Decision Theoretic Learning of Human Facial Displays and Gestures

by

Jesse Hoey

B.Sc. Physics, McGill University, 1992

M.Sc. Physics, University of British Columbia, 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

December 2003

Abstract

We present a vision-based, adaptive, decision-theoretic model of human facial displays and gestures in interaction. Changes in the human face occur due to many factors, including communication, emotion, speech, and physiology. Most systems for facial expression analysis attempt to *recognize* one or more of these factors, resulting in a machine whose inputs are video sequences or static images, and whose outputs are, for example, basic emotion categories. Our approach is fundamentally different. We make no prior commitment to some particular recognition task. Instead, we consider that the *meaning* of a facial display for an observer is contained in its relationship to actions and outcomes. Agents must distinguish facial displays according to their *affordances*, or how they help an agent to maximize utility. To this end, our system learns relationships between the movements of a person's face, the context in which they are acting, and a utility function. The model is a partially observable Markov decision process, or POMDP. The video observations are integrated into the POMDP using a dynamic Bayesian network, which creates spatial and temoral abstractions amenable to decision making at the high level. The parameters of the model are learned from training data using an *a-posteriori* constrained optimization technique based on the expectation-maximization algorithm. The training does not require labeled data, since we do not train classifiers for individual facial actions, and then integrate them into the model. Rather, the learning process *discovers* clusters of facial motions and their relationship to the context automatically. As such, it can be applied to any situation in which non-verbal gestures are purposefully used in a task. We present an experimental paradigm in which we record two humans playing a collaborative game, or a single human playing against an automated agent, and learn the human behaviors. We use the resulting model to predict human actions. We show results on three simple games.

Contents

List of Tables

List of Figures

Acknowledgements

Elite Member	Board Member	Associate Board Member	Financial Donor
Catherine Dat	Jim Little	Craig Boutilier	NSERC
Catherine Hoey	David Lowe	Alan Hu	UBC
John Hoey	David Poole	Ron Rensink	Precarn
Dylan Hoey	Cristina Conati	Enrique Sucar-Sucar	Nissan
		Sid Fels	
		Michiel van de Panne	
		Stan Sclaroff	
		Nando de Freitas	
		Jack Snoeyink	

Preferred Member	Star Contributor	Diamond Contributor
Don Murray	Matthew Wilkins	Valerie McRae
Pantelis Elinas	Bruce Dow	Joyce Poon
Nicole Arksey	Luc Dierckx	Giuliana Villegas
Pascal Poupart	Dave Brent	Sunnie Khuman
Giuseppe Carenini	Mike Sanderson	Monica Glaboff
Robert St-Aubin	Sean Godel	
Andrea Bunt	Chris Majewski	
Kenji Okumak		
Peter Carbonetto		
Kasia Muldner		
Jochen Lang		

Jesse Hoey

The University of British Columbia
December 2003

Chapter 1

Introduction

This thesis describes a model of human gestures and facial expressions that unifies computer vision, uncertain reasoning, and decision theory. The motivation is that, since humans use non-verbal signals in communication, computational agents that interact with humans will need capabilities for learning, recognising and using the extensive panoply of human non-verbal communication skills. The verb *use* is important: non-verbal signals are *useful*. For example, they may predict the future actions of the signaler, or they may request actions from the perceiver. The perceiver of a non-verbal signal must not only *recognise* the signal, but must *understand* what it is useful for. The signal's usefulness will be defined by its relationship to the joint space of both signaler and receiver, their joint actions, their possible futures together, and their individual ways of assigning value to these futures. The model we present in this thesis aims to unify the computer vision aspects of automatically perceiving human non-verbal behaviors with the decision-theoretic aspects of putting those perceptions to use. The ability to explicitly reason about uncertainty plays an important role in this unification. If an agent is to take decisions based upon noisy visual data, then the agent must explicitly model its own uncertainty about its perceptions. Bayesian networks are the ideal tool for modeling uncertainty in video measurements and in decision theoretic models, providing a solid theoretical basis for these models to co-exist. All the models presented in this thesis are Bayesian networks (with utility).

We claim that it is important not to separate computer vision from decision theory when modeling human behavior. It is not sufficient to build computer vision sensors that deliver information to decision-theoretic reasoning modules, which then decide upon actions. Recognition and action are too tightly coupled for this kind of one-way communication. Decision making is difficult, and is best dealt with at a high level of abstraction. Humans seem capable of acting in the world based upon abstract representations of their sensory inputs. If these abstract representations are

sufficient for decision making, then there cannot be *more* useful information in the stimuli than there is in the abstractions. Therefore, an efficient perceptual system (which is presumably what humans have evolved to have) will be able to adapt its low-level representation system to only pick up those parts of the signal that are sufficient for building the high-level abstractions. It is important for low-level vision components to not only send, but also receive information from high-level decision-making components. We believe that a unified model of the two aspects is the most effective method for approaching this problem.

This thesis presents a vision-based, adaptive, Bayesian model of human facial displays and gestures. The model is, in fact, a partially observable Markov decision process, or POMDP, with observations over the space of video sequences of human facial displays and gestures. The POMDP model integrates the recognition of the non-verbal signals with their interpretation and use in a utility-maximization framework. To ease the burden on decision-making, the model builds temporal and spatial abstractions of input video data. The model can be acquired from data, and can be used for decision making based, in part, on the non-verbal behavior of a human through observation. However, optimal decision making in the face of large output spaces is still an open problem, which this thesis does not attempt to solve. Instead, we apply some simple approximate solution techniques to compute policies of action based on non-verbal displays. These approximate policies work fairly well in the examples we present, but would be insufficient in more complex situations. Finding better approximate solutions is part of our current research.

Our work is distinguished from other work on recognising human non-verbal behavior primarily because it does not require labeling of training data for particular facial displays or gestures. We do not train classifiers for different behaviors and then base decisions upon the classifier outputs. Instead, the model training process *discovers* categories of behaviors in the data. This discovery is based, in part, upon what non-verbal displays are *important* for making value-directed decisions. The advantage of this approach is threefold. First, we do not need expert knowledge about which displays are important. Second, since the system learns categories of motions, it will adapt to novel displays without modification, and without *a-priori* specification of said displays. Third, resources can be focused on tasks which will be useful for the agent. It is wasteful to train complex classifiers for the recognition of fine motion if only simple displays are being used in the agent's context.

In constrast, psychologists have advocated the use of coding systems for the description of facial action. In particular, the Facial Action Coding System (FACS) [EW78] has become the standard for psychological inquiries into facial expression. Computer vision researchers have also adopted it as the standard to strive

towards [EP97, TKC01, BLB+03]. However, although the the recognition of facial action units may give the ability to discriminate between very subtle differences in facial motion, or "microactions", it requires extensive training and domain specific knowledge. For example, Bartlett *et al.* write that it would require approximately 250 minutes, or nearly half a million hand-coded frames of video to train their system for most facial action units [BLB+03], while their current data set contains only 17 minutes of coded video frames. Notwithstanding this hurdle, which will surely be cleared eventually, these systems are still not able to accurately distinguish action units from each other in spontaneous video. The current state of the art is only to distinguish brow lowering from random facial motion 75% of the time [BLB+03] with less than 20 training examples. Their system needs over 50 labeled training examples to achieve recognition rates over 90%. As Pentland has pointed out, the importance of such a fine level of representation is not clear for computer vision systems that intend to take actions based on observations of the human, and the action unit analysis may only be useful for the behavioral sciences [Pen00]. Finally, the same type of analysis is not applicable to gestures or body motion, as these do not have well defined standards of form [McN92]: it will be difficult to come up with a small set of "basic" action units which span the space of all possible gestures.

The next section gives a basic introduction to the human facial displays from a psychology perspective. Psychology provides guidelines which we use to design a model of facial displays and gestures for decision making in Section 1.2. Section 1.3 then gives an overview of our model. Section 1.4 describes how the model could be used to for decision making in an autonomous agent. Section 1.5 describes the experiments we have performed. Contributions of the thesis are given in Section 1.6. Finally, Section 1.7 gives an outline of the main body of the thesis.

1.1 Non-Verbal Displays

Communication plays a critical role in life. Be it a conversation, a business transaction, the writing of a letter, or a choice of dress, the diversity of the human act of communication is clearly evolutionarily dominant, allowing for collusion and collaboration [Fri94]. Although the auditory-speech language channel appears to carry the most weight, communication occurs in multiple channels, adding the generation and visual interpretation of gesture and facial expressions to speech and hearing. Gestures and other actions complement speech, adding information and increasing effect. Aristotle, in *Rhetoric*, points out ([Ari], Book II, Chapter 8):

> ... those who heighten the effect of their words with suitable gestures, tones, dress, and dramatic action generally, are especially successful ...

Falling under the umbrella of non-verbal communication, or gesture, the human face, hands and body all perform communicative acts, which have been extensively practiced [dJ32], and studied [Dar72, ER97, Fri94, McN92]. The creation and interpretation of signs has clear evolutionary advantages [Fri94], and these abilities are deeply rooted in the history of the human species, far before the evolution of spoken language. The perception and use of signs, and in particular facial expressions, develops in human children at a very young age. Three month old babies can generate and recognize smiles [RFD97a]. The antecedence of facial expression learning to spoken language acquisition in individual humans seems to mirror the evolutionary development of non-verbal to verbal signals in the human species.

The advantages of non-verbal communication are rapidity and ease of interpretation. Displays of aggression in males, for example, do not require higher intellectual processing, and can be accomplished more effectively with simple gestures or sounds. A more complex example is the initiation of conversation between two humans, a joint action which requires a great deal of synchrony. In fact, timing in conversations in general is critical, yet seems effortless for adult humans. Attempting to negotiate such synchrony using spoken utterances would be costly and slow. Instead, the face, gaze and body posture are used as rapid and effective timing signals. For example, the raising of hands by a listener is a signal that the listener wants to say something, and so is asking the speaker to give her the floor [Cas00]. A speaker who lowers their eyes during a pause in a conversation may be indicating that they still wish to hold the floor, and need only a moment to figure out what to say next [Cho91]. Another well known example is the back-channel display of acknowledgment, which is used by a listener to assure the speaker that their message to date has been properly received, without causing an interruption.

The multiple channels used by humans in their communication also has evolutionary advantages, since they allow for meaning to be more richly conveyed, opening the door to more complex messages. Hand gestures in humans, for example, may complement speech in a global way, adding an overall context to an utterance which could not be simultaneously and synchronously achieved through language alone [McN92]. Hand gestures may also be used to abstract concepts, such as imperatives. For example, the spoken instruction to *"put it down on the table beside the record player"* can be summarized as *"put it there"*, if accompanied by a pointing gesture to the table beside the record player. The increased efficiency this allows is typically cited as one of the primary evolutionary advantages of multimodal communication. Many animals use multiple modalities for communication. A defensive feline does not simply hiss, but raises her hair and tail, and arches her back. In fact, even spiders use vibration, vision and chemical senses for multimodal

communication [UR02].

There has been a growing body of work in the past decade on the communicative function of the face [Fri94, RFD97b] and of the hands [McN92]. This psychological research has drawn three major conclusions. First, non-verbal gestures are often purposeful communicative signals [Fri94]. Second, the purpose is not defined by the gesture alone, but is dependent on both the gesture and the context in which the gesture was emitted [RFD97b, Cho91]. Third, the signals are not universal, but vary between individuals in their physical appearance, their contextual relationships, and their purpose [RFD97b]. We believe that these three considerations should be used as critical constraints in the design of computational communicative agents able to learn, recognise, and use human facial signals.

1.2 Model Design

Psychologists have found that gestures and facial displays are context dependent, are often used for a purpose, and are individually variable. Therefore, the design of a communicative agent that uses non-verbal displays must include the following constraints:

1. Context dependence implies that the agent must model the relationships between the displays and the context in which they are shown.

2. If the agent is to act rationally, then it must be able to compute the utility of taking actions in situations involving purposeful non-verbal displays. It must understand the relationships between the displays, the context, and its own utility function.

3. The signals are individual and context dependent, and so the agent needs to adapt to new interactants and new situations, by learning new relationships between non-verbal signals and other context.

These constraints can be integrated in a decision-theoretic, vision-based model of human non-verbal signals. The model can be used to predict human behavior, or to choose actions which maximize expected utility. We present this model using the example interaction shown in Figure 1.1. Charlie (on the left) is trying to decide upon an action to take. He is considering stealing a cake from the cook's tray, but is watching the cook's behavior. He is trying to optimise over the outcomes that will occur as a result of his actions. However, he is not in complete control over the outcomes, as they also depend on the cook's actions (on the left). Nevertheless, he can observe the non-verbal actions of the cook, and use these to

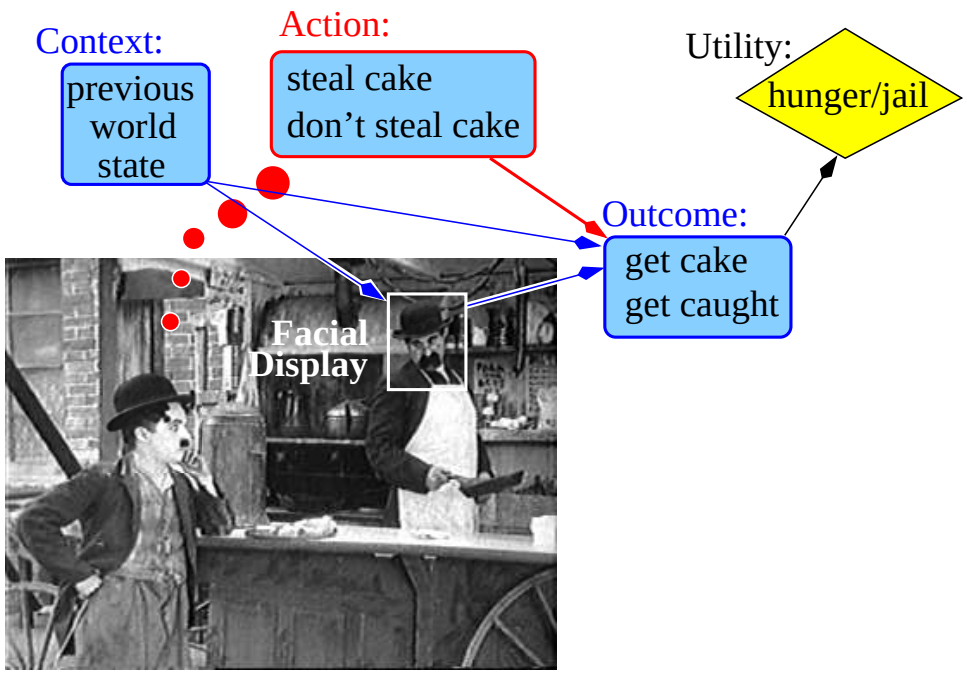


Figure 1.1: Interaction between two humans as a partially observable Markov decision process (POMDP).

predict what the cook's actions will be, and what resulting outcomes are. In general, the outcomes are dependent on both person's actions, and other contextual factors, such as the plate of cakes on the counter [1]. The dependencies shown in Figure 1.1 are those of a partially observable Markov decision process, or POMDP [KLC98].

A POMDP describes the effects of an agent's actions upon its environment, the utility of states in the environment, and the relationship between observations, the actions and the states. The Markovian assumption is that the agent's history is contained in its current state. A POMDP model allows an agent to predict the long term effects of its actions upon his environment, and to choose valuable actions based on these predictions. The POMDP framework addresses the design constraints above in the following ways.

1. The observations of the POMDP are the displays of a human partner in a video stream. The state of the POMDP is the context in which the displays are emitted, and thus, the POMDP models the relationship between displays and context. As psychologists tell us, the displays are only meaningful when analysed in context. Their meaning, then, *is* this relationship between displays and context modeled by the POMDP.

2. A utility function is part of a POMDP, and allows the agent to make decisions leading to valuable states based, in part, upon the "meaning" of displays as contained in the model.

3. POMDPs can be learned from data, allowing for adaptivity. Our work is novel in this regard in that it does not use any prior knowledge about the kinds of facial displays that are expected, but rather builds classes of facial displays from unlabeled data. The models we learn discover the ways in which the learned classes of non-verbal signals are related to the expected utility of the agent's actions. This type of learned model opens the door to the ability to only model those aspects of the signal which are *important* to the agent. It is not necessary to learn models of all possible facial displays, but only those the recognition of which is useful for achieving value.

Rational agents must rapidly make value-directed decisions about actions based on sensory information, and so must have some system for assigning value to states of the world. Decision making is expensive, however, and raw video inputs are far too information intensive for any hope of tractability. The agent must construct abstract representations of the world over which a resource-bounded decision making process can occur. In a POMDP, these abstractions are the non-observable

[1] To see what the outcome was, visit `www.cs.ubc.ca/~jhoey/dtfaces`.

states which describe facial displays at a high level. For example, one such state may correspond with the wink of an eye, whereas another may correspond to a smile. These abstract, high-level states condition the raw video signals using a multi-level dynamic Bayesian network which incorporates both spatial and temporal abstraction. This model is summarized in the next section.

There will be little discussion of speech recognition and natural language understanding in this thesis. Our work focuses solely on recognizing and using non-verbal communicative acts. However, it is well known that gesture and facial expression are intimately tied to speech [McN92, CBC+00], and one might object to our omission. Nevertheless, research has shown that gestures take place *globally* and *synthetically*, as opposed to language, which is linear-segmented (can be broken down in to individual units) [McN92]. Further, semantic gestures do not combine to form larger gestures, but remain one to a clause. Finally, gestures and facial expressions vary from person to person [McN92, Cho91]. These findings give us good reason to keep speech and gesture recognition at a distance, unifying them at some higher level of temporal abstraction. That is, it does not make sense to try to correlate a particular phoneme with one 30ms snapshot taken during a gesture. We must instead seek to correlate some description of the entire gesture with the recognized words or concepts conveyed by the language. Our Bayesian model shows how we can classify sequences of motions in video, and how to learn correlations between those classifications and modeled data from other modalities. With the important assumption of independence at the lowest levels, we can easily incorporate modeled data from any other sensor, so long as it can be described by a set of discrete states.

1.3 Model Structure

This thesis presents a unified Bayesian model of these vision-based aspects of non-verbal display recognition and understanding. The model is a multi-level dynamic Bayesian network, the parameters of which can be learned from unlabeled training data. We consider that the context and the raw video signal are observable, but the model is "free" to come up with an internal intermediate representation to map between fast, continuous input signals to slowly changing, discrete context signals. The parameters are learned using an *a-posteriori* optimization technique based on the expectation-maximization (EM) algorithm [DNR77]. The model also includes utility, and can be used to compute optimal or approximate policies of action. In turn, these computations can guide the classification of the signals towards those which are useful for achieving value. The model performs both spatial and temporal abstraction in order to alleviate the burden on the decision process.

Figure 1.2 shows an schematic overview of the model. We will describe it here using facial displays as an example, but it can be applied equally well to gestures. At the top we see the same model shown in Figure 1.1, with outcomes depending on

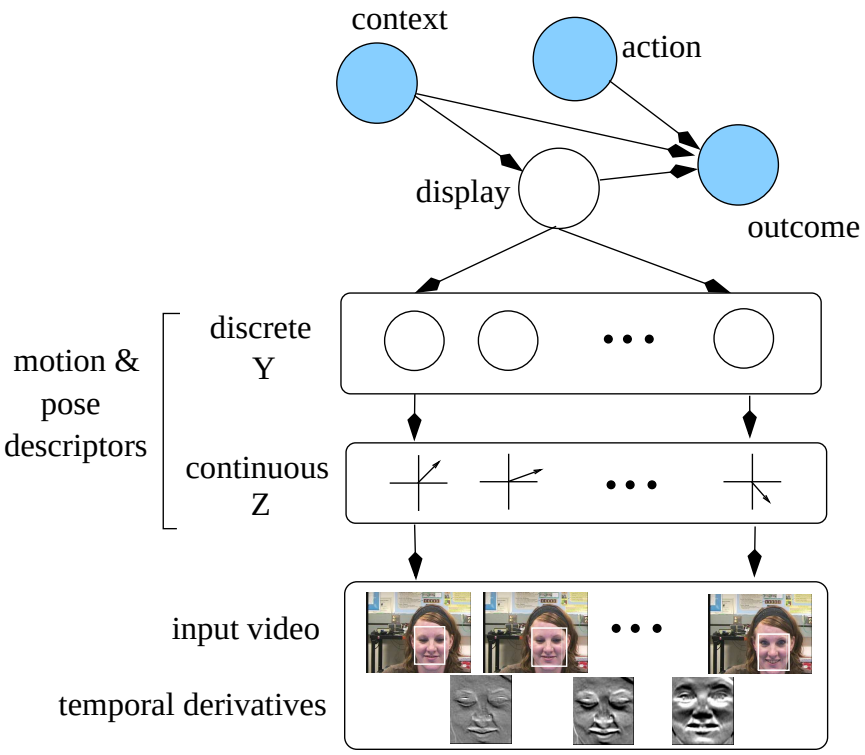


Figure 1.2: Schematic overview of POMDP and its output function for creating spatially and temporally abstract descriptions of video sequences. Actions and context are shown at the top (the utility function is not shown explicitly). The observation (in this case facial expression) is a video sequence shown at the bottom. The intermediate levels are for spatially and temporally abstracting the video.

actions, context, and high-level descriptors of non-verbal displays. At the bottom, we see the input video sequence. The goal is to learn a mapping between the video sequence and the abstract, high-level descriptor of the display. This means that we want to compute the likelihood of observing an input video sequence, given this high-level descriptor: $P(video|display)$.

We consider that analyzing human facial displays from video streams involves modeling both the current configuration or pose of the face and the way the face is currently moving, or the dynamics of the face. Dynamics and configuration complement one another, and are akin to momentum and position in classical dynamics. They can be both useful in describing what a human face does. The instantaneous

pose of the face is given by the image region, I, containing the face. The region is given by an independent tracking process. The instantaneous dynamics of the face is given by the temporal derivative over the region of interest, dI, between successive images. Thus, $video = \{\mathbf{I}, \mathbf{dI}\} = \{I_1, \ldots, I_T, dI_1, \ldots, dI_{T-1}\}$ and we seek to compute $P(\{\mathbf{I}, \mathbf{dI}\}|display)$. We use bold face symbols to denote sets of variables.

We factor the likelihood computation of $P(\{\mathbf{I}, \mathbf{dI}\}|display)$ into three terms:

$$P(\{\mathbf{I}, \mathbf{dI}\}|display) = \int_{\mathbf{Z},\mathbf{Y}} P(\mathbf{I}, \mathbf{dI}|\mathbf{Z})P(\mathbf{Z}|\mathbf{Y})P(\mathbf{Y}|display), \tag{1.1}$$

where each Z is a continuous feature vector representing both the instantaneous dynamics and pose of the face, but in a much lower dimensional space than the input videos and derivatives. The continuous vectors, $\mathbf{Z}$, are then discretized through $P(\mathbf{Z}|\mathbf{Y})$ to a set of multinomial variables, $\mathbf{Y}$, which are high-level descriptors of instantaneous pose and motion over the course of the sequence. Finally, these high-level, discrete variables are temporally abstracted to the display descriptor through the mapping $P(\mathbf{Y}|display)$. The primary contribution of this thesis is a method for analytically computing the likelihood in Equation 1.1 given a parametrization of the three distributions, and for learning the parameters of these distributions from unlabeled training data.

We give a brief description of each of the three terms in Equation 1.1. The details can be found in Chapter 3.

$P(\mathbf{I}, \mathbf{dI}|\mathbf{Z})$ The images and temporal derivatives are considered independent given the continuous feature vectors at each time step and independent of one another given their continuous descriptors. Thus,

$$P(\mathbf{I}, \mathbf{dI}|\mathbf{Z}) = \prod_t P(I_t|Z_{I,t})P(dI|Z_{dI,t}).$$

The likelihood functions, $P(I_t|Z_{I,t})$ and $P(dI|Z_{dI,t})$, are computed using a projection to a set of *a-priori* basis functions. Using pre-determined basis functions defers any commitment to particular types of motion to higher levels of processing, without affecting computational efficiency. We use the complete and orthogonal basis of Zernike polynomials [vZ34], which have useful properties for modeling flow fields [HL00], and images [TC88]. The images are projected directly to the Zernike basis functions. Projections of temporal derivatives are slightly more complex, since they do not characterise motion directly. Instead, the projection is done through the intermediary of the (unknown) optical flow field, which is probabilistically related to the derivative. One of the main contributions of this thesis is a analytical method for evaluating the probability distribution over this projection. This is important, as

the flow fields include noise which is data dependent: it varies spatially across the image according to the local structure. The method we have presented describes how to account for this noise when computing the likelihoods of spatial derivatives given high-level classes [HL03].

Since we do not know which of these basis functions will be useful for any particular task, we use a feature weighting technique that works by placing priors on the cluster means, such that basis functions which discriminate between classes are weighted more highly.

$P(\mathbf{Z}|\mathbf{Y})$ Again, we assume the continuous features are independent given their discrete counterparts, such that

$$P(\mathbf{Z}|\mathbf{Y}) = \prod_t P(Z_{I,t}|W_t)P(Z_{dI,t}|X_t),$$

where $Y_t = \{X_t, W_t\}$. The likelihoods of continuous features given discrete descriptors are modeled with mixtures of Gaussian distributions, such that $P(Z_{I,t}|W_t) = \mathcal{N}(Z_{I,t}; \mu_W, \Lambda_W)$ and $P(Z_{dI,t}|X_t) = \mathcal{N}(Z_{dI,t}; \mu_X, \Lambda_X)$where μ_W and Λ_W (μ_X and Λ_X) are the mean and covariance, respectively, of the Gaussian distribution for a particular value of W (X). States of X and W correspond to classes of instantaneous motion and pose, respectively. For example, during a smiling display, classes of X may correspond to smile expansion or smile relaxation motions, while classes of W may correspond to smiling or neutral poses.

$P(\mathbf{Y}|display)$ This likelihood involves temporal abstraction. Recall that $\mathbf{Y}$ is a sequence of discrete motion and pose descriptors, which independently characterize the *instantaneous* dynamics and configuration in the input video. When used to describe facial motion, they may represent a smiling facial pose, or motions observed during contraction of the eyebrows. On the other hand, *display* is a high-level display descriptor. For example one *display* may describe sequences in which a person's face expands into a smile. Another may describe the contraction of the face into a frown, and the subsequent relaxation back to a neutral pose. The mapping from sequences of instantaneous motions and poses to high-level descriptors of facial display sequences is achieved with mixture of coupled hidden Markov models (CHMM). A CHMM is a dynamic Bayesian network which couples two Markovian processes over the high level descriptors of dynamics and configuration, X and W. The CHMM models the temporal progression of both motion and pose descriptors, as well as the interaction between them. The similarity between the two output distribution

models over dynamics and configuration makes this model elegant and simple to describe, yet powerful in its ability to model facial displays. Each *display* state has a separate CHMM, combined in a mixture model. The distribution over high-level *display*s can be computed using belief propagation [Pea88] in each CHMM.

Chapter 3 describes how to compute the likelihood in Equation 1.1. Chapter 4 shows how the parameters of the model, including the means and covariances of the Gaussian output distributions, the feature weights, and the parameters of the mixture of CHMMs, can learned using a Bayesian *a-posteriori* optimization based on the expectation-maximization algorithm [DNR77]. The use of priors allows learning of the model even in the face of little training data, or online in a reinforcement learning situation.

1.4 Using the Model

It would be foolhardy to attempt to learn models of full-blown human interactions directly. Much work in understanding the learning process must be done before we can hope to reach this lofty goal. Instead, we develop an experimental paradigm which allows us to focus only on a subset of displays in a restricted context. We record human subjects playing cooperative, interactive games which require the use of non-verbal communication. The games are designed such that the communication is critical for winning in the game. Cooperative games are used for two reasons. First, we believe that most simple human interaction is cooperative. This is what makes a species survive! Second, cooperative games are simpler to model because we can assume that each player is working towards the same goal. Thus, a player can do sufficiently well without explicitly modeling his partner's utility function.

The video data taken during play of a particular game is used to learn a decision-theoretic model that can be used to compute strategies of action for a rational agent. There are many ways of approaching the task of finding optimal action plans for multi-agent games. A game-theoretic solution concept involves examining the models and utility functions of all players, and attempting to understand the behavior of all the players in the game simultaneously, assuming that they are all rational agents. However, these types of solutions are used to understand the way in which humans will interact together. We would like our methods to be useful for an artificial agent playing a game with a human. In this case, it is not possible, in general, to know the utility functions of each player in the game, and we must instead adopt the *decision-analytic* approach to games [Mye91]. This approach considers the game from the perspective of individual agents. Each agent's optimal strategy

is to maximize its expected payoff with respect to its beliefs about the strategies of all the other agents. There are problems with this approach. The possible strategies of all other agents will inevitably include their assessments of the possible strategies of all other agents as well. Therefore, to figure out how to behave rationally, an agent would have to know how another rational agent would expect him to behave, which is the problem he started out with. This infinite regress makes it difficult to justify decision-analytic solution concepts from a game theoretic standpoint.

However, a decision-analytic agent can attempt to explicitly model this infinitely nested system of beliefs, as described in [Gmy02]. The idea is that the agent can perform sufficiently well in a game by "cutting off" the infinite nesting at some point. For example, agent A can explicitly model the beliefs of agent B, but not the beliefs that agent B has about agent A's beliefs. This type of analysis may be particularly relevant in cooperative games, in which agents may assume that all collaborators share roughly the same utility function. The games we consider in this thesis are simple enough that these types of considerations do not play a major role. Further work into the use of decision-analytic models for multi-agent systems is warranted for more complex situations.

1.5 Experiments and Results

We present experimental results on data sets taken during play of three games: the imitation game, a robot control "game", and the card matching game. The first (imitation game) only involves facial displays as actions, and so does not have a reward function. This game is used to explore the representational power of our computer vision modeling techniques. The second (robot control) involves a single human performing gestures for robot control. The robot control experiments are intended as a simple demonstration of our system working with something *other than* facial expression, and of our value-directed structure learning techniques. The experimental setup for the robot control experiments is very simple, and we do not claim a gesture recognition system that would deal with orientation, time scaling, or robot movement and viewpoint. The third (card matching game) involves two humans playing a collaborative game. The facial displays are fairly simple, but the decision theory problem is much more complex than the other two games. This data is used to demonstrate how a policy can be computed based, in part, on non-verbal displays.

During play of the imitation game, a human tries to mimic the facial displays of an animated cartoon face. These displays involve some complex facial motions, and we use this game primarily to demonstrate the computer vision modeling tech-

niques. We perform three analyses on this data in order to gain understanding of the different components involved in the model. The first analysis looks at the spatial abstraction of instantaneous motion, and assumes temporal independence: each flow field is independent of all others. This allows understanding of the motion representation we use, and shows how clusters of similar instantaneous motions can be discovered in a data set. The second analysis does the same as the first, but for the facial configuration. The third analysis includes the temporal abstraction of both instantaneous motion and configuration, and looks at how similar sequences of motions (temporally extended facial displays) can be clustered together. We present these three separate treatments because each model performs a different function, and the first two are an integral part of the third. The simple mixture of Gaussians model assumes temporal independence, and learns models which spatially abstract each frame of the video data. The coupled hidden Markov models use these spatial abstraction models as their lowest level, and perform temporal abstraction, learning the temporal progression of the data. We show results from a leave-one-out cross-validation experiment for each of three subjects playing the imitation game, where we attempt to infer the cartoon display that produced the observed facial motion on unseen data. These results are compared to those obtained if the training data is labeled, and we find our clustering method performs well.

Robot control using gestures is the second "game". An operator signals navigation commmands to a robot using hand gestures *forwards, stop, go left* and *go right*, and rewards the robot for performing the correct action after each gesture. The robot learns the gesture categories, their number, and their relationships to its actions and utility functions. It can then use the learned model to take actions after subsequent gestural commands. We perform a leave-one-out cross-validation experiment and measure how much reward the robot would gain by taking these actions on unseen data. We find error rates of only 2% over 48 test sequences.

The card matching game is played by two humans through a computer interface. The players can see, but not hear, one another. They are allowed to communicate through this visual link, but no restrictions are placed on the type of communication. Each player has a hand of three cards, but can only see their own cards, not their partner's. Each player gets to play a single card, and they both win if the suits of their played cards match. The idea is that the players must somehow agree, using only visual communication, about which card to play. This general form of the card matching game has a similar flavor to the Krauss-Glucksberg referential test used to study the development of language [KG77]. In a Krauss-Glucksberg test, the two subjects can hear, but not see, one another, and must choose matching "nonsense" symbols from identical sets. To pass the test, they must develop a

common referential language for the symbols, which often involves one subject imitating the symbols of the other. The general card matching game is similar because there are no pre-defined standard gestures for playing cards, and the players have to develop a communication system for referring to card suits.

We present results on a simplified version of the card matching game, in which one player has the ability to make a "bid" of a card suit. The other player must then simply agree or disagree with the bid. This game simplifies the analysis because it allows the players to use a pre-defined set of gestures: head nods and head shakes. The data is used to train a decision-theoretic model, which must learn the relationship between what a player is doing, the state of the game, what the future actions will be, and how the outcome will benefit the players. The model must also learn what a player expects his partner to display, given the state of the game. The model is used to compute a policy of action using an approximation technique, and is demonstrated on a test data set.

1.6 Contributions

The major contributions of this thesis are as follows. They are ranked in order of importance, and references to our published work on these subjects are shown.

- [Hoe01, Hoe02, HL04, Hoe04] A novel and unified model of human non-verbal behavior in video streams, integrated with decision making over high level context states and actions. A Bayesian *a-posteriori* learning procedure for adapting the parameters of the model to training data. This model aims at unifying computer vision with decision theory. The goal of computer vision systems has traditionally been exactly the task that the computer vision is performing: the goal of a face recognition system is to recognise faces, while the goal of a pedestrian tracker is to track pedestrians. However, the field is ready to move on to domains in which the goal is something larger than just the computer vision task. For example, facial expression recognition can be used by an embodied conversational agent, and pedestrian trackers can be used by smart buildings. Integrating computer vision with actions and goals is an important step, not just for the larger systems that use computer vision, but also for the computer vision task itself. In fact, the task of the computer vision system is *defined* by the goals it is being used to reach. It is important, therefore, not to consider the computer vision task alone, but rather in the context of whatever it is being used for.

  Decision theory is the standard method used by rational agents to choose actions, and most planning tasks can be formulated as decision theoretic mod-

els. This thesis shows how computer vision modeling of human action can be unified with decision theoretic planning in a single, Bayesian model.

- [HL03] A novel probabilistic method for spatially abstracting optical flow fields to a set of basis functions. The usual method for constructing simple descriptions of classes of optical flow fields is to first compute the flow field from the spatio-temporal image derivatives, then to project it to some low dimensional subspace, and finally to model the distribution over projections as arising from a number of discrete categories. However, each stage of this process discards information about the noise inherent in the estimation. We have presented a Bayesian approach that directly models the distribution over spatio-temporal image derivatives given the motion classes. The method includes the application of a Bayesian feature weighting technique, obviating the need for prior selection of basis functions.

- [HL00] Use of Zernike polynomials as descriptors of optical flow. The Zernike polynomial basis functions have been used for analysis of shapes [TC88, BSA91, HSJ95, KK00]. Their advantages are efficiency of computation, and accuracy of reconstruction. Our work is the first to apply the Zernike polynomial basis for representing optical flow [HL00].

1.7 Thesis Outline

The thesis is structured as follows. Chapter 2 discusses some of the previous work on the psychology of facial movement, on the modeling of facial motion and human body motion in general using computer vision, and on the decision-theoretic modeling of human action. Chapter 3 presents a detailed description of a general class of Bayesian hierarchical models of human facial displays which conform to the assumptions made above. This chapter also describes the method we use for tracking a face in a video sequence. Chapter 4 show how the parameters of the models can be learned from data using the expectation-maximization algorithm. Some implementation details are also given in this chapter. Chapter 5 then describes how the learned models can be used to predict human actions in repeated Bayesian games with non-verbal communication, and how to choose optimal actions for a computational agent in an interaction with a human. Chapter 6 demonstrates the model learning techniques described in Chapter 4, and shows results on three sets of data. Our conclusions and directions for future work are described in Chapter 7. There are four appendices which follow the bibliography. They contain most of the mathematical derivations which are not critical an understanding of the work, but

would be necessary for an implementation. The first is a small proof of an equation used in the derivation of the optimization algorithm we use for learning the model. Appendix B shows a derivation used in the tracking equations. Appendix C derives the learning equations for the simple time-independent mixture model, including feature weighting. Appendix D derives the learning equations for the temporal sequence models.

Chapter 2

The Study of Faces and Gestures

This chapter examines some of the prior work on human non-verbal behaviors from the perspective of psychologists and of computer scientists. The literature on this subject is vast, and arises in at least four major research areas. Indeed, much of the work on facial expression and gesture analysis is inherently inter-disciplinary, bridging cognitive science, computer vision, artificial intelligence, and human-computer interaction (HCI) [LS00]. In this chapter, we will first give a broad summary of each of these disciplines, and the relationships between them. The summary will be annotated with some key references to the major works in each field and in each cross-disciplinary endeavour. These will be books, survey articles, or conference proceedings where possible, to give the reader an idea of the broadest place to look for further information. We will then proceed to give a more in depth review of the work in each area. In particular, we will focus on the work in psychology, and the work in computer vision. This will be followed by a review of some of the work bridging these topics. We will show how our work falls into this last category, combining ideas from each of the major research areas.

Figure 2.1 is a schematic of the four major research areas which are important to this thesis. The cognitive science and psychology literature is an important factor, as it studies human behavior, and can give indications about what to expect. Psychologists have been studying facial expression from Darwin's 1872 *The expression of the emotions in man and animals* [Dar72] and Ekman's 1972 *Emotion in the human face* [EFE72], to the more recent expositions in McNeill's 1992 *Hand and Mind: What Gestures Reveal about Thought* [McN92], Fridlund's 1994 *Evolution of Facial Expression* [Fri94], and Russell's 1997 *The Psychology of Facial Expression* [RFD97a]. Section 2.1 discusses this literature in more detail.

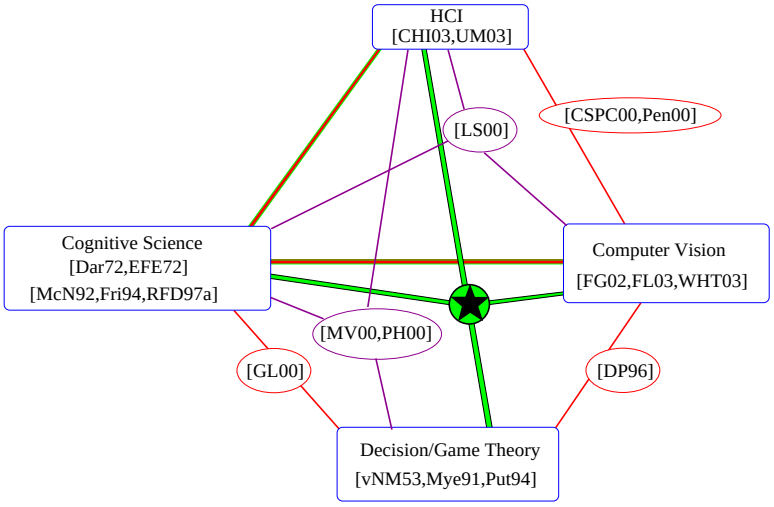


Figure 2.1: Four major research areas (rectangles) and key references for decision theoretic modeling of facial expressions and gestures. Links show that research areas draw from one another. Ellipses show inter-disciplinary research areas and key references. HCI and computer vision both draw heavily from cognitive science, so the links between these fields are emphasised. Finally, the central star shape denotes this thesis topic, and its links to the four major research areas.

Computer vision scientists have been describing human motion for many years. The bi-yearly *Proceedings of the International Conference on Automatic Face and Gesture Recognition* publishes the frontier of research in the field [FG02]. A recent survey on facial expression analysis is Fasel and Luettin's 2003 *Automatic Facial Expression Analysis: A Survey* [FL03], while a more general review of human motion analysis is Wang *et al.*'s 2003 *Recent Developments in Human Motion Analysis* [WHT03]. Much of the computer vision literature on the analysis of faces and gestures draws from cognitive science, but primarily from a small number of works relating to emotion in the human face [EFE72], and from McNeill's book on gestures [McN92]. Section 2.2 delves further into the vast computer vision literature.

Decision theory and game theory play an important role in the design of rational agents. Von Neumann and Morgenstern laid the groundwork for this field in their 1953 *Theory of Games and Economic Behavior* [vNM53]. A more recent book on the subject is Myerson's 1991 *Game Theory: Analysis of Conflict* [Mye91]. Markov decision processes (MDPs) are discussed at length in Puterman's 1994 *Markov Decision Processes: Discrete Stochastic Dynamic Programming* [Put94]. Partially observable Markov decision processes (POMDPs) are discussed by Kaelbling, Littman and Cassandra [KLC98]. We will leave discussion of more recent work with decision

theoretic models to the discussion on MDPs and POMDPs in Chapter 5.

Human Computer interaction, or HCI, considers methods for getting humans and computers to interact more easily. Two major yearly conferences cover much of the most recent research in this field: the CHI (computer-human interaction) conference covers many areas from privacy to video-conferencing [CHI03], while the UM (user modeling) conference deals more specifically with adaptive user models [BCdR03].

Some of the most interesting recent work takes place in the intersections between the four major fields we have just described. Some of these cross-disciplinary areas are also shown in Figure 2.1. Lisetti and Schiano's 2000 *Automatic facial expression interpretation: Where human-computer interaction, artificial intelligence and cognitive science interact* [LS00], gives an informative overview of research combining HCI, computer vision and cognitive science. Cassell's 2000 *Embodied Conversational Agents* [CSPC00] gathers a number of papers from researchers interested in using computer vision in HCI-related tasks. Pentland's 2000 *Looking at People* [Pen00] gives another overview from a computer vision perspective. A new workshop has been initiated to discuss research in using computer vision for HCI: the *IEEE Workshop on Computer Vision and Pattern Recognition for Human Computer Interaction* [CVP03]. The yearly International Conference on Multimodal Interfaces (ICMI) also presents much of the work in this field [ICM03].

Efforts have recently been made at integrating decision theory with HCI. Murray and VanLehn's 2000 paper on *DT Tutor* shows how decision theoretic models can be used to select optimal tutor actions in an intelligent tutoring environment. Paek and Horvitz describe how decision theory can be used to model conversation in their 2000 *Conversation as Action under Uncertainty* [PH00]. Gmytrasiewicz and Lisetti have recently described emotional dynamics using decision theoretic models in their 2000 *Using decision theory to formalize emotions for multi-agent system applications* [GL00]. However, none of these efforts at bringing decision theory to HCI uses computer vision. Active vision is an exception. Darrell and Pentland used decision theoretic models (POMDPs) to control the foveation of an active camera in their 1996 *Active Gesture Recognition using Partially Observable Markov Decision Processes* [DP96].

The work we present in this thesis finds common ground with all four of the major fields we have been discussing. It is motivated by recent work in cognitive science, and uses decision theoretic models with computer vision observations. We learn these models from data taken during interactions between humans and computers. They explicitly integrate computer vision observations with actions, and are designed for human computer interaction.

2.1 Psychology of Facial Expression

Changes in the human face arise from many sources. While it is commonly held that the face is primarily used for the expression of emotion, there are other, perhaps more significant reasons behind facial displays [1]. This is particularly important from a computer vision perspective, for we only have to notice the prevalence of speech-related lip movement during interaction to realise that the modeling of facial motion will involve much more than just emotion recognition. We first give an overview of the research into the expression of emotion in the human face, followed by some of the more recent explorations into other motivations behind facial displays.

Facial expressions and gestures in both humans and animals have been extensively studied over the course of history. Charles Darwin's *The expression of the emotions in man and animals* is perhaps the oldest writing that is still influential today [Dar72]. In it, he attempts to link the emotions with the human face in a two-factor model. He posited that facial expressions were either involuntary expressions of emotion or "willful and employed to social ends" [Fri94]. Darwin's two-factor model was more recently taken up in the *facial expression program* [ER97, Iza97]. The facial expression program assumes that there are somewhere between five and nine basic emotions, and that these emotions are innate, universal and discrete. Other emotions are mixtures, or blends of the basic emotions. Furthermore, each emotion has an involuntary characteristic facial expression associated with it. Expression of emotion in the face can, however, be voluntarily masked, replaced or imitated, but the "true emotion" always leaks out and can be detected in the face.

The facial expression program led to the development of the Facial Action Coding System, or FACS [EW78]. FACS describes the pose of the human face as a combination of basic facial deformations, called *action units*, or AUs. Each action unit is essentially binary, is numbered and is objectively described. For example, AU 1 corresponds to a raised inner brow, while AU10 is the upper lip raised, and AU46 is a wink. A trained human FACS coder can observe a still image of a face, and determine which action units are "on", and which are not. Ekman claims that combining AUs allows a human to express emotion. For example, the expression of happiness corresponds to a mixture of AU $1 + 2 + 10 + 25 + 26$. The FACS coding system has been used extensively in research on the motion of the face [GLW01, Rus97, DBH$^+$99].

Ekman's facial expression program used an experimental method that consisted of showing still photographs of people's faces who were in the midst of displaying what he claimed were prototypical expressions of emotions. The subjects

[1] Hence our use the terminology *display* rather than *expression* to avoid restricting ourselves to emotional expression in the human face.

were then asked to choose one of the five emotions (*fear, happiness, surprise, anger* or *disgust*) as the one most associated with the photographs. Stripped of context as they were, the photographs were very often classified as the emotion they were supposedly displaying [EFE72]. However, in a different experiment, Carroll and Russell [CR96] showed the same photographs to a different set of subjects. Before showing the photographs, they gave the subjects a story to read and told them that the face in the photograph was the face of the person in the story. The subjects were then asked to describe the emotion the person in the photograph was feeling. The results showed that the interpretation of meaning in the face was heavily biased by the context. Subjects ascribed "anger" to "fear" photographs, "hope" to "surprise" photographs. If there is an emotional content to the meaning of facial displays, it does not appear to be categorical and universal, but rather dimensional and contextual. That is, emotions do not occur in discrete categories, such as "happiness" or "sadness", but vary continuously along dimensions such as *arousal* (high-low) and *valence* (positive-negative). Further, emotions are not independent entities that are culturally and individually invariant, but are dependent upon the context in which they occur.

Russell and Fernández-Dols [RFD97b] and Fridlund [Fri94] both point out that Darwin may have been misinterpreted, and that there are other, more pragmatic, ways of understanding facial expressions. They suggest the *behavioral ecology* view of human facial expression, and argue that the human face is used primarily as a *communication tool*, which complements and associates with speech and language. For example, Chovil [Cho91], measured people's facial motion during conversations and noticed that displays *rarely* are expressions of emotion, but rather constitute syntactic and semantic signals. She classified facial displays in terms of their informative function, rather than their physical description. She found that movements in speaker's faces occurred primarily as syntax markers, as illustrations of words, and as additional information. Listeners, on the other hand, used their faces to non-invasively comment on the speaker's dialogue.

There is much evidence to support the hypothesis that faces, as gestures, are only interpretable within the context in which they are displayed, and are individually unique [Fri97, Cho91, McN92]. Context is defined as all circumstances relevant to the display, and may include concurrent or proximate speech and gestures of observer and performer, as well as other environmental factors. For example, eyebrows are sometimes raised during conversation when the speaker is thinking or remembering. However, eyebrows are also raised in back-channel displays of acknowledgment, or when an individual is taking turn in a conversation [Cas00]. In any case, the observed facial display carries little meaning by itself, while the combination of the

current context and the observed facial display is meaningful. Russell provides the following description of an experiment conducted in Russian director Lev Kuleshov's workshop just after the 1917 revolution (From [Rus97], p295):

> ... Kuleshov created three silent film strips each ending with the same footage of a deliberately deadpan face of the actor Ivan Mozhukhin. In one strip, Mozhukhin's face was preceded by a bowl of hot soup, in the second by a dead woman lying in a coffin, and in the third by a young girl playing with a teddy bear. The result was an illusion: Audiences saw emotions expressed in Mozhukhin's expressionless face. [One student] recalled, "The public raved about the acting of the artist. They pointed out the heavy pensiveness of his mood over the forgotten soup, were touched and moved by the deep sorrow with which he looked on the dead woman, and admired the light, happy smile with which he surveyed the girl at play. But we knew that in all three cases the face was exactly the same."

This quote exposes the idea that the interpretation of expressions in faces cannot be decoupled from the context in which they occur.

Non-verbal behavior, including facial displays and gestures, are important aspects of human communication. Agents can use non-verbal signals to facilitate or manage face-to-face interactions. The use of modalities other than speech is an important advantage, which allows for more efficient communication. Consider that agents in a conversation are working together on a *joint project* [Cla96]. That is, they are trying to achieve some common goal which holds value for all involved agents. The agents are working on this goal by communicating propositional content with speech. Using non-verbal displays, however, they can simultaneously issue performative signals, which indicate the reasons for the communication [Aus62, PP00]. Raised eyebrows and rising intonation of speech are typical performatives accompanying a question, and indicate to the listener that they are expected to give an answer. In this case, the content of the question is the proposition, while the raising eyebrows and rising intonation of speech are indicating that the speaker is communicating this statement because he wants the listener to answer a question. Performative acts, therefore, are an efficient method for each agent to *inform* their partner of which actions they are expected to take. That is, the speaker's performative efficiently biases the listener's choice of response action [PP00].

For example, the speaker who raised his eyebrows at the end of a question was telling the listener to answer the question. Suppose the listener, however, was about to make a statement (utter a proposition). She now has two choices. She can either ignore the speaker's performative, and go ahead with her statement, or she can take

the speaker's suggestion and answer the question. In general, she may be able to choose some intermediate actions which attempt to optimise all agent's preferences. The performative is necessary for cooperation in this simple example. Without it, the listener would only have the option of going ahead with her question (she does not know the speaker's utterance required an answer), leading to possible missed goals for the speaker (he does not get his answer). In general, the performative only biases the listener's actions towards those that are of maximal value the speaker. To cooperate, the listener must then choose an action based upon both his values *and* those of the speaker, which are indicated by her performative. At this stage, other factors come into play, such as the power structure between speaker and listener, which may influence how the two values are combined.

This psychology literature lead us to the conclusion that, in order to interact naturally with humans, computational agents will need to recognize (and generate) *situated* and *purposeful* non-verbal signals, which vary widely across individuals and contexts [RN96, Cas00]. That is, they will need to understand the relationship between the observed displays, the context of the interaction, the expected utility (including goals of all involved agents) in order to use the information to make cooperative action choices.

2.2 Describing Human Motion with Computer Vision

Eadweard Muybridge was perhaps the first to analyse human and animal motion from video sequences [Muy87]. Commissioned by Leland Stanford, he attempted to find out what animals and humans actually were doing during typical movements, and his findings shed light on the kinesiology of human and animal motion. For example, his careful analysis of videos of galloping horses demonstrated that the commonly held belief about symmetrical foreleg motion was incorrect, and that galloping was actually a complex, asymmetrical pattern of leg movements.

More recently, computer vision researchers have been trying to automatically analyse human motion. Research has focussed on three major categories: facial expression [MP91, TP91, BY97, EP97, OPB97, LTC97, CET98, DBH$^+$99, TKC01], gesture [SP95, DEP96, WBC97, CT98, VM99, WB99, WPG01], and whole body motion [NA94, BH96, JBY96, Bre97, LB98, LF98, RB99, Bra99a, WCP00, KM00, GJH01, BD01, OHG02]. There is also a large body of work in recognising lip and facial motions connected to speech [LT97, RE01, VBPB03], and on generation of facial motion from speech using models learned from data [BCS97, Bra99b, DH01]. This work has a wide range of potential applications, from visual surveillance to advanced user interfaces. Survey articles have recently been published on both hu-

man motion analysis [Gav99, WHT03], gesture recognition [PSH97], and on facial expression analysis [PR00, FL03]. Lisetti [LS00] also gives an informative multidisciplinary review on facial expression analysis. Our research is distinct from most of this work in that we do not attempt to learn pre-defined categories of motions, instead building a system to automatically discover the important patterns in the data. Research which is similar in spirit to this has been carried out by [DEP96, Bra97, CP99, WCP00, WPG01].

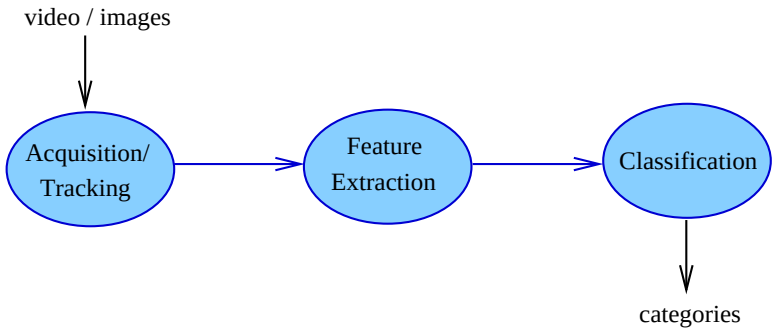


Figure 2.2: General computer vision approach to recognition of human motion maps video inputs to categorical output. It typically involves three stages: acquisition of regions of video, extraction of features, and classification.

Analysis of human motion involves three major stages, as shown in Figure 2.2. First, the acquisition or segmentation of the part(s) of video to be analysed. For example, detecting and tracking of faces, hands, or entire human bodies is the usual first step towards any facial expression, gesture or human motion recognition system. We cover some of the literature on tracking in Section 3.7. Second, the extraction of features from these acquired parts. This is usually done by first estimating some quantity of interest at the pixel level, and then spatially abstracting this to a low dimensional feature vector. Optical flow [MP91, BY97, EP97, CT98, LKCL98, DBH+99], color blobs [SP95, Bre97], deformable models [LTC97], motion energy [BD01], and filtered images [BLB+03], are the more well used pixel-level features. Spatial abstraction is achieved, for example, by using the pixel-level features to modulate a 3D model [EP97], or by projecting them to a low dimensional subspace [TP91, BY97]. The third stage is to classify the extracted features into a discrete set of non-verbal behaviors, such as facial expressions or gestural primitives. Since the extracted features are extended in time, this usually involves some kind of temporal modeling. Spatio-temporal templates [EP97], Dynamic time warping [DEP96], and hidden Markov models [SP95] are all popular approaches.

While much research in the analysis of human motion separates the tasks

of representing the motion and deformation of the body from that of analysing its temporal trajectory, as we have just described, some of the early work considers the spatio-temporal space directly. For example, Niyogi and Adelson [NA94] analysed walking figures by searching the XYT (spatio-temporal) volume of a video segment for repetitive patterns of motion.

The work described in this thesis uses an optical flow and skin-color based tracker to locate and track a human face, described in Section 3.7. Our features are projections of optical flow fields and raw intensity values to a pre-defined set of orthogonal *Zernike* polynomials, described in Chapter 3. Our method for temporal abstraction uses a multi-level dynamic Bayesian network, also described in Chapter 3. Finally, our model explicitly includes actions and utility functions at the highest level (Chapter 5), an addition that is made by none of the above-cited works.

The remainder of this section describes in more detail the previous work the second and third stages of analysis: the representation and classification of human motion. Tracking is not a major focus of this thesis, and we leave review of relevant work to Section 3.7. Most of the methods we review will be strictly *supervised*, in that models are trained for one of several pre-defined categories of human behaviors, and are tested for recognition of the same behaviors. Research on the automatic discovery of human motion patterns from video is more closely related to the approach presented in this thesis, and is reviewed in Section 2.2.3.

2.2.1 Representing the Human Body in Video

One of the most well-used approaches for analysis of human body motion is finding the principal subspace of variation [Koh89]. The basic idea is that although the dimensionality of image space is extremely large (number of dimensions = number of pixels), there is some (typically) low dimensional sub-space in which a particular type of motion can be accurately represented. Finding this subspace, called the principal eigenspace, is achieved using a singular-value decomposition, also known as principal components analysis (PCA). This method has been applied to describing facial structure for face recognition (eigenfaces) [TP91], motion in the face [LKCL98, FBYJ00], and spatio-temporal variation in body motion [BH96, LF98] [2]. One of the difficulties with standard linear PCA is that it can only deal with one mode of variation (e.g. identity *or* expression), but faces vary in many ways at once (e.g. lighting, pose, expression, identity). View-based eigenspaces approach this problem by computing a different sub-space for each

[2]Eigenspaces have been used for many other purposes as well, e.g. modeling pose of static objects [MN95]

view [PMS94]. Another approach is an extension to the standard PCA approach that decomposes along multiple modes at once. This has been called N-mode SVD, or *TensorFaces* when applied to faces [VT02, WA03]. The modes of variation found using PCA are *generative*, in that they describe the overall variation across all classes, and so can be used for accurate reconstruction of images in a dataset. Discriminative methods, on the other hand, find modes of variation that discriminate between classes. The Fisher linear discriminant (FLD), for example, find modes that that describe variation which maximizes the ratio of betweeen class scatter to within class scatter. While FLDs are not useful for describing all images in a dataset, they are very good for distinguishing classes within that dataset [BHK97].

Principal component analysis and Fisher linear discriminants both operate using the second order statistics (variance) of the data. However, much relevant information in a facial expression analysis task may be contained in the higher order relationships between pixels. Independent component analysis (ICA) is a generalization of PCA which capitalises on these higher order statistics for representing images. ICA has an information theoretic and a biological motivation [BS95]. While PCA finds the modes which maximize the *variance*, ICA finds the modes that maximize the *entropy*. Bartlett has used ICA for facial expression analysis [Bar01].

Another popular type of motion and image representation is to project optical flow fields or images to a pre-defined set of basis functions. Translational or affine models are typical for optical flow fields. Mase and Pentland [MP91] used optical flow to model muscle motion over coarse areas of the face. Yacoob and Davis [YD94], and Black and Yacoob [BY97] use a low-order parameterisation of flow fields to describe local areas of deformation in the human face during expressions. Their work is extended in [JBY96] to include models of leg motion. Zernike moments, Hu moments, Legendre moments, and rotational moments have also been studied [TC88, BSA91] for representing images and optical flow fields. Moments of images or optical flow fields are also extensively used. In particular, central moments were used by Little and Boyd [LB98] for representing motion of the human body for gait recognition. The angular radial transform (ART) basis is a type of rotational moment used in the MPEG-7 standard for video encoding [Bob01]. Another popular type of basis set are the Gabor filters, which are attractive due to their close relationship with the receptive fields of simple cells in the visual cortex. Bartlett *et al.* used Gabor-filtered images to train support vector machines for the recognition of action units [BLB$^+$03]. They used hand-labeled feature points on human faces to warp the faces to a canonical view in a pre-processing stage. Gabor wavelets have also been used for face recognition [LBA99].

Template-based approaches store a number of exemplars of face or body

poses, and use some correlation metric to match unlabeled data to the stored exemplars. Bobick and Davis used temporal templates called *motion history images* (MHIs), which encode temporal information in a single image, with more recently occurring events more heavily weighted [BD01]. These MHIs are then spatially abstracted using the Hu moments, and compared to stored templates. The method is tested on aerobics movement sequences, and was used in an interactive play-space for children called the KidsRoom. Efros *et al.* use optical flow templates to characterise motions of soccer players, tennis strokes and ballet steps [ABMM03].

There are a number of model-based approaches in the literature, which attempt to fit facial or body motion and structure to a model of the face or body. Three dimensional models in particular are useful for synthesising computer graphics characters. Terzopoulus and Waters [TK93] used three dimensional wire-frame model of the face driven by tracked facial features. However, their system required facial markers. A similar, but non-invasive, method was used by Essa and Pentland in an approach that uses optical flow to estimate muscle motion [EP97]. They first register the 3D wire-frame mesh to the face by locating the positions of eyes, nose and lips. Optical flow is then measured between registered images, and used in conjunction with the 3D model to estimate the forces underlying the facial motion. The 3D model is also used to constrain the flow field, which can then be used to construct spatio-temporal templates for recognition. Three dimensional models promise to avoid the problems with view-based approaches have with different views of the face or body. Bartlett *et al.* use a 3D face model to which the image is registered [BLB$^+$03]. They currently do the registration manually, although they plan to eventually do so automatically. Models are also extensively used for other body parts. For example, the Vogler and Metaxas fit 3D models of arms to motion capture data [VM98]. Inferring stick figures from motion-capture data is a standard procedure for recovering physically-based models of human motion for graphics applications. However, 3D modeling usually comes at the cost of heavy computational requirements.

Feature points are used for representing motion of faces, hands and bodies. Perhaps the simplest way of doing this is using motion capture systems, or colored markers, which unfortunately are invasive and expensive. Other methods usually involve some kind of model-based approach, in which feature points or edges are located and tracked according to their fit to a model. For example, Lanitis, Taylor and Cootes use flexible shape models which track and adjust to a set of facial feature points [LTC97]. Similar models are used to represent human body silhouettes by Galata *et al.* [GJH99]. Feature points and edges are also used by Tian *et al.* [TKC01] for recognising facial expressions from static images.

A simple and robust method for representing the positions of hands and face regions is the use of image intensity or color "blobs". A blob a statistical model of the distribution of image intensity, color, or optical flow over an oriented region of space. Blobs are estimated from training image, can be simply tracked, and give a coarse estimate of the position and orientation of hands and face. Blobs have been used for analysis of gestures [SP95, JP98, CT98], body motion [Bre97, Bra97, WADP97], and faces [OPB97].

Our work uses Zernike basis functions [vZ34] for holistic representation of the face and facial motion. The Zernike polynomial basis provides a rich and data independent description of optical flow fields and grayscale images which can be seen as an extension of the affine basis. These characteristics make it a desirable approach for unsupervised classification of patterns of motion in the face. When applied to optical flow, the Zernike basis can be seen as an extension of the standard affine basis [HL00]. In fact, Black and Yacoob [BY97] have used the first three orders of the Zernike basis for local representations of motion in the face. The Zernike representation differs from approaches such as Eigen-analysis [TP91], or facial action unit recognition [TKC01] in that it makes no commitment to a particular type of motion, leading to a transportable classification system (e.g., usable for gesture clustering). The Zernike basis has also been used extensively for shape descriptions [Tea80, TC88]. Teh & Chin show that Zernike and pseudo-Zernike moments perform best in terms of noise insensitivity and image reconstruction [TC88]. Zernike moments have also been proposed as shape descriptors for the MPEG-7 video coding standard [KK00], although angular-radial transformations (ARTs) are currently used [Bob01]. ART bases, however, are not orthogonal, making reconstruction (and visualisation) not possible. Zernike polynomials have been used in the vision community for recognising hand poses [HSJ95], handwriting and silhouettes [BSA91], shape-based image retrieval [ZL02], and optical flow fields [HL00].

2.2.2 Classification

Once a region of a set of images containing some human body part(s) has been simply represented, the final task is to classify these representations. The usual method here is to gather and manually label a set of training examples of each motion that needs to be recognised. These training examples are then used to train a classifier, which is tested on unseen data.

Features extracted from a single image can be used directly for classification of the facial pose. For example, the model fits used by Tian *et al.* are used as inputs to a neural network for the classification of FACS units, which are defined for static images [TKC01]. Other methods represent the spatio-temporal volume

using a set of features, explicitly taking the temporal nature of the signal into account [NA94, BD01]. Cutler and Davis analyse periodic motion using Fourier analysis applied to image differences [CD00]. However, one of the most popular approaches is to model the temporal progression of features extracted from images or flow fields using some kind of dynamical model. Simple temporal models of facial expressions were used by Yacoob and Davis and by Black and Yacoob [YD94, BY97]. They characterised the progression of an expression in three phases (beginning, peak and ending), and used a rule-book to classify emotions. Cutler and Turk also use a rule-based temporal classification scheme with features based on the Fourier analysis of blob motions [CT98]. Hidden Markov models (HMMs) are perhaps the best known dynamical model, as their use in speech recognition has been extensively advocated and practiced [RH93]. A hidden Markov model is a *generative* model, in which the observations (feature vectors) arise from one of a discrete set of states. The expected distribution of feature vectors for each such discrete state is characterised with some kind of mixture model, such as a mixture of Gaussians (for continuous features), or a mixture of multinomials (for discrete features). The discrete states are then assumed to be temporally connected in a Markovian chain.

HMMs have been recently been applied to many problems in computer vision. Schlenzig, Hunter and Jain used them to classify hand gestures [SHJ94], while Starner and Pentland used them to recognise American Sign Language (ASL) gestures in a view-based method using color blobs [SP95]. Vogler and Metaxas also recognised ASL signs with HMMs, but used three-dimensional models [VM98]. Their data, however, was captured using magnetic sensors, not computer vision. Morimoto, Yacoob and Davis recognised head gestures using HMMs in a view-based approach [MYD96]. Lien *et al.* and Bartlett *et al.* use HMMs to distinguish FACS facial action units [LKCL98, BLB$^+$03]. While Lien *et al.* use principal components analysis and a vector quantizer to represent the images, Bartlett *et al.* use Gabor filters and support vector machines. Our work used HMMs to recognize facial expressions represented holistically over the face region using projections to Zernike polynomial basis functions [HL00]. Although not in a temporally dynamic setting, HMMs have recently been successfully applied for face recognition, where the Markovian chain is defined over the spatial extent of the face [NI00].

Hidden Markov models are usually only first order, so that the observation at time t is independent of the history given the observation at time $t-1$. However, HMMs with more "memory" represent higher-order temporal dependencies, and can be useful for more complex action recognition. Vogler and Metaxas use second-order HMMs to recognise ASL gestures [VM98], while Galata, Johnson and Hogg use models in which the "memory" length is also learned from the data [GJH99].

Hidden Markov models use a Markov chain over a discrete set of states. A closely relative of the HMM uses continuous state, a model usually referred to as a linear dynamical system (LDS) [Min99]. State estimation in LDSs (forward propagation) is better known as a Kalman filter [Kal60], which has been extensively used in computer vision.

HMMs are really only the beginning of the story on statistical temporal models, however. They are, in fact, a special case of the more general dynamic Bayesian networks (DBNs), which are Bayesian networks in which a discrete time index is explicitly represented. Inference and learning in DBNs is simply an application of network propagation in Bayesian networks [Pea88]. A comprehensive review of DBNs is given by Murphy's thesis [Mur02]. DBNs usually make a Markovian assumption, but explicitly represent conditional independencies in the variables, allowing for more efficient and accurate inference and learning.

A simple DBN extension of HMMs is the coupled hidden Markov model, introduced by Brand, Oliver and Pentland [BOP97] for recognition of simultaneous human actions. CHMMs have two Markovian chains, each modeling a different stream of data. The two chains are coupled to allow modeling of the inter-dependence between the two data streams. They used these models for analysing patterns of pedestrian motions. The coupling between the chains models the interactions between the people. Vogler and Metaxas [VM99] use a similar model for American Sign Language (ASL) recognition, in which the two Markovian chains model the two arms. As pointed out earlier, they do not use computer vision data, but instead capture motion information from wearable magnetic sensors. They show that removing the coupling between the chains leads to increases in efficiency without significantly affecting performance.

Some of the most interesting DBNs model the data at multiple temporal scales. This has been an important issue in speech recognition, where phoneme-level models are combined at a higher level into word or sentence models [RH93]. However, as noted in [BF95], speech recognition systems do not learn high-level semantics (at the word or sentence level), relying heavily on prior knowledge. Computer vision, on the other hand, has made significant progress in learning hierarchical temporal models. A general formulation of hierarchical hidden Markov models is given by Fine, Singer and Tishby [FST98]. Murphy and Paskin have shown how the parameters of this model can be learned efficiently [MP01]. We review some of the work on hierarchical models in computer vision here.

Bregler [Bre97] used a multi-level system which analysed the motion of color blobs using over short time scales using linear dynamical systems (LDSs), and then modeled the LDSs as arising as outputs from a hidden Markov model. The idea is

that long, complex motions can be represented as concatenations of shorter, more simple motions. For example, the motion of the foot during running is bi-phasic (one phase while its on the ground, one while its in the air), and the running motion alternates between the two phases. A similar hybrid dynamical model model has more recently been proposed by Pavlovic, Frey and Huang [PFH99], who also advocate the use of variational methods for estimating parameters.

Bobick and Ivanov [BI98] used a multi-level model that described primitive events using a set of hidden Markov models, and then used a stochastic context-free grammar (SCFG) to provide high-level semantic help. A SCFG enables the modeling of more complex semantic patterns of behavior than a hidden Markov model. They use the SCFG and HMMs to model positions of head and hands acquired using a stereo vision system during the performance of simple gestures, and of musical conducting.

Cohen *et al.* [CSG+03] have recently proposed a multilevel HMM to recognize facial expressions. Their main contribution is a method for automatically segmenting a video stream into facial expression events, assuming a neutral state between each expression. Oliver *et. al* [OHG02] learned a multi-layer model of office activity to choose actions for a computational agent. The model uses multimodal inputs, making only very slight use of computer vision. Our work uses a similar model, called the abstract hidden Markov model, to describe facial expressions [Hoe01].

2.2.3 Unsupervised Classification

Most of the methods we have been describing use extensive training processes with labeled data. That is, their goal is to classify gestures, facial expressions, or body motions into a pre-defined set of categories. Training data is acquired for each of the categories, and is used to train models independently. The likelihood of test data given the learned models is then used for classification of unseen examples. The most significant drawback to such methods is that training data must be manually labeled in a time consuming process. This also implies that adaptation to novel types of motions requires new training data to be acquired and labeled. The alternative is to develop systems that can *discover* categories of motions in training data. These latter systems are more *unsupervised* than the former, and have been much less extensively researched. We give an overview here.

Clustering sequences of data using mixtures of hidden Markov models was proposed by Smyth [Smy97], and has been further investigated by Li and Biswas [LB99]. A mixture of hidden Markov models is a mixture model in which the mixing components are hidden Markov models. Such a model can be trained on unlabeled data, and finds clusters of entire *sequences* of data: it *discovers* which sequences

are similar to each other. This is in contrast to the supervised HMM classification experiments we described in the last section, in which an expert human explicitly labeled this similarity by grouping training data into pre-defined categories.

The mixture of hidden Markov models, and other closely related models have been used in computer vision for unsupervised clustering of data. Clarkson *et al.* [CP99] examine understanding purposeful motion is discussed in the context of wearable computing devices. The idea is to have a camera which is attached to the human subject, and which performs unsupervised, hierarchical modeling of the situations in which the subject finds itself. Their results are promising, and show that unsupervised learning can provide fruitful classes of video data. Alon *et. al.* [ASKP03] have also recently examined such models for action recognition. Darrell, Essa and Pentland [DEP96] examine the same kind of models, but use dynamic time warping (DTW) instead of hidden Markov models as mixture components. They also use these models for generation of facial expressions. They do not explicitly model actions and utilities, however.

2.3 Purposeful modeling of human action

The previous section examined automatic recognition of human actions. However, most of this work takes the slant that the recognition itself is the goal. Clearly, it is what to *do* with the recognised states which is of most interest. Here, we discuss work on integrating the recognition of human action with decision making. Most of this work lies between computer vision and human-computer interaction (HCI), in which computer vision systems provide information about the state or actions of a human user of a computer application. This information is used by the application to tailor its interface. Currently, most HCI systems only make use of human interface actions, such as mouse or keyboard actions, some have begun to integrate visual and auditory information. Clearly, if we want to leave keyboard and mouse behind and progress to more natural interfaces, these multi-modal cues are of utmost importance. Perhaps the simplest application of computer vision for user interfaces is in using head gestures for controlling mouse actions, as described by Kjeldsen [Kje01]. Leibe *et al.* describe a system for interaction with the "perceptive workbench", a virtual reality device in which a user can manipulate objects on a virtual table using their hands [LSR$^+$00]. The device locates and tracks the user with computer vision methods. Pentland reviews a number of systems which make use of facial or gestural inputs [Pen00].

Cassell has stressed the importance of recognising and generating both verbal and non-verbal signals for *embodied conversational agents* (ECAs) [CSPC00]. ECAs

are built upon a conversational architecture: they are designed based upon the psychology of human conversational behaviors. This involves the high-level modeling of communicative and domain goals, the understanding and synthesis of propositional and interactional information, and low-level issues related to recognition of speech and gesture, and to timing. Cassell's group has built an ECA named Rea who interacts with a human as a real-estate agent (hence the name). Rea's visual inputs are the positions and orientations of head and hands from color "blob" trackers.

Jebara and Pentland [JP98] presented *action-reaction learning*, in which a dynamic model was learned from observing video of two persons interacting. The model was then used in a reactive way to simulate interactions for a single user. The features are color blobs of head and hands, and the joint likelihood of the each person's features is accomplished using a variant of the expectation-maximization algorithm. Our work bears a close resemblance to action-reaction learning, but generalizes it by adding high-level context states, actions and utilities. Action-reaction learning is designed strictly for *imitation*-type tasks, while our model is applicable to interactions in more general contexts, in which plans need to be developed autonomously.

A number of robotic systems have made use of computer vision for interactions with humans. Fong *et al.* survey some of the most recent work in building socially interactive robots [FND03]. Breazeal's robot *Kismet* interacts with people on an emotional level, using stereo vision to detect presence, proximity and color [BS99]. Thrun's group have been developing robotic guides. The latest embodiment, *Pearl*, is a robotic helper for the elderly that uses stereo vision [MPR$^+$02]. *José* is a robotic waiter that uses stereo vision for navigation and person detection to accomplish its task of delivering food at a social gathering [EHL$^+$02]. HOMER, Joseé's successor, is a messenger robot that interacts with people [EHL03].

The model we focus on in this thesis is the partially observable Markov decision process, or POMDP. POMDPs were applied to the problem of active gesture recognition in [DP96], in which the goal is to model unobservable, non-foveated regions. POMDPs have also been applied to the dialogue management problem [PH00, RPT00, ZCMG01] for human-computer and human-robot interaction. This work, as Cassell's work on ECAs, models some of the basic mechanics underlying dialogue, such as turn taking, channel control, and signal detection. These agents typically use very few (or none at all) manually specified facial expressions or gestures.

Decision theoretic models have yet to make solid contact with computer vision. The reason is probably because of the lack of optimal or efficient solution algorithms for continuous inputs. However, many approximation techniques have now

been studied [Hau00], and we believe they are appropriate for use in modern computer vision systems. The advantage of using a decision theoretic model is its ability to make provably rational choices in situations involving uncertainty. This thesis presents a method which promises to begin filling this important gap in the interdisciplinary work bridging computer vision, decision theory and human-computer interaction. Its main strength, and what distinguishes it from other research in this area, is that it presents the method as a unified whole, using the paradigm of POMDPs to combine computer vision with actions for user interfaces.

Chapter 3

Modeling Facial Displays

This chapter describes the modeling of human facial motion from video. We are interested in computational models which can recognize and adapt to facial displays and the relationships between the displays and other states of context. These context states, however, exist in a temporally and spatially abstract space as compared to the raw video signal. Our models must therefore perform spatio-temporal abstraction in order to find appropriate correlations between facial displays and context. We first discuss the spatial abstraction methods, which summarize video frames and spatio-temporal derviatives of video frames with projections to a set of basis functions. We then describe how the resulting spatial summarizations are temporally abstracted using dynamic Bayesian networks.

We consider that the instantaneous state of a human's face during a facial display consists of a pose, or configuration, and an instantaneous motion, or dynamics. Dynamics and configuration complement one another, and are akin to velocity and position in classical dynamics. They can be both useful in describing the way a human face moves. In particular, modeling the configuration of the face disambiguates temporal sequences of dynamics, while the dynamics of the face can predict future configurations. Configuration and dynamics are both useful for tracking.

The pose data is simply an image containing the face, while the dynamics data are the spatio-temporal derivatives over the same region from one video frame to the next. Thus, the measurements we start from contain simultaneous descriptions of the instantaneous configuration and dynamics of the face. The spatio-temporal derivatives induce a dense optical flow field, by assuming that the image intensity structure is locally constant across short periods of time. The optical flow field is a projection of the 3D scene velocity to the image plane, and gives the motion in the image at each pixel. The same method is used for spatial abstraction of both the configuration and dynamics of the face. Image regions and optical flow fields are each projected to a set of basis functions, yielding finite dimensional feature vectors.

The distributions of each of these feature vectors (for configuration and dyanmics) are modeled by a mixture model. The states of the mixture model correspond to classes of instantaneous configuration and dynamics of the face in the training data. For example, the configuration classes may correspond to characteristic facial poses, such as the apex of a smile, the apex of an eyebrow raise, or a neutral expression. The dynamics classes are motion classes, and may correspond to, for example, motion during expansion of the face to a smile, or during contraction of the eyebrows from raised to neutral.

The resulting configuration and dynamics mixtures are then each temporally connected in a Markov chain, and the two chains are coupled. This coupled Markov model describes a temporal sequence of facial movement with two interacting distributions over instantaneous pose and dynamics states. Keeping the two chains separated, but coupled, allows additional independencies to be leveraged in the temporal chains. We temporally abstract finite length sequences in this coupled chain by modeling it as the output distribution of another mixture model. This high-level mixture model, then, represents entire sequences of video with a distribution over a finite set of facial display classes. For example, one such class may be a smiling facial display. The corresponding coupled Markov chain would represent sequences of instantaneous facial poses and flows during a typical smiling display. Typically, there will be a small number of classes (< 10), but the model does not preclude more.

Throughout most of this chapter, we will assume that an image region has been selected in each video frame. When modeling facial expression, the human face must be located throughout the video sequence. Specifying these regions involves some method for tracking the region of interest. We discuss the tracking method we have used at the end of this chapter.

This chapter will be structured as follows. The next section describes the Zernike polynomial basis set, which is used for spatial abstraction of images and flow fields. Section 3.2 describes spatial abstraction of the dynamics of the face. including computation of optical flow, projections to basis functions, and modeling of the resulting feature vector using normal distributions. Section 3.3 then describes spatial abstraction of the facial pose, which is very similar to that for the dynamics, but is slightly simpler because it does not involve the computation of optical flow. Section 3.4 briefly discusses some issues related to changes of lighting. These are mainly encountered in more unconstrained environments than we use in our experiments, but are important considerations for future work. Section 3.5 discusses the temporal modeling of sequences of facial pose and dynamics using coupled Markov models, and Section 3.6 describes the temporal abstraction of sequences using mix-

ture models. This section also discusses how to integrate context measurements. We show how the context states can be integrated into the model as conditioning or being conditioned by the mixture components. Section 3.6.4 touches on the issue of temporal segmentation. Finally, Section 3.7 discusses the tracking method we use to locate the face in the video stream, as well as considerations about how to fully integrate the tracking process into the Bayesian model we use for facial motion representation.

3.1 Zernike Polynomial Basis Functions

Spatial abstraction of flow fields and images involves finding appropriate subspaces in which the motions and poses we are trying to categorize are sufficiently well separated. A standard approach to this problem is to compute a data-dependent subspace using principal components analysis, or PCA. This approach has been used for representing general 3-D objects [MN95], facial pose [TP91], and facial motion [FBYJ00]. However, there are two problems with this method. First, the dimensionality of the subspace must be manually specified. While principal components analysis finds a set of basis functions that span the entire observation space, it does not find which correspond to a useful subspace. Typically, the subspace is chosen as the one that accounts for the most variability in the data. However, other choices may be more conducive to the clustering we wish to perform. The second problem with PCA-based approaches is that a separate basis set must be computed and stored for *every* type of motion we wish to recognize.

We believe that a data *independent* subspace surmounts the two aforementioned problems. We choose a complete and orthogonal set of basis functions *a-priori*, and use them for all our modeling methods. The advantage of data independence is that the basis can equally well be used for representing any motion. For example, while we learn classes of face motions in this paper, our system could be easily applied to gestures or gaits, without re-computation of a set of basis functions. Leaving any commitment to particular motions to higher level processing is an advantage in many cases. The usual objection to this type of modeling is that we do not know which set of basis functions are best for a particular modeling task. However, our method includes a feature weighting technique which learns the subset of basis functions which are most useful for the classification task. Thus, as long as we use a complete set of basis functions, we should be able to learn the subset which is best, while only storing the feature weights for each type of motion we wish to recognize, not complete basis sets. We believe that this is an improvement over PCA-based approaches. Further comparison of the PCA approach with the Zernike

basis functions can be found in [HL00].

Data independent basis sets have been used for shape representation in computer vision for many years. In particular, analysis of images using moments has been well studied, including geometric moments, rotational moments, Legendre moments, Zernike and pseudo-Zernike moments [Tea80, TC88]. Teh & Chin show that Zernike and pseudo-Zernike moments perform best in terms of noise insensitivity and image reconstruction [TC88]. Zernike moments have also been proposed as shape descriptors for the MPEG-7 video coding standard [KK00], although angular-radial transformations (ARTs) are currently used [Bob01]. ART bases, however, are not orthogonal, making reconstruction (and visualisation) not possible. In this work, we use the basis of Zernike polynomials as our *a-priori* basis set [vZ34]. The Zernike polynomial basis provides a rich and data independent description of optical flow fields and grayscale images which can be seen as an extension of the affine basis. Zernike polynomials have been used in the vision community for recognising hand poses [HSJ95], handwriting and silhouettes [BSA91], shape-based image retrieval [ZL02], and optical flow fields [HL00].

Zernike polynomials are an orthogonal set of complex polynomials defined on the unit disk [PR89]. The lowest two orders of Zernike polynomials correspond to the standard affine basis. Higher orders represent higher spatial frequencies. The basis is orthogonal over the unit disk, such that each order can be used as an independent characterization of a 2D function, and each such function has a unique decomposition in the basis. Zernike polynomials are expressed in polar coordinates as a radial function, $R_n^m(\rho)$, modulated by a complex exponential in the angle, ϕ, as follows:

$$U_n^m(\rho, \phi) = R_n^m(\rho) e^{im\phi} \tag{3.1}$$

with radial function,$R_n^m(\rho)$, given by

$$R_n^m(\rho) = \sum_{l=0}^{(n-|m|)/2} \frac{(-1)^l (n-l)!}{l![\frac{1}{2}(n+|m|)-l]![\frac{1}{2}(n-|m|)-l]!} \rho^{n-2l}$$

for n and m integers with $n \geq |m| \geq 0$ and $n - m$ even. The first few radial basis functions are therefore:

$$\begin{array}{ccc} R_0^0 = 1 & R_1^1 = \rho & R_2^0 = \rho^2 \\ R_2^2 = 2\rho^2 - 1 & R_3^1 = 3\rho^3 - 2\rho & R_3^3 = \rho^3 \end{array}$$

The indices, n and m, are indicators of the spatial frequency of the Zernike basis function. The larger the value of n, the higher the spatial frequency in the radial direction. Similarly, the larger the value of m, the higher the spatial frequency in the angular direction. For each n, polynomials are defined for a selection of values

for m in the range $\{0, n\}$. Thus, each combination of radial and angular spatial frequencies are available.

The Zernike polynomials are orthogonal on the unit disk and obey the following orthogonality relation:

$$\int_0^1 \int_0^{2\pi} U_n^m(\rho,\phi) U_{n'}^{m'}(\rho,\phi) \rho \, d\phi \, d\rho = \frac{\pi}{(n+1)} \delta_{nn'} \delta_{mm'}, \tag{3.2}$$

where $\delta_{nn'} = 1$ if $n = n'$, and 0 otherwise.

The orthogonality of the basis allows the decomposition of an arbitrary function on the unit disk, $f(\rho, \phi)$, in terms of a unique combination of odd and even Zernike polynomials. That is, [PR89]

$$f(\rho,\phi) \approx \sum_{m=0}^{M} \sum_{n=m}^{N} \left[A_n^m cos(m\phi) + B_n^m sin(m\phi) \right] R_n^m(\rho), \tag{3.3}$$

which can be used to approximate a sufficiently smooth function $f(\rho, \phi)$ to any degree of accuracy by making N and M large enough.

Using the orthogonality relation (Equation 3.2), the coefficients A_m^n and B_m^n can be obtained as

$$\begin{matrix} A_n^m \\ B_n^m \end{matrix} = \frac{\epsilon_m (n+1)}{\pi} \int_0^1 \int_0^{2\pi} f(\rho,\phi) R_n^m(\rho) \begin{matrix} cos(m\phi) \\ sin(m\phi) \end{matrix} \rho \, d\phi \, d\rho, \tag{3.4}$$

where

$$\epsilon_m \equiv \begin{cases} 1 & if \text{m=0} \\ 2 & otherwise \end{cases}$$

Zernike polynomials are defined on a disk, and so an elliptical area must be identified which will be projected onto the Zernike basis. An elliptical region allows for a more flexible image region than a simple disk, and corresponds roughly to the shape of the human face. Although we only consider ellipses with axes aligned with the image axes, it would be possible to allow for rotations of the ellipse in future work. Our tracking method for updating this region in a video stream is described in Section 3.7. Once a scale and centroid have been identified for each flow image, a 2D function (either a flow field or an image region) $f(x, y)$ is projected onto the Zernike basis using the discrete equivalent of Equation 3.4:

$$\begin{matrix} A_n^m \\ B_n^m \end{matrix} = \frac{\epsilon_m (n+1)}{\pi} \sum_x \sum_y f(x,y) R_n^m(\rho) \begin{matrix} cos(m\phi) \\ sin(m\phi) \end{matrix} \tag{3.5}$$

where $\phi = \arctan(y'/x')$, $\rho = \sqrt{x'^2 + y'^2} \leq 1$, $x' = (x - x_c)/r_x$, $y' = (y - y_c)/r_y$, and $\{x_c, y_c\}$ and $\{r_x, r_y\}$ are the centroid and scales of the region of interest.

We can write the projection equation (3.3) for a 2D function $f(x, y)$ as a matrix equation if we write f as a $N \times 1$ column vector by reading pixels from f row-wise from top to bottom, where N is the number of pixels in the function:

$$f = Pz \tag{3.6}$$

The Zernike basis functions are the columns of the $N \times N_z$ matrix P (also arranged row-wise), and the projection coefficients are in the $N_z \times 1$ column vector z, where N_z is the total number of Zernike polynomial basis functions. The columns of P (and the rows of z) correspond to the Zernike polynomials in order of increasing n and m, alternating bewteen A_n^m and B_n^m, such that columns $0, 1, 2, 3 \ldots$ are Zernike polynomials $A_0^0, A_1^1, B_1^1, A_0^2 \ldots$.

3.2 Modeling Facial Dynamics

We wish to classify the dynamics of the human face into a discrete set of classes, starting from image derivatives, ∇f. However, the image derivatives by themselves are not sufficient to describe image motion, because there is a many-to-one correspondence between derivatives and motion. We can constrain the derivatives using the hypothesis that the intensity structure of the scene is locally stable across short time intervals [HS81b]. This allows us to estimate the way things are moving, or the *optical flow*, in the image plane between frames, which is what we want to classify. Optical flow is the projection of the 3D velocity of objects in the scene relative to the camera onto the image plane.

Classification of optical flow fields directly is difficult due to the high dimensionality of the signal. Instead, we classify optical flow in a subspace of flow fields defined by their projections to the Zernike polyomial basis. Thus, we will be finding clusters of subspace optical flow fields by classifying Zernike basis projections. However, simply computing optical flow and subsequently projecting to the Zernike basis does not make use of the uncertainties inherent in the estimation process. We describe an efficient Bayesian solution for directly computing probability distributions over basis coefficients from image gradients using brightness constancy. We show how the method's incorporation of the flow uncertainties improves the estimation of the parametrized flow by discounting image regions with less certain flow vectors. Our mixture model includes weights on the basis coefficients, which describe the effectiveness of each feature at achieving good clusters. Our probabilistic projections ensure that the uncertainties inherent in the calculation of optical

flow are propagated to the cluster membership variables, leading to a more robust clustering.

This section will first describe the extraction of optical flow from video streams. Section 3.2.2 then describes our method for projecting optical flow fields to the Zernike basis, including a method for weighting or selecting features of the projection basis. Some simple experiments are also described, which show the usefulness of the probabilistic projection method.

3.2.1 Optical Flow

As will be shown in Section 3.6, our estimation of optical flow is tightly integrated with the projection to a basis set, and the subsequent classification. Here we describe the underlying mechanics of optical flow estimation. There are many ways to estimate optical flow, usually based on the assumption that local image regions maintain their intensity structure over small periods of time [HS81b]. If some small part of the world appears at time t at some position in the image, then we assume it will appear the same at $t + 1$, albeit in a different position in the image. The difference in position is the (true) optical flow, v:

$$I(x, y, t) \approx I(x + v_x \delta t, y + v_y \delta t, t + \delta t) \tag{3.7}$$

The motion in the image is denoted by a vector optical flow field, $v(x, y)$, which describes the direction and magnitude of the motion of the image pixel at (x, y). That is, if some image region, at (x, y), undergoes an optical flow of $v(x, y)$, the same region can be found after the flow at $(x + v_x(x, y), y + v_y(x, y))$, where v_x, v_y are the components of v in the x and y directions, respectively. The assumption (3.7) is known as *brightness constancy*, and can be used to derive optical flow estimation techniques based on phase, on correlation, or on gradients [BB95].

Gradient based techniques use a Taylor expansion of Equation 3.7, and drop terms above first order [LK81]. Writing the image brightness function as f, such that the spatial and temporal derivatives of the image are ($f_s = \{f_x, f_y\}$) and (f_τ), respectively, the *brightness constancy* assumption is that [1]

$$f_\tau + f_s v = 0 \tag{3.8}$$

The equation says that the change of brightness at a pixel over a time interval is exactly the brightness difference between the current pixel and a pixel separated from

[1] These variables are fields over all N pixels in the image: f_τ is a $N \times 1$ column matrix, $f_s = [f_x f_y]$ is a $N \times 2N$ matrix (where f_x is a $N \times N$ matrix with the horizontal spatial derivative f_x along the diagonal, and similarly for f_y) and $v = [v_x v_y]'$ is a $2N \times 1$ matrix with the components of horizontal and vertical flow.

the current pixel by v, assuming a planar function separating the two pixels. Equation 3.8 actually only constrains the flow in the direction of the spatial derivative (*normal* to the spatial image orientation), since if the flow is normal to the spatial derivative, $f_s v = 0$, and thus v is unconstrained. Typically this is dealt with by introducing constraints on the smoothness of the optical flow field [HS81b, BA96, SAH91]. Since we are projecting to a set of basis functions, we are implicitly constraining the flow to be globally smooth, and so do not need any further regularization.

We also know that the *brightness constancy* assumption is often violated. Occluding edges and reflections are some obvious examples, which can be dealt with using robust methods, for example [BA96]. However, the primary sources of violations of the assumption are failures of the planarity assumption, camera noise, and errors in the derivative calculations. Simoncelli [SAH91] has described how to account for such violations in a Bayesian framework, in which the variability due to each error source is explicitly included in the model. We describe the noise as arising from two independent zero-mean Gaussian noise source, n_1 and n_2, which account for failures of the planarity assumption, and errors in the temporal derivative measurements, respectively. The *brightness constancy* assumption thus becomes [SAH91]

$$f_\tau + f_s \cdot (v - n_1) = n_2, \qquad n_i \sim \mathcal{N}(0, \Lambda_i). \tag{3.9}$$

We can assume that the errors in the spatial derivatives are minimal compared to those in the temporal derivatives, since the temporal sampling is much coarser than the spatial sampling. Equation 3.9 thus describes the conditional probability $P(f_t|v, f_s)$:

$$P(\nabla f|v) \propto \mathcal{N}(f_\tau; -f_s v, f_s \Lambda_1 f_s' + \Lambda_2), \tag{3.10}$$

where $\Lambda_1 = \sigma_1 I_N$, $\Lambda_2 = \sigma_2 I_N$ (I_N is $N \times N$ identity). The important thing to notice about this distribution is the dependence of the variance on the spatial derivative, f_s. The magnitude of the spatial derivative, $\|f_s\|^2$, is the image contrast, which plays an important role in determining the distribution of flow fields [SAH91]. Optical flow is difficult to estimate (and so has high variance) in regions of low contrast.

Using Bayes' rule, we can compute the probability distribution over the flow fields as

$$P(v|f_s, f_t) = \frac{P(f_t|v, f_s)P(v)}{P(f_t)} \tag{3.11}$$

where $P(v)$ is a prior distribution over flow vectors. This distribution is useful if one is interested in the optical flow field itself, in which case a zero-mean prior $P(v) \sim \mathcal{N}(v; 0, \Lambda_p)$ is used in Equation 3.11. Since the likelihood and prior are both

Gaussian, the posterior is also Gaussian, with mean and covariance [SAH91]

$$\mu_v = -\Lambda_v f_s (f_s' \Lambda_1 f_s + \Lambda_2)^{-1} f_t \tag{3.12}$$

$$\Lambda_v = [f_s (f_s' \Lambda_1 f_s + \Lambda_2)^{-1} f_s' + \Lambda_p^{-1}]^{-1} \tag{3.13}$$

If this solution is computed at each pixel independently, then the mean, μ_v, will be approximately the component of the optical flow normal to the local spatial gradient. Estimates of the true optical flow field can be obtained by including a regularizer to enforce smoothness [HS81b]. A simple regularization technique is to simply average over small regions in the image, such that the mean and variance are

$$\mu_v = -\Lambda_v \sum_i w_i f_{si} (f_{si}' \Lambda f_{si} + \Lambda_2)^{-1} f_{ti}$$

$$\Lambda_v = [\sum_i f_{si} (f_{si}' \Lambda_1 f_{si} + \Lambda_2)^{-1} f_{si}' + \Lambda_p^{-1}]^{-1}$$

where the sums are over small image regions around each pixel, f_{si} and f_{ti} are the spatial and temporal derivatives at pixel i, w_i is a weighting function such that points further away are weighting less. We refer to this as the Simoncelli method in the following, and is what we will use to compute optical flow fields where needed. However, we are concerned in this work with *interpreting* the optical flow field as a distribution over a small, temporally and spatially abstract set of discrete states. As we will see in the next section, this will mean replacing the prior over flow fields with a conditional distribution over flow fields given the high-level discrete states. We further parameterize this distribution by projecting to our basis set.

3.2.2 Projections of Optical flow fields

The lowest orders of the Zernike basis, when used to describe optical flow fields, corresponds to the affine basis, which is capable of representing simple planar motions such as translations, rotations, and expansions. The next order polynomials correspond to extensions of the affine basis, roughly the planar projections of *yaw*, *pitch* and *roll*. In particular, Black & Yacoob found these next orders to be particularly useful for modeling motion around the mouth region [BY97].

Equation 3.5 applied to the horizontal flow field, $v_x(x, y)$, gives the projection coefficients, ${}^u A_n^m$ and ${}^u B_n^m$:

$$\begin{matrix} {}^u A_n^m \\ {}^u B_n^m \end{matrix} = \frac{\epsilon_m (n+1)}{\pi} \sum_x \sum_y v_x(x, y) R_n^m(\rho) \begin{matrix} cos(m\phi) \\ sin(m\phi) \end{matrix} \tag{3.14}$$

A similar set of equations is obtained for the vertical flow estimates, ${}^v A_n^m$ and ${}^v B_n^m$. Figure 3.1 shows some example flow fields reconstructed from different orders (values

of n and m) of Zernike polynomials. Higher orders of Zernike polynomials result in flow fields with higher spatial frequencies, representing more complex motions.

The flows can be reconstructed from the coefficients using Equation 3.3, as shown for an example flow field in Figure 3.2. As we reconstruct with more coefficients, we are including higher spatial frequencies, leading to a more accurate reconstruction of the original.

As in Equation 3.6, we can represent the optical flow field projections as $v = Mz$, where

$$v = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \quad M = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix} \quad z = \begin{bmatrix} z_x \\ z_y \end{bmatrix}. \tag{3.15}$$

The columns of P are the N_z basis vectors and z_x, z_y are the Zernike coefficients for horizontal and vertical flow, respectively, $\{{}^uA_n^m, {}^uB_n^m\}$ and $\{{}^vA_n^m, {}^vB_n^m\}$. In practice, M will be some subset of the Zernike basis vectors, the remaining variance in the flow fields being attributed to zero-mean Gaussian noise. Thus, we write $v = Mz + n_p$, where $n_p \propto \mathcal{N}(0, \Lambda_p)$, and so

$$P(v|z) = \mathcal{N}(v; Mz, \Lambda_p) \tag{3.16}$$

The noise, n_p, is a combination of three noise sources: the reconstruction error (energy in the higher order moments not in M), the geometric error (due to discretization of a circular region), and the numerical error (from discrete integration) [LP98]. The choice of a subset of basis elements to use will depend on what the projections are being used for. We discuss a consistent method for making this choice in Section 3.2.4.

3.2.3 Probabilistic Projections

We wish to use these flow field projections for classification tasks, in which some flow field, v, is classified as originating from one of a set of causes, X. In the remainder of this section, we show how to perform this classification using the maximization of a probability function over X. We only discuss classifying individual flow fields in this Section, but the analysis is also used in classifying *sequences* of flow fields, as will be shown in Section 3.6.

Given a set of images, $I_1 \ldots I_{N_t}$, we wish to assign each of the N_{t-1} flow fields one of N_x cluster labels $X_1 \ldots X_{N_x}$, such that the optical flows with the same label are as similar as possible to each other, but as dissimilar as possible from the flow fields with any other label. For example, if describing motion over the human face, states of X may correspond to instantaneous motion fields during *smiling* (mouth expansion), *frowning* (contraction between the eyes), or *talking* (lip

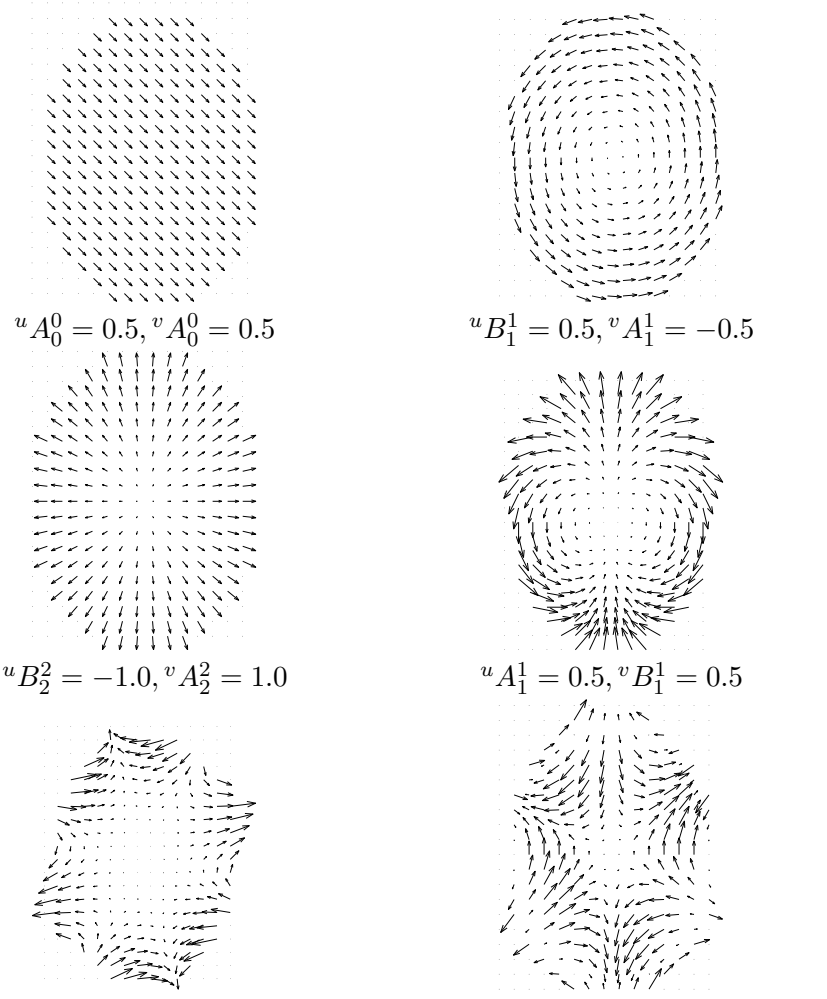


Figure 3.1: Example flows generated from ZPs corresponding to the indicated subsets of the feature dimensions.

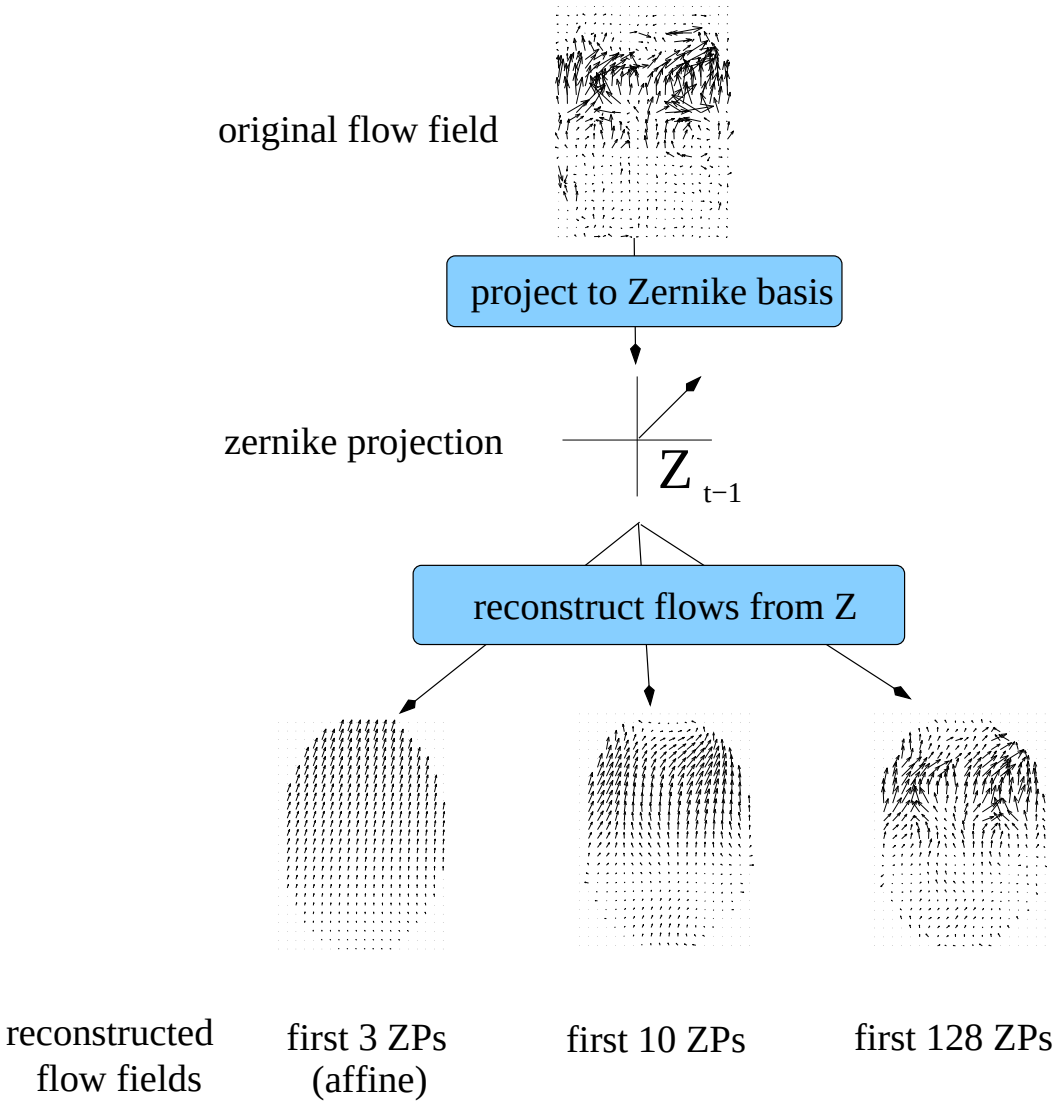


Figure 3.2: Reconstructing flow fields from Zernike projections. Shown are three reconstructions, using increasing number of coefficients from left to right: 3 ZPs or affine ($n = m = 1$), 10 ZPs ($n = m = 3$) and 128 ZPs ($n = m = 14$).

motion). The determination of N_x is discussed in Section 5.3. Figure 3.3 shows our model represented as a Bayesian network. We consider the measurements to be the image spatial ($f_s = \{f_x, f_y\}$) and temporal (f_t) derivatives, calculated using a centered difference method. We can express classification of an image motion as the maximization of the probability distribution over the classes, X, given the spatial and temporal derivatives,

$$P(X|\nabla f, \Theta) \propto P(\nabla f|X, \Theta)P(X|\Theta), \tag{3.17}$$

where Θ are the parameters of the model, and $\nabla f = \{f_x, f_y, f_t\}$. Since we wish to

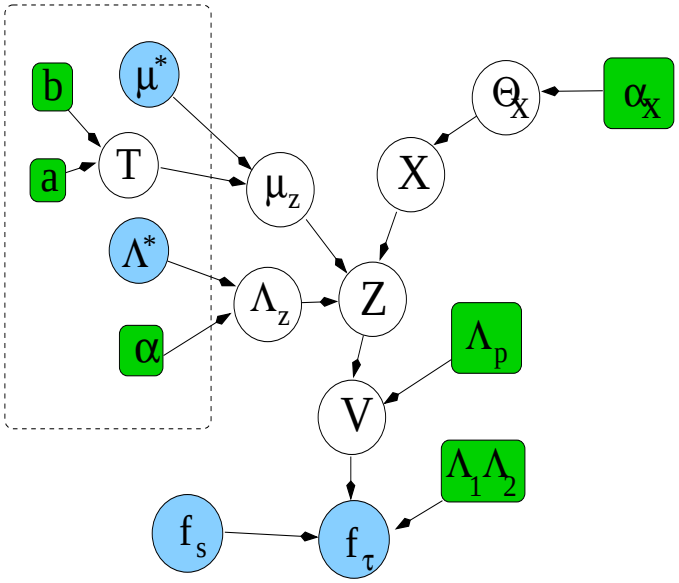


Figure 3.3: Bayesian network for the mixture of Gaussians over optical flow fields with feature weighting. Shaded nodes are observed or fixed (known), while unshaded nodes are unknown random variables. Boxes are fixed hyper-parameters. The dashed line delineates the priors for feature weighting. $X \in 1 \ldots N_x$ are discrete motion classes, Z is the Zernike feature vector (projection of optical flow field), V is the optical flow field, f_s are the spatial derivatives, and f_t is the temporal derivative. μ_z, Λ_z are the parameters of the mixture of Gaussians over the Z vector space, and T are the feature weights. Θ_X are the class probability parameter (a multinomial), and α_X is the parameter of the (conjugate) Dirichlet prior over Θ_X. μ^* and Λ^* are the mean and variance over all classes (of all the data).

classify optical flow fields, we expand the probability distribution over the classes, X, as

$$P(X|\nabla f, \Theta) = \int_v P(\nabla f|v, \Theta)P(v|X, \Theta)P(X|\Theta)$$

where we have assumed the image derivatives to be independent of the high level motion class given the optical flow.

There are three terms in the integration. The prior over classes, $P(X|\Theta)$, is part of our model, parametrized with a multinomial $\Theta_{x,i} = P(X = i)$. The distribution over spatio-temporal derivatives conditioned on the flow, $P(\nabla f|v, \Theta)$, is estimated in a gradient-based formulation using the brightness constancy assumption, and is given by Equation 3.10.

We do not represent $P(v|X)$ directly in our model, but instead we parametrise this distribution using a probabilistic projection of v to the basis of Zernike polynomials. As shown in Equation 3.16, this projection can be written as a distribution over v, given the projection coefficients, z, $P(v|z) \propto \mathcal{N}(v; Mz, \Lambda_p)$. We then parametrise the distribution over z given X with a normal $P(z|X) = \mathcal{N}(z; \mu_{z,x}, \Lambda_{z,x})$. We are assuming that the flow fields will be normally distributed in the space of the basis function projections. The Gaussian noise can be attributed to performance differences in the flow fields (on the part of the human being observed).

We can now write down the likelihood of the image derivatives given the high-level motion class as

$$P(f_\tau|X, f_s, \Theta) = \int_{v,z} \mathcal{N}(f_\tau; -f_s v, A)\mathcal{N}(v; Mz, \Lambda_p)\mathcal{N}(z; \mu_{z,x}, \Lambda_{z,x}) \tag{3.18}$$

where $A = f_s \Lambda_1 f_s' + \Lambda_2$.

A more naïve approach avoids the integrations over z by first computing the mean optical flow field, μ_v, using the Simoncelli method, projecting this field to the Zernike basis, and taking the resulting feature vector, z, as the input data to a classification scheme using $P(z|x)$ [HL00]. That is, the naïve approach considers $P(X|\nabla f) \propto P(M'\mu_v|X, \Theta)P(X|\Theta)$. However, this approach ignores the variance information in the flow calculation, leading to less accurate results. For example, Figure 3.4 shows two frames from a video sequence of a person's face. There is significant motion upwards near and above the eyebrows and downwards along the sides of the jaw. The mean flow field, μ_v, calculated using the method of Simoncelli [SAH91], is shown in Figure 3.4, for the image region of the subject's face as indicated. The certainty of the flow vectors (the trace of the inverse flow variances), is also shown in Figure 3.4 (brighter means the flow estimates are more certain). Large variance flows are prevalent in regions with little contrast since we are using a gradient based optical flow calculation. The jaw, forehead and the background wall are examples. A projection of these flow fields to a low dimensional basis will suffer because of these regions, unless the flow variances are taken into account. The bottom row in Figure 3.4 shows reconstructions of the flow fields from projections to the Zernike basis. On the left, we see a simple projection (dot product), while

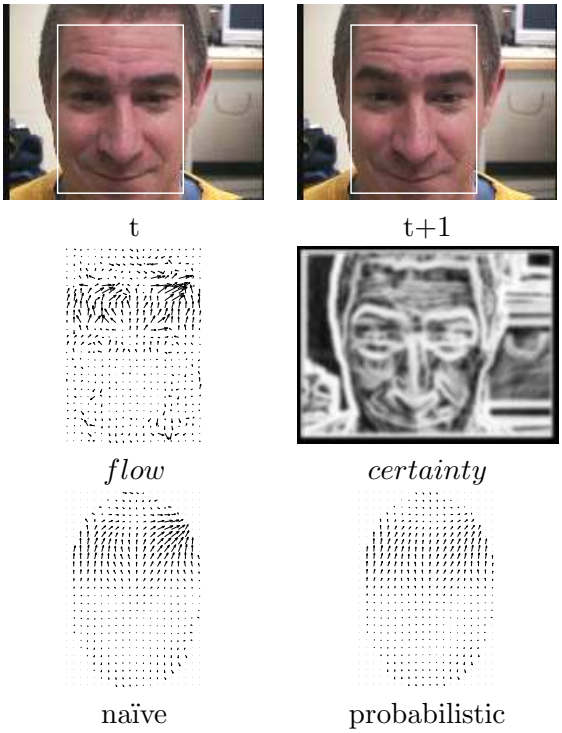


Figure 3.4: Top: two sequential frames from a video sequence. Middle: flow field computing using Simoncelli method (zero-mean prior) and certainty in the flow (trace of inverse variance of the flow). Bottom: naïve and probabilistic reconstructions from low dimensional basis

on the right is the projection which takes the variances on the flow into account. Improvements can be seen in the low contrast area just above the subject's left eyebrow. The Simoncelli method flow field finds nearly horizontal flow vectors in this area, which bias the naïve projections (incorrectly) towards the horizontal. The probabilistic projections discount this area, and the reconstructed flow field is less biased by these errors. We compare our probabilistic projection approach with the naïve approach further in Section 3.2.5.

In the remainder of this section, we show how the integrations in Equation (3.18) can be performed analytically, leading to an efficient method for calculating $P(X|\nabla f)$, taking all variance information in the flow fields into account. We then show how to implement weights on the dimensions of the projections, and how our methods can be implemented in a multi-scale approach. Chapter 4 shows how the distribution $P(X|\nabla f)$ is used to learn the parameters of the model using the expectation-maximization algorithm.

Note that the model does not take violations of the brightness constancy

assumption, such as occlusions, reflections, or transparent motions, into account. It is possible to extend the brightness constancy assumption to account for overall brightness changes [NY93]. We discuss this further in Section 3.4.

Computing the likelihood

Since all terms in Equation 3.18 are Gaussian distributions, we can perform the integrations over v and z analytically by successively completing the squares in v and z to obtain

$$P(f_\tau|Xf_s) = \frac{\sqrt{|\tilde{\Lambda}_{z,x}|}}{\sqrt{|A||\Lambda_{z,x}|}} e^{\frac{1}{2}(\tilde{\mu}'_{z,x}\tilde{\Lambda}^{-1}_{z,x}\tilde{\mu}_{z,x} - \mu'_{z,x}\Lambda^{-1}_{z,x}\mu_{z,x} - \epsilon)} \tag{3.19}$$

where

$$\begin{aligned} \Lambda_w &= (f'_s A^{-1} f_s + \Lambda_p^{-1})^{-1} \\ \tilde{\Lambda}_{z,x} &= (\Lambda_{z,x}^{-1} + M'(\Lambda_p + (f'_s A^{-1} f_s)^{-1})^{-1} M)^{-1} \\ \tilde{\mu}_{z,x} &= \tilde{\Lambda}_{z,x}(\Lambda_{z,x}^{-1}\mu_{z,x} - M'\Lambda_p^{-1}\Lambda_w w) \\ \epsilon &= f'_\tau A^{-1} f_\tau + w'\Lambda_w w \quad w = f'_s A^{-1} f_\tau \end{aligned} \tag{3.20}$$

If we normalize this distribution over X, we can remove all terms which are independent of X, and obtain

$$\frac{P(f_\tau|Xf_s)}{\sum_x P(f_\tau|Xf_s)} = \frac{\sqrt{|\tilde{\Lambda}_{z,x}|}}{\sqrt{|\Lambda_{z,x}|}} e^{\frac{1}{2}(\tilde{\mu}'_{z,x}\tilde{\Lambda}^{-1}_{z,x}\tilde{\mu}_{z,x} - \mu'_{z,x}\Lambda^{-1}_{z,x}\mu_{z,x})}. \tag{3.21}$$

The mean, $\tilde{\mu}_{z,x}$, and covariance, $\tilde{\Lambda}_{z,x}$, are the parameters of the distribution of basis vector coefficients, z:

$$P(z|X\nabla f) = 2\pi^{-\frac{N_z}{2}} |\tilde{\Lambda}_{z,x}|^{-\frac{1}{2}} e^{-\frac{1}{2}(z-\tilde{\mu}_{z,x})'\tilde{\Lambda}^{-1}_{z,x}(z-\tilde{\mu}_{z,x})},$$

The expected value of the Zernike vector for a particular state of $X = x_i$, $\tilde{z}_{x,i}$, is therefore

$$\tilde{z}_{x,i} = \int_z z P(z|x_i \nabla f) = \tilde{\mu}_{z,i} \tag{3.22}$$

and the expected value of Z given the entire model is

$$\begin{aligned} \tilde{z} &= \int_z z P(z|\nabla f) \\ &= \sum_i \int_z z P(z|x_i, \nabla f) P(x_i|\nabla f) \\ &= \sum_i \tilde{\mu}_{z,i} P(x_i|\nabla f). \end{aligned} \tag{3.23}$$

The distribution over X in this expression is computed as

$$P(x_i|\nabla f) = \frac{P(\nabla f|x_i)\Theta_{x,i}}{P(\nabla f)}$$

where $P(\nabla f|x_i)$ is given by Equation (3.19), and $P(\nabla f)$ normalizes the distribution. The expected flow field, $\tilde{v}$, for a given state, $\tilde{v}_x$, and for the whole model, $\tilde{v}$, can be computed as

$$\tilde{v}_x = M\tilde{\mu}_{z,x} \qquad \tilde{v} = M\tilde{z} \tag{3.24}$$

Multi-scale implementation

The brightness constancy assumption fails if the velocity v is large enough to produce aliasing. Therefore, a multi-scale pyramid decomposition of the optical flow field must be used. This results in distribution over the flow vectors, $P(v|\nabla f) \sim \mathcal{N}(v; \mu_v, \Lambda_v)$, where $\Lambda_v = (f_s' A^{-1} f_s)^{-1}$ and $\mu_v = -\Lambda_v f_s' A^{-1} f_\tau$ [SAH91]. Using these coarse-to-fine estimates, Equations 3.20 become

$$\begin{aligned} \tilde{\Lambda}_{z,x} &= (\Lambda_{z,x}^{-1} + M'(\Lambda_p + \Lambda_v)^{-1}M)^{-1} \\ \tilde{\mu}_{z,x} &= \tilde{\Lambda}_{z,x}(\Lambda_{z,x}^{-1}\mu_{z,x} + M'(\Lambda_p + \Lambda_v)^{-1}\mu_v) \end{aligned} \tag{3.25}$$

The mean of this distribution, $\tilde{\mu}_{z,x}$, is a weighted combination of the mean Zernike projection from the data ($M'\mu_v$), and the model mean, $\mu_{z,x}$. The weighting of the data involves the variance Λ_v, which, if we take Λ_1 to be diagonal with entry λ_1, and the scalar variance $\Lambda_2 = \lambda_2$, is

$$\Lambda_v = \left[\frac{f_s' f_s}{\lambda_1 \|f_s\|^2 + \lambda_2}\right]^{-1}$$

In regions of low contrast ($\sigma_1\|f_s\|^2 \ll \sigma_2$), the data weighting increases with the contrast (Λ_v decreases as $\|f_s\|^2$ increases). In regions of high contrast, the $\lambda_1\|f_s\|^2$ term in the denominator normalizes the numerator, $f_s' f_s$, and so the data weighting stays constant. This seems intuitively reasonable, since we would not expect the data to continue being more heavily weighted once the contrast (or signal-to-noise ratio) has increased above the noise level.

3.2.4 Feature weighting

In general, we will not know which basis coefficients are the most useful for our classification task: which basis vectors should be included in M, and which should be left out (as part of n_p). Further, *selecting* a relevant subset of the basis vectors for

clustering can lead to significant computational savings. We use the feature weighting techniques of [CdFGT03], which characterize the relevance of basis vectors by examining how the cluster means, $\mu_{z,x}$, are distributed along each basis dimension, $k = 1 \ldots N_z$. Relevant dimensions will have well separated means (large inter-class distance along that dimension), while irrelevant dimensions will have means which are all similar to the mean of the data, μ^*. Note that the feature weights do not play a role in the recognition task, only in the learning of the model, as we will describe in Chapter 4. The feature weights change the learning process, however, in such a way that the model will be less sensitive to dimensions with lower feature weights.

To implement these notions, we place a conjugate normal prior on the cluster means, $\mu_{z,x} \sim \mathcal{N}(\mu^*, T)$, where T is diagonal with elements $\tau_1^2 ... \tau_{N_z}^2$, and τ_k^2 is the feature weight for dimension k. The prior biases the model means to be close to the data mean along dimensions with small feature weights (small variance of the means), but allows them to be far from the data mean along dimensions with large feature weights (large variance of the means). Thus, τ_k^2 will be large if k is a dimension relevant to the clustering task, while $\tau_k^2 \to 0$ if the dimension is irrelevant. Feature selection occurs if we allow $\tau_k^2 = 0$ for some k.

Conjugate priors are placed on the feature weights, τ_k^2, and on the model covariances, $\Lambda_{z,x}$. Each feature weight is univariate, and so an inverse gamma distribution is the prior on each τ_k^2:

$$P(\tau_k^2 | a, b) \propto (\tau_k^2)^{-a-1} e^{-b/\tau_k^2}. \tag{3.26}$$

This prior allows some control over the magnitude of the learned feature weights, τ_k^2. The model covariances are multivariate, for which the conjugate prior is an inverse-Wishart prior:

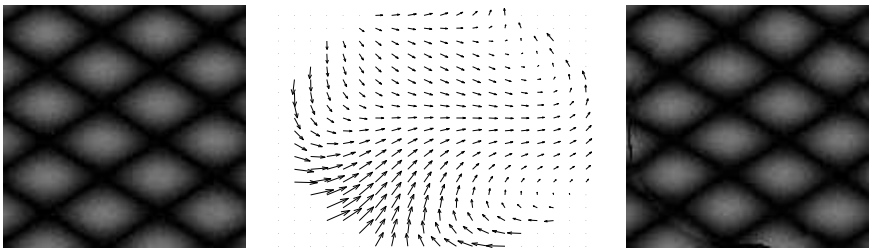
$$P(\Lambda_{z,x} | \alpha, \Lambda^*) \propto |\Lambda_{z,x}|^{-(\alpha+N_z+1)/2} e^{-\frac{1}{2} tr(\alpha \Lambda^* \Lambda_{z,x}^{-1})}, \tag{3.27}$$

where Λ^* is the covariance of all the data, and α is a parameter which dictates the expected size of the clusters (the intra-class distance). This prior stabilizes the cluster learning.

3.2.5 Experiments with Probabilistic Projections

We performed two experiments to examine the advantages of using the probabilistic projection described by Equation (3.25) over the naïve projection. In the first, flow fields were reconstructed from 500 twenty-dimensional Zernike vectors, with all coefficients randomly generated in the interval $[-1, 1]$. The resulting flows (< 5 pixels/frame) were used to warp a synthetic 120×120 image using linear interpolation.

The original image, shown in Figure 3.5(a), is a sine grating with added Gaussian noise $\sigma = 5$ greyscale values. An additional amount of Gaussian noise ($\sigma = 5$ again) was added after the warp. Figure 3.5 (b) and (c) show an example flow field and the corresponding warped image, respectively. Optical flow was projected to



(a) original (b) flow field (c) warped

Figure 3.5: Synthetic images and flows.

the Zernike basis using both naïve and probabilistic methods. The probabilistic projection used a single x state with a zero mean prior, $\mu_{z,x} = 0$, and a diagonal covariance $\Lambda_{z,x} = \sigma_{z,x} I$, with $\sigma_{z,x} = 0.01$. The coefficients were compared with the ground truth. The mean Euclidean distances were 1.08 ± 0.32 for the probabilistic projection, and 1.55 ± 0.37 for the naïve projection. The mean difference (naïve-probabilistic) was 0.48 ± 0.13, showing that the probabilistic projection outperforms the naïve method, as expected.

Our second experiment used the synthetic Yosemite flow-through sequence, constructed from an aerial image and a depth map [BFB94]. Ground truth over the ground region is used to evaluate the performance of the projection methods. There are fourteen 316x252 frames in the sequence, and the flow fields range up to 4 pixels/frame. The ground sections of two frames are shown in the top row of Figure 3.6, while the middle row shows the ground truth and the estimate of the flow field using the method of [SAH91]. We pre-smoothed each image using separable Gaussian filters ($\sigma = 1.0$), and used a 3-level Gaussian pyramid. The noise parameters were set to $\sigma_1 = 0.08$, $\sigma_2 = 1.0$, $\sigma_p = 10.0$, $\sigma_0 = 0.5$ and $\sigma_d = 0.1$. The angular error [2] on the flow estimate is 8.4 ± 12.0. The first 10 Zernike coefficients were estimated for all frames using both naïve and probabilistic projections, and were used to reconstruct flow fields over an ellipsoidal region covering the rigid portion of the scene using Equation 3.3. We used a single zero-mean $\mu_z = 0.0$ model with diagonal covariance $\Lambda_z = 0.001$. The bottom row in Figure 3.6 shows the reconstructed flow fields for the two projections. The average angular errors over all frames for the reconstructions were 6.27 ± 6.36 for the naïve and 5.66 ± 6.22

[2] The angular error, E, between ground truth v_c and estimate v_e is $E = \arccos(v_c \cdot v_e)$, where $v = \frac{1}{\sqrt{u^2+v^2+1}}(u, v, 1)^T$

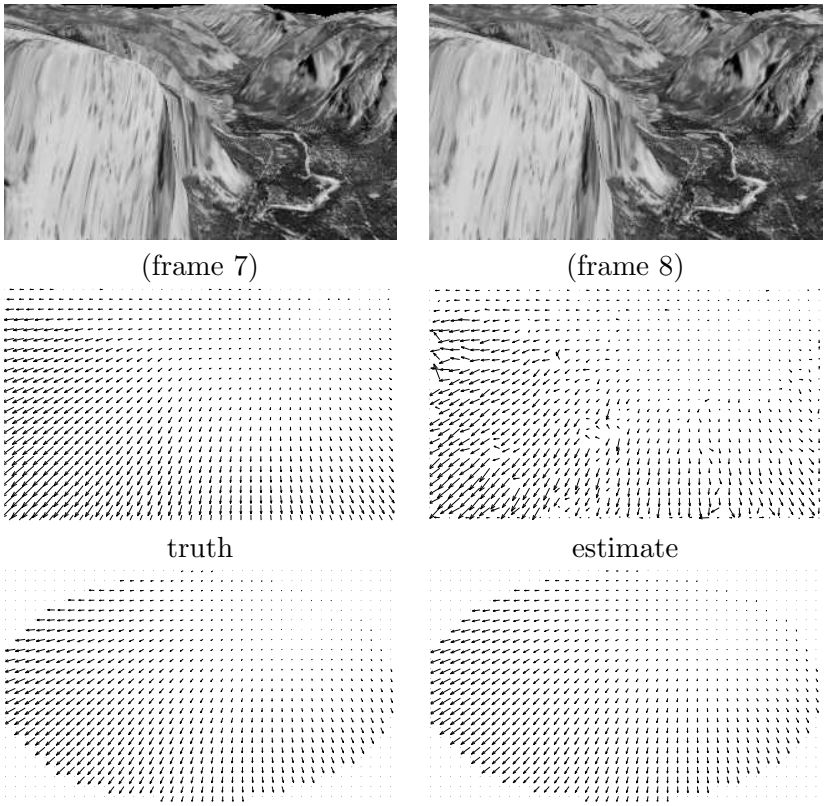


Figure 3.6: Top: two frames from the Yosemite sequence. Middle: Ground truth flow and estimate using [SAH91]. Bottom: reconstructed flow fields.

for the probabilistic projections. Again, we see the advantage of the probabilistic projection.

3.3 Modeling Facial Configuration

We use the Zernike polynomial basis to describe image regions as facial poses. We use a small subset (only the first 32 basis polynomials), which gives a very coarse approximation to the brightness structure over the face. However, the modeling of configuration is only secondary to the dynamics modeling we have just described, and is used to disambiguate zero-motion flow fields from each other, as we discussed in the beginning of this chapter. Our focus has been primarily on the dynamics modeling. It may be desirable in the future to build more precise models of configuration. The temporal modeling techniques we describe in Sections 3.5 and 3.6 would still be applicable, however, and we proceed with this simplified model.

Equation 3.5 is used directly to compute the coefficients, A_n^m and B_n^m, of the projection of an image region, H, to the Zernike basis. H is the model estimate of the image over the projection region. Equation 3.6 allows us to represent H as $H = Pz$, where the columns of P are the N_z basis vectors and z are the Zernike coefficients, A_n^m and B_n^m. As with the flow fields, we can write $H = Pz + n_q$, where $n_q \propto \mathcal{N}(0, \Lambda_q)$, and so $P(H|z) = \mathcal{N}(H; Pz, \Lambda_q)$. As we will show, the same feature weighting techiques apply equally well here for selection of basis vectors.

Figure 3.7 shows some examples of Zernike polynomials of different orders (values of n and m) as grayscale images. Higher orders of Zernike polynomials represent more complex brightness patterns.

Images can be reconstructed from the coefficients using Equation 3.3, as shown for an example image in Figure 3.8. As we reconstruct with more coefficients, we are including higher spatial frequencies, leading to a more accurate reconstruction of the original.

In the following, we only discuss classifying individual images, but the analysis will be combined with that for the flow field projections in Sections 3.5 and 3.6. The classification of image configurations is to assign each of a set of images, $I_1 \ldots I_{N_t}$, to one of N_w cluster labels $W_1 \ldots W_{N_w}$, such that the images with the same label are as similar as possible to each other, but as dissimilar as possible from the images with any other label. For example, if describing the human face, states of W may correspond to instantaneous poses during *smiling* (mouth expanded), *frowning* (contraction between the eyes), or *talking* (lips in a particular configuration). We use the same model described in the last section, as shown in Figure 3.9, which is the same as Figure 3.3, except the labels have changed. The measurements

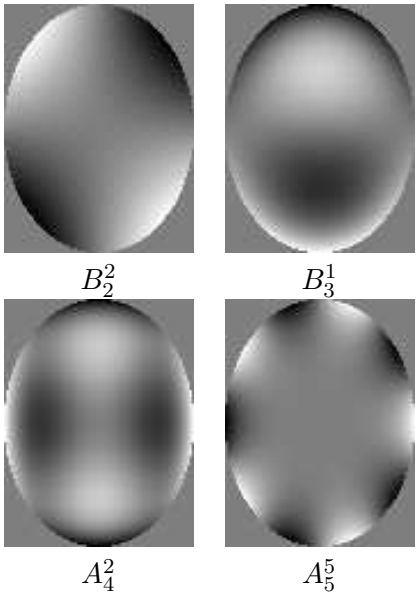


Figure 3.7: Example images generated from individual ZPs as shown.

are now the images, I. The subspace projections over image regions are labeled H. We can express classification of an image as the maximization of the probability distribution over the classes, W, given the image:

$$P(W|I,\Theta) \propto P(I|W,\Theta)P(W|\Theta), \tag{3.28}$$

where Θ are the parameters of the model. Since we plan to classify image projections, we expand this probability distribution as

$$P(W|I,\Theta) = \int_{h,z} P(I|h,\Theta)P(h|z,\Theta)P(z|W\Theta)P(W|\Theta)$$

There are four terms in the integration. The prior over classes, $P(W|\Theta)$, is part of our model, parametrized with a multinomial $\Theta_{w,i} = P(W = i)$. The distribution over z given W is parametetrised with a normal $P(z|W) = \mathcal{N}(z; \mu_{z,w}, \Lambda_{z,w})$. The distribution over the image regions given the Zernike projection is normal, $P(h|z\Theta) = \mathcal{N}(h; z, \Lambda_q)$, as previously described. The distribution over images given the subspace image region, h, $P(I|h,\Theta)$, can be approximated using a normal distribution at each pixel, $P(I|h,\Theta) \sim \mathcal{N}(I; h, \Lambda_h)$.

The integrations over h and z can be performed since all the distributions inside the integral are Gaussian. The results are similar to that for the dynamics (Equation 3.21):

$$P(W|I,\Theta) \propto e^{\mu'_\diamond \Lambda_\diamond^{-1} \mu_\diamond - \mu'_{z,w} \Lambda_{z,w}^{-1} \mu_{z,w}} \tag{3.29}$$

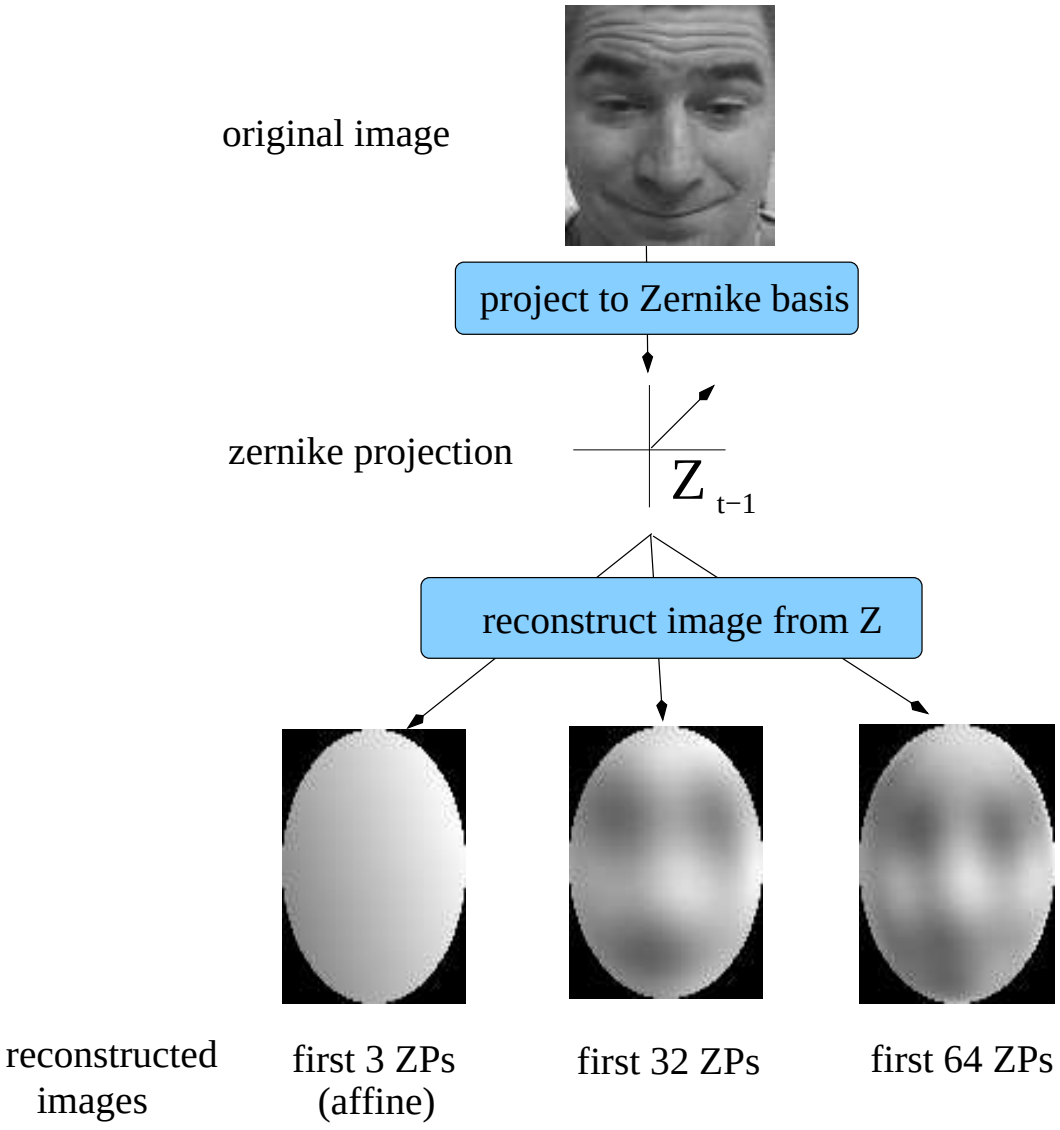


Figure 3.8: Reconstructing images from Zernike projections. Shown are three reconstructions, using increasing number of coefficients from left to right: 3 ZPs or affine ($n = m = 1$), 32 ZPs and 64 ZPs.

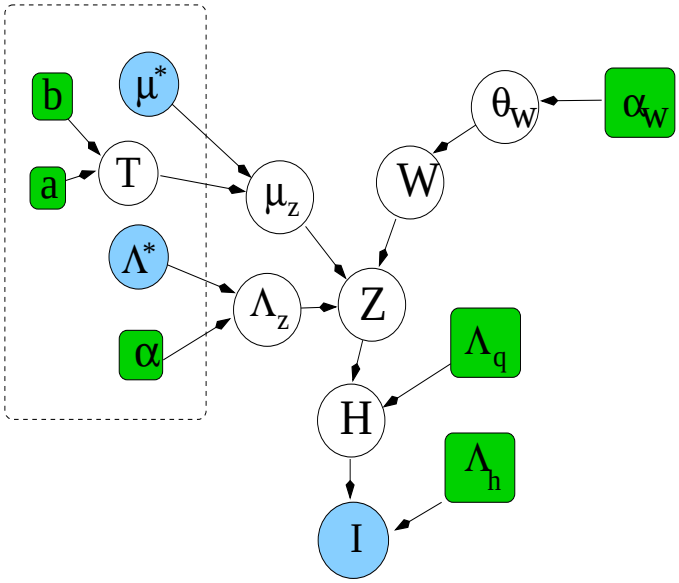


Figure 3.9: Bayesian network for the mixture of Gaussians over images with feature weighting. Shaded nodes are observed or fixed (known), while unshaded nodes are unknown random variables. Boxes are fixed hyper-parameters. The dashed line delineates the priors for feature weighting. $W \in 1 \ldots N_w$ are discrete image classes, Z is the Zernike feature vector (projection of image), H is the projected image region, and I is the full image. μ_z, Λ_z are the parameters of the mixture of Gaussians over the Z vector space, and T are the feature weights. Θ_W are the class probability parameter (a multinomial), and α_W is the parameter of the (conjugate) Dirichlet prior over Θ_W.

where

$$\mu_\diamond = \Lambda_q^{-1} \left[\Lambda_h^{-1} + \Lambda_q^{-1}\right]^{-1} \Lambda_h^{-1} I' + \Lambda_{z,w}^{-1} \mu_{z,w} \tag{3.30}$$

$$\Lambda_\diamond = \left(M' \Lambda_q^{-1} M - M' \Lambda_q^{-1} \left[\Lambda_h^{-1} + \Lambda_q^{-1}\right]^{-1} \Lambda_q^{-1} M - \Lambda_{z,w}^{-1}\right)^{-1} \tag{3.31}$$

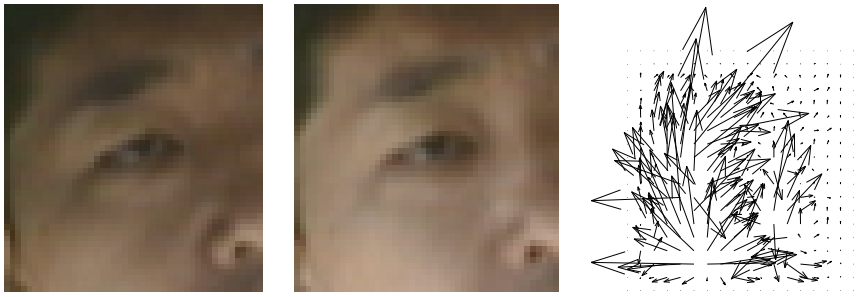
In this case, however, there are no data-dependent variances as in the flow field case. Therefore, the naïve projection is a good approximation to the full integration, and we write $P(W|I\Theta) = P(z_W|W\Theta)P(W|\Theta)$, where $z_W = M'I$ is the projection of the image region to the basis set.

3.4 Lighting Changes

The modeling of pose and dyanmics is sensitive to lighting changes which are due to factors other than motion in the face. Shadows passing over and reflections onto

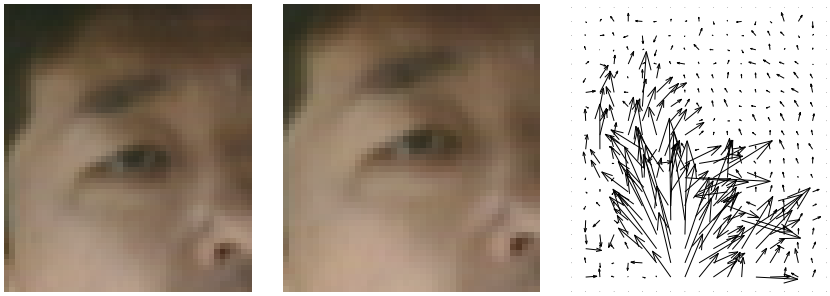
the face are both examples of environmental lighting changes that will seriously affect the models we have described. Such violations are common in videos taken outdoors or with a moving camera. The data we analyse, however, is taken in a laboratory setting, with constant diffuse light, and we do not encounter these problems. Nevertheless, we give a brief discussion of the problem here, as it would have to be dealt with for a system operating outdoors, or on a mobile platform.

The dynamics process is also affected by external lighting changes. As described in Section 3.2.1, optical flow is estimated using brightness constancy, which assumes that any brightness changes in the image are due only to motion in the scene (which is what we are trying to estimate). Brightness changes that occur due to external causes violate the brightness constancy assumption, and result in erroneous flow estimates. An example is shown in Figure 3.10, in which a face (which is stationary) undergoes a lighting change due to a new light source. The flow estimation algorithm assumes the changes in brightness are due to motion of the face, and estimates flow accordingly.



frame 2515 2519 erroneous flow

Figure 3.10: Example of erroneous flow generated by a large lighting change. The subject's head does not move, but the flow estimates are large due to the lighting cast on the subject's face by a passing object.



frame 2541 2542 erroneous flow

Figure 3.11: Example of erroneous flow generated by a small lighting change

Large brightness differences can be easily detected since they result in large violations of brightness constancy. $I_x u + I_y v + I_t = E_b \gg 0$. Thus, we can threshold the value of E_b and reject the corresponding (erroneous) flow fields. Unfortunately, even small brightness changes can cause erroneous flow measurements, as shown in Figure 3.11, in which a small lighting change on the subject's face causes a violation of brightness constancy of the same magnitude as that encountered in normal motion estimation, and so escapes the thresholding operation. It is possible to extend the brightness constancy assumption to account for overall brightness changes [NY93].

The effects on the configuration modeling are clear: the images with lighting effects will project to different regions of the Zernike vector space. This is a well studied (and difficult) problem in computer vision. One solution is to have different models for each possible lighting condition, or to estimate the lighting condition based on the image. However, this involves adding much complexity to the method. Another solution is to use spatial filters to remove the effects of lighting. For example, working in hue-saturation space only (removing luminosity) has been known to remove some of the problems.

3.5 Temporal Modeling

We have seen in the last two sections how to compute spatially abstract representations of images and their derviatives. These models are applicable only for individual time frames, however, and we would like to describe what happens over longer time periods. To this end, we combine the two spatial representations together in a single temporal model, as shown in Figure 3.12. This dynamic Bayesian network is a flavor of hidden Markov model, known as a coupled hidden Markov model, or CHMM [BOP97]. The idea is to model the temporal dependencies of the dynamics and configuration, as well as the interaction between the two, at a high level of abstraction. The coupled hidden Markov model represents the temporal evolution of the dynamics and configuration using two Markovian processes over X and W, which we call the *dynamics process* and the *configuration process* respectively. The transition probabilities in these two processes are parametrized by $\theta_X = P(X_t|X_{t-1}W_t)$, $\theta_W = P(W_t|W_{t-1}X_{t-1})$, while the intial state probabilities are parametrized with $\pi_X = P(X_0)$ and $\pi_W = P(W_0)$. Note that the coupling in this model is reduced from the usual full coupling in a CHMM [BOP97]: we have introduced the additional independency that X_t is independent of W_{t-1} given W_t and X_{t-1}. This is because we do not expect the dynamics to depend on the previous *and* current configurations given the previous dynamics. We do, however, expect that our beliefs about the dynamics will be affected by the current configuration (the

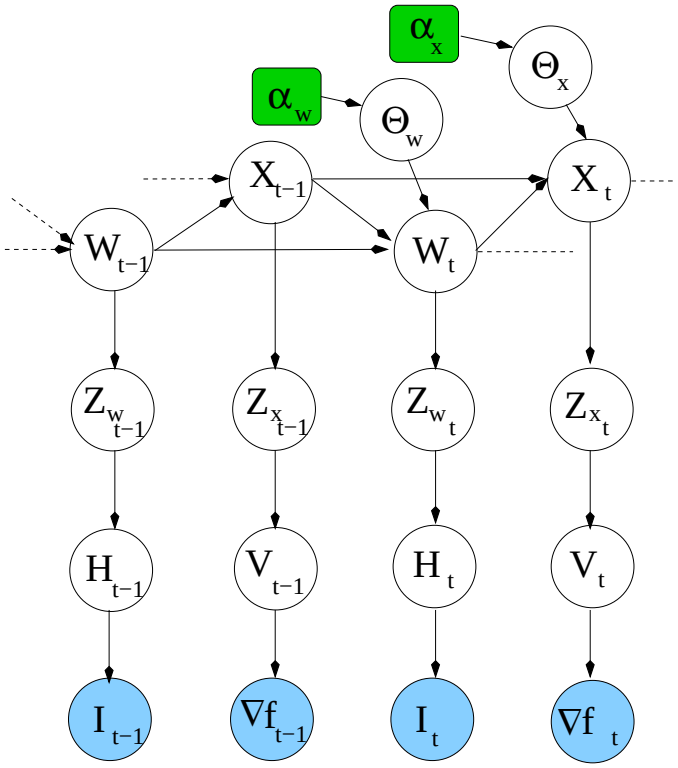


Figure 3.12: Two time slices of a dynamic Bayesian network (DBN) for simultaneous modeling of pose and dynamics. This model combines the mixture models in Figure 3.3 and 3.9 by coupling the high-level mixture variable, X and W in a coupled temporal Markov chain. We have left out explicit representation of the model parameters μ_z, Λ_Z, T, and other low-level noise terms, but included the transition parameters, Θ_X and Θ_W and their priors, α_X and α_W, respectively.

pose gives us information about what changes may be expected) and the previous dynamics (smoothness of the dynamics process).

Computing the likelihood of a sequence of T images and derivatives, $\mathbf{I}, \nabla\mathbf{f}$ given the parameters of the coupled hidden Markov model, Θ, is accomplished by using an extension of the usual "forward" computation from HMM estimation [Rab89]. We can derive a recursive formula for this computation, in which we write $O \equiv I\nabla f$,

and an entire sequence of data as $\mathbf{O} = \underset{1,T}{\{O\}} = \{I_1 \dots I_T, \nabla f_1 \dots \nabla f_T\}$,

$$\begin{aligned} P(\mathbf{O}|\Theta) &= \sum_{ij} P(X_{T,i} W_{T,j} \underset{1,T}{\{O\}} | \Theta) \\ &= \sum_{ij} P(I_T \nabla f_T | X_{T,i} W_{T,j} \underset{1,T-1}{\{O\}} \Theta) P(X_{T,i} W_{T,j} \underset{1,T-1}{\{O\}} | \Theta) \\ &= \sum_{ij} P(I_T | W_{T,i} \Theta) P(\nabla f_T | X_{T,j} \Theta) \sum_{kl} P(X_{T,i} W_{T,j} X_{T-1,k} W_{T-1,l} \underset{1,T-1}{\{O\}} | \Theta) \\ &= \sum_{ij} P(I_T | W_{T,i} \Theta) P(\nabla f_T | X_{T,j} \Theta) \sum_{kl} \Theta_{Xijk} \Theta_{Wjkl} P(X_{T-1,k} W_{T-1,l} \underset{1,T-1}{\{O\}} \Theta) \end{aligned} \tag{3.32}$$

where

$$\theta_{Wjkl} = P(W_t = j | W_{t-1} = k, X_{t-1} = l)$$

and

$$\Theta_{Xijk} = P(X_t = i | X_{t-1} = j W_t = k)$$

Given that we have a coupled hidden Markov model such as we have just described, we may be interested in estimating the temporal state evolution of the dynamics and configuration processes for some set of observations. That is, we want to find the single best state sequences, $X_1 \dots X_T$, and $W_1 \dots W_T$ given observations, $\mathbf{O}_1 \dots \mathbf{O}_T$, which can be formulated as maximizing $P(\underset{1,T}{\{\mathbf{X}, \mathbf{W}\}} | \underset{1,T}{\{\mathbf{O}\}} \mathbf{\Theta})$. The *Viterbi* algorithm performs this maximization. It needs to be modified slightly to accomodate the two temporal chains. Appendix D.1 derives the Viterbi algorithm for this model.

3.6 Temporal abstraction

We have just described the modeling of sequences of input data using the hidden state of two Markovian processes. A single such model is only useful for classifying data on a frame by frame basis, so, except for the tracking and the addition of temporal dependence in the dynamics and configuration processes, we are no further ahead than at the end of Section 3.3. However, recall that we wish to build temporal abstractions into our model, and infer high-level states which can be used in a decision-making agent. The state of the dynamics and configuration processes at any time, t, will be descriptions of instantaneous motion and pose. Over some longer period of time, however, the sequence of states will be a description of the

longer term trajectories of motion and pose. For example, suppose two states of the dynamics process, X_i, X_j, correspond to instantaneous flow fields observed when the subject raises or lowers their eyebrows, respectively. If the subject "flashes" their eyebrows (raises, then quickly lowers them), the sequence of states in the dynamics process over the course of the display may start in X_i at the beginning of the expansion phase, change to X_j at the apex of the display, and remain there until the end of the retraction phase. However, the subject may also hold their eyebrows raised for a period of time, during which the dynamics process will be in a state X_k corresponding to no motion. The sequence of states will then start in X_i, change to X_k at the apex of the display, and change to X_j at the beginning of the retraction phase of the display.

If it is important to distinguish these two displays, then we must hypothesise a higher level, discrete, variable, D, which conditions the lower level variables, as shown in Figure 3.13. This figure shows a mixture of CHMMs like the one in Figure 3.12. For clarity, some of the variables have been grouped together: the joint space of the mixture variables is denoted $Y \equiv \{X, W\}$, the joint space of the Zernike projections is $Z \equiv \{Z_x, Z_w\}$, the joint space of the image and flow projections is $G \equiv \{H, V\}$, and the joint space of observations is denoted $O \equiv \{I, \nabla f\}$. The high-level motion state at some time τ, D_τ, explains the sequence of observations, $\mathbf{0}$, from $t = g_\tau \ldots g_{\tau+1} - 1$. The indicator variables, g_τ, are indices to frame times which demarcate the beginning of each sequence explained by a D_τ. In general, the sequence of data from g_τ to $g_{\tau+1} - 1$ will be part of a longer sequence. The question of how to find the temporal segments, g, is discussed in Section 3.6.4. The parameters of this model are the prior over high-level states, $\Theta_{Di} = P(D_i)$, and a set of parameters as in the coupled hidden Markov model (CHMM) for each state, D. For example, the transition probability over the dynamics process, X, will now be conditioned on D: $\Theta_{Xijkl} = P(X_{g_{\tau+t},i}|X_{g_{\tau+t-1},j}C_{g_{\tau+t},k}D_{\tau,l})$. This model is a mixture of hidden Markov models [Smy97]. The estimation of the likelihood of a sequence of data, $\mathbf{O}$, given this model can be computed as by summing over D as follows

$$P(\mathbf{O}) = \sum_k P(\mathbf{O}|D_k)P(D_k) \tag{3.33}$$

The likelihood of the data for a particular high-level state, D_n, $P(\mathbf{O}|D_n)$, is computed using the same equations as for the CHMM model (Equation 3.32), with all

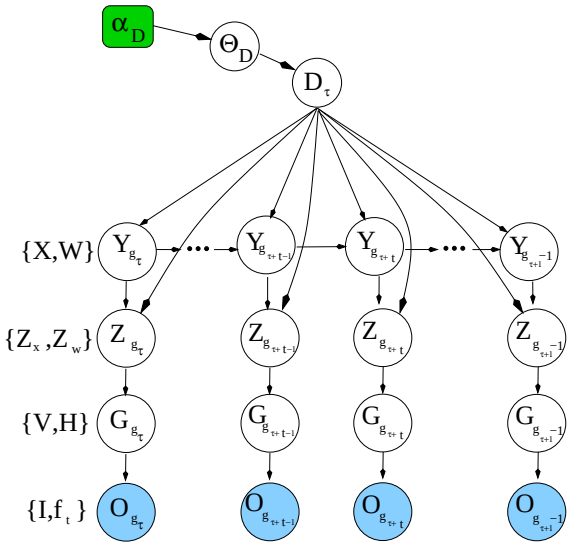


Figure 3.13: Mixture of coupled hidden Markov models as a dynamic Bayesian network. The mixture variable, D, conditions a CHMM like the one shown in Figure 3.12, where we have grouped sets of variables together: $Y \equiv \{X, W\}, Z \equiv \{Z_x, Z_w\}, G \equiv \{H, V\}$, and $O \equiv \{I, \nabla f\}$. The multinomial mixture parameter is Θ_D, and α_D is the parameter of its Dirichlet prior.

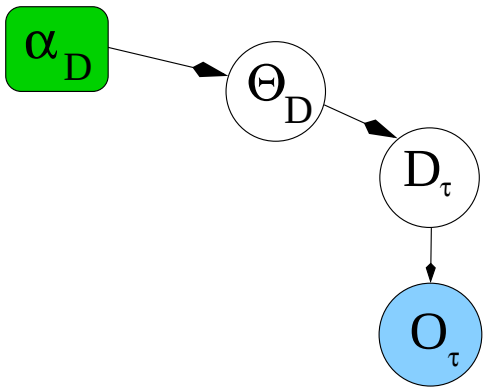


Figure 3.14: Simplified mixture of hidden Markov models as a Bayesian network. This is the same model as shown in Figure 3.13, except we have left out explicit representation of the hidden variables, Y, Z and G.

parameters conditioned on D_n:

$$P(\mathbf{O}|D_n\Theta) = \sum_{ij} P(I_T|W_{T,i}D_n\Theta)P(\nabla f_T|X_{T,j}D_n\Theta) \sum_{kl} \Theta_{Xijkn}\Theta_{Wjkln}P(X_{T-1,k}W_{T-1,l}\underset{1,T-1}{\{O\}}|D_n\Theta) \tag{3.34}$$

We may also want to compute the most likely high-level motion sequence class, D^*, given a set of observations, $\mathbf{O}$, which means solving

$$D^* = \arg\max_i P(D_i|\mathbf{O}) = \arg\max_i P(\mathbf{O}|D_i)\Theta_{Di}$$

where $P(\mathbf{O}|D_i)$ is computed using Equation 3.34, and Θ_{Di} is the model parameter.

The model in Figure 3.13 is a mixture of coupled hidden Markov models, which we will refer to as a 3MG in the following (there are three 'M's in MCHMM). We can simplify notation by representing the network in Figure 3.13 as by the one in Figure 3.14. In this figure, $\mathbf{O}_\tau = \mathbf{0}_{g_\tau} \ldots \mathbf{0}_{g_{\tau+1}-1}$, and we have left out explicit representation of the hidden states, Y, Z, G. This representation makes clear the fact that we are again dealing with a mixture model, as we were in Section 3.2.2, and will be useful in the following section.

3.6.1 Context Dependent Mixtures of Hidden Markov Models

Recall that we are interested in modeling the dependence of facial displays on context. We assume that context is observable, and comes in two flavors: context which

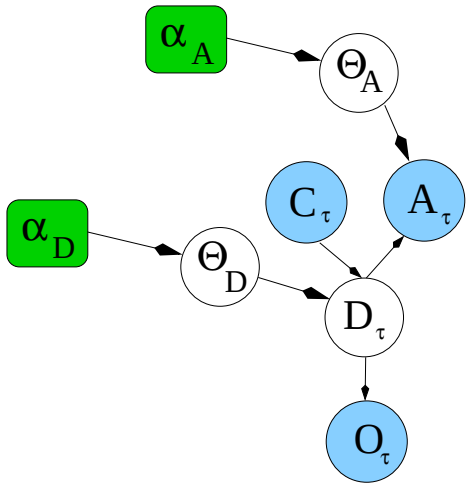


Figure 3.15: Context Dependent Mixture of hidden Markov models as a dynamic Bayesian network, including context variables, C and A, and the additional parameter, $\Theta_A = P(A|D)$.

affects the facial displays, and context which is *affected by* the facial displays. In the simple games we consider in Chapter 5, the state of the game affects what kinds of facial displays the players use, and the actions of the players are affected by the displays of their partner. Thus, the context can either condition (in which case we label it C) or be conditioned by (in which case we label it A), the high-level motion state, D. This gives us the model in Figure 3.15. We refer to this model as a C3MG (a Context dependent 3MG). Since the context is observable, the likelihood of interest is now the likelihood of all the observations,

$$\begin{aligned} P(\mathbf{O}C_iA_j|\Theta) &= \sum_k P(D_k\mathbf{O}C_iA_j|\Theta) \\ &= \sum_k P(A_j|D_k)P(\mathbf{O}|D_k)P(D_k|C_i)P(C_i) \\ &= \sum_k \Theta_{Ajk}\Theta_{Cki}P(\mathbf{O}|D_k)P(C_i) \end{aligned} \tag{3.35}$$

where we have introduced two new parameters, $\Theta_{Ajk} = P(A_j|D_k)$ and $\Theta_{Cki} = P(D_k|C_i)$, and the likelihood of the data, $P(\mathbf{O}|D_k)$, is computed using Equation 3.34.

Again, we want to compute the most likely high-level motion sequence class, D^*, given a set of observations, $\mathbf{O}, C_k, A_l$, which means solving

$$D^* = \arg\max_i P(D_i|\mathbf{O}, C_j, A_k) = \arg\max_i P(\mathbf{O}|D_i)\Theta_{Aki}\Theta_{Dij}$$

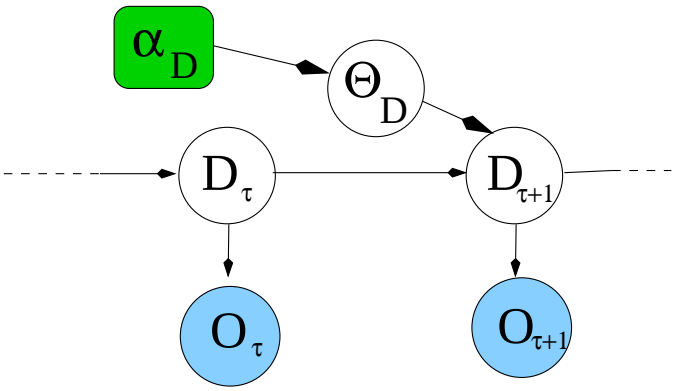


Figure 3.16: Markov chain of mixtures of hidden Markov models (4MG) as a dynamic Bayesian network.

where $P(\mathbf{O}|D_i)$ is computed using Equation 3.34, and $\Theta_{Dij}, \Theta_{Aki}$ are the model parameters.

3.6.2 Markov chains of Mixtures of Hidden Markov Models

The highest level motion descriptors, D, in a 3MG may also not be temporally independent. If we assume that their dependence is also Markovian, then we have the dynamic Bayesian network shown in Figure 3.16. This is a Markov chain of mixtures of hidden coupled hidden Markov models, which we refer to as a 4MG (4 'M's in MMCHMM). Such a model is also known as an embedded hidden Markov model [NI00], or a hierarchical mixture of Markov chains [Hoe01]. The likelihood for this model can be computed in the same way as for a normal hidden Markov model (HMM), using the likelihood function given by Equation 3.34:

$$\begin{aligned} P(\mathbf{O}|\Theta) &= \sum_k P(D_{T,k}\{\underset{1,T}{\mathbf{O}}\}|\Theta) \\ &= \sum_k P(\mathbf{O}_T|D_{T,k}) \sum_l P(D_{T,k}|D_{T-1,l}) P(D_{T-1,l}\{\underset{1,T-1}{\mathbf{O}}\}|\Theta) \\ &= \sum_k P(\mathbf{O}_T|D_{T,k}) \sum_l \Theta_{D,kl} P(D_{T-1,l}\{\underset{1,T-1}{\mathbf{O}}\}|\Theta) \end{aligned} \tag{3.36}$$

This is a recursive formula which can efficiently computed using the usual forwards algorithm from HMM estimation [Rab89].

To compute the most likely sequence of $D_1 \dots D_T$ given a set of sequences of observations $\mathbf{O}_1 \dots \mathbf{O}_T$, we use the standard Viterbi algorithm [Rab89], using $P(\mathbf{O}|D)$ as given by Equation 3.34 as the likelihood function.

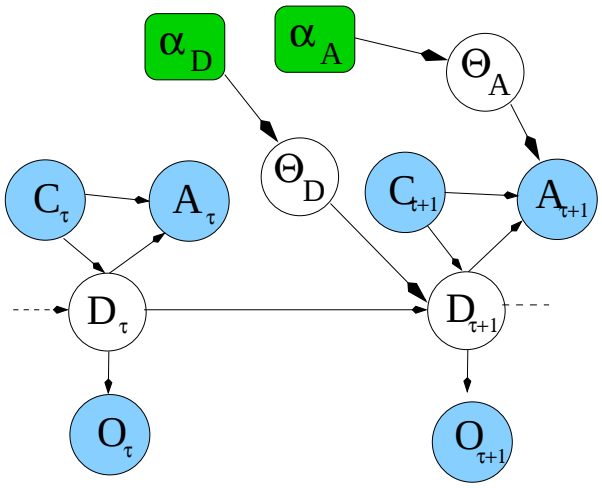


Figure 3.17: Context dependent Markov chain of mixtures of hidden Markov models (C4MG) as a dynamic Bayesian network, including context variables, C and A.

3.6.3 Context Dependent 4MGs

Adding context variables to the model shown in the last section gives us the one shown in Figure 3.17. This model is a C4MG (context dependent 4MG). This model is known as an input-output hidden Markov model, or IOHMM [BF96]. In this case, the context variable, C, is the input observations, and both the image data, $\mathbf{O}$, and the conditioned context, A, are output observations. Again, the likelihood is computed using a recursive formula which combines elements from Equation 3.36 and 3.35.

$$
\begin{aligned}
&P(\mathbf{O}, \mathbf{C}, \mathbf{A}|\Theta) \\
&\quad = \sum_k P(D_{T,k}\{\underset{1,T}{\mathbf{OCA}}\}|\Theta) \\
&\quad = \sum_k P(A_{Tj}|D_{T,k}) P(\mathbf{O}_T|D_{T,k}) \sum_l P(D_{T,k}|C_{T,i}D_{T-1,l}) P(D_{T-1,l}\{\underset{1,T-1}{\mathbf{OCA}}\}|\Theta) \\
&\quad = \sum_k \Theta_{Ajk} P(\mathbf{O}_T|D_{T,k}) \sum_l \Theta_{D,kil} P(D_{T-1,l}\{\underset{1,T-1}{\mathbf{OCA}}\}|\Theta)
\end{aligned} \tag{3.37}
$$

The likelihood is the same calculation as for an IOHMM, except that the output observation distribution, $P(A, O|D)$, is factored into two terms, $P(A|D)P(O|D)$.

To compute the most likely sequence of $D_1 \ldots D_T$ given a set of sequences of observations $\mathbf{O}_1 \ldots \mathbf{O}_T, C_1 \ldots C_T, A_1 \ldots A_T$, we again use the Viterbi algorithm, except that now the likelihood function is a product of two terms, $P(\mathbf{O}|D)$ and $P(A|D)$, and the transition function over D is selected based on the value of C for

each time frame.

3.6.4 Temporal Segmentation

The models we have just presented all describe a finite length sequence of data, denoted by the indicator variable, g_τ. In a complete Bayesian model, these indicators will be part of the model, and must be integrated out, as in a hierarchical hidden Markov model [FST98]. Although linear time algorithms exist for such models [MP01], we claim that the segmentation can be achieved by looking at the context variables, C and A. We have assumed these to be fully observable, and to be happening at event times which span multiple frame times in general. Thus, the frame time of occurence of these context states can be used directly as the indicators, g. For example, in the simple card game we consider in Chapter 5, players take turns playing cards or making bids. These plays and bids are clear temporal markers in the games, and we do not expect facial displays to span the plays. We expect the players to communicate and play sequentially, not simultaneously. Previous authors have approached this temporal segmentation problem by manually fixing it [OHG02], exhaustive search for the most likely time scale [WCP00, WBC97], or by searching for discontinuities in the temporal trajectories [WPG01, RA00].

3.7 Tracking

The previous sections all assumed that some region in the image had been selected for projection. While in some cases we may be interested in the camera's field of view (the entire image), in most we will be interested in the motion of some figure upon some background, for example. Whatever aspect of this figure we are interested in describing the motion of (for example, the human face) must be located in each frame that we wish to process. In a video stream, however, images occur at regular 33 millisecond intervals, and we can use temporal continuity to make this location process much simpler.

One of the primary requirements of our face tracker is that the tracked region be consistently registered to the face: the facial features must show up in exactly the same positions relative to the tracked region. Furthermore, the tracker must be robust to failures and so must be able to re-initialize automatically. For example, if we envision an online user interface, then the tracker must be ready to lock onto people when they appear, and re-initialize after they turn their heads, or bend down to pick up a dropped pencil. However, since our tests are performed with humans using a computer, their faces will tend to be fairly stationary, and will be facing the

camera (mounted above the screen). Therefore, we can focus more resources on the registration than on the localization.

While many head and body trackers have been implemented, most are designed to robustly maintain a rough lock on the tracked region, but maintain tracking through occlusions, full 360° rotations, and distractions (other faces present). For example, the Birchfield tracker uses color histograms and intensity gradients to fit an ellipse to the head region [Bir98]. Jepson *et al.* describe robust, adaptive, appearance models which are learned on-line using the EM algorithm for tracking of natural objects (including faces) [JFEM01]. Again, these models are tailored for robustness in the face of occlusions and variations in 3D pose. There are many other examples of tracking algorithms in the litterature. Our method uses an optical flow tracker with corrections from an exemplar database and skin color detection. It is similar to the method used by Kruger [KZ02]. Exemplars give the ability to accurately register the face. Skin color gives the ability to re-initialize after failure. It may also be desireable to also track facial features, such as pupils or eyes, for additional registration.

Section 3.7.1 gives a procedural description of the optical flow tracker with corrections from exemplar images to avoid long term drift, and briefly discusses some experiences we have had with this tracker for very long video sequences. Section 3.7.2 then describes how the tracker is actually a special case of a more general class of dynamic Bayesian models. This interpretation would allow us to integrate tracking with modeling of facial displays, as described in previous sections, in a consistent framework. Further investigations into this Bayesian tracking model would be necessary in the future.

3.7.1 Tracker: procedural description

The tracking problem is to update a region as described by centroid and scale parameters $\alpha_t = \{x_c, y_c, r_x, r_y\}$ from one frame, t, to the next, $t+1$. The centroid is $\{x_c, y_c\}$ (pixels) while the sides of the image region are $\{r_x, r_y\}$ (pixels). We assume that there is only one region to be tracked in all frames. A schematic of our tracking method is shown in Figure 3.18.

Since optical flow is an estimate of the change between two frames, we get an initial tracker update from the first and second order coefficients of the projected flow estimates

$$\begin{aligned} x'_c &= x_c + {}^u\tilde{A}^0_0 \quad y'_c = y_c + {}^v\tilde{A}^0_0 \\ r'_x &= r_x + {}^u\tilde{A}^1_1 \quad r'_y = r_y + {}^v\tilde{B}^1_1 \end{aligned} \tag{3.38}$$

where $\tilde{A}$, $\tilde{B}$ are components of $\tilde{z}_t$ (Equation 3.23). However, updates using only

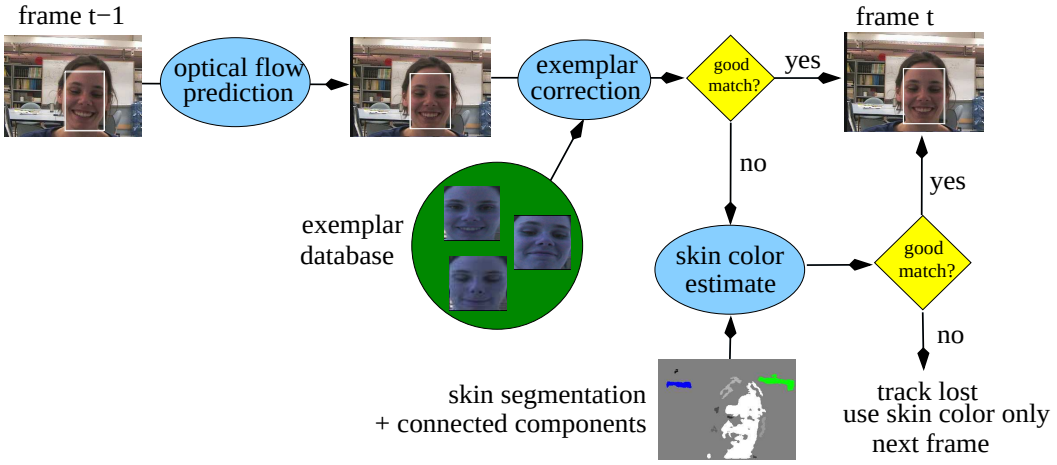


Figure 3.18: Procedural description of the tracking process. Estimates of optical flow are used to update the track, with corrections to prevent long term drift by matching raw image regions to an exemplar database. Skin segmentation is used to re-initialize after tracker failure.

the flow are prone to significant drift over any sequence longer than roughly 600 frames (20 seconds). We correct for this drift at each frame using color template image matching. We maintain a set of templates (also called *exemplars*), which are example images of the region we are tracking in a variety of poses under different lighting conditions. These exemplars should span the space of poses and lighting conditions we expect to see in the subject's face, and restrict the tracker to operate only for a particular subject. Although it may be possible to automatically select the exemplars [KZ02], we select them by hand in the following way. The tracker is run with human supervision until it fails, at which point an exemplar is selected which allows the tracker to recover from failure. This process is continued until the tracking is stable, and then the sequence is re-tracked from the start without supervision. In a typical laboratory setting, only about 20-30 exemplars are needed. In more complex environments, this tracker would probably not suffice, and would need further work.

The exemplars are all scaled to be the same size: $N_e \times N_e$. The scaling can be done using the Mitchell algorithm [Sch92], or using graphics rendering hardware if available. We search over small local corrections to the scale prediction, $\alpha'_t = \{x'_c, y'_c, r'_x, r'_y\}$, and compare the image in each corrected region to each exemplar by first scaling the image region to the fixed exemplar size ($N_e \times N_e$) and then computing the distance using the sum of squares pixelwise metric. The most likely scale correction (which could be nothing) is then computed based on these distances.

Severe tracker failure can still occur for a variety of reasons, including head rotations, changes in lighting, and occlusions by the subject's hands, such as when scratching the face. Tracker failures also occur when the subject's face leaves the camera's field of view. For example, they may bend down to pick up a dropped item. Therefore, a robust tracker needs some way of initializing a track without prior knowledge of the previous location of the region of interest. Tracker failure is signaled if either of two error measures crosses a threshold: first, the degree to which the brightness constancy assumption is violated in the flow estimate, and second, the exemplar match scores. The thresholds are assigned manually after observing the tracker behavior.

We accomplish tracker initialization using skin color segmentation and the exemplar database. We transform the RGB color images to HSV (hue-saturation-value) space, and segment the image using simple thresholding in hue and saturation. Median filtering removes noisy estimates, and we find the connected components in the resulting binary image. The bounding box, b_0, of the largest skin component is our initial estimate of scale. However, such an image region will not necessarily be the same as any exemplar in the database, since the exemplars are manually selected regardless of skin color segmentation. Thus, for each exemplar, say h_k, we maintain a *skin scale mapping*, M_k^s, which maps the bounding box, b_k, of h_k's largest connected skin component, to the actual bounding box that was selected for the exemplar manually, $b^\dagger$: $M_k^s(b_k) = b^\dagger$. The inverses of the skin scale mappings are $M_k^{s^{-1}}(b^\dagger) = b_k$. For a given exemplar, k, our estimate of the scale in the image, α_0^k, given the bounding box b_0 is then $\alpha_0^k = M_k^s(b_0)$. The most likely initial scale is then computed as before, but using these skin-based predictions instead of the usual flow-based predictions. The skin-scale mapping enables the initialization of the tracker to be fairly independent of the performance of the skin segmentation procedure. For example, the skin segmentation shown in Figure 3.18 only classifies about half the subject's face as skin pixels. In this particular example, it is because the HSV thresholds had to be tightened in order to deal with a nearly skin-colored portion of the background. However, as long as the mapping between the skin segmentation bounding box and some exemplar is *consistent*, the correct scale can still be correctly estimated.

Figures 3.19 and 3.20 show some representative results of the tracker's performance on data of a human's face while she is playing a simple card game through a video link with another player (see Chapter 5). The figure shows part of the interface to the tracker software, with the image on the left with the current location of the track. On the right, the interface shows the most recently computed skin segmentation and connected components results (from the last failure, since the

skin estimate is only recomputed when the tracker fails). Components are shown in different colors or shades of grey. Along the bottom of each image are shown a selection of the exemplars, ranked by their match scores from left to right: the best match is on the far left. Only the 10 best exemplars are shown out of a total of 26 in this example. Note that the frame rate in this example is only 12 frames per second, making the tracking more difficult than some of our other experiments, which were run at 30 frames per second. The tracker is operating normally in Figure 3.19, with updates usually based only on optical flow estimates, and occasionally on skin color. Figure 3.20 shows the tracker failing due to the subject throwing her head back while laughing. This pose, in which the face is barely visible, is not contained in the exemplar database, and so the tracker locks onto another part of the image. The tracker recovers after the subject's face returns to a normal, frontal pose. This sequence is nearly 10,000 frames long, or about 14 minutes at 12 fps. The tracker successfully recovered from failure in all but two instances, where manual intervention was required due to the tracker locking onto an image region that was not the face, but had close enough template matches to pass the failure detector. Typical tracker speeds are over 10 frames per second for an tracked region of about 100×80 pixels.

3.7.2 Tracker: Bayesian model

The last section described an optical flow based tracker, with corrections from template database matches. However, this approach separates the tracking problem from the recognition problem, although the two are tightly coupled. This section gives a rough overview of how tracking and recognition can be combined in a single Bayesian model. The method described in this section has not been implemented, and is described for future work only. Nevertheless, we show how our tracker can be derived as a special case of the more general version presented here. The model implements essentially the same tracker as we described in the last section, except that the exemplars are replaced with the configuration modeling described in Section 3.3. Thus, optical flow provides predictions about the tracked region, and the configuration model provides corrections to the track. Addition of the track causes efficiency problems, however, as it must be integrated out during the computation of any quantities of interest. We discuss some approximation methods for surmounting these difficulties.

Figure 3.21 shows the same model as in Figure 3.12, but with the facial region, α, explicitly represented. The scale, α forms another Markovian chain, called the *transformation process*, which conditions both measurements, ∇f and I. In our previous derivations, we have always assumed that the scale is fixed, such

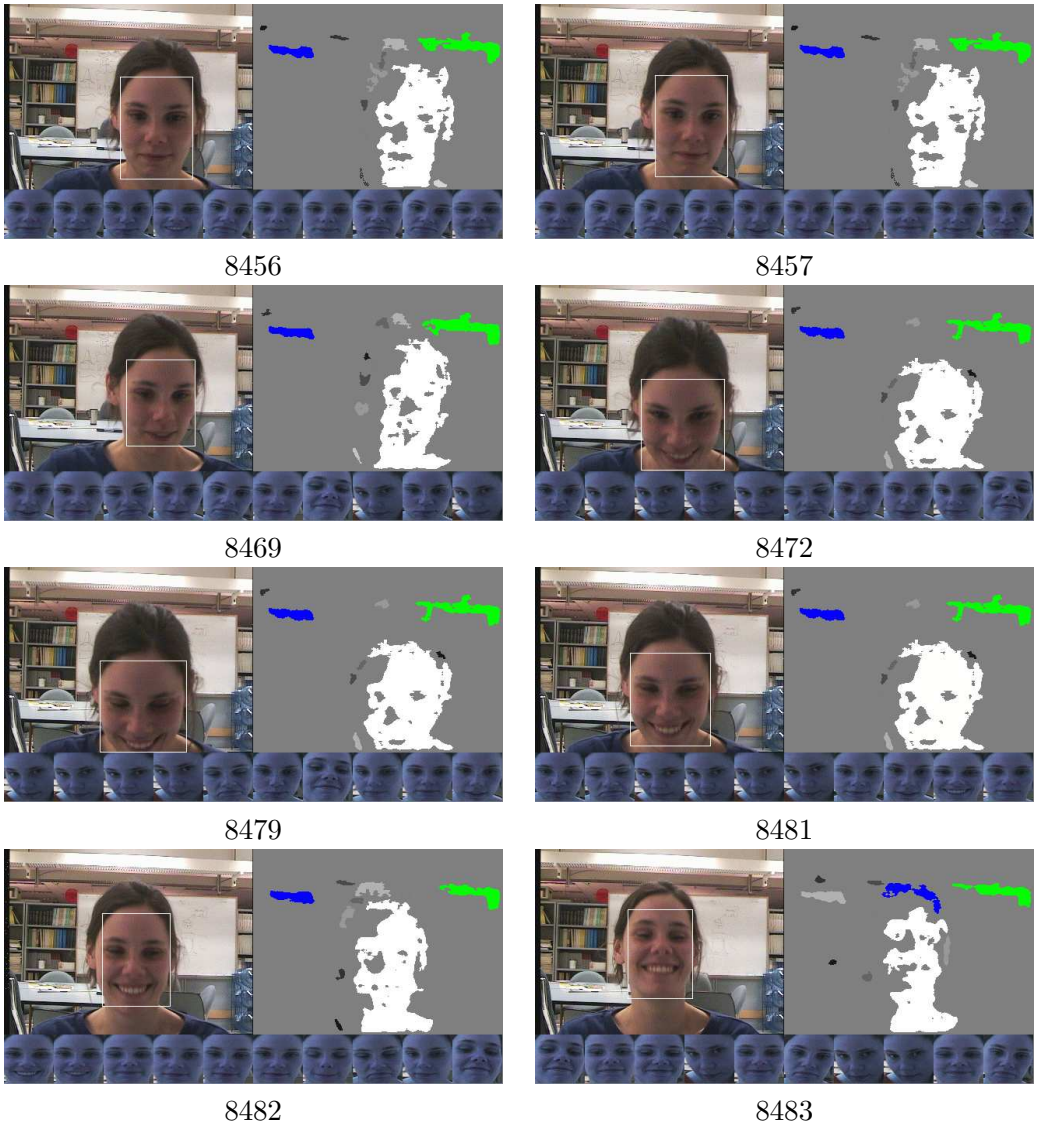


Figure 3.19: Tracker successful operation. Top left of each frame is the image and the tracked region. Top right is the skin color segmentation connected components. Different shades of grey denote different components. Exemplars are shown along the bottom, ranked by increasing match scores from left to right (best exemplar on the left). The frame rate in this sequence is 12 fps. The tracker is operating normally at this point. Re-initialization occurs at frames 8469, 8482 and 8483 due to large motions of the head. The track holds successfully, as exemplars in the database are found that closely match the current pose. However, the tracker is about to fail at frame 8484, shown in Figure 3.20.

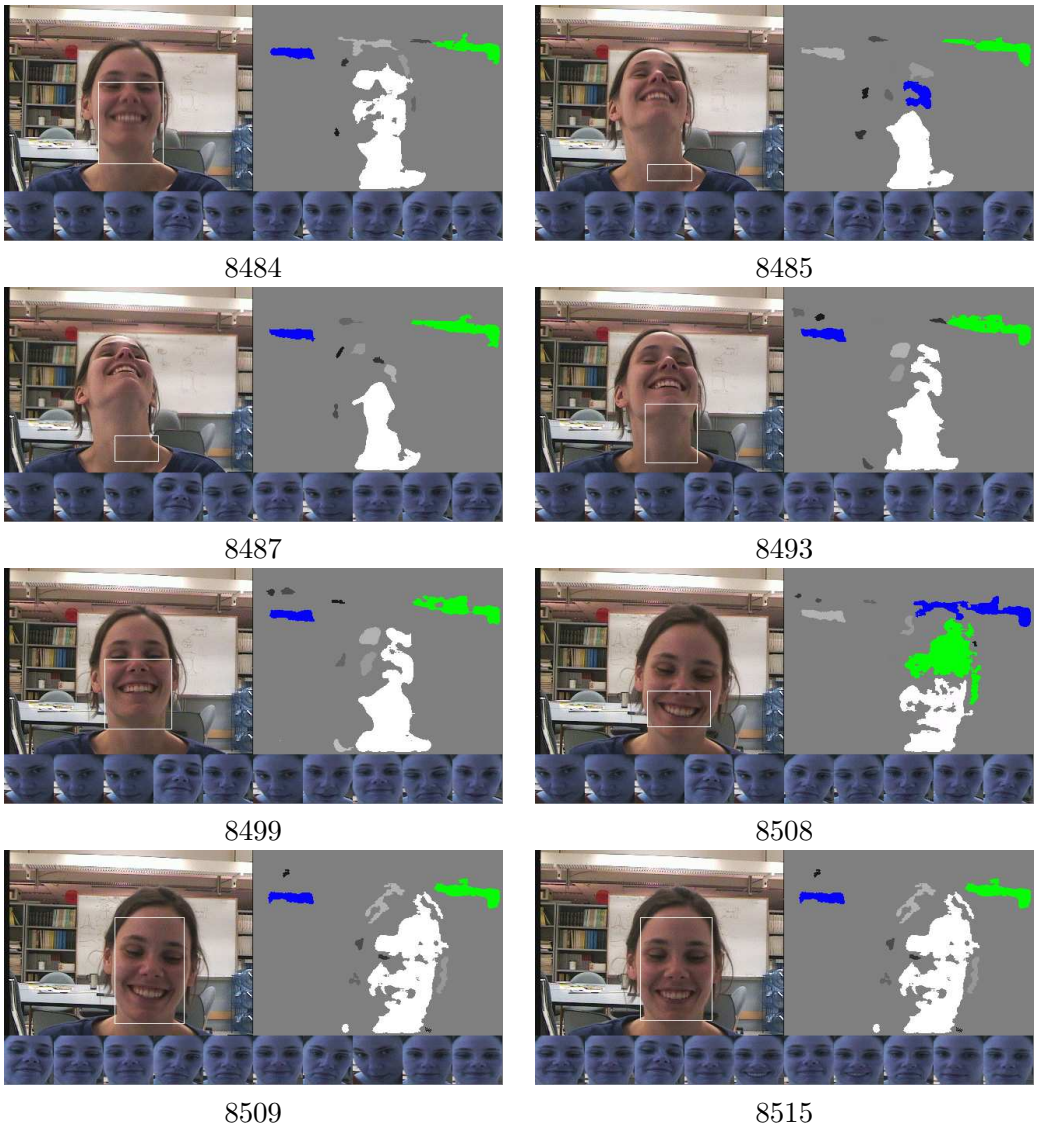


Figure 3.20: Continued from Figure 3.19, the tracker begins to fail at frame 8484 due to a large motion upwards of the face. By the following frame (8485) it has completely failed, and is tracking the subject's neck, because no appropriate exemplar was found in the database of this face pose. However, after the subject's face returns to a frontal pose at frame 8499, the tracker only needs 10 more frames to lock back onto the face and find matching exemplars.

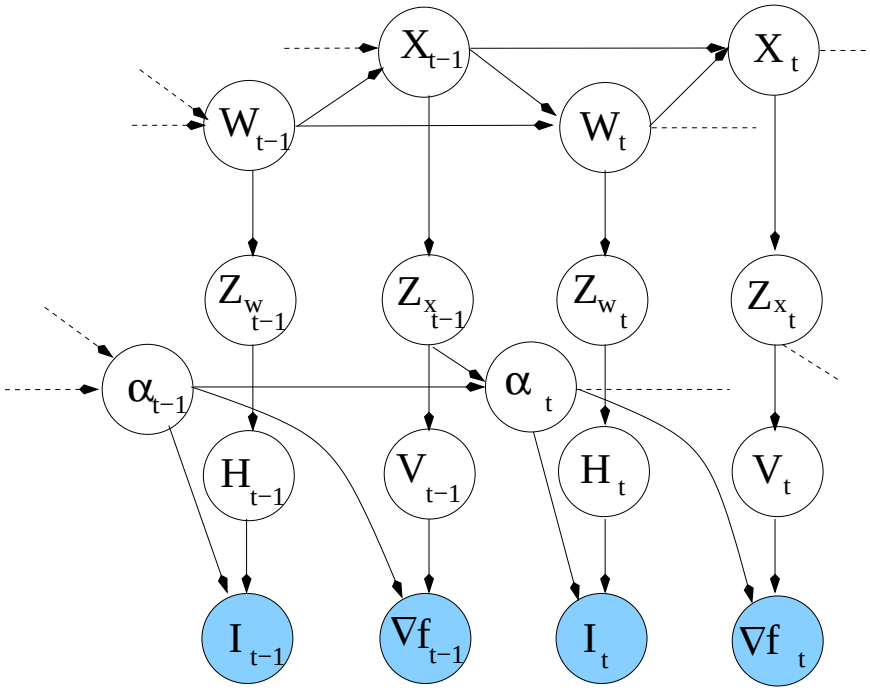


Figure 3.21: Dynamic Bayesian network for full recognition and tracking model.

that, $P(\nabla f|X\alpha) = P(\nabla f_\alpha|X)$ and $P(I|W\alpha) = P(I_\alpha|W)$, where ∇f_α and I_α are the regions of ∇f and I delineated by α. In the full Bayesian model shown in Figure 3.21, this is no longer the case, and the scales must be explicitly included, and integrated out. Given a scale, α, the likelihood functions are computed as they were shown in Sections 3.2 and 3.3.

The Zernike projections of the optical flow field, Z_x, are inputs to the tranformation process: they condition the scale. Recall that the flow field projections, Z_x, are computed from the spatio-temporal image derivatives between the current and previous frames, I_t and I_{t-1}. These projections give us evidence about where the tracked region is at time t, given where it was at time $t-1$: $\alpha_t = \alpha_{t-1} + z_t$. The flow-based updates are not exact, however, and we model deviations from them with additive, zero-mean Gaussian noise, $\alpha_t = \alpha_{t-1} + \tilde{z}_{t-1} + n_\alpha$, where $n_\alpha \sim \mathcal{N}(0, \Lambda_\alpha)$. Therefore, we can write

$$P(\alpha_t|\alpha_{t-1}\tilde{z}_{t-1}) = \mathcal{N}(\alpha_t; \alpha_{t-1} + \tilde{z}_t, \Lambda_\alpha) \tag{3.39}$$

This noise will cause significant drift in the track, as noted in the last section.

The tracking problem is to estimate the expected scale, $\tilde{\alpha}_t$, at time t given

all previous observations, $\{\mathbf{O}\}_{1,t-1}$, and the current image, I_t:

$$\tilde{\alpha}_t = \sum_{\alpha_t} \alpha_t P(\alpha_t | \{\mathbf{O}\}_{1,t-1}, I_t) \tag{3.40}$$

Using Bayes' rule we can write a recursive update for the distribution over α_t:

$$\begin{aligned} & P(\alpha_t | I_t \{\mathbf{O}\}_{1,t-1}) \qquad (3.41) \\ &= \sum_{W_t X_{t-1}} P(\alpha_t W_t X_{t-1} | I_t \{\mathbf{O}\}_{1,t-1}) \\ &= \sum_{W_t X_{t-1}} P(I_t | \alpha_t W_t) P(\alpha_t W_t X_{t-1} | \{\mathbf{O}\}_{1,t-1}) \\ &= \sum_{W_t} P(I_t | \alpha_t W_t) \sum_{\alpha_{t-1} X_{t-1}} \xi(\alpha_t, \nabla f_{t-1}) \sum_{W_{t-1}} \Theta_W \sum_{X_{t-2}} \Theta_X P(X_{t-2} W_{t-1} \alpha_{t-1} | I_{t-1} \{\mathbf{O}\}_{1,t-2}) \qquad (3.42) \end{aligned}$$

where

$$\xi(\alpha_t, \nabla f_{t-1}) = \int_{z_{x,t-1}} P(\alpha_t | \alpha_{t-1} z_{x,t-1}) P(\nabla f_{t-1} | \alpha_{t-1} z_{x,t-1}) P(z_{x,t-1} | X_{t-1})$$

We show in Appendix B that this integration over $z_{x,t-1}$ can be performed analytically, giving

$$\xi(\alpha_t, \nabla f_{t-1}) = \mathcal{N}(\alpha_t; \tilde{\mu}_\alpha, \tilde{\Lambda}_\alpha) e^\gamma \tag{3.43}$$

where

$$\begin{aligned} \tilde{\mu}_\alpha &= \alpha_{t-1} + \tilde{\mu}_{z,x} \\ \tilde{\Lambda}_\alpha &= \Lambda_\alpha + \tilde{\Lambda}_{z,x} \qquad (3.44) \\ \gamma &\propto \tilde{\mu}'_\alpha \tilde{\Lambda}_\alpha^{-1} \tilde{\mu}_\alpha - f'_\tau A^{-1} f_\tau + w' \Lambda_w^{-1} w - \mu'_{z,x} \Lambda_{z,x}^{-1} \mu_{z,x} \qquad (3.45) \end{aligned}$$

As we expect, the mean scale is given by the updates based on the mean Zernike basis projection as given by Equation 3.20, and the covariance on the scale is the sum of the noises from the estimation of z, and from the transformation process.

When the tracker re-initializes after failure, there is no prior evidence in the transformation process, but there is new evidence given by the skin segmentation algorithm, as described in the last section. This is in the form of a bounding box, b_0, which is related to the scale for each exemplar through the skin-scale mapping, M^s. In this case, the expected scale is

$$\tilde{\alpha}_0 = \sum_{\alpha_0} \sum_{k,i} \alpha_0 P(I_0 | \alpha_0, h_k) P(b_0 | h_k, \alpha_0) P(h_k | c_i) P(c_i)$$

Although it is possible to incorporate the skin segmentation evidence at every frame (not just the initial one), the exemplar matches are far more accurate, rendering the skin evidence only useful at the initial frame.

Due to the complex dependence of the image and image derivative regions on the scale, the double sums over scales, α_t, α_{t-1}, in Equation (3.42) cannot be performed analytically, and will make the updates very inefficient. Therefore, we need some way to approximate these updates. Note that any terms in the sums with α_t far (in terms of $\tilde{\Lambda}_\alpha$) from the $\tilde{\mu}_\alpha$ will be small, and can be neglected without significantly affecting tracking accuracy. Therefore, we can approximate the scale updates using a particle filter, and we can expect the number of particles necessary for accurate tracking will be small. The particle filter implements Equation (3.42) as follows, starting from an initial set of particles and their weights at time $t-1$, $\{\alpha^i_{t-1}, a^i_{t-1}\}\ \ i \in 1 \ldots N_p$:

1. re-sample the particles according to their weights, a^i_{t-1}
2. project each particle to time t according to the deterministic dynamics $\alpha^i_t = \alpha^i_{t-1} + \tilde{z}(\alpha^i_{t-1})$
3. add noise $\tilde{\Lambda}_\alpha$ by moving each particle by an amount randomly sampled from $\mathcal{N}(\alpha; 0, \tilde{\Lambda}_\alpha)$
4. compute new weights, a^i_t using Equation (3.43).

The complexity of this procedure is multi-linear in the size of the image region being tracked and on the number of particles necessary for maintaining and accurate track. Typically, only a small number of particles are needed for accurate tracking over long periods. The tracker we described in Section 3.7.1 uses only a single particle, but makes it *active* [Jen02]. To make a particle *active*, we simply add a step in which the particle locally performs gradient descent in the likelihood space.

Chapter 4

Learning Models of Facial Displays

The preceding chapter described a method for representing flow fields and poses over a discrete set of normal distributions. It showed how to combine the modeling of facial dynamics and configurations in a temporal model, and how to build temporal abstractions of the sequences, leading to descriptions of entire sequences of motions and poses over extended periods of time. It also demonstrated high level observable context can be included. In this chapter, we show how to learn the parameters of these models from data.

Our models fall into the general class of Bayesian mixture models with hidden state. We make some noisy observations of the world, which we hypothesise arise from some number of discrete causes. We assume that the causes *generate* the observations through some process, and that this process is corrupted by noise. We also assume that the observations are generated in a multi-stage process, during which temporally and spatially more complex states are generated. The last chapter described how to compute the likelihood of a set of observations given such a model. This likelihood can be used, along with Bayes' rule, to estimate the distribution over the high-level discrete causes given the observations. The current chapter addresses the more difficult problem of estimating the most likely model which could have given rise to our entire set of observations. This problem can be formulated as a constrained optimization of the likelihood of the observations given the model, over the constrained model parameters. We use the well-known *expectation-maximization*, or EM, algorithm to find a locally optimal solution [DNR77]. If we can find a good initialization to the model, then the EM algorithm will optimize this model locally. Section 4.1 is a general presentation of the EM algorithm, including a simple proof of convergence. Section 4.2 then shows a particular application of the EM algo-

rithm to the simple mixture model over flow fields discussed in Section 3.2. Only the major results are presented, the details are shown in Appendix C. Section 4.4 then discusses how to learn the parameters of, including dynamics, configuration and transformation chains, as described in Section 3.7. The learning equations are very similar as those for a hidden Markov model[Rab89], but include propagation of information forwards and backwards through all three Markovian chains, and use the likelihood functions for flow fields and configurations as described in Chapter 3. Appendix D derives the equations in full.

Sections 4.4, 4.5, and 4.6 describe learning the parameters of the more complex models which were presented in Section 3.5. These last two models will be the primary models for analysing the experimental data in Chapter 6. Section 4.9 discusses initialization techniques for the EM algorithm applied to our models. The chapter concludes with a discussion of the implementation of the learning algorithms in software.

4.1 Expectation Maximization

The expectation-maximization (EM) algorithm is a statistical technique for estimating the maximum likelihood values of model parameters in cases where there is missing or incomplete data. It addresses the situation where we have observed some set of data, $\mathbf{y}$, but not the values of some other, unobserved data, $\mathbf{X}$. The random variable X is related to Y through some model or function parametrized by θ. We additionally may have some constraints on the values of θ. Thus, we wish to find the maximum *a-posteriori* (MAP) values of the parameters, θ^*. That is, we wish to solve

$$\theta^* = \arg\max_\theta P(\mathbf{y}|\theta)P(\theta) = \arg\max_\theta \sum_{\mathbf{X}} P(\mathbf{y}, \mathbf{X}|\theta)P(\theta),$$

subject to the constraints on θ. Since $P(\mathbf{y}, \mathbf{X}|\theta)$ is typically a product over all observations in $\mathbf{y}$, the expression above is usually hard to evaluate since it involves a large sum of a large product. The EM algorithm simplifies this problem by allowing us to work with the logarithm of the product, leading to a double sum which can be manipulated to yield tractable solutions. The EM algorithm operates by first computing the expected values of the hidden variables, given the observations, the current guess at the model parameters, and the model constraints. This essentially *fills in* the hidden data, and is the *expectation* step of EM. The *maximization* step of EM then follows by updating the model parameters from the (now complete) data by counting over expectations. This description is slightly simplistic, because the *filling in* of the data is done probabilistically, so each data has a certain probability

of being generated by each hidden state. The update is not simple counting, but rather a constrained optimization of the expected complete data likelihood over the model parameters.
To derive the EM algorithm, we use the fact that $\sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y},\theta') = 1$, so that

$$\begin{aligned} P(\mathbf{y}|\theta) &= P(\mathbf{y}|\theta)^{\sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y},\theta')} \\ &= \prod_{\mathbf{X}} P(\mathbf{y}|\theta)^{P(\mathbf{X}|\mathbf{y},\theta')} \\ &= \prod_{\mathbf{X}} \left(\frac{P(\mathbf{y},\mathbf{X}|\theta)}{P(\mathbf{X}|\mathbf{y},\theta)} \right)^{P(\mathbf{X}|\mathbf{y},\theta')} \\ &= \exp\left\{ \sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y}\theta') \log\left(\frac{P(\mathbf{y},\mathbf{X}|\theta)}{P(\mathbf{X}|\mathbf{y},\theta)} \right) \right\} \end{aligned} \tag{4.1}$$

Taking the logarithm of both sides, we obtain:

$$\log \sum_{\mathbf{X}} P(\mathbf{y}\mathbf{X}|\theta) = \sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y}\theta') \log P(\mathbf{y},\mathbf{X}|\theta) - \sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y},\theta') \log P(\mathbf{X}|\mathbf{y},\theta). \tag{4.2}$$

If we define

$$\begin{aligned} L(\theta) &= \log \sum_{\mathbf{X}} P(\mathbf{y}\mathbf{X}|\theta) \\ Q(\theta|\theta') &= \sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y}\theta') \log P(\mathbf{y},\mathbf{X}|\theta) \\ &= E_{P(\mathbf{X}|\mathbf{y}\theta')} \left[\log P(\mathbf{y},\mathbf{X}|\theta) \right] \\ H(\theta|\theta') &= \sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y},\theta') \log P(\mathbf{X}|\mathbf{y},\theta) \\ &= E_{P(\mathbf{X}|\mathbf{y},\theta')} \left[\log P(\mathbf{X}|\mathbf{y},\theta) \right] \end{aligned}$$

so that

$$L(\theta) = Q(\theta|\theta') - H(\theta|\theta')$$

Now, suppose that we arbitrarily choose some value for θ', and then find the values of θ that maximize $Q(\theta|\theta')$ (using whatever method we want). We can show that the log likelihood of the observations with this new set of parameters will always be greater than or equal to the log likelihood of the observations with the old set of parameters, θ': if $\theta'' = \arg\max_\theta Q(\theta|\theta')$, then $L(\theta'') \geq L(\theta')$. To see why, notice that by definition of our maximization over θ, $Q(\theta''|\theta') \geq Q(\theta'|\theta')$. Further, an application of Jensen's inequality gives (see proof in Appendix A)

$$H(\theta''|\theta') \leq H(\theta'|\theta'). \tag{4.3}$$

Therefore,

$$L(\theta'') - L(\theta') = Q(\theta''|\theta') - Q(\theta'|\theta') + H(\theta'|\theta') - H(\theta''|\theta') \geq 0.$$

If, instead of maximizing $Q(\theta|\theta')$ (which is the maximum likelihood, or ML estimates), we maximize $Q(\theta|\theta') + \log P(\theta)$, the same conclusion holds, but we obtain the MAP estimates. Given a value for the model parameters, θ', the following procedure is guaranteed to find a new value of θ, θ'', with higher log posterior, $L(\theta'') + \log P(\theta'')$:

Procedure `EM_iterate`$(\theta', \mathbf{y})$
1. compute $P(\mathbf{X}|\mathbf{y}, \theta')$
2. find $\theta'' = \arg\max_\theta Q(\theta|\theta') + \log P(\theta)$.

return θ''

The expectation maximization algorithm can now be defined as

Procedure `EM`$(\theta, \mathbf{y})$

do
1. set $\theta' \leftarrow \theta$
2. set $\theta \leftarrow$ `EM_iterate`$(\theta', \mathbf{y})$

while $L(\theta) - L(\theta') + \log \frac{P(\theta)}{P(\theta')} > \epsilon$

return θ

Since the likelihood is guaranteed to increase at each step, the `EM` procedure is guaranteed to converge to the set of parameters, θ, at which $L(\theta)$ has a local maximium (ignoring the possibility of saddle points).

The EM algorithm works well because $Q(\theta|\theta')$ and $P(\mathbf{X}|\mathbf{y}, \theta')$ are easy to maximize over, whereas $L(\theta)$ is hard. Notice that we could replace the function $P(\mathbf{X}|\mathbf{y}, \theta')$ in the above derivation with *any* proper probability distribution over $\mathbf{X}$. In fact, this function need only be chosen so that it increases the log likelihood at each iteration of the EM algorithm. The resulting algorithm is a *generalized* EM (GEM) algorithm, which can lead to incremental versions of the algorithm [NH93]. However, the function that gives the optimal increase is the one we have selected above. We do not discuss GEM algorithms further here, although our results could be easily generalized to allow for incremental variants. The following sections derive implementations of the EM alogrithm for specific models.

4.2 Clustering Individual Flow Fields

The model presented in Section 3.2 is a mixture of Gaussians, with output distributions over the space of Zernike polynomial projections of optical flow fields. The

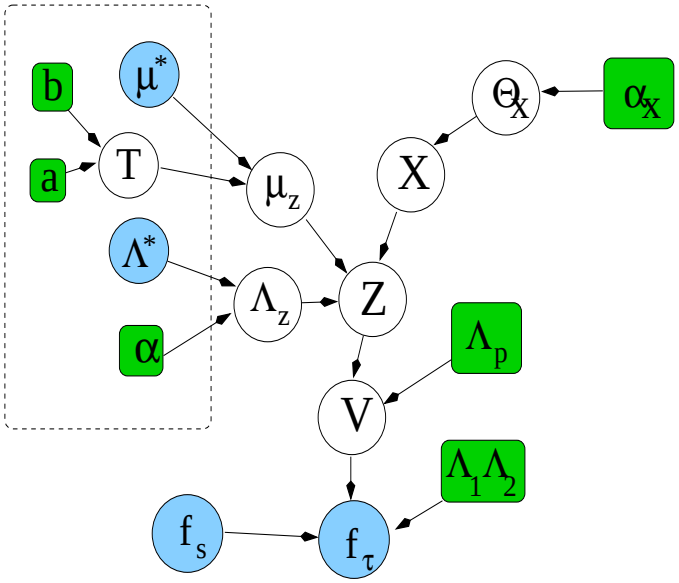


Figure 4.1: Bayesian network for the mixture of Gaussians over optical flow fields with feature weighting. Repeat of Figure 3.3.

Bayesian network, originally shown in Figure 3.3, is repeated in Figure 4.1. To learn the parameters of this model using the expectation-maximization (EM) algorithm, we optimize $Q(\theta|\theta')$ over the model parameters, θ, subject to constraints on θ. In this case, the observations, $\mathbf{y}$ are the spatio-temporal image derivatives, $\nabla\mathbf{f}$. Since we are attempting to learn the distribution over the Zernike vector space, it will be convenient to explicitly include the integration over this space in the Q function. Since the observations in this model are independent, the conditional expectation is written

$$P(\mathbf{X}|\nabla\mathbf{f},\theta') = \prod_{k=1}^{N_t} \int_{z_k} P(X_k z_k|\nabla f_k,\theta') \tag{4.4}$$

where N_t is the number of data points, which is computed using Equations (3.17) and (3.19). The complete data posterior is

$$P(\mathbf{X}\nabla\mathbf{f}|\theta) = \int_{\mathbf{Z}} P(\nabla\mathbf{f}\mathbf{X}\mathbf{Z}|\Theta) = \prod_{k=1}^{N_t} P(\nabla f_k|Z_k\Theta)P(Z_k|X_k\Theta)P(X_k|\Theta)$$

Thus, we are trying to find Θ^*

$$\Theta^* = \arg\max_{\Theta} \left[\sum_{\mathbf{X}} \int_{\mathbf{Z}} P(\mathbf{X}\mathbf{Z}|\nabla\mathbf{f},\theta') \log P(\nabla\mathbf{f}\mathbf{X}\mathbf{Z}|\Theta) + \log P(\Theta)\right] \tag{4.5}$$

over the model parameters, Θ, where θ' are the current estimates.

The complete set of parameters is $\Theta = \{\mu_z, \Lambda_z, \Theta_x, \tau_k, \Lambda_1, \Lambda_2, \Lambda_p, \alpha, a, b\}$. While $\{\mu_z, \Lambda_z, \Theta_x, \tau_k\}$ are learned from data, $\{\Lambda_1, \Lambda_2, \Lambda_p, \alpha, a, b\}$, are fixed. Recall, the bold face variables indicate sets of variables, $\mathbf{X} = \{X_1, ...X_{N_t}\}$, where N_t is the number of flow fields, and similarly for $\mathbf{Z}$ and $\nabla\mathbf{f}$. The number of dynamics and configuration states, N_x, must still be chosen. A method for doing so automatically is discussed in Section 5.3. The EM algorithm alternates between "E" and "M" steps until the increase in the log-posterior becomes smaller than some convergence threshold. The expectation, or "E", step of the EM algorithm is the calculation of the posterior according to Equation (4.4), using the current model parameters, Θ'. The "M" step is then to solve Equation (4.5) for the most likely model parameters, Θ^*. Analytical expressions for these updates can be found by taking derivatives, setting to zero and solving subject to the parameter constraints. The update equations differ from those for a standard mixture of Gaussians with feature weighting [CdFGT03] because of the integrations over $\mathbf{Z}$. To derive the EM update equations, we only perform the integrations at the end. The update equations for each parameter are derived by setting the derivative of the argument in Equation (4.5) with respect to that parameter to zero, and solving for the parameter. The derivations are given in Appendix C, here we present the update results for the mean, $\mu_{z,i}$, the covariance, $\Lambda_{z,i}$, the feature weights, T, and the class probability, Θ_X.

$$\mu_{z,i} = (\xi_{\cdot,i}\Lambda_{z,i}^{-1} + T^{-1})^{-1}\left[\Lambda_{z,i}^{-1}\left(\sum_{k=1}^{N_t}\tilde{\mu}_{z,x}\xi_{k,i}\right) + T^{-1}\mu^*\right] \tag{4.6}$$

$$\tau_k^2 = \frac{b}{a + N_x/2 + 1} + \frac{1}{2a + N_x + 2}\sum_{i=1}^{N_x}(\mu_{z,i,k} - \mu_k^*)^2 \tag{4.7}$$

$$\Lambda_{z,i} = \frac{\sum_{k=1}^{N_t}[\tilde{\Lambda}_{z,x} + (\tilde{\mu}_{z,i} - \mu_{z,i})(\tilde{\mu}_{z,i} - \mu_{z,i})']\xi_{k,i} + \alpha\Lambda^*}{\xi_{\cdot,i} + \alpha + N_z + 1} \tag{4.8}$$

$$\Theta_{x,i} = \frac{\alpha_i + \sum_{k=1}^{N_t}\xi_{k,i}}{\alpha_i + \sum_{i=1}^{N_x}\xi_{\cdot,i}} \tag{4.9}$$

where $\xi_{k,i} = P(X_{k,i}|\nabla f_k)$, $\xi_{\cdot,i} = \sum_{k=1}^{N_t}\xi_{k,i}$, and N_z is the number of features. Thus, the most likely mean for each state x is the weighted sum of the most likely values of z as given by Equation (3.25). Dimensions of the means, $\mu_{z,i}$, with small feature weights, τ_k^2, will be biased toward the data mean, μ^*, in that dimension. This is reasonable, because such dimensions are not relevant for clustering, and so should be the same for any cluster, X. The updates to the feature weights (Equation 4.7) show that those dimensions, k, with $\mu_{z,i,k}$ very different from the data mean, μ_k^*, across all

states, will receive large values of τ_k^2, while those with $\mu_{z,i,k} \sim \mu_k^*$ will receive small values of τ_k^2. Intuitively, the dimensions along which the data is well separated (large inter-class distance) will be weighted more. The covariance is updated based on a combination of the most likely covariance, $\tilde{\Lambda}_{z,i}$ and the covariance of the most likely mean, $\tilde{\mu}_{z,i}$, about the model mean $\mu_{z,i}$, for each data point, $k = 1 \ldots N_t$, weighted by the probability of state i given the data (Equation 4.8). The prior covariance is represented with $\alpha\Lambda^*$, which stabilizes the updates, avoiding matrix singularity problems in the inverses in Equation (3.21). The updates to the distribution over X are simply the summed weights for each datum given each state of X.

4.3 Clustering Individual Poses

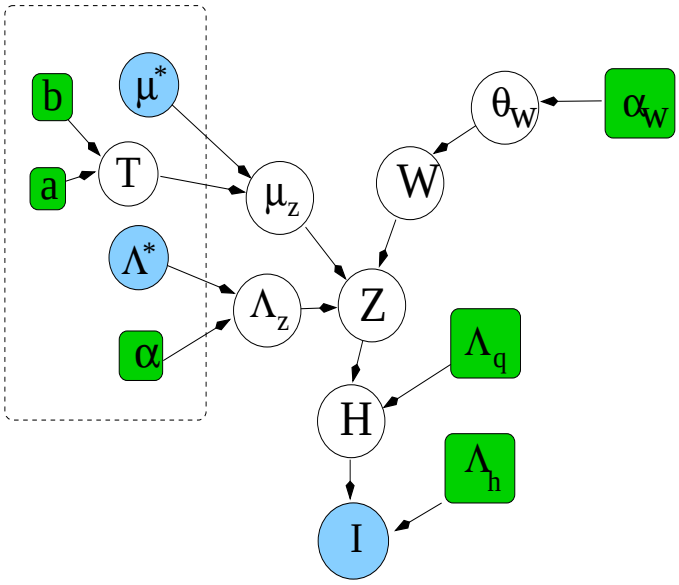


Figure 4.2: Bayesian network for the mixture of Gaussians over images with feature weighting. Repeat of Figure 3.9.

The mixture model over facial poses described in Section 3.3 is very similar to the mixture of flow fields. The Bayesian network, originally shown in Figure 3.9, is repeated in Figure 4.2. The difference is that it is no longer necessary to integrate over the Zernike feature vector. In this case, the update equations are the same as in [CdFGT03]. We are trying to find the parameters Θ^* that maximize

$$\Theta^* = \arg\max_{\Theta} \left[\sum_{\mathbf{W}} P(\mathbf{W}|\mathbf{I}, \theta') \log P(\mathbf{I}\mathbf{W}|\Theta) + \log P(\Theta) \right] \tag{4.10}$$

Note that we are re-using many of the symbols here as in the last section. This should not cause confusion, as we will make the difference clear when necessary. The update equations are as follows:

$$\mu_{z,i} = (\xi_{\cdot,i}\Lambda_{z,i}^{-1} + T^{-1})^{-1}\left[\Lambda_{z,i}^{-1}\left(\sum_{k=1}^{N_t} z_w \xi_{k,i}\right) + T^{-1}\mu^*\right] \tag{4.11}$$

$$\tau_k^2 = \frac{b}{a + N_w/2 + 1} + \frac{1}{2a + N_w + 2}\sum_{i=1}^{N_w}(\mu_{z,i,k} - \mu_k^*)^2 \tag{4.12}$$

$$\Lambda_{z,i} = \frac{\sum_{k=1}^{N_t}[(z_w - \mu_{z,i})(z_w - \mu_{z,i})']\xi_{k,i} + \alpha\Lambda^*}{\xi_{\cdot,i} + \alpha + N_z + 1} \tag{4.13}$$

$$\Theta_{w,i} = \frac{\alpha_i + \sum_{k=1}^{N_t}\xi_{k,i}}{\alpha_i + \sum_{i=1}^{N_w}\xi_{\cdot,i}} \tag{4.14}$$

where $\xi_{k,i} = P(W_{k,i}|I_k)$, $\xi_{\cdot,i} = \sum_{k=1}^{N_t}\xi_{k,i}$, and N_w is the number of states of W.

4.4 Learning Temporal Models

As we described in Section 3.5, the addition of temporal continuity to the basic Gaussian mixture models leads to two temporal Markov chains: the *dynamics* and *configuration* processes. The Bayesian network, originally shown in Figure 3.12, is repeated in Figure 4.3. The optimization we are trying to perform is to find Θ^*

$$\Theta^* = \arg\max_{\Theta}\left[\sum_{\mathbf{Y}} P(\mathbf{Y}|\mathbf{O},\theta')\log P(\mathbf{O},\mathbf{Y}|\Theta) + \log P(\Theta)\right] \tag{4.15}$$

Recall that $Y \equiv \{X, W\}$ is the joint space of the mixture variables and $O \equiv \{I, \nabla f\}$ is the joint space of observations. The expectation step of the EM algorithm is to evaluate $P(\mathbf{Y}|\mathbf{O},\theta')$, which is slightly more complex now, as $\mathbf{Y}$ and $\mathbf{O}$ are sets of variables spanning some time interval. Thus, estimating this quantity involves propagating information forwards and backwards through both temporal chains. Appendix D derives the EM algorithm for this model. The resulting update equations are similar to those for a simple hidden Markov model [Rab89], but involve forwards and backwards variables over both dynamics and configuration chains. The updates for the means and covariances of the output distributions in the dynamics and configuration chains are the same as those derived in the last two sections. The only difference is in the computation of $\xi_{k,i}$, which in this case is conditioned on the data for the entire sequence, $\nabla\mathbf{f}$ or $\mathbf{I}$, not only that for the current frame, k. That is, for the dynamics chain, we have

$$\xi_{k,i} = P(X_{k,i}|\nabla\mathbf{f}\Theta').$$

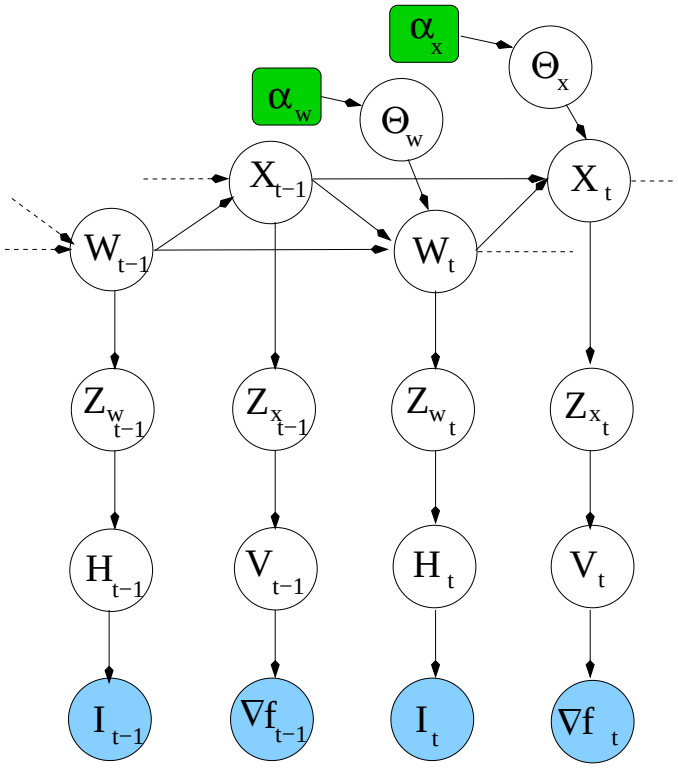


Figure 4.3: Two time slices of a dynamic Bayesian network (DBN) for simultaneous modeling of pose and dynamics. Repeat of Figure 3.12.

while for the configuration chain, we have

$$\xi_{k,i} = P(W_{k,i}|\mathbf{I}\Theta').$$

The *forward-backward* algorithm derived in Appendix D is used to compute this quantity, as well as the sufficient statistics for the transition parameters.

4.5 Mixtures of Hidden Markov Models (3MGs)

Learning equations for a mixture of hidden Markov models (a 3MG), as shown in Figure 4.4 (a repeat of Figure 3.13), are simple to construct once we have the update equations for the hidden Markov model components in the mixture. We are trying to find Θ^*

$$\Theta^* = \arg\max_{\Theta} \left[\sum_{\mathbf{Y},D} P(\mathbf{Y}, D|\mathbf{O}, \theta') \log P(\mathbf{O}, \mathbf{Y}, D|\Theta) + \log P(\Theta) \right]$$

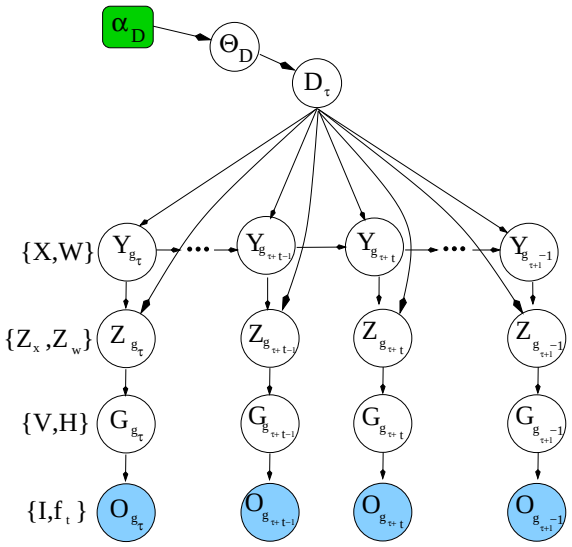


Figure 4.4: Mixture of coupled hidden Markov models as a dynamic Bayesian network. Repeat of Figure 3.13.

However, since $P(\mathbf{O}, \mathbf{Y}, D|\Theta) = P(\mathbf{O}, \mathbf{Y}|D\Theta)P(D|\Theta)$, this is

$$\Theta^* = \arg\max_{\Theta} \left[\sum_{\mathbf{Y}, D} P(\mathbf{Y}|D, \mathbf{O}, \theta') P(D|\mathbf{O}, \theta') \log P(\mathbf{O}, \mathbf{Y}|D\Theta) + \right. \\ \left. + \sum_{D} P(D|\mathbf{O}, \theta') \log P(D|\Theta) + \log P(\Theta) \right] \tag{4.16}$$

The second term in this expression only varies in the optimization over the mixture weight parameter, Θ_D, while the first does not vary over Θ_D. Thus, when optimizing this expression with respect to Θ_D, only the second and last terms are involved. On the other hand, when optimizing with respect to all other parameters in the model, only the first and last terms are involved. Therefore, the update to the mixture weight parameter is similar to those for any mixture model (e.g. Equation 4.9):

$$\Theta_{D,i} = \frac{\alpha_{D,i} + \sum_{k=1}^{N_s} \xi_{k,i}}{\alpha_{D,i} + \sum_{i=1}^{N_x} \xi_{\cdot,i}}$$

where in this case, $\xi_{k,i} = P(D_{k,i}|\mathbf{O}_k)$, and N_s is the number of training sequences. The updates to the component hidden Markov models are then weighted by the posterior probability over the mixture weights, $P(D_k|\mathbf{O})$, which is

$$P(D_k|\mathbf{O}) = \frac{P(\mathbf{O}|D_k)\Theta_{D,k}}{\sum_{D_k} P(\mathbf{O}|D_k)\Theta_{D,k}}$$

where Θ_D are the mixture weight parameters, and $P(\mathbf{O}|D_k)$ is computed from Equation 3.34.

4.6 Context Dependent Mixtures of Hidden Markov Models (C3MGs)

The addition of the observable context changes the learning slightly. We are trying to find Θ^*

$$\Theta^* = \arg\max_{\Theta} \left[\sum_{\mathbf{Y}D} P(\mathbf{Y}D|\mathbf{O}CA\theta') \log P(\mathbf{O}\mathbf{Y}DCA|\Theta) + \log P(\Theta) \right]$$

However, since $P(\mathbf{O}\mathbf{Y}DCA|\Theta) = P(\mathbf{O}\mathbf{Y}AD|C\Theta)P(C|\Theta)$, we again obtain two terms

$$\begin{aligned} \Theta^* = \arg\max_{\Theta} \Bigg[& \sum_{\mathbf{Y}D} P(\mathbf{Y}D|\mathbf{O}AC\theta') \log P(\mathbf{O}\mathbf{Y}AD|C\Theta) \\ & + \sum_{D} P(D|\mathbf{O}CA\theta') \log P(C|\Theta) + \log P(\Theta) \Bigg] \end{aligned} \tag{4.17}$$

The first term in Equation 4.17 is the term that is maximized for a mixture of hidden Markov models, conditioned on the (observed) variable C. The addition of the observable A only adds an extra multiplicative factor into the equations. Maximizing the second term in Equation 4.17 updates the parameterized probability distribution $\Theta_{Dij} = P(D = i|C = j)$ by counting the expected number of times the cluster variable $D = i$ when the context variable $C = j$, N_{Dij}. That is,

$$\Theta_{Dij} = \frac{E_{P(D|\mathbf{O}CA)} N_{Dij}}{\sum_i E_{P(D|\mathbf{O}CA)} N_{Dij}}$$

where $E_{P(D|\mathbf{O}CA)} N_{Dij} = \sum_\tau P(D_\tau = i|\mathbf{O}CA)\delta(C_\tau = j)$. Thus, the update equations for the probability of the weight parameter given the context state, $\Theta_{D,ij} = P(D = i|C = j)$ becomes

$$\Theta_{D,ij} = \frac{\alpha_{D,ij} + \sum_{k|C_k=j} \xi_{k,i}}{\alpha_{D,ij} + \sum_{i=1}^{N_x} \xi_{\cdot,i}}$$

where

$$\xi_{k,i} = P(D_{k,i}|C_k A \mathbf{O}_k) = \frac{P(\mathbf{O}_k|D_{k,i})P(A|D_{k,i})P(D_{k,i}|C_k)}{\sum_{i=1}^{N_x} P(\mathbf{O}_k|D_{k,i})P(A|D_{k,i})PP(D_{k,i}|C_k)}$$

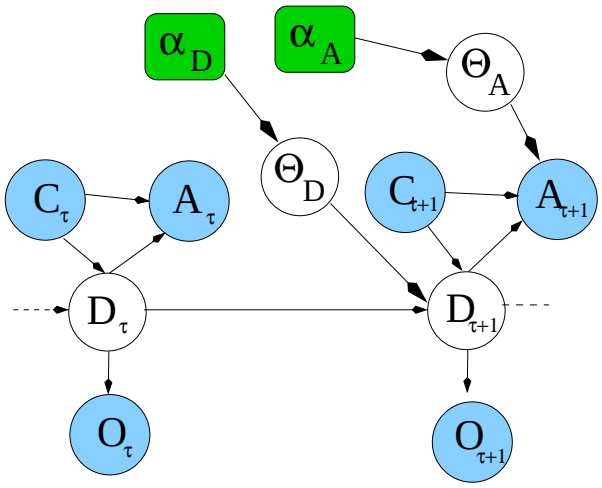


Figure 4.5: Context dependent Markov chain of mixtures of hidden Markov models (C4MG) as a dynamic Bayesian network, including context variables, C and A. Repeat of Figure 3.17.

4.7 Context Dependent Markov Chains of Mixtures of Hidden Markov Models (C4MGs)

As pointed out in Section 3.6.3, the model shown in Figure 4.5 (Repeat of Figure 3.17) can be seen as an input-output hidden Markov model [BF96]. We are trying to maximize

$$\Theta^* = \arg\max_{\Theta} \left[\sum_{\mathbf{D}} P(\mathbf{D}|\mathbf{O}, \mathbf{C}, \mathbf{A}, \theta') \log P(\mathbf{D}, \mathbf{O}, \mathbf{C}, \mathbf{A}|\Theta) + \log P(\Theta) \right]$$

where $\mathbf{O}$ are the sequences of images and derivatives, while $\mathbf{C}$ and $\mathbf{A}$ are the sequences of context variables (input and output, respectively). The update equations for the parameters are nearly the same as for a normal hidden Markov model, except for the factorization of the outputs into two sets, $\mathbf{O}$ and $\mathbf{A}$, and the additional conditioning variable, $\mathbf{C}$. While the additional outputs, $\mathbf{A}$, cause no special problems, the conditioning variable, $\mathbf{C}$, requires that we compute updates for parameters separately for each value of C, and use only those data which co-occurred with that value of C in the updates. The derivation is given in Appendix D.

4.8 Parameter list

Table 4.1 shows all the parameters in the model. The top section shows parameters that have fixed values. These numbers are set manually based on prior expectations

	Parameter	Used for	Value
fixed	σ_1, σ_2	optical flow	0.08,1.0
	$\sigma_{p,x}, \sigma_{p,w}$	projection error	0.01
	N_z	number of features	problem dependent
	α, a, b	feature weights (X chain)	N_z+2,1,0.01
	α, a, b	feature weights (W chain)	N_z+2,1,0.01
	α_x, α_w	transition dirichlet priors (CHMM)	problem dependent
	α_D	transition dirichlet prior (D)	problem dependent
	α_A	action expectation dirichlet prior	problem dependent
learned	$\mu_{z,x}, \Lambda_{z,x}, \tau_x$	Mix. of Gaussians (X chain)	Learned (EM)
	$\mu_{z,w}, \Lambda_{z,w}, \tau_w$	Mix. of Gaussians (W chain)	Learned (EM)
	Θ_X, Θ_W	Transition functions (CHMM)	Learned (EM)
	Θ_D	Transition (D chain)	Learned (EM)
	Θ_A	Action Expectation	Learned (EM)
	N_x, N_w	number of states in X and W chains	Learned (initialization)
	N_D	number of states in D chain	Learned (Sec. 5.3)

Table 4.1: List of parameters in the model. (TOP) fixed parameters (BOTTOM) learned parameters

of their values. The bottom section shows the parameters that are learned using the EM algorithm.

4.9 Initialization

The EM algorithm performs hill climbing on the likelihood surface, converging from its starting point to a local maximum. It is therefore dependent on the initial choice of the parameters. In models with many parameters as we have described, the likelihood surface can contain many local maxima. It is therefore critical to achieve good initialization, introducing as much prior knowledge about the domain before attempting a full maximization of the posterior probability of the model given the data.

Initialization of the simple mixture model (Section 4.2) with N_x classes from a set of single (independent) spatio-temporal derivative fields, $\nabla\mathbf{f}$, proceeds as follows:

1. Compute the most likely values of Z for each frame using a single zero-mean model with diagonal covariance with constant variance $\sigma_z = 0.001$.

2. Select a subset of the data points where $|Z| > \delta$, where δ is some threshold on the magnitude of the Zernike vector. Since much of our data contains very

little motion, (so $Z \sim 0$), this ensures that the clustering is not biased towards clustering on the noise in the frames where little is happening.

3. Select N_x such data points are selected randomly as the initial seeds for N_x-means (K-means with $K = N_x$) clustering with a Euclidean distance measure between values of Z.

4. Fit Gaussian distributions to the classes resulting from the N_x-means clustering.

5. Initialize all the feature weights τ_k^2 to 1.

6. Initialize the state assignment probability Θ_X evenly.

The results for the simple clustering experiments were relatively insensitive to the initialization. The mixture model over the configurations (Section 4.3) is initialized in a similar way, using the projections of image regions to the Zernike basis.

Initialization of the context dependent mixture of coupled hidden Markov models (Section 4.6) with N_d classes from a set of N_t sequences of spatio-temporal derivative fields, $\nabla\mathbf{f}$, and images, $\mathbf{I}$, proceeds as follows:

1. Perform the initialization of a mixture of Gaussians for the dynamics states, X, as described above, from the spatio-temporal derivative fields for each frame, ∇f, considering all frames to be independent. Choose N_x larger than the maximum number of classes of dynamics states we expect (see Section 5.3). Maximization of the posterior ensures that no singularity problems will be encountered.

2. Perform a similar initialization for the configuration states, W, Choose the number of W classes, N_w, larger than the maximum number of classes of configuration states we expect (see Section 5.3).

3. Classify all the data (including the data that did not pass the thresholding test, above) using the two mixture models. This gives two sets of state membership indices, $\mathbf{w}$ and $\mathbf{x}$.

4. Find the set of X states visited by each sequence.

5. Find the largest N_d sets of sequences whose sets of visited X states match exactly, irrespectively of the number of X states visited by the sequences. These N_d clusters group together the sequences which match closely at the Gaussian output level in the dynamics process. The sequences which are not in a group are only similar to a small number of other sequences, and are

neglected for the moment. This is a bootstrapping process whereby only the "best" sequences are used to initialize the model, ensuring that the results are not "washed out" by noisy data.

6. Find the set of X states visited by all the sequences in each cluster, i. This gives the number of X states for the dynamics chain of model i, N_X^i. Do the same for the W states, giving N_W^i.

7. Initialize a coupled hidden Markov model for each cluster, i, by assigning the output distributions (including feature weights, if applicable) to be those in the simple mixture models which are used by the sequences in the cluster. Initialize the transition and initial state probabilities randomly.

8. Train each coupled hidden Markov model, keeping the output distributions fixed.

9. Initialize the mixture probabilities for the mixture of coupled HMMs evenly for each context state.

Smyth [Smy97] suggests a different initialization method for mixtures of hidden Markov models, which fits a simple HMM to each individual sequence, evaluates the log-likelihood of each sequence given every simple HMM, and then clusters the sequences into K groups using the log-likelihood distance matrix. Simple HMMs are then fit to each of these K clusters, and the results are used to initialize the final mixture model. The conditional probabilities of cluster membership, D, given the context variables, C, are initialized by counting the number of observed states C in each cluster. We have experimented with this method, using agglomerative clustering with complete linkage (furthest neighbors merging). However, the resulting models tend to be significantly "washed out", and do not find clusters which are well matched with the context states. The reason is that the individual HMMs are not sufficiently well supported by the data, and tend to learn models heavily biased by the prior distributions. These biases are then propagated to the final model. We have chosen the method described above to take advantage of the good initializations that can be performed at the lowest level (the Gaussian and multinomial output distributions). This level is very important since it involves many parameters and performs the spatial abstraction step which is crucial to the efficient description of our data.

4.10 Software Implementation Details

The models we have been discussing in this chapter and in the last are implemented in software written in C++. The structure of the models lends itself easily to an elegant class structure, with a small number of base classes providing support for derived classes for each of the models we have described. We use a data structure to hold our observations which fits in with the model class structure, allowing for a simple and intuitive programming interface to the model learning algorithms.

A probability distribution, or `PDist`, is the fundamental base class. It is pure virtual, allowing all other classes to include generic `PDist`s without specific class assignment until runtime. The normal distributions which characterize continuous outputs are `Gaussian`s, derived from `PDist`. The probabilistic normal projections are encapsulated in a class, `ZernGauss`, derived from `Gaussian`. The multinomial distribution is encapsulated in a class, `Multinomial`, derived from `PDist`. The simple mixture model with N_X states, a `MG`, also derives from `PDist`, and contains an array of N_x output `PDist`s, and an $N_x \times 1$ array for the mixture weight parameter. The context-dependent mixture model, or `CMG`, derives from `MG` (and so is also a `PDist`), and adds the additional parameters necessary for describing the $P(D|C)$. The hidden Markov model with N_x states, a `HMM`, also derives from `PDist`, contains an array of N_x output `PDist`s, a $N_x \times N_x$ matrix for the transition parameter, and a $N_x \times 1$ array for the initial state parameter. The coupled hidden Markov model, or `XCHMM`, derives from the `HMM` class (and so is also a `PDist`), adds the C transition probability, and another set of output distributions, also `PDist`s.

The mixture model, or `MG` class, can be used at run time to implement a mixture of any `PDist`s. For example, a mixture of Gaussians assigns the `MG` object's N_x output `PDist`s to be `Gaussian` objects. A mixture of HMMs assigns the `MG` object's N_x output `PDist`s to be `HMM` objects. A Markov chain of mixtures of Markov chains, or 4MG, is a `HMM` with output `PDist`s as `MG` objects, each of which has output `PDist`s as `HMM` objects. Thus, we see the class structure leads to an efficient way of implementing hierarchical models of any depth. In this paper, we only use models with two layers.

The hierarchical structure of the models requires a hierarchical data structure, which is encapsulated in the `PData` class. A `PData` object can contain raw floating-point data, or an array of `PData` objects, allowing for a hierarchical structure.

The base `PDist` class defines the three major functions necessary for learning the model parameters. In the following, let X refer to the model states, and O refer to the observations.

1. `pdf` Computes the posterior distribution over the model states, X, given an observation (a *PData* object), O. It first computes the likelihood for each state X, by calling `pdf` on each of its children `PDists`. It then computes the posterior by multiplying the likelihood by the model parameters. For the mixture model, `MG`, this is a simple multiplication by the weights, $P(X)$. For a hidden Markov model, `HMM`, this involves the forwards and backwards passes.

2. `addSuffStats` Called after a call to `pdf`, with a single argument giving the weight assigned to the previous piece of data by a higher level (or 1.0 if at the highest level). This function updates the sufficient statistics for the `PDist` by adding in the data passed to `pdf` weighted by the posterior distribution computed in `pdf` and by the passed in higher level weights.

3. `update` Updates the model parameters based upon the accumulated sufficient statistics, and resets those statistics.

Thus, the EM algorithm for any `PDist` given any set of data in a `PData` object can be implemented as a member function `trainModel` as follows

```
function trainModel(PData **O)
        while (!converged)
                for each data in O
                        pdf(data)
                        addSufficientStats(1.0)
                end
                update()
        end
```

4.10.1 Numerical Scaling

The computation of the data likelihood, described in Chapter 3, involves taking the exponential of what can be a very large quantity. This causes numerical problems in an implementation. However, the exponentials are only used to compute the the sufficient statistic, which is always normalized over higher level states. This normalization allows us to get around the numerical problems with a simple scaling trick, described here for the case of a simple mixture of Gaussians, but which can be applied to any of the distributions learned.

We are trying to compute the sufficient statistic for the simple mixture of Gaussians:

$$s_i = \sum_{j=1}^{N_d} \frac{f_i e^{a_{ji}} p_i}{\sum_{i=1}^{N_x} f_i e^{a_{ji}} p_i}$$

where f_i is the normalizing factor in the Gaussian, $(2/pi)^{-d/2}|\Lambda|^{-1/2}$, a_{ji} is $-(x_j - \mu_i)'\Lambda_i^{-1}(x_j - \mu_i)$, p_i is the model state assignment parameter, N_d is the number of data, and N_x is the number of states (models). Problems occur in this calculation if the factors in the exponential, a_{ji}, are too large (greater than about 10^3), since the exponential cannot be computed. However, since we are normalizing by a sum over states, we can scale each factor by subtracting some large number which is independent of i. The largest such factor is $m_j = \max_i a_{ji}$. Thus, we can multiply both top and bottom by e^{-m_j}, giving

$$s_i = \sum_{j=1}^{N_d} \frac{f_i e^{a_{ji} - m_j} p_i}{\sum_{i=1}^{N_x} f_i e^{a_{ji} - m_j} p_i}$$

Now, all terms in the exponentials are guaranteed to be less than 0, and one is always exactly 1. When computing the log-likelihood, we must also be careful, since the same exponentials appear

$$l = \sum_{j=1}^{N_d} \log \sum_{i=1}^{N_x} f_i e^{a_{ji}} p_i$$

In this case, we multiply the exponential by $1 = e^{-m_j} e^{m_j}$, giving

$$\begin{aligned} l &= \sum_{j=1}^{N_d} \log \left[e^{m_j} \sum_{i=1}^{N_x} f_i e^{a_{ji} - m_j} p_i \right] \\ &= \sum_{j=1}^{N_d} \left[\log \left(\sum_{i=1}^{N_x} f_i e^{a_{ji} - m_j} p_i \right) + m_j \right] \end{aligned}$$

Note that this scaling trick also works well when the factors a_{ji} are really large and negative numbers, for all values of i. Such cases cause problems when computing the log likelihood, since we will be trying to take the logarithm of 0. The scaling technique ensures that the overall *smallness* of the a_{ji} factors is taken out of the sum over states, since we want to take the logarithm of it anyways.

Chapter 5

Facial Displays in Games

The previous chapter described methods for learning statistical models of temporal patterns of motion in the human face. We have demonstrated how such models can be learned from observations, and how the learned models establish classes of facial motions at different levels of temporal abstraction. We have also shown how to incorporate high-level observations of context into the models. Now we may ask: what can such models be used for? One answer is *action*. As we have seen in Chapter 2, humans use facial displays during interactions as relevant signals which may affect actions of other agents. As such, they can be considered as simply other actions in an agent's repertoire.

Suppose that some *rational* agent can describe the probabilistic relationships between classes of facial motions and the states of the world, his actions and his value system. Since facial signals are relevant, this agent must use such relationships in selecting actions which maximize its expected utility [vNM53]. In this chapter, we present the concepts of expected utility maximization using the framework of Markov decision processes, or MDPs [Put94]. MDPs are a simple, yet general, model of the interactions between an agent's actions, the state of the world, and the agent's value system. Since we are modeling observations of the state of the world using a temporally and spatially abstract set of (unobservable) states, we will, in fact, be using a partially observable Markov decision process, or POMDP [KLC98]. Further, since we are modeling the interactions between two agents, we are using multi-agent POMDPs [Gmy02].

Decision making requires an agent to consider the long-term effects of his actions upon his world model, a process which is typically much more difficult than learning models from input data. It is necessary for an agent to build temporally and spatially abstract descriptions of the world over which decision making can be tractably attempted. These descriptions are the agent's *internal state*. This chapter will show that by equating actions and internal states with *context*, the

models of spatial and temporal abstraction presented in Chapter 4 can be used to learn relationships between facial motion and actions, utilities and states. The learned dependencies are at a level of abstraction at which decision making can be realistically attempted, yielding policies of action based, in part, upon observed facial motions.

Attempting to model unconstrained human communication would be foolhardy, however, given the enormously complex social, emotional, and psychological context in which any communication takes place. Instead, in order to study the relationships between action recognition and action, we have developed an experimental paradigm in which we artificially constrain the structure of an interaction between two humans. We impose constraints as rules in a computer game the humans play. We then observe the humans playing the game, and learn models of the relationships between their facial motions and the states and actions in the game. The models are restricted cases of the general model, where the restrictions are the constraints imposed by the game. Subsequent analysis of the learned models reveals how the humans were using their faces for achieving value in the game. We can also extract policies of action from the models. The constraints imposed by the game restrict the human's degrees of freedom in the interaction, and so focus attention in the model on the aspects we are interested in. Further, the constraints simplify the structure of the problems such that solutions become more feasible.

The chapter is structured as follows. Section 5.1 begins with the concepts of a multi-agent interaction from a game-theoretic perspective, and argues that POMDPs are a useful model of such situations. This is followed by an overview of the decision-theoretic foundations, including fully and partially observable Markov decision processes, solution techniques, and how such models can be learned from data. We discuss optimal solution methods for POMDPs with discrete output spaces, and approximate methods for POMDPs with continuous output spaces. Section 5.2 presents a general model of communication with non-verbal signals in a dual-agent situation as a partially observable Markov decision process. Section 5.4 discusses the experimental paradigm we use to evaluate this model. The last three sections describe particular applications of our experimental method to three games.

Section 5.5 presents a very simple *imitation* game in which a human plays with an agent. The imitation game requires the players to use complex facial displays, and we use this game primarily to demonstrate the computer vision modeling techniques discussed in Chapter 3 and 4. We show the learned models, example sequences, and their classifications given the learned models. There are no real decisions to be made in this game, but we present classification results in a leave-one-out cross-validation experiment.

Section 5.6 describes a game in which a human controls a robot using gestures. This game is a simple decision problem, but involves a complex vision task of recognising hand gestures. It is used primarily to demonstrate that our models are not restricted to facial displays alone.

Section 5.7 shows a more complex, *card matching* game, in which two human players face off. The second game, in contrast to the first two, only calls for very simple facial displays, but has a more complex, high-level structure, involving nearly 10^5 states, six actions, and temporal look ahead. This model is used to demonstrate the learning of the decision process, and the subsequent generation of policies of action.

5.1 Decision-Theoretic Foundations

In situations involving other intelligent agents, a decision-making agent who wishes to interpret other agent's actions must be prepared to reason using game theory. Since he is operating in a multi-agent environment, he must think about the fact that the other agents are decision-makers as well, who are possibly considering strategies of action based upon their beliefs about his possible strategies of action. Of course, knowing this, he must account for the other agent's knowing that he knows about their strategic plans based on his beliefs, etc. This infinite regress is the subject of game theory, usually approached by seeking equilibria in the strategy choices of all the agents simultaneously. At such equilibria, no single rational agent would change strategies if he believed no other agent would either. The strength of the concept of equilibria in games lies in its ability to avoid explicit reasoning about infinitely nested beliefs.

However, game-theoretic solution concepts such as equilibria are not sufficient in a general conversational setting. The reason is that the payoffs to (value functions of) other agents may not be known with certainty, and so it may be difficult for an agent to figure out what other agents are planning to do. Further, there may be multiple equilibria in such settings (non-uniqueness), and agents may not act according to their equilibrium strategies (incompleteness) [Gmy02]. To surmount these problems, we consider that each agent will choose strategies of action based upon his beliefs about other agent's internal states, value functions, and strategy choices. This approach is called the *decision-analytic* approach to games [Mye91]. It avoids the difficulties of uniqueness and incompleteness of the equilibrium approach at the cost of explicitly representing the infinitely nested belief systems. That is, suppose that some agent, a, is trying to come up with a strategy in an interaction with agent b. Taking the decision-analytic approach, agent a will try to explicitly

assess agent b's strategy choices. To do so, he may put himself in agent b's shoes, and will realise that agent b is trying to assess agent a's strategy choices. Agent a will then be forced to explicitly assess what agent b's beliefs are about agent a's beliefs, etc, in an infinite regress. Nevertheless, we assume that this regress can be terminated at some (not too deeply nested) point, and the approximate solution will be sufficient for our purposes. Note that, in the games we will describe in the latter part of this chapter, these concepts do not play an important role, but will need to be addressed further for more complex interactions.

The decision-analytic approach to multi-agent games can be formalised as a partially observable Markov decision process, or POMDP [Gmy02]. A POMDP is a probabilistic temporal model of an agent interacting with the environment [KLC98], A POMDP is similar to a hidden Markov model in that it describes observations as arising from hidden states, which are linked through a Markovian chain. However, the POMDP adds actions and rewards, allowing for decision-theoretic planning. In a multi-agent setting, the states of the POMDP include complete POMDP models of other agents. That is, each decision-analytic agent will explicitly represent the POMDP models for each other agent. These types of models have been called *interactive POMDP*s or *I-POMDP*s [Gmy02]. Since we wish to focus on the use of non-verbal displays in games, we will use a restricted form of I-POMDPs, in which much of the state space is assumed to be observable. However, our models should integrate seamlessly with I-POMDPs once the assumptions are relaxed.

5.1.1 Fully Observable MDPs

Markov decision processes (MDPs) have become the semantic model of choice for decision-theoretic planning (DTP) in the AI planning community. Fully-observable MDPs [Bel57, Put94] model the domain of interest with a finite set of (fully observable) states $\mathcal{S}$. Actions of an agent, drawn from a finite set $\mathcal{A}$, induce stochastic state transitions, with $P(S_t|A_t, S_{t-1})$ denoting the probability with which state $S_t \in \mathcal{S}$ is reached when action $A_t \in \mathcal{A}$ is executed at state $S_{t-1} \in \mathcal{S}$. A real-valued reward function R, associates with each state, s, and action, a, its immediate utility $R(s, a)$. Figure 5.1 shows the Bayesian network representation of an MDP.

A *stationary policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ describes a particular course of action to be adopted by an agent, with $\pi(s)$ denoting the action to be taken in state s. If we assume that the agent acts indefinitely (an infinite horizon). The stationarity of a policy means that it will not change over the course of time. We compare different policies by adopting an *expected total discounted reward* as our optimality criterion wherein future rewards are discounted at a rate $0 \leq \beta < 1$, and the value of a policy is given by the expected total discounted reward accrued. The expected value $V_\pi(s)$

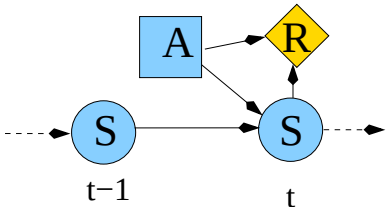


Figure 5.1: Two time slices of a Markov decision process MDP as a Bayesian network. The full network involves infinitely many time slices. Actions, A, induce transitions between states, S. A reward function, R, gives the utility of taking an action in a state.

of a policy π at a given state s satisfies [Put94]:

$$V_\pi(s) = R(s, \pi(s)) + \beta \sum_{s_t \in \mathcal{S}} Pr(s_t | \pi(s_{t-1}), s_{t-1}) \cdot V_\pi(t) \tag{5.1}$$

A policy π is *optimal* if $V_\pi \geq V_{\pi'}$ for all $s \in \mathcal{S}$ and policies π'. The *optimal value function* V^* is the value of any optimal policy.

Value iteration [Bel57] is a simple iterative approximation algorithm for constructing optimal policies. It proceeds by constructing a series of n-stage-to-go value functions V^n. Setting $V^0 = R$, we define

$$V^{n+1}(s) = \max_{a \in \mathcal{A}} \left\{ R(s, a) + \beta \sum_{t \in \mathcal{S}} Pr(s, a, t) \cdot V^n(t) \right\} \tag{5.2}$$

The sequence of value functions V^n produced by value iteration converges linearly to the optimal value function V^*. For some finite n, the actions that maximize Equation 5.2 form an optimal policy, and V^n approximates its value. A commonly used stopping criterion specifies termination of the iteration procedure when

$$\|V^{n+1} - V^n\| < \frac{\epsilon(1-\beta)}{2\beta} \tag{5.3}$$

(where $\|X\| = \max\{|x| : x \in X\}$ denotes the supremum norm). This ensures that the resulting value function V^{n+1} is within $\frac{\epsilon}{2}$ of the optimal function V^* at any state, and that the resulting policy is ϵ-optimal [Put94].

We will deal primarily with finite-horizon problems, in which the decision maker knows about some time, T, in the future at which his decision-making will stop. For these cases, there is no need for discounting, so $\beta = 1$ and value iteration is only iterated up to the horizon:

$$V^T(s) = \max_{a \in \mathcal{A}} \left\{ R(s, a) + \sum_{t \in \mathcal{S}} Pr(s, a, t) \cdot V^{T-1}(t) \right\} \tag{5.4}$$

where, again, $V^0 = R$.

5.1.2 Factored Representations and Structured Representations

The classical technique for representing MDPs is a tabular, or *flat* format, in which the transition function is written as a table giving the probability of going from any state to any other under all the actions. Although the flat representation is often effective for small problems, typical AI planning problems fall prey to Bellman's *curse of dimensionality*: the size of the state space grows exponentially with the number of domain features. The flat representation, which requires explicit enumeration of the state space, is typically infeasible. Further, it is usually difficult to write down, and hard to interpret.

A more interpretable representation is one in which the state space is implicitly represented as the cross product of a set of multinomial, discrete variables. The transition function is then written as a set of conditional probability distributions (CPTs). Each CPT is typically an easily interpretable function of a small set of variables.

Factored MDPs also allow the conditional independencies in the Markov network to be exploited by MDP solvers. In particular, CPTs can be represented using decision trees [DB97] or networks [HSAHB99], allowing similar states to be abstracted. These types of solvers usually gain much in efficiency, and have been successfully applied to large problems [Plu03]. We will use our *SPUDD* solver, which was the first MDP solver to use algebraic decision diagrams for representation and computation [HSAHB99].

5.1.3 Partially Observable Markov Decision Processes

In many cases of interest, the state space, S, is not fully observable, and can only be inferred from observations, resulting in a partially observable Markov decision process, or POMDP. Figure 5.2 shows a POMDP as a Bayesian network. As in the fully observable case, the environment is modeled using a finite set of states, $\mathcal{S}$, and a real valued reward function $R(s)$ maps states to values. The agent has available to it a finite set of actions, $\mathcal{A}$, which induce state transitions according to the probability function $P(S_t|A_t, S_{t-1})$. In the partially observable case, however, the state is not observable directly: it is some hidden underlying cause for the observations the agent makes, O_t, which are generated by states according to the probability function $P(O_t|S_t, A_t)$.

Since the state cannot be directly observed, it must be inferred from the observations, resulting in a *belief state*, $b(s)$, such that $b(s_t)$ is the probability that the system is in state s_t at time t. Updating the belief state in a dynamic Markovian model with observable inputs (the actions) was the subject of Section 3.6.3 and 4.7.

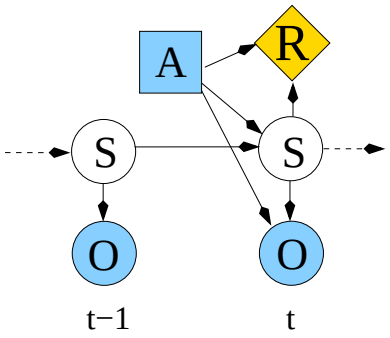


Figure 5.2: Two time slices of a partially observable Markov decision process (POMDP) as a Bayesian network. Shaded nodes are directly observable, while unshaded nodes are unobservable. Actions, A, induce transitions between states, S, which are not observable directly, but only through their effects on a set of observations, O. The reward function, R, gives the utility of being in states S.

It involves only computing the *forwards* pass of the standard forwards-backwards algorithm.

Once the agent has the belief state, however, she is faced with a much more difficult problem: how to choose an optimal action based upon it. The difficulty arises because the belief state lives in a continuous (bounded) space, and so the effects of each action must be evaluated against an infinite number of possible immediate futures that it may lead to. This is in contrast with the fully observable case, where the immediate (one-step look-ahead) future is finitely enumerable. The standard way to approach this problem is to recast it as a fully observable MDP with a continuous, $|\mathcal{S}|$-dimensional state space consisting of all possible belief states. In this case, the value of a belief state, $b(s_t)$, can be recursively computed using Equation 5.2, where we must now integrate over all possible observations and next belief states

$$V(b(s_{t-1})) = \max_{a \in \mathcal{A}} \left[R(b(s_{t-1})) + \beta \int_{o_t} \int_{b(s_t)} P(b(s_t) o_t | a b(s_{t-1})) V(b(s_t)) \right]$$

which, since $R(b(s_{t-1})) = \sum_{s_{t-1}} b(s_{t-1}) R(s_{t-1})$ and

$$\begin{aligned} \int_{b(s_t)} P(b(s_t) o_t | b(s_{t-1}), a) &= \sum_{s_t, s_{t-1}} P(s_t, o_t | s_{t-1}, a) b(s_{t-1}) \\ &= \sum_{s_t, s_{t-1}} P(s_t, s_{t-1} o_t | b(s_{t-1}), a) \\ &= P(o | b(s_{t-1}), a) \end{aligned}$$

is

$$V(b(s_{t-1})) = \max_{a \in \mathcal{A}} \left[\sum_{s_{t-1}} b(s_{t-1}) R(s_{t-1}) + \beta \int_{o_t} P(o_t | b(s_{t-1}), a) V(b(s_t)) \right] \quad (5.5)$$

Notice that we can write $V(b(s)) = \sum_{s \in \mathcal{S}} b(s) V(s)$. Since each $V(s)$ is some real-valued number, $V(b(s))$ is a piece-wise linear function of $b(s)$. If the observations, o, are drawn from a finite set, then it can be shown that the dynamic programming update in Equation 5.5 preserves the piecewise linearity of the value function [KLM96]. This property means that it is possible to solve for the optimal policy, although such a calculation may be difficult due to the explosion of the number linear pieces that make up the value function. It is often the case, however, that many of these linear pieces are dominated everywhere by some others, and can be removed from the representation of the value function, leading to significant efficiency improvements. Incremental pruning [CLZ97] is an algorithm for doing this. It is one of the most efficient algorithms, because it interleaves pruning into the dynamic programming step in such a way that the number of linear pieces is kept to a minimum. Factored state versions of the incremental pruning algorithm have also been developed [HF00], which make use of the structure inherent in the factored representation to increase the efficiency of the algorithm further.

If the observation space is continuous, as in our case, the problem becomes much more difficult. In fact, there are no known algorithms for computing optimal policies for such problems. The intractability of these problems arises because each dynamic programming step involves an integration over the entire observation space, destroying the piecewise-linearity of the value function. The simplest possible approximation technique is to simply consider the POMDP as a fully observable MDP, and to use the value of the state, S, as the most likely value over the belief state, $S = \arg\max_s b(s)$. This is called the *MDP approximation*. A slightly better approximation augments this most likely state with the entropy of the belief state, suitably discretized [RPT00]. For example, the position of a robot could be described as *at x with low certainty*, or the display of a human face could be described as *a smile with high certainty*. More complex approximate algorithms using Monte-Carlo sampling methods have also recently been studied [Thr00]. However, the complexity of these approximations grows very rapidly with the dimensionality of the input space, which is extremely large in our case.

5.1.4 Learning POMDPs

There are two elements to be learned from data in POMDPs. First, the parameters of the underlying Bayesian network: the transition function and the observation

function. Second, the reward function, which maps states and actions to values for the agent.

Reinforcement learning deals with learning of both these quantities [KLM96]. Usually, this involves trading off between exploration of the world in order to learn a model, and exploitation of the learned model in order to achieve optimal rewards. There are many ways to do this. The most popular are model-free approaches, which directly learn the value function without learning models of state transitions and observations. Since we are dealing with very high dimensional observation spaces, these approaches are impractical. Model-based approaches, on the other hand, learn the transition and reward functions from data, and then generate a policy of action from these, as described in the last two sections.

We take a very simple model-based approach, and acquire training data and reinforcement signals during a training phase, during which the agent acts randomly. The models are then learned offline, and the results are used in a testing phase in which the agent acts according to the generated policy. This approach is known as *certainty equivalence* [KLM96]. The learning procedures we have described for POMDPs in Chapter 4 can be applied for learning the parameters of the POMDPs, while the reward function can be learned by simple counting. Although there are significant limitations to certainty equivalence, we are primarily interested in demonstrating how our computer vision techniques for temporal and spatial abstraction can be integrated with the generation of policy. Taking this system one step further to a fully on-line version would involve implementing a more complex learning algorithm, interleaving exploration and exploitation. One such model-based algorithm is prioritized sweeping [KLM96].

5.2 POMDPs For Non-Verbal Displays in Games

As described at the beginning of this chapter, agents in an interaction can be modeled as playing a multi-agent game, and this can be described as a POMDP using the decision-analytic approach. In this section, we restrict our attention to two-agent interactions, and label the two agents a and b. We will be looking at agent a's POMDP, which maintains a dynamically evolving model of the world that we will describe at time t as a state, S_t^a, in some discrete state space $\mathcal{S}^a$. The state space does not have to be fully observable, but can be inferred from some observations, O^s. Each agent has some way of inferring distributions over unobservable states. In general, $\mathcal{S}^a$ includes agent a's model of the other agent in his environment, agent b. We are interested primarily in the non-verbal displays of agent b, so we will explicitly factor agent a's state space into agent a's description of agent b's non-verbal

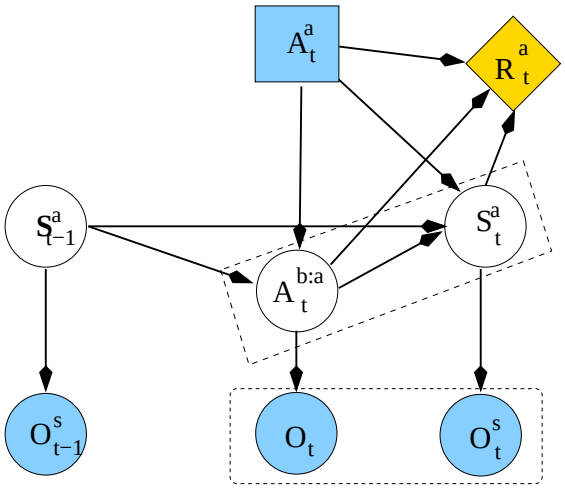


Figure 5.3: Time slice of the graphical model of interaction between two agents in a game. If the nodes are clustered according to the dashed lines, this model is a standard partially observable Markov decision process (POMDP) as in Figure 5.2

action, $\mathcal{A}^{b:a}$ and the remainder of the state space, denoted again as $\mathcal{S}^a$, as shown in Figure 5.3. Again, the actions $A^{b:a}$ do not have to be fully observable, but may need to be inferred from observations, O. As usual, agent a can perform some action, A^a, which comes from some set of actions $\mathcal{A}^a$. Finally, agent a maintains a reward function, R^a, which is a mapping from $S^a_t \times A^a_t \times A^{b:a}_t \rightarrow \mathcal{R}^a$.

Note that we have assumed that the agent has some way of separating the observations in O from those in O^s. This assumption requires that an agent is able to distinguish those observations that are related to the non-verbal actions, $A^{b:a}$, from those related to other states, S^a. If $A^{b:a}$ are facial displays, for example, this could be accomplished through a visual face tracker that explicitly separates visual observations of the face, the hands, the context, and other observations. However, in order to focus our attention on the non-verbal displays, we further assume that only these displays are unobservable directly. The states of context and of other agent actions, S^a, are considered fully observable, as shown in Figure 5.4.

Other than the utility nodes, this model has the same structure as the one discussed in Section 3.6.3. Nodes in Figure 3.17 have been relabeled in Figure 5.4 as $D \rightarrow A^{b:a}$, $A \rightarrow S^a_t$, and $C \rightarrow \{A^a, S^a_{t-1}\}$. Some further differences in the temporal structure can be included or deleted without loss of generality. The observations, O, are entire sequences of video frames and spatio-temporal derivatives. We assume that the temporal segmentation is provided directly by the onset times of S^a and A^a. That is, we assume that observable states and actions are delimiters for tem-

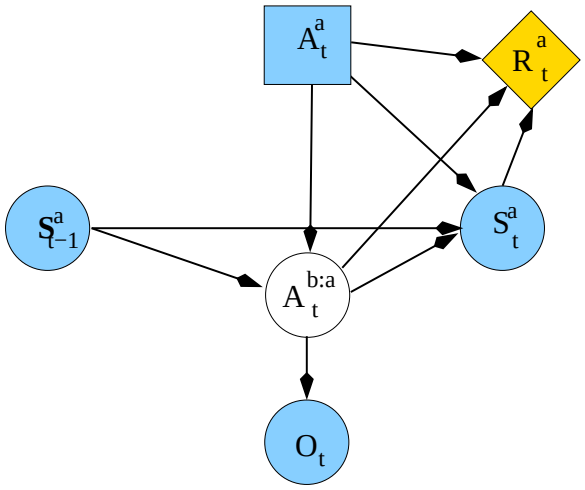


Figure 5.4: Time slice of the graphical model of interaction between two agents a game, in which the state, S^a, is fully observable.

poral events, and that the non-verbal displays will occur between these times. The parameters of such a model can be learned from data, as described in Section 4.7. The utility function can be learned using reinforcement learning, as discussed in Section 5.1.4. An approximate policy can then be derived using the techniques in Section 5.1.

5.3 Value-Directed Structure Learning

The value function, $V(s)$, gives the expected value for the decision maker in each state. However, there may be parts of the state space which are indistinguishable (or nearly so) with respect to certain characteristics, such as value or optimal action choice. These indistinguishable states can be grouped or merged together to form an *aggregate* or *abstract* state. The set of abstract states *partitions* the state space according to some characteristic. States of the original MDP which are part of the same abstract state are not distinguishable insofar as decisions go. Eliminating the distinctions between them by merging states can lead to efficiency gains without compromising decision quality. An agent needs only distinguish those states which are useful to it for achieving value.

In fact, such state aggregation is a form of structure learning based upon the utility of states. This *value-directed* structure learning is in contrast to more data dependent structure learning, in which the structure is determined solely based upon the statistical distribution of the data, and the complexity of the model. For

example, many structure learning algorithms use some simplicity prior (such as the minimum description length [Ris78, Bra99a, WPG01]), and find a trade-off between the model's precision and complexity. Others use an incremental generate-and-test mechanism in which states are added or deleted, and the model is tested to see if any advantage is gained [SO94].

We now discuss a particular technique for value-directed state aggregation applied to learning the number of facial displays or gestures that need to be distinguished in our learned POMDP. As we have mentioned, the state space is represented in a factored POMDP as a product over a set of variables. In our model, the values of one of these variables, $A^{b:a}$, are the (unlabeled) gestures or facial displays. This variable splits the value function into N_a pieces, V_i, one for each value, i, of the variable $A^{b:a}$. Each such V_i gives the values of being in any state in which $A^{B:a} = i$. A similar split occurs for the policy, yielding sub-policies, π_i, giving the actions to take for each $A^{b:a} = i$. The V_i can be compared by computing the difference between them, $d_{ij} = \|V_i - V_j\|$, where $\|X\| \equiv max\{x : x \in X\}$ is the supremum norm. Two sub-policies, π_i and π_j are considered equivalent if the optimal actions agree for every state: if $\pi_i \wedge \pi_j$. These comparisons are used in the following algorithm for learning the number of display states, N_a. The algorithm starts by assigning N_a to be as large as the training data will support, and prunes redundant states.

```
repeat
  1.learn the POMDP model
  2.compute V_i and π_i ∀ i using value iteration
  3.compute d_ij = ||V_i − V_j|| ∀ (i, j), i ≠ j
  4.if ∃(i, j)(π_i ∧ π_j)
  5.    {i, j} = arg min_{kl}(d_kl ∀{k, l} | π_k ∧ π_l)
  6.    merge states i and j
  7.    N_a ← N_a − 1
    end
until N_a stops changing
```

Figure 5.5: Procedure for value-directed structure and parameter learning for POMDPs

There are many potential ways to merge states at step 6, but we simply delete one of the the redundant states. Note that the algorithm could also start with $N_a = 2$ and add states until redundancies appear, but we have not experimented with this version. Added states could be added initialized randomly, or as the most populated current state with added noise.

Learning the POMDP parameters (step 1 in Figure 5.5) involves iterations of expectation-maximization, as implemented by the forwards-backwards algorithm (message passing in the Bayesian network). The complexity of this procedure is $O(N_d(N_x^2 + N_w^2)T)$, where N_x is the number of states in the dynamics process, N_w is the number of states in the configuration process, N_d is the number of high-level facial display states and T is the length of the entire sequence of data. The complexity of value iteration (step 2 in Figure 5.5) has a complexity of $O(N_s^2 N_a H)$, where N_s is the number of states in the POMDP, N_a is the number of actions, and H is the horizon. The remainder of the algorithm is $O(N_s^2)$ (steps 4-7 in Figure 5.5). Therefore, the complete learning procedure has a worst-case complexity of $O(N_d^2(N_x^2 + N_w^2)T + N_d N_s^2 N_a H)$. In typical problems, N_d, N_x and N_w are all quite small numbers, while N_s is very large (exponential in the number of variables in the POMDP). Therefore, even using simple POMDP solution approximations, the complexity will generally be dominated by the second term, $O(N_d N_s^2 N_a H)$. Attempting to compute optimal POMDP solutions would increase this complexity.

5.4 Experimental Design

To investigate the relationships between facial displays and other, conditioning factors, we adopt an experimental paradigm in which we observe humans playing computer games. The games embody constraints to be placed on the POMDPs, so we can focus the learning on specific areas of interest. The following describes the method.

1. Design game and encode it as a POMDP. In the general case, the encoding of the POMDP would be learned directly from data. However, we are constraining many of the variables in the game in order to focus on facial display models, and we so characterize the context in terms of a set of discrete, high-level variables. There is no single way to encode a particular domain as a POMDP. Part of the process involves testing the encoding by manually assigning conditional probability distributions based on prior knowledge of the domain, and then computing a policy to ensure that it correctly predicts actions. This also gives the encoder some idea about how the problem is structured as an MDP, to allow for easier interpretation of results when the model is learned from data.

2. Gather training & test data sets. This involves a human playing the game either against another human or against a computer agent. In the latter case, the agent selects actions randomly in both training and test data sets.

3. Apply the procedure in Figure 5.5 to learn the parameters and structure of the POMDP model from the training data. This also computes an approximate policy of action.

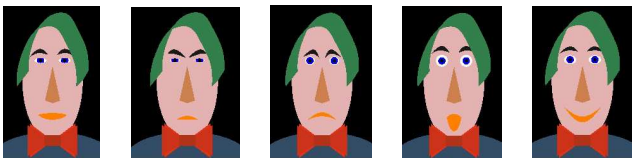
4. Use the model to predict actions in the test data set. In the case of two humans playing against one another, the predictions are over the (observable) actions of one of the players, and can be compared to the actual actions taken in the test data for a performance measure. In the case of a human playing against an agent, the predictions are action selections from the policy, but since the agent was playing randomly, these cannot be compared to the agent's actions. Instead, we use the actions as if they were selected in the real game, and collect rewards based upon them. The total collected rewards are a performance indicator.

The following three sections describe this procedure applied to three simple games. The first (imitation game) only involves facial displays as actions, and so does not have a reward function. This game is used to explore the representational power of our computer vision modeling techniques. The second (robot control) involves a single human performing gestures for robot control. The robot control experiments are intended as a simple demonstration of our system working with something *other than* facial expression. The experimental setup for the robot control experiments is very simple, and we do not claim a gesture recognition system that would deal with orientation, time scaling, or robot movement and viewpoint. This second game also demonstrates our value-directed structure learning techniques. The third (card matching game) involves two humans playing a collaborative game. The facial displays are fairly simple, but the decision theory problem is much more complex than the other two games. This data is used to demonstrate how a policy can be computed based, in part, on non-verbal displays.

5.5 Imitation Game

The imitation game is a single player game in which the player watches a computer animated face on a screen, and is told to imitate the actions of the face. While many face generation systems use complex 3D graphics, this face is a simple cartoon. This allows for fast rendering, and avoids problems with the "uncanny valley", where human-like interfaces look disturbing if they are just less than perfectly realistic [Mor82]. Further, research has shown that humans can interact with simple generated faces as a real human face [RN96]. The animated displays start from a neutral face, as shown in Figure 5.6(a), then warp to one of the 4 poses shown in

Figures 5.6. The pose is held for roughly a second, and the face then warps back to the neutral pose where it remains for an additional second. Although these displays may be reminiscent of so-called *prototypical facial expressions*, the displays they elicited were clearly *not* expressions of emotion, but only reactions to context. That they may have been *interpreted* as emotional displays by the subjects is not relevant here.



(a) a_1 a_2 a_3 a_4

Figure 5.6: (a) neutral face ($A^a = a_1 ... a_4$) Faces which subjects were told to imitate

In this game, the actions of the human player are the imitative displays, and the state of the game is the animated display. The actions of the human are not observable, but must be inferred from a video stream. The additional independencies in this game result in a simplification of the model in Figure 5.3, as shown in Figure 5.7. The agent's actions, A^a, are choices of cartoon facial displays, $a_1 \ldots a_4$. The observations of the human's actions, $\mathbf{O}$, are sequences of video images and the spatio-temporal derivatives between subsequent video frames, as described in Chapter 3. The human player's actions, as modeled by the agent, are described by a discrete N_a-valued variable, $A^{b:a} \in \{a_1 \ldots a_{N_a}\}$, where the agent has control over N_a.

Although it is possible to include a utility function in this game, it would imply associating some reward with particular facial displays (e.g., smiles). The agent could then use the reward function to figure out which of its cartoon displays is most likely to elicit the rewarded displays. However, this game was designed to test the vision sequence representation model, and so the facial displays are not tied to any task, and a utility function is not very useful. Instead, we leave out the utility function, and compute a measure of the ability of the model to represent the different imitation displays. We therefore *hide* the cartoon display labels, A^a, in the test data set, and compute the distribution over this variable:

$$P(A^a|\mathbf{O}) \propto P(\mathbf{O}|A^a)P(A^a) = \sum_i P(\mathbf{O}|A_i^{b:a})P(A_i^{b:a}|A^a)$$

The maximum of this distribution tells us which cartoon display was most likely to have produced the observed human display. Agreement with the actual cartoon displays gives us an indication of how well the model can represent the display imitations.

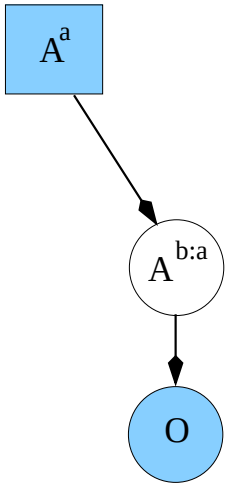


Figure 5.7: Time slice of graphical model of simple imitation game.

5.6 Hand Gestures for Robot Control

This "game" involves a human operator issuing navigation commands to a robot using hand gestures. The robot has four possible navigation actions, *go left, go right, stop* and *go forwards*, and the human operator uses four distinct hand gestures corresponding to each command. The robot, however, must learn the mapping from hand gestures to its actions. It learns this mapping during a training phase during which it acts randomly in response to the operator's hand gestures. Rewards of one unit are explicitly assigned by the operator if the robot performs the correct actions. The robot gets no reward if it performs the wrong action.

Again, each interaction in this game is temporally independent. The POMDP for each time slice is shown in Figure 5.8 from the point of view of the robot. The robot's action is denoted A^a, while the operator's action (to reward the robot or not) is explicitly represented in the model as A^b, and is fully observable. Thus, the reward function is a one-to-one mapping from this action. The robot's observations of the operator's hand gesture, O, is conditioned on the high-level interpretation of the gesture, $A^{b:a}$, which is a discrete-valued variable with N_a values.

An optimal policy of action in this model needs only be computed over a horizon of one time step (since the actions are temporally independent). The policy, π, specifies an action for each possible recognized gesture, $A^{b:a}$, such that $A^a = \pi(A^{b:a} = i)$ is the action which will most likely result in the operator rewarding the robot if $A^{b:a} = i$ is observed. For example, if the command is a *stop* gesture, then the robot's *stop* action will most likely result in a reward. Notice that the MDP approximation can easily lead to sub-optimal action choices. Suppose that the dis-

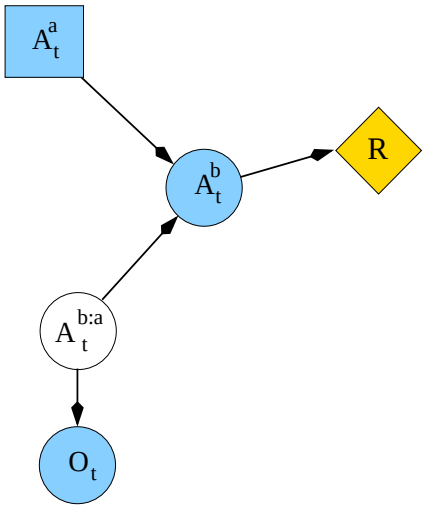


Figure 5.8: Two time slices of a POMDP for robot control gestures. A^a is the robot's action (one of *go left, go right, stop, forwards*), A^b is the operator's action (reward or punish the robot), R is the reward function (a one-to-one mapping from A^b), $A^{b:a}$ is the robot's interpretation of the control command (gesture), and O is the video sequence observation of the gesture.

tribution over gesture interpretations, $P(A^{b:a}|\mathbf{O})$, has a roughly equal value for two values of $A^{b:a}$, so the robot is uncertain about which gesture was actually performed. The MDP approximation will simply choose the most likely one, possibly leading to an error. An optimal solution to the POMDP would include this uncertainty, and might specify some other action (such as *ask for clarification*) in the case of such uncertainty. This type of analysis is important for dialogue modeling [PH00].

5.7 Card Matching Game

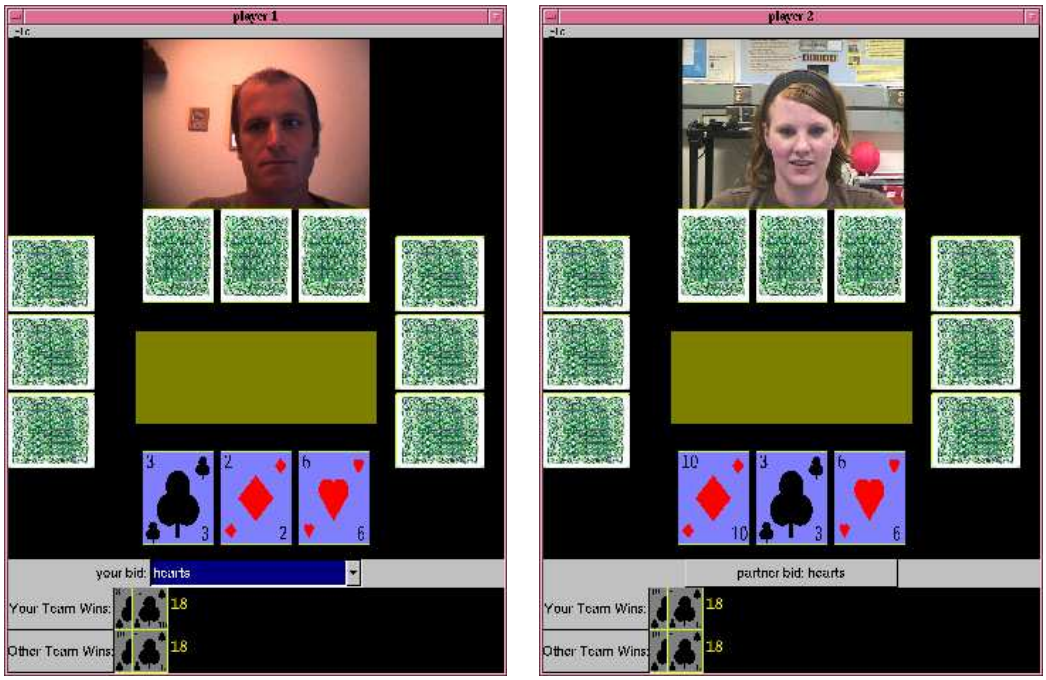
Two teams of two players per team play the card matching game. At the start of a round, each player is dealt three cards: a heart, a diamond and a club. Each player can only see his own set of cards. The values of the cards (ranging from ace to ten), and their placement on the table, are randomly distributed. Each player's cards are dealt from a different deck, which is re-shuffled after every round. The players all play a single card simultaneously, and if the suits of the cards played by the two members of a team match, then that team *reserves* the sum of the values on the two cards, otherwise, that team *reserves* zero. The team with the highest reserve *wins* their reserve, while the other team wins nothing (and loses their reserve). On alternate rounds, a player has an opportunity to send a confidential *bid* to his

partner, indicating a card suit. The *bids* are non-binding and do not directly affect the payoffs in the game. During the other rounds (*displaying rounds*), the player can only see her partner's bids, and then play one of her cards. There is no time limit for playing a card, but the decision to play a card is final once made. Finally, each player can see (but not hear) his teammate through a real-time video link. This game constrains the players to use their gestures by not having an audio link, so the players cannot speak to one another. Players cannot see members of the opposing team, nor can they see the confidential bids of the opposing team.

One of the two teams is simply implemented in software, and does not actually have a video link or a bidding process. Their presence is only to give the human players incentive to cooperate, and to try to choose the highest matching suit. We can, therefore, assume that the human players we will be modeling have positive value assigned to choosing matching pairs of cards with the highest possible value. This will allow us to disregard the opposing team in the analysis, and use a simplified reward structure.

A picture of the game interface during a typical interaction is shown in Figures 5.9 and 5.10. We will refer to the two players by names in what follows: player 1 is "Ann", while player 2 is "Bob". On the left, Ann's interface shows her cards face up below the 'card table', Bob's face, Bob's cards face down below his face, the opposing team's cards on the left and right of the card table, and both teams current winnings along the bottom. On the right, Bob's interface shows the same thing, but he sees Ann's face and sees only his own cards face up. Figure 5.9 shows stage 1, in which Ann (left) has made a bid of her highest card (a heart) to Bob (right). In stage 2, both players have committed their hearts, and the round terminates with the team winning 12 points. The interfaces at the beginning of the next round are shown in Figure 5.10 in which the players are happy to have won, can see their winnings added to their total, and it is now Bob's turn to bid.

We now describe the encoding of the card matching game as a POMDP. The only part of the state which is non-observable are the facial displays of the players. In more complex games, this may not be true, and we would have to model other parts of the state space as non-observable. Our modeling techniques do not preclude this. However, solutions to the POMDPs will become increasingly difficult as the non-observable part of the state space grows. For example, in the card matching game, Bob could maintain beliefs about what the values of Ann's cards were. These beliefs could be useful in taking an optimal decision. If he notices that Ann vehemently shook her head when he bid clubs, sort of grimaced when he bid hearts, and sort of nodded when he bid diamonds, he might assess a distribution over her cards which would help him bid the optimal choice for both. Our assumption, however, seems



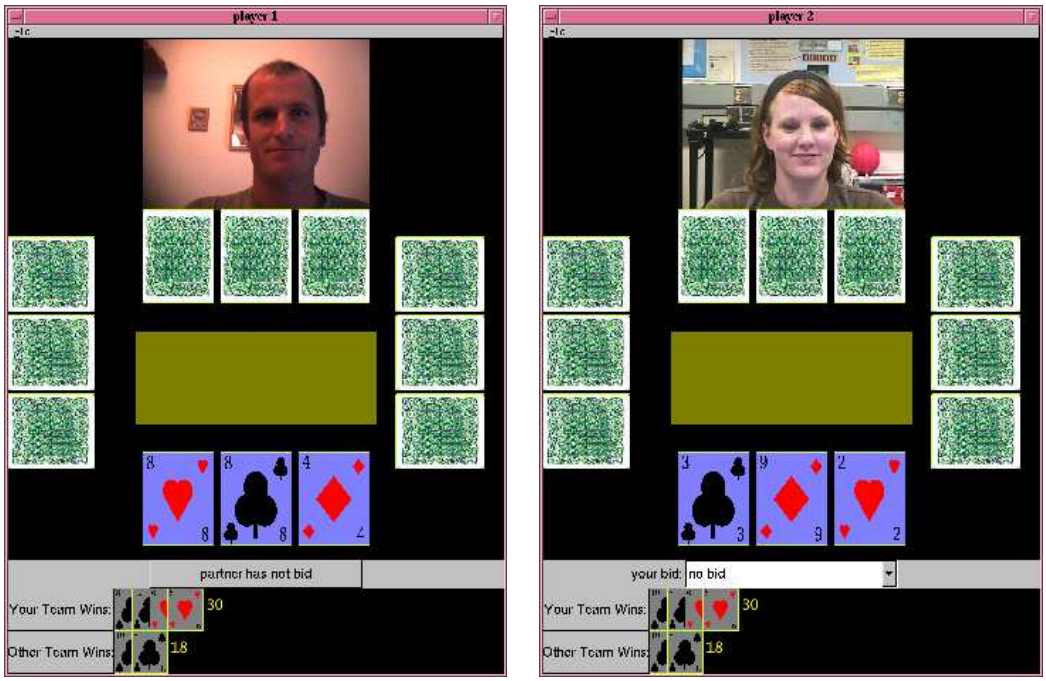
Ann's view Bob's view

stage 1

Ann's view Bob's view

stage 2

Figure 5.9: Game interfaces for two players during a typical round. See text for description.



Ann's view Bob's view

stage 3

Figure 5.10: The beginning of the round subsequent to the one shown in Figure 5.9.

variable	values	description
$B\heartsuit v$	$\{v1, v2, v3\}$	value of Bob's heart card
$B\diamondsuit v$	$\{v1, v2, v3\}$	value of Bob's diamond card
$B\clubsuit v$	$\{v1, v2, v3\}$	value of Bob's club card
Acv	$\{null, v1, v2, v3\}$	value of card Ann committed
Bcv	$\{null, v1, v2, v3\}$	value of card Bob committed
$match$	$\{null, \heartsuit, \diamondsuit, \clubsuit\}$	match of played cards is false ($null$) or one of the suits
$Aact$	$\{null, cmt\heartsuit, cmt\diamondsuit, cmt\clubsuit\}$	Ann's action is nothing ($null$), or commit (cmt) a suit.
bid	$\{null, \heartsuit, \diamondsuit, \clubsuit\}$	current bid suit
$Acom$	$\{d_1 \dots d_{N_d}\}$	state Ann's facial display
$Bact$	$\{null, bid\heartsuit, bid\diamondsuit, bid\clubsuit$ $cmt\heartsuit, cmt\diamondsuit, cmt\clubsuit\}$	Bob's action is nothing ($null$), or bid (bid) or commit (cmt) a suit.

Table 5.1: Variables and action for Bob in the card matching game POMDP during a round when player 2 has the bid.

to be well supported in our experimental data, where the players often commit to bids on the first step of a round, without engaging in complex deliberations about the precise value of all the cards.

For a given player, each round of the game is either a bidding round (when she makes the bid), or not (when she sees her partner's bid). To simplify the analysis, we will consider each of these types of rounds separately. We will show how the two can be combined into a single POMDP in Section 5.7.3.

5.7.1 Bidding Round

There are nine variables which describe the state of the game when a player has the bid. While these may not be the smallest set of variables, they give an intuitive description of the game. Aggregation techniques [SAHB00] can always be used to reduce the state space further. Table 5.1 shows the variables, their values, and gives a brief description of each. The POMDP we will describe can be used for either player. However, to keep things clear, we will develop the model from Bob's perspective. Variable names which are Bob's private, observable, state will start with a B, while those which Bob uses to describe Ann's state and actions will start with an A. Variables which are common knowledge do not start with either letter. The state could be used to model Ann's perspective, of course, by renaming Ann to Bob and vice-versa.

The three suits in the game, hearts, diamonds, and clubs, will be denoted by symbols: $(\heartsuit, \diamondsuit, \clubsuit)$, respectively. Bob's actions (those over which he has control) are $Bact$, which can be $null$ (no action), or sending a confidential bid of one of the suits $(bid\heartsuit, bid\diamondsuit, bid\clubsuit)$ or committing a card of some suit $(cmt\heartsuit, cmt\diamondsuit, cmt\clubsuit)$. Ann's observed actions are part of the state, $Aact$, and can be $null$ (no action), or committing a card of some suit $(cmt\heartsuit, cmt\diamondsuit, cmt\clubsuit)$. The values of Bob's heart, diamond and club cards are $B\heartsuit v, B\diamondsuit v, B\clubsuit v$, respectively. There are ten possible values for each card, which adds much complexity to the decision problem by expanding the state space. To reduce this complexity, we approximate these ten values with three values, $v1, v2, v3$, where cards valued 1-4 are labeled $v1$, 5-7 are $v2$ and 8-10 are labeled $v3$. This approximation may hide some of the structure of the game. For example, if a player's cards are 1, 3 and 4, then this re-labeling will give them all equal value ($v1$). However, we assume that when cards are close in value, the player has little preference over which one is actually played. This assumption seems to be borne out by experience, and the approximation seems sufficient to generate a near optimal policy. Another possibility is variables describing which suit is the highest, second and lowest cards.

The values of Ann's committed cards are Acv and the values of Bob's committed cards are Bcv. These can be $null$, or $v1, v2, v3$. The $match$ variable gives the suit of a match (if the committed cards do indeed match), otherwise it is false ($null$). These variables (Acv, Bcv and $match$) only become non-null after commit actions from both players, and immediately become $null$ again on any subsequent action. The bid variable describes the bid currently on the table (observable to both players on a team). It is affected by Bob's bidding actions, but is set to $null$ after each round. Finally, the $Acom$ variable describes Ann's communication through the video link. It is one of N_d states, $d_1 \ldots d_{N_d}$, which have unspecified meaning. They will, however, obtain meaning through their interactions with the other variables in the POMDP.

The POMDP model is shown in Figure 5.11 as a two time-slice Bayesian network. This is the same model as was shown in Figure 5.3, with the additional assumption that Ann's action, $Aact$ ($A^{b:a}$ in Figure 5.3), is fully observable, and so the action observations, W, do not play a role. Furthermore, we have factored the state, S_t^a, into nine variables, and included the action of the partner (Ann) as part of the state. The conditional dependencies between fully observable variables are easy to learn by counting co-occurrences. In fact, they can be read off from the structure of the game. For example, the heart's value ($B\heartsuit v$) stays the same on a bid action, but gets reset randomly on a commit action. The value of Bob's committed card (Bcv) is exactly the value (from $B\heartsuit v, B\diamondsuit v, B\clubsuit v$) of the card Bob committed (as

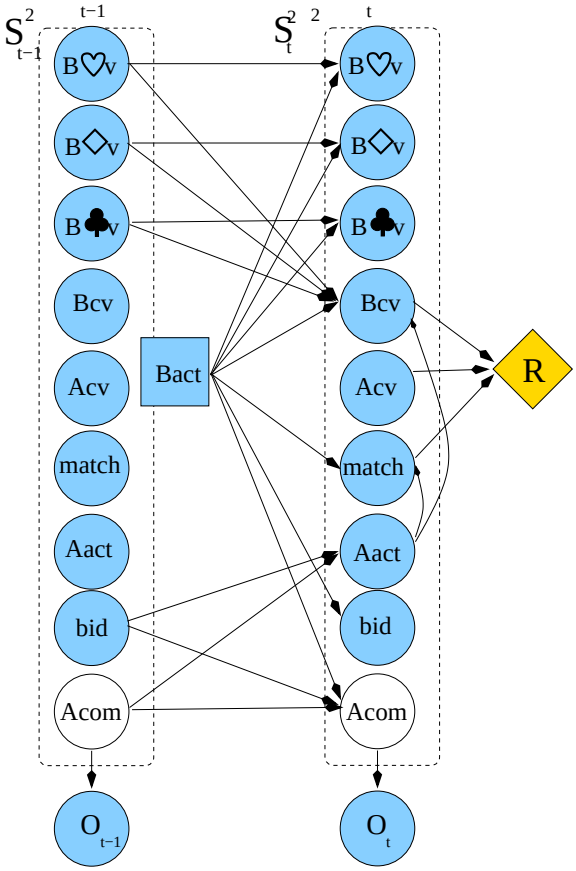


Figure 5.11: Two time slices of Bob's graphical model for card matching game. The variables and actions are described in Table 5.1.

name	S_t	S_{t-1}, A_t	description
Θ_A	$Aact$	$\{bid, Acom, Bact\}$	what Ann will do given Bob's action, the current bid, and Ann's previous display
Θ_D	$Acom$	$\{bid, Acom, Bact\}$	what Ann will display given Bob's action, the current bid, and Ann's previous display
$N(O)$	$\mathbf{O}_t$	$Acom$	probability of a set of observations given high level description of Ann's display

Table 5.2: Conditional probability distributions to be learned from data in POMDP for card matching game from Bob's perspective.

given by $Bact$ if $Aact \in \{cmt\heartsuit, cmt\diamondsuit, cmt\clubsuit\}$. The value of Ann's committed card (Acv) is an even distribution over the possible values if Ann's action ($Aact$) is non-null. Finally, the *match* is suit s if $Aact$ and $Bact$ are both $cmt\langle s\rangle$. The distributions which must be learned from video data are shown in Table 5.2. All three of the distributions in Table 5.2 are part of the model we discussed in Section 3.6.3, and so learning the POMDP reduces to learning the C4MG. The first, Θ_A, describes what Ann will do given Bob's action, the previous bid, and Ann's display. For example, we may expect that Ann will commit her diamond if the previous bid is diamonds and Ann's display was a nod of the head. This distribution is actually independent of $Aact$, since Bob's action is only made public after both players have acted, so that Ann will not be affected by Bob's action in the same time step. The second learned distribution, Θ_D, describes what Ann will display given Bob's action, the previous bid and Ann's previous display. For example, we may expect Ann to nod her head if the previous bid was hearts, her previous display was shaking the head, and Bob has just bid diamonds. Certainly, agreement will be more likely on the second bid than on the first if the first was refused. It should now be clear how these two distributions give "meaning" to Ann's displays, $Acom$. Bob only needs to know what displays to expect from Ann, and what actions Ann's displays are predictive of in any given situation. The POMDP based on these distributions will yield the same policy of action *regardless of what particular displays Ann actually chooses*. It only matters that his displays are consistent predictors of her future actions in each context.

The number of display states to be learned (N_d) must still be specified manually. However, since we are using the results in a probabilistic model (the POMDP), it is only critical to choose N_d *large enough* to have enough states to capture all the important displays in the game. Our value-directed structure learning techniques (Section 5.3) can then be used to find the smallest set of display states which still

yield high expected value policies. Practical concerns limit the value of N_d we start with, since it also depends on the amount of training data available. With a small training set, choosing N_d too large will result in models which are heavily biased by prior distributions. The resulting policies may be very sub-optimal.

The reward function is only based upon fully observable variables, and so can be directly written down. Recall that, in the actual game, the players only get the sum of the values on the cards they played if the card suits match, and only if the sum is greater than the other team's sum, if the other team's cards match. However, the other team only plays a motivational role in the game, and we can disregard it here by assuming the players are rewarded by getting a match, and the reward is the sum of the card values in the POMDP, regardless of the other team's play. We are assuming that the players are striving only to play matched suits which have the greatest sum, since they have no control over the other team's play. Thus, the reward function is $Acv + Bcv$ if *match* is not *null*, otherwise it is 0. Section 6.3 shows values for these distributions learned from training data, and shows a policy of action generated from the resulting POMDP.

5.7.2 Displaying Round

On alternate rounds in the card matching game, a player cannot bid, but only watch for bids from their partner. However, the actions available to the player now involve generating displays indicative of agreement (or not) with the bid. This is not a facial display modeling problem (since a player cannot see their own display), but a *generation* problem: the player must make facial displays. Thus, a POMDP model of the displaying rounds would involve actions for each facial display necessary in the game, and these actions could be used to drive an animated character, for example. However, prior knowledge of the facial displays and a model of how to generate them would be necessary. Since this thesis is not concerned with the generation problem, we do not discuss this round of the game in detail, but only give a rough overview of what the associated decision problem looks like. We then show how the MDP for the displaying round can be integrated with the POMDP for the bidding round to form a single POMDP for each player for the game.

In the displaying round, the variables are the same as in the bidding round, except that A and B are reversed, and there is no need to model the displays of the partner, so that the MDP is fully observable. The actions available to the decision maker are now performing facial displays and committing cards. As we have seen, the displays used by players of the card matching game are simple head nods and shakes, which we use as actions in the displaying round POMDP. The model combines these displays with the actions to commit cards in the game (which

Bob's action	variables affected	description
wait	none	do nothing
shake	*bid*	indicate disagreement with current bid expect next *bid* to be different
nod.cmt♡	all	indicate agreement with current bid and commit ♡
nod.cmt♢	all	indicate agreement with current bid and commit ♢
nod.cmt♣	all	indicate agreement with current bid and commit ♣

Table 5.3: Actions during displaying round.

are final in the round), to give the five actions shown in Table 5.3.

5.7.3 Combining MDPs

Combining the partially observable MDP for the bidding round with the fully observable one for the displaying round involves simply adding a *round* variable, which alternates between *bid* and *display*. When *round* is *bid*, the conditional probability tables are those for the bidding round, and when *round* is *display*, the conditional probability tables are those for the displaying round. The full action set is always available, but bidding round actions have no effect in the displaying round, and vice-versa. Figure 5.12 shows the combined POMDP from Bob's perspective. We have labeled the subset of state variables as $S^a = \{B\heartsuit v, B\diamondsuit v, B\clubsuit v, Acv, Bcv, match, bid\}$, but have kept Ann's action, *Aact*, and Ann's display, *Acom*, factored. The policy generated by this combined POMDP is factored on the *round* variable into the policy for the bidding round, and that for the displaying round. Each of these smaller POMDPs can be solved independently, and the resulting policy combined in this way.

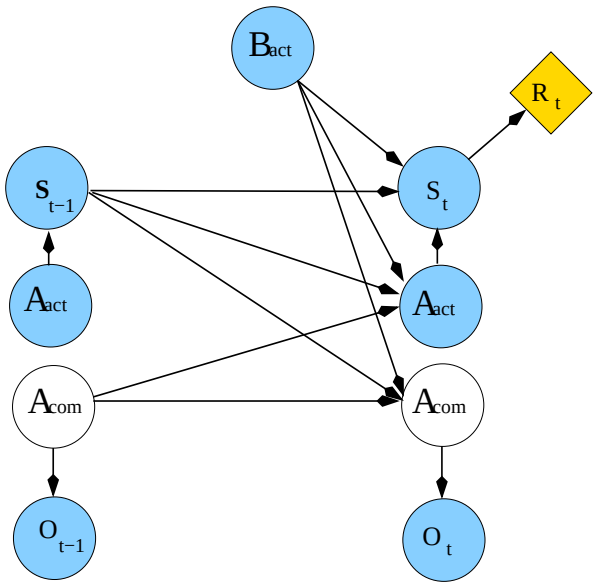


Figure 5.12: Two time slices of combined POMDP for the card matching game.

Chapter 6

Experiments

This chapter presents experimental results from training some of the models presented in Chapters 3 and 4. We present results from the three interactions described in Chapter 5. Section 6.1 describes the imitation game, in which a single subject imitates a cartoon display face (Section 5.5). Section 6.1.1 shows how we can find clusters of similar flow fields using the simple Gaussian mixture model from Section 3.2 (learning described in Section 4.2). This model considers all frames of the video to be independent, and looks only to find flow field models which efficiently span the set of flow fields in the data. The results are similar to those we have presented in [HL03]. Section 6.1.2 shows the same type of procedure applied to (temporally independent) images using the model in Section 3.3 (learning in Section 4.3). The results in these two sections are meant to demonstrate how the spatial abstraction of flow fields and images is performed. The analysis is qualitative, and is meant to get the reader acquainted with our descriptions of the low-level learned models. Section 6.1.3 then shows the results of training models with temporal dependence. In particular, we train the context dependent mixture of coupled hidden Markov models (C3MG), as described in Section 3.6.1. The analysis is this case does not include any utility measures, as described in Section 5.5. Instead, we predict labels on the test sequences as an indication of the representative power of the model.

The robot control experiments involve a single person gesturing to a robot, and are described in Section 6.2. The results are from cross-validation experiments in which the robot actions are chosen according to the learned model for the test data set, and the accumulated reward is used as the measure of success.

Section 6.3 presents results on the data taken during the simple card matching game described in Section 5.7. In this case, we train the full model including actions, and attempt to find policies of action based upon the model. While the facial displays are simpler during this game, the experiments we present show how

actions can be incorporated and how policies can be discovered based upon the learned visual models.

6.1 Imitation Game

Subjects were seated in front of a computer terminal on which an animated cartoon shows facial displays according to the game described in Section 5.5. Three subjects were told that their task is to imitate these displays, and were shown each of displays initially and told to practice imitating them. Once they were satisfied with their imitations, they pressed a key, and the system began recording a video sequence through a Sony EVI-D30 color camera mounted above the computer screen. While the subjects were being recorded, the cartoon face performed a series of 40 randomly selected facial displays over a period of 2 minutes. Frames were captured at 160×120 with a BTTV frame grabber card on a desktop Pentium III PC running the Linux operating system. The frame rates were almost always above 28fps.

After the experiment, most subjects reported either that they did not notice a significant difference between cartoon displays a_1 and a_2, or that they could not find a way to imitate the second one, a_2, due to the extremely down-turned mouth.

6.1.1 Clustering Flow Fields

We first examine the results of learning the simple mixture of Gaussians model with feature weighting described in Section 4.2 on data taken during the imitation game. Given the video sequence, we want to discover the major categories of instantaneous motions (flow fields) present. This is an important preliminary step to learning the full temporal models, as it constitutes the lowest level and performs the spatial abstraction.

We clustered a set of 904 frames from a 3600 frame sequence of a person performing 4 different facial expressions during the imitation game. As described in Section 5.5, the subject was imitating an on-screen cartoon face which was displaying 4 prototypical expressions: happy, sad, surprised and angry. The person's face was tracked using the flow-based tracker described in Section 3.7. The video frames in this data set are not labeled, and so the analysis is qualitative: our methods *discover* clusters of optical flow fields, and we *interpret* these clusters, which can be related to established high-level concepts.

We automatically selected 904 frames which had significant motions in them by thresholding the mean magnitude of the optical flow. Applying our methods to all 3600 frames does not substantially change the result, since the flow fields from the other frames all fall close to the origin, and so are represented by one of the

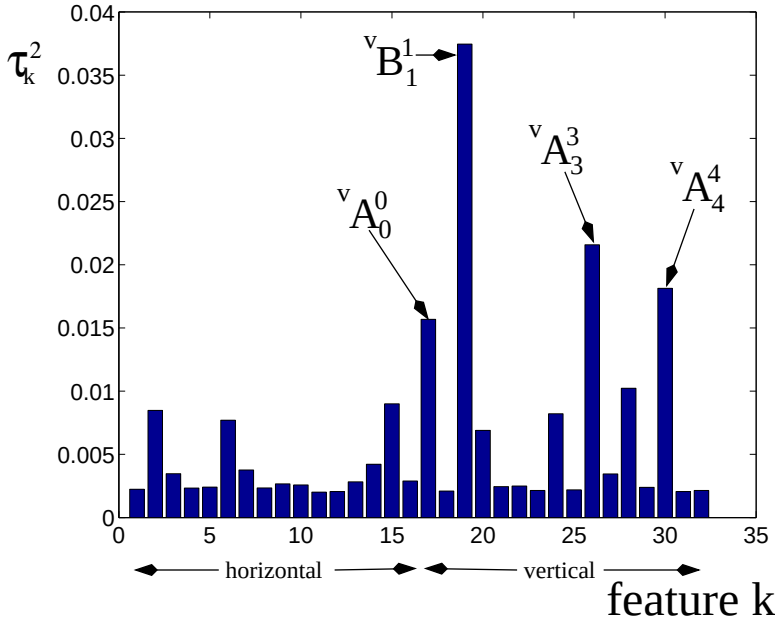


Figure 6.1: Feature weights learned for facial expression data.

learned clusters. We used the first 16 Zernike coefficients for each horizontal and vertical flow, resulting in a 32-dimensional basis vector, z. The noise parameters were set to $\sigma_1 = 0.08$, $\sigma_2 = 1.0$, $\sigma_p = 10.0$, $\sigma_0 = 0.5$ and $\sigma_d = 0.1$. The feature weighting parameters were set to $a = 1$, $b = 0.01$ and $\alpha = 34$. The parameters $\mu_{z,x}$ and $\Lambda_{z,x}$ were initialized by choosing K data points randomly as the initial seeds for K-means clustering, and Gaussians were fit to the resulting classes. Feature weights τ_k^2 were initialized to 1. Results were relatively insensitive to the initialization.

We trained a model with 8 classes. We choose 8 classes because it is large enough to see some structure in the learned classes, but small enough to allow for significant numbers of data points in each class. Figure 6.1 shows the final values of the feature weights, τ_k^2. The first 16 dimensions are the Zernike coefficients corresponding to horizontal flow ($^uA_n^m$, $^uB_n^m$ for $n < 5$), while the last 16 are those corresponding to vertical flow ($^vA_n^m$, $^vB_n^m$ for $n < 5$), ordered by increasing n and m values. The feature weights are clearly favoring the vertical flows, because a major component of the facial expressions are raising and lowering of eyebrows. The four most relevant features are $\{^vB_1^1, ^vA_3^3, ^vA_4^4, ^vA_0^0\}$. There are six other moderately relevant features. The remaining 22 features are irrelevant.

Figure 6.2 shows the reconstructed Zernike vectors plotted along two of the relevant features ($^vB_1^1, ^vA_0^0$). The clusters are denoted by the shape and color of the data points. Reconstructed optical flow fields (using Equation 3.24) are shown

for representative frames within each cluster. The classes are roughly (1) little or no motion, (2) eyebrows raising slowly, (3) jaw expanding, (4) eyebrows raising quickly, (5) jaw relaxing and (6) eyebrows lowering. The remaining two classes corresponded to translational motions (7) up and (8) down. However, these clusters only accounted for a small fraction of the data, and are not considered further.

Figure 6.3 shows an example of a raising eyebrows event. The original images are shown with the tracked regions superimposed. The temporal derivatives are shown below the images. We do not show the multi-scale temporal derivatives, only those at the highest resolution. They are meant to be demonstrative of the change in the image only. The flow fields computed using the Simoncelli method are shown below the derivatives. Finally, the the expected reconstructed flow fields from the model states are shown. The two central flow fields (105-106-107) are detected as state 4 (eyebrows raising rapidly), surrounded by more slowly raising eyebrow motions (state 2). Once the eyebrows reach their apex, the state returns to 1 (no motion) by frame 108.

Figure 6.4 shows a smiling event detected as state 3 from frame 2174-2178, surrounded by state 1 events (no motion). Figures 6.5 and 6.6 show the sequel to Figure 6.3, in which the subject's face returns to neutral. He begins by lowering his eyebrows (Figure 6.5, frames 115-117), which is classified as state 6, followed by a relaxation of his smile (Figure 6.6, frames 162-164), which is classified as state 5.

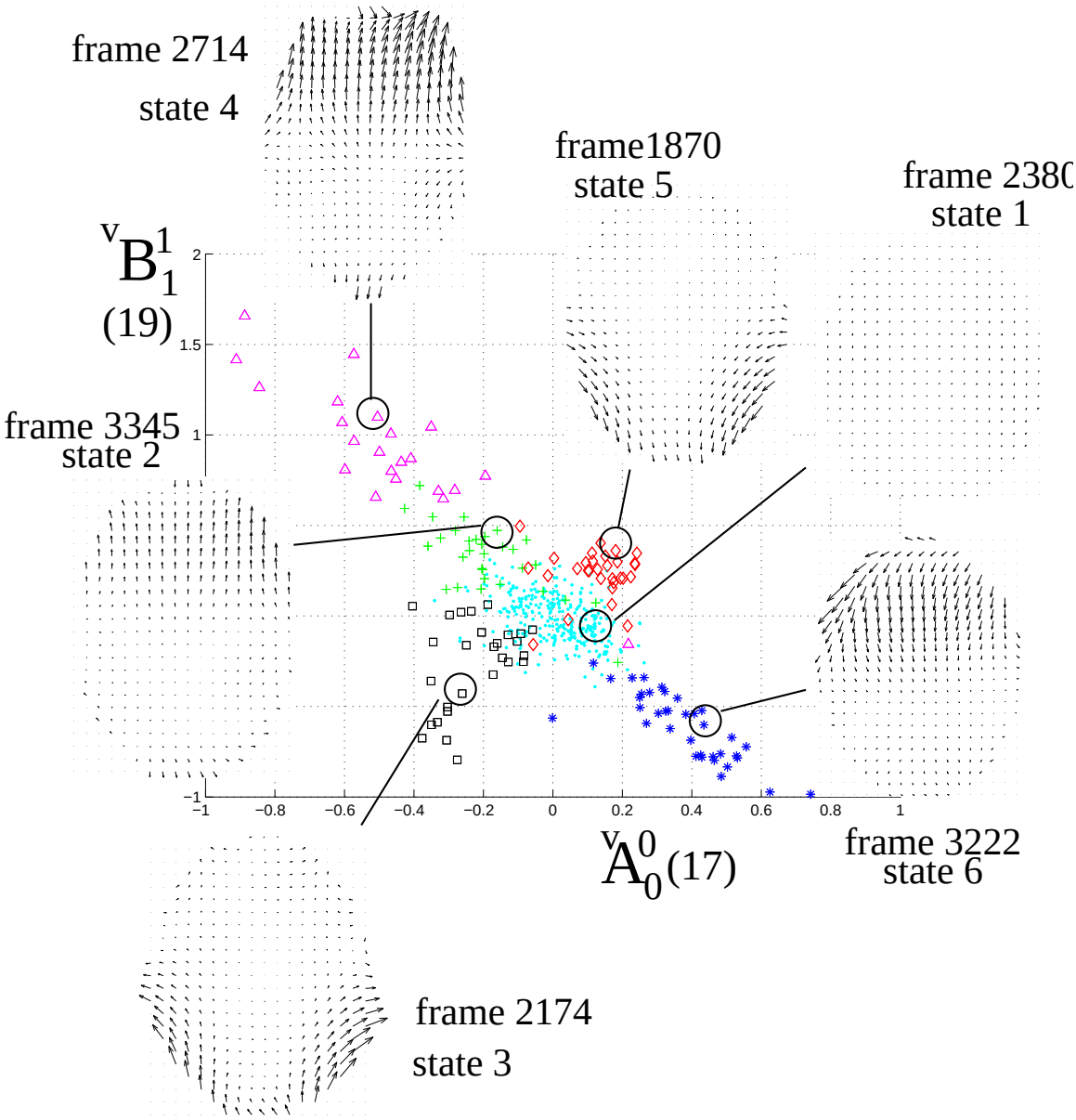


Figure 6.2: Clustering result for facial expressions along two most relevant features.

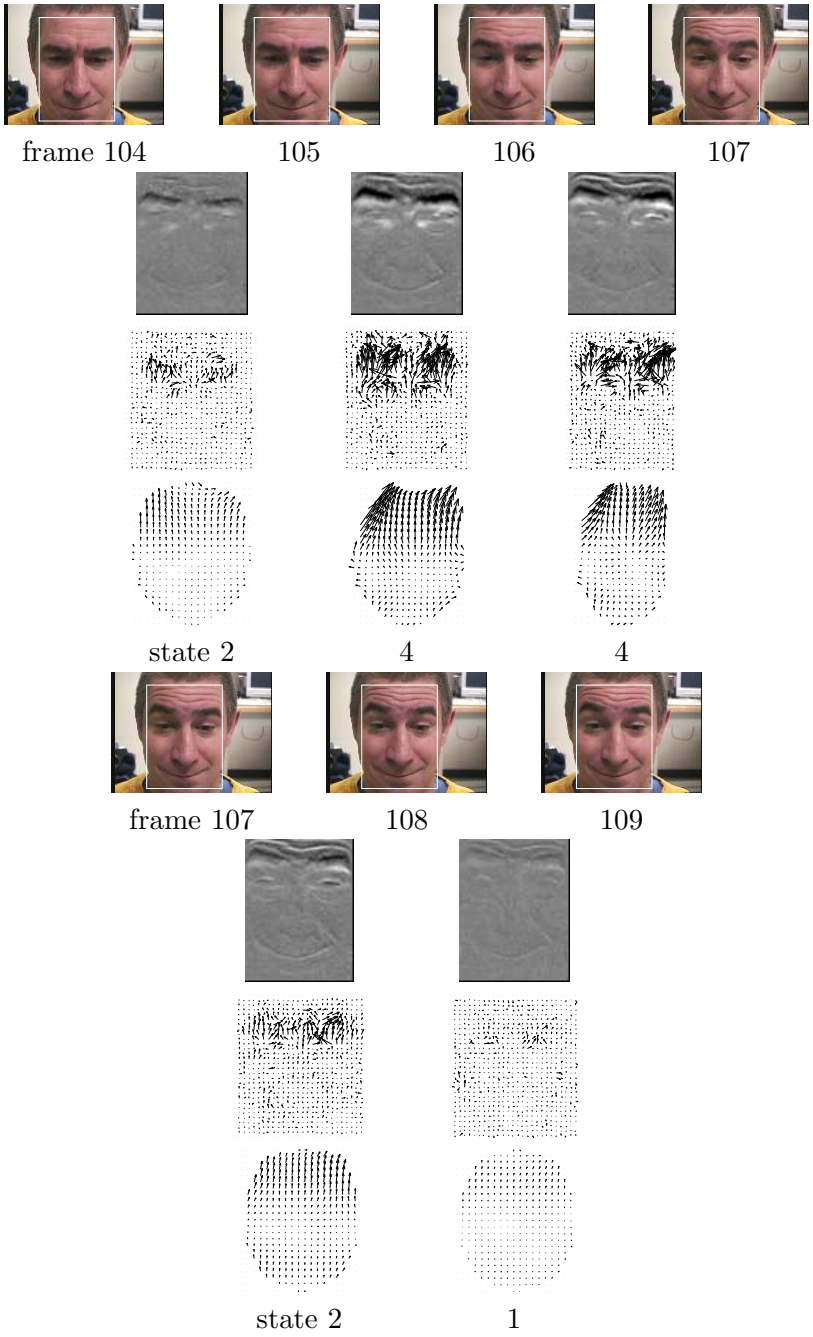


Figure 6.3: Eyebrow raising classified as states 2 and 4. The corresponding eyebrow lowering is shown in Figures 6.5 and 6.6.

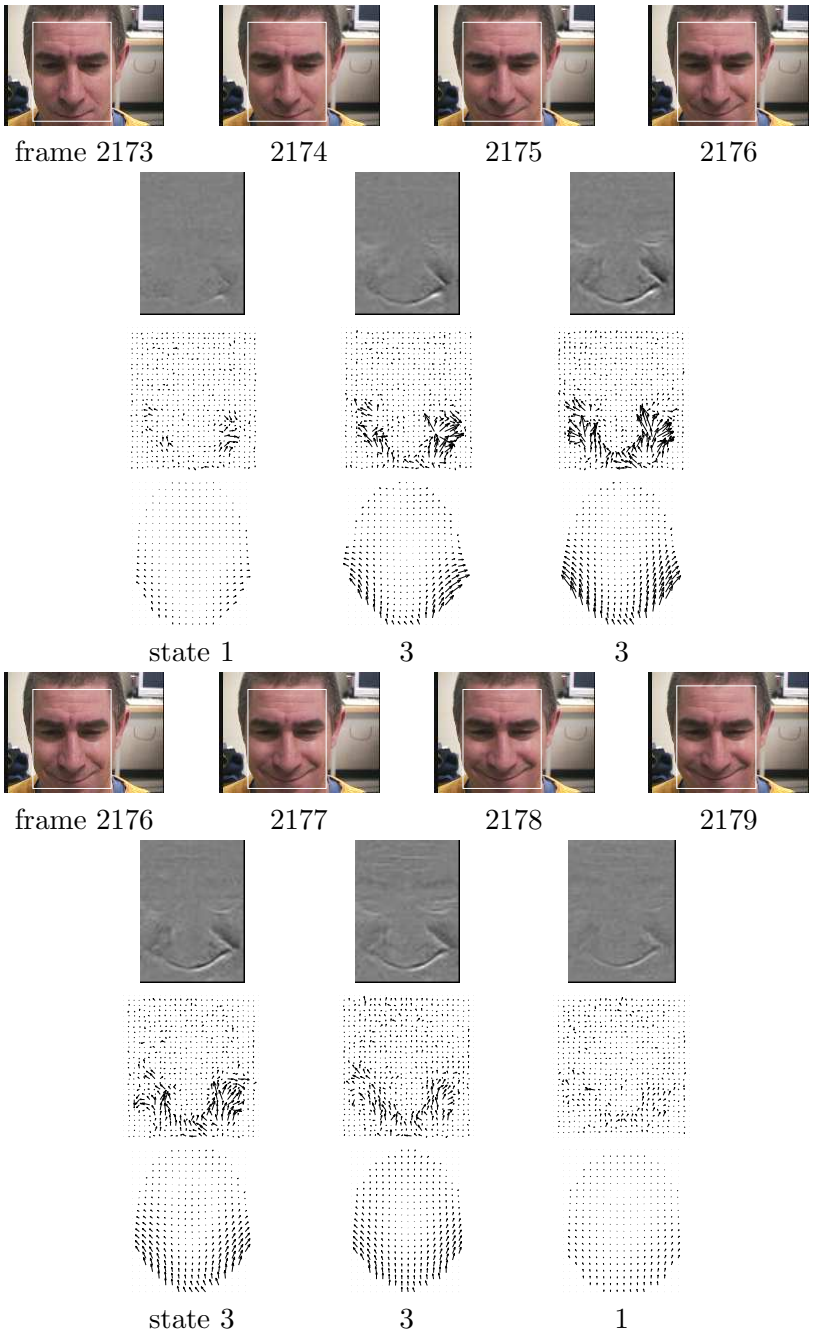


Figure 6.4: Smiling event classified as state 3, surrounded by no motion (state 1) events.

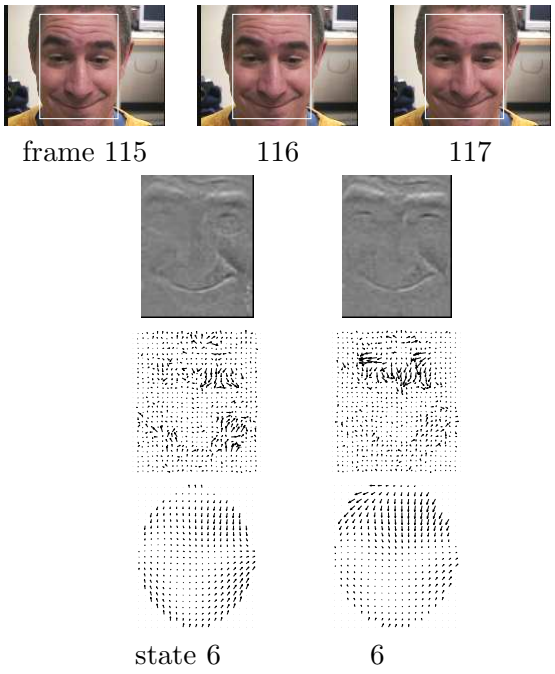


Figure 6.5: Eyebrow lowering event classified as state 6.

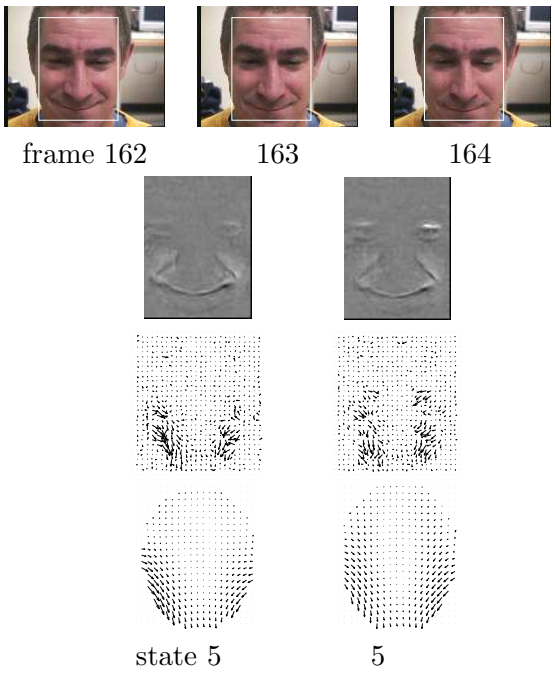


Figure 6.6: Smile returning to neutral classified as state 5.

6.1.2 Clustering Images

In this section, we show results from training the simple mixture of feature-weighted Gaussians on the pose information (projections of the images to the Zernike basis). Again we clustered the data from the imitation game. In this experiment, however, we use all 3600 frames for the training data.

We used the first 64 Zernike coefficients. The feature weighting parameters were set to $a = 1$, $b = 0.01$ and $\alpha = 34$. The parameters $\mu_{z,x}$ and $\Lambda_{z,x}$ were initialized by choosing K data points randomly as the initial seeds for K-means clustering, and Gaussian distributions were fit to the resulting classes. The feature weights τ_k^2 were all initialized to 1. The results were relatively insensitive to the initialization.

We trained a model with 8 classes. Figure 6.7 shows the final values of the feature weights, τ_k^2. The two most relevant features are A_4^4 and B_7^7, while $A_2^4, B_2^4, B_5^7, A_1^1$ and B_1^7 are also significant.

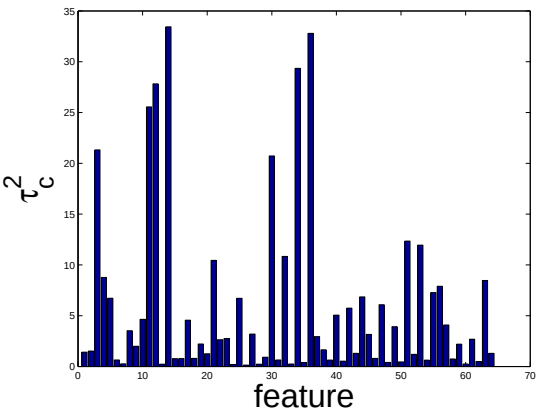


Figure 6.7: Feature weights learned for facial expression data.

Figure 6.8 shows the reconstructed Zernike vectors plotted along two of the relevant features (A_4^4, B_7^7). The clusters are denoted by the shape and color of the data points. Also shown are the means and level curves of the covarainces of the Gaussian output distributions. Reconstructed configurations from the mean Zernike feature vectors are also shown as grayscale images. The classes are roughly: (1) (3) and (8) surprised, (2) frown, (4) and (5) neutral, (6) smile eyebrow raised and (7) smile. Figure 6.9 show the classification of the same frames shown in Figure 6.3, taken during a smiling event. The frames with the subject in a smiling pose are classified as state 6, which we have previously recognized as a smiling pose.

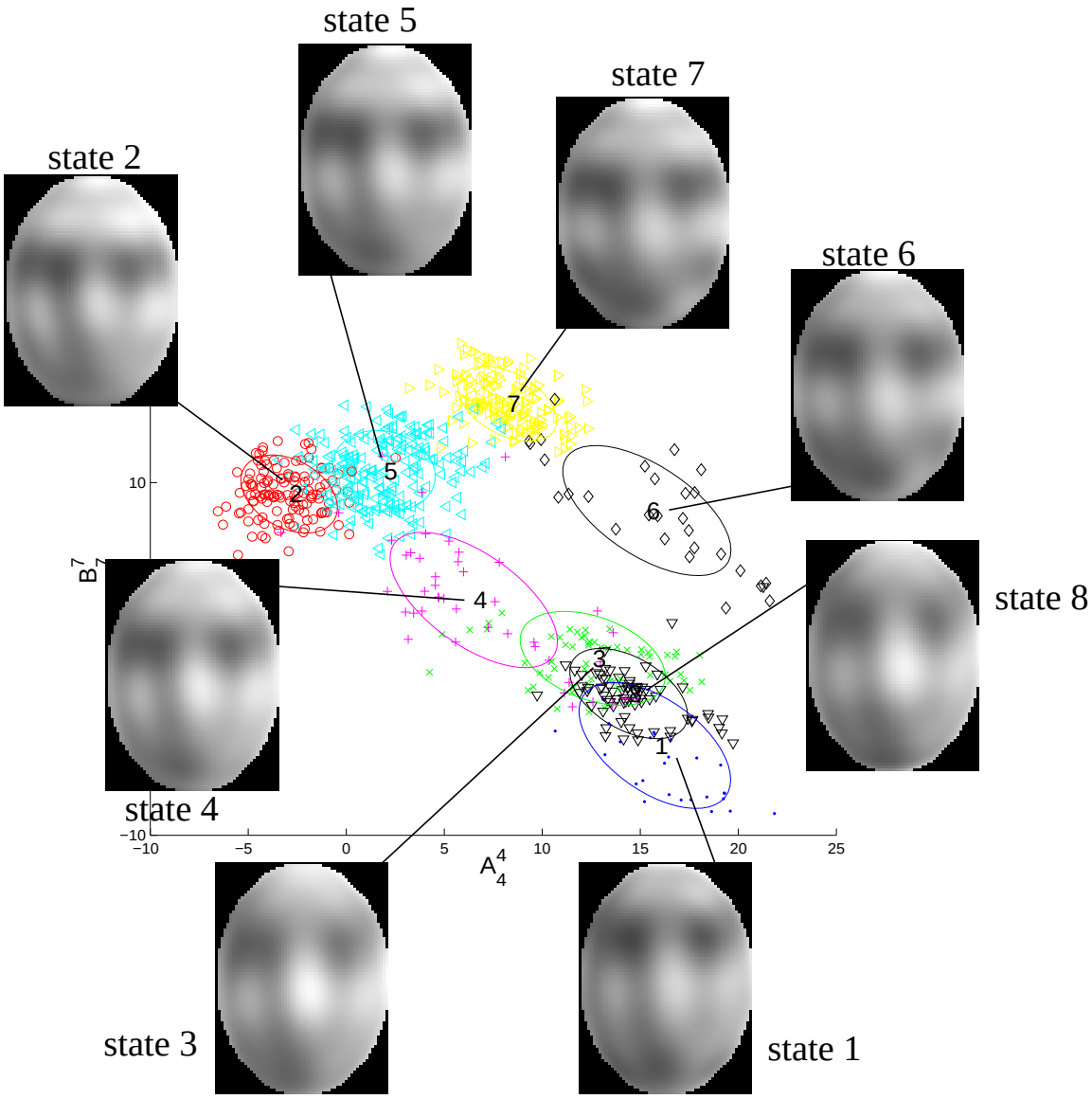


Figure 6.8: Clustering result for facial expressions along two most relevant features. The 8 means are numbered, and their associated covariances are shown as level curves. Reconstructed images for the mean Zernike feature vectors are shown.

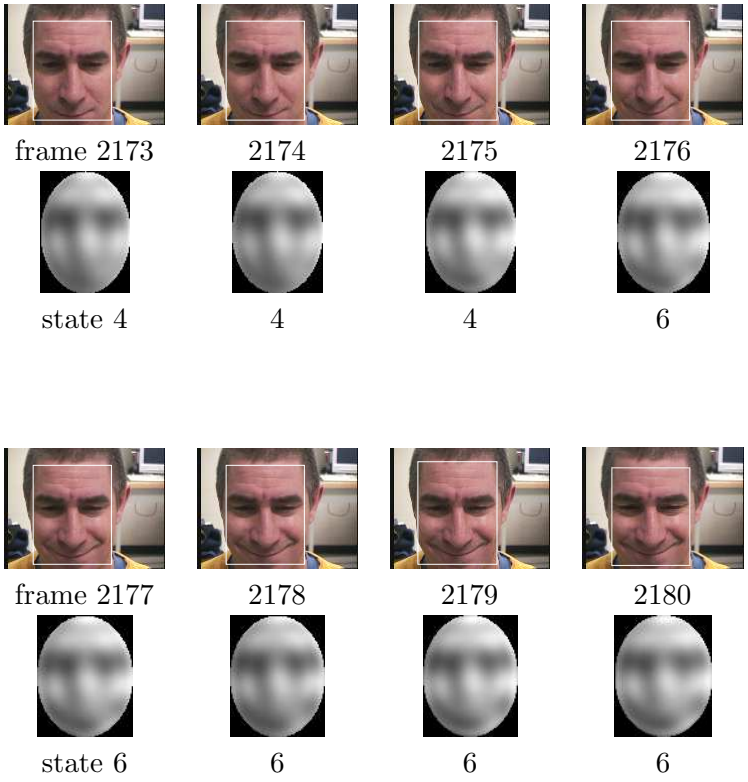


Figure 6.9: Smiling pose classified as state 6. The original frames with tracked facial region are shown, along with the expected reconstructions given the simple mixture model. The most likely states of the mixture are given below the reconstructions.

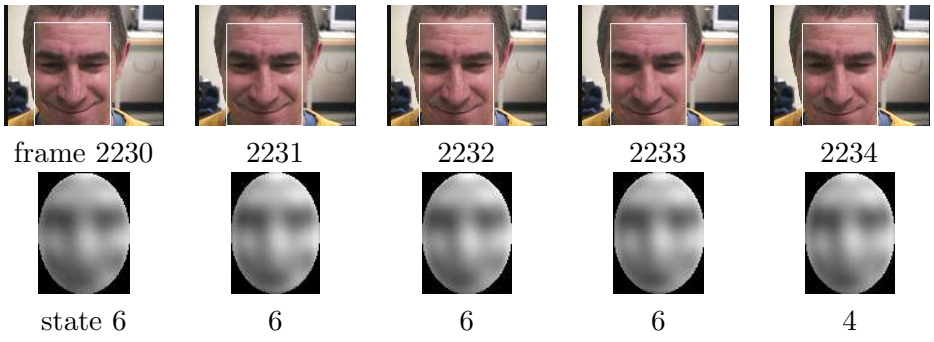


Figure 6.10: Return from smiling pose classified as state 4.

	cluster			
	D_1	D_2	D_3	D_4
C_1	0.01	0.01	0.61	.37
C_2	0.02	0.48	0.17	0.33
C_3	0.01	0.97	0.01	0.01
C_4	0.91	0.07	0.01	0.01

Table 6.1: Probability distribution $P(A^{b:a}|A^a)$ learned for subject $\mathcal{A}$

6.1.3 Clustering Display Sequences

The videos from the imitation game described in the last section were temporally segmented using the onset times of the cartoon displays and the resulting sequences were input to the mixture of coupled HMM clustering and training algorithm described in Section 4.6 using 4 clusters (the number of displays the subjects were trying to imitate). The Viterbi algorithm was used to assign cluster membership, D, to each sequence, which were then compared to the known classes of displays the subjects were trying to imitate. The exact model recovered is partially dependent on the randomness in the initialization procedure. However, we found the recovered cluster membership to be fairly consistent, an indication that our initialization procedure is robust to the random starting points. The results we present in the following section are usually the results we obtained on the first trial. Some, however, were given multiple trials and the most often re-occurring results are presented.

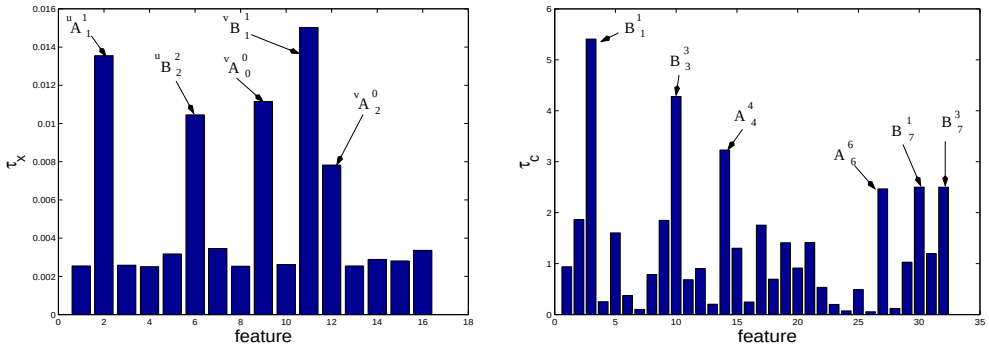
The remainder of this section evaluates the results from one of the subjects who performed the experiment. The results from this subject are demonstrative of the results from the other subjects. We first show the learned high-level probability distribution, $P(A^{b:a}|A^a)$, which describes the likelihood of observing each high-level motion state given each cartoon display. We then show two of the models learned for this subject: one for "smiling" imitations, and one for "surprised" imitations. For each model, we show the learned feature weights and output distributions for both dynamics and configuration chains. We also show how it analyses two sequences.

Table 6.1 shows the learned model parameter, $P(A^{b:a}|A^a)$, for subject $\mathcal{A}$, in which each row is one A^a state (cartoon display on screen) and each column is one recovered cluster, $A_1^{b:a}...A_4^{b:a}$. To simplify notation, we use $C \equiv A^a$ and $D \equiv A^{b:a}$. We see that most of the responses to the cartoon display C_4 were classified as D_1, and most of the responses to C_3 were classified as D_2. Responses to C_2 were split between those that looked the same as responses to C_3 (and so were classified as D_2, and those that looked similar to some of the responses to C_1 (classified together in state D_4. The D_3 model classified the majority of the responses to C_1. In the

following sections, we describe each model, and show some sequences which were classified as belonging to that model.

Model D_1

Feature weights for the model 1 dynamics and configuration chains are shown in Figure 6.11. The dynamics chain has four significant features: two in the horizontal flow components: ${}^{u}A_1^1, {}^{v}B_2^2$, and three in the vertical flow components, ${}^{v}A_0^0, {}^{v}B_1^1$, and ${}^{v}A_2^0$. The three most significant features in the configuration chain are B_1^1, B_3^3, and A_4^4.



dynamics configuration

Figure 6.11: Feature weights τ_k^2 for model 1 dynamics and configurations chains.

The output distributions of the four states (X) in the dynamics chain are shown in Figure 6.12, plotted along two most significant feature dimensions, ${}^{u}A_1^1$ and ${}^{v}B_1^1$. Two states ($X = 2, 4$) correspond to no motion (the face is stationary), while the other two correspond to expansion upwards and outwards in the bottom of the face region ($X = 1$), and contraction downwards and inwards in the bottom of the face region ($X = 3$). We will see that these states correspond to the expansion and relaxation phase of smiling.

The output distributions of the five configuration states are shown in Figure 6.13. There are two states ($C = 2$ and $C = 4$) which describe the face in a fairly relaxed pose, while $C = 1$ and $C = 3$ describe "smiling" configurations.

We now show how the C3MG model analyses a particular sequence from the imitation game for which the most likely high level state is $D = 1$. We show the expected feature vector for each frame in the dynamics and configuration chains, Z_x and Z_w, respectively, and the expected output flow fields and poses. Figure 6.14 shows the trajectory of the reconstructed dynamics Zernike vector (Z_x) along the

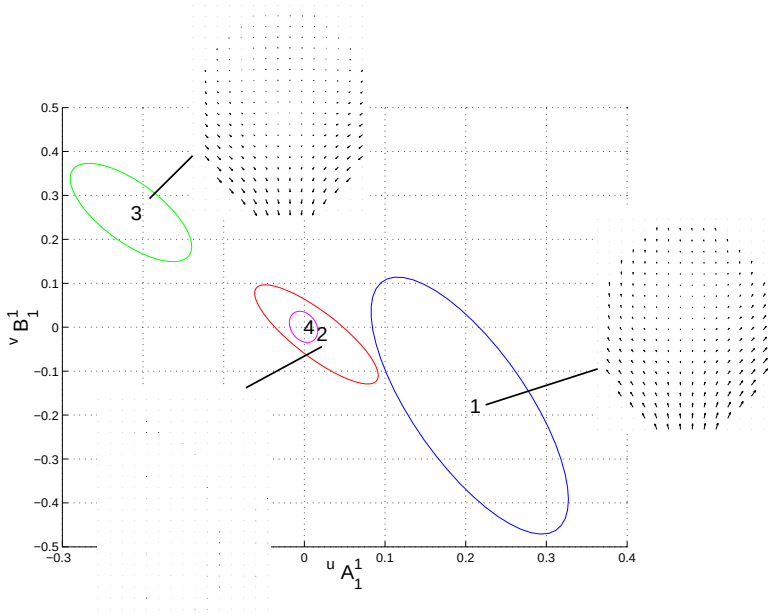


Figure 6.12: Dynamics chain model 1 output states plotted along two most significant dimensions according to feature weights, ${}^{u}A_1^1, {}^{v}B_1^1$. Reconstructed flow fields for X state means are also shown.

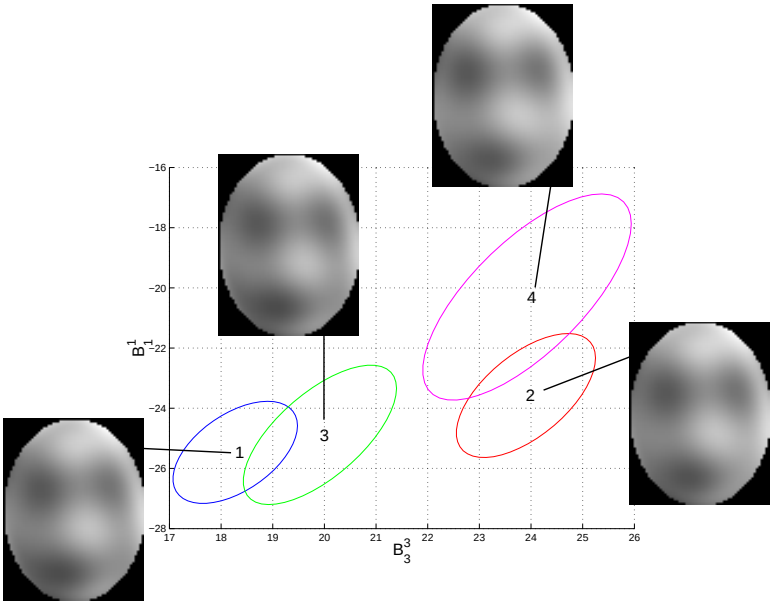


Figure 6.13: Configuration chain model 1 output distributions. Reconstructed grayscale images are shown for C state means.

two most important feature vector dimensions for a sequence classified as model $D = 1$. We see the sequence starts in state 2 (no motion), enters state 1 (smile expansion), then back to state 2 (holding the smile), finally contracting (state 3) back to a relaxed pose (state 1 again).

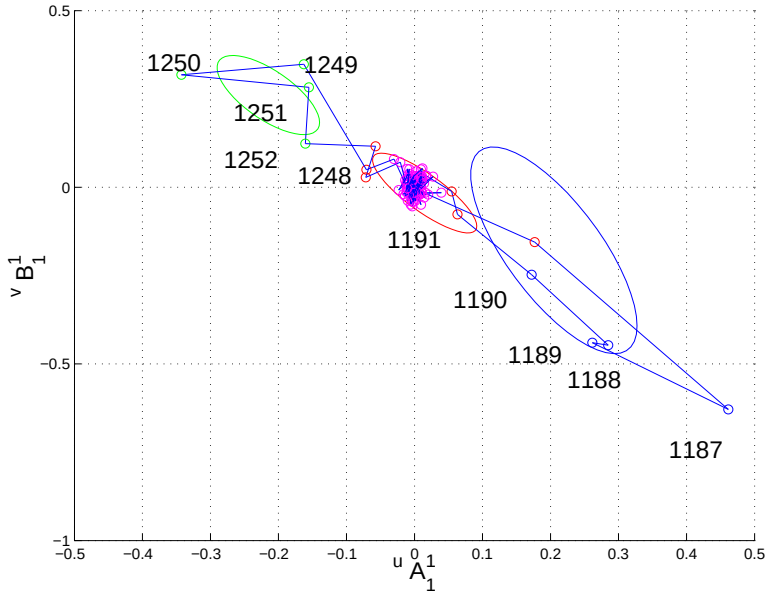


Figure 6.14: Trajectory of dynamics feature vector (Z_x) for sequence 13 (for which model 1 has highest posterior) is shown given model 1. The level curves of the Gaussian output covariances for model 1 are also shown.

Figure 6.15 shows the trajectory of the reconstructed configuration Zernike vector (Z_c) along the two most important feature vector dimensions for a sequence classified as model $D = 1$. We see that the face is in the relaxed pose, $W = 2$, at the beginning and the end of the sequence, and in the "smiling" pose, $W = 1$ and $W = 3$, in the middle from frames 1189 to 1250.

Figures 6.16 and 6.17 show model's explanation of the same sequence, for the frames indicated in Figures 6.14 and 6.15. We see the high level distribution over D is peaked at $D = 1$. Distributions over dynamics and configuration chains show which state is most likely at each frame. The expected pose, H, and flow field, V, are shown conditioning the image, I, and the temporal derivative, f_t, respectively.

Model D_2

Feature weights for the model 2 dynamics and configuration chains are shown in Figure 6.18. The dynamics chain has three significant features, all in the vertical

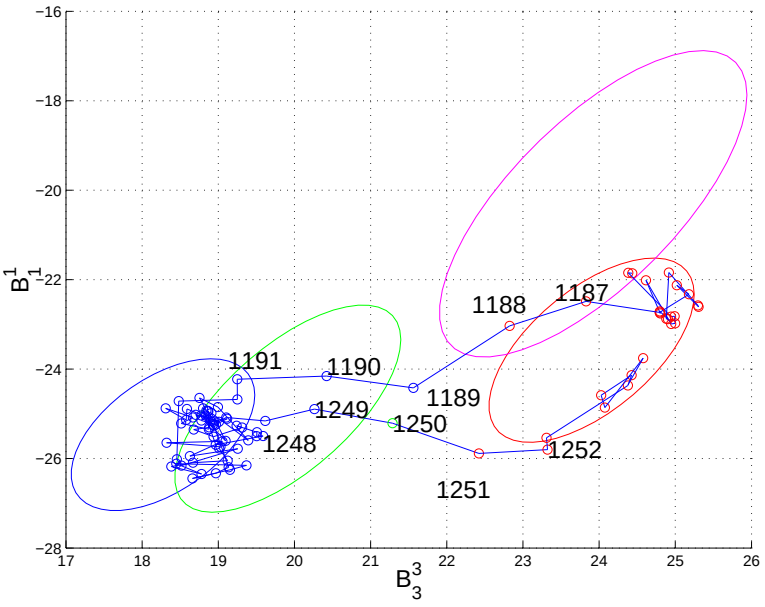


Figure 6.15: Trajectory of configuration feature vector (Z_c) for sequence 13 (for which model 1 has highest posterior) is shown given model 1. The level curves of the Gaussian output covariances for model 1 are also shown.

flow components: ${}^{v}A_0^0, {}^{v}B_1^1$, and ${}^{v}B_3^1$. The three most significant features in the configuration chain are B_1^1, A_4^0, and A_4^4.

The output distributions of the seven states (X) in the dynamics chain are shown in Figure 6.19, plotted along two most significant feature dimensions, ${}^{v}A_0^0$ and ${}^{v}B_1^1$. States $X = 2$, $X = 3$ and $X = 6$ all describe little or no motion flows. State $X = 1$ describes flows upwards in the upper part of the image, roughly corresponding to eyebrows raising. State $X = 5$ also includes upwards motion in the upper part of the image, but also includes motion downwards and inwards in the lower part of the image, corresponding to eyebrows raising and jaw dropping at the same time. State $X = 4$ describes motion downwards in the upper part of the image, with slight upwards motion in the lower part of the image, corresponding to the face relaxing from an expanded state. State $X = 7$ describes small motions downwards in the upper part of the image.

The output distributions of the five configuration states are shown in Figure 6.20. There are two states ($C = 1$ and $C = 5$) which describe the face in a fairly relaxed pose, while $C = 2$ and $C = 3$ describe configurations in which the eyebrows are raised and the jaw is dropped. State $C = 4$ appears to be a "smiling" pose.

We now show how the C3MG model analyses a particular sequence from the

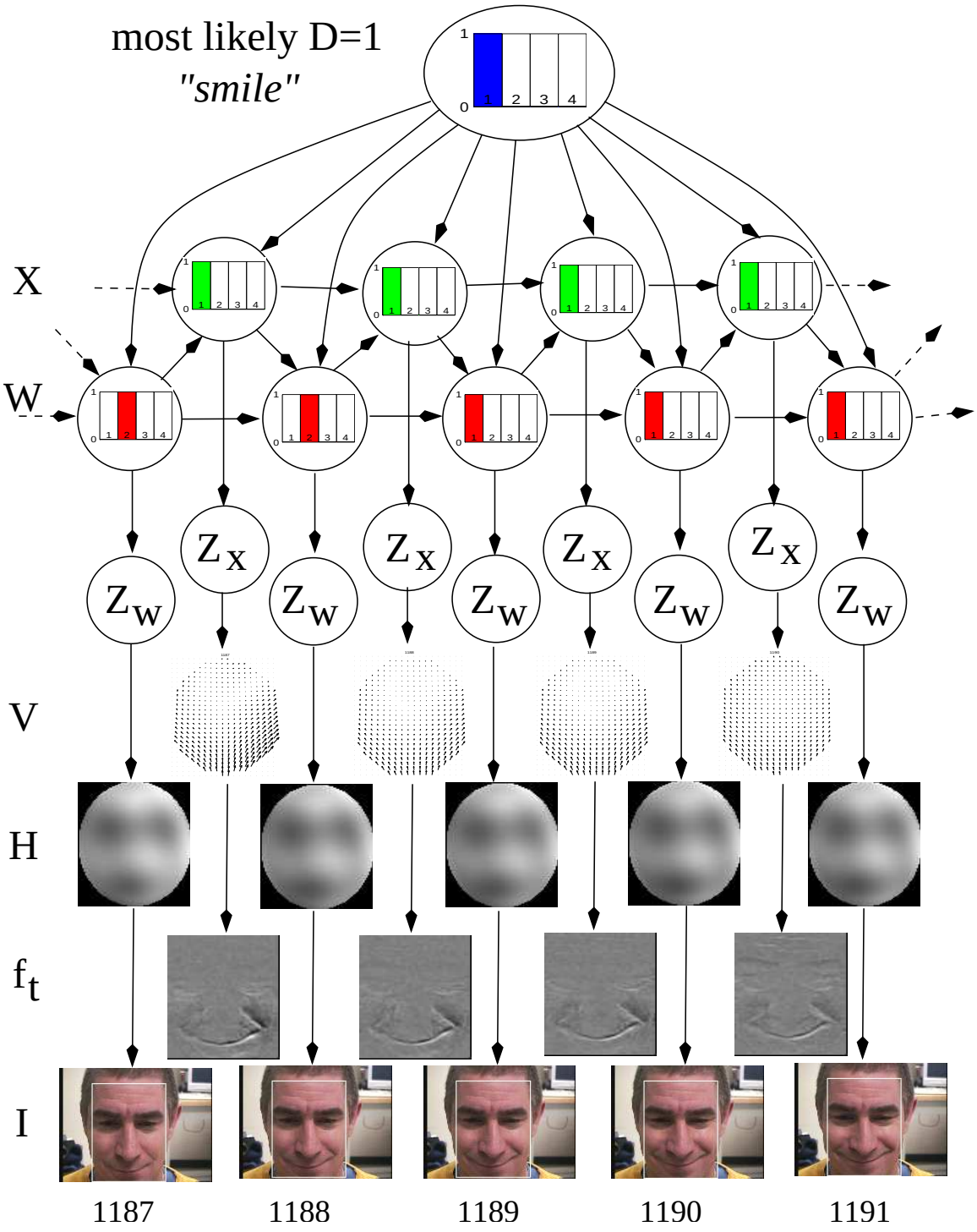


Figure 6.16: The face starts in its rest configuration ($W = 2$), and expands into a smiling pose ($W = 1$) with the $X = 1$ flow field.

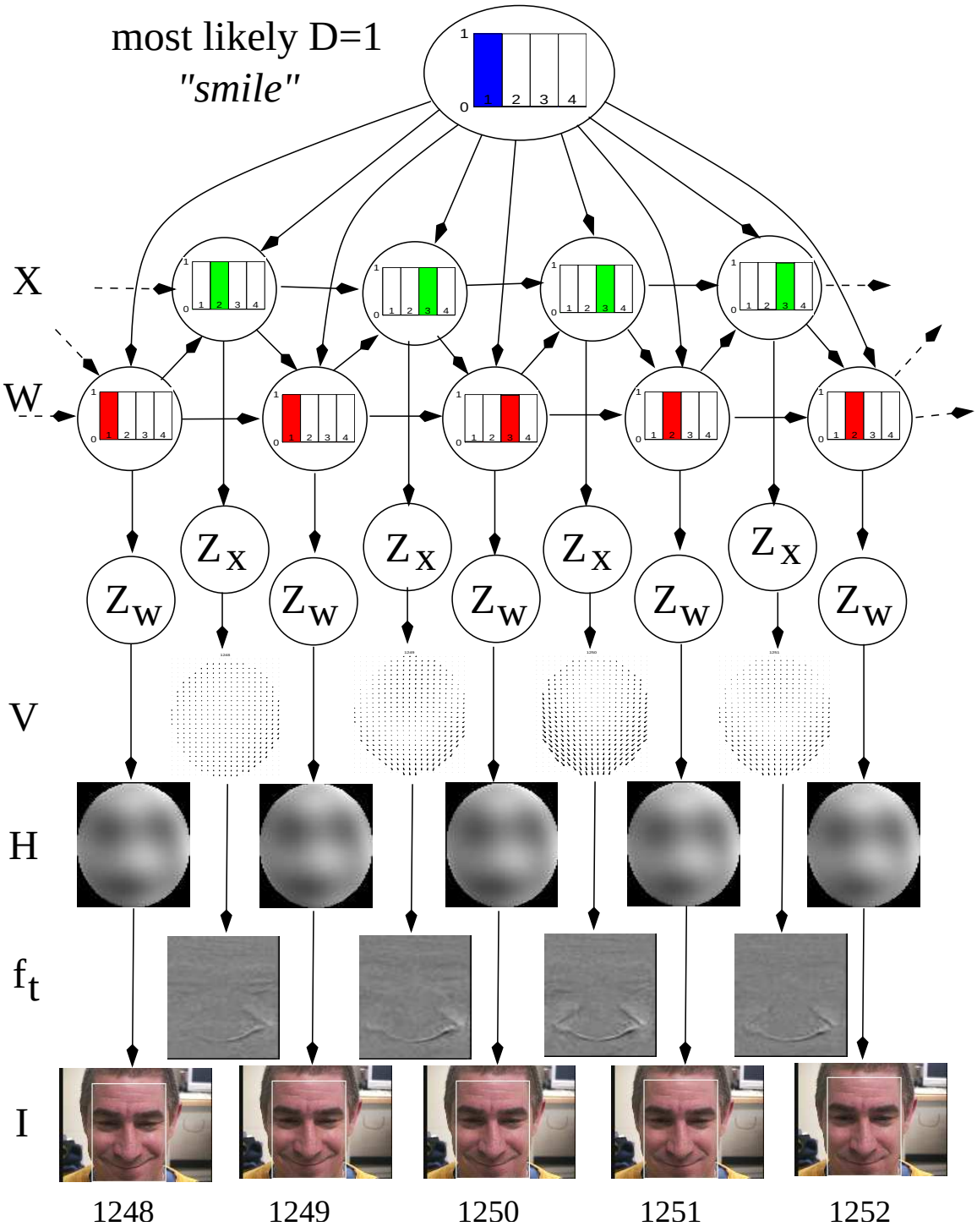
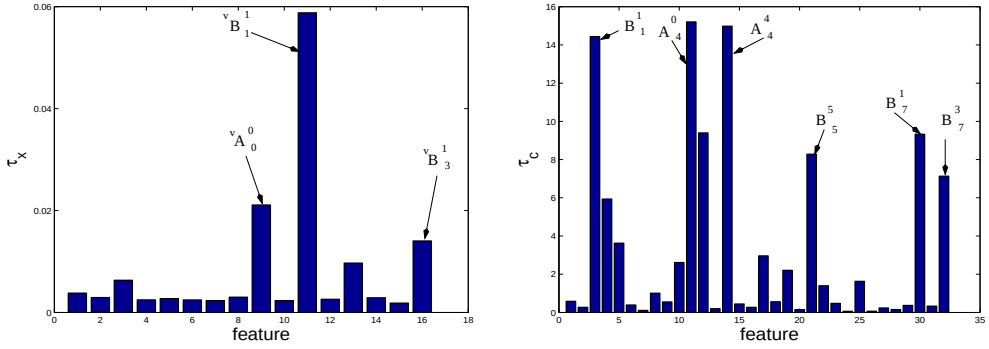


Figure 6.17: The face starts in a smiling ($W = 1$) pose, and contracts with the $X = 2$ flow field to a rest configuration ($W = 2$).



dynamics configuration

Figure 6.18: Feature weights τ_k^2 for model 2 dynamics and configuration chains.

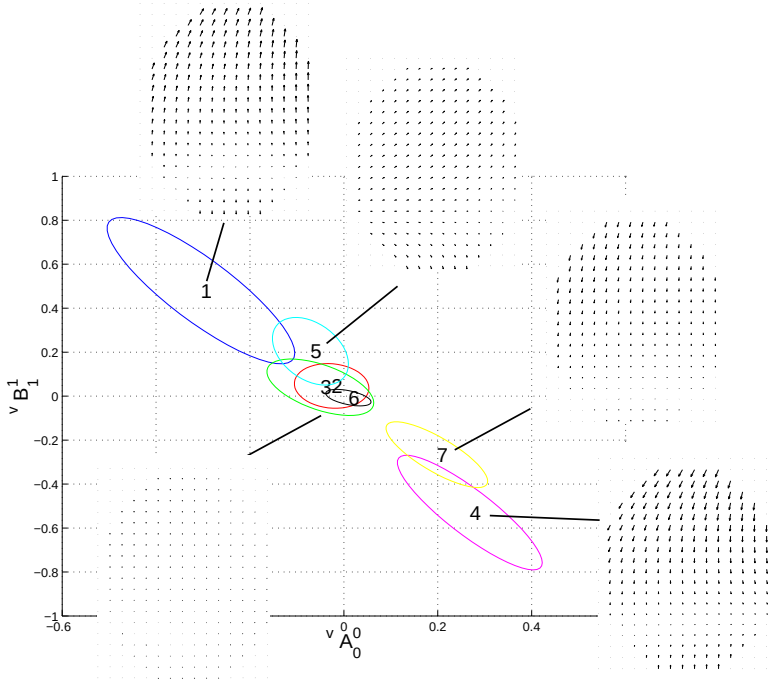


Figure 6.19: Dynamics chain model 2 output states plotted along two most significant dimensions according to feature weights, ${}^vA_0^0, {}^vB_1^1$. Reconstructed flow fields for X state means are also shown.

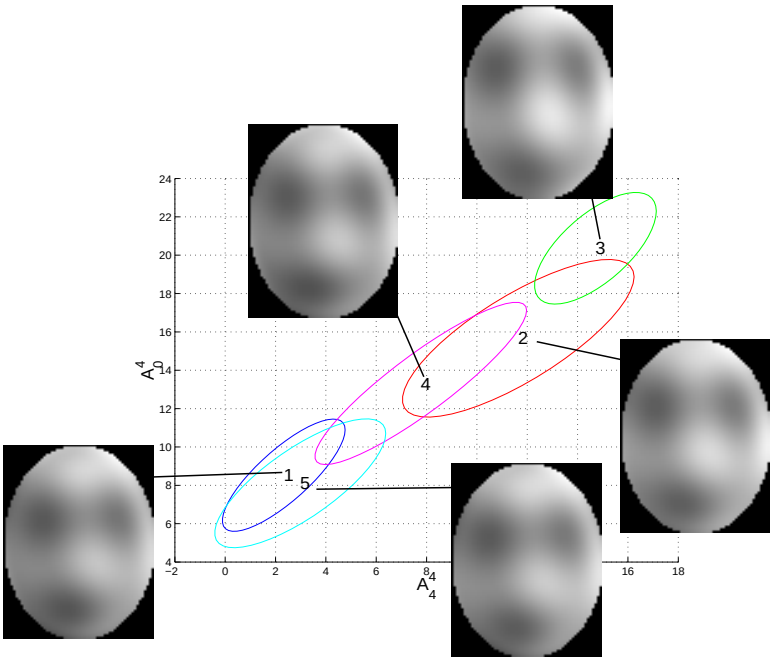


Figure 6.20: Configuration chain model 2 output distributions. Reconstructed grayscale images are shown for C state means.

imitation game for which the most likely high level state is $D = 2$. We show the expected feature vector for each frame in the dynamics and configuration chains, Z_x and Z_w, respectively, and the expected output flow fields and poses. Figure 6.21 shows the trajectory of the reconstructed dynamics Zernike vector (Z_x) along the two most important feature vector dimensions for a sequence classified as model $D = 2$. We see the sequence starts with no motion, enters state 1 (eyebrow raise), then back to no motion (holding the eyebrows raised), finally contracting (state 4) back to a relaxed pose.

Figure 6.22 shows the trajectory of the reconstructed configuration Zernike vector (Z_c) along the two most important feature vector dimensions for a sequence classified as model $D = 2$. We see that the face is in the relaxed pose, $W = 5$, at the beginning and the end of the sequence, and in the "eyebrow raised" pose, $W = 2$ and $W = 3$, in the middle from frames 1002 to 1059.

Figures 6.23 through 6.24 show the model's explanation of the same sequence, for the frames indicated in Figures 6.21 and 6.22. We see the high level distribution over D is peaked at $D = 2$. Distributions over dynamics and configuration chains show which state is most likely at each frame. The expected pose, H, and flow field, V, are shown conditioning the image, I, and the temporal derivative, f_t, respectively.

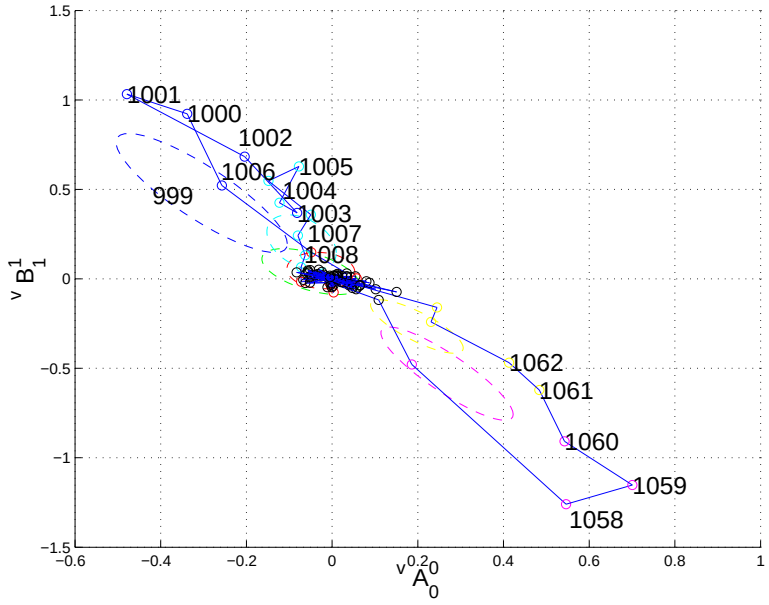


Figure 6.21: Trajectory of dynamics feature vector (Z_x) for sequence 11 (for which model 2 has highest posterior) is shown given model 2. The level curves of the Gaussian output covariances for model 2 are also shown.

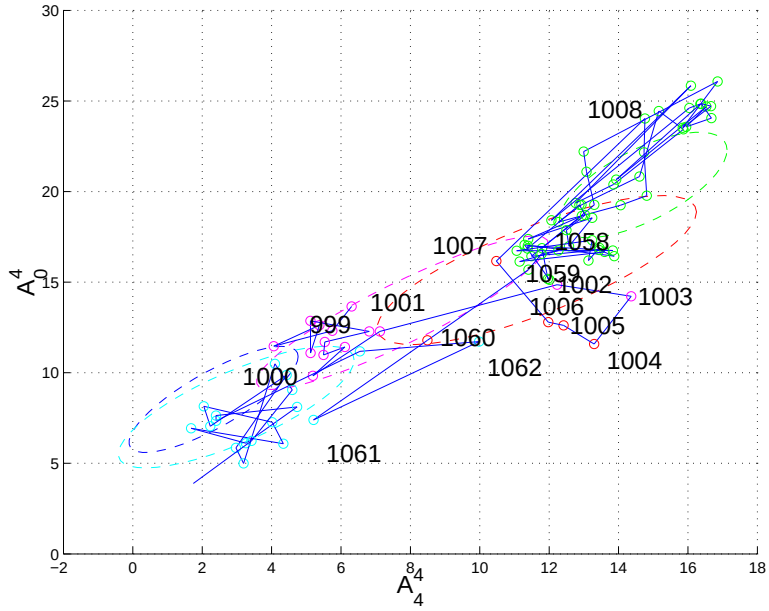


Figure 6.22: Trajectory of configuration feature vector (Z_c) for sequence 11 (for which model 2 has highest posterior) is shown given model 2. The level curves of the Gaussian output covariances for model 2 are also shown.

		Subject A model inference				Subject B model inference				Subject C model inference			
		1	2	3	4	1	2	3	4	1	2	3	4
actual	1	23	0	0	1	20	8	3	2	0	6	6	6
	2	5	2	9	2	0	12	0	6	9	31	8	0
	3	0	4	32	0	0	0	40	5	1	1	16	6
	4	1	0	4	37	0	4	3	17	0	0	3	27
success:		78%				74%				62%			

Table 6.2: Confusion matrices and success rates from cross-validation experiments inferring cartoon displays from sequences of three subjects.

6.1.4 Inferring Agent Actions

The preceding section gave a qualitative analysis of one subject's displays in the imitation game, and the models we could learn from a set of training data. We can obtain quantitative results by attempting to infer the cartoon display given the human imitation based on the learned model, as described in Section 5.5. We did this analysis using a leave-one-out cross validation experiment for each of three subjects who participated in the imitation game. There were 40 sequences (imitations) for each subject, one of which was removed. The remaining 39 sequences were used to train the C3MG as in the last section. The learned model was used to infer the cartoon display, A^a, from the remaining (left-out) sequence. The most likely value was chosen and compared to the actual display. This process was repeated three times for each subject with different random initializations. The cross-validation is only performed within each subject's data pool, since the models are designed to be person dependent. Table 6.2 shows the confusion matrices for the three subjects (summed over the three experiments), and the total success rate. The majority of the mis-classifications are from display a_2, which subjects reported as being difficult to distinguish from a_1. The models make quite accurate predictions of a_3 (83%) and a_4 (84%).

However, it is important to note that these results are obtained by always looking at the most likely cartoon display, which ignores the model's explicit representation of uncertainty. That is, we estimate $P(A^a|\mathbf{O})$, which is a *distribution* over A^a, and the peak of the distribution is used in Table 6.2. However, there are some cases where there is a second display which is nearly as likely as the best one. To demonstrate this, Table 6.3 shows the confusion matrices obtained if we classify

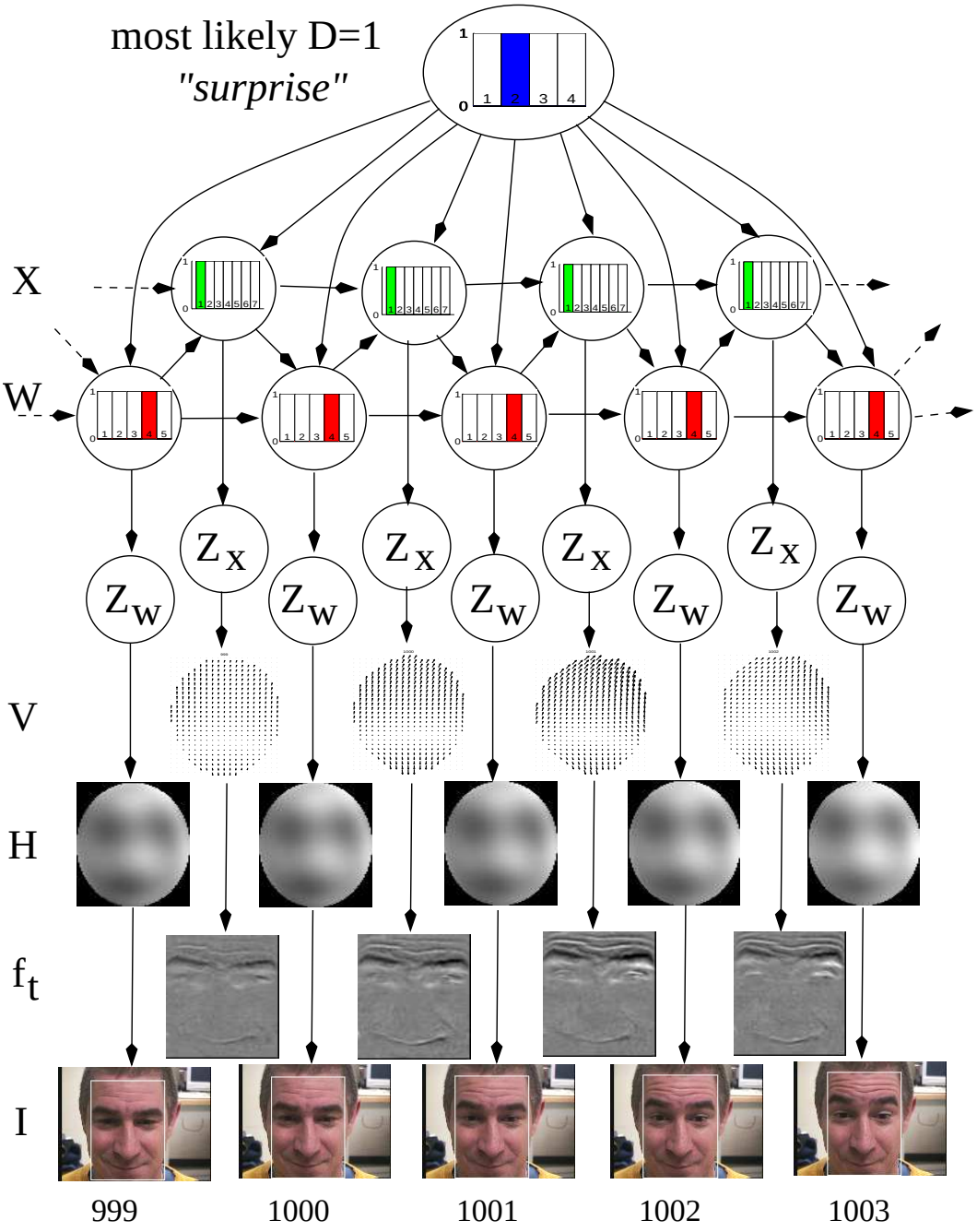


Figure 6.23: The face starts in its rest configuration ($W = 4$), expands with the $X = 1$ flow field towards a "eyebrows raised" ($W = 2$) configuration (see Figure 6.24).

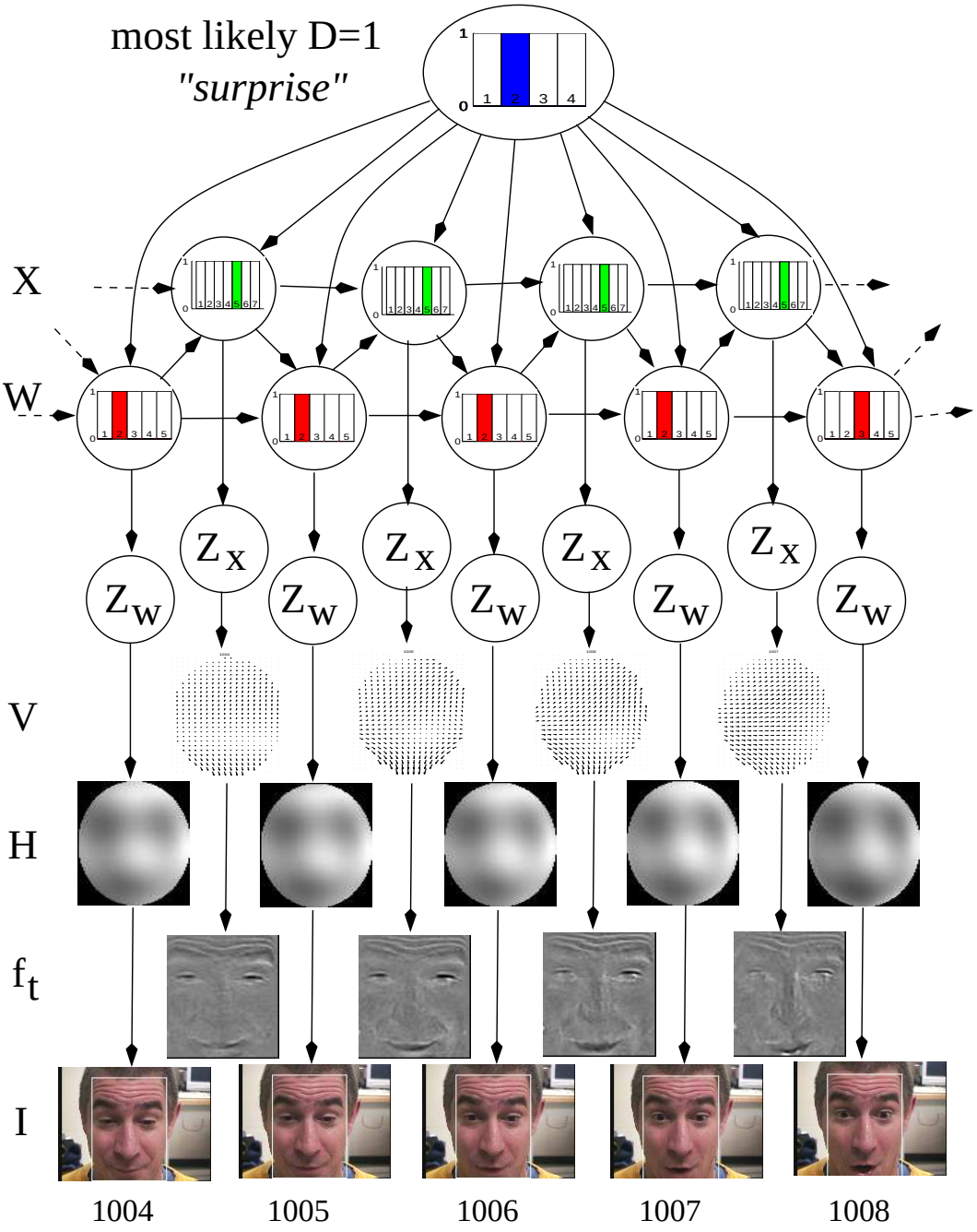


Figure 6.24: The face starts in an "eyebrows raised" pose ($W = 2$), and the mouth opens with the $X = 5$ flow field, resulting in a "surprised" configuration ($W = 3$).

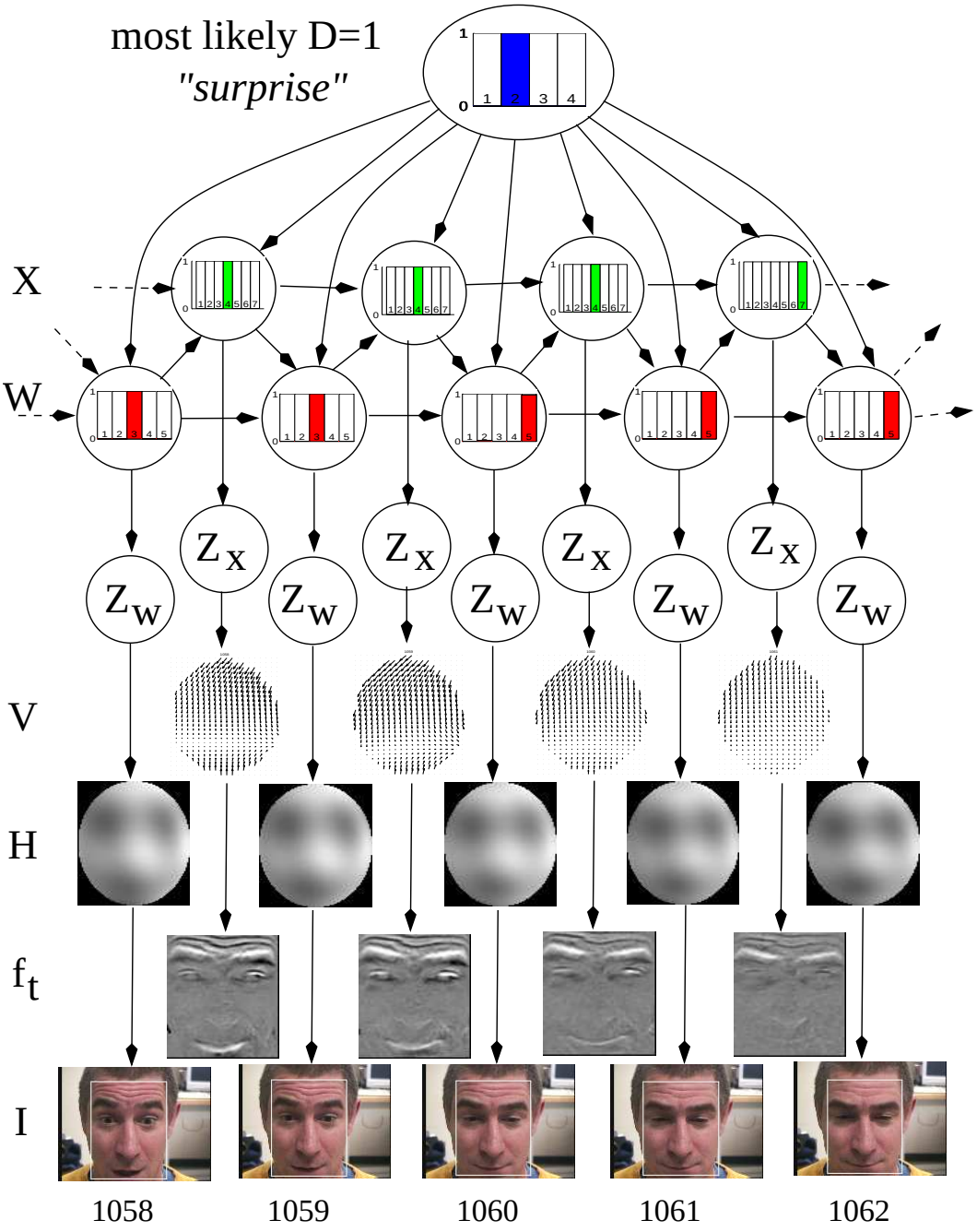


Figure 6.25: The face starts in a "surprised" configuration ($W = 3$), and relaxes with the $X = 4$ flow field to a rest pose ($W = 5$).

		Subject A model inference				Subject B model inference				Subject C model inference			
		1	2	3	4	1	2	3	4	1	2	3	4
actual	1	24	0	0	0	26	2	3	2	9	0	3	6
	2	1	14	1	2	0	18	0	0	0	41	7	0
	3	0	0	36	0	0	0	45	0	1	1	21	1
	4	1	0	1	40	0	1	0	23	0	0	0	30
success:		95%				93%				84%			

Table 6.3: Confusion matrices and success rates from cross-validation experiments inferring cartoon displays from sequences of three subjects, using the top two most likely cartoon displays if the most likely is uncertain.

		Subject A model inference				Subject B model inference				Subject C model inference			
		1	2	3	4	1	2	3	4	1	2	3	4
actual	1	6	2	0	0	10	0	1	0	0	4	1	1
	2	2	1	3	0	0	6	0	0	0	15	1	0
	3	0	0	12	0	0	0	15	0	0	0	8	0
	4	0	0	0	14	0	0	0	8	0	0	0	10
success:		82%				97%				82%			

Table 6.4: Confusion matrices and success rates from supervised experiments inferring cartoon displays from sequences of three subjects.

the sequence correctly if it falls in the top two most likely displays, but only if the probability of the most likely display is less than 0.5. We see that many of the mis-classified sequences were assigned maximum likelihood with much uncertainty. These results can be compared to the results obtained in a supervised experiment, where each sequence is explicitly labeled, so D is observed. These results are shown in Table 6.4. When compared to the unsupervised maximimum likelihood experiments (Table 6.2), we see that the supervised models perform substantially better for two subjects (B and C), but only slightly better for subject A. This is as expected (better performance in the supervised experiments), but the unsupervised models perform well, in particular when the probability distribution is taken into account (Table 6.3), in which case the unsupervised models outperform the supervised ones.

6.2 Robot Control Gestures

We recorded a set of examples of four hand gestures, designed for simple robotic direction control: *forwards*, *stop*, *go left* and *go right*. A dozen examples of each gesture were performed by a single subject in front of a stationary camera during a training session. Video was grabbed from a IEEE 1394 (Firewire) camera at 150×150 with a narrow field of view. The region of interest was taken to be the entire image, and so no tracking was required. Clearly, this would only be possible with a static camera. Sequences were taken of a fixed length of 90 frames. A robotic agent (not embodied at this stage) chose actions in response to each gesture according to a random policy, and was rewarded by the operators *good* or *bad* action, *Aact*, for choosing the correct action.

The gestures were performed in sequence, and the subject practiced performing the gestures, so that they would be as similar as possible. It is important to re-state that these experiments are not meant to demonstrate gesture recognition. It is clear that, with this simple tracking and registration method (taking the whole image), this system would not deal with the high variability in gesture orientation or speed. These experiments are meant as a simple demonstrate the value-directed structure learning techniques: they show how our system can correctly discover the number of *meaningful* gestures in a simple interaction. They are also meant to demonstrate that our system can be easily applied to more than just facial displays: our representation can deal with the complexity of a gestural motion.

We trained the POMDP with $N_a = 6$ states. The value function and policy are shown in Figure 6.26 as decision diagrams. The policies for states d_2 and d_5 are equivalent and their values are identical, and so the value-directed structure learning algorithm merges them first by simply deleting state d_5. The POMDP is re-trained, resulting in a five-state value function (not shown), in which two more states are found to agree and are merged. Again the POMDP is re-trained, this time giving a value function and policy in which no displays are found to be redundant, shown in Figure 6.27.

Figures 6.28 and 6.29 show the final 4-state model's explanation of two parts of a *stop* gesture sequence. We see the high level distribution over D is peaked at $D = 3$. Distributions over dynamics and configuration chains show which state is most likely at each frame. The expected pose, H, and flow field, V, are shown conditioning the image, I, and the temporal derivative, f_t, respectively. Stop gestures consist of an expansion phase (Figure 6.28) followed by a retraction phase (Figure 6.29).

Figures 6.30 and 6.31 show the model's interpretation of a part of a *forwards* sequence, classified as model d_4, in the new 4-state POMDP. Forwards gestures consist of one or more iterations of the motions shown: the hand moves forwards

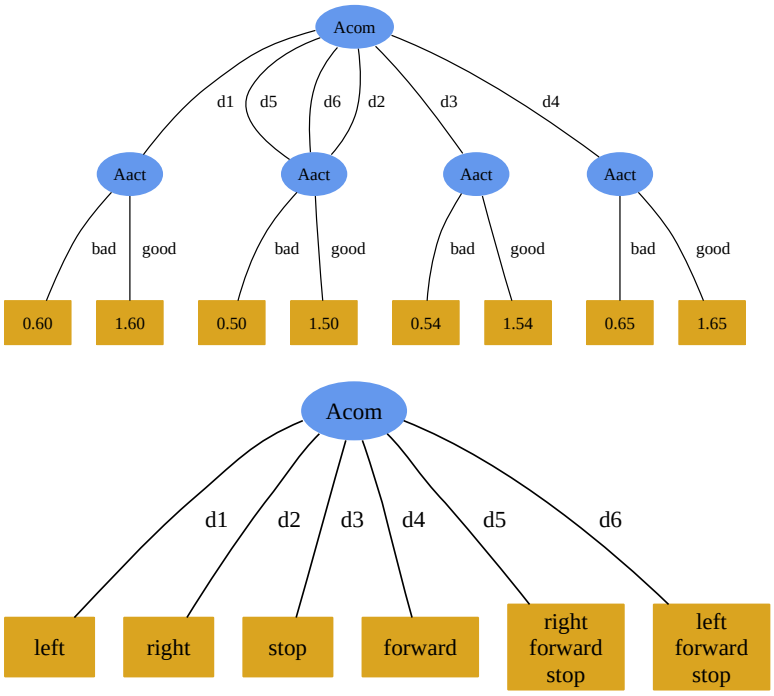


Figure 6.26: Original six-state value function (top) and policy (bottom), shown as decision diagrams. States are the labels on each path from the root to a leaf, which contains the value or optimal action for that state.

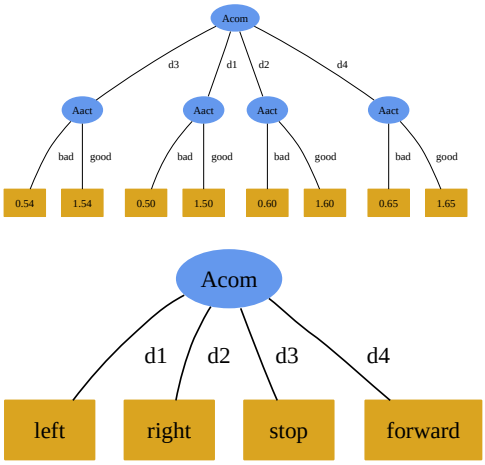


Figure 6.27: Final four-state merged value function (top) and policy (bottom). The value function

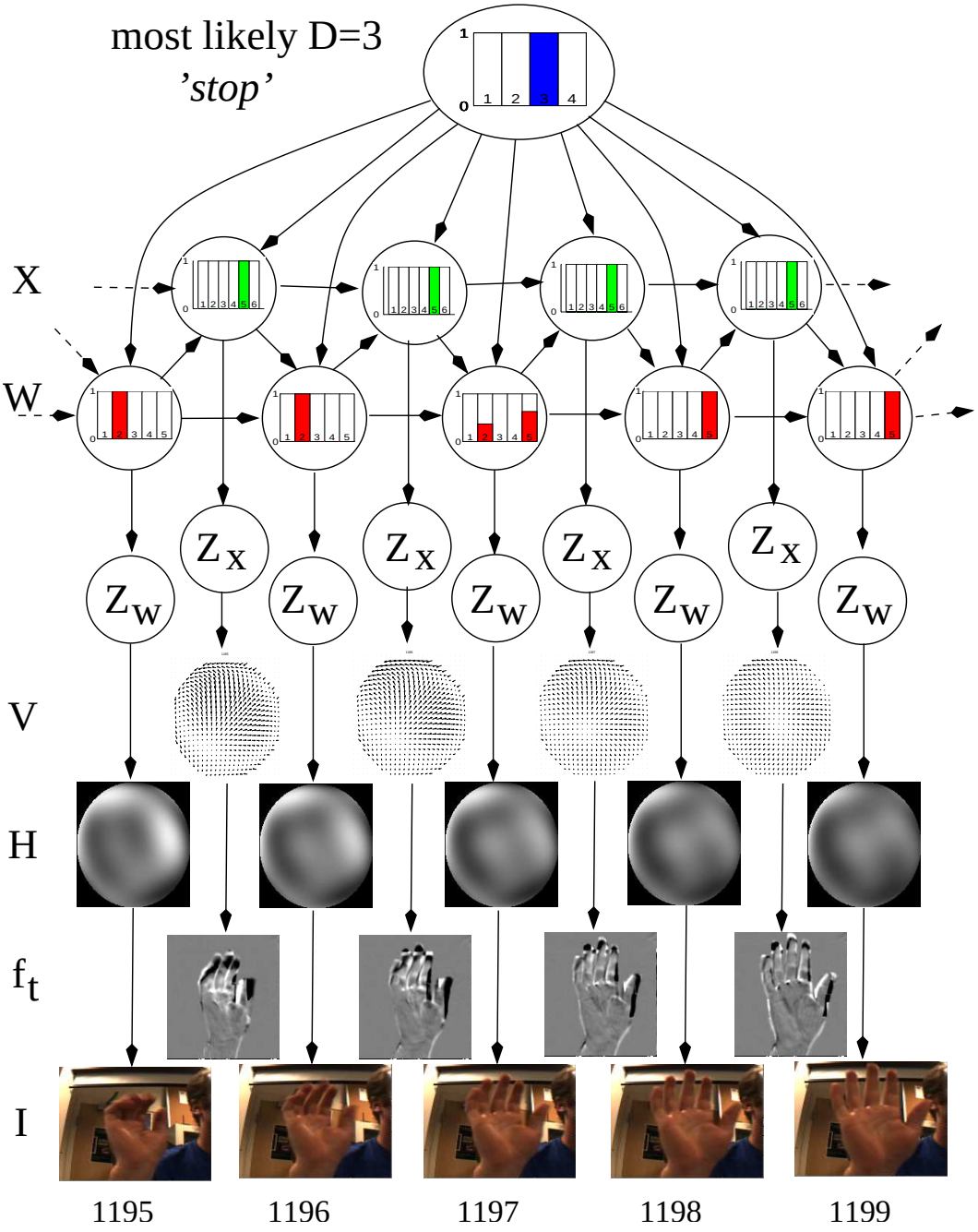


Figure 6.28: Final 4-state gesture model's explanation of a stop gesture as d_3. Continued in Figure 6.29.

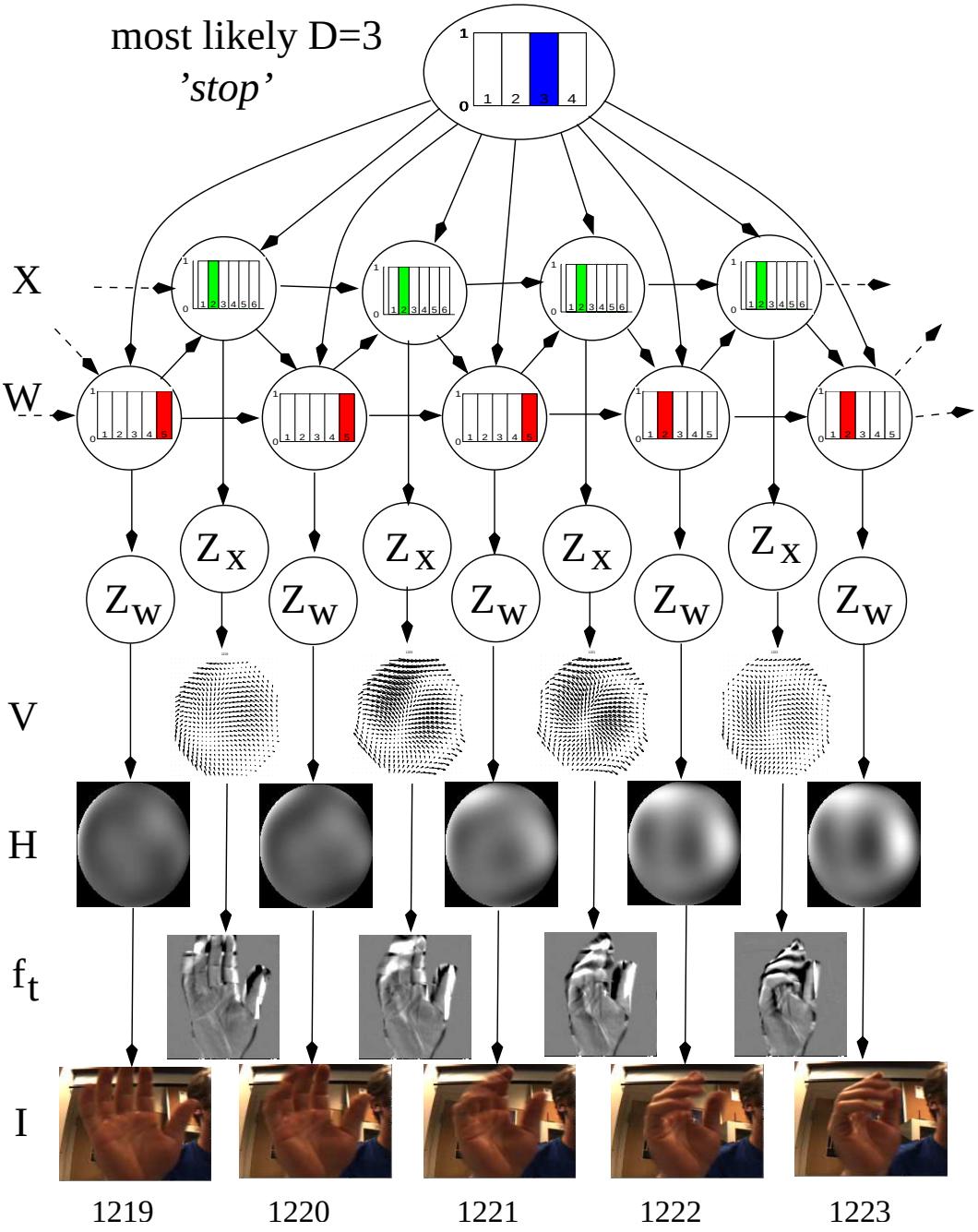


Figure 6.29: Final 4-state gesture model's explanation of a stop gesture as d_3. Continuation of Figure 6.28.

and to the right as it opens (Figure 6.30) and then backwards and to the left as it closes (Figure 6.31).

To evaluate how well the model chooses actions, we performed a cross-validation experiment in which the POMDP was trained on all but one sequence of each gesture. The model was then used to choose actions based upon the four sequences left out. If the action is correct, one reward is given. This process is repeated for 12 different sets of four test sequences, and the total rewards gathered give an indication of how well the model performs on unseen data. The model chose the correct action 47 out of a total of $12 \times 4 = 48$ times, for a total success rate of 47/48 or 98%. The one failure was due to a mis-classification of a "left" gesture as a "right" gesture due to a large rightwards motion of the hand at the beginning of the stroke. The final POMDP models learned that there were $N_a = 4$ states in all 12 cases.

6.3 Card Matching Game

The card matching game, as described in Section 5.7, was played by two players, "Bob" and "Ann", separated by a partition. The players were both students in our laboratory. The players could not make visual or audio contact with each other, except the visual contact available through the game interface. Each player viewed their partner through a direct s-video link from their workstation to a Sony EVI s-video camera mounted about their partner's screen. The average frame rate at 320×240 resolution was over 28fps. This data was also recorded along with a game log describing the actions of each player in the game (card values and suits, bids and committed cards). There were no restrictions placed on the type of communication allowed between the players through the video link: the players were not given any instructions about the types of displays they could or could not use. The rules of the game were explained to the subjects, and they played four games of five rounds each. In the following, we will focus on the model learned from Bob's perspective during his bidding rounds (so the displays will be those of Ann when Bob was bidding). Bob bid first in each game, and so there were three bidding turns for him per game. We will use data from the first three games to train the POMDP model described in Section 5.7.1. The learned model's performance will then be tested on the data from the last game.

We do not analyse Bob's displays here because he was, in fact, the author of the thesis, and his results may be biased. In general, this experiment should be performed with arbitrary subjects. However, we do not intend this experiment as a general human-computer interaction study, but rather as a proof of concept: our

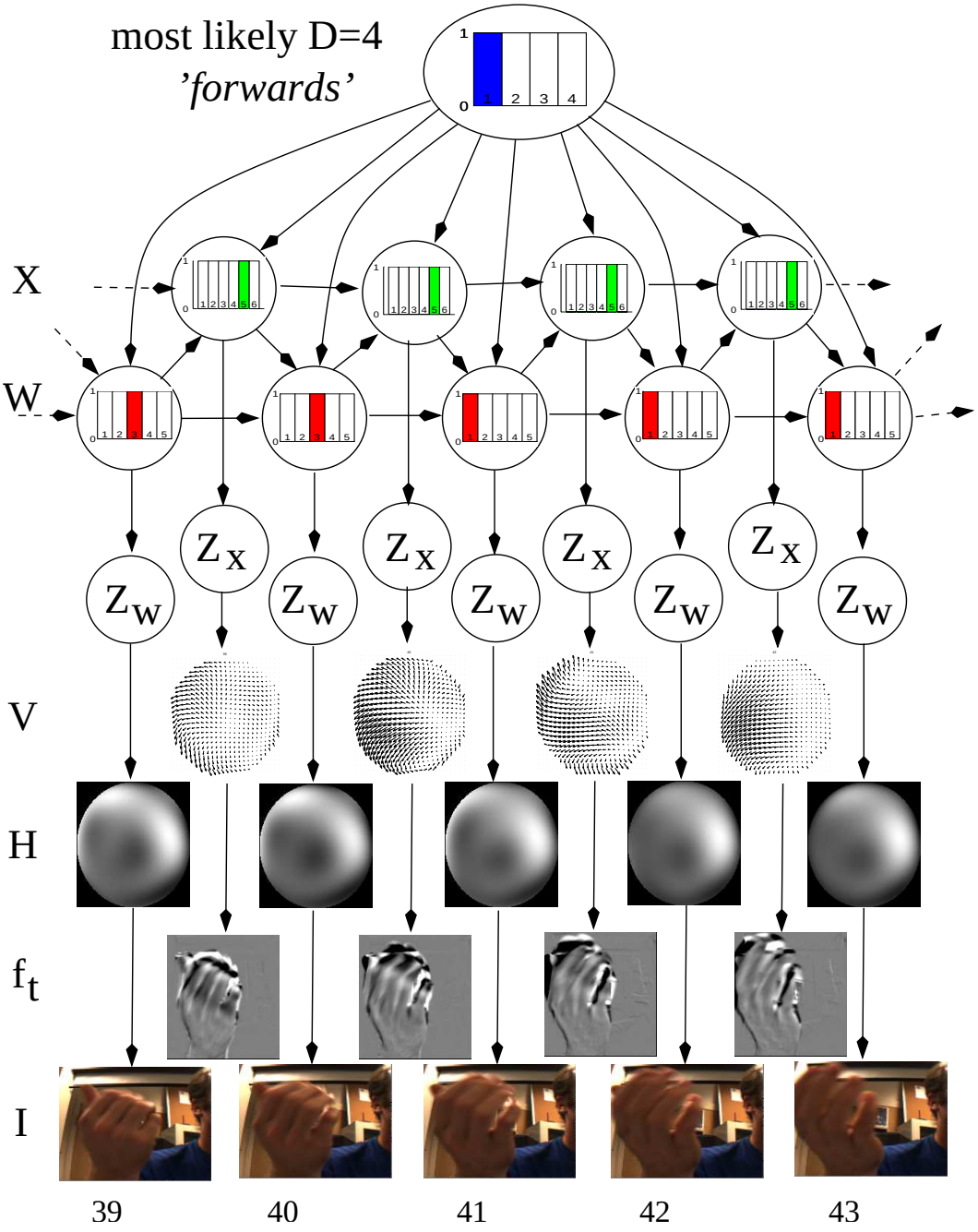


Figure 6.30: Final 4-state gesture model's explanation of a forwards gesture as d_3. Continued in Figure 6.31.

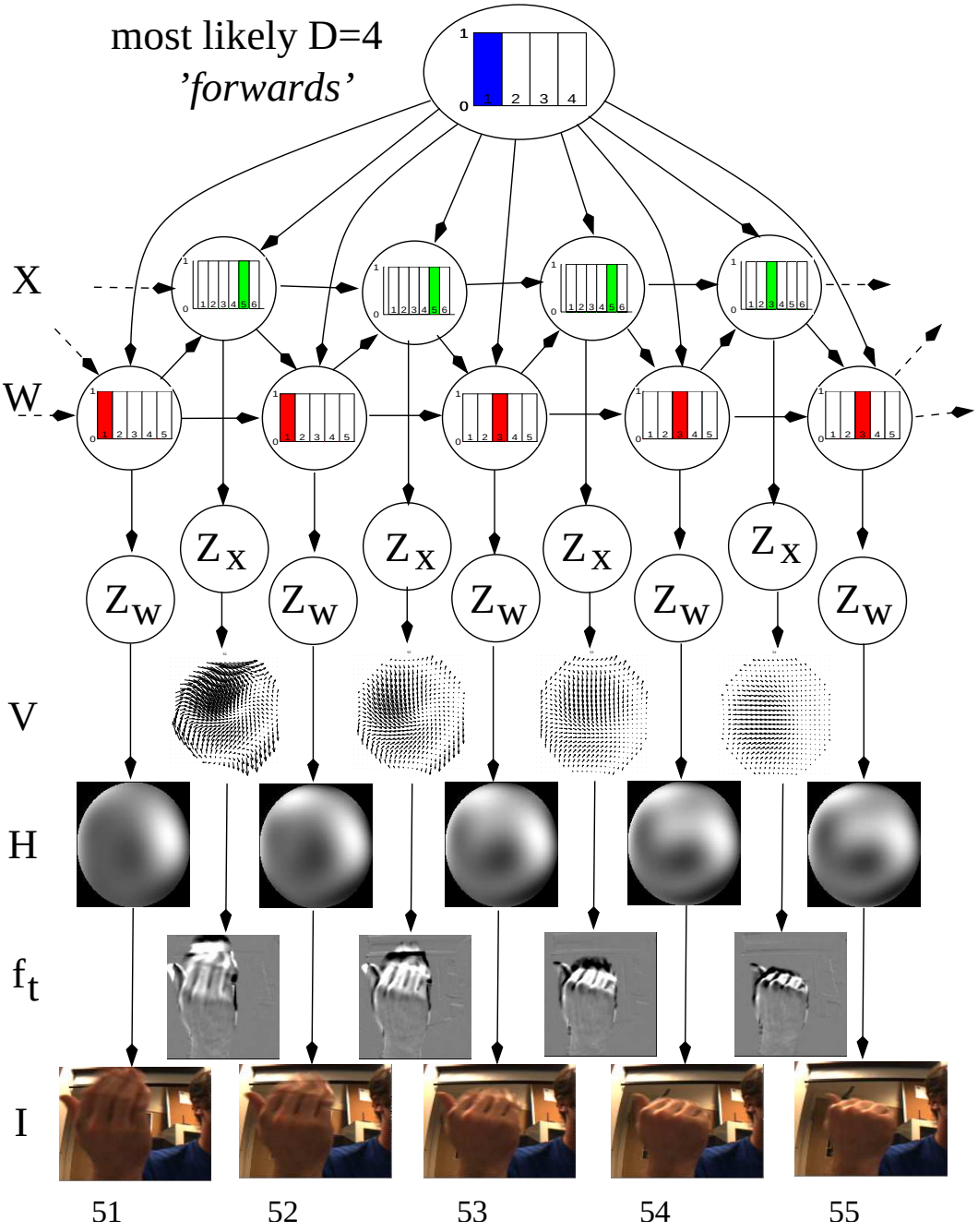


Figure 6.31: Final 4-state gesture model's explanation of a forwards gesture as d_3. Continuation of Figure 6.30.

POMDP model can be applied to this setting, and can learn a valuable policy of action. The generalisability of the system to arbitrary subjects is a subject of future work.

The training data set is certainly large enough to learn models of facial displays, but is quite small when it comes to learning an optimal policy. To see why, notice that, to learn an accurate POMDP, we need examples of every possible valuable state-action pair. Even if many of the 10,000 states and 6 actions in the card matching game will never be visited, the amount of training data required is still quite large. Nevertheless, under certain symmetry assumptions, it may be possible to make near-optimal decisions based only on a small training set. Equivalently, in an on-line reinforcement learning method, it may be possible to start exploiting the learned model with little exploration. For example, if we assume that the players behaviour is symmetrical under permutation of the card suits, then we can average the conditional probability distributions over all such permutations. This allows us to *fill in* more of the model without having to explicitly explore those situations. For example, suppose that a player notices that a certain facial display in response to a bid of hearts is usually followed by her partner committing hearts. If she assumes that her partner has no special preference between the heart and the diamond suits, then she may extrapolate her experiences with hearts to diamonds, and assume that the same facial display in response to a bid of diamonds will be followed by her partner committing diamonds. However, these types of arguments are dependent on the particular game being played, and generalisations require more complex models than the POMDPs considered here. We will, however, show in Section 6.3.5 how they can be used for the card game, and how it makes for much better policies when training with little data.

As discussed in Section 5.7, we can construct MDPs for the two turns independently, and here we focus only on the Bob's bid turns. Table 6.5 shows the game log from Bob's bidding rounds in the three games in the training set, as well as the game used for testing (see Section 6.3.4). The first and third training games involved an initial bid that was refused, followed by a second bid that was accepted. All other bids were accepted on the first bid. Notice that, in some cases (the first, third and last rounds in Table 6.5), Bob's bid pointed to Ann's second highest card, but was nevertheless immediately accepted by Ann. In all three cases, however, this strategy paid off because they got their reward faster, and it was equal or better to the optimal reward. For example, in the first round, Bob bids his high club (7), and Ann accepted and commited her second highest card (a 7 as well), giving a total payoff of 12. Why did Ann not try to get Bob to agree to a play of diamonds? If she had disagreed with his bid of clubs, Bob would surely have bid his second highest

card, which may have been a diamond or a heart with equal probability. The additional payoff to the team is $(3 - \epsilon)$ if Bob's second bid was diamonds, and $(-5 - \epsilon)$ if Bob's second bid was hearts, where ϵ is the expected difference between Bob's highest and second highest card. If Ann and Bob were not communicating over a limited channel, then they could surely come to agreement (after some deliberation) about the most optimal solution. However, the communication difficulties imposed made the risk of failure high enough so that Ann simply accepted the first bid. This approximate solution works well in many cases.

6.3.1 Learning a POMDP

We learned the parameters of the POMDP for the bidding rounds from Bob's perspective, using the methods described in Section 4.7. The model was trained with four display states, which is as large as we think it possible to learn reliable models given the training set size. Two of the learned display states described sequences with little motion ("null states). The other two corresponded roughly to "nodding" and "shaking" of the head.

A slightly modified version of the structure learning algorithm described in Section 5.3 was applied. In this case, we do not compare value functions and policies for every state of the game, as they will often not agree in many places. Instead, we look at where the majority of states agree on a merge. Two states were merged, resulting in a three-state model. The two merged models, d_1 and d_2, both described "null" sequences, with little facial motion. This redundancy shows up in the value function, $V(s)$, a small portion of which is shown in Figure 6.32. The highest value

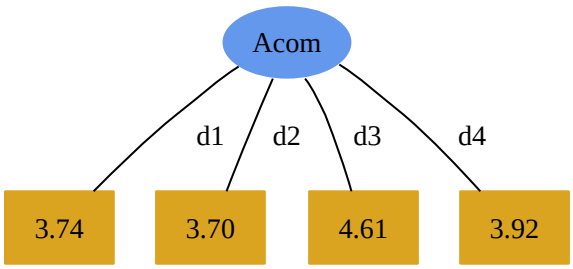


Figure 6.32: Small portion of the two stage-to-go value function of the card matching game with four display states.

is when the partner nods (d_3), because we expect to get a reward of 3 within one time step. The expected values for the null displays are almost equal. This indicates that making the distinction between d_1 and d_2 is not useful for determining value. This same redundancy shows up in the policy for the four state model, as shown in

	round	frames	Bob's cards			Ann's cards			Ann's display	*bid*	Bob's action	Ann's action
			♡	♢	♣	♡	♢	♣	*Acom*		*Bact*	*Aact*
training game 1	1	40-150	3_1	4_1	7_3	2_1	10_3	7_2	d_2	-	bid♣	-
	1	151-295							d_2	♣	cmt♣	cmt♣
	2	725-827	2_1	5_2	2_1	7_2	3_1	8_3	d_1	-	bid♢	-
	2	828-976							d_3	♢	bid♣	-
	2	977-1048							d_2	♣	cmt♣	cmt♣
	3	1544-1626	6_2	9_3	4_1	1_1	4_1	8_3	d_1	-	bid♢	-
	3	1627-1709							d_2	♢	cmt♢	cmt♢
training game 2	1	11-195	1_1	4_1	4_1	2_1	7_2	7_2	d_3	-	bid♢	-
	1	196-351							d_2	♢	cmt♢	cmt♢
	2	762-830	7_2	3_1	2_1	9_3	3_1	7_2	d_1	-	bid♡	-
	2	831-942							d_2	♡	cmt♡	cmt♡
	3	1407-1555	7_2	1_1	4_1	7_2	7_2	1_1	d_1	-	bid♡	-
	3	1556-1651							d_2	♡	cmt♡	cmt♡
training game 3	1	11-129	1_1	3_1	9_3	7_2	9_3	10_3	d_1	-	bid♣	-
	1	130-188							d_2	♣	cmt♣	cmt♣
	2	736-834	7_3	6_2	4_1	1_1	7_2	5_2	d_3	-	bid♡	-
	2	835-957							d_3	♡	bid♢	-
	2	958-1052							d_2	♢	cmt♢	cmt♢
	3	1502-1576	10_3	1_1	4_1	8_3	3_1	9_3	d_1	-	bid♡	-
	3	1577-1653							d_2	♡	cmt♡	cmt♡
test game	1	60-163	3_1	10_3	8_3	1_3	6_2	8_3	d_2	-	bid♢	-
	1	164-339							d_2	♢	cmt♢	cmt♢
	2	950-1047	7_2	4_1	8_3	2_1	1_1	5_2	d_2	-	bid♣	-
	2	1048-1129							d_2	♣	cmt♣	cmt♣
	3	1561-1633	9_3	10_3	3_1	1_1	2_1	6_2	d_2	-	bid♡	-
	3	1634-1800							d_3	♡	bid♢	-
	3	1801-2136							d_3	♢	cmt♢	cmt♢

Table 6.5: Log for the card matching games. The card values are shown for each suit and each player as the actual value on the card subscripted with the value in the model. Thus, i_j means the players saw a card with value i, but it is characterized as $\mathtt{v}\langle\mathtt{j}\rangle$ in the POMDP. A blank means the card values were the same as the previous sequence. Ann's display, *Acom*, is the most likely as classified by the final model. The first three games are used for training the model, the last for testing (see Section 6.3.4)

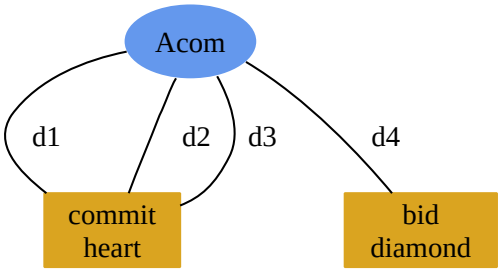


Figure 6.33: Small portion of the two stage-to-go policy for the card matching game with four display states.

Figure 6.33. The model was re-trained with only three states after merging states d_1 and d_2. The result was, as expected, one "null" state (d_1), one "nodding" state (d_2) and one "shaking" state (d_3). The remainder of this section discusses this reduced three-state model.

6.3.2 Structure of Learned Model

Recall from Section 5.2 that there are two conditional probability distributions to be learned from the training data, both of which are model parameters in the C4MG model we described in Section 3.6.3. Due to the lack of some instances of game states in the training data, the learned conditional probabilities contained some distributions which were only based on the Dirichlet smoothing prior. For example, there is no instance in the data when the player bid hearts after bidding clubs (and receiving a head shake of disagreement from her partner), so the action $bid\heartsuit$ was never taken in a state where bid was ♣. for any value of $Acom$.

Figure 6.34 shows the learned conditional probability distribution Ann's action, $Aact$, given the current bid and Ann's display, $Acom$, as a decision tree. The leaves show the distribution over the possible values of $Aact$, from top to bottom: $null, cmt\heartsuit, cmt\diamondsuit, cmt\clubsuit$. We see that, if the bid is null, we expect Ann to do nothing in response. If the bid is some suit, s, and Ann's display ($Acom$) is the "nodding" display d_2, then there is a good chance that Ann will commit her card of suit s. On the other hand, if Ann's display is the "shaking" display, d_3, or the "null" display, d_1, then we expect her to do nothing (and wait for another bid from Bob).

The conditional probability distribution of Ann's display, $Acom$, at time t, given the previous and current bids, bid_{t-1}, and bid_t, respectively, are different for each of Bob's actions. This is because Ann observes Bob's bid the moment he makes it. One example, for Bobs action $bid\diamondsuit$, is shown in Figure 6.35. These distributions

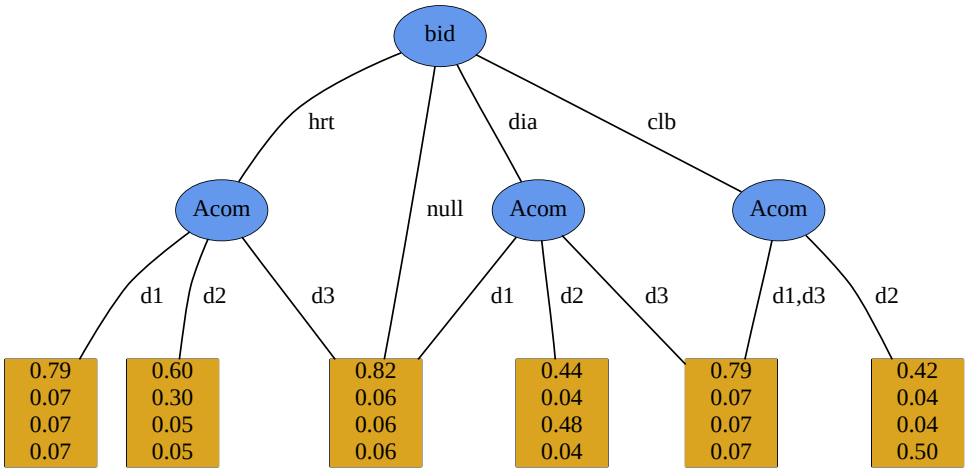


Figure 6.34: Learned conditional probability distribution over Ann's action, *Aact*, given the current *bid* and Ann's display, *Acom*. The leaves of the decision tree show the distribution over the possible values of *Aact* from top to bottom: $null, cmt\heartsuit, cmt\diamondsuit, cmt\clubsuit$.

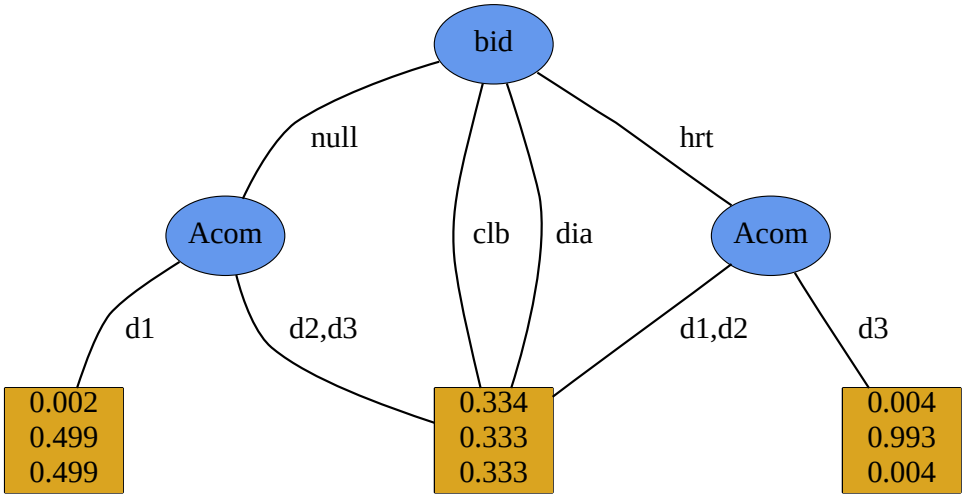


Figure 6.35: Learned conditional probability distribution over Ann's display, *Acom*, at time t, given the previous and current bids, bid_{t-1}, and bid_t, respectively, for Bob's action $bid\diamondsuit$. The leaves of the decision tree show the distribution over the possible values of *Acom* from top to bottom: $d1, d2, d3$.

carry two important pieces of information for Bob:

1. At the beginning of a round, any bid is likely to elicit a non-null display d_2 or d_3. As shown in Figure 6.35, the expected distribution over $Acom = d_1, d_2, d_3$ after action $Bact = bid\diamondsuit$ if the current bid is *null* (at the beginning of a round) and the previous display was d_1 (a null display) is $0.01, 0.49, 0.49$. Thus, d_1 (null) display is not very likely, while d_2 and d_3 (nod and shake) are equally likely.

2. A "nodding" display is more likely after a "shaking" display if the bid is changed. As shown in Figure 6.35, the expected distribution over $Acom = d_1, d_2, d_3$ after action $bid\diamondsuit$ if the current bid is $\heartsuit$ and the previous $Acom$ was d_3 is $0.004, 0.993, 0.004$: if Bob bids diamonds and sees a d_3 display (a shake), then a bid of clubs will most likely elicit a d_2 display (a nod).

6.3.3 Policy of Action

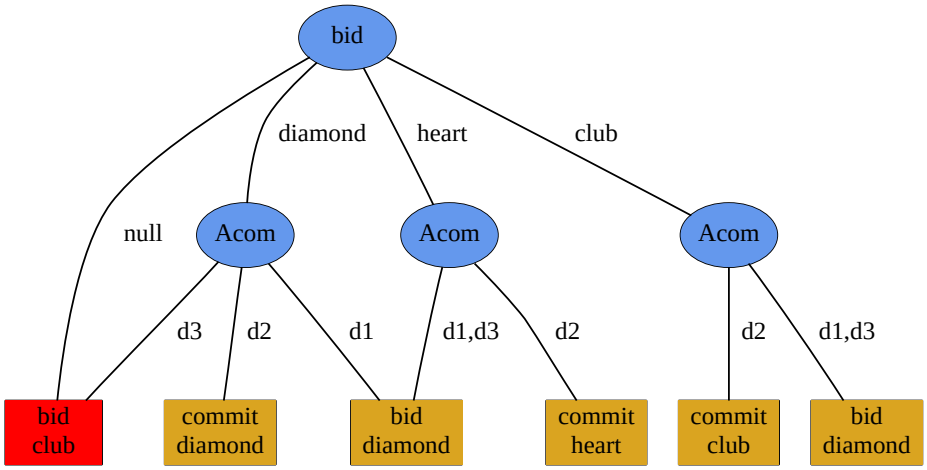


Figure 6.36: Policy of action in the card matching game for the situation in which $B\heartsuit v = v3, B\diamondsuit v = v3$ and $B\clubsuit v = v1$.

The full policy for the game is quite large, and we instead show the sub-policy for one particular set of values for the cards. In particular, Figure 6.36 shows the policy of action if the player's cards have values $B\heartsuit v = v3, B\diamondsuit v = v3$ and $B\clubsuit v = v1$. To consult the policy, simply follow the path from the root to a leaf corresponding to the state and read the recommended action at the leaf. The lightly colored leaves are actions which are "correct" in that they make sense given our interpretation of the display states (d_1 is *null*, d_2 is a nod, and d_3 is a shake).

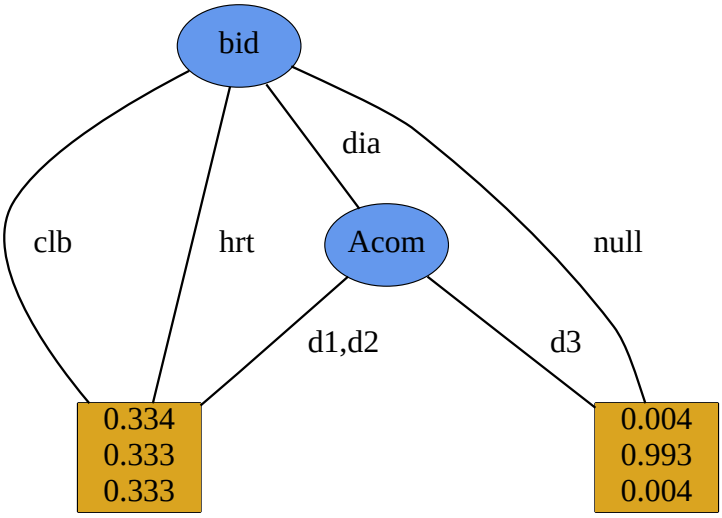


Figure 6.37: Learned conditional probability distribution over Ann's display, *Acom*, at time t, given the previous and current bids, bid_{t-1}, and bid_t, respectively, for Bob's action *bid*♣. The leaves of the decision tree show the distribution over the possible values of *Acom* from top to bottom: $d1, d2, d3$.

The darkly shaded nodes are actions that are "incorrect", usually due to a lack of information about these states in the training data. The policy recommends the following actions

- If there is no bid on the table, then bid the club. This is a sub-optimal decision due to lack of training data, and resulting asymmetries between suits in the conditional probability distributions. Figure 6.37 shows the probability distribution over Ann's display, *Acom*, for Bob's action, *bid*♣. As the figure shows, when the bid is null, the expected *Acom* is d_2 with high probability, and so the optimal bid is always clubs regardless of the values of the cards. This probability distribution arises because in the training data, every bid of ♣ is followed by a display d_2 (is accepted).

- If the bid is diamonds, and the partner nodded (d_2), then commit the diamond. Otherwise, if the partner did nothing (d_1), then bid the diamond (again). Finally, if the partner shook their head (d_3), then bid the club. Again, this last action is sub-optimal.

- If the bid is hearts and the partner shook nodded their head (d_2), then commit the heart, otherwise, bid the diamond (the other high card).

- If the bid is clubs and the partner shook nodded their head (d_2), then commit

the club, otherwise, bid the diamond.

6.3.4 Using the Policy

game	Bob's cards			Ann's display	bid	Bob's action	policy normal	policy permute
	♡	♢	♣	*Acom*		*Bact*	$\pi(s)$	$\pi'(s)$
1	1	1	3	d_2	-	bid♣	bid♣	bid♢/♣
1				d_2	♣	cmt♣	cmt♣	cmt♣
2	1	2	1	d_1	-	bid♢	bid♣	bid♢
2				d_3	♢	bid♣	bid♢	bid♡
2				d_2	♣	cmt♣	cmt♣	cmt♡
3	2	3	1	d_1	-	bid♢	bid♣	bid♢
3				d_2	♢	cmt♢	cmt♢	cmt♢
4	1	1	1	d_3	-	bid♢	bid♣	cmt♡/♣
4				d_2	♢	cmt♢	cmt♢	cmt♢
5	2	1	1	d_1	-	bid♡	bid♣	bid♡
5				d_2	♢	cmt♡	cmt♡	cmt♡
6	2	1	1	d_1	-	bid♡	bid♣	bid♡
6				d_2	♡	cmt♡	cmt♡	cmt♡
7	1	1	3	d_1	-	bid♣	bid♣	bid♣
7				d_2	♣	cmt♣	cmt♣	cmt♣
8	3	2	1	d_3	-	bid♡	bid♣	bid♡
8				d_3	♡	bid♢	bid♢	bid♢
8				d_2	♢	cmt♢	cmt♢	cmt♢
9	3	1	1	d_1	-	bid♡	bid♣	bid♡
9				d_2	♢	cmt♡	cmt♡	cmt♡

Table 6.6: Log for the training games showing the predicted actions from the policy, $\pi(s)$, and from the symmetrized policy, $\pi'(s)$. Shaded entries are incorrect policy predictions.

Tables 6.6 and 6.7 show Ann's displays, the bids, Bob's cards and Bob's actions during the training and test game, respectively (see Table 6.5 for other variables). The second-to-last column shows the predictions of Bob's actions of the policy, $\pi(s)$, discussed in the last section. This policy correctly predicted 14 out of 20 actions in the training games, and 5 out of 7 actions in the test game The incorrect actions are shaded in Tables 6.6 and 6.7. However, as we have pointed

game	Bob's cards			Ann's display	bid	Bob's action	policy normal	policy permute
	♡	♢	♣	*Acom*		*Bact*	$\pi(s)$	$\pi'(s)$
1	1	3	3	d_2	-	bid♢	bid♣	bid♢/♣
1				d_2	♢	cmt♢	cmt♢	cmt♢
2	2	1	3	d_2	-	bid♣	bid♣	bid♣
2				d_2	♣	cmt♣	cmt♣	cmt♣
3	3	3	1	d_2	-	bid♡	bid♣	bid♡
3				d_3	♡	bid♢	bid♢	bid♢
3				d_3	♢	cmt♢	bid♣	bid♡

Table 6.7: Log for the testing game showing the predicted actions from the policy, $\pi(s)$, and from the symmetrized policy, $\pi'(s)$. Shaded entries are incorrect policy predictions.

out, many of the problems with this policy can be attenuated by applying symmetry arguments to construct a symmetrised policy, $\pi'(s)$. This is discussed in the next section.

6.3.5 Symmetry Considerations

As we have pointed out, symmetry arguments can be applied to the learned POMDP in order to attenuate the effects of the lack of training data. In particular, we may assume that player's do not have any particular preference over card suits, such that the conditional probability tables should be symmetric under permutation of suits. Therefore, we can "symmetrise" the probability distributions by simply averaging over the six card suit permutations. Figures 6.38 and 6.39 show the symmterised conditional probability distributions over *Aact* and *Acom*. These can be compared to the unsymmetrised versions in Figures 6.34 and 6.35, respectively.

Figure 6.40 shows a portion of the symmetrised policy computed for the symmetrised POMDP. The predictions of this policy are shown in the last columns of Tables 6.6 and 6.7 for training and test data, respectively. The symmetrised policy correctly predicts all but one of Bob's actions in the training game, for an error rate of only 5%. The mis-classification was due to the subject looking at something to one side of the screen yielding significant horizontal head motion, leading to a classificaiton as d_3.

The symmetrised policy correctly predicts all but one of Bob's actions in the test game. It does correctly predict the re-bid in the third round, where the first

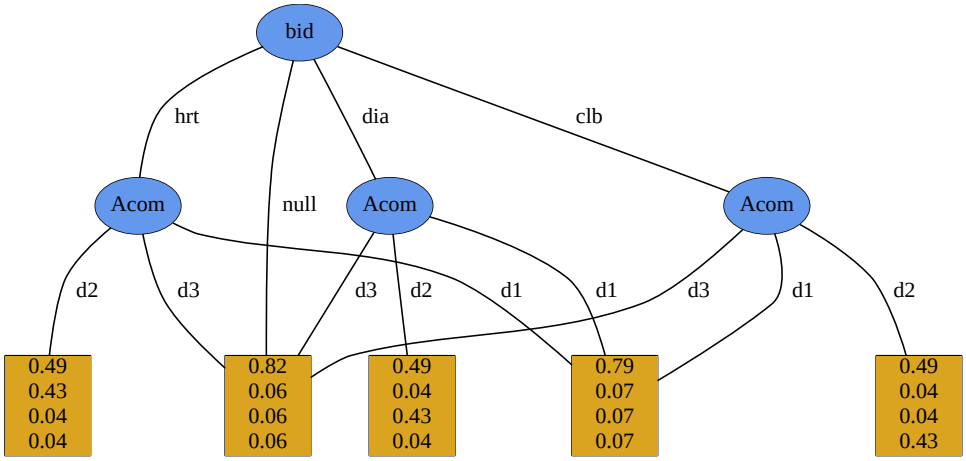


Figure 6.38: Learned conditional probability distribution over Ann's action, *Aact*, given the current *bid* and Ann's display, *Acom*. The leaves of the decision tree show the distribution over the possible values of *Aact* from top to bottom: $null, cmt\heartsuit, cmt\diamondsuit, cmt\clubsuit$.

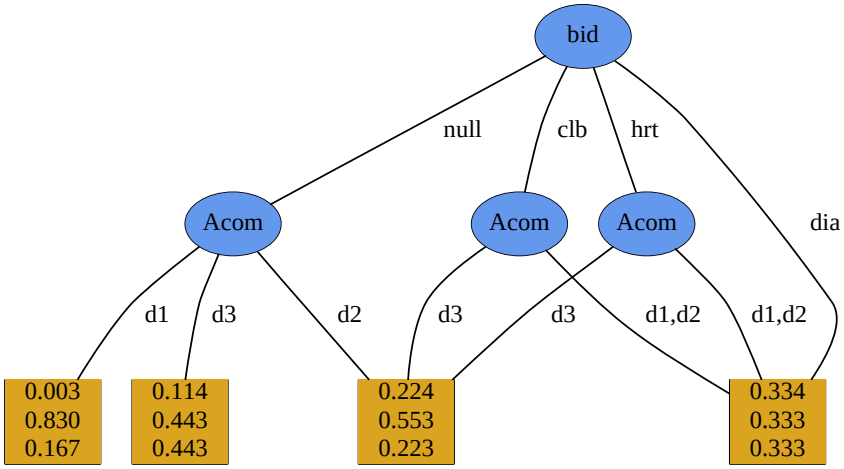


Figure 6.39: Learned conditional probability distribution over Ann's display, *Acom*, at time t, given the previous and current bids, bid_{t-1}, and bid_t, respectively, for Bob's action $bid\diamondsuit$. The leaves of the decision tree show the distribution over the possible values of *Acom* from top to bottom: $d1, d2, d3$.

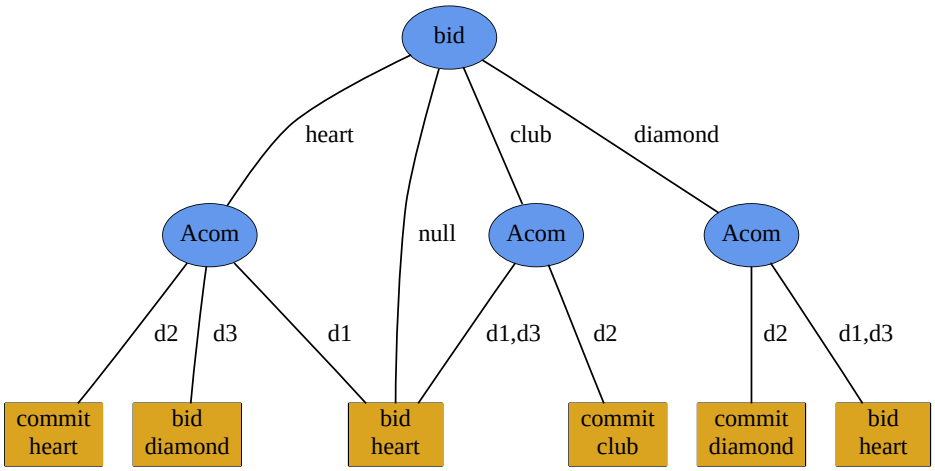


Figure 6.40: Policy of action in the card matching game for the situation in which $B\heartsuit v = v3, B\diamondsuit v = v3$ and $B\clubsuit v = v1$.

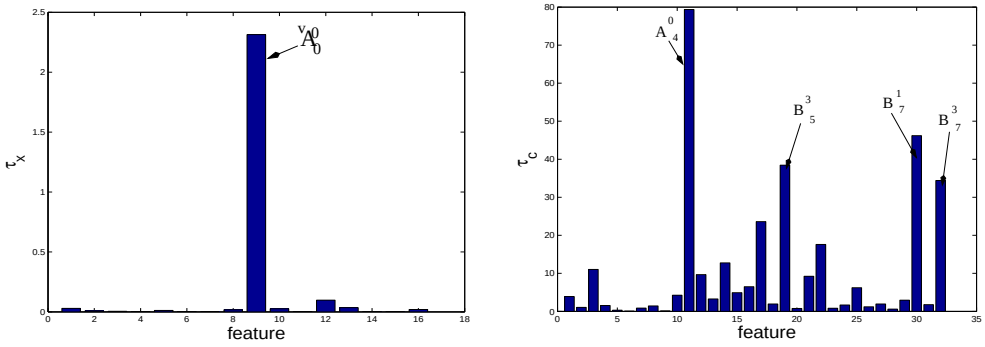
bid (of hearts) was refused by the partner's display. Figure 6.40 shows the policy of action in this situation. At the beginning, when there is no bid on the table (so *bid* is *null*), the policy recommends that Bob bid his heart (as he did). The C4MG then recognized Ann's subsequent display as state d_3 (shake), and the policy recommends changing the bid to diamonds. The C4MG classifies the following display as d_3, which is incorrect (Ann actually nodded her head), so the policy recommends bidding the heart again. The last sequence is longer than usual (over 300 frames), and includes some horizontal head motion in the beginning which appears as shaking in the model. This mis-classification may expose a weakness of the temporal segmentation method we use, which is based entirely on the observable actions and game states. Although this sequence is long, it is only the (fairly vigorous) head nod at the very end which is the important display. Perhaps a termporal segmentation which focussed more on later motions would be more attuned to this kind of sequence.

6.3.6 Output Models

Model 2 : "nodding"

Feature weights for the dynamics and configuration chains are shown in Figure 6.41. There is one significant feature in the dynamics chain, ${}^{v}A_0^0$, which describes vertical translational movement. The significant features in the configuration process are A_4^0, B_5^3, B_7^1 and B_7^3.

The output distributions of the seven states (X) in the dynamics chain are



dynamics configuration

Figure 6.41: Feature weights τ_k^2 for model 2 dynamics and configuration chains

shown in Figure 6.42, plotted along two most significant feature dimensions, ${}^{u}A_1^1$ and ${}^{v}B_1^1$. Four states ($X = 1, 2, 3, 7$) correspond to no motion (the face is stationary), while two correspond to translation upwards ($X = 4, 5$), and the last to translation downwards ($X = 6$).

The output distributions of the five configuration states are shown in Figure 6.43. States $1 - 6$ and 8 describe orientations of the face (tilted upwards, facing forwards, etc), while state 7 describes an unusual pose that was caused by the subject adjusting her hair, so her hand entered into the top left of the facial region.

We now show how the C4MG model analyses a particular sequence from the card matching game for which the most likely high level state is $D = 3$. We show the expected feature vector for each frame in the dynamics and configuration chains, Z_x and Z_w, respectively, and the expected output flow fields and poses. Figure 6.44 shows the trajectory of the reconstructed dynamics Zernike vector (Z_x) along the two most important feature vector dimensions for a sequence classified as model $D = 3$. The sequence alternates between motion upwards (states 4 and 5) and motion downwards (state 6).

Figure 6.45 shows the trajectory of the reconstructed configuration Zernike vector (Z_c) along the two most important feature vector dimensions for a sequence classified as model $D = 3$.

Figures 6.46 and 6.47 show the model's explanation of the same sequence, for the frames indicated in Figures 6.44 and 6.45. We see the high level distribution over D is peaked at $D = 3$. Distributions over dynamics and configuration chains show which state is most likely at each frame. The expected pose, H, and flow field, V, are shown conditioning the image, I, and the temporal derivative, f_t, respectively.

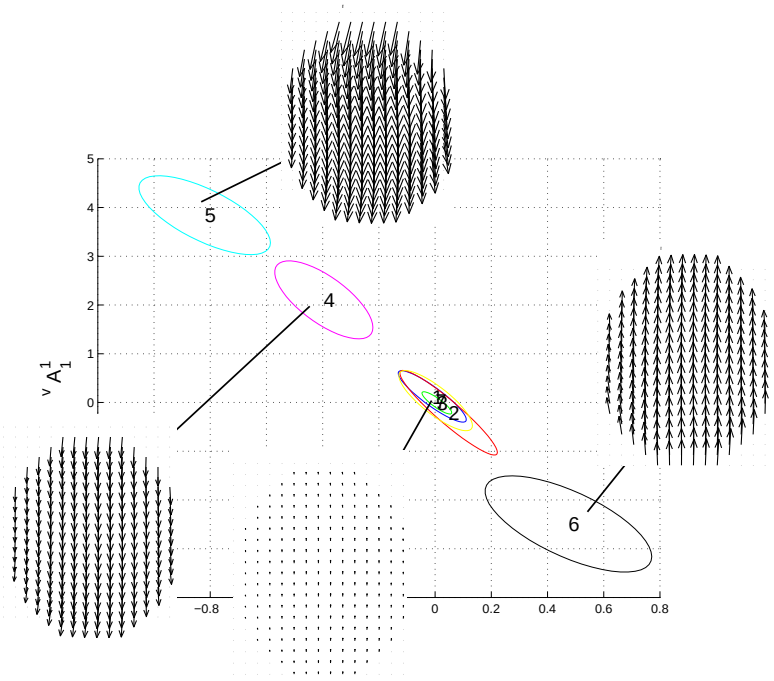


Figure 6.42: Dynamics chain model 2 output states plotted along two most significant dimensions according to feature weights, ${}^{v}A_0^0, {}^{v}A_0^2$. Reconstructed flow fields for X state means are also shown.

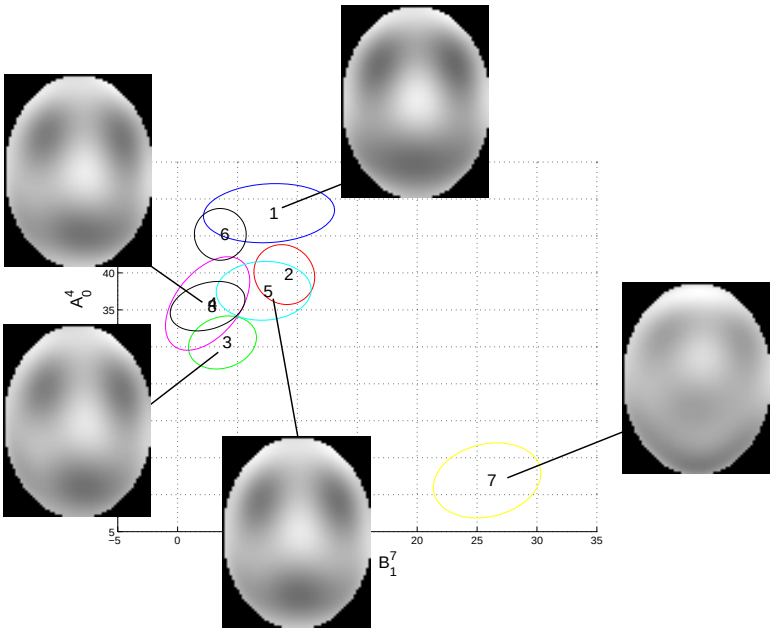


Figure 6.43: Configuration chain model 2 output distributions plotted along two most significant dimensions according to feature weights, ${}^{v}A_4^0, {}^{v}A_7^1$. Reconstructed grayscale images are shown for C state means.

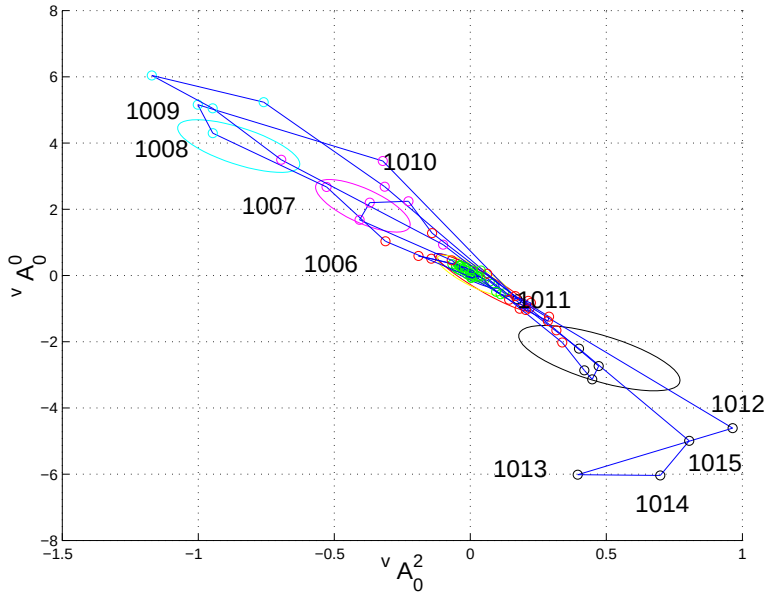


Figure 6.44: Trajectory of most likely dynamics feature vector (Z_x) for sequence 6 (for which model 2 has highest posterior). The level curves of the Gaussian output covariances for model 2 are also shown.

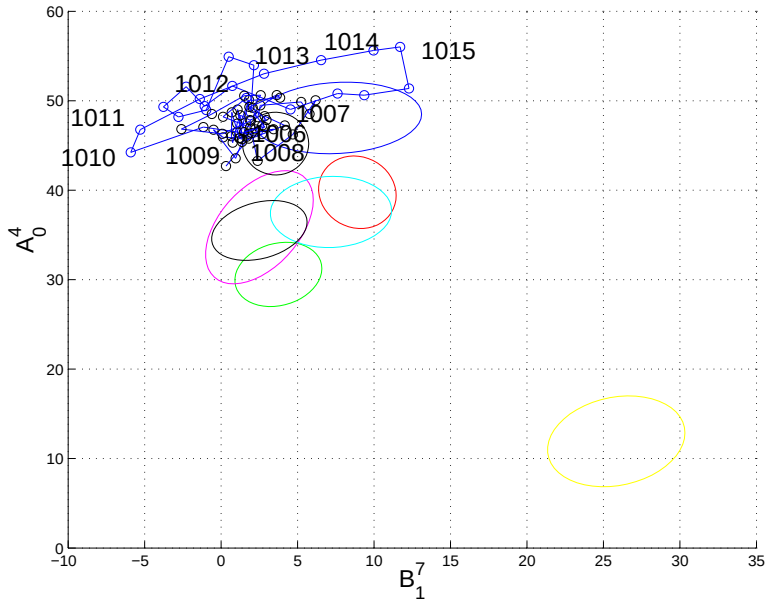


Figure 6.45: Trajectory of most likely configuration feature vector (Z_c) for sequence 6 (for which model 2 has highest posterior). The level curves of the Gaussian output covariances for model 2 are also shown.

Model 3: "shaking"

Feature weights for the dynamics and configuration chains are shown in Figure 6.48. There is one significant feature in the dynamics chain, ${}^{u}A_0^0$, which describes horizontal translational movement. The significant features in the configuration process are A_4^0, A_4^4, B_5^3 and B_7^3.

The output distributions of the six states (X) in the dynamics chain are shown in Figure 6.42, plotted along two most significant feature dimensions, ${}^{u}A_0^0$ and ${}^{v}A_0^0$. Two states ($X = 2, 3$) correspond to no motion (the face is stationary), while two ($X = 1, 5$) correspond to motion to the left and right, respectively. The remaining two account for small motion up and down.

The output distributions of the five configuration states are shown in Figure 6.50. Different angles of the face are represented: state 1 is angled to the left, states 3 and 4 are angled to the right, and states 2 and 8 are facing forward.

We now show how the C4MG model analyses a particular sequence from the card matching game for which the most likely state is $D = 4$. We show the expected feature vector for each frame in the dynamics and configuration chains, Z_x and Z_w, respectively, and the expected output flow fields and poses. Figure 6.51 shows the trajectory of the reconstructed dynamics Zernike vector (Z_x) along the two most

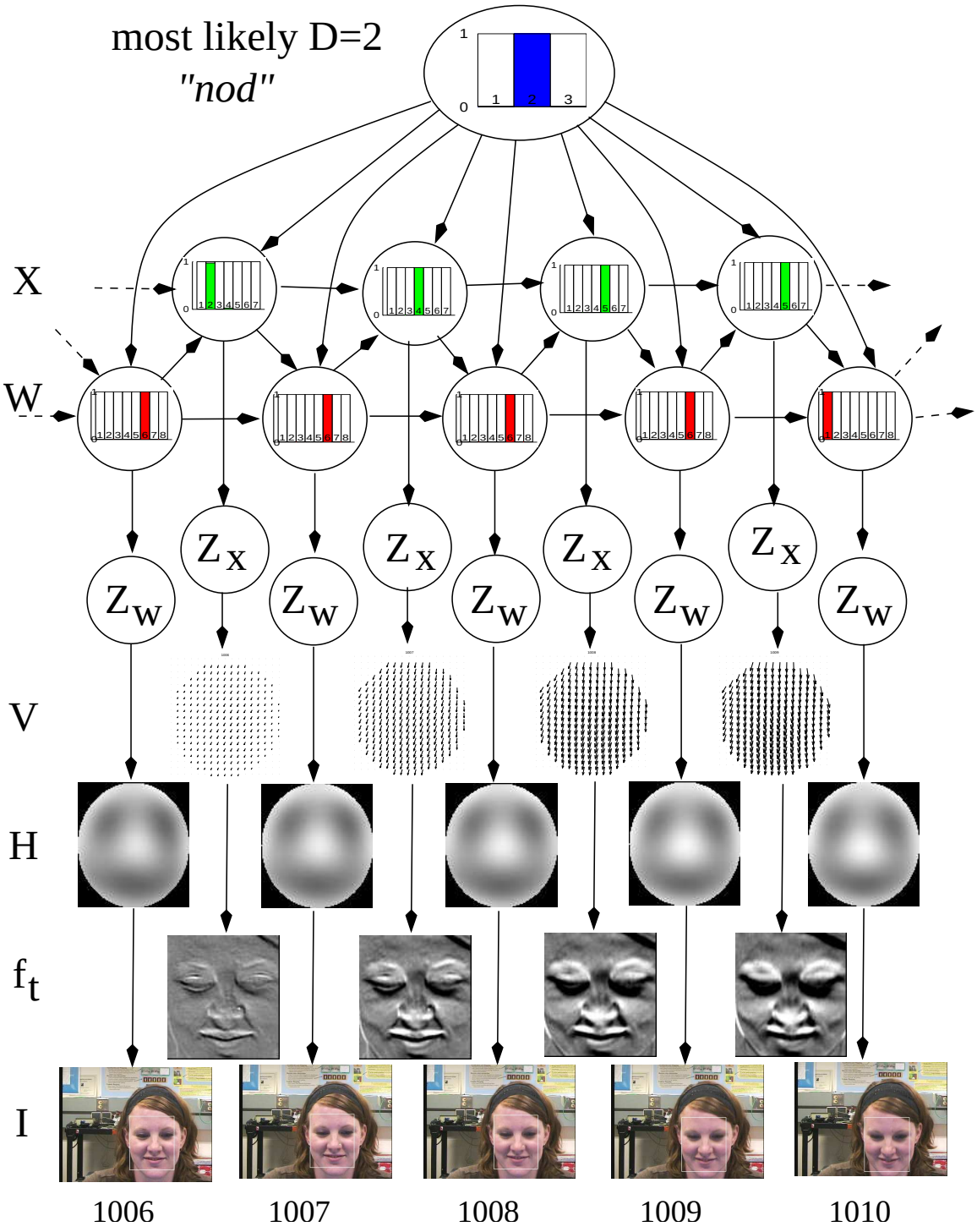


Figure 6.46: The C4MG model's explanation of part of a sequence during the card matching game. The subject's face tilts downwards during a nodding motion.

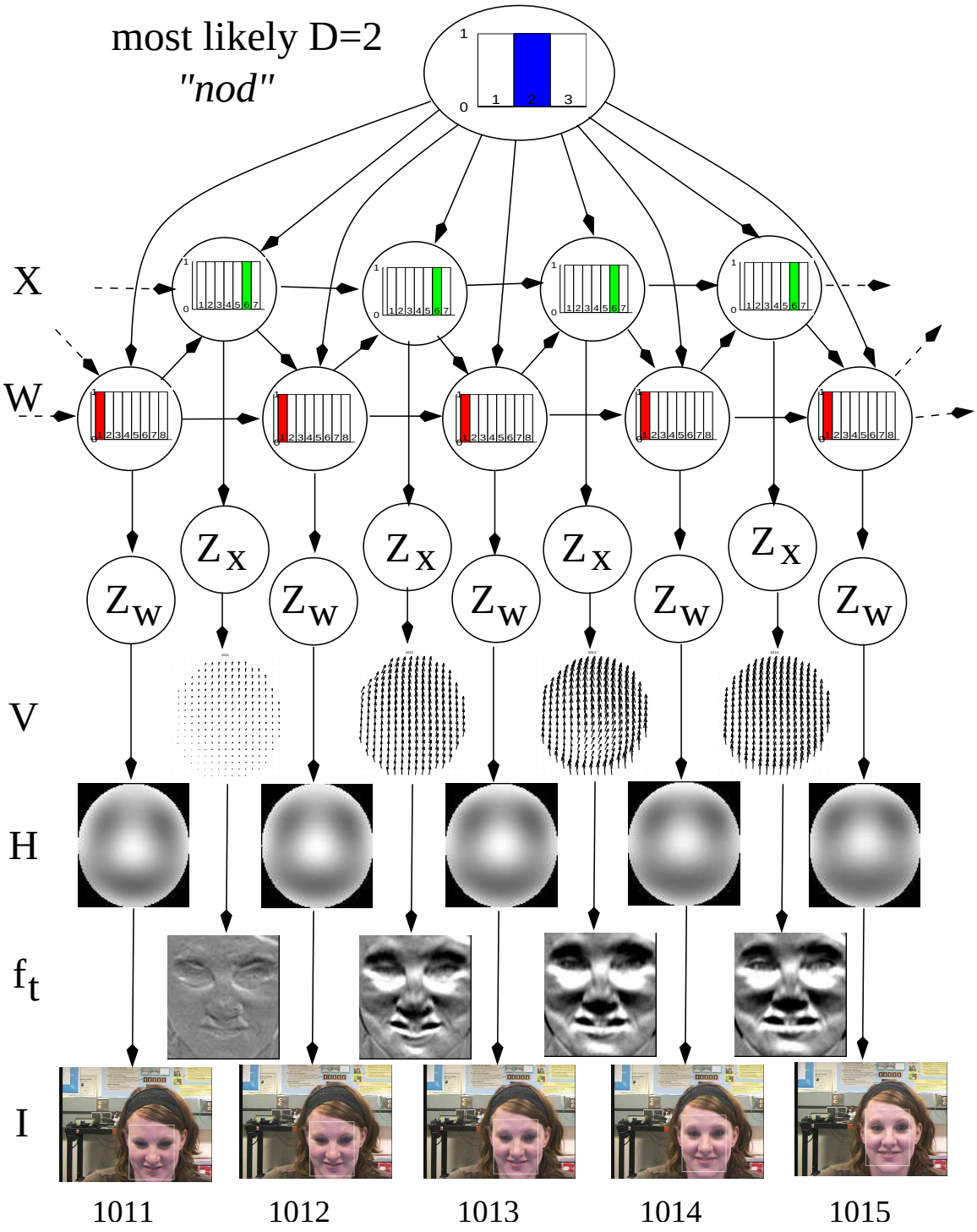
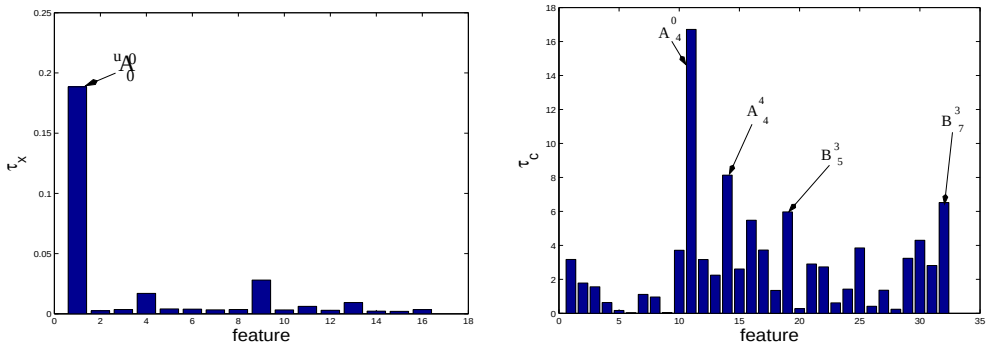


Figure 6.47: The C4MG model's explanation of part of a sequence during the card matching game. The subject's face tilts upwards during a nodding motion.



dynamics configuration

Figure 6.48: Feature weights τ_k^2 for model 3 dynamics chain

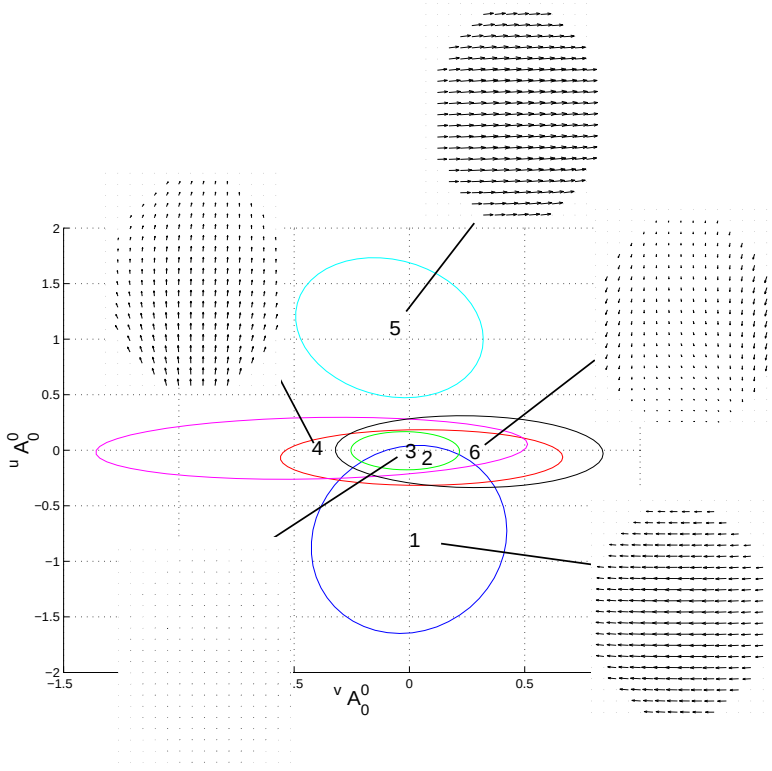


Figure 6.49: Dynamics chain model 3 output states plotted along two most significant dimensions according to feature weights, ${}^{u}A_0^0, {}^{v}B_0^0$. Reconstructed flow fields for X state means are also shown.

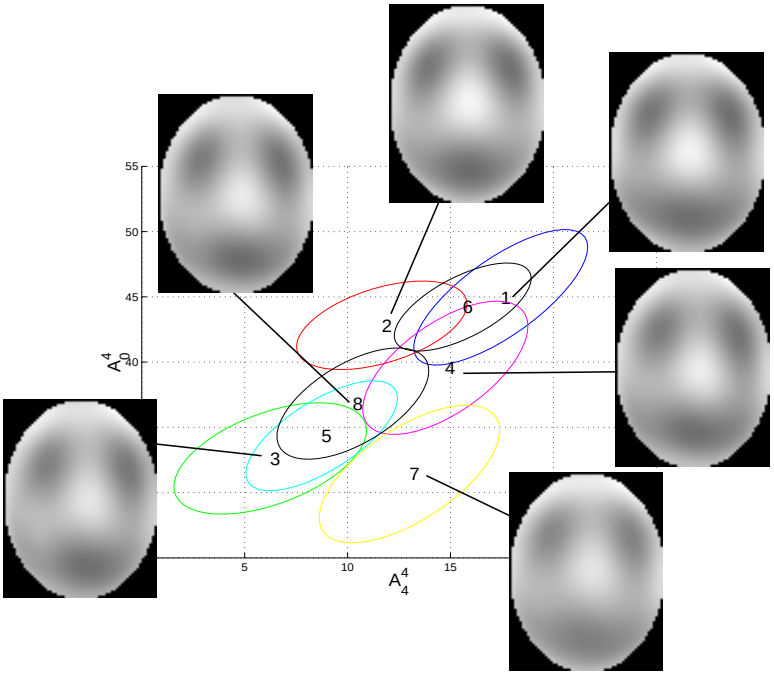


Figure 6.50: Configuration chain model 3 output distributions. Reconstructed grayscale images are shown for C state means.

important feature vector dimensions for a sequence classified as model $D = 4$. The sequence alternates between motion to the right (state 5) and to the left (state 1).

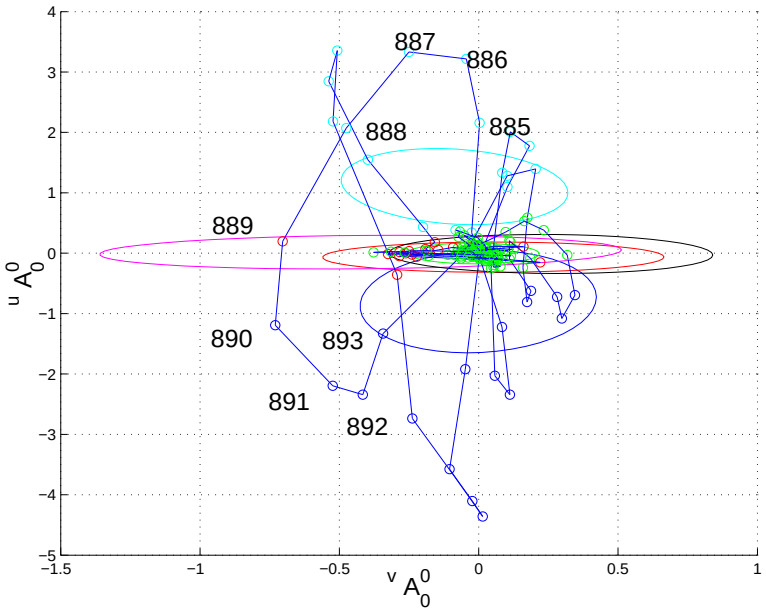


Figure 6.51: Trajectory of most likely dynamics feature vector (Z_x) for sequence 5 (for which model 3 has highest posterior). The level curves of the Gaussian output covariances for model 3 are also shown.

Figure 6.52 shows the trajectory of the reconstructed configuration Zernike vector (Z_c) along the two most important feature vector dimensions for a sequence classified as model $D = 3$.

Figures 6.53 and 6.54 show the model's explanation of the same sequence, for the frames indicated in Figures 6.51 and 6.52. We see the high level distribution over D is peaked at $D = 4$. Distributions over dynamics and configuration chains show which state is most likely at each frame. The expected pose, H, and flow field, V, are shown conditioning the image, I, and the temporal derivative, f_t, respectively.

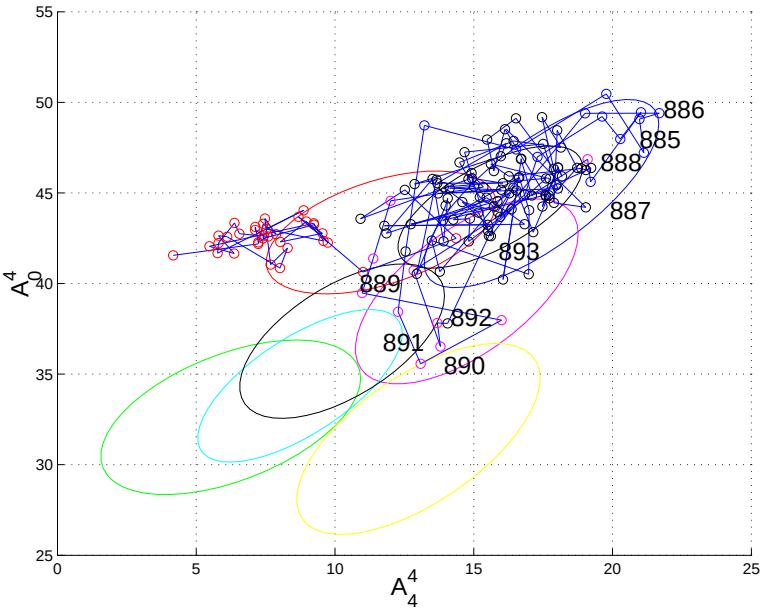


Figure 6.52: Trajectory of most likely configuration feature vector (Z_c) for sequence 5 (for which model 3 has highest posterior). The level curves of the Gaussian output covariances for model 3 are also shown.

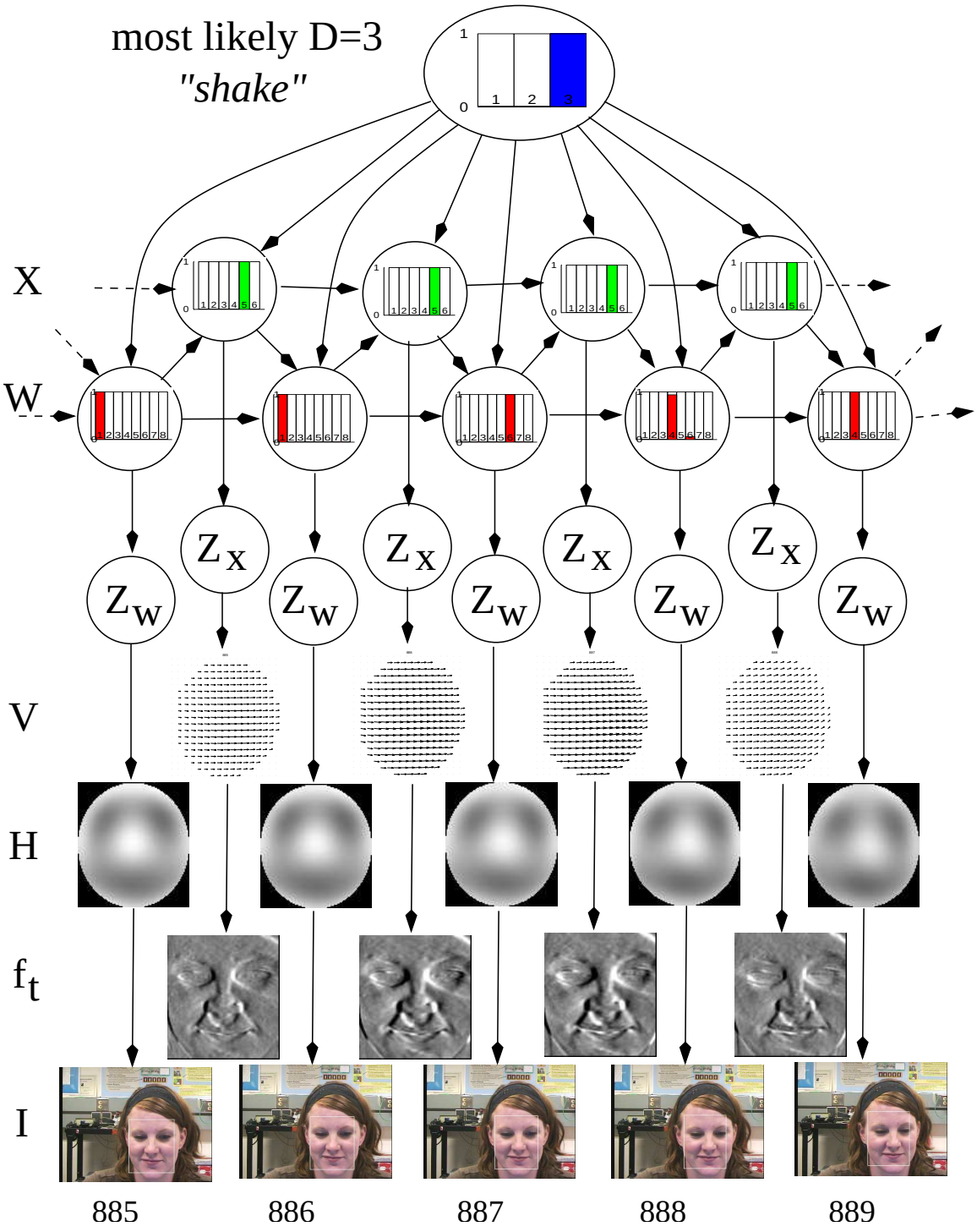


Figure 6.53: The C4MGs explanation of part of a sequence during the card matching game. The subject's face swings to the right during a shaking motion.

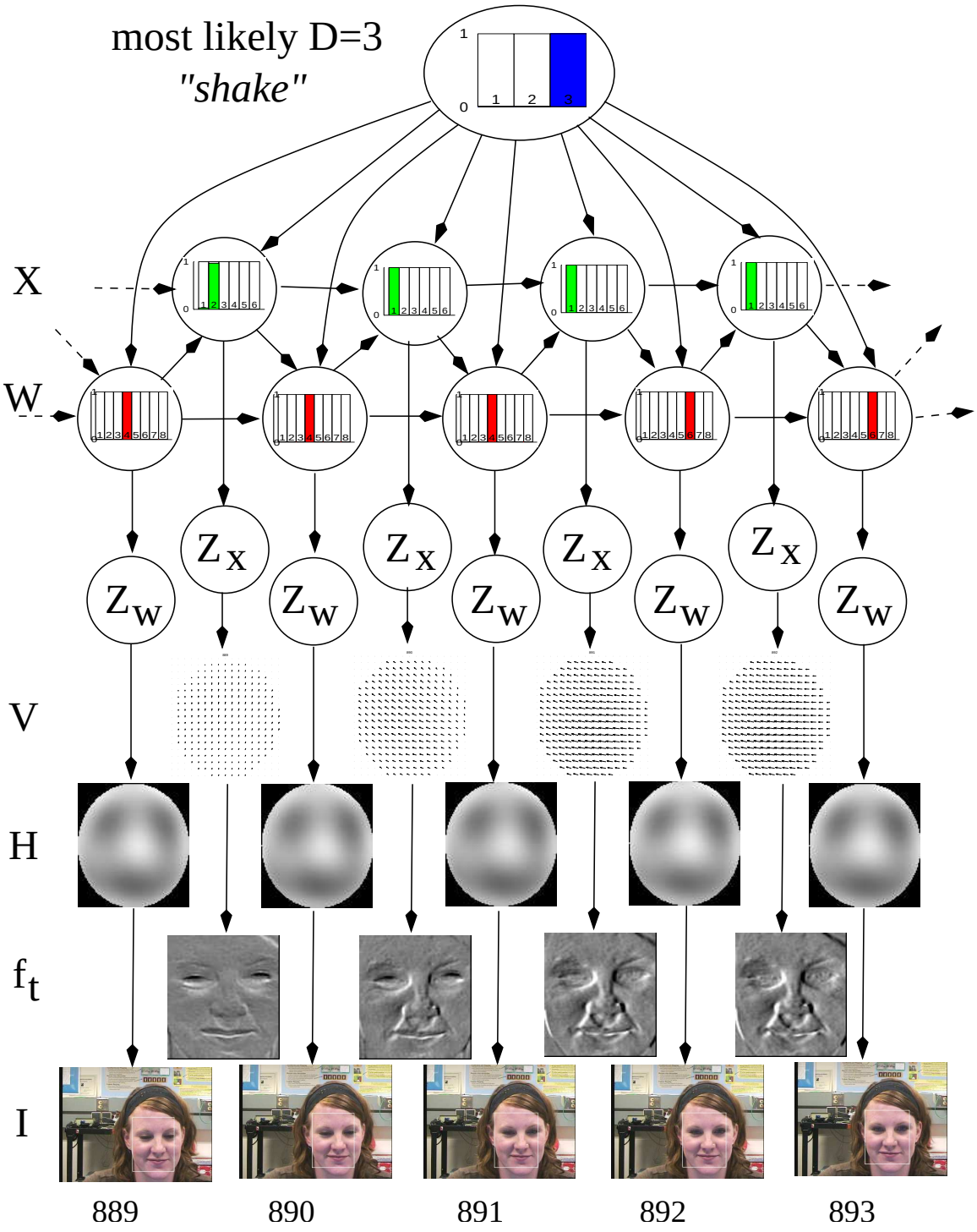


Figure 6.54: The C4MGs explanation of part of a sequence during the card matching game. The subject's face swings to the left during a shaking motion.

Chapter 7

Conclusions

Computer vision has traditionally focussed on recognition for its own sake. Computer vision modules are designed to perform specific, pre-defined tasks in isolation from the system they are to be used in. This approach tends to drive computer vision research into areas where it may not be entirely necessary to go. If computer vision is being used to achieve some goal, then it is the goal itself that should define what the computer vision task should be, not some pre-defined notion of what types of inputs will be needed to accomplish the goal. Goals usually change over time, due to learning, a changing environment, or changing resource bounds. Computer vision systems must therefore be able to self-modify to reflect these changes. This implies that decoupling the task from the goal is not the optimal approach, and we have argued in this thesis for a model that unifies the low-level computer vision tasks with the high-level goal oriented systems in a coherent whole using Bayesian networks.

For example, a security system may have the goal of detecting subversive actions. The traditional way of approaching this problem is to first have experts in decision-making for security *define* subversive action, and a set of inputs that would be sufficient for its detection. Computer vision researchers then build systems to detect these inputs. For example, the decision experts may deem that face recognition, and person detection and tracking are important, and computer vision systems for accomplishing this will be implemented. However, these types of systems are inflexible, in that they cannot easily be changed to reflect new techniques for subversion, new visual inputs, or changing resource bounds.

In contrast, the models we describe in this thesis require a different design approach. The decision experts first define the security task as a set of states that the system can be in, and the conditional dependencies between the states. The computer vison inputs do not need to be explicitly defined, only included in the model as generic 'visual information' variables. The model is then trained in

the actual environment it is to be used in, and the learned model 'discovers' the categories of visual inputs which are important to the task. This type of analysis unifies the decision process with the learning of computer vision models.

The model developed in this thesis combines computer vision, probabilistic modeling and decision theory. The three are closely related, and together are a powerful solution concept for vision-based computer tasks.

Computer Vision The computer vision task is to detect human motions from video streams, and to compress the motions into spatially and temporally abstract representations suitable for integration with higher level processing. This thesis has investigated data-independent sets of basis functions for simultaneously modeling instantaneous configuration and dynamics of human facial displays and hand gestures.

Probabilistic Models The uncertainties inherent in the sensing task call for probabilistic modeling techniques. Dynamic Bayesian networks (DBNs) in particular are a structured representation of uncertain beliefs for sequential data. I have shown how to use DBNs to model human behaviors in video sequences, allowing for temporally abstract representations of sequences of motion to be built automatically

Decision Theory Rational decisions require that non-verbal human behaviors must not only be sensed and represented, but must also be used for choosing actions which optimize utility over outcomes. Expected utility maximization is the standard approach for rational decision making, and forms the basis for Markov decision processes (MDPs). MDPs, and their close relatives, POMDPs (partially observable MDPs), have become the semantic model of choice for representing large planning problems. Our work on structured representations of MDPs has allowed them to be applied to larger problems than before. In particular, the research presented in this thesis has focussed on representing human non-verbal interactions with POMDPs.

This thesis has shown how partially observable Markov decision processes, or POMDPs, can be used to combine computer vision, probabilistic modeling and decision theory. The model allows an agent to incorporate actions and utilities into the sensing and representation of visual observations, and provides top-down value-based evidence for the learned probabilistic models: the agent can learn models most conducive for achieving value in a particular task. One of the key features of this technique is that it does not require labeled data sets. That is, the model makes no prior assumptions about the form or number of non-verbal behaviors used in an interaction, but rather *discovers* this from the data during training.

This model is of interest to researchers in both computer vision and decision theory. Computer vision scientists will find a new model for human action in video streams, and a method for learning the model from data. Further, they will see how the model can be attached to a high-level decision process that implicitly defines the computer vision task. This definition is a general one: the systems must be designed so that they can learn from a set of training data. Performance can be explicitly evaluated on a task, giving the computer vision researchers solid feedback on their algorithms. Decision theorists, on the other hand, will find an output model that brings a large and important data source into contact with their models. While they have been traditionally focussing on solution techniques for "toy" problems, they are always interested in real data. The model we have presented in this thesis connects them to vision data, and opens the door to research on much more difficult problem solutions than have usually been attempted.

7.1 Contributions

The main contribution of this thesis is a novel and unified model of human non-verbal behavior in video streams, integrated with decision making over high level context states and actions. This includes a Bayesian *a-posteriori* learning procedure for adapting the parameters of the model to training data. The learning does not require labeled data, so the model automatically discovers the categories of behaviors relevant to a particular task. The learned models can be used to predict human behavior, or by an autonomous agent to choose actions during collaboration with a human. We have demonstrated this model in three interactive settings.

1. During play of an imitation game, a human tries to mimic the facial displays of an animated cartoon face. The resulting displays involve some complex facial motions, and we used this game primarily to demonstrate the computer vision modeling techniques. We showed how our model could represent the facial motions during the four different imitative displays, and we demonstrated the function of the different components of the model. Finally, we performed a quantitative analysis on this data, showing prediction of the cartoon displays on unseen data, with rates comparable to the equivalent supervised experiments, in which the data is labeled.

2. Robot control using gestures is the second "game". An operator signals navigation commands to a robot using hand gestures, and rewards the robot for performing the correct action after each gesture. The robot learns the gesture categories, their number, and their relationships to its actions and utility

functions. It can then use the learned model to take actions after subsequent gestural commands. We performed a leave-one-out cross-validation experiment and measure how much reward the robot would gain by taking these actions on unseen data. We find error rates of only 2% over 48 test sequences.

3. The card matching game is played by two humans through a computer interface. The players can see, but not hear, one another. They are allowed to communicate through this visual link, but no restrictions are placed on the type of communication. Each player has a hand of three cards, but can only see their own cards, not their partner's. Each player gets to play a single card, and they both win if the suits of their played cards match. The idea is that the players must somehow agree, using only visual communication, about which card to play. This game involves fairly simple facial displays, but has a more complex decision structure, and we used it to demonstrate how computer vision is integrated with decision theory. We trained the model on a set of three games and tested it on a fourth. The model learned categories of motion that made intuitive sense in the card matching game, and the learned POMDP conditional probability tables showed the kind of structure one would expect in the game. Further, it predicted the actions of the human players correctly in all but one case out of seven on test data.

This work has also introduced a novel probabilistic method for spatially abstracting optical flow and images to a set of basis functions, including a Bayesian feature weighting technique to avoid prior selection of features. We have shown how this probabilistic modeling technique is a part of the general Bayesian solution to the problem of recognising and interpreting video sequences. Further, we have shown how a particular basis set, the Zernike polynomials, are useful descriptors of both optical flow and images, and have argued the advantages of using a pre-determined set of functions, as opposed to a data-dependent one.

7.2 Future Work

This thesis researches the unification of computer vision and decision theory for modeling human non-verbal displays. We briefly examine the future challenges in the computer vision and decision theoretic fields, and then describe some of the open problems when the two are combined.

7.2.1 Modeling human non-verbal behaviors in context

Learning decision-theoretic models of human behavior from unlabeled training data presents a number of different challenges. These center in particular on segmentation and tracking, as well as multimodal analysis.

Spatial Segmentation and Representation

There is a tradeoff in modeling human behavior between spatial segmentation and spatial representation. A holistic approach, such as we have taken in this thesis, spatially segments the entire region of interest (such as the face, or the hands) from the background, and represents the motion and spatial structure in this region using a relatively high dimensional feature vector. However, it is equally possible to break the region in which the behavior is occurring into smaller areas, modeling motion and intensity in each piece using a lower dimensional vector. For example, the facial display models we have presented involve representing the entire facial region using the basis of Zernike polynomials. We have also experimented with segmenting the face into a number of smaller regions over eyes and mouth. The corresponding Zernike projections can each be lower dimensional, but need to be combined into a single vector. It is not yet clear how this tradeoff can be optimally exploited. Future work will involve investigating the advantages of each, and methods for automatically segmenting the images spatially. The spatial segmentation/representation problem is closely related to the tracking problem.

Tracking

Throughout this thesis, we have assumed that a spatial region of interest has been tracked through each video sequence. We have described a method for tracking based on optical flow and exemplar matching (Section 3.7), which effectively decouples tracking from recognition. In fact, this is a crude approximation: tracking, spatial segmentation, recognition and decision-making are highly interrelated. To see why, recall that it is advantageous to describe decision making processes in terms of a small number of high-level, discrete states. The mapping between the high-dimensional, continuous video signal and this small high-level state space has been one of the central topics of this thesis. This abstraction of the visual space must proceed such that the high-level state space is sufficient for decision-making. There may be many events in the video sequence, and it is the job of the tracking and spatial segmentation processes to separate these events from background. The separation, however, depends on the task being performed. We have briefly described a method for integrating the tracking and recognition process in Section 3.7.2. This

technique combines tracking with learning of the model. Value directed learning would further imply that the track would be geared towards achieving value in the high-level task. These issues would require substantial future work, and would be related to the issues in solving POMDPs described below (Section 7.2.2).

Observation Function

The observation function we have presented in this thesis is one example of the type of function that may be desirable for modeling human behavior. We have described how it models instantaneous pose and dynamics simultaneously, and how these two are useful for recognition and for tracking. However, there are other types of information which may need to be included for other application areas. Edges, corners, invariant features, color, and texture are all examples. Further, the modeling of both pose and dynamics leads to questions about the role of each in the recognition task. We have performed experiments attempting to delve further into these issues, but the results were inconclusive, and would require further work.

Multimodal communication

As was pointed out in the introduction, speech, gesture and facial displays are highly correlated. We have not investigated this correlation in this thesis, but combining cues from these multiple communication modalities would be necessary in future work. For example, one of the major components of facial motion during conversation is simply due to the motion of the mouth for speech [VBPB03]. Gestures are also commonly used for emphasis and commentary during speech [McN92]. The models we have described in this thesis could be extended to integrate these different modalities. However, the question remains of the level at which to integrate them. One possibility is to combine them at the level of the continuous feature vectors, another is to combine them at the level of the decision-making, using each as high-level, discrete conditioning factors. Some cues may be more amenable to combination at a low level, others at a high level.

Temporal segmentation

As described in Section 3.6.4, the input video sequences are temporally segmented using only the observable high-level context and actions. However, we saw examples in Chapter 6 in which this caused some problems. In particular, the assignment of a facial display to a particular video segment requires that there be some information about that segment containing a 'meaningful' facial display. However, there are cases in which the subjects were moving their heads or faces for reasons other than

communication. This could be detected using models which are common for dialogue management, distinguishing different levels of communication [Cla96, PH00]. These models essentially build a hierarchy of attentional levels, from the channel level for the perception and generation of behaviors, to the conversation level, at which propositional content is modeled. A conversation must include a connection between partners at all levels. Detection and tracking of eyes could also be very useful in this regard [Ver03].

7.2.2 Optimal or approximate high-dimensional POMDP solutions

The models we have described are POMDPs with extremely high-dimensional output spaces. Optimal and approximate solutions to these models lie at the frontier of POMDP research. One of the main research directions is in methods to compress belief spaces, allowing for more tractable approximate solutions while not compromising decision quality [PB03]. Since belief spaces are abstractions of the output spaces, the belief state compressions can give indications of how to construct representations of the video sequences which are most conducive to achieving value within the POMDP's task. Thus, solutions to the POMDPs are intimately connected with the learning of the models, including the representations of vision-based observations: a rational learning agent must learn the POMDPs and solve them concurrently. Future work will involve investigating belief state compression techniques and their relationship to video outputs. The goal will be learning and solution techniques which construct sparse representations of visual data that lead to optimal decisions for obtaining value in the long term.

7.2.3 Unification of decision theory and computer vision for embodied human-interactive agents

Artificial agents that provide an effortless interface between humans and machines will be useful in two major areas. First, intelligent agents are already being used to help humans in difficult situations, and their roles will increase in the future with the advent of more visually driven systems. For example, agents can increase automobile safety by monitoring and interacting with drivers, or can help people with mental disabilities perform everyday tasks with dignity. Second, the study of interactive agents provides a wealth of information about human psychology. Watching humans interact with simple machines gives insight into the workings of the human brain.

Intelligent agents should be able to sense, interpret, act in conjunction with, and learn from a complex and changing environment containing other intelligent agents, particularly humans. The long term goals of this research are to under-

stand how agents can learn and use relationships between visual observations of other agents, their actions, other contextual information and utility. Learning these relationships will allow agents to adapt to new environments, new tasks, and new collaborators, without manual intervention. As this thesis has argued, computer vision alone is not sufficient, but must be combined with decision theory. The use of probabilistic modeling techniques consolidates the two, and allows the whole to be described on sound theoretical ground. In turn, the decision theoretic modeling is useful for the vision modeling, as it allows value-directed structures to be learned.

The experimental paradigm we have presented can be used in future work to investigate more complex interactions involving non-verbal behaviors. Games involving gestures, facial displays and speech will be important for these investigations.

Bibliography

[ABMM03] Alexei A. Afros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *Proceedings of International Conference on Computer Vision*, pages 726–733, Nice, France, 2003.

[Ari] Aristotle. Rhetoric. Written circa 350BC.

[ASKP03] Jonathan Alon, Stan Sclaroff, George Kollios, and Vladimir Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of Intl. Conference on Computer Vision and Pattern Recognition 2003*, pages 682–688, Madison, Wisconsin, June 2003.

[Aus62] J.L. Austin. *How to do things with words.* Oxford University Press, 1962.

[BA96] Michael Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, January 1996.

[Bar01] Marian Stewart Bartlett. *Face Image Analysis by Unsupervised Learning.* Kluwer Academic Publishers, Norwell, MA, 2001.

[BB95] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, 1995.

[BCdR03] Peter Brusilovsky, Albert Corbett, and Fiorella de Rosis, editors. *Proceedings of 9th International Conference on User Modeling*, volume 2702 of *LNCS*. Springer-Verlag, January 2003.

[BCS97] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: Driving visual speech with audio. *Computer Graphics*, 31(Annual Conference Series):353–360, 1997.

[BD01] A.F. Bobick and J.W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(3), March 2001.

[Bel57] Richard E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, 1957.

[BF95] Yoshua Bengio and Paolo Frasconi. Diffusion of context and credit information in Markovian models. *Journal of Artificial Intelligence Research*, 3:249–270, 1995.

[BF96] Y. Bengio and P. Frasconi. Input-output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, September 1996.

[BFB94] J.L. Barron, D.J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[BH96] A. Baumberg and D. Hogg. Generating spatio-temporal models from examples. *Image and Vision Computing*, 14(8):525–532, 1996.

[BHK97] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projections. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):711–720, July 1997.

[BI98] A.F. Bobick and Y.A. Ivanov. Action recognition using probabilistic parsing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 196–202, Santa Barbara, CA, 1998.

[Bir98] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 203–209, Santa Barbara, CA, 1998.

[BLB+03] M.S. Bartlett, G. Littlewort, B. Braathen, T.J. Sejnowski, and J.R. Movellan. A prototype for automatic recognition of spontaneous facial actions. In S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 382–386. MIT Press, 2003.

[Bob01] Miroslaw Bober. Mpeg-7 visual shape descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):716–719, June 2001.

[BOP97] Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden Markov models for complex action recognition. In *International Conference on Computer Vision and Pattern Recognition*, Puerto Rico, 1997.

[Bra97] Matthew Brand. Learning concise models of human activity from ambient video via a structure-inducin m-step estimator. Technical Report TR-97-25, Mitsubishi Electric Research Laboratory, November 1997.

[Bra99a] Matthew Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11:1155–1182, 1999.

[Bra99b] Matthew Brand. Voice puppetry. In *Proc. ACM SIGGRAPH*, pages 21–28, August 1999.

[Bre97] Chris Bregler. Learning and recognising human dynamics in video sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 568–574, 1997.

[BS95] A. Bell and T. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[BS99] Cynthia Breazeal and Brian Scassellati. A context-dependent attention system for a social robot. In *Proceedints of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI99)*, pages 1146–1151, Stockholm, Sweden, 1999.

[BSA91] S.O. Belkasim, M. Shridhar, and M. Ahmadi. Pattern recognition with moment invariants: A comparative study and new results. *Pattern Recognition*, 24(12):1117–1138, 1991.

[BY97] Michael Black and Yaser Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motions. *International Journal of Computer Vision*, 25(1):23–48, 1997.

[Cas00] Justine Cassell. More than just another pretty face: Embodied conversational agents. *Communications of the ACM*, 43(4):70–78, 2000.

[CBC+00] Justine Cassell, Timothy Bickmore, Lee Campbell, Hannes Vilhjámsson, and Hao Yan. Human conversation as a system framework: Designing embodied conversational agents. In Justine Cassell, Joseph Sullivan, Scott Prevost, and Elizabeth Churchill, editors, *Embodied Conversational Agents*, chapter 2, pages 29–63. MIT Press, 2000.

[CD00] Ross Cutler and Larry Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):348–362, 2000.

[CdFGT03] P. Carbonetto, N. de Freitas, P. Gustafson, and N. Thompson. Bayesian feature weighting for unsupervised learning with application to object recognition. In C M Bishop and B J Frey, editors, *Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, Florida, January 2003.

[CET98] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *Lecture Notes in Computer Science*, 1407:484–490, 1998.

[CHI03] *Proc. SIGCHI*, 2003.

[Cho91] Nicole Chovil. Social determinants of facial displays. *Journal of Nonverbal Behavior*, 15(3):141–154, Fall 1991.

[Cla96] Herbert H. Clark. *Using Language*. Cambridge University Press, Cambridge, UK, 1996.

[CLZ97] Anthony Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental pruning: A simple, fast exact method for partially observable Markov decision processes. In *Proceedings of Uncertainty in Artificial Intelligence*, 1997.

[CP99] Brian Clarkson and Alex Pentland. Unsupervised clustering of ambulatory audio and video. In *Proc. ICASSP*, 1999.

[CR96] J.M. Carroll and J.A. Russell. Do facial expressions signal specific emotions? Judging the face in context. *Journal of Personality and Social Psychology*, 70:205–218, 1996.

[CSG+03] Ira Cohen, Nice Sebe, Ashutosh Garg, Lawrence S. Chen, and Thomas S. Huang. Facial expression recognition from video sequences: temporal and static modeling. *Computer Vision and Image Understanding*, 91:160–187, 2003.

[CSPC00] Justine Cassell, Joseph Sullivan, Scott Prevost, and Elizabeth Churchill, editors. *Embodied Conversational Agents*. MIT Press, 2000.

[CT98] Ross Cutler and Matthew Turk. View-based interpretation of real-time optical flow for gesture recognition. In *Proc. Intl. Conference on*

Automatic Face and Gesture Recognition, pages 98–104, Nara, Japan, April 1998.

[CVP03] *IEEE Workshop on Computer Vision and Pattern Recognition for Human Computer Interaction*, Madison, Wisconsin, June 2003.

[Dar72] Charles Darwin. *Expressions of the Emotions in Man and Animals*. Abermale, London, 1872.

[DB97] Richard Dearden and Craig Boutilier. Abstraction and approximate decision theoretic planning. *Artificial Intelligence*, 89:219–283, 1997.

[DBH+99] Gianluca Donato, Marian Stewart Bartlett, Joseph C. Hager, Paul Ekman, and Terrence J. Sejnowski. Classifying facial actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(10):974–989, October 1999.

[DEP96] Trevor J. Darrell, Irfan A. Essa, and Alex P. Pentland. Task-specific gesture analysis in real-time using interpolated views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1236–1242, 1996.

[DH01] Vincent E. Devin and David C. Hogg. Reactive memories: An interactive talking head. In *Proc. British Machine Vision Conference*, pages 234–243, 2001.

[dJ32] Andrea de Jorio. *Gesture in Naples and gesture in classical antiquity : a translation of "La mimica degli antichi investigata nel gestire napoletano", Gestural expression of the ancients in the light of Neapolitan gesturing, translated by Adam Kendon*. Indiana University Press, Boomington, Indiana, 2000 (Original 1832).

[DNR77] A.P. Dempster, N.M.Laird, and D.B. Rubin. Maximum likelihood from incomplete data using the EM algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.

[DP96] Trevor Darrell and Alex Pentland. Active gesture recognition using partially observable Markov decision processes. In *13th IEEE Intl. Conference on Pattern Recognition*, Vienna, Austria, 1996.

[EFE72] Paul Ekman, Wallace V. Friesen, and P. Ellsworth. *Emotion in the Human Face*. Pergamon, New York, 1972.

[EHL+02] Pantelis Elinas, Jesse Hoey, Darrell Lahey, Jeff Montgomery, Don Murray, Stephen Se, and James J. Little. Waiting with José, a vision based mobile robot. In *Proc. International Conference on Robotics and Automation (ICRA)*, pages 678–682, Washington, D.C., May 2002.

[EHL03] Pantelis Elinas, Jesse Hoey, and James J. Little. Homer: Human oriented messenger robot. In *Proc. AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments*, Stanford, CA, March 2003.

[EP97] I.A. Essa and A. Pentland. Coding analysis, interpretation, and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):757–763, July 1997.

[ER97] P. Ekman and E.L. Rosenberg. *What the Face Reveals*. Oxford Univ. Press, New York, 1997.

[EW78] Paul Ekman and W.Friesen. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press, Palo Alto, CA, 1978.

[FBYJ00] David J. Fleet, Michael J. Black, Yaser Yacoob, and Allan D. Jepson. Design and use of linear models for image motion analysis. *International Journal of Computer Vision*, 36(3):171–193, 2000.

[FG02] *Proceedings of IEEE International Conference on Face and Gesture*, Washington, DC, May 2002.

[FL03] Beat Fasel and Juergen Luettin. Automatic facial expression analysis: A survey. *Pattern Recognition*, 36(1):259–275, 2003.

[FND03] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3-4):143–166, March 2003.

[Fri94] Alan J. Fridlund. *Human facial expression: an evolutionary view*. Academic Press, San Diego, CA, 1994.

[Fri97] Alan J. Fridlund. The new ethology of human facial expressions. In James A. Russell and Jose Miguel Fernández-Dols, editors, *The Psychology of Facial Expression*, chapter 5, pages 103–129. Cambridge University Press, Cambridge, UK, 1997.

[FST98] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical Hidden Markov Model: Analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

[Gav99] D.M. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[GJH99] Aphrodite Galata, Neil Johnson, and David Hogg. Learning structured behaviour models using variable length Markov models. In *IEEE Workshop on Modelling People*, Corfu, Greece, September 1999.

[GJH01] Aphrodite Galata, Neil Johnson, and David Hogg. Learning variable-length Markov models of behavior. *Computer Vision and Image Understanding*, 81(3):398–413, 2001.

[GL00] Piotr J. Gmytrasiewicz and Christine L. Lisetti. Using decision theory to formalize emotions for multi-agent system applications: preliminary report. In *Second ICMAS-2000 Workshop on Game Theoretic and Decision Theoretic Agents*, Boston, 2000.

[GLW01] John Gottman, Robert Levenson, and Erica Woodin. Facial expressions during marital conflict. *Journal of Family Communication*, 1(1):37–57, 2001.

[Gmy02] Piotr J. Gmytrasiewicz. Toward optimal planning in multiagent environments: Basic framework. Technical report, University of Chicago, Chicago, IL, November 2002.

[Hau00] Milos Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[HF00] Eric A. Hansen and Zhengzhu Feng. Dynamic programming from pomdps using a factored state representation. In *Proc. Fifth International Conference on Artifical Intelligence Planning and Scheduling*, Breckenridge, CO, April 2000.

[HL00] Jesse Hoey and James J. Little. Representation and recognition of complex human motion. In *Proc. IEEE CVPR*, pages 752–759, Hilton Head, SC, June 2000.

[HL03] Jesse Hoey and James J. Little. Bayesian clustering of optical flow fields. In *Proc. International Conference on Computer Vision (ICCV)*, pages 1086–1093, Nice, France, October 2003.

[HL04] Jesse Hoey and James J. Little. Decision theoretic modeling of human facial displays. In *Proc. European Conference on Computer Vision (ECCV)*, Prague, CZ, 2004.

[Hoe01] Jesse Hoey. Hierarchical unsupervised learning of facial expression categories. In *Proc. ICCV Workshop on Detection and Recognition of Events in Video*, pages 99–106, Vancouver, Canada, July 2001.

[Hoe02] Jesse Hoey. Clustering contextual facial display sequences. In *Proceedings of IEEE International Conference on Face and Gesture*, Washington, DC, May 2002.

[Hoe04] Jesse Hoey. Value-directed learning of gestures and facial displays. In *Under Review for IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, 2004.

[HS81a] H.V. Henderson and S.R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23:53–60, 1981.

[HS81b] B.K.P. Horn and B.G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[HSAHB99] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 99)*, Stockholm, 1999.

[HSJ95] E. Hunter, J. Schlenzig, and R. Jain. Posture estimation in reduced-model gesture input systems. In *International Workshop on Automatic Face- and Gesture-Recognition*, pages 290–295, Zurich, Switzerland, 1995.

[ICM03] *International Conference on Multimodal Interfaces*, Vancouver, BC, November 2003. ACM Press.

[Iza97] Carroll E. Izard. Emotions and facial expressions: A perspective from differential emotions theory. In James A. Russell and Jose Miguel Fernández-Dols, editors, *The Psychology of Facial Expression*, chapter 5, pages 103–129. Cambridge University Press, Cambridge, UK, 1997.

[JBY96] Shannon Ju, Michael Black, and Yaser Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proc. IEEE*

Conference on Automatic Face and Gesture Recognition, pages 38–44, 1996.

[Jen02] Cullen Jennings. *Probabilistic evidence combination for robust real time finger recognition and tracking*. PhD thesis, University of British Columbia, November 2002.

[JFEM01] Allan D. Jepson, David J. Fleet, and Thomas F El-Maraghi. Robust online appearance models for visual tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 415–422, Kauai, Hawaii, 2001.

[JP98] A. Jebara and A. Pentland. Action reaction learning: Analysis and synthesis of human behaviour. In *IEEE Workshop on The Interpretation of Visual Motion*, 1998.

[Kal60] Emil Kalman, Rudolph. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[KG77] R.M. Krauss and S. Glucksberg. Social and nonsocial speech. *Scientific American*, 236:100–105, 1977.

[Kje01] Rick Kjeldsen. Head gestures for computer control. In *Proceedings of the Workshop on Recognition And Tracking of Face and Gesture – Real Time Systems (RATFG-RTS)*, pages 61–67, Vancouver, Canada, July 2001.

[KK00] Whoi-Yul Kim and Yong-Sung Kim. A region-based shape descriptor using Zernike moments. *Signal Processing: Image Communication*, 16:95–102, 2000.

[KLC98] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[KLM96] Leslie Pack Kaelbling, Michael Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[KM00] Ioannis Kakadiaris and Dimitris Metaxas. Model-based estimation of 3d human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1453–1459, 2000.

[Koh89] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1989.

[KZ02] Volker Krüger and Shaohua Zhou. Exemplar-based face recognition from video. In *Proceedings of IEEE International Conference on Face and Gesture*, Washington, DC, May 2002.

[LB98] James J. Little and Jeffrey Boyd. Recognizing people by their gait: the shape of motion. *Videre*, 1(2), Winter 1998.

[LB99] Cen Li and Gautam Biswas. Temporal pattern generation using hidden Markov model based unsupervised classification. In D.J. Hand, J.N. Kok, and M.R. Berthold, editors, *Proc ICML*, number 1642 in LNCS, pages 245–256, Berlin, 1999. Springer-Verlag.

[LBA99] M.J. Lyons, J. Budynek, and S. Akamatsu. Automatic classification of single facial images. *PAMI*, 21(12):1357–1362, December 1999.

[LF98] Michael E. Leventon and William T. Freeman. Bayesian estimation of 3-d human motion from and image sequence. Technical Report TR-98-06, Mitsubishi Electric Research Laboratory, July 1998.

[LK81] Bruce Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. International Joint Conference on Artificial Intelligence*, 1981.

[LKCL98] James Jenn-Jier Lien, Takeo Kanade, Jeffrey F. Cohn, and Ching-Chung Li. Subtly different facial expression recognition and expression intensity estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 443–449, Santa Barbara, CA, 1998.

[LP98] Simon X. Liao and Miroslaw Pawlak. On the accuracy of Zernike moments for image analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(12):1358–1364, December 1998.

[LS00] Christine L. Lisetti and Diane J. Schiano. Automatic facial expression interpretation: Where human-computer interaction, artificial intelligence and cognitive science interact. *Pragmatics and Cognition*, 8(1):185–235, 2000.

[LSR+00] Bastian Leibe, Thad Starner, William Ribarsky, Zachary Wartell, David Krum, Justin Weeks, Brad Singletary, and Larry Hodges. Towards spontaneous interaction with the perceptive workbench, a semi-immersive virtual environment. In *IEEE Virtual Reality*, pages 13–20, New Brunswick, NJ, March 2000.

[LT97] Jeurgen Luettin and Neil A. Thacker. Speechreading using probabilistic models. *Computer Vision and Image Understanding*, 65(2):163–178, February 1997.

[LTC97] A. Lanitis, C.J. Taylor, and T.F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):743–756, July 1997.

[McN92] David McNeill. *Hand and Mind: What Gestures Reveal about Thought*. University of Chicago Press, Chicago, IL, 1992.

[Min99] Thomas P. Minka. From hidden Markov models to linear dynamical systems. Technical Report 531, M.I.T., 1999.

[MN95] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[Mor82] Masahiro Mori. *The Buddha in the Robot*. Charles E. Tuttle Co., 1982.

[MP91] Kenji Mase and Alex Pentland. Recognition of facial expression from optical flow. *IEICE Transactions E*, 74(10):3474–3483, 1991.

[MP01] Kevin Murphy and Mark Paskin. Linear time inference in hierarchical HMMs. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, Vancouver, BC, 2001.

[MPR+02] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma. Experiences with a mobile robotic guide for the elderly. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[Mur02] Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, July 2002.

[Muy87] Eadweard Muybridge. *Animal Locomotion*. University of Pennsylvania, Philadephia, 1887.

[MYD96] Carlos Morimoto, Yaser Yacoob, and Larry Davis. Recognition of head gestures using hidden Markov models. In *Proceeding of ICPR*, pages 461–465, Austria, 1996.

[Mye91] Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, Massachussetts, 1991.

[NA94] S.A. Niyogi and E.H. Adelson. Analyzing and recognizing walking figures in xyt. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 469–474, 1994.

[NH93] Radford M. Neal and Geoffrey E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. Technical report, Dept. of Computer Science, University of Toronto, 1993.

[NI00] A. Nefian and M. Hayes III. Maximum likelihood training of the embedded HMM for face detection and recognition. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 153–160, 2000.

[NY93] Shahriar Negahdaripour and Chih-Ho Yu. A generalized brightness change model for computing optical flow. In *Proceedings of Fourth International Conference on Computer Vision*, pages 2–11, Berlin, Germany, May 1993.

[OHG02] Nuria Oliver, Eric Horvitz, and Ashutosh Garg. Layered representations for human activity recognition. In *Proceedings of International Conference on Multimodal Interfaces*, Pittsburgh, PA, October 2002.

[OPB97] N. Oliver, A. Pentland, and F. Berard. LAFTER: Lips and face real time tracker. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 977–984, 1997.

[PB03] Pascal Poupart and Craig Boutilier. Value-directed compression of POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 1547–1554, Vancouver, 2003.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[Pen00] Alex Pentland. Looking at people: sensing for ubiquitous and wearable computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):107–119, 2000.

[PFH99] Vladimir Pavlovic, Brendan J. Frey, and Thomas S. Huang. Variational learning in mixed-state dynamic graphical models. In *Proc. Uncertainty in Artificial Intelligence (UAI)*, pages 522–530, Stockholm, Sweden, July 1999. Morgan Kaufmann.

[PH00] Tim Paek and Eric Horvitz. Conversation as action under uncertainty. In *Proceedings of Uncertainty in Artificial Intelligence*, Stanford, CA, June 2000.

[Plu03] Mark Plutowski. Mdp solver for a class of location-based decisioning tasks. In *Uncertainty in Artificial Intelligence Bayesian Modeling Applications Workshop*, Mexico, August 2003.

[PMS94] Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *PRoc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 199–204, Seattle, WA, June 1994.

[PP00] Isabella Poggi and Catherine Pelachaud. Performative facial expressions in animated faces. In Justine Cassell, Joseph Sullivan, Scott Prevost, and Elizabeth Churchill, editors, *Embodied Conversational Agents*, chapter 6, pages 155–187. MIT Press, 2000.

[PR89] Aluizio Prata and W.V.T. Rusch. Algorithm for computation of Zernike polynomials expansion coefficients. *Applied Optics*, 28(4):749–754, February 1989.

[PR00] M. Pantic and L.J.M. Rothkrantz. Automatic analysis of facial expressions: The state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12), December 2000.

[PSH97] Vladimir Pavlovic, R. Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligenc (PAMI)*, 19(7):677–695, July 1997.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, New York, NY., 1994.

[RA00] Yong Rui and P. Anandan. Segmenting visual actions based on spatio-temporal motion patterns. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, Hilton Head, SC, June 2000.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–296, February 1989.

[RB99] J Rittscher and Andrew Blake. Classification of human body motion. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 679–685, Corfu, Greece, 1999.

[RE01] Lionel Reveret and Irfan Essa. Visual coding and tracking of speech related facial motion. In *Proc of the IEEE CVPR International Workshop on Cues in Communication*, pages 221–229, Dec 2001.

[RFD97a] James A. Russell and Jose Miguel Fernández-Dols, editors. *The Psychology of Facial Expression*. Cambridge University Press, Cambridge, UK, 1997.

[RFD97b] James A. Russell and Jose Miguel Fernández-Dols. What does facial expression mean? In James A. Russell and Jose Miguel Fernández-Dols, editors, *The Psychology of Facial Expression*, chapter 1, pages 3–30. Cambridge University Press, Cambridge, UK, 1997.

[RH93] L.R. Rabiner and B.H. Huang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[Ris78] J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.

[RN96] Byron Reeves and Clifford Nass. *The media equation*. Cambridge University Press, 1996.

[RPT00] N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000)*, Hong Kong, 2000.

[Rus97] James A. Russell. Reading emotions from and into faces: Resurrecting a dimensional-contextual perspective. In James A. Russell and

Jose Miguel Fernández-Dols, editors, *The Psychology of Facial Expression*, chapter 13, pages 295–320. Cambridge University Press, Cambridge, UK, 1997.

[SAH91] E.P. Simoncelli, E.H. Adelson, and D.J. Heeger. Probability distributions of optical flow. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pages 310–315, Mauii, Hawaii, USA, 1991.

[SAHB00] Robert St-Aubin, Jesse Hoey, and Craig Boutilier. APRICODD: Approximate policy construction using decision diagrams. In *Neural Information Processing Systems*, volume 14, pages 1089–1095, 2000.

[Sch92] D. Schumacher. General filtered image rescaling. In D. Kirk, editor, *Graphics Gems II*. Harcourt Brace Jovanovich, 1992.

[SHJ94] J. Schlenzig, E. Hunter, and R. Jain. Vision based hand gesture interpretation using recursive estimation. In *Proc. Asilomar Conference on Signals, Systems and Computation*, pages 394–399, October 1994.

[Smy97] Padhraic Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, 1997.

[SO94] Andreas Stolcke and Stephen M. Omohundro. Best-first model merging for hidden Markov model induction. Technical Report TR-94-003, International Computer Science Institute, Berkeley, CA, January 1994.

[SP95] Thad Starner and Alex P. Pentland. Visual recognition of american sign language using hidden Markov models. In *International Workshop on Automatic Face and Gesture Recognition*, pages 189–194, Zurich, Switzerland, 1995.

[TC88] Cho-Huak Teh and Roland T. Chin. On image analysis by the methods of moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):496–513, July 1988.

[Tea80] Michael Reed Teague. Image analysis via the general theory of moments. *Journal of Optical Society of America*, 70(8):920–930, 1980.

[Thr00] S. Thrun. Monte Carlo POMDPs. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 12, pages 1064–1070. MIT Press, 2000.

[TK93] D. Terzopoulus and K.Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(6):569–579, June 1993.

[TKC01] Yingli Tian, Takeo Kanade, and Jeffrey F. Cohn. Recognizing action units for facial expression analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2), February 2001.

[TP91] Matthew Turk and Alex P. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[UR02] George W. Uetz and J. Andrew Roberts. Multisensory cues and multimodal communication in spiders: Insights from video/audio playback studies. *Brain, Behavior and Evolution*, 59:222–230, 2002.

[VBPB03] E. Vatikiotis-Bateson, P. Perrier, and G. Bailly, editors. *Advances in audio-visual speech processing*. MIT Press, Cambridge, MA, 2003.

[Ver03] R. Vertegaal. Attentive user interfaces. *Communications of ACM*, 46(3), March 2003.

[VM98] Christian Vogler and Dimitris Metaxas. ASL recognition based on a coupling between HMMs and 3D motion analysis. In *IEEE Intl. Conference on Computer Vision*, pages 363–369, Mumbai, India, 1998.

[VM99] Christian Vogler and Dimitris Metaxas. Parallel hidden Markov models for american sign language recognition. In *Proceedings International Conference on Computer Vision (ICCV)*, Corfu, Greece, September 1999.

[vNM53] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, 3 edition, 1953.

[VT02] M. Alex O. Vasilescu and Demetri Terzopoulus. Multilinear analysis of image ensembles: Tensorfaces. In A Heyden, editor, *Proc. of European Conference on Computer Vision (ECCV)*, volume 2350 of *LNCS*, pages 447–460. Springer-Verlag, 2002.

[vZ34] F. von Zernike. Beugungstheorie des schneidenvarfahrens und seiner verbesserten form, der pahsekontrastmethode. *Physica*, I:689–704, 1934.

[WA03] Hongcheng Wang and Narendra Ahuja. Facial expression decomposition. In *Proceedings of International Conference on Computer Vision*, pages 958–965, Nice, France, October 2003.

[WADP97] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[WB99] Andrew D. Wilson and Aaron F. Bobick. Parametric hidden Markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(9):884–900, September 1999.

[WBC97] Andrew D. Wilson, Aaron F. Bobick, and Justine Cassell. Temporal classification of natural gesture and application to video coding. In *Proc. CVPR*, Puerto Rico, June 1997.

[WCP00] Christopher R. Wren, Brian P. Clarkson, and Alex P. Pentland. Understanding purposeful human motion. In *Proc. Face and Gesture Recognition*, Grenoble, France, 2000.

[WHT03] Liang Wang, Weiming Hu, and Tieniu Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36:585–601, 2003.

[WPG01] Michael Walter, Alexandra Psarrou, and Shaogang Gong. Data driven gesture model acquisition using minimum description length. In *Proc. British Machine Vision Conference*, Manchester, UK, September 2001.

[YD94] Yaser Yacoob and Larry Davis. Computing spatio-temporal representations of human faces. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 65–69, 1994.

[ZCMG01] Bo Zhang, Qinsheng Cai, Jianfeng Mao, and Baining Guo. Planning and acting under uncertainty: A new model for spoken dialogue system. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 572–579, Seattle, WA, August 2001.

[ZL02] Dengsheng Zhang and Guojun Lu. An integrated approach to shape based image retrieval. In *Proceedings of 5th Asian Conference on Computer Vision (ACCV)*, Melbourne, Australia, January 2002.

Appendix A

Proof of Equation (4.3)

$$
\begin{aligned}
H(\theta^{''}|\theta') - H(\theta'|\theta') &= \sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y},\theta') \left[\log P(\mathbf{X}|\mathbf{y},\theta^{''}) - \log P(\mathbf{X}|\mathbf{y},\theta')\right] \\
&= \sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y},\theta') \log \left[\frac{P(\mathbf{X}|\mathbf{y},\theta^{''})}{P(\mathbf{X}|\mathbf{y},\theta')}\right] \\
&\leq \sum_{\mathbf{X}} P(\mathbf{X}|\mathbf{y},\theta') \left[\frac{P(\mathbf{X}|\mathbf{y},\theta^{''})}{P(\mathbf{X}|\mathbf{y},\theta')} - 1\right] \qquad (\text{using } \log x \leq x - 1) \\
&= \sum_{\mathbf{X}} \left[P(\mathbf{X}|\mathbf{y},\theta^{''}) - P(\mathbf{X}|\mathbf{y},\theta')\right] \\
&= 0
\end{aligned}
$$

This shows that $H(\theta^{''}|\theta') \leq H(\theta'|\theta')$. Notice that if we had used some distribution other than $P(\mathbf{X}|\mathbf{y},\theta')$ in Equation (4.1), this inequality would not hold. In such cases, one must maximize the expression in Equation (4.2) with respect to both this arbitrary distribution and the model parameters. This leads to a more general exposition of the EM algorithm [NH93].

Appendix B

Tracking updates

We are trying to estimate

$$\begin{aligned}\xi(\alpha_t, \nabla f_{t-1}) &= \int_{z_{x,t-1}} P(\alpha_t|\alpha_{t-1}z_{x,t-1})P(\nabla f_{t-1}|\alpha_{t-1}z_{x,t-1})P(z_{x,t-1}|X_{t-1}) \\ &= \int_{z_{x,t-1}v_{t-1}} P(\alpha_t|\alpha_{t-1}z_{x,t-1})P(\nabla f_{t-1}|\alpha_{t-1}v_{t-1})P(v_{t-1}|z_{x,t-1})P(z_{x,t-1}|X_{t-1})\end{aligned} \tag{B.1}$$

All the terms in this equation are Gaussian distributions, and so we can write the intergrand as a single exponential, $e^{\frac{1}{2}\gamma}$ If we write $\delta_\alpha = \alpha_t - \alpha_{t-1}$, and drop the temporal subscript, the exponent is

$$\begin{aligned}\gamma = &-(\delta_\alpha - z)'\Lambda_\alpha^{-1}(\delta_\alpha - z) - (f_\tau + f_s v)'A^{-1}(f_\tau + f_s v) \\ &- (v - Mz)'\Lambda_p^{-1}(v - Mz) - (z - \mu_{zx})'\Lambda_{z,x}^{-1}(z - \mu_{zx})\end{aligned}$$

Completing the squares over v gives

$$\begin{aligned}\gamma = &-(v - \mu_w)'\Lambda_w^{-1}(v - \mu_w) + \mu_w'\Lambda_w^{-1}\mu_w - (\delta_\alpha - z)'\Lambda_\alpha^{-1}(\delta_\alpha - z) \\ &- f_\tau'A^{-1}f_\tau - z'M'\Lambda_p^{-1}Mz - z'\Lambda_{z,x}^{-1}z \\ &+ 2\mu_{z,x}'\Lambda_{z,x}^{-1}z - \mu_{z,x}'\Lambda_{z,x}^{-1}\mu_{z,x}\end{aligned}$$

where, $w = f_s'A^{-1}f_\tau$ and

$$\begin{aligned}\Lambda_w &= (f_s'A^{-1}f_s + \Lambda_p^{-1})^{-1} \\ \mu_w &= \Lambda_w(\Lambda_p^{-1}Mz - w)\end{aligned}$$

The integration over v in Equation (B.1) can be performed, leaving

$$\begin{aligned}\gamma = \mu_w'\Lambda_w^{-1}\mu_w &- (\delta_\alpha - z)'\Lambda_\alpha^{-1}(\delta_\alpha - z) - f_\tau'A^{-1}f_\tau \\ &- z'M'\Lambda_p^{-1}Mz - z'\Lambda_{z,x}^{-1}z + 2\mu_{z,x}'\Lambda_{z,x}^{-1}z - \mu_{z,x}'\Lambda_{z,x}^{-1}\mu_{z,x}\end{aligned}$$

We can now complete the square in z, which gives

$$\begin{aligned}\gamma = &-(z-\tilde{\mu}_z^\alpha)'\tilde{\Lambda}_z^{\alpha^{-1}}(z-\tilde{\mu}_z^\alpha) + \tilde{\mu}_z^{\alpha'}\tilde{\Lambda}_z^{\alpha^{-1}}\tilde{\mu}_z^\alpha \\ &- f_\tau' A^{-1} f_\tau + w'\Lambda_w^{-1} w - \mu_{z,x}'\Lambda_{z,x}^{-1}\mu_{z,x} - \delta_\alpha'\Lambda_\alpha^{-1}\delta_\alpha\end{aligned}$$

where

$$\begin{aligned}\tilde{\Lambda}_z^\alpha &= (\Lambda_x^{-1} + M\Lambda_p^{-1}M + \Lambda_\alpha^{-1} - M'\Lambda_p^{-1}\Lambda_w\Lambda_p^{-1}M)^{-1} \\ &= (\tilde{\Lambda}_z^{-1} + \Lambda_\alpha^{-1})^{-1} \\ \tilde{\mu}_z^\alpha &= \tilde{\Lambda}_z^\alpha(\Lambda_{z,x}^{-1}\mu_{z,x} - M\Lambda_p^{-1}\Lambda_w w + \Lambda_\alpha^{-1}\delta_\alpha) \\ &= \tilde{\Lambda}_z^\alpha(\tilde{\Lambda}_{z,x}^{-1}\tilde{\mu}_{z,x} + \Lambda_\alpha^{-1}\delta_\alpha)\end{aligned}$$

The integral over z in Equation (B.1) can be performed, leaving

$$\gamma = \tilde{\mu}_z^{\alpha'}\tilde{\Lambda}_z^{\alpha^{-1}}\tilde{\mu}_z^\alpha - f_\tau' A^{-1} f_\tau + w'\Lambda_w^{-1} w - \mu_{z,x}'\Lambda_{z,x}^{-1}\mu_{z,x} - \delta_\alpha'\Lambda_\alpha^{-1}\delta_\alpha$$

Finally, we complete the square in δ_α, giving

$$\begin{aligned}\gamma = &-(\delta_\alpha - \tilde{\mu}_\alpha)'\tilde{\Lambda}_\alpha^{-1}(\delta_\alpha - \tilde{\mu}_\alpha) + \tilde{\mu}_\alpha'\tilde{\Lambda}_\alpha^{-1}\tilde{\mu}_\alpha \\ &- f_\tau' A^{-1} f_\tau + w'\Lambda_w^{-1} w - \mu_{z,x}'\Lambda_{z,x}^{-1}\mu_{z,x}\end{aligned}$$

where

$$\begin{aligned}\tilde{\Lambda}_\alpha &= (\Lambda_\alpha^{-1} - \Lambda_\alpha^{-1}\tilde{\Lambda}_z^\alpha\Lambda_\alpha^{-1})^{-1} \\ \tilde{\mu}_\alpha &= \tilde{\Lambda}_\alpha\Lambda_\alpha^{-1}\tilde{\Lambda}_z^\alpha\tilde{\Lambda}_{z,x}^{-1}\tilde{\mu}_{z,x}\end{aligned}$$

We can simplify these expressions using matrix inversion lemma [HS81a]:

$$(B^{-1} + CD^{-1}C')^{-1} = B - BC(D + C'BC)^{-1}C'B$$

and so

$$\begin{aligned}\tilde{\Lambda}_\alpha &= (\Lambda_\alpha^{-1} - \Lambda_\alpha^{-1}(\tilde{\Lambda}_{z,x}^{-1} + \Lambda_\alpha^{-1})^{-1}\Lambda_\alpha^{-1})^{-1} \\ &= \Lambda_\alpha + \tilde{\Lambda}_{z,x}\end{aligned}$$

and thus

$$\begin{aligned}\tilde{\mu}_\alpha &= (\Lambda_\alpha + \tilde{\Lambda}_{z,x})\Lambda_\alpha^{-1}\tilde{\Lambda}_{z,x}^\alpha\tilde{\Lambda}_{z,x}^{-1}\tilde{\mu}_{z,x} \\ &= (\Lambda_\alpha + \tilde{\Lambda}_{z,x})(\Lambda_\alpha + \tilde{\Lambda}_{z,x})^{-1}\tilde{\mu}_{z,x} \\ &= \tilde{\mu}_{z,x}\end{aligned}$$

Appendix C

Update Equations for Mixture Model

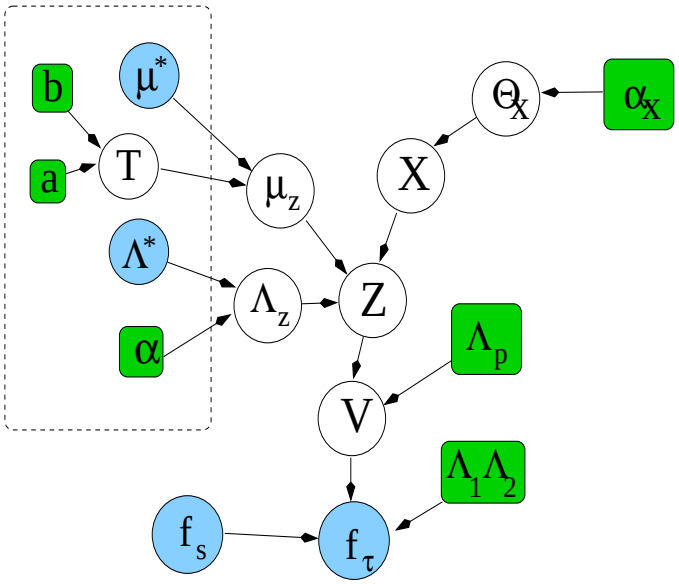


Figure C.1: Bayesian network for the mixture of Gaussians over optical flow fields with feature weighting. Shaded nodes are observed or fixed (known), while unshaded nodes are unknown random variables. Boxes are fixed hyper-parameters. The dashed line delineates the priors for feature weighting. $X \in 1 \ldots N_x$ are discrete motion classes, Z is the Zernike feature vector (projection of optical flow field), V is the optical flow field, f_s are the spatial derivatives, and f_t is the temporal derivative. μ_z, Λ_z are the parameters of the mixture of Gaussians over the Z vector space, and T are the feature weights. Θ_X are the class probability parameter (a multinomial), and α_X is the parameter of the (conjugate) Dirichlet prior over Θ_X.

Figure C.1 shows the simple mixture model with feature weighted Gaussian distributions that was discussed in Section 3.2.2. Here we derive the update equations for the parameter learning using the expectation maximization (EM) algorithm. Our goal is to show how Equations 4.6 – 4.9 are derived.

To update the output mean, we set the derivative with respect to the mean for state $X = x_i$, $\mu_{z,i}$, to zero.

$$\left[\sum_{\mathbf{X}} \int_{\mathbf{Z}} P(\mathbf{XZ}|\nabla\mathbf{f}, \theta') \frac{\partial}{\partial \mu_{z,i}} \log P(\nabla\mathbf{f}\mathbf{XZ}|\Theta)\right] + \frac{\partial}{\partial \mu_{z,i}} \log P(\mu_{z,i}) = 0$$

where $P(\mu_{z,i}) \sim \mathcal{N}(\mu^*, T)$ is the prior distribution over the mean for state i, such that

$$\frac{\partial}{\partial \mu_{z,i}} \log P(\mu_{z,i}) = -T^{-1}(\mu^* + \mu_{z,i})$$

The log of the complete posterior is a sum of logarithmic terms, and only those involving μ_z are non-zero, such that:

$$\frac{\partial}{\partial \mu_{z,i}} \log P(\nabla\mathbf{f}\mathbf{XZ}|\Theta) = \sum_{k=1}^{N_t} \frac{\partial}{\partial \mu_{z,i}} \log[P(Z_k|X_k, \Theta)]$$

The sum over k can be taken outside the expression, giving

$$\sum_{k=1}^{N_t} \sum_{\mathbf{X}} \int_{\mathbf{Z}} P(\mathbf{XZ}|\nabla\mathbf{f}, \theta') \frac{\partial}{\partial \mu_{z,i}} \log[P(Z_k|X_k, \Theta)] = T^{-1}(\mu^* + \mu_{z,i})$$

The derivative will pick out the i^{th} value from the sum over X_k,

$$\sum_{k=1}^{N_t} \sum_{X_1} \cdots \sum_{X_{k-1}} \sum_{X_{k+1}} \cdots \sum_{X_{N_t}} \int_{\mathbf{Z}} P(\mathbf{XZ}|\nabla\mathbf{f}, \theta') \frac{\partial}{\partial \mu_{z,i}} \log[P(Z_k|X_{k,i}, \Theta)] = T^{-1}(\mu^* - \mu_{z,i})$$

the sums over X_i and the integrations over Z_i for $i = 1 \ldots k-1, k+1 \ldots N_t$ can be performed, giving unity, and leaving

$$\sum_{k=1}^{N_t} \int_{z_k} P(X_{k,i} z_k|\nabla\mathbf{f}, \Theta') \frac{\partial}{\partial \mu_{z,i}} \log[P(z_k|X_{k,i}, \Theta)] = T^{-1}(\mu^* - \mu_{z,i}). \tag{C.1}$$

Since $P(z_k|X_{k,i}, \Theta) \sim \mathcal{N}(\mu_{z,i}, \Lambda_{z,i})$, the derivative gives $\Lambda_{z,i}^{-1}(z_k - \mu_{z,i})$. Further, since the measurements are independent in the mixture model, $P(X_{k,i} z_k|\nabla\mathbf{f}\Theta') = P(z_k|X_{k,i}\nabla f_k\Theta')P(X_{k,i}|\nabla f_k\Theta')$, we have

$$\left[\sum_{k=1}^{N_t} P(X_{k,i}|\nabla f_k\Theta')\Lambda_{z,i}^{-1} + T^{-1}\right] \mu_{z,i} = \Lambda_{z,i}^{-1} \sum_{k=1}^{N_t} \int_{z_k} z_k P(z_k|X_{k,i}\nabla f_k\Theta')P(X_{k,i}|\nabla f_k\Theta') + T^{-1}\mu^*.$$

and we can solve for $\mu_{z,i}$

$$\mu_{z,i} = (\xi_{\cdot,i}\Lambda_{z,i}^{-1} + T^{-1})^{-1}\left[\Lambda_{z,i}^{-1}\left(\sum_{k=1}^{N_t}\tilde{\mu}_{z,x}\xi_{k,i}\right) + T^{-1}\mu^*\right]$$

where $\xi_{k,i} = P(X_{k,i}|\nabla f_k\Theta')$, $\xi_{\cdot,i} = \sum_{k=1}^{N_t}\xi_{k,i}$ and $\tilde{\mu}_{z,x}$ is given by Equation (3.25). Thus, the most likely mean for each state x is the weighted sum of the most likely values of z as given by Equation (3.25). Dimensions of the means, $\mu_{z,i}$, with small feature weights, τ_k^2, will be biased toward the data mean, μ^*, in that dimension. This is reasonable, because such dimensions are not relevant for clustering, and so should be the same for any cluster, X.

The updates to the feature weights, τ_k^2, are found by taking the derivative with respect to τ^2. However, the complete data likelihood, $P(\nabla\mathbf{fXZ}|\Theta)$, is independent of the feature weights, so we are solving

$$\frac{\partial}{\partial\tau_k^2}\log[\prod_{j=1}^{N_z}P(\tau_j^2)\prod_{i=1}^{N_x}P(\mu_{z,i})] = \frac{\partial}{\partial\tau_k^2}[\sum_{j=1}^{N_z}\log P(\tau_j^2) + \sum_{i=1}^{N_x}\log P(\mu_{z,i})] = 0$$

where $P(\tau_j^2)$ are the prior distributions over the feature weights along each dimension of the output space, given by Equation (3.26), and $P(\mu_{z,i}) \sim \mathcal{N}(\mu^*, T)$, so that

$$\frac{\partial}{\partial\tau_k^2}\log P(\tau_k^2) = \frac{\partial}{\partial\tau_k^2}\left[-(a+1)\log(\tau_k^2) - \frac{b}{\tau_k^2}\right] = -\frac{a+1}{\tau_k^2} + \frac{b}{(\tau_k^2)^2}$$

and

$$\begin{aligned}\frac{\partial}{\partial\tau_k^2}\log P(\mu_{z,i}) &= -\frac{1}{2}\left[\frac{\partial}{\partial\tau_k^2}\log(|T|) + \frac{\partial}{\partial\tau_k^2}\sum_{k=1}^{N_z}\frac{(\mu_{z,i,k}-\mu_k^*)^2}{\tau_k^2}\right] \\ &= -\frac{1}{2}\frac{1}{\tau_k^2} + \frac{1}{2}\frac{(\mu_{z,i,k}-\mu_k^*)^2}{(\tau_k^2)^2}\end{aligned}$$

where $\mu_{z,i,k}$, μ_k^* are the k^{th} dimensions of $\mu_{z,i}$ and μ^*, respectively. Thus, we have

$$-\frac{a+1}{\tau_k^2} + \frac{b}{(\tau_k^2)^2} - \frac{1}{2}\sum_{k-1}^{N_z}\frac{1}{\tau_k^2} + \frac{1}{2}\sum_{k-1}^{N_z}\frac{(\mu_{z,i,k}-\mu_k^*)^2}{(\tau_k^2)^2} = 0$$

which is

$$\frac{b}{(\tau_k^2)^2} + \frac{1}{2}\sum_{k-1}^{N_z}\frac{(\mu_{z,i,k}-\mu_k^*)^2}{(\tau_k^2)^2} = \frac{a+1}{\tau_k^2} + \frac{N_z}{2\tau_k^2}$$

Solving for τ_k^2 yeilds

$$\tau_k^2 = \frac{b}{a+N_x/2+1} + \frac{1}{2a+N_x+2}\sum_{i=1}^{N_x}(\mu_{z,i,k}-\mu_k^*)^2$$

The updates to the feature weights show that those dimensions, k, with $\mu_{z,i,k}$ very different from the data mean, μ_k^*, across all states, will receive large values of τ_k^2, while those with $\mu_{z,i,k} \sim \mu_k^*$ will receive small values of τ_k^2. Intuitively, the dimensions along which the data is well separated (large inter-class distance) will be weighted more.

The updates to the covariance matrix, $\Lambda_{k,i}$, are found in a similar way to the mean. We present the main points here. We set the derivative with respect to the covariance for state $X = x_i$, $\Lambda_{z,i}$, to zero.

$$\left[\sum_{\mathbf{X}} \int_{\mathbf{Z}} P(\mathbf{XZ}|\nabla\mathbf{f}, \theta') \frac{\partial}{\partial \Lambda_{z,i}} \log P(\nabla\mathbf{f}\mathbf{XZ}|\Theta)\right] + \frac{\partial}{\partial \Lambda_{z,i}} \log P(\Lambda_{z,i}) = 0$$

where $P(\Lambda_{z,i})$ is the prior distribution over the covariance for state i, given by Equation (3.27), so that

$$\begin{aligned}\frac{\partial}{\partial \Lambda_{z,i}} \log P(\Lambda_{z,i}) &= \frac{\partial}{\partial \Lambda_{z,i}} \left[-\frac{\alpha + N_z + 1}{2} \log |\Lambda_{z,x}| - \frac{1}{2} tr(\alpha \Lambda^* \Lambda_{z,x}^{-1})\right] \\ &= -\frac{\alpha + N_z + 1}{2} \Lambda_{z,x}^{-1} + \frac{1}{2} \Lambda_{z,x}^{-1} \alpha \Lambda^* \Lambda_{z,x}^{-1} \end{aligned} \tag{C.2}$$

Following the same derivation as that leading up to Equation (C.1), we obtain

$$\sum_{k=1}^{N_t} \int_{z_k} P(X_{k,i} z_k | \nabla f_k \Theta') \frac{\partial}{\partial \Lambda_{z,i}} \log[P(z_k | X_{k,i}, \Theta)] = \frac{\alpha + N_z + 1}{2} \Lambda_{z,x}^{-1} - \frac{1}{2} \Lambda_{z,x}^{-1} \alpha \Lambda^* \Lambda_{z,x}^{-1}. \tag{C.3}$$

The derivative on the left gives

$$\frac{\partial}{\partial \Lambda_{z,i}} \log P(z_k | X_{k,i}, \Theta) = -\frac{1}{2} \Lambda_{z,i}^{-1} + \frac{1}{2} \Lambda_{z,i}^{-1} (z_k - \mu_{z,i})(z_k - \mu_{z,i})' \Lambda_{z,i}^{-1}$$

multiplying Equation (C.3) on the left and right by $\Lambda_{z,i}$ gives

$$\sum_{k=1}^{N_t} \int_{z_k} P(X_{k,i} z_k | \nabla f_k \Theta') [(z_k - \mu_{z,i})(z_k - \mu_{z,i})' - \Lambda_{z,i}] = (\alpha + N_z + 1)\Lambda_{z,i} - \alpha \Lambda^*$$

which is

$$\sum_{k=1}^{N_t} \int_{z_k} P(X_{k,i} z_k | \nabla f_k \Theta')(z_k - \mu_{z,i})(z_k - \mu_{z,i})' + \alpha \Lambda^* = \sum_{k=1}^{N_t} \int_{z_k} P(X_{k,i} z_k | \nabla f_k \Theta') \Lambda_{z,i} + (\alpha + N_z + 1)\Lambda_{z,i}.$$

Since

$$P(X_{k,i} z_k | \nabla f_k \Theta') = P(z_k | X_{k,i} \nabla f_k \Theta') P(X_{k,i} | \nabla f_k \Theta')$$

and

$$\int_{z_k} P(z_k | X_{k,i} \nabla f_k \Theta') = 1$$

we have

$$\sum_{k=1}^{N_t} P(X_{k,i}|\nabla f_k \Theta') E_{P(z_k|X_{k,i}\nabla f_k\Theta')}[(z_k - \mu_{z,i})(z_k - \mu_{z,i})'] + \alpha\Lambda^* = [\xi_{\cdot,i} + (\alpha + N_z + 1)]\Lambda_{z,i}$$

so that we can solve for $\Lambda_{z,i}$ as

$$\Lambda_{z,i} = \frac{1}{\xi_{\cdot,i} + \alpha + N_z + 1}\left[\sum_{k=1}^{N_t} P(X_{k,i}|\nabla f_k \Theta') E_{P(z_k|X_{k,i}\nabla f_k\Theta')}[(z_k - \mu_{z,i})(z_k - \mu_{z,i})'] + \alpha\Lambda^*\right]$$

We must now evaluate the expected covariance $\langle (z_k - \mu_{z,i})(z_k - \mu_{z,i})' \rangle$, where the expectation is taken with respect to the distribution over z given the state, i, and the data, ∇f, $P(z_k|X_{k,i}\nabla f_k\Theta')$. Expanding,

$$\begin{aligned}
\langle (z_k - \mu_{z,i})(z_k - \mu_{z,i})' \rangle &= \langle z_k z_k' - 2\mu_{z,i} z_k' + \mu_{z,i}\mu_{z,i}' \rangle \\
&= \langle z_k z_k' \rangle - 2\mu_{z,i}\langle z_k' \rangle + \mu_{z,i}\mu_{z,i}' \\
&= \langle z_k z_k' \rangle - \mu_{z,i}\tilde{\mu}_{z,i}' - \tilde{\mu}_{z,i}\mu_{z,i}' + \mu_{z,i}\mu_{z,i}'
\end{aligned}$$

where we have used Equation (3.22). Since the expected covariance over z is

$$\begin{aligned}
\tilde{\Lambda}_{z,i} &= \langle (z - \tilde{\mu}_{z,i})(z - \tilde{\mu}_{z,i})' \rangle \\
&= \langle zz' - 2\tilde{\mu}_{z,i} z' + \tilde{\mu}_{z,i}\tilde{\mu}_{z,i}' \rangle \\
&= \langle zz' \rangle - \tilde{\mu}_{z,i}\tilde{\mu}_{z,i}'
\end{aligned}$$

we can write

$$\langle (z_k - \mu_{z,i})(z_k - \mu_{z,i})' \rangle = \tilde{\Lambda}_{z,i} + \tilde{\mu}_{z,i}\tilde{\mu}_{z,i}' - \mu_{z,i}\tilde{\mu}_{z,i}' - \tilde{\mu}_{z,i}\mu_{z,i}' + \mu_{z,i}\mu_{z,i}'$$

So that the updates to the covariance can be written as

$$\Lambda_{z,i} = \frac{\sum_{k=1}^{N_t}(\tilde{\Lambda}_{z,x} + \tilde{\mu}_{z,i}\tilde{\mu}_{z,i}' - \mu_{z,i}\tilde{\mu}_{z,i}' - \tilde{\mu}_{z,i}\mu_{z,i}')\xi_{k,i} + \mu_{z,i}\mu_{z,i}'\xi_{\cdot,i} + \alpha\Lambda^*}{\xi_{\cdot,i} + \alpha + N_m + 1}.$$

which is

$$\Lambda_{z,i} = \frac{\sum_{k=1}^{N_t}[\tilde{\Lambda}_{z,x} + (\tilde{\mu}_{z,i} - \mu_{z,i})(\tilde{\mu}_{z,i} - \mu_{z,i})']\xi_{k,i} + \alpha\Lambda^*}{\xi_{\cdot,i} + \alpha + N_z + 1}.$$

The covariance is updated based on a combination of the most likely covariance, $\tilde{\Lambda_{z,i}}$ and the covariance of the most likely mean, $\tilde{\mu}_{z,i}$, about the model mean $\mu_{z,i}$, for each data point, $k = 1 \ldots N_t$, weighted by the probability of state i given the data. The prior covariance is represented with $\alpha\Lambda^*$, which stabilizes the updates, avoiding matrix singularity problems in the inverses in Equation (3.21).

The updates to the class probability, Θ_x, are given by taking the derivative as usual, except that we must now enforce the contraint that Θ_x is a proper probability distribution, $\sum_{i=1}^{N_x} \Theta_{x,i} = 1$. We do this using a Lagrange multiplier, so we want to solve

$$\left[\sum_{\mathbf{X}} \int_{\mathbf{Z}} P(\mathbf{XZ}|\nabla\mathbf{f}, \theta') \frac{\partial}{\partial \Theta_{x,i}} \log P(\nabla\mathbf{f}\mathbf{XZ}|\Theta)\right] - \lambda \frac{\partial}{\partial \Theta_{x,i}} (\sum_{i=1}^{N_x} \Theta_{x,i} - 1) + \frac{\partial}{\partial \Theta_{x,i}} \log P(\Theta_x) = 0$$

The prior over Θ_x is distributed according to a Dirichlet distribution, $P(\Theta_x) = \prod_{i=1}^{N_x} \Theta_{x,i}^{\alpha_{x,i}}$. Taking the derivatives, and performing the integration over z_k, this is

$$\sum_{k=1}^{N_t} P(X_{k,i}|\nabla f_k, \theta') \frac{1}{\Theta_{x,i}} + \frac{\alpha_{x,i}}{\Theta_{x,i}} = \lambda \tag{C.4}$$

Multiplying by $\Theta_{x,i}$ and summing over i gives

$$\sum_{i=1}^{N_x} \left[\sum_{k=1}^{N_t} P(X_{k,i}|\nabla f_k, \theta') + \alpha_{x,i}\right] = \lambda \sum_{i=1}^{N_x} \Theta_{x,i} = \lambda$$

We substitute this value of λ back into Equation (C.4) and solve for $\Theta_{x,i}$ to obtain

$$\Theta_{x,i} = \frac{\alpha_{x,i} + \xi_{\cdot,i}}{\sum_{i=1}^{N_x} (\alpha_{x,i} + \xi_{\cdot,i})}.$$

Appendix D

Estimating C4MG Model Parameters

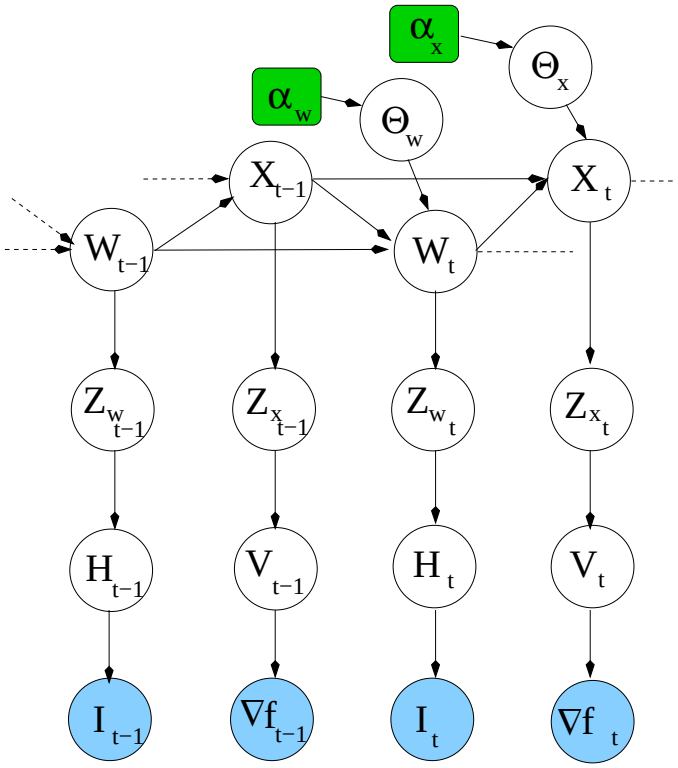


Figure D.1: Two time slices of a dynamic Bayesian network (DBN) for simultaneous modeling of pose and dynamics. Repeat of Figure 3.12.

This appendix derives the parameter learning equations for the context-

dependent Markov chain of mixtures of hidden Markov models (C4MG) described in Section 4.7. However, we first start by deriving the update equations for the coupled hidden Markov model for temporal modeling of pose and dynamics, as discussed in Section 4.4. The Bayesian network for this model is shown in Figure D.1.

We use the notation that a set of variables $X_t \dots X_T$ is written $\{\mathbf{X}\}_{t,T}$ To simplify things further, we denote $X_t \dots X_T, W_t \dots W_T$ as $\{\mathbf{XW}\}_{t,T}$. If no subscript is given, it is assumed to be $1, T$, the full variable set. Finally, we denote the observations $\mathbf{O} \equiv \mathbf{I}\nabla\mathbf{f}$, and the hidden state, $\mathbf{Y} \equiv \mathbf{XW}$ We assume that every term is conditioned on the model parameters, θ'. The update equation for X transition probability, Θ_{Xijk} is derived in a similar way to the class probability in the simple mixture model. Due to the Markovian dependence in the dynamics and configuration chains, we must take derivatives of a more complex expression, subject to the constraint that $\sum_i \Theta_{Xijk} = 1$

$$[\sum_{\mathbf{X},\mathbf{W}} \int_{\mathbf{Z}} \sum_{\mathbf{H}} P(\mathbf{XWZH}|\nabla\mathbf{f}, \mathbf{I}, \theta') \frac{\partial}{\partial \Theta_{Xijk}} \log P(\nabla\mathbf{fXWZH}|\Theta)] - \lambda \frac{\partial}{\partial \mu_{z,i}} (\sum_i \Theta_{Xijk} - 1) + \frac{\partial}{\partial \Theta_{Xijk}} \log P(\Theta_{Xijk}) = 0 \tag{D.1}$$

The prior over Θ_{Xijk} is distributed according to a Dirichlet distribution, $P(\Theta_{Xijk}) = \prod_{ijk} \Theta_{Xijk}^{\alpha_{Xijk}}$, and so

$$\frac{\partial}{\partial \Theta_{Xijk}} \log P(\Theta_{Xijk}) = \alpha_{Xijk} / \Theta_{Xijk}.$$

The complete data posterior can be computed as

$$\begin{aligned} &P(\nabla\mathbf{fXWZH}|\Theta) = \\ &\prod_{k=2}^{N_t} [P(\nabla f_k|z_k) P(z_k|x_k) P(I_k|x_k) P(h_k|c_k) \\ &\qquad P(x_k|c_k x_{k-1}) P(c_k|x_{k-1} c_{k-1}) P(x_1|c_1) P(c_1)] \end{aligned} \tag{D.2}$$

Therefore,

$$\frac{\partial}{\partial \Theta_{Xijk}} \log P(\nabla\mathbf{fXWZH}|\Theta) = \sum_{t=1}^{N_t} \frac{\partial}{\partial \Theta_{Xijk}} \log P(x_t|c_t x_{t-1})$$

The derivative picks out the values of $x_t = i, c_t = j$ and $x_{t-1} = k$, so that the integrations over $\mathbf{H}$ and $\mathbf{Z}$, as well as the sums over X_i and W_i for $i = 1 \dots t-2, t+$

$1 \ldots N_t$ can all be performed in Equation (D.1), leaving

$$\begin{aligned}\lambda &= \sum_{t=1}^{N_t} P(X_{t,i}W_{t,j}X_{t-1,k}|\mathbf{O}\theta') \frac{\partial}{\partial \Theta_{Xijk}} \log \Theta_{Xijk} + \frac{\alpha_{Xijk}}{\Theta_{Xijk}} \\ &= \sum_{t=1}^{N_t} P(X_{t,i}W_{t,j}X_{t-1,k}|\mathbf{O}\theta') \frac{1}{\Theta_{Xijk}} + \frac{\alpha_{Xijk}}{\Theta_{Xijk}}\end{aligned}$$

Multiplying by Θ_{Xijk} and summing over i, gives

$$\sum_{i=1}^{N_x} \sum_{t=1}^{N_t} P(X_{t,i}W_{t,j}X_{t-1,k}|\mathbf{O}\theta') + \alpha_{Xijk} = \lambda \sum_{i=1}^{N_x} \Theta_{Xijk} = \lambda$$

so we can solve for Θ_{Xijk} as

$$\Theta_{Xijk} = \frac{\alpha_{Xijk} + \sum_{t=1}^{N_t} P(X_{t,i}W_{t,j}X_{t-1,k}|\mathbf{O}\theta')}{\sum_i \left[\alpha_{Xijk} + \sum_{t=1}^{N_t} P(X_{t,i}W_{t,j}X_{t-1,k}|\mathbf{O}\theta')\right]} \tag{D.3}$$

The update equation for W transition probability, Θ_{Wijk}, and for the initial state probabilities, Π_{Xij} and Π_{Wi}, are derived in a similar way, giving

$$\Theta_{Wijk} = \frac{\alpha_{Wijk} + \sum_{t=1}^{N_t} P(W_{t,i}X_{t-1,j}W_{t-1,k}|\mathbf{O}\theta')}{\sum_i \left[\alpha_{Wijk} + \sum_{t=1}^{N_t} P(W_{t,i}X_{t-1,j}W_{t-1,k}|\mathbf{O}\theta')\right]} \tag{D.4}$$

$$\Pi_{Xij} = \frac{\alpha_{Xij} + \sum_{t=1}^{N_t} P(X_{1,i}W_{1,j}|\mathbf{O}\theta')}{\sum_i \left[\alpha_{Xij} + \sum_{t=1}^{N_t} P(X_{1,i}W_{1,j}|\mathbf{O}\theta')\right]} \tag{D.5}$$

$$\Pi_{Wi} = \frac{\alpha_{Wi} + \sum_{t=1}^{N_t} P(W_{1,i}|\mathbf{O}\theta')}{\sum_i \left[\alpha_{Wi} + \sum_{t=1}^{N_t} P(W_{1,i}|\mathbf{O}\theta')\right]} \tag{D.6}$$

The update equations for the parameters output distributions in the dynamics chain, $\mu_{z,i}, \Lambda_{z,i}$, are almost the same as those for the simple mixture model derived in Appendix C.

$$\left[\sum_{\mathbf{X},\mathbf{W}} \int_{\mathbf{Z}} \sum_{\mathbf{H}} P(\mathbf{XWZH}|\nabla\mathbf{f}, \mathbf{I}, \theta') \frac{\partial}{\partial \mu_{z,i}} \log P(\nabla\mathbf{fXWZH}|\Theta)\right] + \frac{\partial}{\partial \mu_{z,i}} \log P(\mu_{z,i}) = 0$$

The derivative picks out only the terms involving μ_z, and all the sums over $\mathbf{W}$ and $\mathbf{H}$ can be performed, leaving the equivalent of Equation (C.1):

$$\sum_{k=1}^{N_t} \int_{z_k} P(X_{k,i}z_k|\mathbf{O}) \frac{\partial}{\partial \mu_{z,i}} \log[P(z_k|X_{k,i}, \Theta)] = T^{-1}(\mu^* - \mu_{z,i}). \tag{D.7}$$

Now, however, the observations are not all independent, and we have that

$$P(X_{k,i}z_k|\mathbf{O}) = P(z_k|X_{k,i}\mathbf{O})P(X_{k,i}|\mathbf{O}).$$

The probability of a given state of the dynamics process at time t, $X_{t,i}$, given all the data, $\mathbf{O}$, can be computed as follows

$$\begin{aligned}
P(X_{t,i}|\underset{1,T}{\{\mathbf{O}\}}) &= \sum_j P(X_{t,i}W_{t,j}\underset{1,T}{\{\mathbf{O}\}})/P(\underset{1,T}{\{\mathbf{O}\}}) \\
&= \sum_j P(\underset{t+1,T}{\{\mathbf{O}\}}|X_{t,i}W_{t,j}\underset{1,t}{\{\mathbf{O}\}})P(X_{t,i}W_{t,j}\underset{1,t}{\{\mathbf{O}\}})/P(\underset{1,T}{\{\mathbf{O}\}}) \\
&= \sum_j P(\underset{t+1,T}{\{\mathbf{O}\}}|X_{t,i}W_{t,j})P(X_{t,i}W_{t,j}\underset{1,t}{\{\mathbf{O}\}})/P(\underset{1,T}{\{\mathbf{O}\}}) \\
&= \frac{\sum_j \beta^x_{t,ij}\kappa^x_{t,ij}}{\sum_{ij} \beta^x_{t,ij}\kappa^x_{t,ij}}
\end{aligned}$$

where $\kappa^x_{t,ij} = P(X_{t,i}W_{t,j}\underset{1,t}{\{\mathbf{O}\}})$ and $\beta^x_{t,ij} = P(\underset{t+1,T}{\{\mathbf{O}\}}|X_{t,i}W_{t,j})$ are forwards and backwards variable for the dynamics process. We show recursive derivations for these parameters later in this Appendix [1]. Returning to Equation (D.7), we see that the update equations derived for the simple mixture model (Equations (4.6), (4.7), and (4.8)), can be used for the Markovian model if we use $\xi_{k,i} = P(X_{t,i}|\underset{1,T}{\{\mathbf{O}\}})$ as derived above.

Update equations for emission probability over the exemplars, H, given the configuration class, W, Θ_{Hij}, can be similarly derived leading to

$$\Theta_{Hij} = \frac{\delta_{Hij} + \sum_{k=1}^{N_t} P(H_{k,i}W_{k,j}|\mathbf{O})}{\sum_i \left[\delta_{Hij} + \sum_{k=1}^{N_t} P(H_{k,i}W_{k,j}|\mathbf{O})\right]}$$

where δ_{Hij} is the prior distribution, which is Dirichlet $Dir(\Theta_{Hij}, \delta_{Hij})$. We can expand

$$P(H_{k,i}W_{k,j}|\mathbf{O}) = P(H_{k,i}|W_{k,j}\mathbf{O})P(W_{k,j}|\mathbf{O})$$

And since

$$P(H_{k,i}|W_{k,j}\mathbf{O}) = P(H_{k,i}|W_{k,j}I_k) = \frac{P(I_k|H_{k,i})\Theta_{H,ij}}{\sum_i P(I_k|H_{k,i})\Theta_{H,ij}}$$

[1] These quantities are the equivalents of the usual "forwards" and "backwards" variables, α and β, in hidden Markov model parameter and likelihood estimation [Rab89]. We are using κ^x here since α is our scale variable.

and the probability of a given state of the configuration process at time t, $W_{t,i}$, is

$$\begin{aligned}
P(W_{t,i}|\underset{1,T}{\{\mathbf{O}\}}) &= \sum_j P(W_{t,i}X_{t-1,j}\underset{1,T}{\{\mathbf{O}\}})/P(\underset{1,T}{\{\mathbf{O}\}}) \\
&= \sum_j P(\nabla f_t \underset{t+1,T}{\{\mathbf{O}\}}|W_{t,i}X_{t-1,j}I_t\underset{1,t-1}{\{\mathbf{O}\}})P(W_{t,i}X_{t-1,j}I_t\underset{1,t-1}{\{\mathbf{O}\}})/P(\underset{1,T}{\{\mathbf{O}\}}) \\
&= \sum_j P(\nabla f_t \underset{t+1,T}{\{\mathbf{O}\}}|W_{t,i}X_{t-1,j})P(W_{t,i}X_{t-1,j}I_t\underset{1,t-1}{\{\mathbf{O}\}})/P(\underset{1,T}{\{\mathbf{O}\}}) \\
&= \frac{\sum_j \beta^w_{t,ij}\kappa^w_{t,ij}}{\sum_{ij}\beta^w_{t,ij}\kappa^w_{t,ij}}
\end{aligned}$$

we obtain

$$P(H_{k,i}W_{k,j}|\mathbf{O}) = \frac{P(I_k|H_{k,i})\Theta_{H,ij}}{\sum_i P(I_k|H_{k,i})\Theta_{H,ij}}\frac{\sum_j \beta^w_{t,ij}\kappa^w_{t,ij}}{\sum_{ij}\beta^w_{t,ij}\kappa^w_{t,ij}}$$

where $\kappa^w_{tjk} = P(X_{t,i}W_{t,j}X_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\}}I_t)$ and $\beta^w_{ij} = P(\nabla f_t \underset{t+1,T}{\{\mathbf{O}\}}|W_{t,i}X_{t-1,j})$ are the forwards and backwards variables for the configuration process.
The expected number of transitions in the dynamics chain is

$$\begin{aligned}
&\sum_t P(X_{t,i}W_{t,j}X_{t-1,k}|\mathbf{O}) \\
&\quad= \sum_t P(X_{t,i}W_{t,j}X_{t-1,k}\underset{1,T}{\{\mathbf{O}\}})/P(\mathbf{O}) \\
&\quad= \sum_t P(\underset{t+1,T}{\{\mathbf{O}\}}\nabla f_t|X_{t,i}W_{t,j}X_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\}}I_t)P(X_{t,i}W_{t,j}X_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\}}I_t)/P(\mathbf{O}) \\
&\quad= \sum_t P(\underset{t+1,T}{\{\mathbf{O}\}}|\nabla f_t X_{t,i}W_{t,j})P(\nabla f_t|X_{t,i}W_{t,j})P(X_{t,i}|W_{t,j}X_{t-1,k}) \\
&\qquad P(\underset{t+1,T}{\{\mathbf{O}\}}|\nabla f_t X_{t,i}W_{t,j})P(W_{t,j}X_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\}}I_t)/P(\mathbf{O}) \\
&\quad\propto \sum_t P(\underset{t+1,T}{\{\mathbf{O}\}}|X_{t,i}W_{t,j})P(\nabla f_t|X_{t,i})\Theta_{X,ijk}P(W_{t,j}X_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\}}I_t)/P(\mathbf{O}) \\
&\quad= \sum_t \beta^x_{tij}P(\nabla f_t|X_{t,i})\Theta_{X,ijk}\kappa^w_{jk}/P(\mathbf{O})
\end{aligned}$$

where $P(\mathbf{O}) = \sum_{ijk}\beta^x_{tij}P(\nabla f_t|X_{t,i})\Theta_{X,ijk}\kappa^w_{jk}$ and the likelihood of the image derivates, $P(\nabla f|X_{t,i})$, is calculated from Equation (3.19).

The expected number of transitions in the configuration chain is

$$\begin{aligned}
&\sum_t P(W_{t,i}X_{t-1,j}W_{t-1,k}|\mathbf{O}) \\
&\quad = \sum_t P(W_{t,i}X_{t-1,j}W_{t-1,k}\underset{1,T}{\{\mathbf{O}\}}/P(\mathbf{O}) \\
&\quad = \sum_t P(\underset{t,T}{\{\mathbf{O}\}}|W_{t,i}X_{t-1,j}W_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\}})P(W_{t,i}X_{t-1,j}W_{t-1,k}\underset{1,t-1}{\{\}}\underset{1,t-1}{\{\mathbf{O}\}})/P(\mathbf{O}) \\
&\quad = \sum_t P(\nabla f_t I_t \underset{t+1,T}{\{\mathbf{O}\}}|W_{t,i}X_{t-1,j})P(W_{t,i}|X_{t-1,j}W_{t-1,k})P(X_{t-1,j}W_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\}})/P(\mathbf{O}) \\
&\quad = \sum_t P(\nabla f_t \underset{t+1,T}{\{\mathbf{O}\}}|W_{t,i}X_{t-1,j})P(I_t|W_{t,i}) \\
&\qquad \Theta_{Wijk}P(X_{t-1,j}W_{t-1,k}\underset{1,t-1}{\{\{\mathbf{O}\}\}})/P(\mathbf{O}) \\
&\quad = \sum_t \beta^w_{ij}P(I_t|W_{t,i})\Theta_{Wijk}\kappa^x_{t-1,jk}/P(\mathbf{O})
\end{aligned}$$

The likelihood of the image, $P(I_t|W_{t,k})$ is calculated from the configuration mixture model.
The expectation of the number of inital states in the dynamics chain is

$$\begin{aligned}
P(X_{1,i}W_{1,j}|\mathbf{O}) &= P(X_{1,i}W_{1,j}|\mathbf{O}) \\
&= \frac{P(X_{1,i}W_{1,j}\mathbf{O})}{P(\mathbf{O})} \\
&= \frac{\kappa^x_{1,ij}\beta^x_{1,ij}}{\sum_{ij}\kappa^x_{1,ij}\beta^x_{1,ij}}
\end{aligned}$$

while the expectation of the number of initial states in the configuration chain is

$$\begin{aligned}
\sum_j P(X_{1,j}W_{1,i}|\mathbf{O}) &= \sum_j P(X_{1,j}W_{1,i}|\mathbf{O}) \\
&= \sum_j \frac{\kappa^x_{1,ij}\beta^x_{1,ij}}{\sum_{ij}\kappa^x_{1,ij}\beta^x_{1,ij}}
\end{aligned}$$

The forwards variable for the configuration process is recursively computed as fol-

lows

$$\begin{aligned}
\kappa^w_{t,ij} &= P(W_{t,i}X_{t-1,j}\underset{1,t-1}{\{\mathbf{O}\}}I_t) \\
&= P(I_t|W_{t,i})P(W_{t,i}X_{t-1,j}\underset{1,t-1}{\{\mathbf{O}\}}) \\
&= P(I_t|W_{t,i})\sum_k P(W_{t,i}X_{t-1,j}W_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\}}) \\
&= P(I_t|W_{t,i})\sum_k P(W_{t,i}|X_{t-1,j}W_{t-1,k})P(X_{t-1,j}W_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\}}) \\
&= P(I_t|W_{t,i})\sum_k \Theta_{W,ijk}P(X_{t-1,j}W_{t-1,k}) \\
&= P(I_t|W_{t,i})\sum_k \Theta_{W,ijk}\kappa^x_{t-1,jk}
\end{aligned}$$

The backwards variable for the configuration process is recursively computed as follows

$$\begin{aligned}
\beta^w_{t,ij} &= P(\nabla f_t\underset{t+1,T}{\{\mathbf{O}\}}|W_{t,i}X_{t-1,j}) \\
&= \sum_k P(\nabla f_t|X_{t,k}W_{t,i}X_{t-1,j}\underset{t+1,T}{\{\mathbf{O}\}})P(X_{t,k}\underset{t+1,T}{\{\mathbf{O}\}})|W_{t,i}X_{t-1,j}) \\
&= \sum_k P(\nabla f_t|X_{t,k})P(\underset{t+1,T}{\{\mathbf{O}\}})|X_{t,k}W_{t,i}X_{t-1,j})P(X_{t,k}|W_{t,i}X_{t-1,j}) \\
&= \sum_k P(\nabla f_t|X_{t,k})\Theta_{X,kij}\beta^x_{t,ki}
\end{aligned}$$

The forwards variable for the dynamics process is recursively computed as follows

$$\begin{aligned}
\kappa^x_{t,ij} &= P(X_{t,i}W_{t,j}\underset{1,t}{\{\mathbf{O}\alpha\}}) \\
&= \sum_k P(X_{t,i}W_{t,j}X_{t-1,k}\underset{1,t}{\{\mathbf{O}\alpha\}}) \\
&= \sum_k P(\nabla f_t|X_{t,i}W_{t,j}X_{t-1,k})P(X_{t,i}W_{t,j}X_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\alpha\}}I_t) \\
&= P(\nabla f_t|X_{t,i})\sum_k P(X_{t,i}|W_{t,j}X_{t-1,k})P(W_{t,j}X_{t-1,k}\underset{1,t-1}{\{\mathbf{O}\alpha\}}I_t) \\
&= P(\nabla f_t|X_{t,i})\sum_k \Theta_{X,ijk}\kappa^w_{t,jk}
\end{aligned}$$

The initial value of the forwards dynamics variable is

$$\begin{aligned}
\kappa^x_{1,ij} &= P(X_{1,i}W_{1,j}\nabla f_1 I_1 b_1) \\
&= P(\nabla f_1|X_{1,i})P(I_1|W_{1,j})P(b_1|W_{1,j})P(X_{1,i}|W_{1,j})P(W_{1,j}) \\
&= P(\nabla f_1|X_{1,i})P(I_1|W_{1,j})P(b_1|W_{1,j})\Pi_{X,ij}\Pi_{W,j}
\end{aligned}$$

The backwards variable for the dynamics process is recursively computed as follows

$$\begin{aligned}
\beta^x_{t-1,ij} &= P(\underset{t,T}{\{\mathbf{O}\}}|X_{t-1,i}W_{t-1,j}) \\
&= \sum_k P(I_t|\underset{t+1,T}{\{\mathbf{O}\}}\nabla f_t W_{t,k}X_{t-1,i}W_{t-1,j})P(\nabla f_t \underset{t+1,T}{\{\mathbf{O}\}}W_{t,k}|X_{t-1,i}W_{t-1,j}) \\
&= \sum_k P(I_t|W_{t,k})P(\nabla f_t \underset{t+1,T}{\{\mathbf{O}\}}|W_{t,k}X_{t-1,i}W_{t-1,j})P(W_{t,k}|X_{t-1,i}W_{t-1,j}) \\
&= \sum_k P(I_t|W_{t,k})P(\nabla f_t \underset{t+1,T}{\{\mathbf{O}\}}|W_{t,k}X_{t-1,i})P(W_{t,k}|X_{t-1,i}W_{t-1,j}) \\
&= \sum_k P(I_t|W_{t,k})\beta^w_{t,ki}\Theta_{W,kij}
\end{aligned}$$

The dynamics backwards process is initialized at time T to be even over the joint space of X, W:

$$\beta^x_{T,ij} = (N_x * N_w)^{-1}$$

The procedure for updating computing the sufficient statistics of the transition parameters for each training sequence is

Initialize κ^x_1
forwards pass
for t = 2:T
    compute κ^w_t from κ^x_{t-1}
    compute κ^x_t from κ^w_t
end

backwards pass
Initialize β^x_T
for t = T:T-1
    compute β^w_t from β^x_t
    compute β^x_{t-1} from β^w_t
end

update sufficient statistics
compute $E_{P(\mathbf{Y}|\mathbf{O}\theta')}(N_{Xijk})$
compute $E_{P(\mathbf{Y}|\mathbf{O}\theta')}(N_{Wijk})$

The likelihood of a sequence of data, $\underset{1,T}{\{\mathbf{O}\}}$ is easily computed from κ^x:

$$P(\underset{1,T}{\{\mathbf{O}\}}) = \sum_{jk} \kappa^x_{T,jk}$$

D.1 Estimating State

Given that we have a coupled hidden Markov model such as we have just described, we may be interested in estimating the temporal state evolution of the dynamics and configuration processes are for some set of observations. That is, we want to find the single best state sequence, $\mathbf{Y}_1 \dots \mathbf{Y}_{N_t}$, given observations, $\mathbf{O}_1 \dots \mathbf{O}_{N_t}$, which can be formulated as maximizing $P(\underset{1,N_t}{\{\mathbf{y}\}}|\underset{1,N_t}{\{\mathbf{O}\}}\mathbf{\Theta})$, which is equivalent to maximizing $P(\underset{1,N_t}{\{\mathbf{y}\}}\underset{1,N_t}{\{\mathbf{O}\}}|\mathbf{\Theta})$. The *Viterbi* algorithm performs this maximization. It needs to be modified slightly to accomodate the two temporal chains. Define $\delta_t(i,j)$ to be the highest probability along a single path which accounts for the first t observations and ends in state $x_t = i, c_t = j$:

$$\delta_t(i,j) = \max_{\underset{1,t-1}{\{\mathbf{Y}\}}} P(\underset{1,t-1}{\{\mathbf{Y}\}} x_{t,i} c_{t,j} \underset{1,t}{\{\mathbf{O}\}})$$

We derive an inductive formula for this quantity as follows

$$\begin{aligned}
\delta_t(i,j) &= \max_{\underset{1,t-1}{\{\mathbf{Y}\}}} P(\underset{1,t-1}{\{\mathbf{Y}\}} x_{t,i} c_{t,j} \underset{1,t}{\{\mathbf{O}\}}) \\
&= \max_{\underset{1,t-1}{\{\mathbf{Y}\}}} P(\underset{1,t-1}{\{\mathbf{Y}\}}|\underset{1,t}{\{\mathbf{O}\}}) P(\mathbf{O}_t x_{t,i} c_{t,j} | P(\underset{1,t-1}{\{\mathbf{Y}\}}\underset{1,t}{\{\mathbf{O}\}}) \\
&= \max_{\mathbf{Y}_{t-1}} \max_{\underset{1,t-2}{\{\mathbf{Y}\}}} P(\underset{1,t-1}{\{\mathbf{Y}\}}|\underset{1,t}{\{\mathbf{O}\}}) P(\mathbf{O}_t x_{t,i} c_{t,j} | \mathbf{Y}_{t-1}) \\
&= \max_{k,l} \max_{\underset{1,t-2}{\{\mathbf{Y}\}}} P(\underset{1,t-2}{\{\mathbf{Y}\}} x_{t-1,k} c_{t-1,l} | \underset{1,t}{\{\mathbf{O}\}}) P(\mathbf{O}_t x_{t,i} c_{t,j} | x_{t-1,k} c_{t-1,l}) \\
&= \max_{k,l} \delta_{t-1}(k,l) P(\mathbf{O}_t | x_{t,i} c_{t,j} \mathbf{Y}_{t-1}) P(x_{t,i} c_{t,j} | x_{t-1,k} c_{t-1,l}) \\
&= \max_{k,l} \left[\delta_{t-1}(k,l) \Theta_{X,ijk} \Theta_{W,jkl}\right] P(\nabla f_t | x_{t,i}) P(I_T | c_{t,j}) \qquad \text{(D.8)}
\end{aligned}$$

The optimal state sequence is given by the argument which maximized Equation (D.8) for each t, i and j. We do this using the array $\psi_t(i,j)$. The procedure is as follows

1. Initialization

$$\begin{aligned}
\delta_1(i,j) &= \Pi_{W,j} \Pi_{X,ij} P(\nabla f_t | x_{t,i}) P(I_T | c_{t,j}) \\
\psi_1(i,j) &= 0
\end{aligned}$$

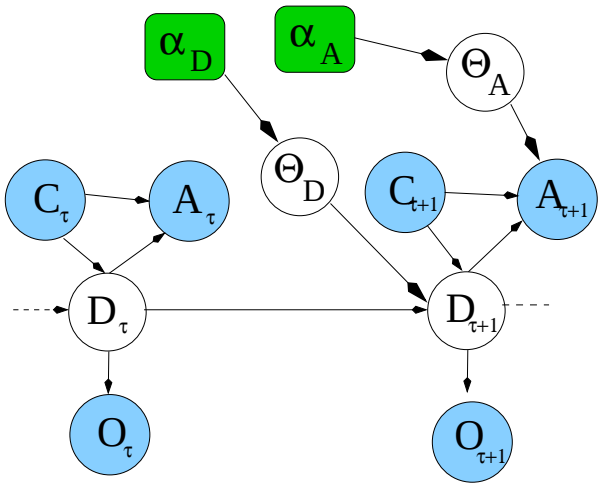


Figure D.2: Context dependent Markov chain of mixtures of hidden Markov models (C4MG) as a dynamic Bayesian network, including context variables, C and A. Repeat of Figure 3.17.

2. Recursion

$$\delta_t(i,j) = \max_{k,l} \left[\delta_{t-1}(k,l)\Theta_{X,ijk}\Theta_{W,jkl}\right] P(\nabla f_t|x_{t,i})P(I_T|c_{t,j})$$
$$\psi_1(i,j) = \arg\max_{k,l} \left[\delta_{t-1}(k,l)\Theta_{X,ijk}\Theta_{W,jkl}\right]$$

3. Termination

$$\mathbf{Y}^*_{N_t} = \arg\max_{k,l} \left[\delta_{t-1}(k,l)\Theta_{X,ijk}\Theta_{W,jkl}\right]$$
$$\mathbf{Y}^*_t = \psi_{t+1}(X^*_{N_t}, W^*_{N_t})$$

where $\mathbf{Y}^*_t = \{X^*_t, W^*_t\}$ is the state at time t along the optimal path.

D.2 C4MG Paramter Updates

Figure D.2 shows the Bayesian network for the context-dependent Markov chain of mixtures of hidden Markov models (C4MG) discussed in Section 4.7. This model can be seen as an input-output hidden Markov model [BF96], and we are trying to maximize

$$\Theta^* = \arg\max_{\Theta} \left[\sum_{\mathbf{D}} P(\mathbf{D}|\mathbf{O},\mathbf{C},\mathbf{A},\theta') \log P(\mathbf{D},\mathbf{O},\mathbf{C},\mathbf{A}|\Theta) + \log P(\Theta)\right]$$

Recall that this Markov chain operates at a higher level than the one we have been previously discussin in this appendix, and so there is no scale variable to marginalise here, and the hidden state is D, which is not factored into two separate Markov chains. The update equations are exactly those for an input-output hidden Markov model, except for the fact that we keep the two output observations, A and O, factored.

The update equation for the D transition parameter, $\Theta_{Dijk} = P(D_{t,i}|D_{t-1,j}C_{t,k})$, is then

$$\Theta_{Dijk} = \frac{\alpha_{Dijk} + \sum_{t\in\{1\ldots N_t\}|C_t=k} P(D_{t,i}D_{t-1,j}|\mathbf{O},\mathbf{A},\mathbf{C}\theta')}{\sum_i \left[\alpha_{Dijk} + \sum_{t\in\{1\ldots N_t\}|C_t=k} P(D_{t,i}D_{t-1,j}|\mathbf{O},\mathbf{A},\mathbf{C}\theta')\right]} \tag{D.9}$$

where the sum over the temporal sequence is only over time steps in which $C_t = k$. The summand can be factored as follows

$$\begin{aligned}
&P(D_{t,i}D_{t-1,j}|\mathbf{O},\mathbf{A},\mathbf{C}\theta') \\
&\quad = P(D_{t,i}D_{t-1,j}\mathbf{OAC})/P(\mathbf{OAC}) \\
&\quad = P(\underset{t+1,T}{\{\mathbf{OAC}\}}|D_{t,i}D_{t-1,j}\underset{1,t}{\{\mathbf{OAC}\}})P(D_{t,i}D_{t-1,j}\underset{1,t}{\{\mathbf{OAC}\}})/P(\mathbf{OAC}) \\
&\quad = \beta_{t,i}P(A_t|D_{t,i},C_{t,k})P(\mathbf{O}_t|D_{t,i})\Theta_{Dijk}\alpha_{t-1,j}
\end{aligned}$$

where $\alpha_{t,j} = P(D_{t,j}\underset{1,t}{\{\mathbf{OAC}\}})$ and $\beta_{t,i} = P(\underset{t+1,T}{\{\mathbf{OAC}\}}|D_{t,i})$ are the usual forwards and backwards variables, for which we can derive recursive updates

$$\begin{aligned}
\alpha_{t,j} &= \sum_k P(D_{t,j}D_{t-1,k}\underset{1,t}{\{\mathbf{OAC}\}}) \\
&= \sum_k P(\mathbf{O}_t A_t|D_{t,j}D_{t-1,k}C_t\underset{1,t-1}{\{\mathbf{OAC}\}})P(D_{t,j}D_{t-1,k}C_t\underset{1,t-1}{\{\mathbf{OAC}\}}) \\
&= \sum_k P(\mathbf{O}_t|D_{t,j})P(A_t|D_{t,j}C_t)P(D_{t,j}|D_{t-1,k},C_t)P(D_{t-1,k}\underset{1,t-1}{\{\mathbf{OAC}\}}) \\
&= \sum_k P(\mathbf{O}_t|D_{t,j})\Theta_{A*j*}\Theta_{Djk*}\alpha_{t-1,k}
\end{aligned}$$

where we write $\Theta_{A*j*} = P(A_t = *|D_{t,j}C_t = *)$

$$\begin{aligned}
\beta_{t-1,i} &= \sum_k P(\underset{t,T}{\{\mathbf{OAC}\}}D_{t,k}|D_{t-1,i}) \\
&= \sum_k P(\underset{t+1,T}{\{\mathbf{OAC}\}}|\mathbf{O}_t A_t C_t D_{t,k} D_{t-1,i})P(\mathbf{O}_t A_t C_t D_{t,k}|D_{t-1,i}) \\
&= \sum_k \beta_{t,k}\Theta_{A*k*}P(\mathbf{O}_t|D_{t,k})\Theta_{Dki*}
\end{aligned}$$

The updates to the output distribution over A , $\Theta_{Aijk} = P(A_{t,i}|D_{t,j}C_{t,k})$ are

$$\Theta_{Aijk} = \sum_{t \in \{1 \ldots N_t\} | A_t = i \wedge C_t = k} P(D_{t,j}|\mathbf{OAC}) \tag{D.10}$$

where the summand is expanded as

$$\begin{aligned} P(D_{t,j}|\mathbf{OAC}) &= P(D_{t,j}\mathbf{OAC})/P(\mathbf{OAC}) \\ &= P(\{\underset{t+1,T}{\mathbf{OAC}}\}|D_{t,j}\{\underset{1,t}{\mathbf{OAC}}\})P(D_{t,j}\{\underset{1,t}{\mathbf{OAC}}\}) \\ &= \beta_{t,j}\alpha_{t,j} \end{aligned}$$