface in which the user could navigate a space of video, sound and text objects. Objects could change state – display at a higher resolution, add more detailed description, and/or show pointers to background information – based on the proximity of the user in the 3-D virtual space and or whether the user was gesturing at the object. Most recently they have connected this system to a WWW navigator (Figure 2-4(a)).

2.5.3 Interactive Video Games

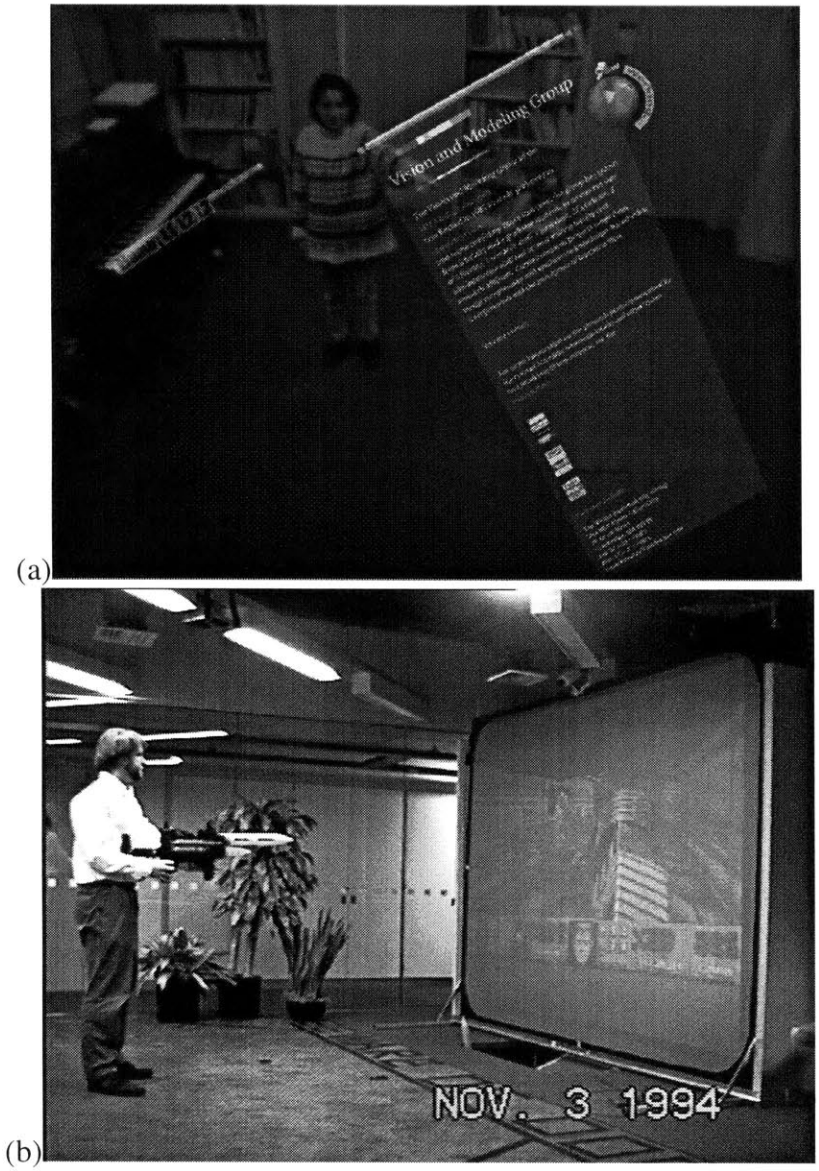
Perception of a user's pose using computer vision has direct application as an interface to video games [80]. Rather than use buttons or dials to move a game figure, a game character can be directly or indirectly controlled using the user's own body. This can result in a more

visceral, and athletic, experience for the game player. At the Media Lab, K. Russell, T. Starner, and C. Wren have implemented an interactive full-body interface to the popular DOOM video game (Figure 2-4(b)). DOOM is a first person multi-player game in which the goal is to explore a 3-D maze, fight and kill any monsters or opposing players, and obtain treasure. Traditional interfaces to this game are all based on PC keyboards, which are relatively cumbersome to manipulate. Using the full body interface, body position and gesture are used to control the game. Navigation is performed using a mixture of first and third person interface metaphors: translation is determined from the position of the user in the 3-D space, rotation adjusted based on pointing to one side or the other, and gun control based on either sound input commands or two handed gestures.

2.5.4 Other Applications

Many other applications could be adapted to this type of interface; for example an interactive aerobics trainer, which, unlike the ubiquitous home video tape, would not only lead the user through a routine, but also watch to see if the user is successfully completing the workout and adjust the pace if necessary. In addition, when connected to a wide area network this interface can be naturally applied to the domain of teleconferencing and telepresence, allowing the user to control a full-body representation of his or her body in a shared virtual world.[3]

[3] A first experiment along these lines was the distributed ALIVE system shown at SIGGRAPH96, see [31].



(a)

(b)

Figure 2-4: Examples of interactive video games and web navigation using the full-body interface: (a) web navigation using real gestures, (b) SURVIVE, a full-body interface to the DOOM video game.

Chapter 3

Real-time Hand and Face Gesture Analysis

Gestures are an important aspect of human interaction, both interpersonally and in the context of man-machine interfaces. There are many facets to the modeling and recognition of human gesture: gestures can be made by hands, faces, or the entire body, they can be static or dynamic, person-specific or cross-cultural, tied to linguistic utterances or meaningful in their own right. Here we consider vision-based analysis of user-specific *spatio-temporal* gestures, which can be described as a set of poses or expressions observed over a finely sampled temporal sequence. We will focus on methods which learn a task-specific representation of hand and face gestures for use in real-time recognition and tracking tasks.

The analysis presented in this chapter assumes high-resolution imagery of the hand or face is available; Chapter 4 will show how we use an active camera to obtain foveated imagery of hands or faces in an unconstrained environment, and utilize estimates of pose or expression in an interactive interface.

Our system achieves robust real-time performance in gesture analysis by exploiting the principle of using only as much "representation" as needed. Hands and faces are complex

3D articulated structures, whose kinematics and dynamics are difficult to model with full realism. Consequently, instead of performing model-based reconstruction and attempting to extract explicit 3D model parameters (for example see [23, 39, 43]), we use a direct approach which represents the object performing the gesture with a vector of similarity scores to a set of 2-D views. With this approach we can perform recognition and tracking on objects that are either too difficult to model explicitly or for which a model recovery method is not feasible in real time.

As we shall see below, our appearance-based approach affords several advantages, such as the ability to form a sparse representation that models only the poses of the hands that are relevant to desired gestures, and the ability to learn the models directly from the data using unsupervised clustering. We combine the dimensionality reduction offered by appearance-based analysis with a supervised learning interpolation stage that maps view model outputs into a task dependent coordinate system, in which recognition and interactive control are straightforward.

3.1 Appearance-based Representation

We adopt a appearance-based representation of gesture performance, where the appearance of a target object (e.g., the face or hand) is described by its similarity to a set of iconic views that span the set of poses and configurations of the target object.

This approach is related to the idea of view-based representation, as advocated by Ullman [79] and Poggio [63], for representing 3-D objects by interpolating between a small set of 2-D views. Recognition using views was analyzed by Breuel, who established that there are reasonable bounds on the number of views needed for a given error rate [16]. However, the view-based models used in these approaches rely on a feature-based representation of an image, in which a "view" is the list of vertex locations of semantically relevant features. Unfortunately, the automatic extraction of these features remains a difficult problem.

More closely related to our approach is the work of Turk and Pentland[78], who used combinations of low-order eigenvectors to describe a *space* of target appearances. In this way they were able to detect and recognize human faces. Murase and Nayar[58] later generalized this appearance-based approach to accurately recognize a set of industrial objects and determine their pose. The difference between these appearance-based methods and the earlier view-based methods is that precise feature detection is unnecessary (they can be applied directly to edge maps, optical flow or normalized intensity), and that they can capture much larger range of variation in target appearance.

In this paper we are not interested in recognition from static imagery, but rather in real-time analysis of a spatio-temporal pattern, in particular people's hand gestures and facial expressions. To accomplish this we have extended the notion of appearance modeling into the temporal domain by use of elastic spatio-temporal matching. We have also coupled a task-dependent interpolation stage with our appearance analysis framework, providing direct connection between the user's gestures and task control.

Our work differs from other appearance-modeling research in the use of combinations of spatial views to describe image appearance. We use the view with the maximum similarity (minimum distance) to localize the position of the object, and the entire set of view scores at that point to characterize the actual pose of the object. We will use the term "view model" to mean the iconic representation of a single example of a target, and the term "appearance model" to mean the vector of similarities between a target and a set of view models.

3.1.1 Correlation-Based Similarity

For the presentation in this chapter, we have chosen normalized correlation to be the similarity measure between an image and a set of spatial view models to obtain real-time performance. (Similarity can also be defined directly as a likelihood measure defined by an eigenvector representation of a class of images; this will be discussed further in Section

4.3.) Given a set of models indexed by the variable m, $1 \leq m \leq M$, the view-model similarity function is

$$R_m(i,j) = \frac{S_m(i,j) - S'_m(i,j)}{n_m^2 \sigma_I \sigma_{T_m}} \ , \tag{3.1}$$

where

$$S_m(i,j) = n_m \sum_{\{u,v|d_m(u,v)=1\}} T_m(u,v) I(i+u, j+v) \ ,$$

$$S'_m(i,j) = \sum_{\{u,v|d_m(u,v)=1\}} T_m(u,v) \sum_{\{u,v|d_m(u,v)=1\}} I(i+u, j+v) \ ,$$

and where $T_m()$ is the value of pixels in the image used to define the view model, $I(i,j), 0 < i < W, 0 < j < H$ is the new image being searched, $d_m(i,j)$ is set 1 for valid view model locations and 0 for "don't-care" locations, $n_m = \sum_{i,j} d_m(i,j)$ is the number of valid pixel locations in the view model, and σ_I, σ_{T_m} are the standard deviation of the observed and model images, respectively. We define the spatial maxima and the corresponding image offsets for each view model:

$$r_* = \max_{i,j,m} R_m(i,j), \qquad (\hat{i}, \hat{j}, \hat{m}) = \arg\max_{i,j,m} R_m(i,j).$$

and store them in a vector of spatial similarity scores, called an "appearance model":

$$\mathbf{r} = [\hat{i}/W, \hat{j}/H, R_1(\hat{i},\hat{j}), R_2(\hat{i},\hat{j})..., R_M(\hat{i},\hat{j})]^T.$$

With a smooth similarity function, the similarity score of a particular view model as the object undergoes non-linear transformations such as rotation, scale, or articulation will be a roughly convex function. The peak of the function will be centered at the parameter values corresponding to the pose of the object used to create the view model. For example, Figure 3-1(a) shows three images of an eyeball that were used to create view models for gaze tracking; one looking 30 degrees left, one looking center-on, and one looking 30 degrees to

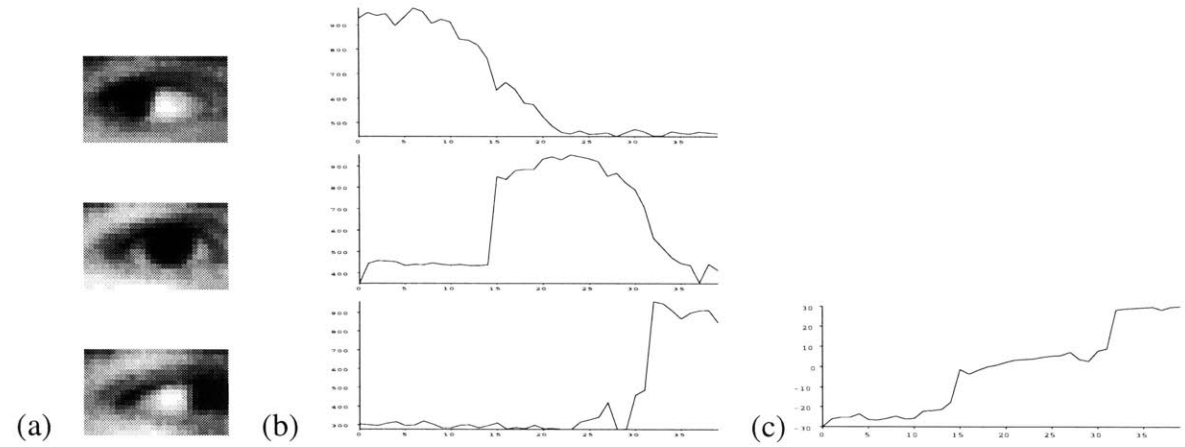


Figure 3-1: (a) Three spatial views of an eyeball at $+30$, 0, and -30 of gaze angle. (b) Normalized correlation scores of the view models when tracking a eyeball tracking from approximately -30 to $+30$ degrees of gaze angle with two reported saccades. (c) Interpolated gaze angle showing these saccades, using RBF method described in Section 3.4.

the right. Figure 3-1(b) shows the normalized correlation score for each view model when tracking a eyeball rotating from left to right, with two saccades. Each view model shows a roughly convex curve centered about the gaze angle used to create the view model.

Given a set of view models which sample a transformation parameter finely enough over a range of interest, the spatial appearance model (set of similarity scores) is a sufficient representation of the signal such that one can estimate the actual transform parameters for new views by interpolation. Section 3.4 will present a method to perform this task. In general, rather than actually estimate 3-D parameters explicitly, our method uses the normalized correlation scores directly for recognition and control. Examples of this will be shown in Section 3.5.

3.1.2 Temporal view analysis

A similar view-based dimensionality reduction is also possible in the temporal domain. We construct temporal view patterns comprised of spatial appearance models observed over

time, and use a correlation-based similarity function defined over these dimensions (t, m) to compute a spatio-temporal appearance model. Because of temporal scaling due to different sampling or performance rates, there is variation in time which we need to accommodate. Our solution is to allow the observed sequence to be arbitrarily time-warped to each stored temporal view model before computing a similarity score. The result of this stage is a spatio-temporal appearance model (vector of temporal similarity scores) containing scores which characterize both spatial *and* temporal properties of the input signal.

To find the similarity between two sequences, we again use a normalized correlation metric, but after using the Dynamic Time Warping (DTW) method to temporally align the two sequences, thus allowing the time-course of a gesture to vary. The DTW method involves the use of dynamic programming techniques [9] to solve an elastic pattern matching task, and was originally developed to solve the time alignment problem in the speech and signal processing literature [70]. Note that DTW is a simplification of Hidden Markov Modeling (HMM); they are equivalent for the relatively simple (non-branching) sequences we will be considering.

We use a version of the DTW method that has been modified to match backward in time, so that we can constrain the temporal endpoints of the two sequences to be the same, but allow the starting point to match elastically. To temporally align an observed sequence of spatial view model score vectors, $\mathcal{R} = \{\mathbf{r}[0], \mathbf{r}[1], ..., \mathbf{r}[T]\}$, with a temporal view model of spatial scores $\mathcal{T}_p = \{\mathbf{s}_p[0], \mathbf{s}_p[1], ..., \mathbf{s}_p[T'_p]\}$, where the number of vectors in each sequence are not necessarily equal, we consider a grid for each view model whose horizontal axis is associated with $\mathcal{R}$ and whose vertical axis is associated with $\mathcal{T}_p$. Each element of this grid contains a distance measure $D_{p,i,j}$ measuring the Euclidean distance between $\mathbf{r}[i]$ and $\mathbf{s}_p[j]$. The best time warp will minimize the accumulated distance backward along a monotonic path through the grid from (T, T'_p) to $(0, 0)$. The DTW algorithm uses a partial sum variable, $C_{p,i,j}$, to recursively compute a minimal solution; $C_{p,i,j}$ is defined to be the minimum cost to align $\mathbf{r}[i..T]$ with $\mathbf{s}_p[j..T'_p]$. A backward matching DTW method can be defined with

$C_{p,i,j} = D_{p,i,j} + \min(C_{p,i+1,j+1}, C_{p,i+1,j}, C_{p,i,j+1})$, except on the border of the grid, where $C_{p,T,T'_p} = D_{p,T,T'_p}$, $C_{p,i,T'_p} = D_{p,i,T'_p} + C_{p,i+1,T'_p}$, and $C_{p,T,j} = D_{p,T,j} + C_{p,T,j+1}$. When the recursion is complete, the minimum distance between $\mathcal{R}$ and $\mathcal{T}_p$ is simply $C_{p,0,0}$.

This method constrains the observed sequence and the gesture pattern to be aligned at the current time step. However, we do not know *a priori* the actual start point of the gesture in our buffer of observed spatial view scores, so we must relax the requirement that the start point of both sequences also be aligned. We define the score of a gesture model to be the minimum of any of the partial sums which account for all of the time samples in the temporal model, independent of how much of the buffer of currently observed scores is matched: $\mathcal{D}(\mathcal{R}, \mathcal{T}_p) = \min_{0 \geq t \geq T} C_{p,t,0}$.

To find the actual alignment we simply backtrack through $C_{i,j}$ along the directions of partial sum minima. We define a warp function $w_p(j)$, which returns the index of the observed sequence $\mathbf{r}$ corresponding to the j-th element of $\mathbf{s}_p$ vectors in view $\mathcal{T}_p$. This can be computed by setting

$$j^* = 0; \quad w_p(0) = i^* = \arg \min_{0 \geq t \geq T} C_{p,t,0} \, ,$$

and then iterating

$$d^* = \min(C_{p,i^*+1,j^*}, C_{p,i^*,j^*+1}, C_{p,i^*+1,j^*+1})$$

while incrementing i^* when $d^* = C_{p,i^*+1,j^*}$, j^* when $d^* = C_{p,i^*,j^*+1}$, and both when $d^* = C_{p,i^*+1,j^*+1}$. Setting $w_p(j^*) = i^*$ after each iteration, we have the optimal time-alignment when $i^* = T$ and $j^* = T'_p$.

With the optimal path through the grid in hand, we compute the normalized correlation of these time-aligned sequences, which we define to be g_p where p is the index over the set

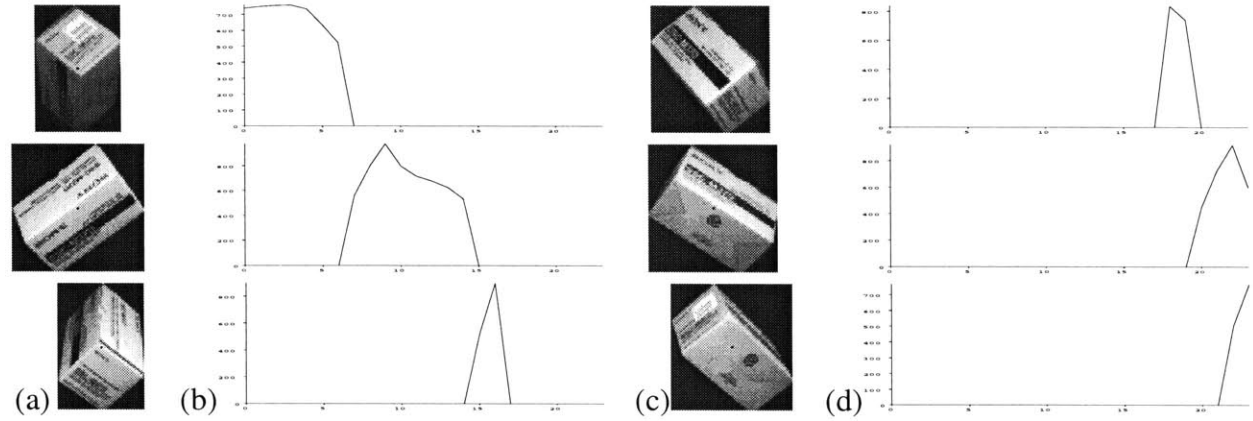


Figure 3-2: (a,c) Spatial view models automatically acquired from a sequence of images of a rotating box. (b,d) Normalized correlation scores for each model as a function of angle.

of temporal views:[1]

$$g_p = \frac{M * T \sum_{m,t} (\mathbf{s}_p)_m[t] r_m[w_p(t)] - \sum_{m,t} (\mathbf{s}_p)_m[t] \sum_{m,t} r_m[w_p(t)]}{(M * T)^2 \sigma_{\mathbf{r}} \sigma_{\mathbf{s}_p}} .$$

Finally then, **g** is the spatio-temporal appearance model, a vector of combined spatio-temporal view scores computed by cascading the temporal normalized correlation scores (with elastic matching) onto the spatial normalized correlation scores.

3.1.3 Implementation Issues and Real-time performance

The majority of the computational burden in this method involves finding the set of view model correlation scores. To compute correlation scores, we rely on a special-purpose image processing computer which is designed for quick correlation searching. The view model acquisition, evaluation, and maxima finding are implemented on a Cognex 4400

[1]This expression for g_p can be arrived at from the formula defined by Eq. (1), by first replacing m with p (the indexing variable), then u with m and v with t (the dimensions correlation is computed over), then $I(m,t)$ with $r_m[w_p(t)]$ and $T_p(m,t)$ with $(\mathbf{s}_p)_m[t]$ (the objects compared), and finally n_p with $M * T$ (the number of items in the comparison), so that $g_p = R_p(0,0)$.

vision processor developed by the Cognex Corporation (other processing is performed on Sun-4 and SGI Indigo-2 workstations). We have found that when recognizing hand gestures where the hand fills 1/8th to 1/4th of the video frame, it is possible to acquire these images at an input resolution of 128x120 and achieve good recognition performance. We have tested our system at this resolution on examples which used up to 40 models (on a run of the rotating box example); the time required to exhaustively search all models in this case was on the order of 200-300ms. (At this resolution our system can store up to 100 view models in memory accessible by the searching hardware.) Using the predictive search pruning mechanism described in [25], which exploits temporal correlation in the observed view scores, we were able to reduce the processing time for this example to under 100ms. The vision processing for the face interpolation and hand gesture recognition examples described at the end of this paper ran at rates in excess of 10 Hz.

3.2 Learning View and Appearance Models

A key problem for this approach is how to learn a set of view models that span the target object's range of appearance over both space and time. When objects are non-rigid, either constructed out of flexible materials or an articulated collection of rigid parts, such as hands and faces, the dimensionality of the space of possible appearances is very large. Full enumeration of the space in these cases is intractable even if a complete 3-D model is available. However, many appearances may never be encountered in a particular task due to additional constraints. These may be physical (some joints may not be completely independent), or behavioral (some poses or motions may never be used in the actual communication between user and machine). We therefore use a learning method which derives from training data an appropriate set of view models that are sufficient to characterize the entire range of target appearances.

We use a simple incremental unsupervised clustering scheme. The system begins with

one spatial view model taken from an image region defined by the user in the initial frame of the image sequence. The target object is then tracked by using the similarity search function given above. When the maximum search score (r_*) falls below a certain threshold θ, a new view model is added to the appearance model (i.e., the search set) using the image at the offset associated with the current best score. To add a new model, we construct a new template $T(u, v)$ with the values of the pixels in the image in the region covered by the view in the appearance model with the maximum similarity score. In our implementation and in all the results shown in this chapter, we set $\theta = 0.7$ and have empirically found this to yield good results.[2] This parameter determines the tradeoff between the number of views used (and thus storage space and computation time) and the accuracy of the appearance model representation (a higher threshold leads to a representation with more views, which can represent finer details).

For acquiring new temporal views, we use the same learning rule, but using the elastic-matching correlation function defined in the previous section. We do assume that during the learning process there is a temporal segmentation signal, so that sequences are presented to the temporal view formation process discretely. Thus, the first sequence is used to define the first temporal view model; additional sequences are added to the set of view models (the spatio-temporal appearance model) when they fail to match against the current set with a sufficient score. During run time there is no temporal segmentation signal needed.

For simple objects and transformations, this clustering method can build an appearance model which adequately covers the entire space of possible target appearances. For example, for a convex rigid body undergoing a 1-D rotation with fixed relative illumination, a small number of view models can match object across a range of the rotation transformation. Figures 3-2 illustrates this with an example of a box rotating about a single axis.

[2]This assumes that the target object is indeed present in the entire image sequence. In practice we also use a second threshold, $\theta_0 = 0.6$, and disregard any images that match with $r_* \leq \theta_0$. During the training phase, if the target object leaves the scene or is occluded and the best score falls below this threshold, we will not build spurious view models.

3.3 Limitations and extensions of the appearance-based method

Our iconic, appearance-based approach to representing spatio-temporal events has the advantages of being efficient to implement and having a data-driven learning method which makes few assumptions on the exact character of the input signal. However, this approach also has several limitations, particularly in the ability of an iconic method to generalize across all possible gestures of a certain semantic class.

An appearance-based method using purely iconic view model templates cannot be expected to perform generalization across multiple users or when a single user performs a gesture that is not well modeled by previously-seen spatio-temporal patterns. We therefore use our method in environment where the user is known via face recognition or other methods, and where the the user is willing to use standard patterns when interacting with the system (note that the user can teach the system the patterns they want to use; they are not fixed ahead of time).

Furthermore, our use of direct modeling of view intensity makes our system sensitive to direction of illumination (which will effect shading) and viewing direction, although it provides for invariance to gross illumination changes. In practice, we have simply controlled the illumination, for instance, using near-band IR illumination from LEDs (which is visible to standard CCD cameras but invisible to humans), or using the location-dependent training scheme in Section 4.4.

However, even with these restrictions there are many domains in which our method can be useful. Examples include a user interacting with a workstation, a driver in an automobile, and camera control in teleconferencing. In each of these cases the set of users is small and/or user identity is available through other means, there are clear advantages to control via characteristic hand or face gestures, and the imaging conditions are relatively constant.

3.4 Interpolation from Visual Input to Task Control

We use a task-dependent interpolation method to map from the set of view model scores to a result vector used for recognition or control. Interpolation is done using a supervised learning paradigm, based on a set of training examples which define the desired result for a particular set of view model outputs. Using the Radial Basis Function (RBF) method presented in [62], we compute a result vector $\hat{\mathbf{y}}$ to be a weighted sum of radial functions centered at an exemplar value:

$$\hat{\mathbf{y}}(\mathbf{g}) = \sum_{i=1}^{n} c_i \mathcal{F}(||\mathbf{g} - \mathbf{g}^{(\mathrm{i})}||) \ , \tag{3.2}$$

where

$$\mathbf{c} = \mathbf{F}^{-1}\mathbf{y}, \quad (\mathbf{F})_{ij} = \mathcal{F}(||\mathbf{g}^{(\mathrm{i})} - \mathbf{g}^{(\mathrm{j})}||), \quad \mathbf{y} = [\mathbf{y}^{(1)}, ..., \mathbf{y}^{(\mathrm{n})}]^{\mathbf{T}} \ , \tag{3.3}$$

$\mathbf{g}$ are the computed spatio-temporal view-model scores, and $\{(\mathbf{y}^{(\mathrm{i})}, \mathbf{g}^{(\mathrm{i})})\}$ are a set of exemplar result and view-model score pairs (which may be scalar or vector valued). $\mathcal{F}$ is the RBF, which in our implementation was simply $\mathcal{F}(\S) = \S$.

We use the interpolation stage to map the observed view model scores into a quantity which is directly useful for a particular task. For example, if we wanted to estimate the eye gaze angle for the example in Figure 3-1, we could use an RBF interpolator with a one dimensional output space corresponding to gaze angle and three exemplars, containing the view model scores corresponding to each view model angle:

$$\{(y^{(i)}, \mathbf{g}^{(i)})\} = \{(-30, [1.0, 0.3, 0.3]^T), (0, [0.3, 1.0, 0.3]^T), (30, [0.3, 0.3, 1.0]^T)\}$$

Using this RBF configuration, it is straightforward to recover an estimate of the underlying eye gaze angle from the three spatial view model outputs. The interpolated gaze angle is shown in Figure 3-1(c).

3.5 Examples of Appearance-based Analysis

3.5.1 Mimicking Facial Expressions

The modeling and tracking of expressions and faces has been a topic of increasing interest recently. Facial animation is a difficult problem due to the sheer complexity of realistic facial models: dozens of degrees of freedom are present in a face, and to control them for computer animation using conventional keyframe or motor control techniques is quite difficult. A much more natural approach is to use one's own face to control the model face parameters. We present a system which tracks facial expressions in real-time without mechanical actuators or make-up, using our interpolated appearance-based vision methods.

This approach follows in the tradition of others who have explored techniques for visual analysis of facial expression [87, 74, 10, 34, 51]. In our method spatial view outputs are interpolated to control the motor states of a 3-D computer graphics face, using the face model employed in Essa and Pentland [34]. The result vector $\hat{\mathbf{y}}$ is defined to be the motor state of the animated face. Training examples are acquired by setting the model face to generate a particular expression, asking the user to mimic the expression, and recording the pair of vision scores and muscle parameters.

We have implemented real-time facial expression tracking, for use in interactive animation or telepresence. Figure 3-3(a) shows 5 frames of a 125 frame video sequence of a user making a smile and surprise expression are shown. Spatial view models covering the entire face were acquired from a separate training sequence containing the same user making these expressions. The unsupervised clustering method returned view models corresponding to a neutral, smile, and surprise expression. (No temporal views were used in this example, since we were interested in tracking static pose.) An RBF interpolator was trained using perceptual/motor state pairs for these three expressions; the resulting (interpolated) motor control values for the entire sequence are shown in Figure 3-3(c), and the rendered facial

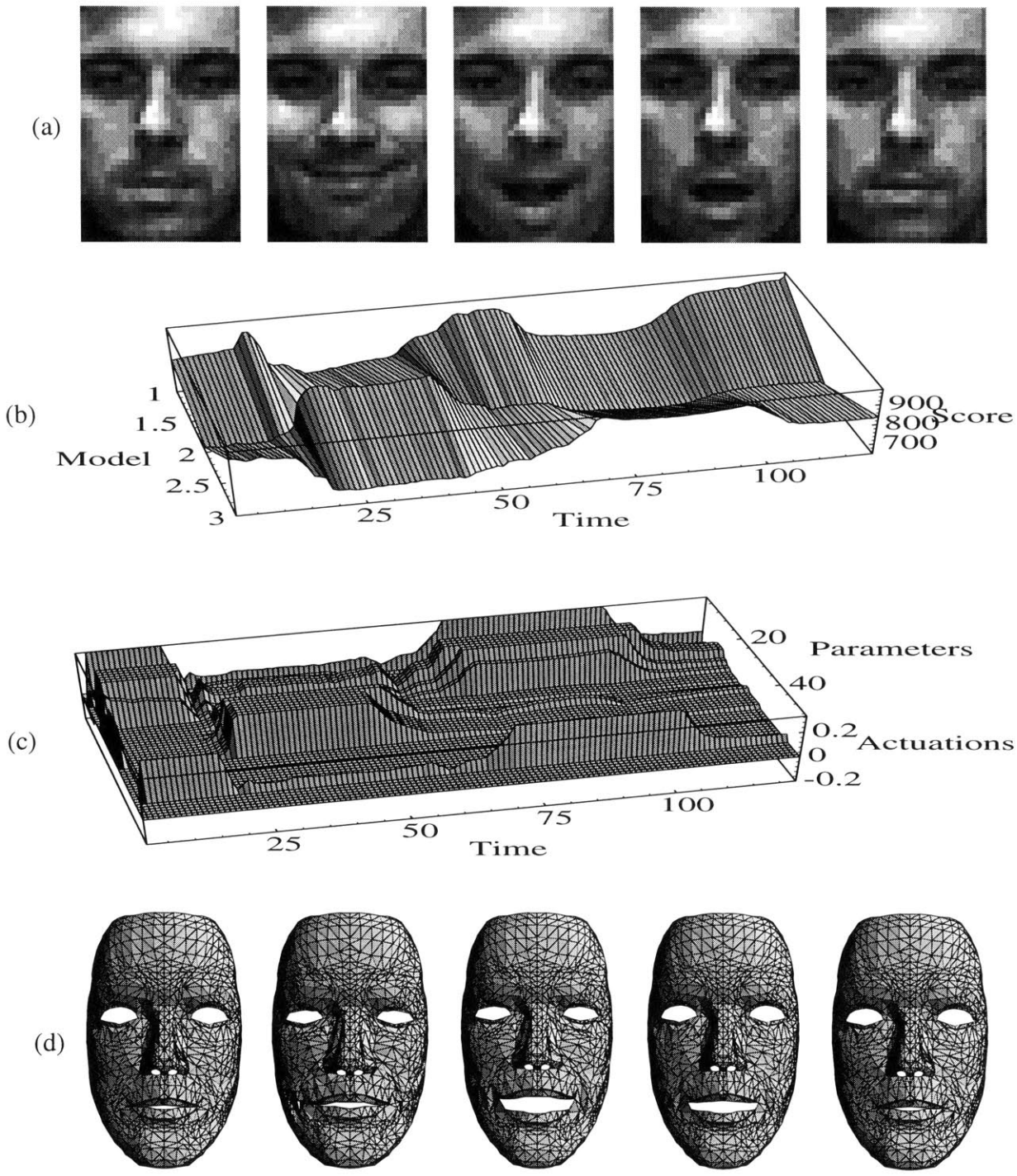


Figure 3-3: Interactive facial expression tracking in real time. A set of normalized correlation view models are used to characterize facial state, and then used to interpolate a set of motor control parameters for a physically based face model. View models are acquired using unsupervised clustering while the interpolation is trained using supervised learning. See text for details.

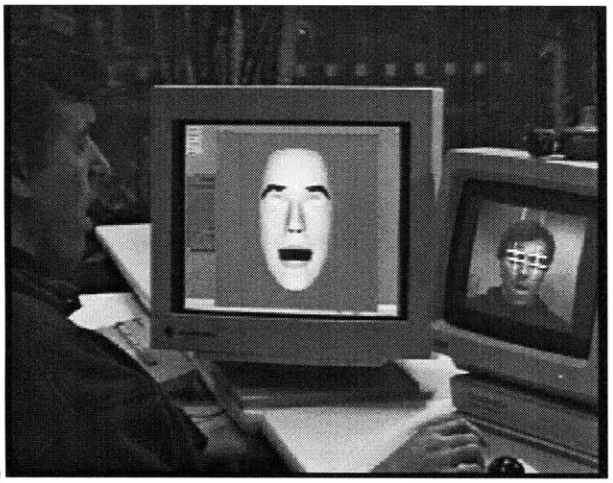
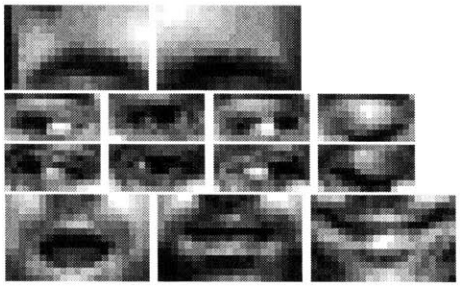
(a)  (b) 

Figure 3-4: (a) Overview shot of full system for tracking facial expression in real-time. User is on left, vision system and camera is on right, and animated face is in the center of the scene. The animated face matches the state of the users face in real-time, including eye-blinks (as is the case in this shot.) (b) Spatial view models learned by unsupervised clustering method for region-based face mimicking example; independent view models were found for eyes, eye-brows, and mouth regions.

mesh for five frames of these motor control values is shown in Figure 3-3(d). Note that the unsupervised clustering method will often return more view models than the actual number of expressions; the RBF method makes no assumptions in this regard.

When there are only a few canonical expressions that need be tracked/matched, this full-face view-based approach is robust and simple. However if the user wishes to exercise independent control of the various regions of the face, then use of full-face models will be overly restrictive. For example, if the user trains two expressions, eyes closed and eyes open, and then runs the system and attempts to blink only one eye, the rendered face will be unable to match it, instead half-closing both eyes. A solution is to decouple the regions of the face which are independent geometrically (and to some degree, independent in terms of muscle effect.) In this approach, separate appearance models are computed for each facial region, and multiple RBF interpolations are performed. Each interpolator drives a distinct subset of the motor state vector.

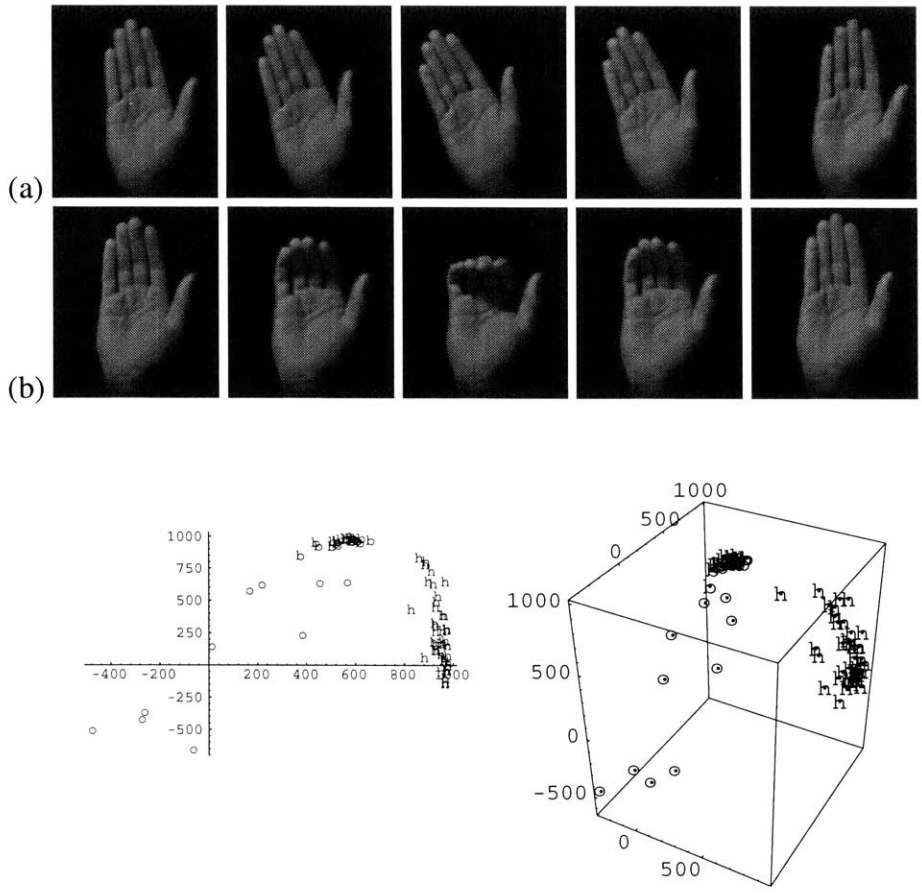
Figure 3-4 shows a picture of the set-up of the system as it is being run in an interactive setting, the regions used for the decoupled view models, and the actual view models acquired for the smile/surprise sequence. During run-time, the animated face mimics the facial state of the user, matching in real time the position of the eyes, eyelids, eyebrows and mouth of the user. In the example shown in this picture, the users eyes are closed, so the animated face's eyes are similarly closed.

Realistic real-time performance of animated facial expressions has been achieved with this method: our prototype system combining vision and graphics processing runs in excess of 8 frames/sec with approximately 0.5 sec lag. Typical users can use the system for periods of approximately 15 minutes without having to retrain the view models. In addition, our system offers an extremely low bitrate mechanism for facial teleconferencing. In the above example, vision scores can be encoded in approximately 64 bits per frame; at 8 Hz, only 512 bytes/sec of bandwidth is required. In the domains where the user is known and the imaging conditions relatively controlled, our method can provide a real-time solution to low-bitrate facial coding for teleconferencing and telepresence applications.

3.5.2 Recognition of Hand Gestures

Another application of our appearance-based interpolation framework is the recognition of hand gestures. An RBF-based classifier can be defined by interpolating the spatio-temporal similarity scores into a space whose axes correspond to the recent performance of the patterns we wish to detect. In contrast to the previous example, this classification task calls for a discrete decision rather than a continuous modulation of an input signal. We achieve this by selecting only the maximal component of the interpolated result.

We tested our system on a recognition task with two target gestures. These patterns were selected to be a simple interface to a video conference control system, and are arbitrary. Fourty-two examples of a "hello" gesture were collected, twenty-six examples of "good-



(c) (d)

Figure 3-5: Formation of spatial and temporal view models for hand gesture analysis. Example of (a) "hello" and (b) "good-bye" gesture are shown. Spatial and temporal view models were found for these gestures, as described in the text. (c,d) Scatter plot of spatio-temporal correlation feature vector on gestures in test set (h=hello, b=bye, o=other). (c) shows case where 2 temporal models were found, (d) shows case with three temporal models.

(a)

n_{train}=3	(actual) hello	(actual) bye	(actual) other
(predicted) hello	3928	104	68
(predicted) bye	0	2492	8
(predicted) other	68	239	593

(b)

n_{train}=39	(actual) hello	(actual) bye	(actual) other
(predicted) hello	2069	18	13
(predicted) bye	0	1297	3
(predicted) other	2	36	462

(c)

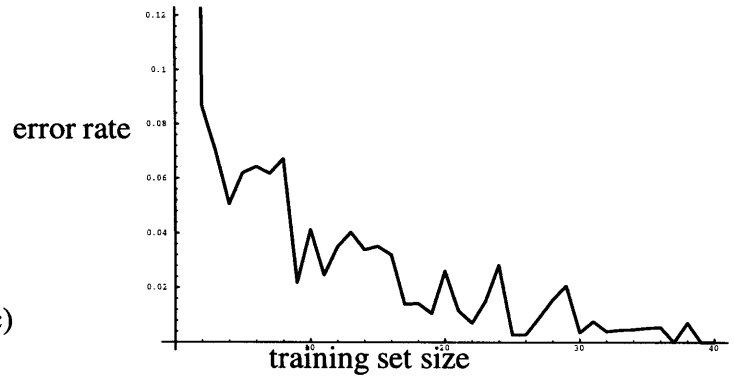


Figure 3-6: Classification of spatio-temporal view model scores using RBF-based method described in text. (a) Confusion matrix obtained using RBF classification method with a training set containing one example of each gesture; rows indicate predicted gesture, columns indicate actual gesture ($n_{train} = 3$, $n_{test} = 75$, results summed over 100 runs, for a total of 7500 test trials). (b) Confusion matrix obtained using training set containing half of all gestures (n_{train}=39, n_{test}=39, results summed over 100 runs, for a total of 3900 test trials). (c) Plot of error rate as a function of training set size. Error rate ranged from 6.5% for $n_{train} = 3$ to 1.8% for $n_{train} = 39$.

bye" and ten examples of other gestures intended to generate false alarms in the classifier. Each user is free to define "hello" or "goodbye" to be any repeatable spatio-temporal pattern. We assume a training phase where the gesture is performed in front of a known background (and we set $d_m(i,j)$ accordingly). Users performed the example gestures discretely, so that temporal segmentation was provided in the training phase.

Figure 3-5(a,b) shows a representative example of a hello and a goodbye gesture produced by one user. For each trial we randomly selected a subset of gestures to train the classifier, and tested on the remaining gestures. "Hello" and "good-bye" gestures in the training set were input to the unsupervised clustering procedure, which computed spatial and temporal views as described above. In each run five spatial views and two or three temporal views were found by the clustering procedure. The resulting spatio-temporal appearance model scores of these view models proved to be a good classification mechanism. Figure 3-5(c,d) shows example scatter plots of temporal appearance vectors ($\mathbf{g}$), labeled with the actual gesture (Hello, Bye, Other). Figure 3-5(c) shows a case where two temporal view models were found, this leading to a 2-D spatio-temporal appearance vector. Figure 3-5(d) shows a case where 3 temporal view models were found for the data resulting in 3-D spatio-temporal appearance vector. In both cases a clear separation of the target patterns is present in the spatio-temporal appearance vectors.

To perform classification, we configured an RBF interpolator with a three dimensional output space and an input space corresponding to the dimensionality of the spatio-temporal appearance model. The RBF was defined with exemplars of the form $(\mathbf{y}^{(i)}, \mathbf{g}^{(i)})$, where $\mathbf{g}^{(i)}$ are the spatio-temporal appearance vectors in the training set, $\mathbf{y}^{(i)}$ is set to $[1,0,0]^T$ if the i-th gesture in the training set is a "hello" gesture, $[0,1,0]^T$ if "good-bye", and $[0,0,1]^T$ if it is a conflictor or no gesture ("other"). We classified all of the gestures in the test set, defining the predicted class based on the index of the largest interpolated score in the result vector. We classified the gesture "hello" if the first element was largest, "good-bye" if the second was largest, and "other" if the third was largest.

Figures 3-6 shows the classification results for different training set sizes, The confusion matrices for a training set of 3 and 39 gestures from the full set of 78, are shown, based on data summed over 100 trial runs. (The graph is estimated from data summed over 10 trial runs per training set size. With only three training examples (one of each type, randomly selected) we obtained a remarkable success rate (ratio of the number of correct trials to the total number of trials) of 93.5%. When the training set size was allowed to increase to be half the size of the data set, performance increased to 98.2%.

3.6 Summary of appearance-based gesture analysis and interpolation

We have developed and implemented a real-time system for learning, tracking, and recognizing complex objects and gestures defined as characteristic spatio-temporal patterns. The use of an appearance-based representation allows us to model (and search) only the portion of an object's appearance-space which is actually used by a user in the gestures to be analyzed. Our appearance-based approach also allow analysis without having to recover exactly the underlying object pose parameters. Using task-dependent interpolation based on the Radial Basis function method, we have been able to achieve fast and robust analysis and synthesis of facial expressions, and accurately recognize hand gestures using only conventional video camera images as input. This system has promise as a new approach in the interactive animation, video tele-conferencing, and personalized interface domains.

Chapter 4

Active Tracking of Expression and Pose

Ideally, a system for interactive interface should know whether a user is paying attention, in particular where the user is looking during a dialog. We would like to have a wireless, unconstrainted interface, which can detect where the user is looking and what his or her facial state/expression is. To accomplish this, tracking and expression analysis must occur over a wide spatial range. The person tracking methods presented in Chapter 2 can follow a user around a room, but cannot discriminate detailed structure. The appearance-based analysis presented in Chapter 3 can capture detailed expression structure, but require high-resolution images of the hand or face.

To extend the appearance-based gesture analysis method to work in unconstrained environments, we propose the use an active camera to obtain high-resolution images of the users hand or face. In this chapter we show how appearance-based methods can estimate expression or pose from an active, high-resolution camera, where the active camera is driven by tracking an unconstrained user in a fixed low-resolution, camera view.

4.1 Active Camera Architecture

To provide high resolution images for gesture recognition, we augment the existing wide field-of-view camera in our interactive environment with an active, narrow-field-of-view camera. Information about feature (e.g. head/hand) location from the person tracking methods run on a wide field-of-view camera is used to drive the motor control parameters of the narrow field camera. We use the figure/ground segmentation and contour analysis routines described in Chapter 2 to determine head or hand location. This location is then translated into gaze angles for the active camera's motor system, and a foveated image of that body part is acquired. We then apply the appearance based analysis presented in Chapter 3 to these high-resolution images. Camera control is performed open-loop using the general person tracking routines, and closed-loop, using feedback from the expression and pose analysis (Figure 4-1.) Figure 4-2(a,b) shows example output from the wide angle camera and the narrow angle camera, as the narrow camera tracked the users head given the head position information computed by the ALIVE routines on the wide angle image.

4.2 Foveated Expression Tracking Example

We have integrated the active vision sensor with the appearance-based gesture analysis to track facial expression. Using a simplified, two expression model (neutral and surprised), we tracked facial expressions as the user moved about the scene and the narrow angle camera followed the face. Figures 4-3,4-4 show the results of tracking these two expressions using the narrow angle camera input when the user is in two different locations in the scene, using a single set of view models. The view models were acquired at a location in the scene different from the two locations where we ran these tracking experiments.

For each of these experiments a surprise measure was interpolated from the view scores using the radial basis function method described in Section 3.4. To produce the interpolated

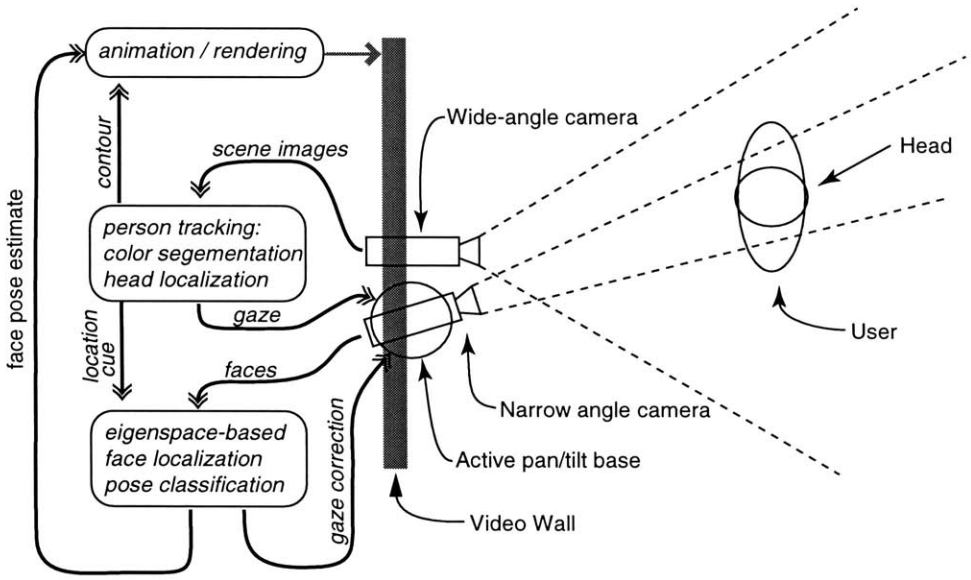


Figure 4-1: Overview of system for face/body tracking and pose estimation, Objects are rendered on Video Wall and react to facial pose or expression of user. Static, wide-field-of-view, camera tracks user's head, and drives gaze control of active, narrow-field-of-view camera. Appearance/view-based analysis is run on face images from active camera, to provide pose or expression estimates for objects/agents to react to, and to provide closed loop face tracking feedback for active camera.

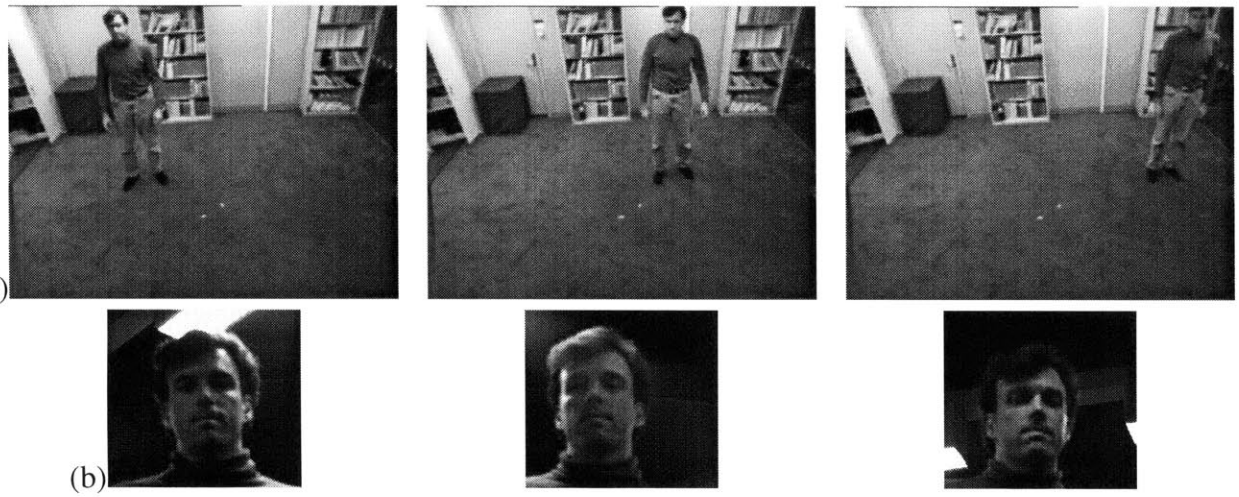


Figure 4-2: Active tracking in an interactive room; images acquired from (a) wide and (b) narrow field of view cameras as user moved across scene and narrow camera tracks head.

surprise measures shown in these figures, we mapped the vision scores to a one-dimensional motor value, labeled "surprise". With this formulation, the distinction between expression recognition and expression tracking becomes blurred; the surprise measure can be used directly for animation, or peaks can be found and used for recognition/detection.

Figure 4-5 shows the plot of the two view model scores and the interpolated surprise measure for the entire run. During this run, the user began standing in the middle of the scene, made three surprise expressions, then moved to the left, back to the middle, and finally to the right of the scene, and repeated the three expressions at each location. In the graphs we can see that the view scores fall to zero as the user moves to a new location and the camera saccades to find the face again. When fixated on the face, the two fixed view-models extract useful information about the surprise expression, as is evidenced by the four sets of three peaks in the interpolated surprise measure. Each peak corresponds to the user performing the surprise expression, which he did three times at each of the four locations in the scene.

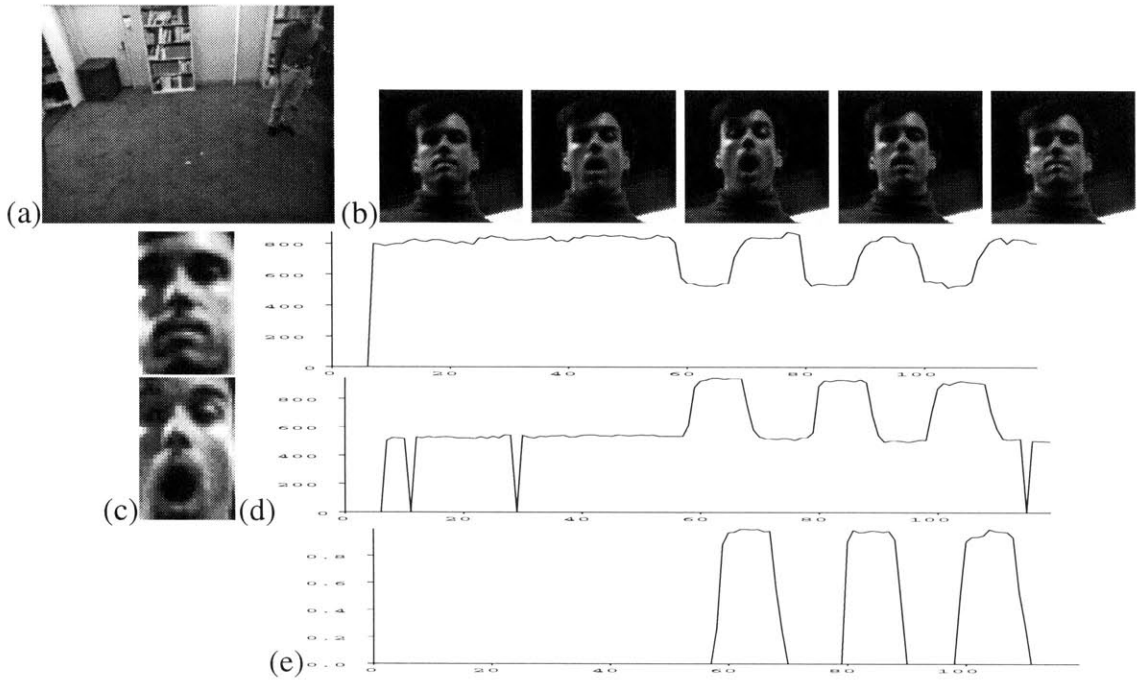


Figure 4-3: View-based expression tracking using foveated face images. (a) Wide-angle view of scene. (b) Foveated images of face while user performs one "surprise" expression. (c) Normalized correlation "view" templates of neutral and surprise expression. Views were trained while user was at a different location in the scene. (d) Normalized correlation score of view templates evaluated on sequence in (b). User performed *three* surprise expressions in the sequence. (e) Plot of surprise measure interpolated from view template scores. Three peaks are present corresponding to the three surprise expressions.

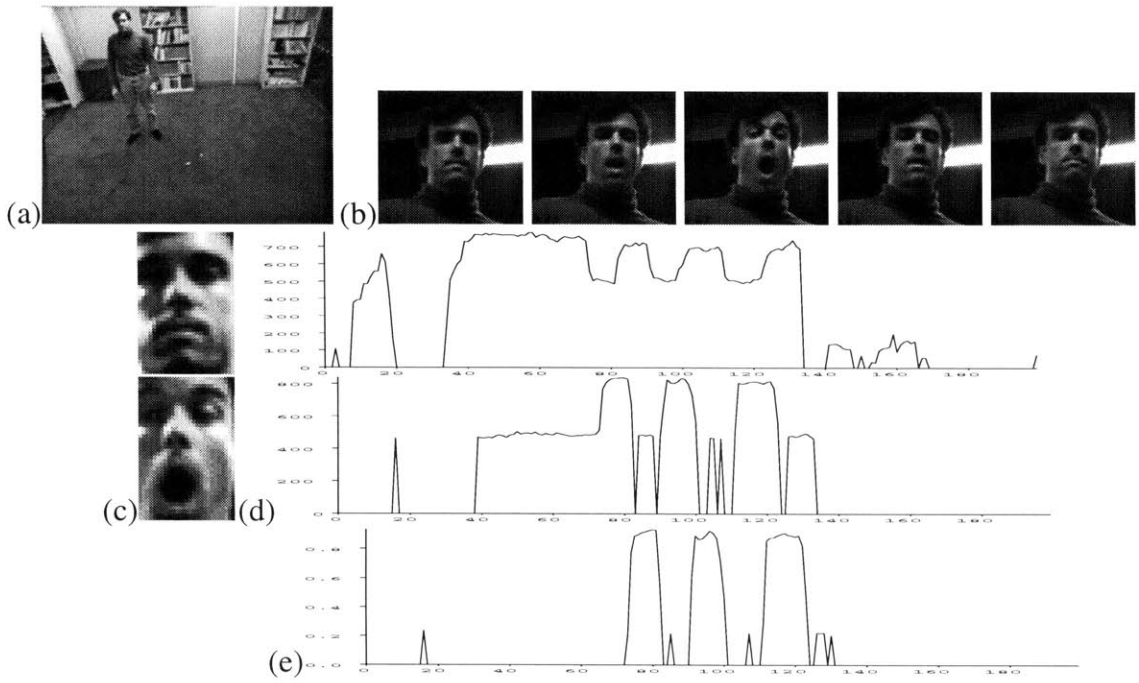


Figure 4-4: Same surprise expression performed at different location in scene, analyzed using same foveated view templates as as in previous figure.

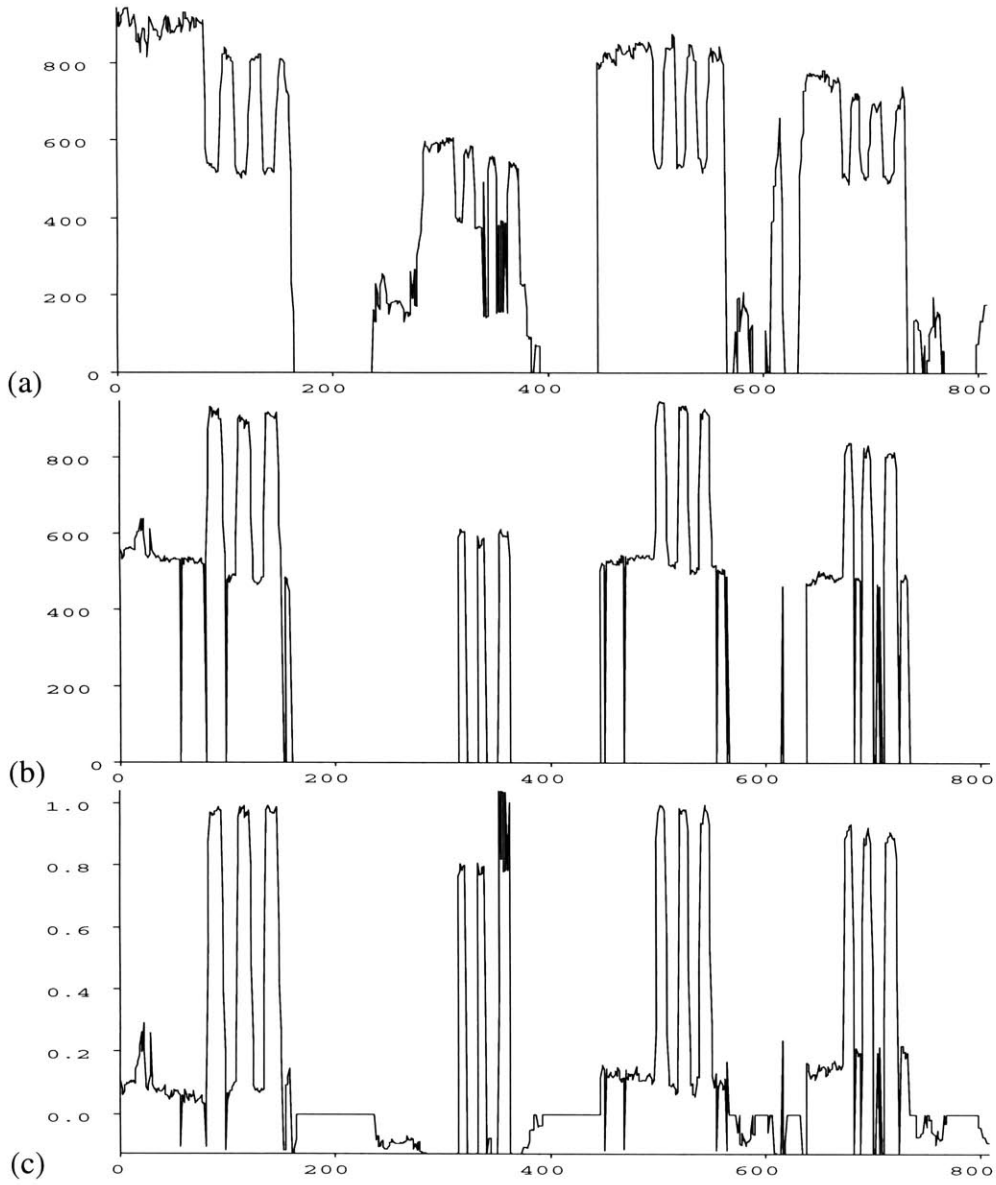


Figure 4-5: Results from extended run where user moved to four locations in scene and performed three surprise expressions in each location. (a) Neutral view model score, (b) surprise view model score, and (c) interpolated surprise measure. The active camera followed the users face (scores drop to zero during camera motion), and the surprise measure picks up the three expressions the user performed at each location.

4.3 Pose Estimation with Eigenspaces

While the appearance-based framework based on normalized correlation views has been successful at the simple tracking shown above, it is a fundamentally limited method of characterizing higher dimensional spaces. Our most recent work on tracking facial pose relies on a more powerful technique for describing the distribution of a class of face images, the eigenspace technique.

We extend the appearance-based approach to utilize an eigenspace representation for each view, as proposed by [61]. In this formulation a separate set of "eigenfaces" is computed for each possible object pose. Object pose is identified by computing the eigenspace projection of the input image onto each eigenspace and selecting the one with the lowest residual error (or "distance-from-feature-space" (DFFS) metric [61]). This scheme can be viewed as a *multiple-observer* system where separate eigenspaces are simultaneously "competing" to describe the input image.

The key difference between the view-based and a traditional parametric eigenspace representation (e.g. [58]) can be understood by considering the geometry of facespace. In the high-dimensional vector space of an input image, multiple-orientation training images are represented by a set of M distinct regions, each defined by the scatter of N example images. Multiple views of a face form non-convex (yet connected) regions in image space [11]. Therefore the resulting ensemble is a highly complex and non-separable manifold. The parametric eigenspace attempts to describe this ensemble by a projection onto a single low-dimensional linear subspace (corresponding to the first n eigenvectors of the NM training images). In contrast, the view-based approach corresponds to M independent subspaces, each describing a particular region of the facespace (corresponding to a particular view of a face). The relevant analogy here is that of modeling a complex distribution by a single cluster model or by the union of several component clusters. The latter (view-based) representation can yield a more accurate representation of the underlying geometry

depending on the degree of manifold complexity of the data, as was argued in Chapter 2 (and [25]).

4.3.1 MAP estimation with Eigenspaces[1]

Recently Moghaddam & Pentland [56] have shown that the DFFS measure can be combined with a corresponding "distance-in-feature-space" (DIFS) to yield an estimate of the probability density function for a class of images. This *likelihood* estimate can be made optimal (with respect to information-theoretic *divergence*) and can be computed solely from the low-dimensional subspace projection coefficients, thus yielding a computationally efficient estimator for high-dimensional probability density functions.

Specifically, given a set of training images $\{\mathbf{x}^t\}_{t=1}^{N_T}$, from an object class Ω (in this case a collection of user views from a single location and pose), we wish to estimate the *likelihood* function for this data — *i.e.,* $P(\mathbf{x}|\Omega)$. We note that from a probabilistic perspective, the class-conditional density $P(\mathbf{x}|\Omega)$ is the most important data representation to be learned. This density is the critical component in detection, recognition, prediction, interpolation and general inference. In our case, having learned these densities for several pose classes $\{\Omega_1, \Omega_2, \cdots, \Omega_n\}$, we can formulate either a maximum-likelihood estimate

$$\Omega^{\text{ML}}(\mathbf{x}) = \text{argmax}\{P(\mathbf{x}|\Omega_i)\} \tag{4.1}$$

or a maximum *a posteriori* estimate

$$\Omega^{\text{MAP}}(\mathbf{x}) = \Omega_j \text{ s.t. } P(\Omega_j|\mathbf{x}) > P(\Omega_i|\mathbf{x}) \; \forall i \neq j \tag{4.2}$$

[1] This subsection was written by Baback Moghaddam, as part of a joint publication with the author [29].

using Bayes rule

$$P(\Omega_i|\mathbf{x}) = \frac{P(\mathbf{x}|\Omega_i)P(\Omega_i)}{\sum_{j=1}^{n} P(\mathbf{x}|\Omega_j)P(\Omega_j)} \tag{4.3}$$

We now review how an arbitrary density estimate $P(\mathbf{x}|\Omega_i)$ can be computed using the eigenspace technique of [56] specialized to the case of a Gaussian distribution. Given a set of m-by-n images $\{I^t\}_{t=1}^{N_T}$, we can form a training set of vectors $\{\mathbf{x}^t\}$, where $\mathbf{x} \in \mathcal{R}^{N=mn}$, by lexicographic ordering of the pixel elements of each image I^t. The basis functions in a Karhunen-Loeve Transform (KLT) [47] are obtained by solving the eigenvalue problem

$$\Lambda = \Phi^T \Sigma \Phi \tag{4.4}$$

where Σ is the covariance matrix of the data, Φ is the eigenvector matrix of Σ and Λ is the corresponding diagonal matrix of eigenvalues. In PCA, a partial KLT is performed to identify the largest-eigenvalue eigenvectors and obtain a principal component feature vector $\mathbf{y} = \Phi_{\mathrm{M}}^T \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ is the mean-normalized image vector and Φ_M is a submatrix of Φ containing the principal eigenvectors. PCA can be seen as a linear transformation $\mathbf{y} = \mathcal{T}(\mathbf{x}) : \mathcal{R}^N \rightarrow \mathcal{R}^M$ which extracts a lower-dimensional subspace of the KL basis corresponding to the maximal eigenvalues. This corresponds to an orthogonal decomposition of the vector space $\mathcal{R}^N$ into two mutually exclusive and complementary subspaces: the principal subspace (or feature space) $F = \{\Phi_i\}_{i=1}^{M}$ containing the principal components and its orthogonal complement $\bar{F} = \{\Phi_i\}_{i=M+1}^{N}$. As shown in [56], an estimator for $\hat{P}(\mathbf{x}|\Omega)$ is given by:

$$\begin{aligned} \hat{P}(\mathbf{x}|\Omega) &= \left[\frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{M}\frac{y_i^2}{\lambda_i}\right)}{(2\pi)^{M/2}\prod_{i=1}^{M}\lambda_i^{1/2}}\right] \cdot \left[\frac{\exp\left(-\frac{\epsilon^2(\mathbf{x})}{2\rho}\right)}{(2\pi\rho)^{(N-M)/2}}\right] \\ &= P_F(\mathbf{x}|\Omega)\, \hat{P}_{\bar{F}}(\mathbf{x}|\Omega) \end{aligned} \tag{4.5}$$

In general, brute-force computation of pose likelihoods in real-time is computationally infeasible. Fortunately, most of the information computed by a brute force evaluation of DFFS is of little importance–what is of interest is the location of the minima of the distance function. Following [19], we use the zero-th order eigenvectors, E0, to perform spatial localization within the foveated camera view. We compute a coarse to fine search using the E0 template for each pose, and find the pose and offset which has maximal normalized correlation response. We then fully evaluate the higher order eigenvectors at this location for each pose, and compute the pose class likelihood as given above.

4.4 Location-dependent Eigenspace Learning

Face images obtained from our active camera can be used to compute pose estimates, using the eigenspaces technique described above. However, with a user moving in 3-space, we have to deal with considerable variations in scale (size of head), and illumination changes (such as shadows) that are not well modeled by a single eigenspace. These variations have large-scale geometric effects, just as do changes in pose. Our approach is to define multiple *sets* of eigenspaces, indexed over both pose and location in the world. A set of eigenspaces is constructed corresponding to each facial pose and world location. Each pose class is defined by a set of location specific pose class statistics:

$$\Omega_i = \{\Omega_{i,l}\}, l \in \mathcal{L}, \tag{4.6}$$

where the set of world locations is given by

$$\mathcal{L} = \{\mathbf{z}_0, \mathbf{z}_1, ...\mathbf{z}_L\}, \tag{4.7}$$

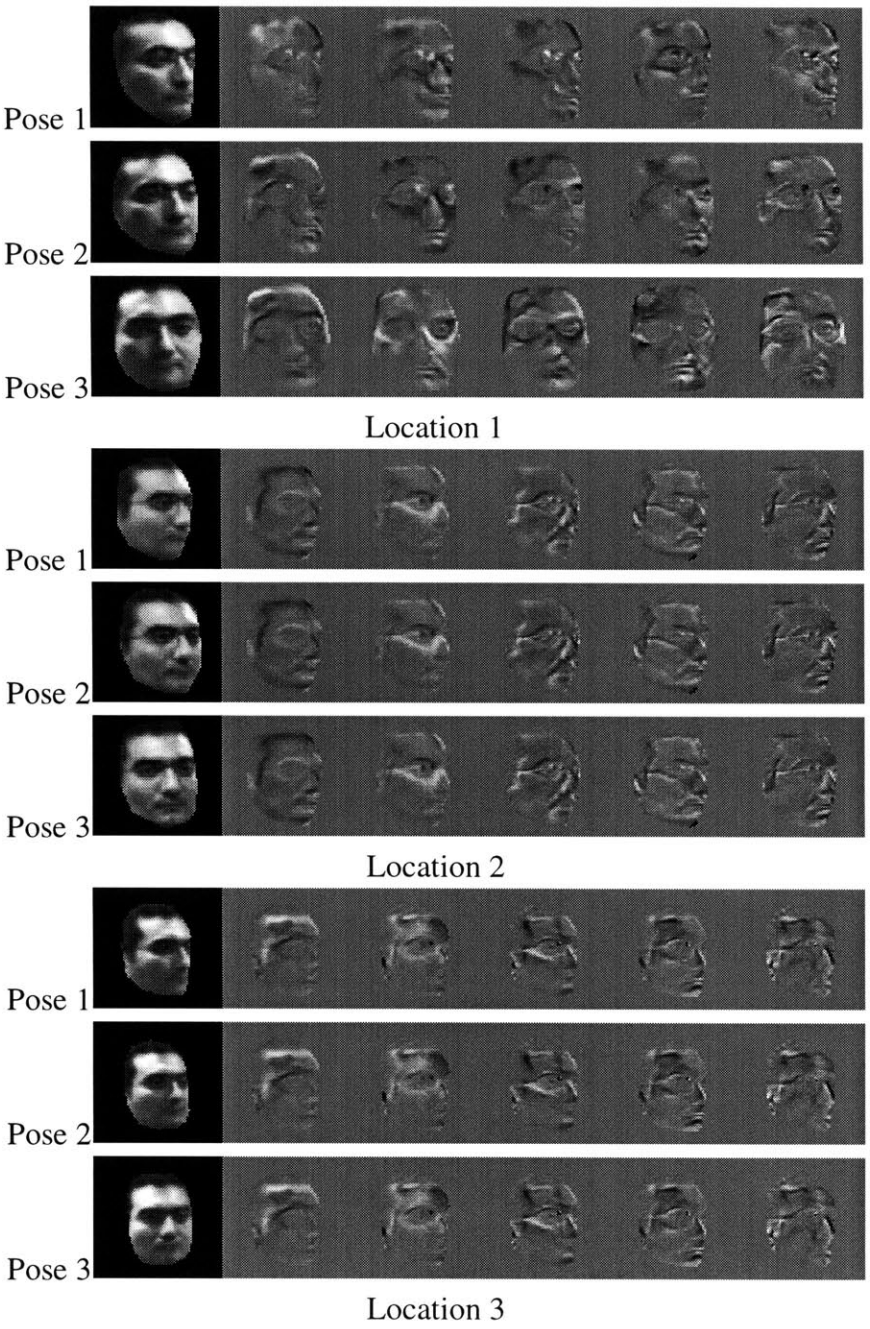


Figure 4-6: Multiple-Pose Eigenspaces for 3 different spatial locations

where $\mathbf{z}$ is a 3-D coordinate vector. To compute a composite pose class likelihood, we consider the estimation problem to be a case of estimation given spare observations. We approximate the probability at locations where no training data is available. Given an observed face image x at a world location z^*, we compute an approximate probability via interpolation among the K nearest locations which have actual probability estimates. Using a linear interpolant, we have

$$\hat{P}(\mathbf{x}, z^*|\Omega_i) \approx \sum_{k=0}^{K} w(k)\hat{P}(\mathbf{x}|\Omega_{i,n(k)}), \tag{4.8}$$

where $n(k)$ is a function that returns the k-th nearest location to z^* in $\mathcal{L}$, and $w(k)$ weights the distance of each location

$$w(k) = \frac{||z^* - n(k)||^2}{\sum_{j=0}^{K} ||z^* - n(j)||^2} \tag{4.9}$$

This offers much increased accuracy over computing a single set of pose eigenspaces for use over the entire room environment. Figure 4-6 shows sets of eigenspaces for three different poses collected at three different world locations.

Note that we need not evaluate the eigenspaces for each possible world location, since the person tracking routines provide an estimate of the users position that is sufficient to restrict the set of eigenspaces used by the system. The run-time computational burden of having L different world locations each with a separate set of pose templates is k times the cost of a single location, since we need not evaluate the eigenspace likelihoods that for locations that are not in the nearest neighbor set [82].

We evaluated the tracking and pose estimation performance of our system. Eigenspaces were trained for each of 3 poses at 10 different world locations, using a sample set size of 10 images at each location. The locations were set to be in two concentric semi-circles on the floor of the workspace, at camera pan angles of -32, -16, 0, 16, and 32 degrees, and

ranges of 80 and 120 inches. The active camera was fitted with a lens of 50mm focal length (c-mount type). Figure 4-6 shows three of the eigenspaces that were trained.

In these experiments we have used multiple views of a *single* user to construct eigenspaces. In our data set, each eigenspace describes variations in appearance due to expressions, slight mis-alignments and with and without glasses. (The method, however, can easily be extended to multiple users.)

We then evaluated the performance of the system against new images of the same user both at the locations where the eigenspaces were trained, and at randomly selected floor locations. We used the spatial localization method described above, evaluating eigenspaces at the location at which the corresponding E0 template had maximal normalized correlation response.

First, we note that eigenspace face analysis can improve head tracking accuracy using closed-loop feedback to guide the active camera. Figure 4-7 compares the camera position in the case of open loop control, when the gaze angle is determined only by the wide-angle person finder, and closed loop control, when the gaze angle is corrected by the offset of the face in the current foveated image. During normal system operation, we set a threshold on DFFS value to determine the transition between open and closed loop state, so that the closed loop signal does not contribute when there is no face in the active camera field of view. During these runs, the user was approximately twelve feet from the camera, and walked freely in approximately a ten by ten foot area. Total time for computing pose estimates and active tracking, including closed loop feedback, was less than 1/5 second.

Second, we show the pose classification rate for our system. In a trial with $n = 25$ observations, where 10 of these observations were at the training locations and the remainder at locations chosen with a uniform probability across the workspace, we computed the pose class confusion matrix. Three pose classes were used, one for looking to the left of the screen (Ω_1), one for looking at the center of the screen (Ω_2), and one for looking at the right of the screen (Ω_3). Recall that the screen was situated in front of the 15'x15' space,

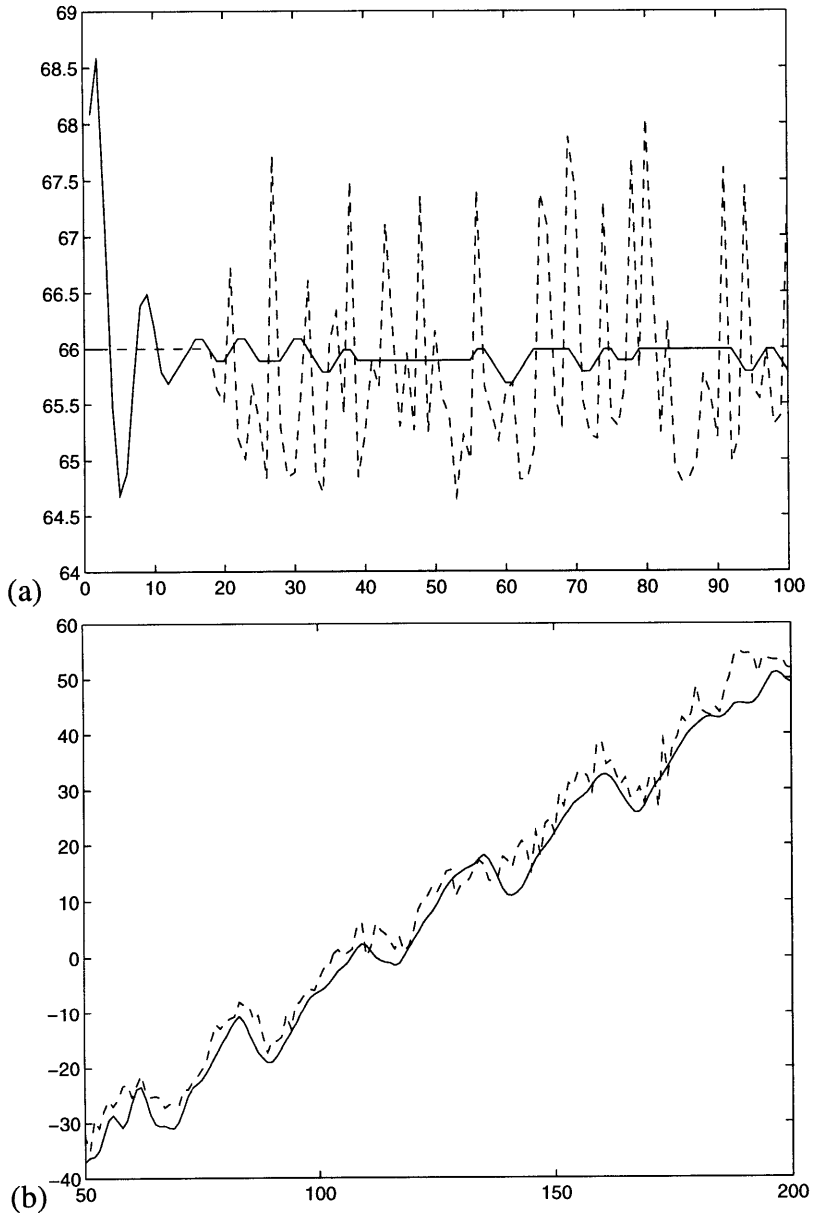


Figure 4-7: Tracking results: plot of pan angle for (a) stationary user, and (b) moving user who walked across room while oscillating head. Dashed line shows pan position under open-loop control; solid line shows pan position under closed-loop control.

(a) trained locations	(actual) Ω_1	Ω_2	Ω_3
(observed)			
Ω_1	10	0	0
Ω_2	0	10	3
Ω_3	0	0	7

(b) untrained locations	(actual) Ω_1	Ω_2	Ω_3
(observed)			
Ω_1	14	2	2
Ω_2	1	12	3
Ω_3	0	1	10

(c) all trials	(actual) Ω_1	Ω_2	Ω_3
(observed)			
Ω_1	24	2	2
Ω_2	1	22	6
Ω_3	0	1	17

Table 4.1: Results of pose classification experiment determining the pose of a user facing a display screen as the user stood at various locations in an interactive room. The task was to classify where on the video screen the user was looking; left (Ω_1), center (Ω_2), or right (Ω_3). A multiple location/multiple pose eigenspace technique was used on the output of an active camera tracking the users head, as described in the text. The confusion matrix was computed for (a) trials at trained locations, (b) trials at non-trained locations, (c) all trails. An overall success rate of 84% was achieved.

and was itself 8'x10'. Results of our system on this experiment are shown in Table 4.1. We obtained an overall success rate of 84% (63/75) for all trails, which breaks down to a success rate of 90% (27/30) on the trails at the locations were the eigenspaces were trained, and 80% (36/45) on the trails at randomly selected locations.

4.5 Summary of active gesture analysis method

In conclusion, we have shown that by integrating person tracking routines, an active camera, and multiple eigenspace pose models, we can accurately estimate the direction of gaze of a user interacting with a large screen video display. In the experiment described here, the user was on average 15' from the cameras and the display, and yet our system could discriminate pose classes which amounted to 10-15 degrees of gaze angle. Our system runs in real time, and is used in applications for interacting with virtual environments or agents that can respond appropriately to the users gaze, such as showing more information about an object of interest.

Chapter 5

Attention for Recognition

In this chapter gesture recognition is reformulated as an active process, so that a recognition task can guide the acquisition of foveated imagery. We address the problem of "what to look at", and "when" during recognition, e.g. how to guide the active camera of the previous chapter. We develop a perceptual action-selection system that implements visual attention based on reinforcement learning. Using on a simple reward schedule tied to recognition performance, this attention system learns the appropriate object (e.g., hand, head) to foveate in order to maximize recognition performance.

As described in Chapter 2, visual routines for person tracking can be used to track the head and hands of a user, and an active camera guided to obtain foveated images for gesture recognition. If we know *a priori* which body parts to foveate to detect the gesture of interest, or if we have a sufficient number of active cameras to track all body parts, then we have solved the problem. Of course, in practice there are more possible loci of gesture performance than there are active cameras, and we have to address the problem of selectively observing parts of the scene, i.e., attention. We desire a method for perceptual action selection that can learn from experience and model the fact that we only have partial observations of the actual state in the world. Inspired by the success of statistical methods

for hidden state learning in the domain of static perception (e.g., Hidden Markov Models), for active tasks we use a hidden state learning model with both action and perception: the Partially Observable Markov Decision Process (POMDP). In our system we use a POMDP formalism to define perceptual action selection in a recognition task, and solve for foveation policies using reinforcement learning methods.

As we will describe in the following sections, we have formulated an "Active Gesture Recognition" task using the POMDP framework, and have found that instance-based Q-learning is a feasible means for finding foveation policies. We define a special action to signify recognition and an associated reward function, which leads to a Q-value space that is interpretable as confidence that the target is present.

An important issue in instance- or memory-based approaches is how to find the appropriate prior experience on which to base estimates of utility. We will present an extension to the K-nearest neighbor algorithm, commonly used in instance-based methods, which removes the arbitrary selection of K and assures all experience of a particular action chain is pooled together when computing utility. This is important for our active gesture recognition task since it is essential to average experience over trials both with the putative target and with distractors.

In addition, we found that instance-based Q-learning failed on our task when we introduced multiple targets, as the Q-value space becomes quite complex when multiple sources of positive reward are present. To solve the task under these conditions, we propose a multiple model approach, where a separate Q-learning module attempts to generate foveation actions appropriate under the assumption that a particular target is present. As we will show, this method is empirically viable, and can be expressed concisely as a Q-learning system with vector-valued Q and reward functions.

5.1 The Active Gesture Recognition Problem

We define an Active Gesture Recognition (AGR) task as follows. First we assume there is some state representation of the world, describing the person configuration (or more generally, the scene configuration). Second, we assume that portions of the state of the world are only revealed via a moving fovea, and that a set of actions exist to perform that foveation. Some portion of the world state (e.g., a low-resolution view) may be fully observable. The position of the camera constitutes the perceptual state of the system; we define the full "state" to be the concatenation of world state and perceptual state. Third, we assume that, in addition to actions for foveation, there is also a special action labeled `accept`, and that the execution of this action by the AGR system signifies detection of a target world state (or target sequence of states). Finally, the goal of the AGR task is to execute the `accept` action whenever a target pattern is present, and not to perform that action when any other pattern (e.g. distractor) is present. A pattern is simply a certain world state, or sequence of world states. The AGR system should use the foveation actions to selectively reveal the hidden state needed to discriminate the target pattern.

An important problem in applying reinforcement learning to this task is that our perceptual observations may not provide a complete description of the user's state. Indeed, because we have a foveated image sensor in our interactive interface environment (as described in Chapter 4) we know that the true world state is hidden as the camera can only foveate on a single body part of the user at any given moment in time. By definition, a system for perceptual action-selection must not assume a full observation of state is available, otherwise there would be no meaningful perception taking place.

The AGR task can be considered as a Partially Observable Markov Decision Process (POMDP), which is essentially a Markov Decision Process without direct access to state [72, 48]. A POMDP includes a set of states in the world $\mathcal{S}$, a set of observations $\mathcal{O}$, a set of actions $\mathcal{A}$, and a reward function R. Note that we do not assume the system has access

to $\mathcal{S}$, nor does it know *a priori* the transition likelihood between states or the likelihood function mapping states to observations.

In the AGR task we define the reward function to provide a unit positive reward whenever the `accept` action is performed and the target pattern is present (as defined by an oracle, external to the AGR system), and a fixed negative reward of magnitude α when `accept` is performed and a distractor (non-target) pattern is being presented to the system. The parameter α expresses the trade-off between the two types of recognition errors, false alarms and misses; for the results presented in this thesis we have taken a conservative approach and set $\alpha = 10$. (This is similar to the idea of disproportionally penalizing the Q-value of perceptually aliased states, in Whitehead's Lion algorithm [85].) Zero reward is given whenever a foveation action is performed.

We wish to find a *policy*, a mapping from state (in the case of an MDP) or some function on observations (in the case of a POMDP) to action which maximizes the expected future reward, suitably discounted to bias towards timely performance. Given the reward function in the AGR task, this will correspond to a policy which successfully recognizes the target pattern. We have implemented the AGR task in two domains: a static image domain used for algorithm evaluation and pedagogical purposes, and the interactive interface domain described in previous chapters.

5.1.1 AGR on static imagery

We have implemented a version of the AGR task with target patterns which are defined as simple images. In this domain, world state is simply a single high-resolution image, and the observation consists of a subsampled version of the entire image, plus a full-resolution window over a foveated region of the image. The fovea is a fixed size rectangle and can be moved by executing a set of foveation actions. (See Figure 5-1(a)). Gaussian noise with variance σ^2 is added to both the low and high resolution observations in our implementation.

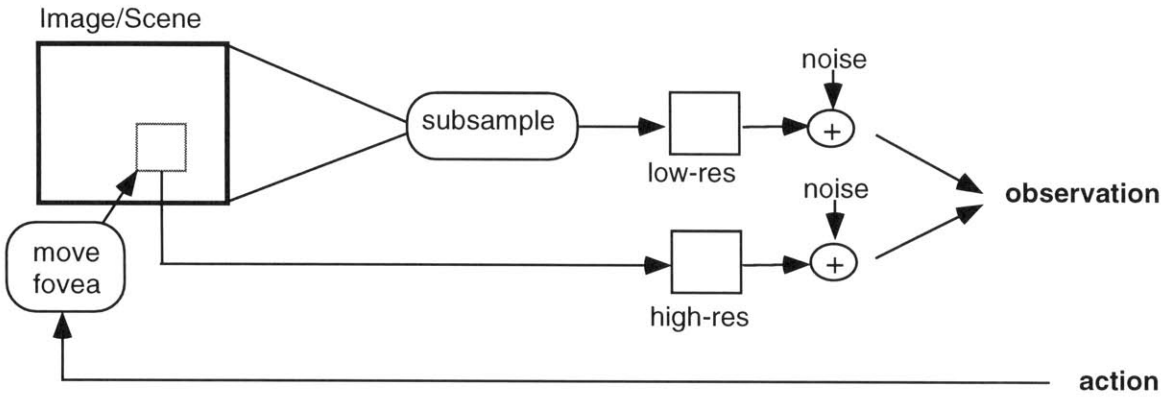


Figure 5-1: Formulation of AGR task with simple static images as world state.

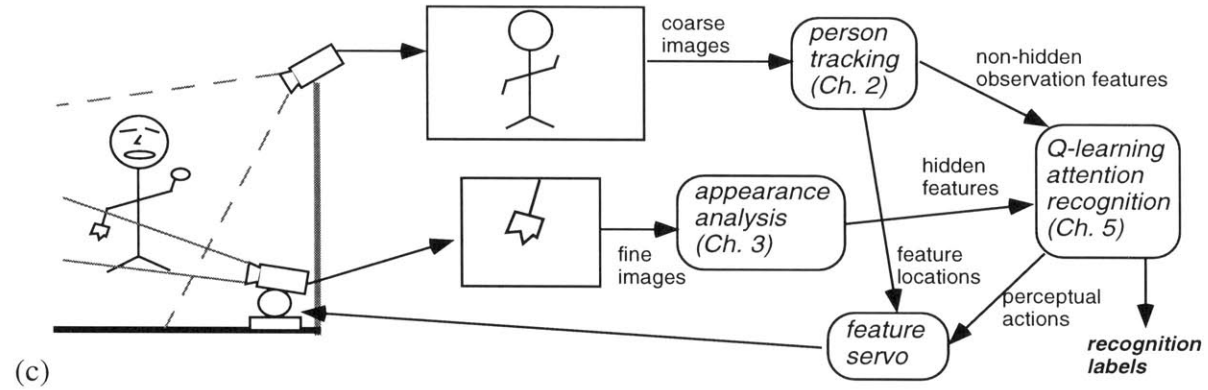


Figure 5-2: Four gesture patterns in interactive interface domain which require foveated images for discrimination. (a) Overview of AGR system in interactive domain; active camera tracks the relevant hand to identify the gesture, as described in text. (b) Output from wide field of view camera; (c) output from narrow field-of-view active camera.

feature	*values*	*observability precondition*
`person-present`	(true, false)	(always observable)
`left-arm-extended`	(true, false)	(always observable)
`right-arm-extended`	(true, false)	(always observable)
`face-foveated`	(true, false)	(always observable)
`left-hand-foveated`	(true, false)	(always observable)
`right-hand-foveated`	(true, false)	(always observable)
`face`	(neutral, smile, surprise, ...)	`face-foveated` == true
`left-hand`	(neutral, point, open, ...)	`left-hand-foveated` == true
`right-hand`	(neutral, point, open, ...)	`right-hand-foveated` == true

Table 5.1: Set of features used in POMDP formulation of Active Gesture Recognition task in interactive interface domain. This representation is computable in real-time using person tracking and gesture recognition routines described in the previous chapters.

In this domain we compute the set of foveation actions by analyzing the set of target and distractor images and determining locations which can possibly discriminate perceptually aliased pairs. Each pair of images which is not discriminable using the low-resolution (fully-observable) portion of the observation is passed to a high resolution comparison stage. In this stage the images are compared and all points which differ are marked into a candidate foveation mask. The marked points in the mask are then clustered, yielding a set of final foveation targets.

When observations are compared in this domain, we perform a test which measures whether the average pixel deviation between the two images (including both low and high resolution) is greater than two standard deviations. If not, they are considered to be "equal" observations, for the purposes of the matching function described later in this chapter.

5.1.2 AGR in the interactive interface domain

In the AGR task applied to the interactive interface domain, we assume primitive routines exist to provide the continuous valued control and tracking of the different body parts that represent/contain hidden state. We represent body pose and hand/face state using a simple

feature set, based on the representation produced by our body tracker presented in Chapter 2 and appearance-based recognition system presented in Chapter 3, and we define the world state (which we also call user state in this application) to be a configuration of the user's body pose and hand/face expression.

In this domain user state is defined by the pose, facial expression, and hand configurations, expressed in nine variables (see Table 5.1). Three of these are boolean, `person-present`, `left-arm-extended`, and `right-arm-extended`, and are provided directly by the person tracker. Three more are provided by the foveated gesture recognition system, `face`, `left-hand`, `right-hand`, and take on an integer number of values according to the number of view-based expressions/hand-poses: in our first experiments `face` can be one of `neutral`, `smile`, or `surprise`, and the hands can each be one of `neutral`, `point`, or `open`. In addition, three boolean features represent the internal state of the vision system: `head-foveated`, `left-hand-foveated`, `right-hand-foveated`.

At each timestep, the full state $s \in \mathcal{S}$ is defined by these features. An observation, $o \in \mathcal{O}$, consists of the same feature variables, except that those provided by the foveated gesture system (e.g., head and hands) are only observable when foveated. Thus the `face` variable is hidden unless the `head-foveated` variable is set, the `left-hand` variable hidden unless the `left-hand-foveated` variable set, and similarly with the right hand. Hidden variables are set to a `undefined` value.

The set of actions, $\mathcal{A}$, available to the AGR system are 4 foveation commands: `look-body`, `look-head`, `look-left-hand`, and `look-right-hand` plus the special `accept` action. Each foveation command causes the active camera to follow the respective body part, and sets the internal foveation feature bits accordingly.

5.2 Background: POMDPs

Formally, a POMDP is defined as a tuple, $< \mathcal{S}, \mathcal{O}, \mathcal{A}, T, R >$, where $\mathcal{S}$ is a finite set of states, $\mathcal{O}$ is a finite set of observations, $\mathcal{A}$ is a finite set of actions, T is model of state transition probabilities, and R models the reward associated with executing a particular action in a particular state. After executing a particular action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, the world transitions to a new state s' with probability $T(s, a, s')$, and the agent receives a reward $R(s, a)$ and observation $o \in \mathcal{O}$ with probability $O(s, a, o)$.

Given a POMDP, one wishes to construct a *policy*, π which maps states to actions, and provides an (optimal) action to be taken given the world is in a particular state. Policies may be deterministic or stochastic, with the latter being more complex but yielding higher average rewards in some cases [71].

Methods for constructing a policy based on a POMDP model can be divided into two approaches: indirect and direct. Indirect methods attempt to estimate the actual state of the world, and then treat the problem as a conventional Markov Decision problem (MDP) with full state access. Direct methods forgo recovery of the actual state and attempt to characterize optimal behavior given only actions and observations. We will discuss examples of each of these, as well as hybrid approaches that use weak methods to model internal state.

Indirect methods combine a predictive forward model that recovers an estimate of absolute state with a conventional MDP learning method. Given absolute state, recovering an optimal policy for a MDP is a well-studied problem. Q-Learning is one widely used method based on a reinforcement learning paradigm for finding an optimal policy. The Q function maps state and action pairs to a *utility* value, which is the expected discounted future reward given that state and action. Optimal Q values can be computed on-line using a *Temporal Differences* method [73], which is an incremental form of a full Dynamic Programming approach [7]. In the case of deterministic state transitions, the optimal Q

function must satisfy

$$Q(s,a) = r + \gamma \max_{a \in \mathcal{A}} Q(s',a)$$

where s' is the next state after executing action a in state s. To find an optimal Q, the difference between the two sides of this equation is minimized:

$$Q(s,a) \leftarrow Q(s,a) + \eta(r + \gamma \max_{a \in \mathcal{A}} Q(s',a))$$

This has been shown to converge to optimal policies when the environment is Markovian and the Q-function is represented literally as a lookup table [83].

Several approaches have been proposed which combine a predictive model to transform a POMDP to a MDP and then use Q-learning to construct a policy function. The Perceptual Distinctions Approach (PDA) developed by Chrisman [22] uses a state-splitting approach to construct a model of the domain. A modified *Expectation-Minimization* (EM) algorithm was used to alternate between maximizing the model probability given a fixed number of internal states, and splitting perceptually aliased states into new states. A similar method was developed by McCallum, the Utile Distinction Memory approach (UDM), which split states based on the predicted reward [52]. Given a state estimator learned via PDA or UDM, or simply computed from T and O, an optimal policy can be constructed [21].

However these indirect methods have been criticized as being computationally intractable in realistic environments [44]. Indeed, the empirical results suggest that a prohibitive amount of training time is required with these algorithms [53].

Rather than convert a POMDP to a MDP through state estimation, direct methods update a value function over observations without recovering absolute state. Jaakkola et. al. present a Monte-Carlo algorithm for policy evaluation and improvement using Q-values defined over actions and observations [44]. They provide a recursive formulation of the Monte-Carlo algorithm, and prove the policy improvement method will converge to at least

a local maximum.

A hybrid approach between these two extremes has been explored by Lin [46] and McCallum [53]. These methods use a memory-based approach to identifying state. Lin's *window-Q* algorithm supplies a history of the N most recent observations and actions to a conventional Q-learning algorithm. The hope is that perceptually aliased states can be discriminated by knowing a portion of the state history. A significant drawback to this approach is the use of fixed window size. McCallum's *Instance-Based* state identification method extends this approach to use windows of varying length depending on the current sequence. Q-learning is performed over a space of states that is defined using a sequence similarity measure over the observations. The instance-based approach seems to have the best empirical results among those published, and has the advantage that the algorithm is intuitive and simple to implement.

5.3 Hidden-State Reinforcement Learning

To find policies for AGR tasks we have implemented an memory-based method for hidden state reinforcement learning, based on McCallum's instance-based approach, which we will describe in detail. This method performs Q-learning [73, 83], but replaces the absolute state with a distributed state representation. (As is usual with Q-learning, we do not assume any access to or knowledge of the set of states $\mathcal{S}$, or the likelihood functions $T(s,a,s')$,$O(s,a,o)$.) Given a history of action, reward, and observation tuples, $(a[t], r[t], o[t]), 0 \leq t \leq T$, a Q-value is also stored with each timestep, $q[t]$, and Q-learning is performed by evaluating the similarity of recently observed tuples with sequences farther back in the history chain. Q-values are computed, and the Q-learning update rule applied, maintaining this distributed, memory-based representation of Q-values.

As in traditional Q-learning, at each timestep the utility of each action in the current state is evaluated. If full access to the state was available and a table used to represent Q

values, this would simply be a table look-up operation, but in a POMDP we do not have full access to state. Using McCallum's Nearest Sequence Memory (NSM) algorithm, we instead find the K nearest neighbors in the history list relative to the current time point, and compute their average Q value. For each element on the history list, we compute the sequence match criteria with the current time point, $M(i,T)$, where

$$M(i,j) = S(i,j) + M(i-1,j-1) \quad \text{if } S(i,j) > 0 \text{ and } i > 0 \text{ and } j > 0$$

$$0 \quad \text{otherwise}\,.$$

We modify McCallum's algorithm slightly (for reasons made clear in Section 5.8) and define $S(i,j)$ to be 1 if $o[i] = o[j]$ or $a[i] = a[j]$, 2 if both are equal, and 0 otherwise. Using a superscript in parentheses to denote the action index of a Q-value, we then compute

$$Q^{(a)}[T] = (1/K)\sum_{i=0}^{T} v^{(a)}[i]q[i]\,, \tag{5.1}$$

where $v^{(a^*)}[i]$ indicates whether the history tuple at timestep i votes when computing the Q-value of a new action a^*: $v^{(a^*)}[i]$ is set to 1 when $a[i] = a^*$ and $M(i-1,T)$ is among the K largest match values for all k which have $a[k] = a^*$, otherwise it is set to 0. Given Q values for each action the optimal policy is simply

$$\pi[T] = \arg\max_{a \in \mathcal{A}} Q^{(a)}[T]\,. \tag{5.2}$$

The new action $a[T+1]$ is chosen either according to this policy or based on an exploration strategy. In either case, the action is executed yielding an observation and reward, and a new tuple added to the history. The new Q-value is set to be the Q value of the chosen

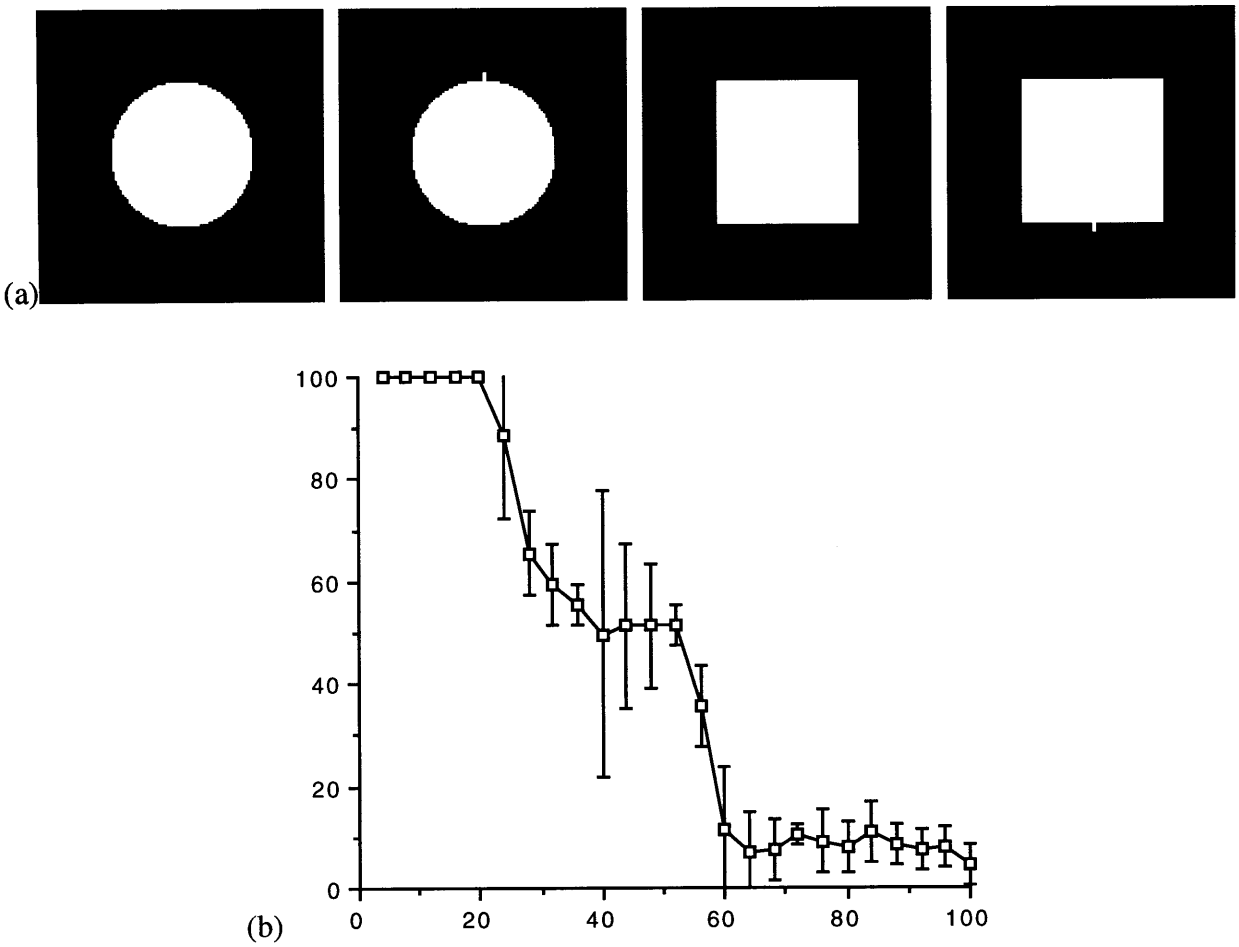


Figure 5-3: AGR results for simple imagery targets: (a) set of four target patterns, where the first and last pair are indistinguishable at low-resolution (b) recognition performance (percentage of correct trials) for varying amounts of observation noise.

action, $q[T+1] = Q^{(a[T+1])}[T]$. The update step of Q learning is then computed, evaluating

$$U[T+1] = \max_{a \in \mathcal{A}} Q^{(a)}[T+1]\,, \tag{5.3}$$

$$q[i] \leftarrow (1-\beta)q[i] + \beta(r[i] + \gamma U[T+1])\,, \tag{5.4}$$

for each i such that $v^{(a[T+1])}[i] = 1$.

Figure 5-3 shows the results using this hidden-state Q-learning algorithm on an AGR

task using image-based world state. In this example four different world states were used for the target and distractors; varying amounts of noise were added to the actual observations at run-time. We ran the system by randomly selecting one of the patterns as target, and using the other three as distractors. This Q-learning system easily learns the correct foveation policies to actively recognize these targets; e.g., to foveate up to recognize one of the square targets, and to foveate down to recognize one of the circle targets. For moderate levels of noise, each pattern can be easily discriminated from all the others, however, as σ exceeds 20.0[1], the high-resolution information in the signal becomes insufficient to discriminate the two square targets and the two circle target, so performance drops by half. As σ approaches 60.0, the low-resolution information becomes indistinguishable, and all the targets are confused, leading to a recognition rate of zero.

5.4 Deterministic Exploration and Training Strategy

This instance-based Q-learning can find optimal policies to solve AGR tasks both in the image-based example given above, and in the interactive interface domain. Figure 5-4 shows the recognition performance using simple target and distractor gestures in the latter domain. In these simple cases an exploration model based on random search (e.g., explore a random action either when the maximum utility is negative or with random probability, typically $p = 0.05$) is sufficient to find a good policy.

The targets in Figures 5-3 and 5-4 are all "simple" because they can be discriminated with a single foveation action and resulting observation. However, with more complicated targets random search has a considerably more difficult time finding a good policy. Empirically we found that for targets which required more than one foveation action (in addition to the `accept` action) to discriminate from distractors, a good policy could not be found with

[1]The image range was 0..255.

(a)

gesture pattern	scene	features
0		person-present, left-arm-extended, left-hand-point
1		person-present, left-arm-extended, left-hand-open
2		person-present, right-arm-extended, right-hand-point
3		person-present, right-arm-extended, right-hand-open

(b)

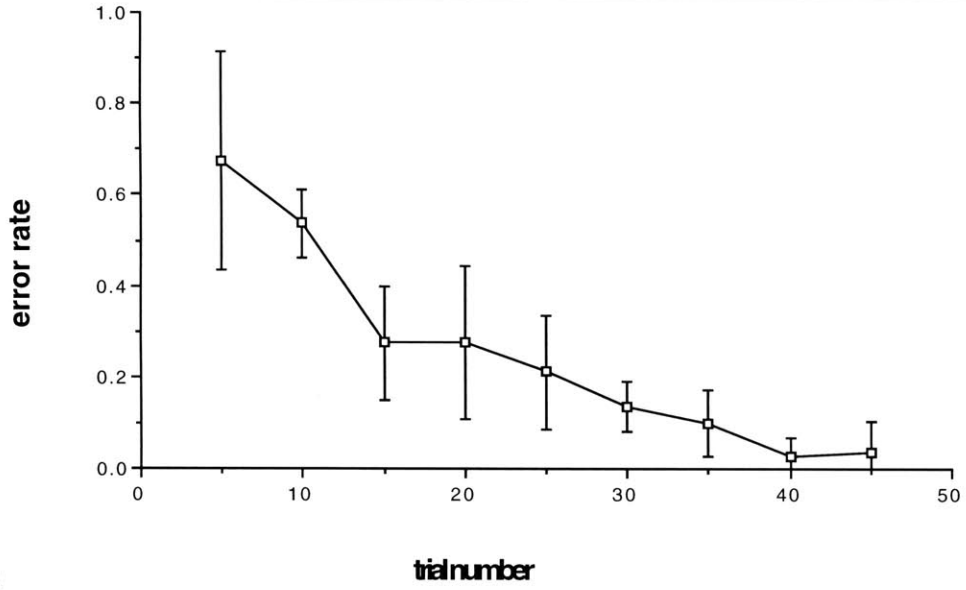


Figure 5-4: Simple target performance with random exploration (a) set of targets which can be discriminated using a single foveated observation. For each run, one gesture was randomly chosen as target, and the other three used as distractors. (b) results averaged over 100 runs; error rate is plotted as a function of trial number (proportional to timestep). Random exploration is sufficient to learn a recognition policy in these cases.

(a)

gesture pattern	scene	features
5		person-present, face-surprise
6		person-present, left-arm-extended, left-hand-open, right-arm-extended, right-hand-open
7		person-present, left-arm-extended, left-hand-open, right-arm-extended, right-hand-point
8		person-present, left-arm-extended, left-hand-point, right-arm-extended, right-hand-open

(b)

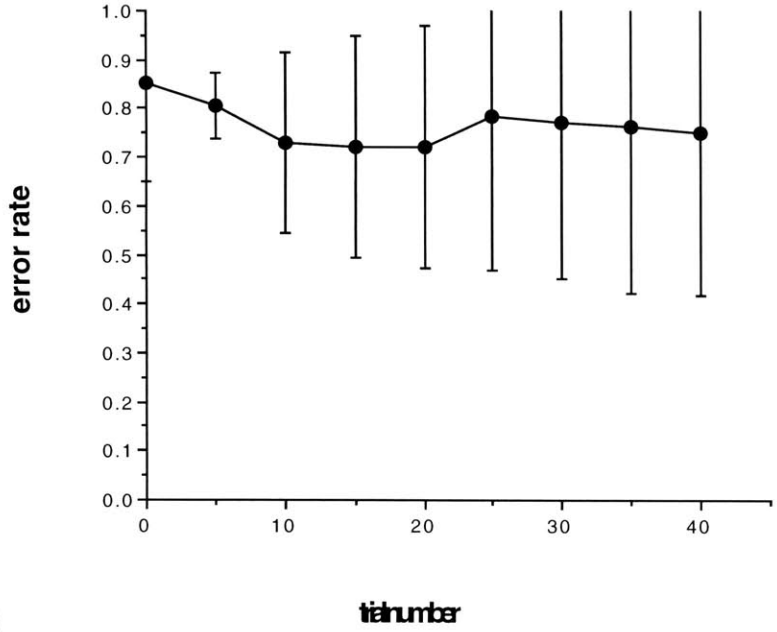


Figure 5-5: Complex target performance with random exploration (a) set of targets which cannot be discriminated by a single foveated observation (b) error rate plotted as a function of trail number. Random exploration is insufficient to learn a recognition policy in these cases.

random exploration. Figure 5-5 shows an example of random exploration failing to learn a policy for targets which require more sophisticated discrimination polices.

Our Q-learning system can become stuck in local minima if it follows a purely greedy strategy (e.g. no exploration). Also, varying amounts of exploration experience are required for different sets of targets and distractors for the system to converge to an adequate level of performance. To learn the globally optimal policy the system must experience the global optimal path at least once; this may never happen without exploration, and may take a considerable amount of time with purely stochastic exploration.

To overcome this problem, we have devised deterministic exploration regimes which provide quick convergence to an optimal policy, given random presentation of target and distractors. Our exploration strategy is based on the observation that the system needs to build an accurate evaluation of the utility (Q-value) of the `accept` action given different foveated observations. Depending on the complexity of the recognition task, which is determined by the target and distractor set, some number of foveated observations is ideally needed to resolve the target identity.

We define the complexity of an AGR problem to be the smallest number of foveated observations sufficient to uniquely identify the target gesture from the distractor patterns. A 0-th order AGR problem is one in which the target is uniquely identifiable without any foveated observations; a 1-st order problem requires a single foveated observation (say a facial expression); a 2-nd order problem requires foveating on two body parts, etc.[2]

In a 0-th order problem, a sufficient (and trivial) deterministic exploration regime is to execute the `accept` action under all observation conditions and compute Q-values as given above. In a higher order problem, a single positive reward is insufficient to determine policy, since the system may well be perceptually aliased. For the system to learn this, it must execute each possible foveation action (or foveation sequence for problems of 2-nd

[2]For example, to discriminate gesture 6 from gesture 7,8 in the example shown in Table 5-9 requires a foveated observation of both hands, so this is a second-order AGR task.

and higher order) several times with both target and distractor before assessing the reliability of the accept command.

Our deterministic exploration strategy is to explore this foveation/accept space assuming a problem of fixed order and ensure that the system will experience all relevant action chains both with the target and with the distractor. To train our system, we follow a three stage procedure. First, we provide the system with the deterministic experience described above, e.g., the system selects actions according to to a pre-scheduled list which enumerates action chains up to a predetermined depth (typically 2, in our experiments) followed by an `accept` terminal. Each action chain is executed d (a given parameter) times for each target/distractor, with the intention being the system should have sufficient initial experience with both target and distractor to determine the true utility of each action. Throughout the training phase, we randomly switch between target and distractors, so that the utility of actions will be averaged over trials with each.

Second, we run a batch version of the Q-learning update rule until the utility values converge. We cycle through the memory structure, updating the utility of each q value as given in Eq. 5.4. When the cumulative change is less than a fixed value or stops decreasing, we consider the utility values to have converged,.

Finally, we let the system run according to policy, (i.e. select the action with maximal Q-value) and evaluate the recognition performance. During testing we randomly switch between target and distractors, and record trials when the system executes `accept` on a distractor, or fails to execute `accept` on a target. These trials are marked as errors; all other trials are considered correct. When the time-averaged performance rate stops increasing, we freeze the utility values of the entire system, re-measure recognition performance, and record this performance measure as the final value.

5.5 Variable-K Nearest Neighbor Algorithm

The key to successful application of instance-based Q-learning algorithm is the identification of the appropriate instances (e.g. memories) on which to base estimates of the utility of a particular new action. In our AGR task, with deterministic exploration as given above, it is especially important that the utility of a particular action given a particular observation history be estimated from previous experience with that action and observation history *on both target and distractor trials*. Estimates of utility which are based only trials with the target present will greatly overestimate the utility of all actions, since with the target present by definition any action chain followed by `accept` will yield positive reward. Conversely, estimates of utility based only on trials with distractors will grossly underestimate the true utility. Effective learning occurs in our system only when the utility of a particular action is considered over trials with both target and distractor.

This poses a problem for an instance-based algorithm which finds voting instances using a strict K-nearest neighbor method, since it becomes critical to have chosen the correct value of K. A value which is too small will miss some relevant experience, and yield an unreliable utility estimate; a value which is too large will merge together instances which are actually different states, returning us to the problem of perceptual aliasing. Since the number of instances (memories) which are relevant to a particular utility estimate will vary greatly, no fixed value of K will suffice.

A simple example illustrates the problem: consider the estimation of the utility of *accept*, when the previous observation/action sequence was to foveate the face and observe a neutral expression. Assume that the target and all distractors all have a face with neutral expression, so this should be of no use in discriminating target from distractor. During the training phase the system will have explored this sequence several times, some with the target and some with the distractor. When match similarities are computed, the memories with maximum match value will indeed all correspond to the relevant previous experience.

There may be 20 such previous experiences, but a strict K-nearest neighbor algorithm with K=4 will ignore 16 of them. If the 4 memories happen to miss experiences with the target, which is not unlikely, then the computed utility will be incorrect.

Rather than set K very large and risk merging truly different states, we instead propose a new algorithm, which we call *Variable K Nearest Neighbors*. Variable-K differs from strict-K in that all memories with a given match length are included in the voting set, e.g., are pooled together when estimating utility.

In the Variable-K algorithm, we assert a minimum percentage kp of experience to be included. (If there is no noise in the match function, this can be effectively set to zero; in our experiments we used $kp = 5\%$) We histogram the match lengths of all memories, and sort them from largest to smallest. We then find the match length value m' such that the $kp\%$ of memories have larger match lengths than m'. We then subtract a fixed amount kt to this match length value, and include all memories with match value greater than $m' - kt$ into the voting set. Figure 5-6 depicts the threshold computation used in the Variable-K algorithm.

The effect of this algorithm is to find a percentage of nearest neighbors with largest match values, as does strict-K, but then to include all other memories with roughly the same match length. In the above example, all 20 experiences would be included in the voting set, and utility accurately estimated. In practice we have used $kt = 1$, and found this algorithm greatly improved the performance of instance based Q-learning given the multiple trial (e.g. nonstationary) structure of our AGR task.

5.6 Multiple Model Q-Learning Algorithm

With a variable-K algorithm, we found the instance-based hidden state reinforcement learning described above to be an effective way to perform action-selection for foveation when the task is recognition of a *single* object from a set of distractors. E.g., the data in Figures

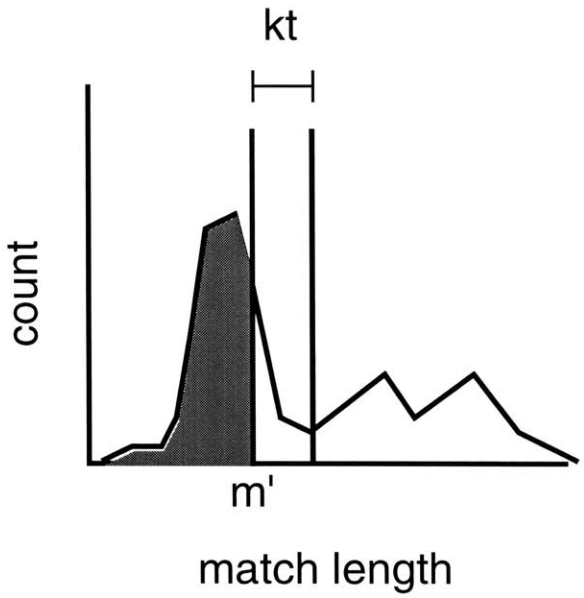


Figure 5-6: Variable K nearest neighbor algorithm. To compute voters for computation of utility, the match value of each memory compared against the current time point is computed. These values are histogrammed (displayed in order of decreasing match length) and a preliminary threshold corresponding to the match length value m' which kp percent of the match length values are larger than. (Here the match length histogram is plotted for decreasing match length, and the shaded area corresponds to $kp\%$ of the total histogram area.). We then subtract kt to this preliminary threshold to find the final threshold. All memories with match length greater than or equal to $m' - kt$ are included in the voting set.

5-3,5-4, and 5-5 reflect averages over multiple trials, each of which picked one gesture as target and the remaining three as distractors. Thus, a policy was learned for recognizing a single gesture. In none of these cases did the system learn a foveation policy sufficient to recognize *multiple* gestures.

In fact, we did not find that this type of system performed well when the AGR task was extended to include more than one target gesture. When multiple `accept` actions were added to enumerate the different targets, we were not able to find exploration strategies that would converge in reasonable time. This is not unexpected, since the addition of multiple causes of positive reward makes the Q-value space considerably more complex. To remedy this problem, we propose a multiple model Q-learning system. In a multiple model approach to the AGR problem, separate learning agents model the task from each target's perspective. Conceptually, a separate Q-learning agent exists for each target, maintains its own Q-value and history structure, and is coupled to the other agents via shared observations. Since we can interpret the Q-value of an individual AGR agent as a confidence value that its target is present, we can mediate among the actions predicted by the different agents by selecting the action from the agent with highest Q-value (Figure 5-7).

Formally, in our multiple model Q-learning system all agents share the same observation and selected action, but have different reward and Q-values. Thus they can actually be considered a single Q-learning system, but with vector reward and Q-values. Our multiple model learning system is thus obtained by rewriting Eqs. (5.1)-(5.4) with vector $q[t]$ and $r[t]$. Using a subscript j to indicate the target index, we have

$$Q_j^{(a)}[T] = (1/K) \sum_{i=0}^{T} v^{(a)}[i] q_j[i] \tag{5.5}$$

$$\pi[T] = \arg\max_{a \in \mathcal{A}} \left(\max_j Q_j^{(a)}[T] \right) . \tag{5.6}$$

Rewards are computed with: if $a[T] =$ `accept` then $r_j[T] = R(j,T)$ else $r_j[T] = 0$;

$R(j,T) = 1$ if gesture j was present at time T, else $R(j,T) = -\alpha$. Further,

$$U_j[T+1] = \max_{a \in \mathcal{A}} Q_j^{(a)}[T+1] \,, \tag{5.7}$$

$$q_j[i] \leftarrow (1-\beta)q_j[i] + \beta(r_j[i] + \gamma U_j[T+1]) \quad \forall\, i \text{ s.t. } v^{(a[T+1])}[i] = 1 \,. \tag{5.8}$$

Note that our sequence match criteria, unlike that in [53], does not depend on $r[t]$; this allows considerable computational savings in the multiple model system since $v^{(a)}$ need not depend on j.

5.7 The action overlap problem with multiple-Q

The multiple-model Q-learning algorithm is simple to implement and scales well in terms of adding little to the cost of training. As long as modules corresponding to different targets call for the same action in each perceptual condition that has significant utility, one can add arbitrarily many modules and use the policy given in Eq. 5.6. For example, Figure 5-8(c) shows the performance for Q-learning on a fixed set of target/distractor patterns running separately for each target, and running with the multiple model algorithm. As the recognition policies for each of these targets never "overlap" in the sense that they never call for different actions in the same perceptual condition, they never interfere, and the recognition rate is perfect in each case.

However when the policies for different targets do overlap, they can easily interfere and disrupt recognition performance. Figure 5-8(e) shows results with a set of targets whose recognition policies overlap; to recognize target I the optimal policy is to foveate the top of the circle when a coarse circle is observed, but to recognize target III the optimal policy is to foveate the right of the circle under the same perceptual condition. When both policies are active, they can interfere with each other and significantly reduce recognition performance. The pathology is that with a deterministic policy, when there are competing

(different) actions with substantial utility in different modules the system will always pick one action and not the other, causing the second policy to never be executed (and thus the corresponding target never recognized).

We have investigated three approaches to overcoming this "action overlap" problem. Common to all of them is the addition of a reset behavior, which returns the camera position to the initial condition whenever the current maximum utility value falls below zero (e.g. the system is lost). With this, direct deterministic learning, stochastic policies, and a new persistence algorithm all can solve the action overlap problem.

5.7.1 Learning to overcome action overlap

With a strict deterministic policy, it is still possible for the system to learn to act differently after it tries an action, fails to recognize the target, and resets to the initial condition. The variable length match voting scheme described above allows the length of an history chain matching to compute utility to vary for different actions. Thus, after resetting, the utility of repeating the action taken prior to resetting will ideally be lower. The system can infer this from previous experience with repeating the action twice, if this appropriate prior experience is available.

The system will naturally experience the effect of multiply repeating the particular action with an intervening reset, since this is the "pathological" behavior described above. The utility of the alternate (overlapped) action will remain the same, since it will still be matching against the same experience after the reset as it did at the first timestep with the given trial. The second action will this get chosen after the system explores the first action and resets; the "prior experience" matched to compute the utility of the first action will then consist of prior trials where the system tried a dual repeat of the first action.

In this way, deterministic multiple-model policies can overcome the action overlap problem. However, in practice it can take a considerable amount of time to learn this

solution strategy. We thus turn to alternative means of dealing with the problem, using stochastic and/or persistent policies.

5.7.2 Stochastic policies

Another solution to the overlap problem is to use a stochastic policy [71]. With a stochastic policy and the reset function given above, some fraction of the time the alternate action (the one with lower utility) will be executed, so there is a non-zero probability the correct policy to recognize the second target will be chosen.

However this may take a long time, and if there are multiple states of overlap in the policies (e.g. they overlap not just on the first action, but on subsequent actions as well) it could take a considerable amount of time to allow a "overlapped" policy to fully execute. (E.g. to execute as if it were the only active policy).

5.7.3 Persistent and exclusive modules

This leads us to an explicit algorithm which guarantees the policy in each Q module will fully execute (persistence), and a further variant which prevents a module which had control and failed to find non-zero reward to regain control until either all modules have had a chance to execute or the world state changes (exclusivity).[3] (This assumes a signal exists to indicate a change in world state; if no signal is available we do not use the exclusivity algorithm).

We use two boolean variables, A_k, and C_k, to indicate whether a Q module k is *active* and/or *current*. We modify the policy of Eq. 5.6 such that the new action is selected only from modules which are marked current.

$$\pi[T] = \arg\max_{a \in \mathcal{A}} \left(\max_{\{j | C_k = 1\}} Q_j^{(a)}[T] \right) . \tag{5.9}$$

[3]These algorithms were based on the helpful suggestion of Prof. Aaron Bobick.

We update the current value after each timestep, according to which modules predicted the chosen action. Initially, and after each `accept` or reset action, C is set the value of A in each module; A is initially set to 1, and is reset to all 1's whenever the world state changes. At each subsequent timestep C is cleared for each module that did not call for the chosen action:

$$C_j = 0 \text{ if } \left(\max_{a \in \mathcal{A}} Q_j^{(a)}[T] \neq \pi[T]\right), \forall j .$$

This has the effect of guaranteeing that one particular policy will be carried out to its conclusion (either generating `accept` or encountering uniform non-positive utility).

Further, when the reset action is performed (forced) by the system, all modules which are current are deactivated. E.g., on reset,

$$A_k = A_k \text{ and } (\text{ not } C_k)$$

and on changing world state, or when all modules are inactive, we set

$$A_k = 1 \quad \forall k .$$

This ensures that the correct module will get eventually get control and exercise its corresponding policy, even if other modules initially have higher utility values.

Figure 5-8(c) shows the recognition performance using this approach on the same set of image-based targets/distractors used in Figure 5-8(a,b). The persistence/exclusivity algorithm prevents the dithering associated with action overlap, and leads to the expected recognition performance.

5.8 Multiple Model Active Recognition Results

Finally, we show the effects of our learning system on a complex multiple target task in the interactive interface domain. As described earlier, we use the person tracking methods described in Chapter 2, and the active gesture analysis methods presented in Chapter 3 and 4, to derive a real-time, foveated representation of a user. Figure 5-2 depicts the overall system for active tracking of a user, and shows example imagery obtained by the system.

The use of this feature-based representation exploits the generalization ability of the person tracking and spatio-temporal pattern perception system described in earlier chapters. Though this is by design, and is not learned per se by the Q-learning method, this allows the overall system to easily perform generalization. Each instance of the feature vector depicted in 5.1 represents a wide class of input imagery, and experience the Q-learning system has will any of them will generalize to later performance with the others.

Again, we note that the feature-based representation of user state is perceptually aliased, in the sense of [85]. Our perceptual aliasing is due to the fact that we have a foveated sensor, which can not observe all parts of the scene simultaneously. The body and arm features are observable in the coarse view (i.e. fully observable) but the hand and face features require a high-resolution foveated view. The latter can only be seen, and are thus only set in the feature vector, when the camera is pointing at the appropriate body part.

We collected example output from the perceptual systems, and ran off-line experiement to test the recognition performance of the learning system. The set of gesture patterns collected and used in this experiment is shown in Figure 5-9. We ran our learning system as defined above, keeping constant the parameters $\beta = 0.1, \gamma = 0.5, \alpha = 10$. Figure 5-10 shows results for each target plotted for different amounts of initial exploration experience according to the parameter d, which represents the average number of times the system initially explored a given action chain, divided by the number of targets and distractors. Targets which are easily discriminable have good performance even with little initial train-

ing, but the second-order targets require values of $d > 2$ to obtain adequate performance. Figure 5-11 shows the results for this experiment cumulative across all targets; for $d > 2$ accuracy rates in excess of 95% were obtained.

In our present implementation, real-time performance ($>$ 5Hz) on the AGR task is possible as long as the history list representation can be maintained in main memory on a contemporary workstation. Currently, approximately ten thousand timesteps can be stored in main memory, so real-time performance is possible for runs of up to 30 minutes. A topic of future work is to extend the system to selectively forget and/or merge portions of the history list, so that the system can run indefinitely in constant-size memory.

5.9 Discussion of attention system

The examples we have shown demonstrate the ability of hidden-state Q-learning, using an instance-based utility representation, to learn where to look to discriminate target from distractor patterns. Conceptually, the system constructs an action-selection mechanism which operates by comparing prior experiences which had similar recent action/observation history compared to the most recent time point. Since observations include both the state of the user and the state of the perception system (e.g. where the active camera is looking), this means that the system builds predictive models that can combine both what the user will do next, and where to look to confirm the relevant part of the hidden state.

With static targets, as we have shown, (implicit) state transition modeling captures only the foveation dynamics. Conversely, if the high-resolution camera were fixed but the targets were dynamic, then the underlying states would reflect the temporal structure of the target (or at least as could be best predicted). As mentioned in the introduction, this latter configuration is essentially that of an HMM. The major difference would be the lack of an assumed number of internal (hidden) states in the POMDP approach. Modeling the temporal structure of a dynamic signal is important in any interaction domain where

temporal context can aid recognition; a good example of this is the domain of driving interaction, where Pentland and colleagues [60] have shown human performance can be effectively and usefully predicted in an interactive system.

Because our system is based on Q-learning, it has the ability to learn from delayed rewards. This is important, since we want our system to be able to learn plans even when it is only rewarded only at the end of a trial. In the case of the AGR system, this is the case as reward is only distributed on the performance of the recognition action, which happens at the end of a sequence of foveation actions. Learning the correct foveation actions is the key problem faced by the system; were it only to model the expected instantaneous reward, it could never learn foveation behavior. In the Q-Learning framework, however, reward is propagated back through the utility structure, so future expected reward can effect current action choices.

The use of a reinforcement/reward paradigm offers considerable flexibility; in addition to application to pattern recognition, one can envision a range of interaction regimens that are applicable to this learning framework. Higher-level actions in an interface can and should be included into the POMDP action set, which may lead to a more powerful and robust system than is possible with a system that relies on an intermediate gesture representation to conditions these higher-level actions. Simply adding additional terms to the reward function to encourage speedy recognition of certain default or priority actions would be perhaps the most straightforward way of bringing higher-level interface design issues directly into the learning framework. Much in the way that general-purpose optimization frameworks have proven powerful for scene description and structure recovery, it is our belief that reinforcement protocols hold promise for modeling a wide range of performance in interactive systems.

In summary, gesture recognition systems often require high-resolution images of hands and faces when used in unconstrained environments (including ours, described in Ch. 3), but the issue of how to decide what to foveate is often left unresolved or addressed only with

ad-hoc solutions. In this chapter we have resolved this, and have developed a mechanism to selectively foveate salient body parts in an active gesture recognition task where foveation is guided by recognition performance. We adopt a Partially Observable Markov Decision Process formalism, using an action set comprised of foveation actions as well as a special recognition action. Execution of this action is rewarded based on whether the target is in the scene. We use instance-based hidden state reinforcement learning to learn a policy which models both when to execute the recognition action and what foveation commands to execute to properly discriminate the target gesture. To accurately pool experience when estimating the utility of a new action, we implement a variable-K nearest neighbor algorithm which includes all experience with a given action chain. This method works for single target recognition tasks, however multiple targets create undesired complexity in a scalar Q-value space due to multiple sources of positive reward. To overcome this we define a multiple-model Q-learning paradigm with vector-valued Q and reward functions. With this framework, our system can actively recognize targets from a set of gestures in real-time.

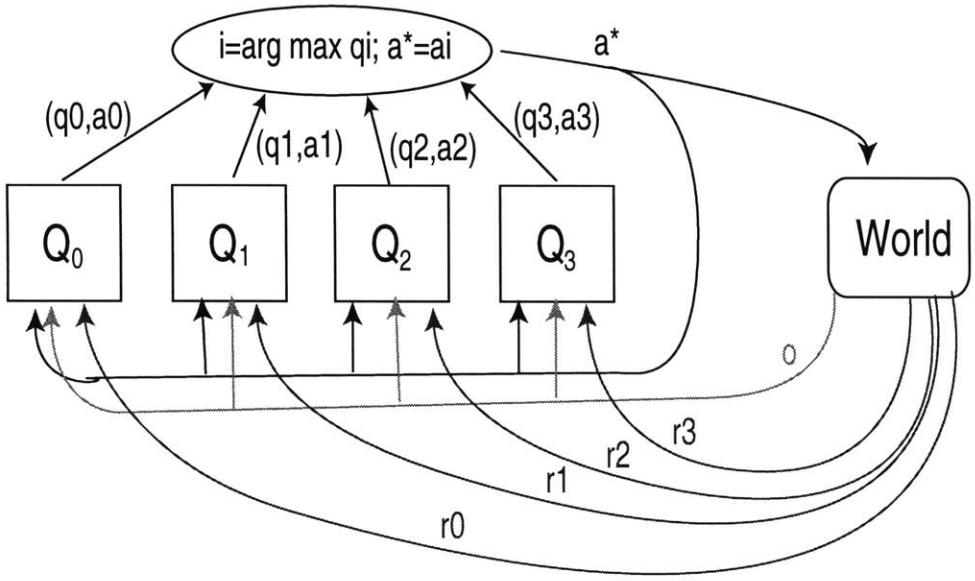


Figure 5-7: Multiple model Q-learning: one Q-learning agent for each target gesture to be recognized, with coupled observation and action but separate reward and Q-value.

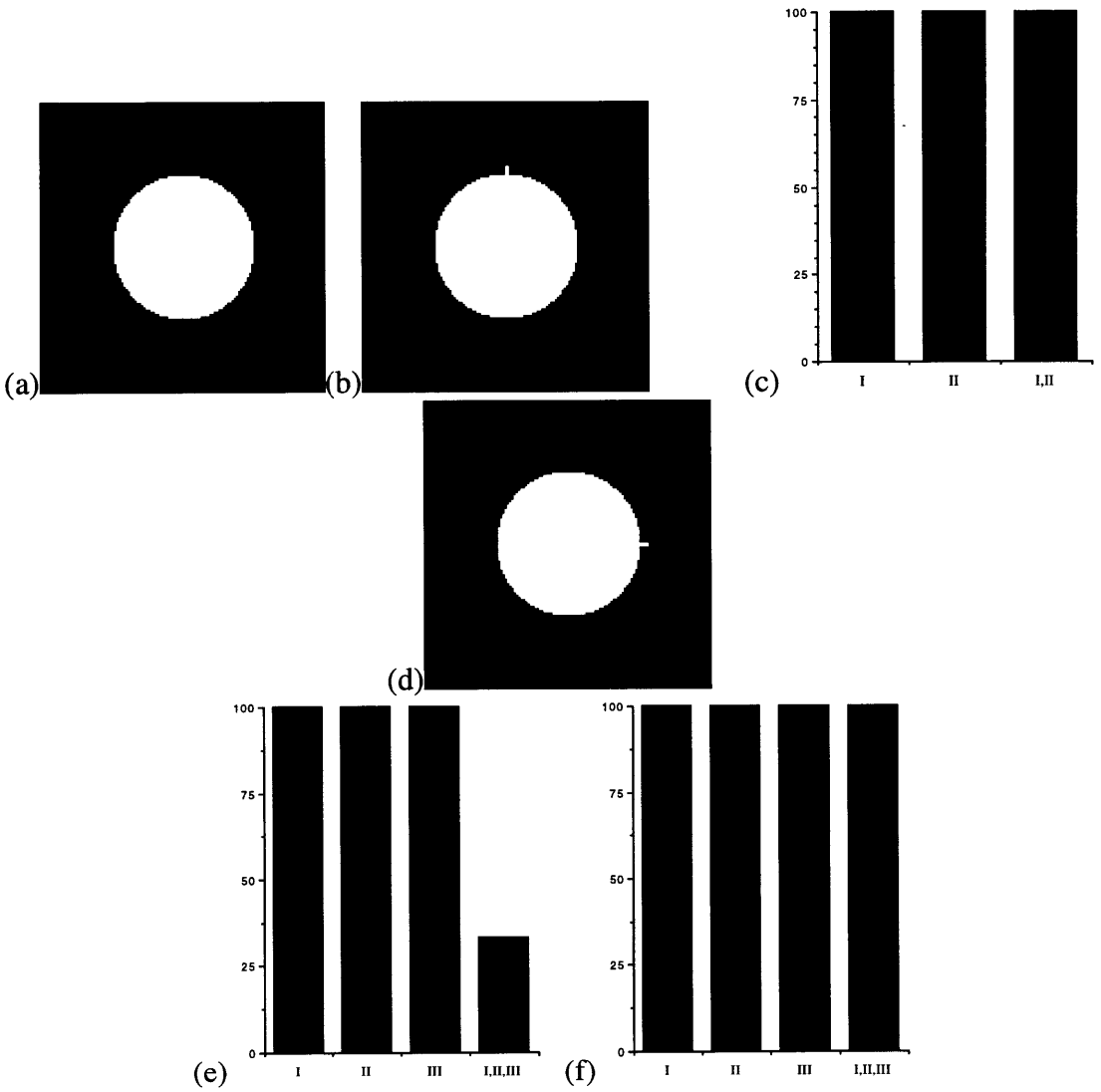


Figure 5-8: Performance for increasing number of Q-learning modules, (a) target I, (b) target II, (c) recognition results of target I alone (with target II the distractor), and target II alone (with target I the distractor), and with both target I,II using the multiple model Q-learning algorithm. For these two targets, no degradation of performance is found with the multiple model algorithm. (d) target III, and (e) performance for each of the three targets alone, (with the other two as distractors), and for all three combined using the multiple model approach. With these three targets, there is the "action overlap" problem, and performance is degraded using a greedy policy. (f) performance with a persistent/exclusive policy. Using this algorithm algorithm, optimal performance is obtained.

gesture pattern	scene	features
0		person-present, left-arm-extended, left-hand-point
1		person-present, left-arm-extended, left-hand-open
2		person-present, right-arm-extended, right-hand-point
3		person-present, right-arm-extended, right-hand-open
4		person-present, face-smile
5		person-present, face-surprise
6		person-present, left-arm-extended, left-hand-open, right-arm-extended, right-hand-open
7		person-present, left-arm-extended, left-hand-open, right-arm-extended, right-hand-point
8		person-present, left-arm-extended, left-hand-point, right-arm-extended, right-hand-open

Figure 5-9: The set of targets used in the multiple-model active gesture recognition example.

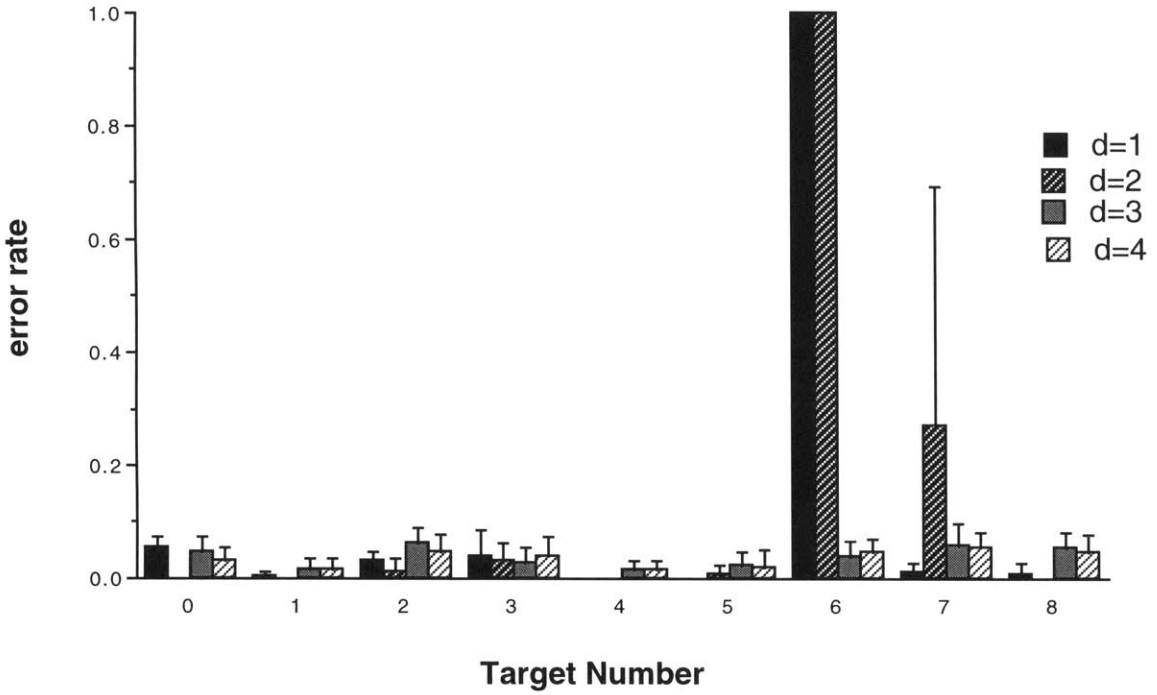


Figure 5-10: Recognition results of multiple model/target example, plotted for each target, with varying amounts of initial training (d=1..4).

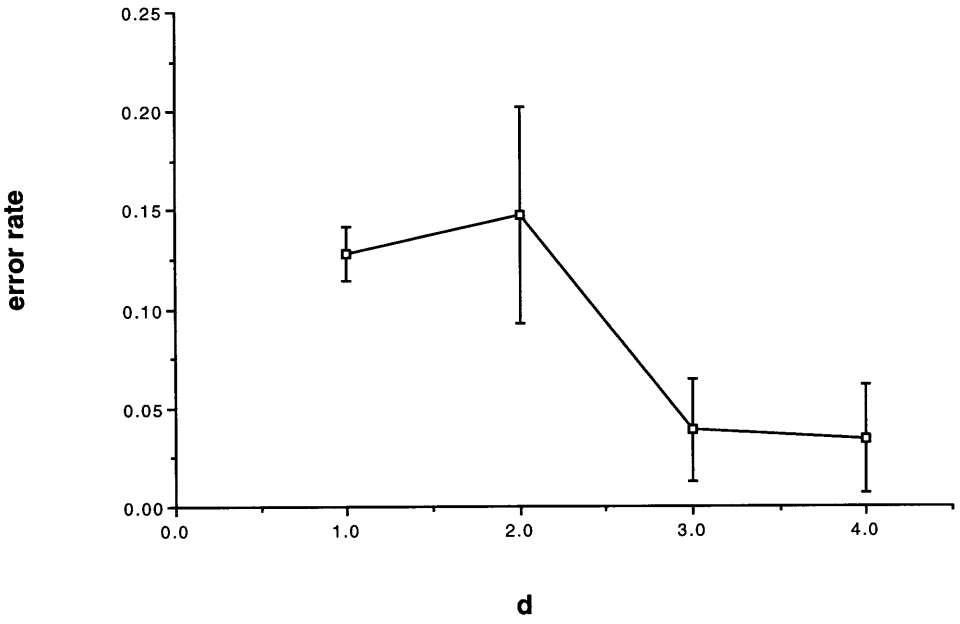


Figure 5-11: Recognition results of multiple model/target example averaged across all targets, plotted for varying amounts of initial training.

Chapter 6

Conclusion

Perceptual Computer Interfaces, systems which can directly sense and respond to a user's pose and/or expression, are an interesting new domain for machine vision and learning. We described an interactive interface environment based on the "magic mirror" paradigm, in which the user interacts with a large screen video display. Simple real-time vision routines can be applied in this domain to provide an interaction between people and virtual agents or objects. The gross 3D position of the user, the location of hands and head, and coarse information about overall pose can be recovered using classical image processing techniques: figure-ground extraction, connected components, and context-based search. This system allowed a user to interact and navigate in a virtual world using familiar and intuitive means, through the use of passive sensing with no explicit markers.

To be effective, a perceptive interface needs to be responsive both to the overall pose and position of the user, and to detailed gestures he or she may perform. We described methods for learning, tracking, and recognizing complex gestures defined as characteristic spatio-temporal patterns. Our use of an appearance-based representation allowed us to model (and search) only the portion of an object's appearance-space which is actually used in the gestures to be analyzed. The appearance-based approach also allowed analysis without

having to recover exactly the underlying object pose parameters. Using task-dependent interpolation based on the Radial Basis Function method, we were able to achieve fast and robust analysis and synthesis of facial expressions, and accurately recognize hand gestures, using only conventional video camera images as input.

However, this gesture recognition system requires high-resolution images of hands and faces to be used in unconstrained environments. To obtain these, we used an active pan-tilt camera equipped with a narrow field-of-view lens. The addition of active sensing in a gesture recognition system makes deciding what object or feature to foveate based on the current task and visual input, i.e., how to perform visual attention, a key issue. To solve this problem, we have developed a principled mechanism to track and foveate salient body parts in an active gesture recognition task where foveation is guided by recognition performance. Our attention system is based on the Partially Observable Markov Decision Process formalism, using an action set comprised of foveation actions as well as a special action signifying recognition. We use instance-based hidden state reinforcement learning to learn a policy which models both when to execute the recognition action and what foveation commands to execute to properly discriminate the target gesture.

In instance-based reinforcement learning, it is important to accurately pool experience when estimating the utility of a new action. To do this, we implement a variable-K nearest neighbor algorithm which includes all prior experience with a given action chain. Additionally, we found that multiple targets create undesired complexity in a scalar utility space due to multiple sources of positive reward. This problem was solved by the derivation and application of a multiple-model Q-learning paradigm with vector-valued Q and reward functions. With these algorithms our system can actively recognize targets from a set of gestures in real-time.

We have implemented a perceptual computer interface using this attention system which can track a person as they walk freely about a room, respond to the overall coarse state of the user, and selectively attend to fine-grained state of the users face or hand. This allows

agents or computer programs which use the interface to condition their computation based directly on the users interest, as expressed through body pose and facial expression. These interfaces have great potential to be useful in allowing natural and intuitive modes of human-to-computer communication when controlling machine systems, providing new forms of expression in an interface, and in conveying a rich description of the user in person-person communication mediated by computer systems, such as telepresence or teleconferencing.

Bibliography

[1] Aloimonos, Y., I. Weiss, and A. Bandopadhay, Active Vision, Intl. Journal of Computer Vision, 1(3):333-356, 1987.

[2] Aloimonos, Y., ed., Active Perception, Lawrence Erlbaum Associates, 1993.

[3] Badler N.I., Phillips C.B. and Webber B.L., Simulating Humans, Computer Graphics Animation and Control, Oxford University Press, 1993

[4] Bajcsy, R., Active Perception, Proceedings of the IEEE, 76(8):996-1005, August 1988

[5] Ballard, D., Reference Frames for Animate Vision, in Proc. 11th IJCAI, pp. 1635-1641, August 1989.

[6] Ballard, D., and Brown, C., (1982) Computer Vision, Prentice-Hall, Englewood

[7] Barto, A. G., Bradtke, S. J., and Singh, S. P., Real-time learning and control using asynchronous dynamic programming. Computer Science Technical Report 91-57, University of Massachusetts, August 1991.

[8] Bates J., Altucher J., Hauptman A., Kantrowitz M., Loyall A.B., Murakami K., Olbrich P., Popovic Z., Reilly W.S., Sengers P., Welch W., Weyhrauch P. and Witkin A., Edge of Intention, SIGGRAPH-93 Visual Proceedings, Machine Culture, ACM SIGGRAPH, pp. 113-114, 1993.

[9] Bellman, R. E., Dynamic Programming. Princeton, NJ: Princeton Univ. Press, 1957.

[10] Beymer, D.J., Face recognition under varying pose, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 756-761, Seattle, WA, 1994.

[11] Bichsel, M., and Pentland, A., Human Face Recognition and the Face Image Set's Topology, CVGIP: Image Understanding, vol. 59, no. 2, pp. 254-261, 1994.

[12] Blake, A., Curwen, R., and Zisserman, A., A Framework for Spatiotemporal Control in the Tracking of Visual Contours, Intl. Jnl. Computer Vision, 11:2, pp. 127-146, 1993.

[13] Blumberg B., Action-Selection in Hamsterdam: Lessons from Ethology, Proceedings of the Third International Conference on the Simulation of Adaptive Behavior, MIT Press, Brighton, August 1994.

[14] Blumberg, B and Tinsley Galyean. Multi-level Direction of Autonomous Creatures for Real-Time Virtual Environ ments. Proceedings of SIGGRAPH 95 , 1995.

[15] Bobick, A., and Bolles, R., Representation space: an approach to the integration of visual information, Proc IEEE Conf. Computer Vision and Pattern Recog., pp. 492-499, 1989.

[16] Breuel, T., View-based Recognition, IAPR Workshop on Machine Vision Applications, Tokyo, Japan, 1992.

[17] Broadwell, P, Myers, R., and Schaufler, R., (1985) Plasm: A Fish Sample, Siggraph 85 Art Show, Siggraph Visual Proceedings, ACM Press. Also available as gold, IndyZone2 CDROM, Summer 1994, Silicon Graphics Inc.

[18] Burden R. and Faires J.,(1989) Numerical Analysis, PWS-Kent Publishing Co. Boston.

[19] Burl, M.C., et al., Automating the Hunt for Volcanos on Venus, Proc. IEEE Conf. on Computer Vision & Pattern Recognition, Seattle, WA, June 1994.

[20] Casey, M., Gardner, W., and Basu, S., Vision Steered Beam-forming and Transaural Rendering for the Artificial Life Interactive Video Environment (ALIVE) . Proc. Audio Eng. Soc. Convention 1995.

[21] Cassandra, A., Kaelbling, L. P., and Littman, M., Acting optimally in partially observable stochastic domains. In Proc. AAAI-94, pages 1023–1028. Morgan Kaufmann, 1994.

[22] Lonnie Chrisman Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In Proc. AAAI-92, pages 183–188. Morgan Kaufmann, Los Altos, California, 1992.

[23] Cipolla, R., Okamotot, Y., and Kuno, Y., Qualitative visual interpretation of 3D hand gestures using motion parallax, IAPR Workshop on Machine Vision Applications, Tokyo, Japan, 1992.

[24] Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Kenyon, R., and Hart, J.C., The CAVE, Audio Visual Experience Automatic Virtual Environment, Communications of the ACM, June 1992, pp. 64-72, 1992.

[25] Darrell, T., and Pentland, A., Space-Time Gestures. Proceedings IEEE CVPR-93, New York, IEEE Comp. Soc. Press, 1993.

[26] Darrell, T., and Pentland, A. P.,. Classification of Hand Gestures using a View-Based Distributed Representation In Advances in Neural Information Processing Systems 6, Morgan Kauffman, 1994.

[27] Darrell, T., and Pentland, A., Attention-driven Expression and Gesture Analysis in an Interactive Environment, in Proc. Intl. Workshop on Automatic Face and Gesture Recognition (IWAFGR '95), Zurich, Switzerland, 1995.

[28] Darrell, T., Essa, I., and Pentland, A., Correlation and Interpolation Networks for Real-Time Expression Analysis/Synthesis, Advances in Neural Information Processing Systems-7, Tesauro, Touretzky, and Leen, eds., MIT Press. Conference, 1994.

[29] Darrell, T., Moghaddam, B., and Pentland, A., Active Face Tracking and Pose Estimation in an Interactive Room, to appear Proc. CVPR-96. 1996.

[30] Darrell, T., Maes, P., Blumberg, B., Pentland, A. P., A Novel Environment for Situated Vision and Behavior, Proc. IEEE Workshop for Visual Behaviors, IEEE Comp. Soc. Press, Los Alamitos, CA, 1994

[31] Darrell, T., Blumberg, B., Maes, P., and Pentland, A., ALIVE: dreams and illusions, Visual Proceedings of SIGGRAPH 95, ACM Press, 1995.

[32] Deering. M., High Resolution Virtual Reality. Computer Graphics, Vol. 26, 2, July 1992, pp. 195-201, 1992.

[33] Essa, I., Analysis, Interpretation, and Synthesis of Facial Expressions, PhD thesis, Massachusetts Institute of Technology, MIT Media Laboratory, Cambridge, MA 02139, USA, 1994.

[34] Essa, I., and Pentland, A. P., A vision system for observing and extracting facial action parameters, In Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1994.

[35] Essa, I., and Pentland, A., Facial Expression Recognition using a Dynamic Model and Motion Energy, In Proc. Fifth Intl. Conf. Computer Vision, IEEE Computer Society, pp. 76-83, 1995.

[36] Featherstone R., (1987) Robot Dynamics Algorithms, Kluwer Academic Publishers, Boston.

[37] Fiala, J. C., Lumina, R., Roberts, K., and Waverling, A. J., TRICLOPS: A Tool for Studying Active Vision, Intl. Jnl. Computer Vision, 12:2, pp. 231-250, 1994.

[38] Fisher S.S., Girard M. and Amkraut S., Menagerie, Tomorrow's Realities, SIGGRAPH-93 Visual Proceedings, ACM SIGGRAPH 1993, pp. 212-213, 1993.

[39] Fukumoto, M., Mase, K., and Suenaga, Y., Real-Time Detection of Pointing Actions for a Glove-Free Interface, IAPR Workshop on Machine Vision Applications. Tokyo, Japan, 1992.

[40] Gelb, A., ed. Applied Optimal Estimation, MIT Press, 1974.

[41] Horn, B.K.P., Robot Vision, M.I.T. Press, Cambridge, MA, 1991.

[42] Hutchins E. L., Hollan J. D. and Norman D. A., Direct Manipulation Interfaces, (1988) in: User-centered system design: new perspectives on human-computer interaction, D. A. Norman and S.W. Draper (eds), Lawrence Erlbaum Associates, pp 87-124.

[43] Ishibuchi, K., Takemura, H., and Kishino, F., Real-Time Hand Shape Recognition using Pipe-line Image Processor, IEEE Workshop on Robot and Human Communication, pp. 111-116, 1992.

[44] Jaakkola, T., Singh, S., and Jordan, M., Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. In Advances In Neural Information Processing Systems 7, MIT Press, 1995.

[45] Krueger M.W., Artificial Reality II, Addison Wesley, 1990.

[46] Lin, L., and Michell, T., Reinforcement learning with hidden states. In Proc. AAAI-92. Morgan Kaufmann, 1992.

[47] Loeve, M.M., Probability Theory, Van Nostrand, Princeton, 1955.

[48] Lovejoy, W., A survey of algorithmic methods of partially observed markov decision processes. Annals of Operation Reserach, 28:47–66, 1991.

[49] Maes P. Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, Bradford Books/MIT Press, 1991.

[50] Maes P., Agents that Reduce Work and Information Overload, Communications of the ACM, July , 1994.

[51] Mase, K., Recognition of facial expressions for optical flow, IEICE Transactions, Special Issue on Computer Vision and its Applications, E 74(10), 1991.

[52] McCallum., R. A., Overcoming incompleat perception with utile distinction memory. In Proceedings Tenth Machine Learning Conference. Morgan Kaufmann, 1993.

[53] McCallum, R. A., Instance-based State Identification for Reinforcement Learning. In Advances In Neural Information Processing Systems 7, MIT Press, 1995.

[54] Girard, Michael and A. A. Maciejewski. Computational Modeling for the Computer Animation of Legged Figures. Pro ceedings of SIGGRAPH 85 (San Francisco, CA, July 22-26, 1985). In Computer Graphics 19, 263-270.

[55] McKenna, Michael and David Zeltzer. Dynamic Simulation of Autonomous Legged Locomotion. Proceedings of SIGGRAP H 90 (Dallas, TX, August 6-10, 1990). In Computer Graphics 24, 4 (August 1990), pp.29-38.

[56] Moghaddam, B., and Pentland, A., Probabilistic Visual Learning for Object Detection, Proceedings of the International Conference on Computer Vision 1995 (ICCV'95), Cambridge, MA, June 1995.

[57] Mountford S.J. and Gaver W.W., Talking and listening to computers, In: The art of human-computer interface design, Brenda Laurel (ed), Addison Wesley, 1990.

[58] Murase, H., and Nayar, S.K., Visual Learning and Recognition of 3D Objects from Appearance, Int'l Journal of Computer Vision, vol. 14, no. 1, 1995.

[59] Renault, O., Thalmann, N., Thalmann, D., A vision-based approach to behavioral animation. The Journal o f Visualization and Computer Animation 1(1),1990, pp.18-21.

[60] Pentland, A., and Liu, A., Towards Augmented Control Systems, *Proc. Intelligent Vehicles '95*, Detroit, MI, 1995.

[61] Pentland, A., Moghaddam, B., and Starner, T., View-based and modular eigenspaces for face recognition. In Computer Vision and Pattern Recognition Conference, pages 84–91. IEEE Computer Society, 1994.

[62] Poggio, T., and Girosi, F., A theory of networks for approximation and learning, MIT AI Lab TR-1140, 1989.

[63] Poggio, T., and Edelman, S., A Network that Learns to Recognize Three Dimensional Objects, *Nature*, Vol. 343, No. 6255, pp. 263-266, 1990.

[64] Rabiner, L. R., and Juang, B., H., An Introduction to Hidden Markov Models, IEEE ASSP Magazine, Janurary 1986.

[65] Raibert M. and Hodgins J., Animation of Dynamic Legged Locomotion, Computer Graphics: Proceedings of SIGGRAPH '91, 25(4), ACM Press, July, 1991.

[66] Reynolds C.W., (1987) Flocks, Herds and Schools: A Distributed Behavioral Model, Computer Graphics: Proceedings of SIGGRAPH '87, 21(4), ACM Press, July 1987.

[67] Rheingold, H., Virtual Reality, Simon and Schuster, 1991.

[68] Rimey, R., and Brown, C., Controlling Eye Movements with Hidden Markov Models, Intl. Jnl. Computer Vision, 7:1, pp. 47-66, 1991.

[69] Rimey, R., and Brown, C., Control of Selective Perception Using Bayes Nets and Decision Theory, Intl. Jnl. Computer Vision, 12:2, pp. 173-208, 1994.

[70] Sakoe, H., and Chiba, S., Dynamic Programming optimization for spoken word recognition, *IEEE Trans. ASSP*, Vol. 26, pp. 623-625, 1980.

[71] Singh, S., Learning to Solve Markovian Decision Processes. PhD thesis, University of Massachusetts, February 1994.

[72] Sondik, E. J. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. Operations Reserach, 26(2):282–304, 1978.

[73] Sutton, R. S., Learning to predict by the method of temporal differences. Machine Learning, 3:9–44, 1988.

[74] Terzopoulus, D., and Waters, K., Analysis and synthesis of facial image sequences using physical and anatomical models, IEEE Trans. PAMI, 15(6):569–579, 1993.

[75] Terzopoulus, D., and Waters, K., Analysis and Synthesis of Facial Image Sequences Using Physical and Anatomical Models, IEEE Trans. Pattern Analysis and Machine Intelligence, 15(6), pp. 569-579, June 1993

[76] Thalmann, N., and Thalmann D.,, Artificial Life and Virtual Reality. John Wiley & Sons, Chichester U.K, 1994.

[77] Tu, X., and Terzopoulos, D., Artificial Fishes: Physics, Locomotion, Perception, Behavior Computer Graphics: Proceedings of SIGGRAPH 1994, ACM Press, July 1994.

[78] Turk, M., and Pentland, A. P., Eigenfaces for Recognition, Journal of Cognitive Neuroscience, vol. 3, pp. 71-89, 1991.

[79] Ullman, S., and Basri, R., Recognition by Linear Combinations of Models, *IEEE PAMI*, Vol. 13, No. 10, pp. 992-1007, 1991.

[80] Vincent V.J., Mandala: Virtual Village, SIGGRAPH-93 Visual Proceedings, Tomorrow's Realities, ACM SIGGRAPH 1993, pp. 207, 1993.

[81] Waters K., and Terzopoulos, D., Modeling and animating faces using scanned data, The Journal of Visualization and Computer Animation, 2:123–128, 1991.

[82] Weng, J.J., On Comprehensive Visual Learning, Proc. NSF/ARPA Workshop on Performance vs. Methodology in Computer Vision, Seattle, WA, June 1994.

[83] Watkins, C., and Dayan, P., Q-learning. Machine Learning, 8:279–292, 1992.

[84] White, C., A survey of solution techniques for the Partially Observed Markov Decision Process Annals of Operation Research, 32:215–230, 1991.

[85] Whitehead, S., Active perception and reinforcement learning. In Proc. 7th Intl. Conf. ML, June 1990.

[86] Wilhelms J., R. Skinner. A 'Notion' for Interactive Behavioral Animation Control. IEEE Computer Graphics and Applications 10(3) May 1990, pp. 14-22.

[87] Williams, L., Performance-driven facial animation, Computer Graphics: Proceedings of SIGGRAPH 1990, 24(4), ACM Press, pp. 235-242.

[88] Wren, C., Darrell, T., Starner, T., Johnston, M., Russell, K., Azarbayejani, A., and Pentland, A. pfinder: A Real-Time System for Tracking People, SPIE Conference on Real-Time Vision, M. Bove, Ed., Philadelpia, PA, July 1995.

[89] Wren, C., Sparacino, F., et al., Perceptive Spaces for Performance and Entertainment: Untethered Interaction using Computer Vision and Audition, available as MIT Media Lab Vismod TR-372.

[90] Yacoob, Y., and Davis, L., Computing spatio-temporal representations of human faces, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 70-75, Seattle, WA, 1994.

[91] Zeltzer D., Task-level graphical simulation: abstraction, representation and control, in: N.I. Badler, B.A. Barsky and D. Zeltser (editors), Making them move: mechanics, control and animation of articulated figures, Morgan Kauffman, pp. 3-33, 1991