Understanding Expressive Action

by

Christopher R. Wren

B. S. in C. S. & E., Massachusetts Institute of Technology (1993)
M. S. in E. E. & C. S., Massachusetts Institute of Technology (1996)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

March 2000

Author .....
Department of Electrical Engineering and Computer Science
May 1, 2000

Certified by .....
Alex P. Pentland
Academic Head, Program in Media Arts & Sciences
Professor of Media Arts & Sciences
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Understanding Expressive Action

by

Christopher R. Wren

Submitted to the Department of Electrical Engineering and Computer Science
on May 1, 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

We strain our eyes, cramp our necks, and destroy our hands trying to interact with computer on their terms. At the extreme, we strap on devices and weigh ourselves down with cables trying to re-create a sense of place *inside* the machine, while cutting ourselves off from the world and people around us. The alternative is to make the real environment responsive to our actions. It is not enough for environments to respond simply to the presence of people or objects: they must also be aware of the subtleties of changing situations. If all the spaces we inhabit are to be responsive, they must not require encumbering devices to be worn and they must be adaptive to changes in the environment and changes of context.

This dissertation examines a body of sophisticated perceptual mechanisms developed in response to these needs as well as a selection of human-computer interface sketches designed to push the technology forward and explore the possibilities of this novel interface idiom. Specifically, the formulation of a fully recursive framework for computer vision called Dyna that improves performance of human motion tracking will be examined in depth. The improvement in tracking performance is accomplished with the combination of a three-dimensional, physics-based model of the human body with modifications to the pixel classification algorithms that enable them to take advantage of this high-level knowledge. The result is a novel vision framework that has no completely bottom-up processes, and is therefore significantly faster and more stable than other approaches.

Thesis Supervisor: Alex P. Pentland
Title: Academic Head, Program in Media Arts & Sciences
Professor of Media Arts & Sciences

Dedicated to:
Donna Khodarahmi Wren, M.D.
my love, forever.

Contents

List of Figures

Chapter 1

Philosophical Foundations

People live and work. We move through space and organize our environment. We dance. Our most important experiences are interactions with other people, and we spend a lifetime learning to communicate. Our perceptual capabilities and limitations shape the way we perceive the people and world around us.

Today's computers have *no idea* that any of this is going on.

We strain our eyes, cramp our necks, and destroy our hands trying to interact with these machines on their terms. At the extreme, we strap on devices and weigh ourselves down with cables trying to re-create a sense of place *inside* the machine, while cutting ourselves off from the world and people around us.

The alternative is to make the real environment responsive to our actions. When computers start to possess the same perceptual competencies that we use to communicate and interact, amazing things start to happen: computers disappear into the environment, and the things we do naturally suddenly *become* the interface. Interactions with the environment that were previously only meaningful to us are now also meaningful to the environment. Previously inert objects evolve into useful tools. Dead spaces come alive.

It is not enough for environments to respond simply to the presence of objects, they must also be aware of the subtleties of changing situations. If all the spaces we inhabit are to be responsive, they must not require encumbering devices to be worn, or unreasonable constraints to be placed on the environment. They must be adaptive to changes in the environment and changes of context. All these things indicate that these spaces will require significant perceptual capabilities. The required computational tools include computer vision, audio understanding including speech and prosody understanding, and other specialized, remote sensing technologies.

1.1 Claims

This dissertation examines a body of sophisticated perceptual mechanisms developed in response to these needs as well as a selection of human-computer interface sketches de-

signed to push the technology forward and explore the possibilities of this novel interface idiom. Specifically, the formulation of a fully recursive framework for computer vision called Dyna that improves performance of human motion tracking will be examined in depth. The improvement in tracking performance is accomplished with the combination of a three-dimensional, physics-based model of the human body with modifications to the pixel classification algorithms that enable them to take advantage of this high-level knowledge. The result is a novel vision framework that has no completely bottom-up processes, and is therefore significantly faster and more stable than other approaches.

The Dyna framework allows every level of processing to take advantage of the constraints implied by the embodiment of the observed human. Higher level processes take advantage of these constraints explicitly in the process of reestimating the body state, or pose. Lower level processes gain the advantage of the distilled body knowledge in the form of the predicted observations made possible by temporal constraints. These predictions enable more robust classification decisions which in turn result in more accurate pose estimates. Systems which omit the full model will fail to capture important aspects of human motion and their performance will suffer as a result. Understanding embodiment is crucial to perceiving human motion. Systems that rely on any completely bottom-up processing will incur unacceptable performance penalties to recover from inevitable low-level errors, if it is possible to recover at all. Recursive frameworks are also crucial to perceiving human motion.

1.1.1 Embodiment

The fact that people are embodied places powerful constraints on their motion. The laws of physics, the construction of the human skeleton, the layout of the musculature, the various levels of organization within the nervous system, the context of the task, and even forces of habits and culture all conspire to limit the possible configurations and trajectories of the human form. The kinematic constraints of the skeleton are instantaneous. They are always true (we hope), and serve to bound the domain of feasible estimates. The rest of these constraints exist to some degree in the temporal domain: given past observations, they tell us something about future observations.

The times scales of these phenomenon cover a wide range. The laws of physics apply in a continuous, instantaneous fashion. The subtle limits of muscle action may play out on time scales of milliseconds. Temporal structure due to the nervous system may range from tenths of seconds to minutes. Depending on the definition of a task, the task context may change over fractions of a minute or fractions of an hour. Subtle affects of affect might change over hours or even days. Habits and cultural norms are developed over a lifetime.

A truly complete model of human embodiment would encompass all of these things. Unfortunately many of these phenomenon are currently beyond the scope of current modeling techniques. Neuroscience is only beginning to explain the impact of the structures of the peripheral nervous system on motion. Models of the effect of higher processes such as affect, task and culture are even farther away.

The things that we can model explicitly include the instantaneous geometric constraints (blobs, perspective, and kinematics) and the dynamic constraints of Newton's Laws. Blobs

represent a visual constraint that we are composed of parts and those parts appear in images as connected, visually coherent regions. Perspective constraints model the relationship between multiple views of the human body caused by our 3-D nature and the perspective projection of the world onto a CCD by a lens inside a camera. Kinematic constraints are the skeletal or connective constraints between the parts of the body: the length of a limb, the mechanics of a joint, and so on.

Newton's Laws represent a set of dynamic constraints: constraints in time. The assumption of bounded forces in the system implies bounded accelerations. Bounded accelerations in turn imply smoothness of the pose trajectory in time. Since the articulated frame of the body is complex and involves revolute joints, this isn't simply a smoothness constraint. It is a shaping function that is related to the global mass matrix which is a non-linear, time-varying function of the pose.

The rest of the constraint layers (neuromuscular, contextual, and psychological) can currently only be modeled statistically through observation. Fortunately the recursion discussed below offers a natural way to factor out these influences and treat them separately from the geometry and physics. Unfortunately, further factorization of the signal is a poorly understood problem. Since the signals occupy unknown, likely overlapping frequency bands, this leads to hard problems in system identification. As a result, in this work, these separate influences will be treated as a single, unified influence. This is obviously a simplification, but it is curently a necessary simplification.

1.1.2 Recursion

The geometric constraints discussed above are useful for regularizing pose estimation, but the dynamic constraints provide something even more important. Since they represent constraints in time they allow prediction into the future. This is important because on the time scale of video, where new observations are measured every 33ms, physics is a powerful predictor.

With a model of the observation process, predictions of 3-D body pose in the near future can be turned into predictions of observations. These predictions can be compared to actual observations when they are made. Measuring the discrepancy between prediction and observation provides useful information for updating the estimates of the pose. These differences are called innovations because they represent the aspects of the observations that are unexplained by the model.

This link between model and observation is the powerful idea behind all recursive filters, including the well known Kalman filters. Kalman filters are the optimal recursive filter formulation for the class of problems with linear dynamics, linear mappings between state and observation, and white, Gaussian process noise. Extended Kalman filters generalize the basic formulation to include the case of analytically linearizable observation and dynamic models.

Recursive filters are able to cope with data in real time thanks to a Markovian assumption that the state of the system contains all the information needed to predict its behavior. For example, the state of a rigid physical object would include both the position and velocity.

The update of the state estimate then only requires combining the innovation with the dynamic, observation, and noise models.

The complete recursive loop includes measurement, comparison of predicted observation to actual observation, corresponding update of state estimate, prediction of future state estimate, and rendering of the next predicted observation. This is the basic flow of information in a Kalman filter, and applies equally well to recursive filters in general.

For the case of observing the human body, this general framework is complicated by the fact that the human body is a 3-D articulated system and the observation process is significantly non-trivial. Video images of the human body are extremely high-dimensional signals and the mapping between body pose and image observation involves perspective projection. These unique challenges go beyond the original design goals of the Kalman and extended Kalman filters and they make the task of building systems to observe human motion quite difficult.

1.1.3 Classification

Dynamic constraints enable the prediction of observations in the near future. These predictions, with the proper representation, can be used by low-level perceptual processes to resolve ambiguities. Specifically we incorporate the predictions as prior information into the probabilistic blob tracker. The tracker is the first process to be applied to the pixels. There is no part of the system that is bottom-up. Even the lowest level classification process incorporates high-level model influence in the form of state predictions represented as prior probabilities for classification.

Most computer vision systems are modularized to help reduce software complexity, manage bandwidth, and improve performance. Often, low-level modules, comprised of filter-based pattern recognition pipelines, provide features to mid-level modules that then use statistical or logical techniques to infer meaning. The mid-level processes are made tractable by the dimensionality reduction accomplished by the low-level modules, but these improvements can incur a cost in robustness. These systems are often brittle. They fail when the implicit assumptions in a low-level filter are violated. Once a low-level module fails, the information is lost. Even in the case where the mid-level module can employ complex models to predict the failure, there is no way to avert the failure if there is no downward flow of information. The system in forced to rely on complex heuristics to attempt repair.

Dyna avoids this problem by providing the 3-D body model with a powerful form of influence over the low-level blob tracking system. This influence is more significant than simply modifying or bounding a search routine. Our classifier actually produces different results in the presence of feedback: results that reflect global classification decisions instead of decisions that may be misleading or incomplete despite being locally optimal. This modification is made possible due to the statistical nature of our blob tracker. Prior information generated by the body model transforms the bottom-up, maximum likelihood blob tracker into a maximum *a posteriori* classifier.

1.1.4 Expression

An appropriate model of embodiment allows a perceptual system to separate the necessary aspects of motion from the purposeful aspects of motion. The necessary aspects are a result of physics and are predictable. The purposeful aspects are the direct result of a person attempting to express themselves through the motion of their bodies. Understanding embodiment is the key to perceiving expressive motion.

Human-computer interfaces make measurements of a human and use those measurements to give them control over some abstract domain. The sophistication of these measurements range from the trivial keyclick to the most advanced perceptual interface system. Once the measurements are acquired the system usually attempts to extract some set of features as the first step in a pattern recognition system that will convert those measurements into whatever domain of control the application provides. Those features are usually chosen for mathematical convenience or to satisfy an *ad hoc* notion of invariance.

The innovations process discussed above is a fertile source of features that are directly related to the embodiment of the human. When neuromuscular, contextual or psychological influences affect the motion of the body, these effects will appear in the innovations process. This provides direct access for learning mechanisms to these influences without compounding them with the effects of physics, kinematics, imaging, or any other process that can be modeled by the system. This tight coupling between appearance, motion and, behavior is a novel and powerful implication of the framework.

1.2 Expression and Context

The primary motivation for the design and implementation of Dyna was to improve the ability of people to interact with computation. In addition to measuring raw tracking improvements provided by the framework, it is important to explore the important improvements to interface performance. This dissertation will demonstrate the operation of the framework in the context of three applications. The first will showcase enhanced performance for a pre-existing motion tutor called the T'ai Chi Teacher. The second application is a game called Wakka that involves fast-paced manipulation of virtual objects. The final application is a command and control interface built on top of the distributed, open-source game Netrek.

1.2.1 T'ai Chi

The T'ai Chi Teacher is an example of an application that is significantly enhanced by the recursive framework for motion understanding. The basis for this improvement is simply improved tracking stability. The system is an interactive instructional system that teaches a human student to perform a selection of upper-body T'ai Chi gestures[5].

There are several processes that work together to make the T'ai Chi Teacher possible. The Teacher mirrors the actions of the user to provide instant feedback, plays back previously

recorded motions by the student to facilitate reflection, and interprets the motions to generate instructional critiques. All of these processes assume a certain level of stability from the tracker. The mirroring becomes intermittent and distracting in the presence of small tracker failures. The playback engine assumes that a small amount of smoothing will generate a clean version of the user's gesture, and the output becomes less informative when more significant errors disrupt the tracking data. Most importantly, the critique engine is unduly sensitive to tracking errors that violate the noise models of the underlying learning algorithm.

Without changing the T'ai Chi Teacher code, it was possible to improve the robustness of the system by replacing the tracking system with one that follows the framework described above. The improved tracking stability translates into fewer distracting errors, and thus opens the Teacher up to a wider audience. Chapter 5 provides data to illustrate these improvements.

1.2.2 Wakka

Wakka is a virtual manipulation game. Computer vision allows a virtual actor to mimic the motions of the human's upper body. The goal is for the human to touch objects in the virtual world by guiding the hands of the virtual actor. The world contains static objects (Wuggles) and moving objects (Bubbles). Wuggles can always be whacked by returning to the same physical location as before since they do not move in the virtual environment and the mapping between physical and virtual is fixed. Bubbles move, so popping a bubble requires a visuomotor search to find the correct virtual location.

This context is very simple. There are a very small number of things that people do with their hands when playing Wakka. Hands are usually resting, whacking (often in a ballistic fashion once the user learns the system), or performing the more complex feedback loop required to find and pop a bubble. The pace of the game is fast enough to discourage motivated players from wasting time performing extraneous gesticulations. This narrow context makes Wakka ideal for initial experiments exploring the power of the separation between predictable and purposeful aspects of motion.

Chapter 5 includes results on automatic discovery of movement primitives present in tracking data from the Wakka context. These primitives are made more apparent by the separation of predictable and purposeful motion descried above.

1.2.3 Netrek Collective

We have also built a system for interface study around the the game Netrek. Netrek specifies a closed world that is simple enough to be tractable, but not as trivial as the Wakka context. The need to communicate abstract concepts like strategy and coordination provides opportunities to push the limits of what we expect from interfaces.

Context and coordination are very important in Netrek. There are programs, called robots, that know the basics of playing Netrek, but they do not have a very good strategic engine or

any ability to cooperate with other members of their team. Marcus Huber explored the idea of augmenting these robots to include cooperation with each other and found a significant advantage over uncoordinated robots[24]. We take a different approach. With a sufficiently sophisticated interface a human should be able to add strategy, coordination and structure to the robots' activities. This symbiosis between human and robots is called the Netrek Collective.

Chapter 6 includes preliminary results on automatic discovery of movement primitives present in tracking data from the Netrek context as well as an examination of the future potential of this application.

1.3 Document Structure

The next chapter explores the theoretical groundwork that supports this link between the perceptual system and the embodiment of the human being perceived. Chapter 3 describes the intellectual context of the of this work and reviews relevant literature. Chapter 4 details the implementation of the perceptual system motivated by Chapter 2. Chapter 5 presents experimental results that demonstrate improved tracking performance and stability and provides details on several applications, including the T'ai Chi Teacher and Wakka. Chapter 6 presents initial work on the Netrek Collective, and preliminary results that point toward the exciting implications of this framework for interpretation of complex human actions.

Chapter 2

Theoretic Foundations

This chapter will expand on the ideas presented in Chapter 1, while linking them to their roots in stochastic estimation theory.

The fundamental idea presented in Chapter 1 is that perception is improved when it is coupled with expectations about the process being observed: specifically a model with the ability to make qualified predictions into the future given past observations. A logical framework for creating and employing this kind of model in a perceptual system can be found in the control and estimation literature. Since the human body is a physical system, it shares many properties with the general class of dynamic systems. It is instructive to approach the task of understanding human motion in the same way that an engineer might approach the task of observing any dynamic system.

One possible simplified block diagram of a human is illustrated in Figure 2-1. The passive, physical reality of the human body is represented by the *Plant*. The propagation of the system forward in time is governed by the laws of physics and is influenced by signals, $\mathbf{u}$, from *Control*. On the right, noisy observations, $\mathbf{y}$, can be made of the Plant. On the left, high level goals, $\mathbf{v}$, are supplied to the Controller.

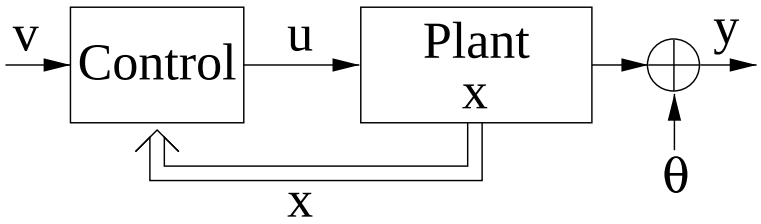

Figure 2-1: A systems view of the human body.

The observations are a function of the system state according to some measurement process, $h(\cdot)$. In our case this measurement process corresponds to the imaging process of a camera. As such, it is a non-linear, incomplete transform: cameras do not directly measure velocity, they are subject to occlusion, and they project 3-D information into 2-D observations:

$$\mathbf{y}_{t+1} = h(\mathbf{x}_{t+1}) + \boldsymbol{\theta}_t \tag{2.1}$$

The measurement process is also noisy. $\boldsymbol{\theta}_t$ represents additive noise in the observation. The $\boldsymbol{\theta}_t$ are assumed to be samples from a white, Gaussian, zero-mean process with covariance $\boldsymbol{\Theta}$:

$$\boldsymbol{\theta}_t \leftarrow \mathcal{N}(\mathbf{0}, \boldsymbol{\Theta}) \tag{2.2}$$

The state vector, $\mathbf{x}$, completely defines the configuration of the system in phase-space. The Plant propagates the state forward in time according to the system constraints. In the case of the human body this includes the non-linear constraints of kinematics as well as the constraints of physics. The Plant also reacts to the influences of the control signal. For the human body these influences come as muscle forces. It is assumed that the Plant can be represented by an unknown, non-linear function $f(\cdot, \cdot)$:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \tag{2.3}$$

The control signals are physical signals, for example, muscle activations that result in forces being applied to the body. The Controller obviously represents a significant amount of complexity: muscle activity, the properties of motor nerves, and all the complex motor control structures from the spinal cord up into the cerebellum. The Controller has access to the state of the Plant, by the process of proprioception:

$$\mathbf{u}_t = c(\mathbf{v}_t, \mathbf{x}_t) \tag{2.4}$$

The high-level goals, $\mathbf{v}$, are very high-level processes. These signals represent the place where intentionality enters into the system. If we are building a system to interact with a human, then we get the observations, $\mathbf{y}$, and what we're really interested in is the intentionality encoded in $\mathbf{v}$. Everything else is just in the way.

2.1 A Classic Observer

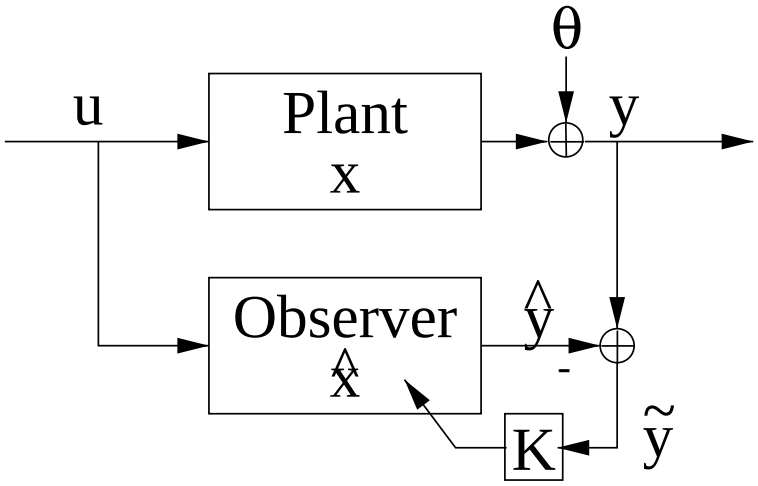

Figure 2-2: Classic observer architecture

A classic observer for such a system takes the form illustrated in Figure 2-2. This is the underlying structure of recursive estimators, including the well known Kalman and extended Kalman filters.

The *Observer* is an analytical model of the physical Plant:

$$\mathbf{x}_{t+1} = \mathbf{\Phi}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t + \mathbf{L}_t \boldsymbol{\xi}_t \tag{2.5}$$

The unknown, non-linear update equation, $f(\cdot,\cdot)$ from Equation 2.3, is modeled as the sum of two non-linear functions: $\Phi(\cdot)$ and $B(\cdot)$. $\Phi(\cdot)$ propagates the current state forward in time, and $B(\cdot)$ maps control signals into state influences. $\mathbf{\Phi}_t$ and $\mathbf{B}_t$ from Equation 2.5 are linearizations of $\Phi(\cdot)$ and $B(\cdot)$ respectively, at the current operating point. The right-hand term, $(\mathbf{L}_t \boldsymbol{\xi}_t)$, represents the effect of noise introduced by modeling errors on the state update. The $\boldsymbol{\xi}_t$ are assumed to be samples from a white, Gaussian, zero-mean process with covariance $\mathbf{\Xi}$ that is independent of the observation noise from Equation 2.2:

$$\boldsymbol{\xi}_t \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{\Xi}) \tag{2.6}$$

The model of the measurement process is also linearized. $\mathbf{H}_t$ is a linearization of the non-linear measurement function $h(\cdot)$:

$$\mathbf{y}_{t+1} = \mathbf{H}_t \mathbf{x}_{t+1} + \boldsymbol{\theta}_t \tag{2.7}$$

Linearizations, such as those invoked to form $\mathbf{\Phi}_t$ and $\mathbf{H}_t$, are performed by computing the Jacobian matrix. The Jacobian of a multivariate function of $\mathbf{x}$ such as $\mathbf{\Phi}(\cdot)$ is computed as the matrix of partial derivatives at the operating point $\mathbf{x}_t$ with respect to the components of $\mathbf{x}$:

$$\mathbf{\Phi}_t = \nabla_{\mathbf{x}} \mathbf{\Phi}\big|_{\mathbf{x}=\mathbf{x}_t} = \begin{bmatrix} \frac{\partial \mathbf{\Phi}_1}{\partial \mathbf{x}_1}\Big|_{\mathbf{x}=\mathbf{x}_t} & \frac{\partial \mathbf{\Phi}_1}{\partial \mathbf{x}_2}\Big|_{\mathbf{x}=\mathbf{x}_t} & \cdots & \frac{\partial \mathbf{\Phi}_1}{\partial \mathbf{x}_n}\Big|_{\mathbf{x}=\mathbf{x}_t} \\ \frac{\partial \mathbf{\Phi}_2}{\partial \mathbf{x}_1}\Big|_{\mathbf{x}=\mathbf{x}_t} & \ddots & & \vdots \\ \vdots & & \ddots & \\ \frac{\partial \mathbf{\Phi}_m}{\partial \mathbf{x}_1}\Big|_{\mathbf{x}=\mathbf{x}_t} & \cdots & \cdots & \frac{\partial \mathbf{\Phi}_m}{\partial \mathbf{x}_n}\Big|_{\mathbf{x}=\mathbf{x}_t} \end{bmatrix}$$

This operation is often non-trivial.

Estimation begins from a prior estimate of state: $\hat{\mathbf{x}}_{0|0}$. Given the current estimate of system state, $\hat{\mathbf{x}}_{t|t}$, and the update Equation 2.5, it is possible to compute a prediction for the state at $t+1$:

$$\hat{\mathbf{x}}_{t+1|t} = \mathbf{\Phi}_t \hat{\mathbf{x}}_{t|t} + \mathbf{B}_t \mathbf{u}_t \tag{2.8}$$

Notice that $\boldsymbol{\xi}_t$ falls out since:

$$E[\boldsymbol{\xi}_t] = E\left[\mathcal{N}(\mathbf{0}, \mathbf{\Xi})\right] = \mathbf{0} \tag{2.9}$$

Combining this state prediction with the measurement model provides a prediction of the next measurement:

$$\hat{\mathbf{y}}_{t+1|t} = \mathbf{H}_t \hat{\mathbf{x}}_{t+1|t} \tag{2.10}$$

Again, $\boldsymbol{\theta}_t$ drops out since:

$$E[\boldsymbol{\theta}_t] = E\left[\mathcal{N}(\mathbf{0}, \boldsymbol{\Theta})\right] = \mathbf{0} \tag{2.11}$$

Given this prediction it is possible to compute the residual error between the prediction and the actual new observation $\mathbf{y}_{t+1}$:

$$\tilde{\mathbf{y}}_{t+1} = \boldsymbol{\nu}_{t+1} = \mathbf{y}_{t+1} - \hat{\mathbf{y}}_{t+1|t} \tag{2.12}$$

This residual, called the innovation, is the information about the actual state of the system that the filter was unable to predict, plus noise. A weighted version of this residual is used to revise the new state estimate for time $t+1$ to reflect the new information in the most recent observation:

$$\hat{\mathbf{x}}_{t+1|t+1} = \hat{\mathbf{x}}_{t+1|t} + \mathbf{K}_{t+1}\tilde{\mathbf{y}}_{t+1} \tag{2.13}$$

In the case of the Kalman filter, the weighting matrix is the well-known Kalman gain matrix. It is computed from the estimated error covariance of the state prediction, the measurement models, and the measurement noise covariance, $\boldsymbol{\Theta}$:

$$\mathbf{K}_{t+1} = \boldsymbol{\Sigma}_{t+1|t}\mathbf{H}_t^T \left[\mathbf{H}_t \boldsymbol{\Sigma}_{t+1|t}\mathbf{H}_t^T + \boldsymbol{\Theta}_{t+1}\right]^{-1} \tag{2.14}$$

The estimated error covariance of the state prediction is initialized with the estimated error covariance of the prior state estimate, $\boldsymbol{\Sigma}_{0|0}$. As part of the state prediction process, the error covariance of the state prediction can be computed from the error covariance of the previous state estimate using the dynamic update rule from Equation 2.5:

$$\boldsymbol{\Sigma}_{t+1|t} = \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{t|t} \boldsymbol{\Phi}_t^T + \mathbf{L}_t \boldsymbol{\Xi}_t \mathbf{L}_t^T \tag{2.15}$$

Notice that, since $\mathbf{u}_t$ is assumed to be deterministic, it does not contribute to this equation.

Incorporating new information from measurements into the system reduces the error covariance of the state estimate: after a new observation, the state estimate should be closer to the true state:

$$\boldsymbol{\Sigma}_{t+1|t+1} = \left[\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H}_t\right] \boldsymbol{\Sigma}_{t+1|t} \tag{2.16}$$

Notice, in Equation 2.5, that that classic Observer assumes access to the control signal $\mathbf{u}$. For people, remember that the control signals represent muscle activations that are unavailable to a non-invasive Observer. That means that an observer of the human body is in the slightly different situation illustrated in Figure 2-3.

2.2 A Lack of Control

Simply ignoring the $(\mathbf{B}_t\mathbf{u})$ term in Equation 2.5 results in poor estimation performance. Specifically, the update Equation 2.13 expands to:

$$\hat{\mathbf{x}}_{t+1|t+1} = \hat{\mathbf{x}}_{t+1|t} + \mathbf{K}_{t+1}(\mathbf{y}_{t+1} - \mathbf{H}_t(\boldsymbol{\Phi}_t \hat{\mathbf{x}}_{t|t} + \mathbf{B}_t \mathbf{u}_t)) \tag{2.17}$$

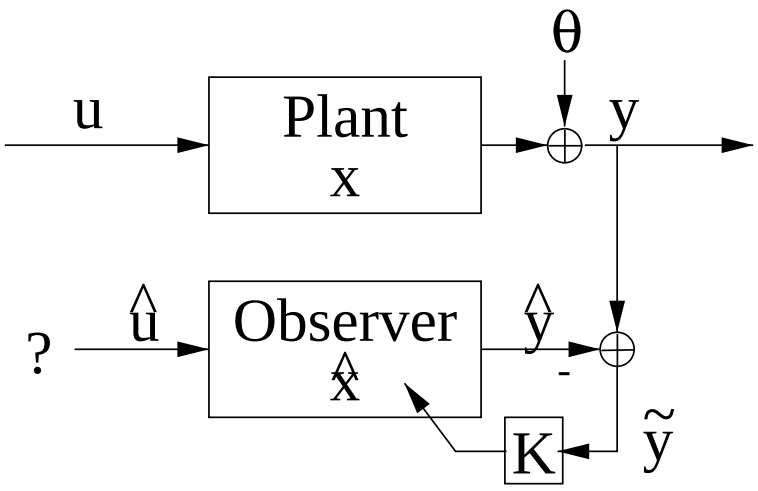

Figure 2-3: An Observer of the human body can't access $\mathbf{u}$

In the absence of access to the control signal $\mathbf{u}$, the update equation becomes:

$$\tilde{\hat{\mathbf{x}}}_{t+1|t+1} = \hat{\mathbf{x}}_{t+1|t} + \mathbf{K}_{t+1}(\mathbf{y}_{t+1} - \mathbf{H}_t(\mathbf{\Phi}_t\hat{\mathbf{x}}_{t|t} + \mathbf{B}_t\mathbf{0})) \tag{2.18}$$

The error ε between the ideal update and the update without access to the control signal is then:

$$\begin{aligned} \varepsilon &= \left|\hat{\mathbf{x}}_{t+1|t+1} - \tilde{\hat{\mathbf{x}}}_{t+1|t+1}\right| && (2.19)\\ &= \mathbf{K}_{t+1}\mathbf{H}_t\mathbf{B}_t\mathbf{u}_t && (2.20) \end{aligned}$$

Treating the control signal, $\mathbf{u}_t$, as a random variable, we compute the control mean and covariance matrix:

$$\begin{aligned} \bar{\mathbf{u}} &= E[\mathbf{u}_t] && (2.21)\\ \mathbf{U} &= E[(\mathbf{u}_t - \bar{\mathbf{u}})(\mathbf{u}_t - \bar{\mathbf{u}})^T] && (2.22) \end{aligned}$$

If the control covariance matrix is small relative to the model and observation noise, by which we mean:

$$\begin{aligned} \|\mathbf{U}\| &<< \|\mathbf{\Xi}_t\| && (2.23)\\ \|\mathbf{U}\| &<< \|\mathbf{\Theta}_t\| && (2.24) \end{aligned}$$

then the standard recursive filtering algorithms should be robust enough to generate good state and covariance estimates. However, as $\mathbf{U}$ grows, so will the error ε. For large enough $\mathbf{U}$ it will not be possible to hide these errors within the assumptions of white, Gaussian process noise, and filter performance will significantly degrade [1].

It should be obvious that we expect $\mathbf{U}$ to be large: if $\mathbf{u}$ had only negligible impact on the evolution of $\mathbf{x}$, then the human body wouldn't be very effective. The motion of the human body is influenced to a large degree by the actions of muscles and the control structures

driving those muscles. This situation will be illustrated in Chapter 4.

2.3 Estimation of Control

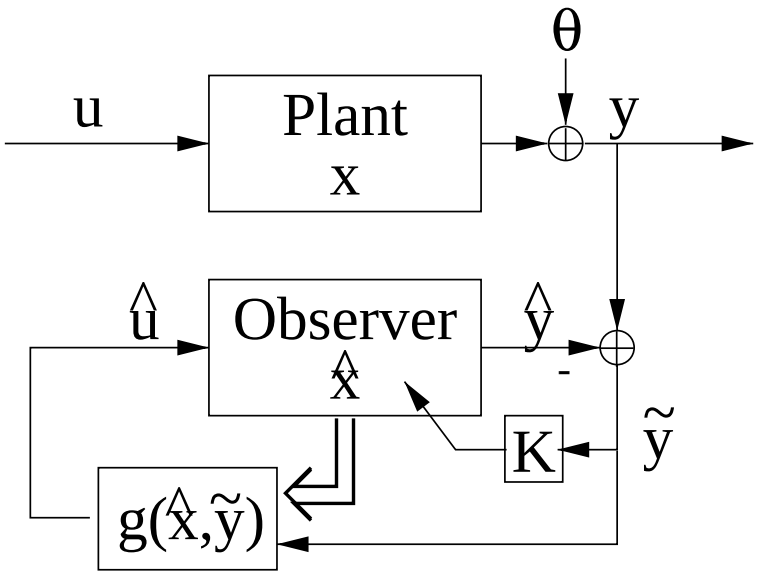

Figure 2-4: An observer that estimates $\hat{\mathbf{u}}$ as well as $\hat{\mathbf{x}}$

It is not possible to measure $\mathbf{u}_t$ directly. It is inadvisable to ignore the effects of active control, as shown above. An alternative is to estimate $\mathbf{u}_{t+1|t}$. This alternative is illustrated in Figure 2-4: assuming that there is some amount of structure in $\mathbf{u}$, the function $g(\cdot,\cdot)$ uses $\hat{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ to estimate $\hat{\mathbf{u}}$.

The measurement residual, $\tilde{\mathbf{y}}_{t+1}$ is a good place to find information about $\mathbf{u}_t$ for several reasons. Normally, in a steady-state observer, the measurement residual is expected to be zero-mean, white noise, so $E[\tilde{\mathbf{y}}_t] = 0$. From Equation 2.20 we see that without knowledge of $\mathbf{u}_t$, $\tilde{\mathbf{y}}_{t+1}$ will be biased:

$$E[\tilde{\mathbf{y}}_{t+1}] = \mathbf{H}_t\mathbf{B}_t\mathbf{u}_t \tag{2.25}$$

This bias is caused by the faulty state prediction resulting in a biased measurement prediction. Not only will $\tilde{\mathbf{y}}_{t+1}$ not be zero-mean, it will also not be white. Time correlation in the control signal will introduce time correlation in the residual signal due to the slow moving bias. Specific examples of such structure in the residuals will be shown in Chapter 4.

Learning the bias and temporal structure of the measurement residuals provides a mechanism for learning models of $\mathbf{u}$. Good estimates of $\mathbf{u}$ will lead to better estimates of $\mathbf{x}$ which are useful for a wide variety of applications including motion capture for animation, direct manipulation of virtual environments, video compositing, diagnosis of motor disorders, and others. However, if we remain focused on the intentionality represented by $\mathbf{v}$ on the far left of Figure 2-1, then this improved tracking data is only of tangential interest as a means to compute $\hat{\mathbf{v}}$.

The neuroscience literature[43] is our only source of good information about the control structures of the human body, and therefore the structure of $\mathbf{v}$. This literature seems to indicate that the body is controlled by the setting of goal states. The muscles change activation in response to these goals, and the limb passively evolves to the new equilibrium point. The time scale of these mechanisms seem to be on the scale of hundreds of milliseconds.

Given this apparent structure of $\mathbf{v}$, we expect that the internal structure of $g(\cdot,\cdot)$ should contain states that represent switches between control paradigms, and thus switches in the high-level intentionality encoded in $\mathbf{v}$. Chapter 4 discusses possible representations for $g(\cdot,\cdot)$ and Chapter 5 discusses results obtained in controlled contexts (where the richness of $\mathbf{v}$ is kept manageable by the introduction of a constrained context).

2.4 Images as Observations

There is one final theoretic complication with this formulation of an observer for human motion. Recursive filtering matured under the assumption that the measurement process produced low-dimensional signals under a measurement model that could be readily linearized: such as the case of a radar tracking a ballistic missile. Images of the human body taken from a video stream do not fit this assumption: they are high dimensional signals and the imaging process is complex.

One solution, borrowed from the pattern recognition literature, is to place a deterministic filter between the raw images and the Observer. The measurements available to the Observer are then low-dimensional features generated by this filter [46]. This situation is illustrated in Figure 2-5.

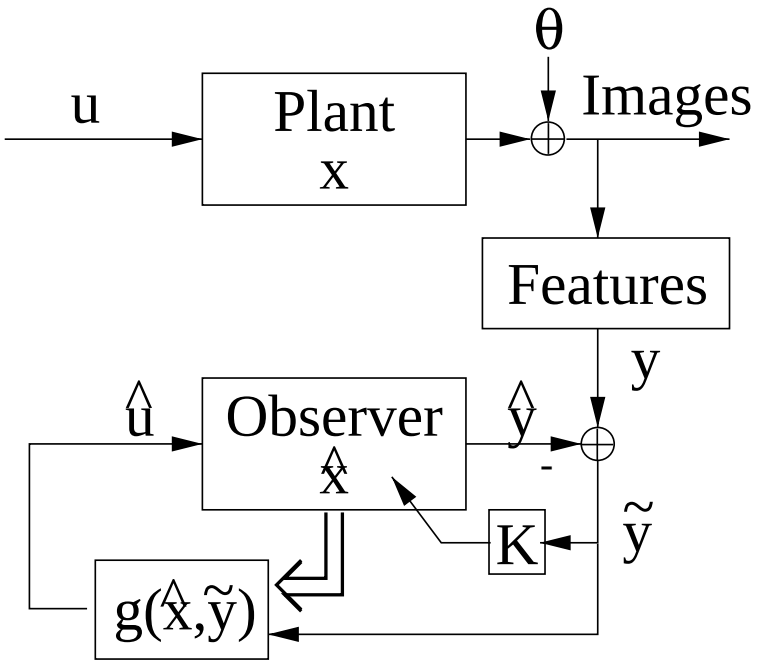

Figure 2-5: Feature Extraction between image observations and the Observer.

One fatal flaw in this framework is the assumption that it is possible to create a stationary filter process that is robust and able to provide all the relevant information from the image as a low dimensional signal for the Observer. This assumption essentially presumes a pre-existing solution to the perception problem. A sub-optimal filter will succumb to the problem of perceptual aliasing under a certain set of circumstances specific to that filter. In these situations the measurements supplied to the Observer will be flawed. The filter will have failed to capture critical information in the low-dimensional measurements. It is unlikely that catastrophic failures in feature extraction will produce errors that fit within the assumed white, Gaussian, zero-mean measurement noise model. Worse, the situation in Figure 2-5 provides no way for the predictions available in the Observer to avert these failures. This problem will be demonstrated in more detail in Chapter 3.

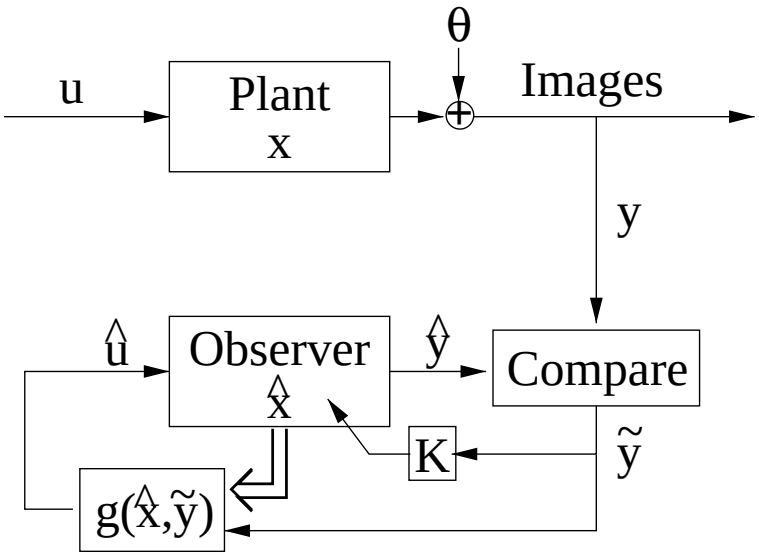

Figure 2-6: The Observer driving a steerable feature extractor.

A more robust solution is illustrated in Figure 2-6. A steerable feature extraction process takes advantage of observation predictions to resolve ambiguities. It is even possible to compute an estimate of the observation prediction error covariance, $(\mathbf{H}_t \mathbf{\Sigma}_{t+1|t} \mathbf{H}_t^T)$ and weight the influence of these predictions according to their certainty. Since this process takes advantage of the available predictions it does not suffer from the problems described above, because prior knowledge of ambiguities enables the filter to anticipate catastrophic failures. This allows the filter to more accurately identify failures and correctly propagate uncertainty, or even change modes to better handle the ambiguity. A fast, robust implementation of such a system is described in detail in Chapter 4.

2.5 Sumarry

So we see that exploring the task of observing the human from the vantage of classical control theory provides interesting insights. The powerful recursive link between model and observation will allow us to build robust and fast systems. Lack of access to control signals represent a major difference between observing built systems and observing biological

systems. Finally that there is a possibility of leveraging the framework to help in the estimation of these unavailable but important signals.

For the case of observing the human body, this general framework is complicated by the fact that the human body is a 3-D articulated system and the observation process is significantly non-trivial. Video images of the human body are extremely high-dimensional signals and the mapping between body pose and image observation involves perspective projection. These unique challenges go beyond the original design goals of the Kalman and extended Kalman filters and they make the task of building systems to observe human motion quite difficult. The details involved in extending the basic framework to this more complex domain are the subject of Chapter 4. First, Chapter 3 will explore the alternatives to this framework that are to be found in the literature.

Chapter 3

Background

In recent years there has been much interest in tracking the human body using 3-D models with kinematic and dynamic constraints. Perhaps the first efforts at body tracking were by Badler and O'Rourke in 1980, followed by Hogg in 1988 [35, 34]. These early efforts used edge information to drive a kinematic model of the human body. These systems require precise hand initialization, and can not handle the full range of common body motion.

Following this early work using kinematic models, some researchers began using dynamic constraints to track the human body. Pentland and Horowitz employed non-rigid finite element models driven by optical flow in 1991[37], and Metaxas and Terzopolous' 1993 system used deformable superquadrics [29, 32] driven by 3-D point and 2-D edge measurements. Again, these systems required precise initialization and could handle only a limited range of body motion.

More recently, authors have applied variations on the basic kinematic analysis-synthesis approach to the body tracking problem [41, 4, 22]. Gavrila and Davis [20] and Rehg and Kanade[40] have demonstrated that this approach has the potential to deal with limited occlusions, and thus to handle a greater range of body motions.

Dyna, the system described here, combines the dynamic modeling work with the advantages of a recursive approach, by use of an extended Kalman filter formulation that couples a fully dynamic skeletal model with observations of raw pixel values, as modeled by probabilistic 'blob' models.

Dyna also explicitly incorporates learned patterns of control into the body model. The approach we take is based on the behavior modeling framework introduced by Pentland and Liu in 1995[38]; it is also related to the behavior modeling work of Blake[26] and Bregler[8]. However, this controller operates on a 3-D non-linear model of human motion that is closer to true body dynamics than the 2-D linear models previously employed.

3.1 Particle Filtering

It is useful to compare the approach advocated here with that taken by particle filtering approaches such as the CONDENSATION algorithm proposed by Isard and Blake[27]. Both approaches have their roots in recursive filtering and control theory as described in Chapter 2.

The fundamental difference between this method and particle filtering methods is in the representation of the probability density from which the state estimate is drawn. Recursive filtering methods, of which the Kalman filter is an example, represent the density parametrically as a unimodal Gaussian. The CONDENSATION algorithm is a framework for representing the density as a set of samples drawn from the underlying distribution and is therefore not parametric, nor even necessarily unimodal. The Multiple Hypothesis Testing (MHT) framework from the control and estimation community[1] provides an extension to recursive filtering frameworks that allows multi-modal, parametric representations. Recent work by Rehg[11] shows how unimodal trackers such as the one described here may be embedded in the MHT framework. This section will discuss these issues in detail, as they relate to this work.

All recursive filtering frameworks require a mechanism for propagation forward in time of the estimated density. In Kalman filtering this is accomplished in parameter space using the model of state dynamics to update both the mean, $\hat{\mathbf{x}}_t$, and variance, $\mathbf{\Sigma}_t$, of the density. Kalman filters assume a linear model of the state dynamics, $\mathbf{\Phi}$. The mean update is accomplished by equation 3.1, and the variance update by equation 3.2:

$$\hat{\mathbf{x}}_{t+1|t} = \mathbf{\Phi}_t \hat{\mathbf{x}}_{t|t} + \mathbf{B}_t \mathbf{u}_t \tag{3.1}$$

$$\mathbf{\Sigma}_{t+1|t} = \mathbf{\Phi}_t \mathbf{\Sigma}_{t|t} \mathbf{\Phi}_t^T + \mathbf{L}_t \mathbf{\Xi}_t \mathbf{L}_t^T \tag{3.2}$$

For non-linear dynamics, the system is linearized near the operating point at each time step.

In the CONDENSATION algorithm, the distribution is represented non-parametrically, so it must employ a different mechanism for state update. Specifically the distribution is represented as a set of samples, S_t, in model state space with an associated set of weights Π_t. The system dynamics are represented as a density conditional on the current model state: $p(X_{t+1}|X_t)$. Model update begins by selecting a sample, $s_t^{(i)}$, from the set S_t with probability determined by the set of weights Π_t. A new sample is then drawn from the corresponding density $p(X_{t+1}|X_t) = s_t^{(i)}$, and becomes the sample $s_{t+1}^{(i)}$ in the updated representation. The new weight $\pi_{t+1}^{(i)}$ is determined by comparing the sample $s_{t+1}^{(i)}$ to the image evidence.

A tracker using CONDENSATION must maintain a large number of samples to adequately represent the underlying distribution. The literature calls for approximately 1500 samples to track a bouncing ball with a single degree of freedom, and as many as 15,0000 samples to track the hand during a drawing task [28]. The advantage is that the distribution is not required to be unimodal. Since the Kalman filter chooses a parametric representation that is unimodal it is possible for ambiguous observations to move the single mode away from the truth. Dynamic state updates will likely cause the location of the mode to diverge

further from the truth over time. The potential for multi-modality in the representation of density in CONDENSATION means that it is possible for a portion of the probability mass to be allocated to divergent hypotheses. As long as there are a sufficient number of samples available to adequately represent the distribution, alternate hypotheses may be propagated forward as long as there are observations to support them.

One issue with Kalman-derived filtering techniques is that it is difficult to formulate closed-form update equations for arbitrary parametric density representations as was done for the unimodal Gaussian case in equation 3.1. MHT is a method for maintaining a set of N unimodal Gaussian hypotheses in the Kalman filtering framework without the need to reformulate the update equations. The MHT method is essentially a bank of Kalman filters. Each Kalman filter makes predictions according to its own, possibly unique, state estimate and dynamic model. The resulting innovations sequence is a measure of how well each filter is explaining the observed data. Filters with large innovations are poor predictors and are candidates for reinitialization [1]. Due to the fact that each filter accounts parametrically for some component of the underlying density, only a handful of hypotheses are required, as compared to the thousands of point hypotheses required by CONDENSATION. The literature shows that MHT can track a human figure, even with relatively weak underlying unimodal trackers, with as few as 10 hypotheses[11].

So these methods represent extremes of a continuum where computational cost can be traded off against modeling cost. For systems where the cost of model-building is very high compared to the cost of computation, CONDENSATION is a good, general method for tracking. For situations where the dynamics is reasonably well understood and the cost of computation is very high (as in the case of real-time human motion understanding), then parametric recursive filtering, possibly in conjunction with an MHT framework, is a better choice.

3.1.1 Recent Work on CONDENSATION

In the time since the first description of CONDENSATION in 1996 [26] there have been several documenting modifications to CONDENSATION that help avoid the curse of dimensionality involved in tracking systems with many degrees of freedom or with complex dynamic models. Mostly the improvements in computational efficiency of the algorithm are achieved by incorporating more sophisticated domain models. The pure form of CONDENSATION was the general tracker that learned dynamics from observation at high computational cost. These extensions involve replacing flocks of hypotheses with domain-specific models that are better at capturing the true dynamics of the systems. The logical conclusion to this trend is the embedding of DYNA in an MHT framework.

MacCormick and Blake [31] propose an extension that culls samples that represent inconsistent model states. This requires the CONDENSATION implementation to have more domain knowledge. MHT, as described in [11], implicitly includes this capability. For the task of tracking single degree of freedom motion of human bodies, this improvement allows tracking 2-D movement of head and shoulders of two people with 2000 samples. That is compared with tracking three people, two stationary and one moving with a single degree

of freedom with 1000 samples.

Duetscher, North, Bascle, and Blake [15] adds the notion of kinematic singularities and end-stops. By adding significant domain knowledge they bring CONDENSATION much closer to MHT, requiring 50 samples to track arm motion with two degrees of freedom (elbow flexion and and shoulder rotation). They also track walking motion that is similar to, but less energetic than the dancing motions tracked by the MHT system in [11] with only 10 hypotheses. The paper does not report the number of particles required to track the walking sequence.

3.2 Analysis-Synthesis

The analysis-synthesis approach is a widely used approximation to the general framework of recursive filtering. There are two major differences between the recursive framework advocated here and the analysis-synthesis (AS) framework: AS includes a significant feed-forward component and lacks a model of system dynamics. Each of these issues are discussed in detail below.

3.2.1 Overview of Analysis-Synthesis

An AS tracker is composed of an Analysis module, a Synthesis module and a model of the system to be tracked. The Analysis module is composed of a set of image processing routines that extracts features from image observations. The Synthesis module renders the model state as a synthetic image composed of features congruent to those extracted by the image processing routines. The overall goal is to directly compare the model state to the observed image. An iterative process updates the model state until the best match between image and model is found. This structure is illustrated in Figure 3-1.

Due to the complexity of real world image production, direct comparison is very difficult. The Synthesis module would have to generate photo-realistic images of each hypothesized configuration. Further, direct image comparison would likely be dominated by aspects of the scene that may be uninteresting: the direction of illumination for example. As a result the comparison is actually done in an abstract feature space. Image processing routines are used to extract a set of features from the image. In practice these features are often edge features, but in principle could be any low-level feature. The model state is rendered into this feature space by the Synthesis model. The set of features produced by the Synthesis module is then compared to the set of features extracted by the Analysis module. Discrepancies between analysis features and synthesis features represent a residual to be minimized by the system. The minimization of this residual is solved using iterative gradient descent in model space. This requires a model of how variation in feature space maps into variation in model space. Since Synthesis likely involves complex non-linear components from revolute model variables, perspective projection, and other processes, this model is usually a local, linear approximation. This places constraints on the gradient descent step size.

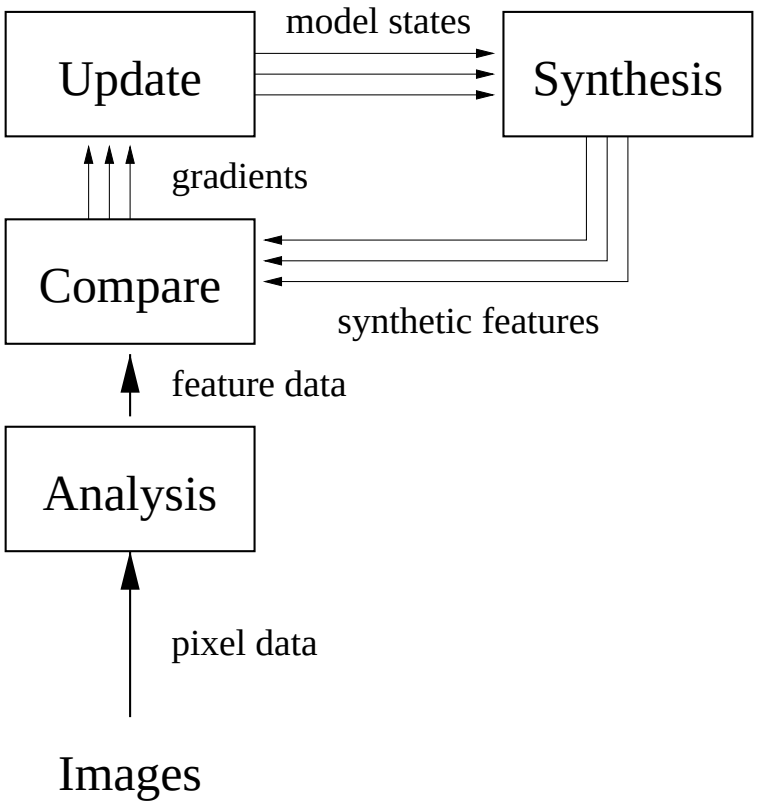

Figure 3-1: System diagram of an Analysis-Synthesis tracker. Multiple thin arrows represent an iterative loop operating at a higher frequency than the observation rate.

In the case of tracking articulated bodies or body-parts, the model usually consists of a kinematic linkage. This model can be linearized around the current estimated operating point to yield a set of Jacobian matrices relating variation in feature space to model state perturbance. These Jacobians are used to transform the measured residuals into model adjustments. The iteration process may be weighted by temperature coefficients, or filtered by other, more sophisticated, gradient decent approaches.

3.2.2 The Feed-Forward Nature of Analysis

If we formulate the Recursive Filter using the AS terminology, as in Figure 3-2, a major difference between the two approaches is emphasized: in AS the Analysis module is feed-forward, meaning that the image processing routines operate without access to any context that may be available in the rest of the system. The Analysis module in the Recursive Filter explicitly takes advantage of prior information for processing, as described in Chapter 4.

This one failing has a serious impact on the stability of the system. The Analysis module does not have access to the high-level constraints coded in the system model or Synthesis module. In order to track features reliably it must either encode some or all of these constraints within itself, or rely on heuristics independent of the known image context. At the same time, if the Analysis module fails, the gradient decent must proceed without any

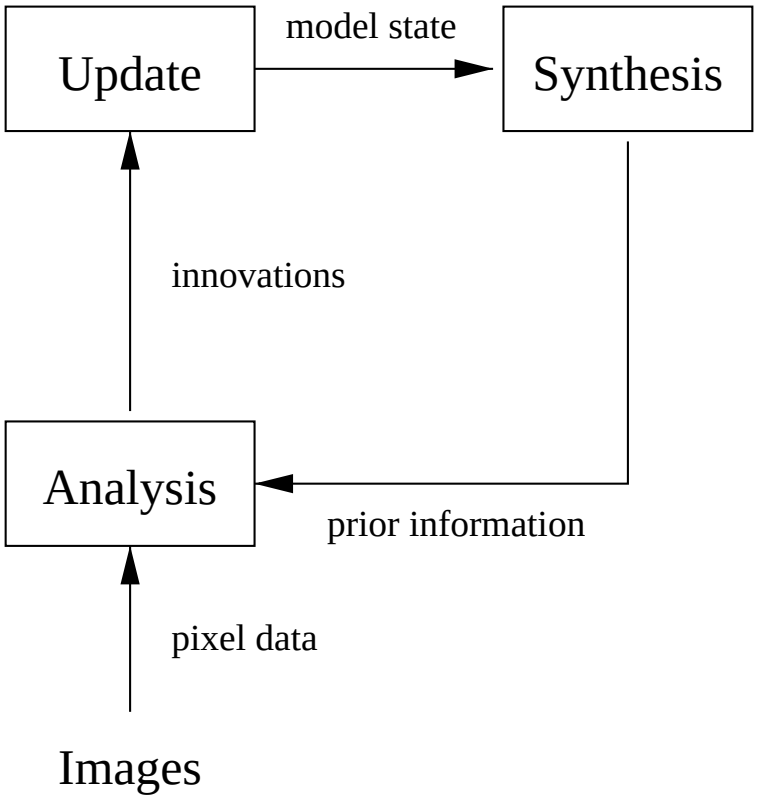

Figure 3-2: System diagram of a Recursive Filter formulated to facilitate comparisons with the Analysis-Synthesis system illustrated in Figure 3-1

relevant observations. There is no one place where images and high-level knowledge are brought together. The two domains are separated by the feature abstraction. The benefits of breaking this barrier are discussed in detail in Section 5.1.

It is important to note that merely providing prior information to an Analysis module is not enough. The implementation must have enough mathematical flexibility to incorporate the information in a meaningful way. As an example, convolution trackers often use simple Kalman predictors to improve tracking performance, but is this a truly recursive system? Consider the normal operation of a convolution tracker: an image patch is copied from an image, possibly matted, and then convolved with an image (or a region of an image). The maximum resulting value is accepted as the new location of the feature represented by the image patch. This process is shown graphically in Figure 3-3.

The predictions from the Kalman filter would indicate the most likely new location as well as an estimate of the error in this prediction. This region of uncertainly is shown as a dotted oval in Figure 3-4. The easiest, and most common, way to incorporate this prior knowledge into the tracker is to constrain the search to the likely area as shown in Figure 3-4. This does not result in a better answer. It merely results in a more efficient answer since less area needs to be searched[23]. If the "right" answer is within the oval it will be found faster, but there will be no difference between the result of this search and the result of the slower search. A truly recursive system would yield a different maximum.

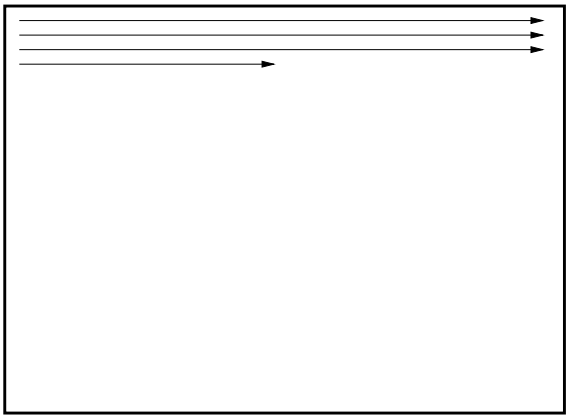

Figure 3-3: Path of a convolution tracker over an image.

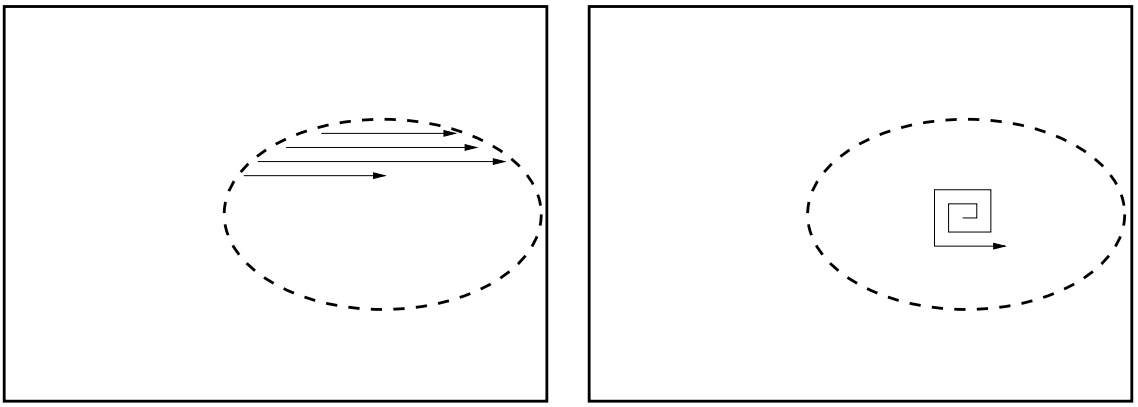

Figure 3-4: Prior information used to optimize the path of a convolution tracker over an image.

This raises the question of feature selection. Probabilistic features such as the blobs presented in Chapter 4 provide a mathematically natural method of integrating high-level knowledge. Features such as convolution patches, edges, and other deterministic, filter-based features, do not provide easy integration of context.

3.2.3 Lack of System Dynamics

The second major difference between the two techniques is the form of the model update. In AS, a distance metric is induced in the chosen feature space. These gradients are then transformed into model space to determine the appropriate system update. The system update is completely driven by observations. The Recursive Filter update is shaped not only by observation models, but also the system dynamics, as described in Chapter 4. This enables the Recursive Filter to take advantage of the constraints of system dynamics when determining the appropriate system update. Since real dynamic systems are heavily constrained by the laws of physics and the structure of their control mechanisms, the system dynamics is an important source of context for the observer.

The lack of these constraints on the update mechanism in AS means that past model configurations do not inform the current choice of state estimate. Ambiguities or errors in feature extraction may result in the iteration process finding an explanation that, while plausible in isolation, is impossible due to the physical and practical limitations on the actual system being observed. One example of this sort of failure is the instantaneous flip between two solutions that are ambiguous in the absence of dynamic constraints.

The iterative search in AS is initialized with the previous frame's estimate of the model state. In the absence of feature extraction errors, and given the stability of the gradient decent algorithm used, this chaining represents a very weak form of smoothness constraint. Gradient decent should lead the the local minimum that is nearest with respect to the local Jacobians and the feature metric. In this sense, nearby minima are rejected by the algorithm. However, this smoothness constraint is not specific to the real system and is not powerful enough to keep the tracker from being distracted from the overall best estimate by feature extraction errors in a single frame. The amount that the state estimate moves toward an erroneous feature is only limited by iteration bounds placed on the search, if any, not by any explicit dynamic constraints in the system.

3.2.4 How Real Systems Address These Shortcomings

People do build AS systems. They must find ways to overcome the shortcomings in the framework discussed above. This section will discuss several systems, their contribution to AS work, and how they relate to true recursive filtering.

Gavrilla and Davis [20, 21] employ a kinematic model of the human body in their AS framework. The kinematic skeleton is augmented with a skin composed of tapered superquadrics. The subjects wear tight clothing, so the chains of tapered superquadrics are a good approximation to the external silhouette of the body. Tracking commences from a known initial configuration. Edge energy is used as the common feature space: it is extracted from the image by a filter and rendered from the model. This, combined with the model parameter search for each image, makes the system computationally very expensive. This is an example of classic AS applied to the tracking of the human body. Later improvements included an improvement in the performance of the search algorithm that avoided the inversion of Jacobian matrices that mapped motion in feature space to motion on model parameter space.

Kakadiaris and Metaxas [30] also use contour information combined with a detailed volumetric model of the human body. The system is hand initialized and tracking proceeds from that known configuration. The subjects also wear tight clothing to improve the model fidelity. Unlike the above system, this system uses 3-D deformable models to track features in the image sequence. These low-level trackers represent a further dimensionality reduction, and thus undoubtedly improve the computational performance of the system. However, the trackers do not receive assistance from the body model and are subject to failures. The system partially deals with these failures by predicting occlusions. These predictions are used to select fortuitous views from the collection of available cameras in an attempt to avoid

using data from ambiguous 2-D tracking results. Dynamics are used to make observation predictions, but these predictions are only used to reinitialize the 2-D trackers, and this is equivalent to simply initializing a search, as discussed above. The 2-D trackers will still make locally optimal decisions regardless of global model state.

Pavlović, Rehg, Cham and Murphy [36] present a system that tracks walking motion parallel to the image plane. This system also dispenses with the precise volumetric and kinematic models used above. The system uses image correlation to track parts of the body. The trackers are initialized by a generalized form of MHT that consists of many Kalman filters embedded in a Bayes network. The Kalman filters employ learned dynamic models to make predictions, and the innovations process is used to drive model switching. The two main drawbacks of this work is the bottom-up tracking and the weak body model. The image correlation tracker are reinitialized by the model for each image, but again, this is simply initializing a search. The weak, 2-D model limits the method to parallel motions, such as walking past a camera parallel to the image plane.

Delamarre and Faugeras [14] use the AS framework to track the human body. The silhouette is the common feature space. Silhouettes are tracked using active contours that are sensitive to optic flow and image intensity. The 3-D body model is a kinematic chain composed of spheres, parallelpipeds, and truncated cones. This model is projected into the image plane and the model is adjusted to match the observed silhouette. Since active contours are used, it is possible that prior information could help drive the contour tracker. This is a distinct advantage over AS implementations that use edge energy, or other rigid features.

The physical interpretation of springs attaching the model to the observed silhouette is used to provide a smooth metric for gradient decent. This physics model is not used to propagate information between frames as in the approach described here. This means that the tracker does not incorporate dynamic constraints, or any mechanism for making predictions.

Delamarre and Faugeras attempt to solve the problems of depth ambiguity and occlusion by using mutliple viewpoints. Constraints from all sillhouettes are applied to the model, and this simultaneous set of constraints is solved by gradient decent. The claim is that occlusions are resolved by having multiple viewpoints. It is not clear how many non-occluded viewpoints are required for successful tracking. Feedback from dynamic constraints could allow this system to predict through simultaneous occlusion in all cameras.

3.3 Prior Recursive Systems

Previous recursive systems have mostly relied on optical flow for tracking. Similar to regularization work, they attempt to induce a low dimensional model as a minimum mean squared error explanation of noisy flow data. They modify the regularization by utilizing the benefits of Kalman filtering: prediction of observations and corresponding revision of the state estimate given new observations. This ideally eliminates the search inherent in the regularization framework.

Pentland and Horowitz [37] demonstrate the power of this framework for tracking 3-D and $2\frac{1}{2}$-D motions on deformable and articulated objects in video. They use modal analysis to project the flow field into relatively low-dimensional translation and deformation components. In this way they take advantage of the regularization approach to reduce the dimensionality of the optic flow data, but avoid the usual search in model parameter space. These low-dimensional signals are then used by an extended Kalman filter framework to estimate the new best estimate of pose.

The mapping of optic flow data to model influence requires the ability to predict model state and project that state into the image plane to determine proximity. This requires careful initialization. Although the framework should support prediction and resolution of ambiguities such as occlusion, those issues are not addressed in the the literature.

Bregler and Malik [8] present a very interesting recursive framework for tracking of the human body similar to DYNA that includes probabilistic feature extraction influenced by dynamic models, and the possibility for tightly coupled behavioral models. The system relies on optic flow data, but tracks blobs in that data. Unfortunately the early work on tracking focused entirely on 2-D features and 2-D models. As a result the motions tracked were all 2-D motions parallel to the camera. The dynamic models were all learned from observation. This is interesting, but also limits the system because it learns models in 2-D that are not good approximations of the real system dynamics due to the unobservability of truly 3-D motions in projective 2-D imagery.

In later work [9], the system was extended to 3-D but the extension came at the expense of the interesting recursive aspects of the earlier system. The 3-D system involved a return to more classical regularization work with the addition of clever representations for rotation that make the system tractable. This system requires hand initialization and utilizes an iterative solution for flow regularization on a per-frame basis. Like most of work covered in this chapter the focus was on off-line motion capture, not real-time interface.

3.4 Summary

Many systems have been built to address the problem of visually tracking human motion, but they fall into three basics categories: analysis-synthesis, recursive systems, and the more recent particle filtering methods. Each of these methods has serious shortfalls. Analysis-synthesis methods rely on fallible feature filters and require expensive searches. The previous recursive work mostly relies on optical flow which requires significant computational power to extract. Particle filtering methods are the closest approach to a truly recursive system, but their abandonment of parametric models requires massive Monte-Carlo simulations to adequately represent the underlying probability density. No system to date has encompassed all the important aspects of DYNA and, as a result, no system can claim similar robustness combined with the computational efficiency necessary for human-computer interface. The next chapter will cover the details of the implementation of the DYNA framework.

Chapter 4

Perceptual Machinery

This chapter describes the DYNA system, a real-time, fully-dynamic, 3-D person tracking system that is able to tolerate full (temporary) occlusions of body parts, and whose performance is substantially unaffected by the presence of multiple people. The system is driven by 2-D *blob features* observed in two or more cameras [2, 54]. These features are then probabilistically integrated into a fully-dynamic 3-D skeletal model, which in turn drives the 2-D feature tracking process by setting appropriate prior probabilities.

The feedback between 3-D model and 2-D image features is a form of extended Kalman filter. One unusual aspect of our approach is that the filter directly couples raw pixel measurements with an articulated dynamic model of the human skeleton. In this aspect our system is similar to that of Dickmanns in automobile control [16], and our results show that we obtain similar advantages in efficiency and stability though this direct coupling. Previous attempts at person tracking have utilized a generic set of image features (e.g., edges, optical flow) that were computed as a preprocessing step, without consideration of the task to be accomplished. These systems are discussed in detail in Chapter 3.

We will show how this framework can go beyond passive physics of the body by incorporating various patterns of control (which we call 'behaviors') that are *learned* from observing humans while they perform various tasks. Behaviors are defined as those aspects of the motion that cannot be explained solely by passive physics or the process of image production. In the untrained tracker these manifest as significant structures in the innovations process (the sequence of prediction errors). Learned models of this structure can be used to recognize and predict this purposeful aspect of human motion.

The human body is a complex dynamic system, whose visual features are time-varying, noisy signals. Accurately tracking the state of such a system requires use of a recursive estimation framework, as illustrated in figure 4-1. The framework consists of several modules. Section 4.1 details the module labeled "2-D Vision". The module labeled "Projective Model" is described in [2] and is summarized. The formulation of our 3-D skeletal physics model, "Dynamics" in the diagram, is explained in Section 4.2, including an explanation of how to drive that model from the observed measurements. The generation of prior information for the "2-D Vision" module from the model state estimated in the "Dynamics" module is covered in Section 4.3. Section 4.4 explains the behavior system and its inti-

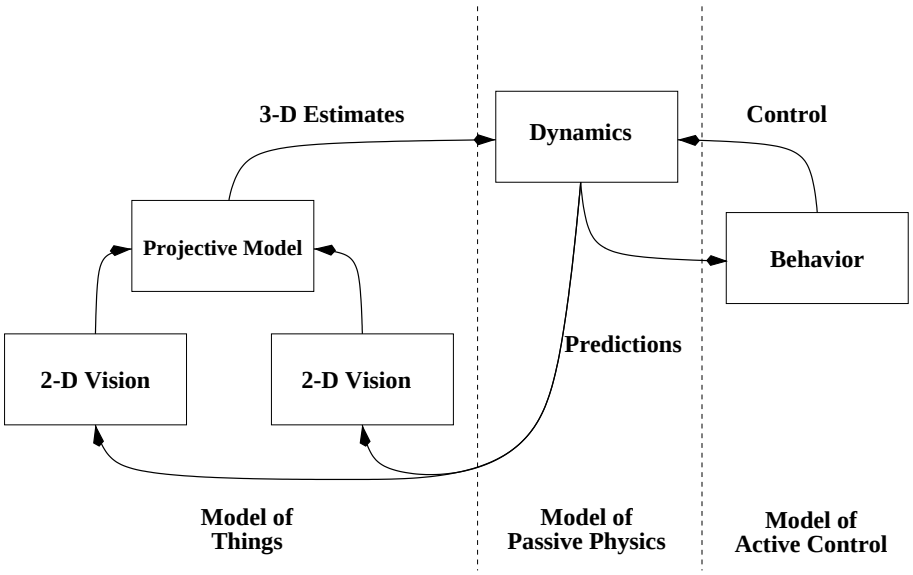

Figure 4-1: The Recursive Filtering framework. Predictive feedback from the 3-D dynamic model becomes prior knowledge for the 2-D observations process. Predicted control allows for more accurate predictive feedback.

mate relationship with the physical model. Finally, we report on experiments showing an increase in 3-D tracking accuracy, insensitivity to temporary occlusion, and the ability to handle multiple people in Section 5.1.

4.1 The Observation Model

Our system tracks regions that are visually similar, and spatially coherent: blobs. We can represent these 2-D regions by their low-order statistics. This compact model allows fast, robust classification of image regions.

Given a pair of calibrated cameras, pairs of 2-D blob parameters are used to estimate the parameters of 3-D blobs that exist behind these observations. Since the stereo estimation is occuring at the blob level instead of the pixel level, it is fast and robust.

This section describes these low-level observation and estimation processes in detail.

4.1.1 Blob Observations

Clusters of 2-D points, (i, j), have 2-D spatial means and covariance matrices, which we shall denote $\boldsymbol{\mu}_s$ and $\boldsymbol{\Sigma}_s$. The blob spatial statistics are described in terms of these second-order properties. For computational convenience we will interpret this as a Gaussian model. The Gaussian interpretation is not terribly significant, because we also keep a pixel-by-pixel *support map* showing the actual blob occupancy in the video frame [52].

Figure 4-2: A person interpreted as a set of blobs.

The visual appearance of the pixels, (y, u, v), that comprise a blob can also be modeled by second order statistics in color space: the 3-D mean, $\boldsymbol{\mu}_c$ and covariance, $\boldsymbol{\Sigma}_c$. As with the spatial statistics, these chromatic statistics are interpreted as the parameters of a Gaussian distribution in color space. We chose the YUV representation of color due to its ready availability from video digitization hardware and the fact that in the presence of white luminants it confines much of the effect of shadows to the single coordinate y [52].

Given these two sets of statistics describing the blob, the overall blob description becomes the concatenation $(ijyuv)$, where the overall mean is:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_s \\ \boldsymbol{\mu}_c \end{bmatrix}$$

and the overall covariance is:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_s & \boldsymbol{\Lambda}_{sc} \\ \boldsymbol{\Lambda}_{cs} & \boldsymbol{\Sigma}_c \end{bmatrix}$$

This framework allows for the concatenation of additional statistics that may be available from image analysis, such as texture or motion components. Figure 4-2 shows a person represented as a set of blobs. Spatial mean and covariance is represented by the iso-probability contour ellipse shape. The color mean is represented by the color of the blob. The color covariance is not represented in this illustration.

4.1.2 Frame Interpretation

To compute $\Pr(\mathbf{O}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, the probability that a given pixel observation, $\mathbf{O}$, is a member of a given blob, k, we employ the Gaussian assumption to arrive at the formula:

$$\Pr(\mathbf{O}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{\exp(-\frac{1}{2}(\mathbf{O} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{O} - \boldsymbol{\mu}_k))}{(2\pi)^{\frac{m}{2}} |\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \tag{4.1}$$

where $\mathbf{O}$ is the concatenation of the pixel spatial and chromatic characteristics. Since the color and spatial statistics are assumed to be independent, the cross-covariance $\boldsymbol{\Lambda}_{sc}$ goes to zero, and the computation of the above value can proceed in a separable fashion [52].

For a frame of video data, the pixel at (i, j) can be classified by selecting, from the k blobs being tracked, that blob which best predicts the observed pixel:

$$\Gamma_{ij} = \arg\max_k [\Pr(\mathbf{O}_{ij}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)] \tag{4.2}$$

where Γ_{ij} is the labeling of pixel (i, j). Due to the connected nature of people, it is possible to increase efficiency by growing out from initial position estimates to the outer edge of the figure. This allows the algorithm to only touch the pixels that represent the person and those nearby [52].

4.1.3 Model Update

Once the pixels are labeled by Γ, blob statistics can be reestimated from the image data. For each class k, the pixels marked as members of the class are used to estimate the new model mean $\boldsymbol{\mu}_k$:

$$\hat{\boldsymbol{\mu}}_k = E[\boldsymbol{O}] \tag{4.3}$$

and the second-order statistics become the estimate of the model's covariance matrix $\boldsymbol{\Sigma}_k$,

$$\hat{\boldsymbol{\Sigma}}_k = E[(\boldsymbol{O} - \boldsymbol{\mu}_k)(\boldsymbol{O} - \boldsymbol{\mu}_k)^T] \tag{4.4}$$

4.1.4 A Compact Model

These updated blob statistics represent a low-dimensional, object-based description of the video frame. The position of the blob is specified by the two paramaters of the distribution mean vector $\boldsymbol{\mu}_s$: i and j. The spatial extent of each blob is represented by the three free parameters in the covariance matrix $\boldsymbol{\Sigma}_s$. A natural interpretation of these parameters can be obtained by performing the eigenvalue decomposition of $\boldsymbol{\Sigma}_s$:

$$\boldsymbol{\Sigma}_s \begin{bmatrix} | & | \\ L_1 & L_2 \\ | & | \end{bmatrix} = \begin{bmatrix} | & | \\ L_1 & L_2 \\ | & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{4.5}$$

Without loss of generality, $\lambda_1 \geq \lambda_2$, and $\|L_1\| = \|L_2\| = 1$. With those constraints, λ_1 and

Figure 4-3: The hand as an iso-probability ellipse.

λ_2 represent the length of the semi-major and semi-minor axes of the iso-probability contour ellipse defined by $\mathbf{\Sigma}_s$. The vectors L_1 and L_2 specify the direction of these axes. Since they are perpendicular, they can be specified by a single parameter, say ω, the rotation of the semi-major axis away from the x axis. Thus we can represent the model $\{\boldsymbol{\mu}_s \mathbf{\Sigma}_s\}$ with five parameters:

$$\{i, j, \lambda_1, \lambda_2, \omega\}$$

Furthermore, these parameters have the convenient physical interpretation of representing the center, length, width, and orientation of an ellipse in the image plane, as shown in Figure 4-3.

Since the typical blob is supported by tens to hundreds of pixels, it is possible to robustly estimate these five parameters from the available data. The result is a stable, compact, object-level representation of the image region explained by the blob.

4.1.5 Recovery of a Three Dimensional Model

These 2-D features are the input to the 3-D blob estimation framework used by Azarbayejani and Pentland [2]. This framework relates the 2-D distribution of pixel values to a tracked object's 3-D position and orientation.

Inside the larger recursive framework, this estimation is carried out by an embedded extended Kalman filter. It is the structure from motion estimation framework developed by

Figure 4-4: The hand as a 3-D blobject.

Azarbayejani to estimate 3-D geometry from images. As an extended Kalman filter, it is itself a recursive, nonlinear, probabilistic estimation framework. Estimation of 3-D parameters from calibrated sets of 2-D parameters is computationally very efficient, requiring only a small fraction of computational power as compared to the segmentation algorithms described above[3]. The reader should not be confused by the embedding of one recursive framework inside another: for the larger context this module may be considered an opaque filter.

The same estimation machinery used to recover these 3-D blobs can also be used to quickly and automatically calibrate pairs of cameras from blob observation data[2].

After this estimation process, the 3-D blob has nine parameters:

$$\{x, y, z, l_1, l_2, l_3, w_1, w_2, w_3\}$$

As above, these parameters represent the position of the center of the blob, the length of the fundamental axes defined by an iso-probability contour, and the rotation of these axes away from some reference frame. Figure 4-4 shows the result of this estimation process.

4.2 Modeling Dynamics

There is a wide variety of ways to model physical systems. The model needs to include parameters that describe the *links* that compose the system, as well as information about the hard *constraints* that connect these links to one another. A model that only includes this information is called a *kinematic* model, and can only describe the static states of a system. The state vector of a kinematic model consists of the model state, $\mathbf{x}$, and the model parameters, $\mathbf{p}$, where the parameters, $\mathbf{p}$ are the unchanging qualities of the system (the length or mass of a given link, for example).

A system in motion is more completely modeled when the *dynamics* of the system are modeled as well. A dynamic model describes the state evolution of the system over time. In a dynamic model the state vector includes velocity as well as position: $\mathbf{x}, \dot{\mathbf{x}}$, and the model parameters, $\mathbf{p}$. The state evolves according to Newton's First Law:

$$\ddot{\mathbf{x}} = \mathbf{W} \cdot \mathbf{X} \tag{4.6}$$

where $\mathbf{X}$ is the vector of external forces applied to the system, and $\mathbf{W}$ is the inverse of the system mass matrix. The mass matrix describes the distribution of mass in the system.

4.2.1 Hard Constraints

Hard constraints represent absolute limitations imposed on the system. One example is the kinematic constraint of a skeletal joint. The model follows the *virtual work* formulation of Witkin[51]. The Witkin formulation has several advantages over reduced dimensionality solutions such as that described by Featherstone[17]: the constraints can be modified at run-time, and the modularity inherent in the mathematics drastically simplifies the implementation. The one significant disadvantage, which will be addressed below, is computational efficiency.

In a virtual work constraint formulation, all the links in a model have full range of unconstrained motion. Hard kinematic constraints on the system are enforced by a special set of forces $\mathbf{C}$:

$$\ddot{\mathbf{x}} = \mathbf{W} \cdot (\mathbf{X} + \mathbf{C}) \tag{4.7}$$

The constraints are specified as mathematical relationships between objects that are defined to be zero when the constraint is satisfied. The constraint forces C, are chosen to insure that the constraints stay satisfied. In Figure 4-5 the constraint would be expressed as the distance between the line and the center of the object.

The constraints are functions of model state and time: $\mathbf{c}(\mathbf{x}, t)$. When a constraint is satisfied then $\mathbf{c} = 0$. If a constraint is to remain satisfied, then the constraint velocity must remain zero, $\dot{\mathbf{c}} = 0$, and the constraint must not accelerate away from a valid state: $\ddot{\mathbf{c}} = 0$ or the system would soon be in an invalid state. The constraint forces from Equation 4.7 and Figure 4-5 keep the system from accelerating away from constraint satisfaction. The road

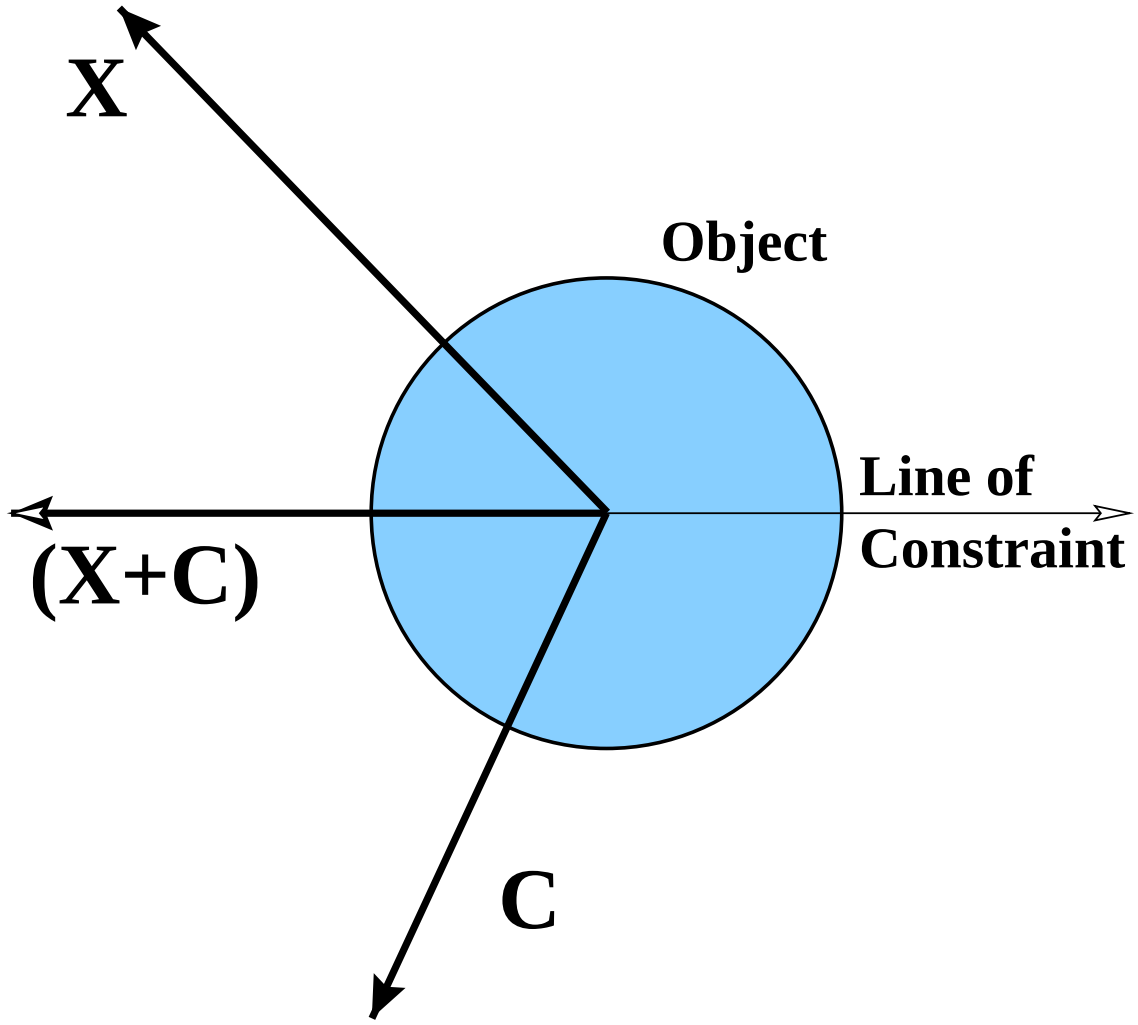

Figure 4-5: The 2-D object is constrained to move on the indicated line. An external force, **X**, is applied to the object. The constraint force, **C**, keeps the object from accelerating away from the constraint.

to understanding these forces begins by differentiating **c**:

$$\dot{\mathbf{c}} = \frac{\partial}{\partial \mathbf{x}}\mathbf{c}(\mathbf{x}, t) = \frac{\partial \mathbf{c}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial \mathbf{c}}{\partial t} \tag{4.8}$$

and again:

$$\ddot{\mathbf{c}} = \frac{\partial \mathbf{c}}{\partial \mathbf{x}}\ddot{\mathbf{x}} + \frac{\partial \dot{\mathbf{c}}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial^2 \mathbf{c}}{\partial t^2} \tag{4.9}$$

Combining with Equation 4.7 and setting $\ddot{\mathbf{c}} = 0$ yields a relationship between the model state, the applied external forces, and the constraint Jacobian, where the constraint restitution force **C** is the only unknown:

$$\frac{\partial \mathbf{c}}{\partial \mathbf{x}}\mathbf{W} \cdot (\mathbf{X} + \mathbf{C}) + \frac{\partial \dot{\mathbf{c}}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial^2 \mathbf{c}}{\partial t^2} = 0 \tag{4.10}$$

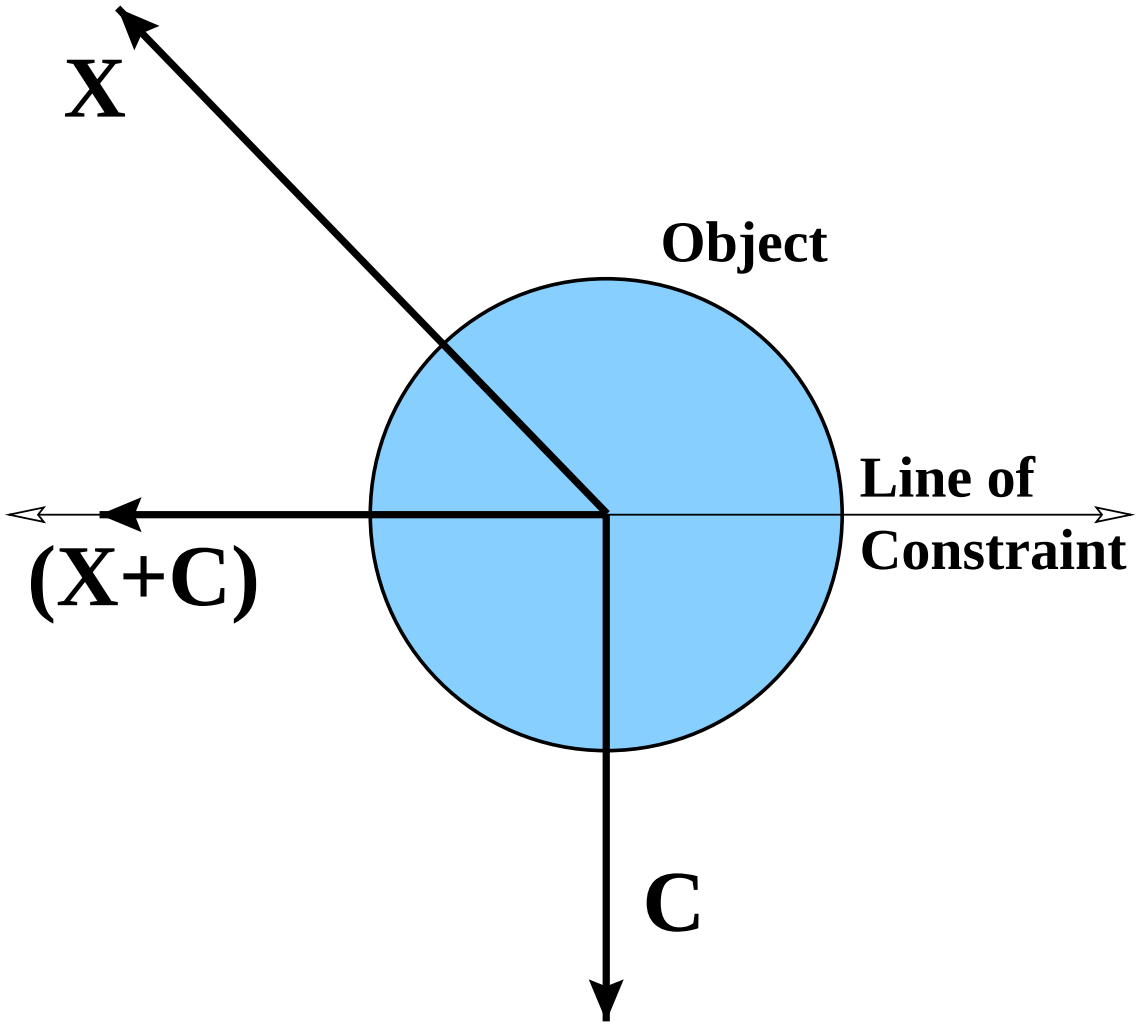

Figure 4-6: No work is performed if the constraint force lies in the null space compliment of the constraint Jacobian.

The force in Figure 4-5 satisfies the relationship in Equation 4.10, as do many other possible vectors. Equation 4.10 is an under-determined system since the dimensionality of $\mathbf{c}$ will always be less than the dimensionality of $\mathbf{x}$, or the system would be fully constrained and wouldn't move at all. For example, in Figure 4-5, $\mathbf{c}$ is the distance between the center of the object and the line, a one dimensional value, while $\mathbf{x}$ is two dimensional, three if the object is allowed rotational freedom in the plane.

One problem with that choice of $\mathbf{C}$ in Figure 4-5 is that it will add energy to the system. Equation 4.10 only specifies that the force exactly counteract the component of $\mathbf{X}$ that is in violation of the constraints. Since constraints that add energy to the models will lead to instability, an additional requirement that the force $\mathbf{C}$ do no work on the system is employed, where work would be $\mathbf{C}\dot{\mathbf{x}}$. If $\mathbf{C}$ is always applied perpendicular to any direction of motion allowed by the constraint then this will be satisfied. In the example case of the object on the line, this means that $\mathbf{C}$ must be perpendicular to the line of constraint, as shown in Figure 4-6.

More generally, the valid displacements for the system are described by the null space of the constraint Jacobian:

$$\frac{\partial \mathbf{c}}{\partial \mathbf{x}} d\mathbf{x} = 0 \tag{4.11}$$

since valid displacements are required to leave $\mathbf{c} = 0$. The disallowed displacements are ones that do change $\mathbf{c}$:

$$d\mathbf{x} = \boldsymbol{\lambda} \frac{\partial \mathbf{c}}{\partial \mathbf{x}} \tag{4.12}$$

So, to do no work, the constraint force $\mathbf{C}$ is required to lie in the same subspace:

$$\mathbf{C} = \boldsymbol{\lambda} \frac{\partial \mathbf{c}}{\partial \mathbf{x}} \tag{4.13}$$

Combining that equation with Equation 4.10 results in a linear system of equations with only the one unknown, $\boldsymbol{\lambda}$:

$$-\left[\frac{\partial \mathbf{c}}{\partial \mathbf{x}} \mathbf{W} \frac{\partial \mathbf{c}}{\partial \mathbf{x}}^T\right] \boldsymbol{\lambda} = \frac{\partial \mathbf{c}}{\partial \mathbf{x}} \mathbf{W}\mathbf{X} + \frac{\partial \dot{\mathbf{c}}}{\partial \mathbf{x}} \dot{\mathbf{x}} + \frac{\partial^2 \mathbf{c}}{\partial t^2} \tag{4.14}$$

This equation can be rewritten to emphasize its linear nature. $\mathbf{J}$ is the constraint Jacobian, $\boldsymbol{\rho}$ is a known constant vector, and $\boldsymbol{\lambda}$ is the vector of unknown Lagrange multipliers:

$$-\mathbf{J}\mathbf{W}\mathbf{J}^T \boldsymbol{\lambda} = \boldsymbol{\rho} \tag{4.15}$$

To obtain $\mathbf{C}$, $\boldsymbol{\lambda}$ is substituted back into Equation 4.13. Many fast, stable methods exist for solving equations of this form.

All the components of Equation 4.14 that relate directly to the constraints are linearizations, so they must be recomputed at each integration step. This provides the opportunity to create and delete constraints at run-time simply by modifying the calculation of the constraint Jacobian.

Modularization

Constraints are written as mathematical relationships between points in space. Often these points are located at some arbitrary location in the local coordinate space of some object. Implementing constraints to understand each type of object, possibly having different internal representations for state, would make the constraints unnecessarily complex. Witkin suggests inserting an abstraction layer between objects and constraints, called connectors[51]. Thus $\mathbf{c}$ for a constraint between two objects becomes:

$$\mathbf{c}(\mathbf{x}) = f(\mathbf{a}(\mathbf{x}_1), \mathbf{b}(\mathbf{x}_2)) \tag{4.16}$$

The constraint Jacobian can then be decomposed by the chain rule:

$$\frac{\partial \mathbf{c}}{\partial \mathbf{x}} = \frac{\partial \mathbf{c}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{x}_1} + \frac{\partial \mathbf{c}}{\partial \mathbf{b}} \frac{\partial \mathbf{b}}{\partial \mathbf{x}_2} \tag{4.17}$$

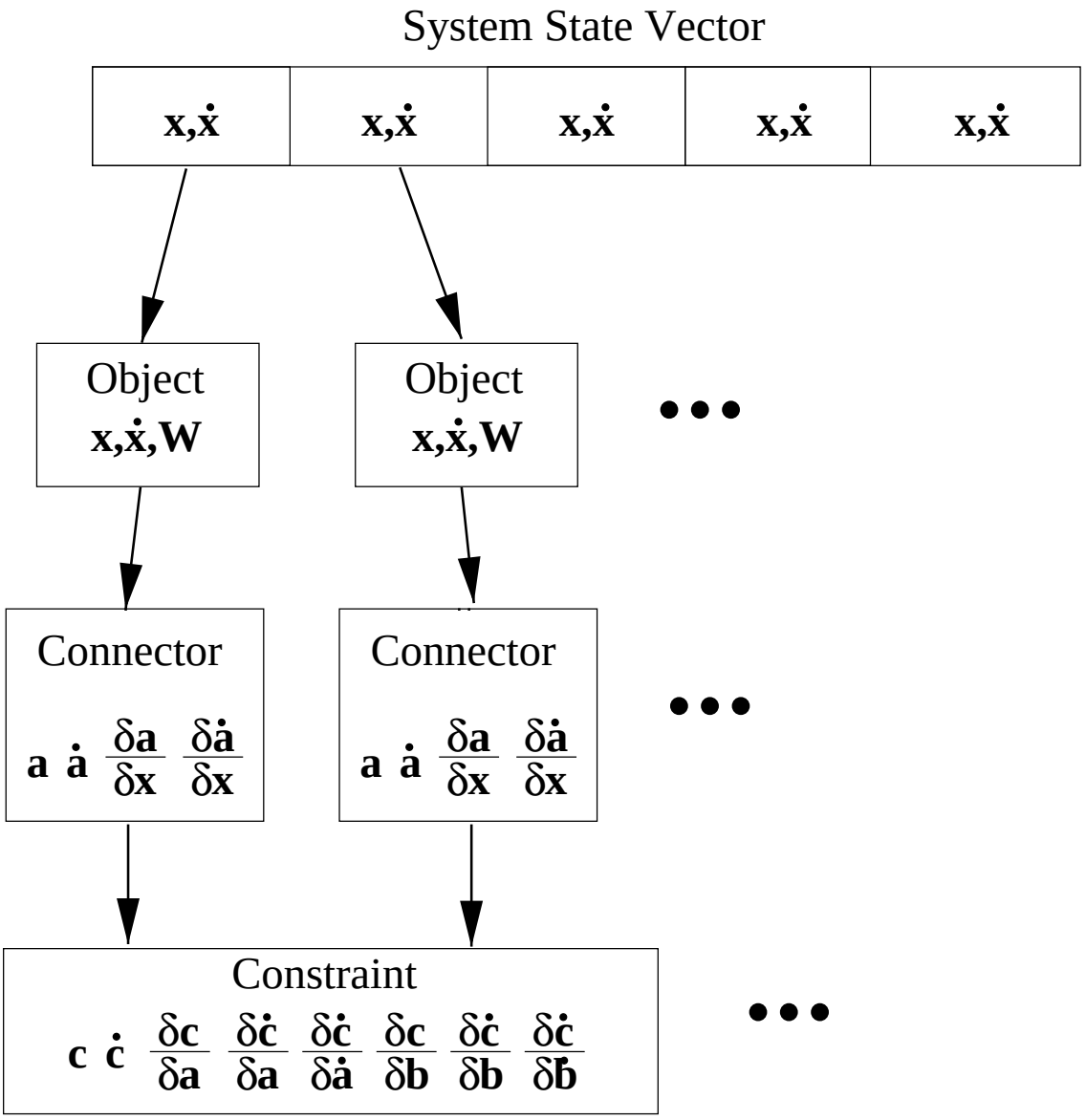

Figure 4-7: The constraint system is made up of software modules that cooperate to construct Equation 4.14. Connectors are an abstraction that allows the Constraints to be more general, and hence more reusable.

The constraint module can then compute $\frac{\partial \mathbf{c}}{\partial \mathbf{a}}$ and $\frac{\partial \mathbf{c}}{\partial \mathbf{b}}$ without regard to the underlying implementation while the connectors are responsible for calculating $\frac{\partial \mathbf{a}}{\partial \mathbf{x}}$. The constraint velocity Jacobian can be computed in the same way;

$$\frac{\partial \dot{\mathbf{c}}}{\partial \mathbf{x}} = \frac{\partial \dot{\mathbf{c}}}{\partial \mathbf{a}}\frac{\partial \mathbf{a}}{\partial \mathbf{x}_1} + \frac{\partial \dot{\mathbf{c}}}{\partial \dot{\mathbf{a}}}\frac{\partial \dot{\mathbf{a}}}{\partial \mathbf{x}_1} + \frac{\partial \dot{\mathbf{c}}}{\partial \mathbf{b}}\frac{\partial \mathbf{b}}{\partial \mathbf{x}_2} + \frac{\partial \dot{\mathbf{c}}}{\partial \dot{\mathbf{b}}}\frac{\partial \dot{\mathbf{b}}}{\partial \mathbf{x}_2} \tag{4.18}$$

Figure 4-7 shows how this information moves through the system. Each block represents a different software module. This abstraction is very powerful: in the system the same constraint code implements pin joints in 2-D models and ball joints in 3-D models. Because constraints involving rotational motion are somewhat more complex, the system differentiates between connectors without orientation, called Points, and connectors with orientation, called Handles.

Multiple Objects and Constraints

Systems with only a single constraint are rather limiting. Multiple objects and constraints fit easily into the framework. For multiple objects, the state vector $\mathbf{x}$ becomes the concatenation of all the individual object state vectors. So in a 3-D model where every object has 6 degrees of freedom, with 5 objects the state vector would have dimensionality 30.

The mass matrix is similarly the concatenation of the individual mass matrices. Assuming static geometry for each object, the individual mass matrix is constant in the object local coordinate system. This mass matrix is transformed to global coordinates and added as a block to the global mass matrix. Since the global mass matrix is block diagonal, the inverse mass matrix is simply the concatenation of the individually inverted mass matrices, and so doesn't take an inordinate amount of time to compute.

Objects are enumerated with the order that they contribute to the global state vector. Constraints are similarly enumerated. A constraint between two objects contributes two blocks to the constraint Jacobian. The constraints appear on the row according to the constraint's enumeration and the columns associated with the constrained objects. The structure of the constraint Jacobian is illustrated in Figure 4-8 for a model of the upper body with five links: torso, left upper arm, left lower arm, right upper arm, and right lower arm. The other values in Equation 4.14 are constructed in a similar fashion.

The global inverse mass matrix is block diagonal and the global constraint Jacobian is block sparse. Both are large. Solving Equation 4.14 for $\boldsymbol{\lambda}$ requires sparse matrix methods to be accomplished efficiently. Sparse matrix methods were used to construct $\mathbf{JWJ}^T$. An implementation of Linear Biconjugate Gradient Descent for sparse matrices was used to solve the resulting linear system. The algorithms were taken from Numerical Recipes[39]. These improvements made the constraint system tractable on contemporary hardware. The rest of the matrix manipulations are handled with a basic C++ matrix library.

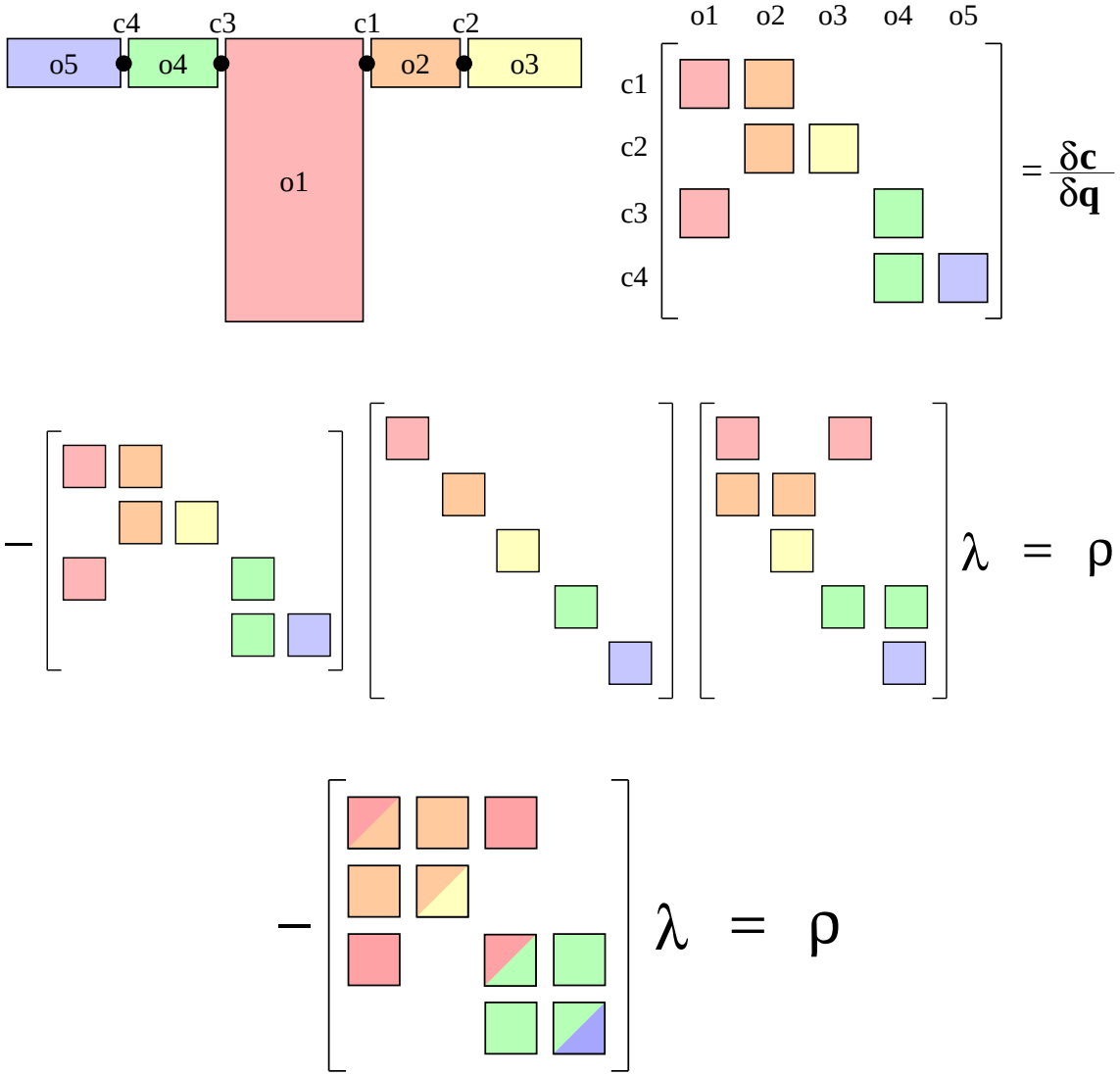

Figure 4-8: **Top:** the individual constraint Jacobians each contribute one block per object that they affect to the global constraint Jacobian. **Middle:** each object also contributes to the block-diagonal inverse mass matrix from Equation 4.15. **Bottom:** Sparsely connected systems result in a block-sparse linear system.

Discretization Error

The constraints of Equation 4.14 are only true instantaneously. When the equations are solved at discrete time steps then errors are introduced and the system drifts away from the manifold of valid states. A restoring force is used to keep the system from accumulating errors over time:

$$\ddot{\mathbf{x}} = \mathbf{W} \cdot (\mathbf{X} + \mathbf{C} + \mathbf{F}) \tag{4.19}$$

Where $\mathbf{F}$ is determined by the relationship:

$$F = \alpha \mathbf{c} \frac{\partial \mathbf{c}}{\partial \mathbf{x}} + \beta \dot{\mathbf{c}} \frac{\partial \mathbf{c}}{\partial \mathbf{x}} \tag{4.20}$$

This applies a restoring force in the constrained direction that brings the system back toward the nearest valid state and a damping force that reduces illegal velocity. The parameters α and β are fixed. In practice the selection of these parameters has very little impact on model stability since deviations from constraints remain small. A typical value for α is $1000\frac{N}{m}$ and a typical value for β is $4\frac{Ns}{m}$.

Distributed Integration

Once the global forces are projected back into the allowable subspace and corrected for discretization error, all further computation is partitioned among the individual objects. This avoids computing the very large global version of Equation 4.7. This is possible since the inverse mass matrix $\mathbf{W}$ is block diagonal, so once the global value for $\mathbf{C}$ is determined, Equation 4.7 breaks down into a set of independent systems. This distributed force application and integration also provides the opportunity for objects to transform the applied forces to the local frame and to deal with forces and torques separately. This simplifies the implementation of the dynamics subsystem significantly, since each link is treated as a free six degree of freedom body.

4.2.2 Soft Constraints

Some constraints are probabilistic in nature. Noisy image measurements are a constraint of this sort, they influence the dynamic model but do not impose hard constraints on its behavior. As a result, the absolute constraint satisfaction described in the previous section is not appropriate.

Soft constraints are more appropriately expressed as a potential field acting on the dynamic system. The addition of a potential field function to model a probability density function pushes the model toward the most likely value. In general a soft constraint might be any function:

$$\mathbf{X}_{\text{soft}} = f_{\text{soft}}(\mathbf{S}, \mathbf{x}, \dot{\mathbf{x}}, \mathbf{p}) \tag{4.21}$$

where $\mathbf{S}$ is some parameterization over the family of potential fields specified by $f(\cdot)$.

The simplest function is the constant potential. Gravity is well-modeled by a constant field

over the scales of the model. So the potential field is simply:

$$\mathbf{X}_{\text{soft}} = mg \tag{4.22}$$

where g is acceleration due to gravity, and m is the mass of the link affected by $\mathbf{X}_{\text{soft}}$.

A soft constraint that attracts a body part to a specific location is somewhat more complex:

$$\mathbf{X}_{\text{soft}} = k(\mathbf{x}_0 - \mathbf{x}) \tag{4.23}$$

where $\mathbf{x}_0$ is the desired position and k is a constant multiplier that affect the "softness" of the constraint. Care must be taken when choosing k to avoid introducing instabilities into the model. Values of k that are too large start to turn the soft constraint into something more like a hard constraint. In this case the constraint would be better modeled by the techniques described above.

It is also possible to construct anisotropic constraints

$$\mathbf{X}_{\text{soft}} = \frac{(\mathbf{x} - \mathbf{x}_0)}{\|(\mathbf{x} - \mathbf{x}_0)\|}(\mathbf{x} - \mathbf{x}_0)K^{-1}(\mathbf{x} - \mathbf{x}_0) \tag{4.24}$$

where K is a shaping matrix that determines the weighting of various directions. This allows soft constraints to have stronger influence in a particular direction. This is useful for modelling the influence of the blob observations discussed above, or any other regular, non-isotropic force field.

Note that functions may be arbitrarily complex. A good example is a controller of the form described in Section 4.4. Despite their complexity, the dynamics engine may represent them as a time-varying potential field. The forces applied by the controller simply become another force affecting the dynamic evolution of the model. The neuroscience literature supports this model[43].

4.2.3 Observation Influence

The 3-D observations described in Section 4.1 supply constraints on the underlying 3-D human model. Due to their statistical nature, observations are easily modeled as soft constraints. Observations are integrated into the dynamic evolution of the system by describing them with potential fields, as discussed in Section 4.2.2. These potential fields apply forces to the body model, causing it evolve over time toward the observations. The strength of these fields is related to the Kalman gain in a classic Kalman filter.

4.3 The Inverse Observation Model

In the open-loop system, the vision system uses a Maximum Likelihood framework to label individual pixels in the scene (Equation 4.2). To close the loop, we need to incorporate information from the 3-D model. This means generating 2-D statistical models from the 3-D body model that can be utilized by the vision system to improve its decisions.

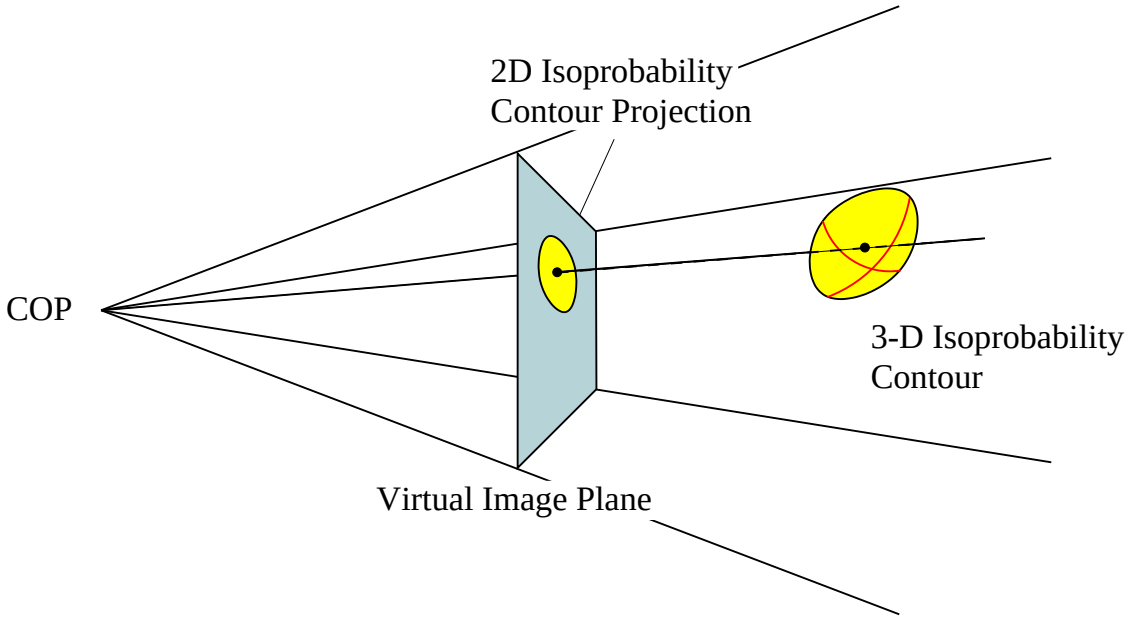

Figure 4-9: Projecting a 3-D blob into a 2-D image plane.

The current state of the model $(\mathbf{x}_t, \dot{\mathbf{x}}_t)$ specifies the best estimate of the configuration of the body in model space given past observations. The first step in predicting future observations is to propagate the state forward according to the dynamic constraints described above. The external forces acting on the system can be assumed to be constant for the period of forward prediction (33ms in the case of video rate observations), or can be predicted forward in time by behavior models as described below in Section 4.4.

Once a best estimate of the future configuration of the system, $\mathbf{x}_{t+\Delta}$, has been computed, the next step is to generate a set of hypothetical 3-D observations that we would expect to see given that configuration. This involves generating distributions that represent 3-D ellipsoids that fit the observable portions of the model links. These distributions are transformed into the camera reference frame as described in Section 4.1.5. In this frame they are described as in Section 4.1 by either their second order statistics:

$$\{\boldsymbol{\mu}_k^\circ, \boldsymbol{\Sigma}_k^\circ\}$$

or, by their free parameters:

$$\{x, y, z, l_1, l_2, l_3, w_1, w_2, w_3\}$$

The process of identifying the observable portions of these links is discussed in Section 4.2.3.

To be used as a prior for the classification decision in Equation 4.2, these 3-D distributions must be rendered into 2-D image coordinates using perspective projection. These projected distribution will be described by their second order statistics:

$$\{\boldsymbol{\mu}_k^\star, \boldsymbol{\Sigma}_k^\star\}$$

or alternatively by the free parameters:

$$\{i, j, \lambda_1, \lambda_2, \omega\}$$

The computation of $\boldsymbol{\mu}_k^\star$ from $\boldsymbol{\mu}_k^\circ$ is a straightforward application of the forward projective camera model. The parameters x, y, z map into the parameters i, j by perspective projection:

$$\boldsymbol{\mu}_k^\star = \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \frac{1}{1+z} \tag{4.25}$$

The true perspective projection of a 3-D Gaussian distribution over $\Re^3$ is not a Gaussian distribution over the image coordinates $\Re^2$. It is necessary to employ an approximation to perspective projection that will yield a Gaussian distribution in image coordinates to obtain a value for $\boldsymbol{\Sigma}_k^\star$

Orthographic projection of a Gaussian distribution does result in a Gaussian distribution. This process involves integrating over the Gaussian in the direction of projection. Orthographic projection of the 3-D prior onto a XY plane passing through the mean is thus equivalent to taking the marginal of a zero-mean Gaussian distribution:

$$\mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \sigma_x & \lambda_{xy} \\ \lambda_{yx} & \sigma_y \end{bmatrix}\right) = \int_{-\infty}^{\infty} \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \sigma_x & \lambda_{xy} & \lambda_{xz} \\ \lambda_{yx} & \sigma_y & \lambda_{yz} \\ \lambda_{zx} & \lambda_{zy} & \sigma_z \end{bmatrix}\right) \partial z \tag{4.26}$$

Orthographic projection does not account for the scaling effect of perspective projection, so simply using orthographic projection would result in priors with significantly exaggerated covariances. A solution is to use the scaled orthographic approximation to perspective projection. Scaled orthographic projection uses perspective projection to map an intermediate 2-D orthographic projection into the virtual image plane. Since the plane of orthographic projection is parallel to the virtual image plane, this operation is equivalent to a scale. Scaling a Gaussian distribution retains the Gaussian nature, so we have the approximation we need. As illustrated in Figure 4-10, by placing the plane of orthographic projection at z, we can compute the 2-D blob covariance prior, $\boldsymbol{\Sigma}_k^\star$, from the 3-D covariance $\boldsymbol{\Sigma}_k^\circ$:

$$\boldsymbol{\Sigma}_k^\star = \begin{bmatrix} \sigma_i & \lambda_{ij} \\ \lambda_{ji} & \sigma_j \end{bmatrix} = \begin{bmatrix} \frac{1}{1+z} & 0 \\ 0 & \frac{1}{1+z} \end{bmatrix} \begin{bmatrix} \sigma_x & \lambda_{xy} \\ \lambda_{yx} & \sigma_y \end{bmatrix} \begin{bmatrix} \frac{1}{1+z} & 0 \\ 0 & \frac{1}{1+z} \end{bmatrix} \tag{4.27}$$

The result of this process is a prior distribution on image observations in the next frame:

$$\Pr(\mathbf{O}_{ij} | \boldsymbol{\mu}_k^\star, \boldsymbol{\Sigma}_k^\star) \tag{4.28}$$

Integrating this information into the 2-D statistical decision framework of Equation 4.2 results in a Maximum *A Posteriori* decision rule for pixel classification:

$$\Gamma_{ij} = \arg\max_k \left[\Pr(\mathbf{O}_{ij} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^\alpha \cdot \Pr(\mathbf{O}_{ij} | \boldsymbol{\mu}_k^\star, \boldsymbol{\Sigma}_k^\star)^{1-\alpha} \right] \tag{4.29}$$

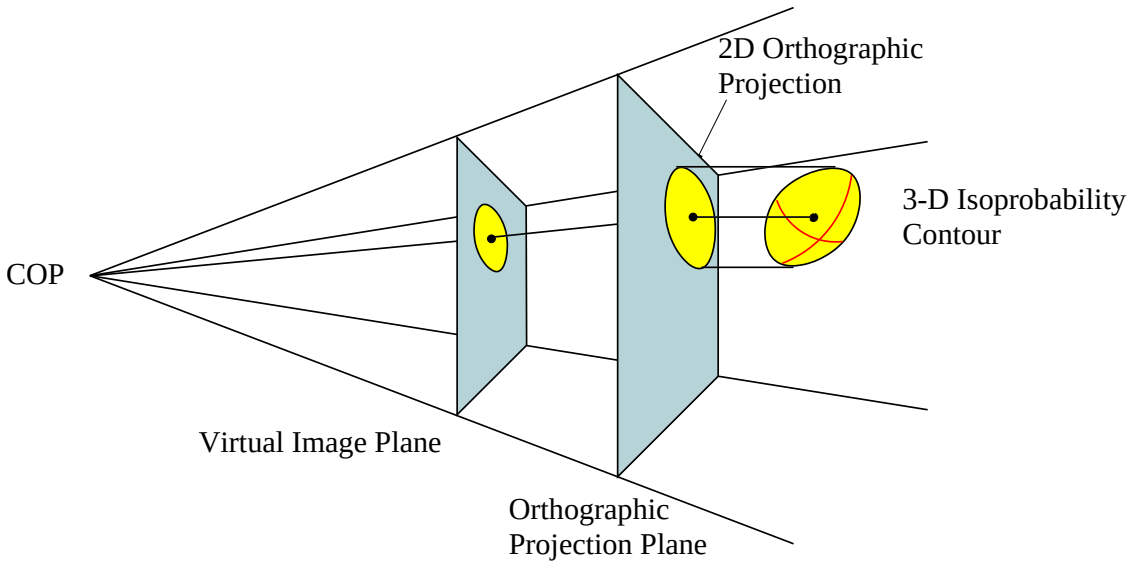

Figure 4-10: Scaled-Orthographic projection approximation for $\mathbf{\Sigma}_k^\star$

where α is the weighting parameter that indicates the importance of the prior information:

$$0 \leq \alpha \leq 1 \tag{4.30}$$

4.4 A Model for Control

Observations of the human body reveal an interplay between the passive evolution of a physical system (the human body) and the influences of an active, complex controller (the nervous system). Section 4.2 explains how, with a bit of work, it is possible to model the physical aspects of the system. However, it is *very* difficult to explicitly model the human nervous and muscular systems, so the approach of using observed data to estimate probability distributions over control space is very appealing.

4.4.1 A Model for Control

Kalman filtering includes the concept of an *innovations process*. This is the difference between the actual observation and the predicted observation transformed by the Kalman gain:

$$\boldsymbol{\nu}_t = \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}_t\mathbf{\Phi}_t\hat{\mathbf{x}}_{t-1}) \tag{4.31}$$

The innovations process $\boldsymbol{\nu}$ is the sequence of information in the observations that was not adequately predicted by the model. If we have a sufficient model of the observed dynamic process, and white, zero-mean Gaussian noise is added to the system, either in observation or in the real dynamic system itself, then the innovations process will be white. Inadequate models will cause correlations in the innovations process.

Since purposeful human motion is not well modeled by passive physics, we should expect

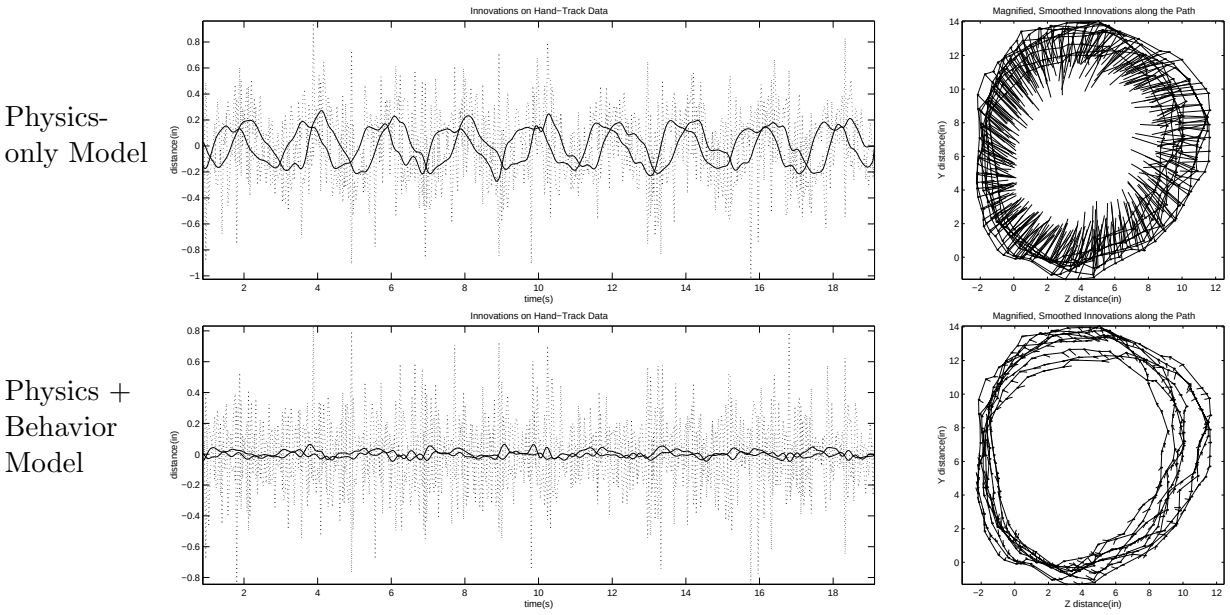

Figure 4-11: Modeling tracking data of circular hand motion. Passive physics alone leaves significant structure in the innovations process. **Top Left:** Smoothing the innovations reveals unexplained structure. **Top Right:** Plotting the Innovations along the path makes the purposeful aspect of the action clear. **Bottom:** In this example, using a learned control model to improve predictions leaves only white process noise in the innovations process. The smoothed innovations stay near zero.

significant structure in the innovations process.

A simple example is helpful for illustrating this idea. If we track the hand moving in a circular motion, then we have a sequence of observations of hand position. This sequence is the result of a physical thing being measured by a noisy observation process. For this simple example we making the assumption that the hand moves according to a linear, constant velocity dynamic model. Given that assumption, it is possible to estimate the true state of the hand, and predict future states and observations. If this model is sufficient, then the errors in the predictions should be solely due to the noise in the system.

The upper plots in Figure 4-11 show that model is not sufficient. Smoothing $\boldsymbol{\nu}$ reveals this significant structure (top left). Plotting the innovations along the path of observations makes the relationship between the observations and the innovations clear: there is some un-modeled process acting to keep the hand moving in a circular motion (top right). This un-modeled process is the purposeful control signal that being applied to the hand by the muscles.

In this example, there is one active, cyclo-stationary control behavior, and its relationship to the state of the physical system is straightforward. There is a one-to-one mapping between the state and the phase offset into the cyclic control, and a one-to-one mapping between the offset and the control to be applied. If we use the smoothed innovations as our model

and assume a linear control model of identity, then the linear prediction becomes:

$$\hat{\mathbf{x}}_t = \mathbf{\Phi}_t \hat{\mathbf{x}}_{t-1} + \mathbf{I}\mathbf{u}_{t-1} \tag{4.32}$$

where $\mathbf{u}_{t-1}$ is the control signal applied to the system. The lower plots in Figure 4-11 show the result of modeling the hand motion with a model of passive physics and a model of the active control. The smoothed innovations are basically zero: there is no part of the signal that deviates from our model except for the observation noise.

In this simple, linear example the system state, and thus the innovations, are represented the same coordinate system as the observations. With more complex dynamic and observations models, such as described in Section 4.2, they could be represented in any arbitrary system, including spaces related to observation space in non-linear ways, for example as joint angles.

The next section examines a more powerful form of model for control.

4.4.2 Multiple Behavior Models

Human behavior, in all but the simplest tasks, is not as simple as a single dynamic model. The next most complex model of human behavior is to have *several* alternative models of the person's dynamics, one for each class of response. Then at each instant we can make observations of the person's state, decide which model applies, and then use that model for estimation. This is known as the *multiple model* or *generalized likelihood* approach, and produces a generalized maximum likelihood estimate of the current and future values of the state variables [48]. Moreover, the cost of the Kalman filter calculations is sufficiently small to make the approach quite practical.

Intuitively, this solution breaks the person's overall behavior down into several "prototypical" behaviors. For instance, we might have dynamic models corresponding to a relaxed state, a very stiff state, and so forth. We then classify the behavior by determining which model best fits the observations. This is similar to the multiple model approach of Friedmann, and Isard[19, 26].

Since the innovations process is the part of the observation data that is unexplained by the dynamic model, the behavior model that explains the largest portion of the observations is, of course, the model most likely to be correct. Thus, at each time step, we calculate the probability $Pr^{(i)}$ of the m-dimensional observations $\mathbf{Y}_k$ given the i^{th} model and choose the model with the largest probability. This model is then used to estimate the current value of the state variables, to predict their future values, and to choose among alternative responses.

4.4.3 Hidden Markov Models of Control

Since human motion evolves over time, in a complex way, it is advantageous to explicitly model temporal dependence and internal states in the control process. A Hidden Markov Model (HMM) is one way to do this, and has been shown to perform quite well recognizing human motion[45].

The probability that the model is in a certain state, S_j, given a sequence of observations, $\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_N$, is defined recursively. For two observations, the density associated with the state after the second observation, $\mathbf{q}_2$, being S_j is:

$$\Pr(\mathbf{O}_1, \mathbf{O}_2, \mathbf{q}_2 = S_j) = \left[\sum_{i=1}^{N} \pi_i b_i(\mathbf{O}_1)\mathbf{a}_{ij}\right] b_j(\mathbf{O}_2) \tag{4.33}$$

where π_i is the prior probability of being in state i, and $b_i(\mathbf{O})$ is the probability of making the observation $\mathbf{O}$ while in state i. This is the Forward algorithm for HMM models.

Estimation of the control signal proceeds by identifying the most likely state given the current observation and the last state, and then using the observation density of that state as described above. We restrict the observation densities to be either a Gaussian or a mixture of Gaussians. There are well understood techniques for estimating the parameters of the HMM from data.

4.4.4 Behavior Alphabet Auto-Selection

Classic HMM techniques require the training data to be segmented prior to parameter estimation. Since we don't necessarily know how to choose a gesture alphabet *a priori*, we cannot perform this pre-segmentation. We would prefer to automatically discover the optimal alphabet for gestures automatically from gesture data. The COGNO architecture performs this automatic clustering[12].

Unfortunately, the phrase "optimal" is ill-defined for this task. In the absence of a task to evaluate the performance of the model, there is an arbitrary trade-off between model complexity and genralization of the model to other data sets[47]. By choosing a task, such as discriminating styles of motion, we gain a well-defined metric for performance.

One of our goals is to observe a user who is interacting with a system and be able to automatically find patterns in their behavior. Interesting questions include:

- Is this (a)typical behavior for the user?
- Is this (a)typical behavior for anyone?
- When is the user transitioning from one behavior/strategy to another behavior/strategy?
- Can we do filtering or prediction using models of the user's behavior?

We must find the behavior alphabets that pick out the salient movements relevant to the above questions. There probably will not be one canonical alphabet for all tasks but rather many alphabets each suited to a group of tasks. Therefore we need an algorithm for automatically generating and selecting effective behavior alphabets. The goal of finding an alphabet that is suitable for a machine learning task can be mapped to the concept of feature selection.

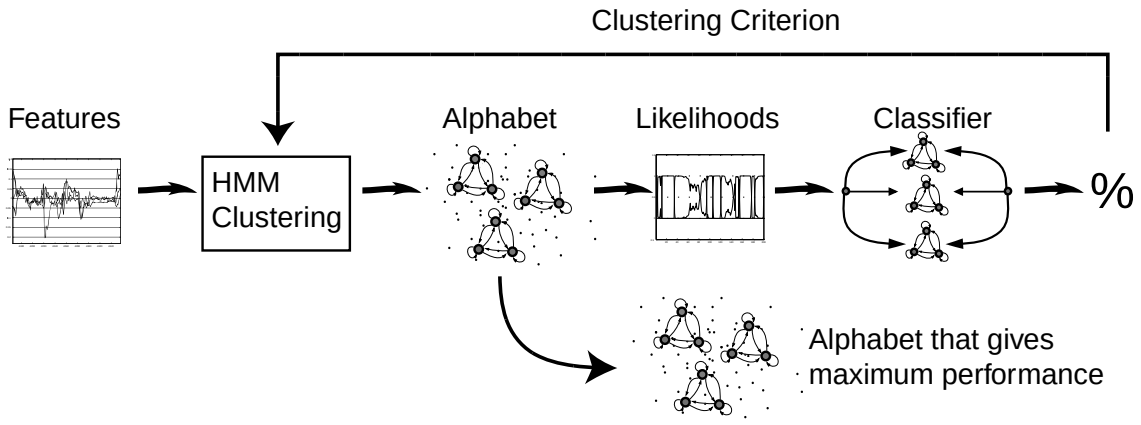

Figure 4-12: COGNO pipeline for alphabet selection.

COGNO Algorithm

In rough terms, COGNO is a clustering algorithm[12]. We chose to use HMMs to model each behavior in an alphabet. Candidate alphabets were generated by clustering the raw features with HMMs. Free parameters of the clustering were:

1. N, Number of HMMs (number of behaviors)
2. S, Number of States per HMM (complexity)
3. T, Time Scale (typical behavior length)

A task-related criterion is used as the performance metric for a meaningful search of this parameter space.

For each set of parameters, we cluster the raw features using an algorithm that can be interpreted as K-Means where the Gaussians are replaced with HMMs. A more complete description of the algorithm can be obtained in [12]. The centroids that result are HMMs that each encode a time sequence in raw feature space (i.e. a behavior). Each HMM is a behavior or symbol in an alphabet that is used to convert the raw features to soft symbols or likelihoods by means of the Forward Algorithm. So if the number of HMMs used is N, then the alphabet size is N and the raw features will be mapped to a likelihood space of N dimensions. The next step is to use the likelihoods to build a classifier for a given task and evaluate the classifier's performance. The classifier's performance is then fed back to the cluster parameter search for model selection. This process is illustrated in Figure 4-12 and outlined below:

1. Input Raw Features (Innovations), ν
2. For each (N, S, τ)
3. Cluster ν with N S-state HMMs at Time Scale τ HMMs H
4. Use HMMs obtained to generate likelihood traces $L(t) = P(\nu(t)|H)$

5. Use L to train and test a classifier for a given task

6. Select H that maximizes step 5's performance.

Examples with data from naive users are presented in Chapter 5.

4.5 Conclusion

This chapter presents a framework for human motion understanding, defined as estimation of the physical state of the body combined with interpretation of that part of the motion that cannot be predicted by passive physics alone. The behavior system operates in conjunction with a real-time, fully-dynamic, 3-D person tracking system that provides a mathematically concise formulation for incorporating a wide variety of physical constraints and probabilistic influences. The framework takes the form of a non-linear recursive filter that enables pixel-level processes to take advantage of the contextual knowledge encoded in the higher-level models. Some of the benefits of this approach including increase in 3-D tracking accuracy, insensitivity to temporary occlusion, and the ability to handle multiple people will be demonstrated in the next chapter.

The intimate integration of the behavior system and the dynamic model also provides the opportunity for a richer sort of motion understanding. The innovations are one step closer to the original intent, so the statistical models don't have to disentangle the message from the means of expression.

Chapter 5

Results

This chapter will provide data to support the DYNA framework. The first part of the chapter will report on the state of the model within DYNA and the quantitative effects of tracking improvements. The rest of the chapter details qualitative improvements in human-computer interface performance in the context of several benchmark applications.

5.1 Tracking Results

The dynamic skeleton model currently includes the upper body and arms. The full dynamic system loop, including forward integration and constraint satisfaction, iterates on a 500MHz Alpha 21264 at 600Hz. Observations come in from the vision system at video rate, 30Hz, so this is sufficiently fast for real-time operation. Figure 5-1 shows the real-time response to various target postures. The model interpolates those portions of the body state that are not measured directly, such as the upper body and elbow orientation, by use of the model's intrinsic dynamics, the kinematic constraints of the skeleton, and and the behavior (control) model.

The model also rejects noise that is inconsistent with the dynamic model. This process isn't equivalent to a simple isometric smoothing, since the mass matrix of the body is anisotropic and time-varying. When combined with an active control model, tracking error can be further reduced through the elimination of overshoot and other effects. Table 5-2 compares noise in the physics+behavior tracker with the physics-only tracker noise. It can be seen that there is a significant increase in performance.

The plot in Figure 5-3 shows the observed and predicted X position of the hand and the corresponding innovation trace before, during and after the motion is altered by a constraint that is modeled by the system: arm kinematics. When the arm reaches full-extension, the motion is arrested. The system is able to predict this even and the near-zero innovations after the event reflect this. Non-zero innovations before the event represent the controlled acceleration of the arm in the negative X direction. Compare to the case of a collision between a hand and the table illustrated in Figure 5-4. The table is not included in the system's model, so the collision goes unpredicted. This results in overshoot, and a

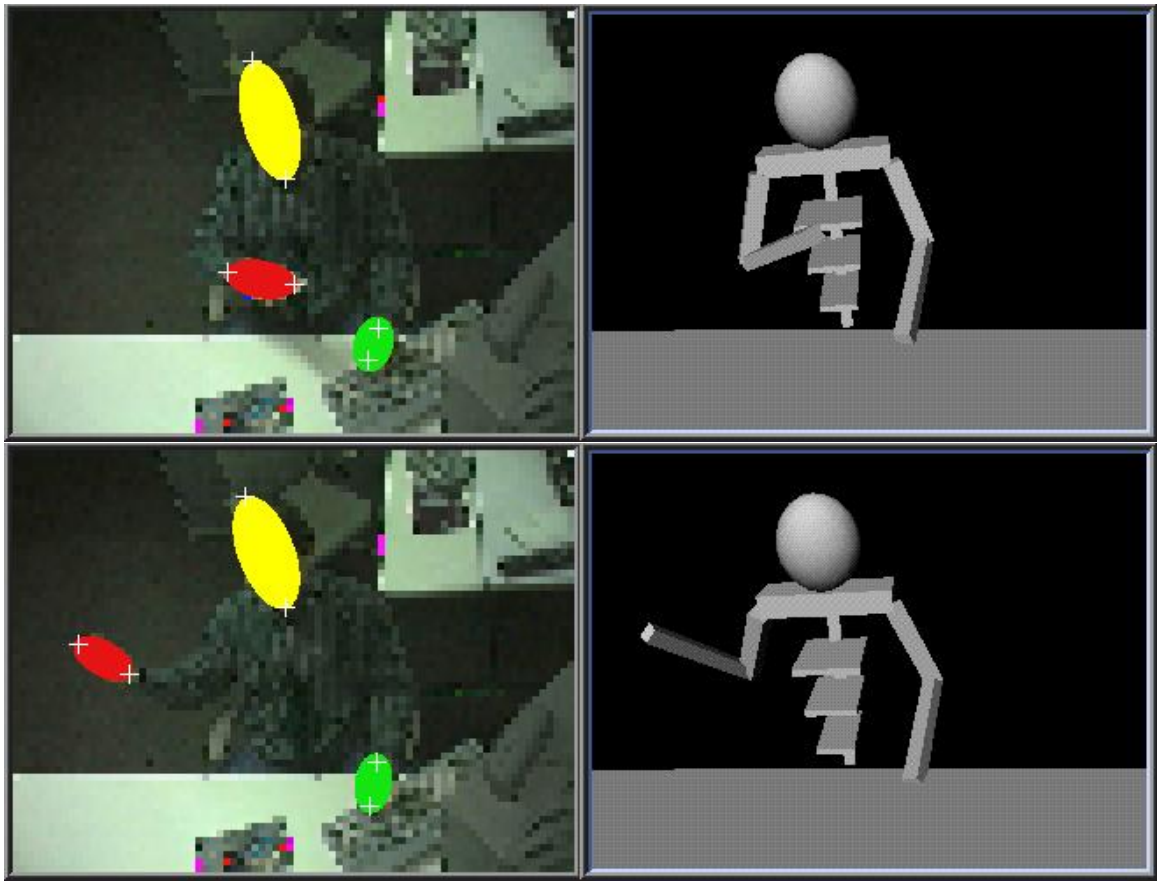

Figure 5-1: **Left:** video and 2-D blobs from one camera in the stereo pair. **Right:** corresponding configurations of the dynamic model.

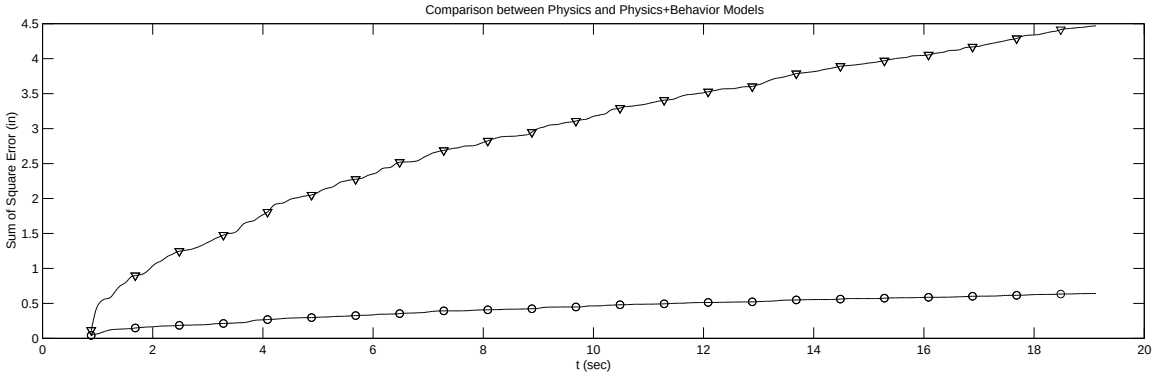

Figure 5-2: Sum Square Error of a Physics-only tracker (triangles) vs. error from a Physics+Behavior Tracker

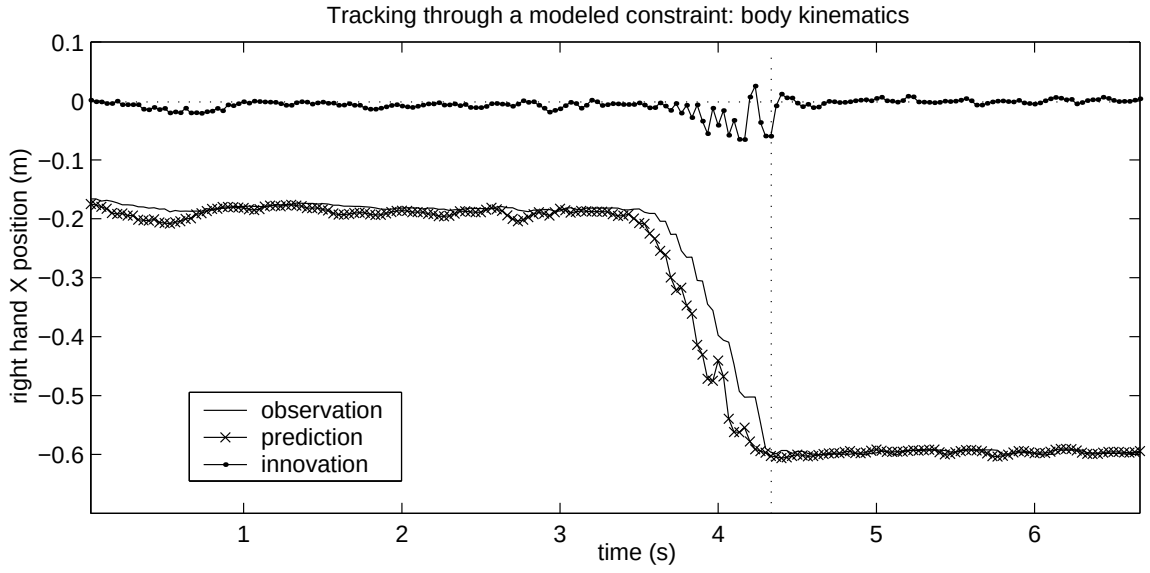

Figure 5-3: Observed and predicted X position of the hand and the corresponding innovation trace before, during and after expression of a modeled constraint: arm kinematics.

corresponding signal in the innovations process after the event.

Figure 5-5 illustrates one of the most significant advantages to tracking of feedback from higher-level models to the low-level vision system. The illustrated sequence is difficult to track due to the presence of periodic, binocular, flesh-flesh occlusions. That is, one hand is occluded by the other from both camera viewpoints in a periodic fashion: in this example at approximately 1Hz. The binocular nature of the occlusion events doesn't allow for view selection to aid tracking: there is no unambiguous viewpoint available to the system. Flesh-flesh occlusions are particularly difficult for tracking systems since it's easier to get distracted by an object with similar appearance (like another hand) than it is to be distracted by an object with a very different appearance (like a green shirt sleeve). The periodic nature of the occlusions means that the system only has a limited number of unambiguous observations to gather data before another occlusion again disrupts tracker stability.

Without feedback, the 2-D tracker fails if there is even partial self-occlusion, or occlusion of an object with similar appearance (such as another person), from a single camera's perspective. In the even more demanding situation of periodic, binocular, flesh-flesh occlusions, the tracker fails horribly. The middle pair of plots in Figure 5-5 show the results. The plots from a cross-eyed stereo pair. The low-level trackers fail at every occlusion causing the instantaneous jumps in apparent hand position reported by the system. Time is along the X axis, from left to right. The other two axes represent Y and Z position of the two hands. The circular motion was performed in the Y-Z plane, so X motion was negligible. It is not shown in the plot.

The situation with feedback, as illustrated in the lower pair of plots in Figure 5-5, is much better. Predictions from the dynamic model are used to resolve ambiguity during 2-D tracking. The trackers survive all the occlusions and the 3-D estimates of hand position reveal a clean helix through time (left to right), forming rough circles in the Y-Z plane.

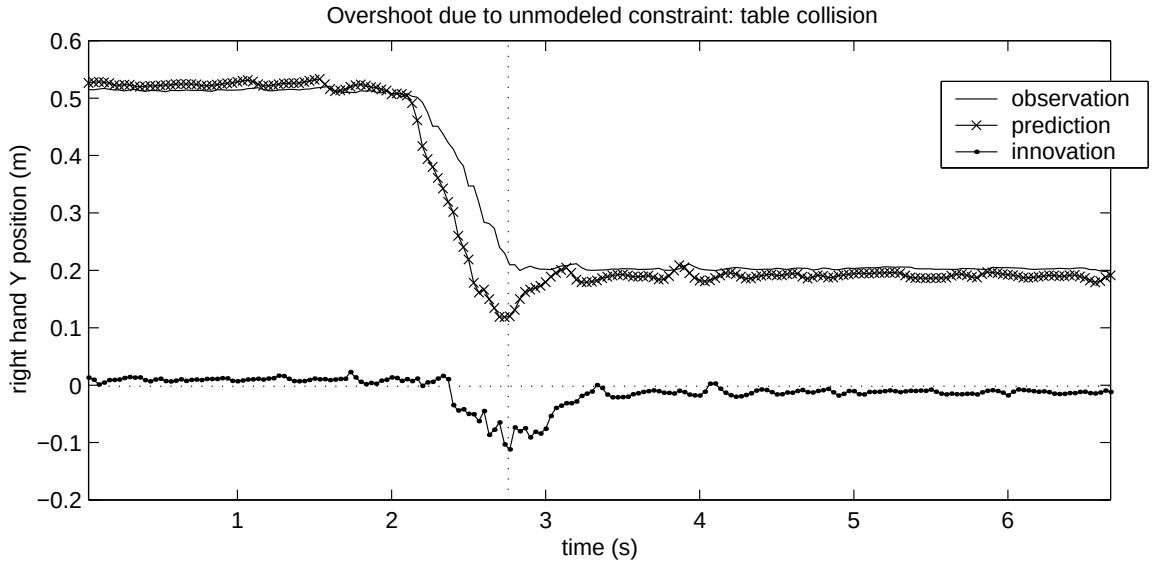

Figure 5-4: Observed and predicted Y position of the hand and the corresponding innovation trace before, during and after expression of a un-modeled constraint: collision with a table.

With models of behavior, longer occlusions could be tolerated.

5.2 Applications

Section 5.1 provided quantitative measures of improvement in tracking performance. This section will demonstrate improvements in human-computer interaction by providing case studies of several complete systems that use the perceptual machinery described in Chapter 4.

The three cases are the T'ai Chi instructional system, the Whack-a-Wuggle virtual manipulation game, and the strategy game Netrek.

5.2.1 T'ai Chi

The T'ai Chi sensei is an example of an application that is significantly enhanced by the recursive framework for motion understanding simply by benefiting from the improved tracking stability. The sensei is an interactive instructional system that teaches the human a selection of upper-body T'ai Chi gestures[5].

The sensei is embodied in a virtual character. That character is used to demonstrate gestures, to provide instant feedback by mirroring the student actions, and to replay the student motions with annotation. Figure 5.2.1 shows some frames from the interaction: the sensei welcoming the student on the left, and demonstrating one of the gestures on the right. The interaction is accompanied by an audio track that introduces the interaction verbally and marks salient events with musical cues.

There are several kinds of feedback that the sensei can provide to the student. The first is the instant gratification associated with seeing the sensei mirror their motions. This allows the student to know that the sensei is attending to their motions and gives immediate feedback regarding their perceived posture relative to the ideal gestures they were just shown. When the sensei decides that feedback is appropriate this mirroring stops: indicating to the student that the interaction paradigm has changed. In this feedback mode the sensei can either critique individual gestures or remind the user of the global order of the sequence. The left and center images in Figure 5-7 show an example of a critique of a specific gesture. Visual and musical cues indicate the temporal location of the error and the sensei's gaze direction indicates the errant hand. The right image in Figure 5-7 is an example of feedback regarding the overall structure of the T'ai Chi sequence.

There are several technologies at work making these interactions possible. The mirroring is accomplished simply by piping the tracking data through the inverse-kinematics engine inside the sensei's body. This is technically simple, but it is imperative that it be robust. It is a meaningful event when the sensei stops mirroring: it signals to the student that they should stop and prepare to receive feedback. Tracking failures can cause the sensei to pause for an indeterminate length of time. Even worse, tracking failures can cause the sensei to spontaneously contort into one of many unpleasant configurations. Either event will obviously detract from the student's learning experience.

The technology behind the identification and interpretation of T'ai Chi gestures is somewhat more complex. To summarize: the sensei learns T'ai Chi gestures by watching a human perform the motions. The system builds Hidden Markov Model (HMM) of each gesture. The HMMs are comprised of a sequence of states with Markov temporal dynamics and Gaussian output probabilities. In effect they are capturing a mean path through parameter space that represents the gesture, and a covariance that specifies an envelope around the mean path that represents the observed variability. The gestures are recognized in the usual way [10]. Once a gesture is recognized, the lattice is examined to find the point at which the observed gesture differ the most from the learned ideal, weighted by the allowable variation [5]. Tracking errors can have a large impact on this process. If a critical chunk of a gesture is missed due to tracking error then it may be unrecognizable, or worse, the system may attempt to correct illusory motion errors and confuse the student.

The individual T'ai Chi gestures are relatively complex. Each gesture takes several second to perform. Critiques often involve playback of observed and ideal gestures, and as a result a single critique may last several tens of seconds. The result is that the frequency of interaction is relatively low. Tracking errors that result in misclassified gestures or illusory motion errors can thus lead to a significant break in the experience. Failures thus lead to frustration. Frustration is antithetical to the underlying goal of T'ai Chi: relaxation and focus.

Figure 5-8 shows a flawless execution of a five gesture sequence as tracked by a bottom-up 3-D blob tracker. The individual gestures are hand-labelled at the top of the plot. The plots show, from top to bottom, the X, Y and Z position of the left hand, right hand, and head. Y is positive up. X is positive to the student's right. Z is positive away from the user. The following discussion will focus on two salient features. The first feature is the initial bump in the Y axis of both hands. This corresponds to the up and down motion of "Opening Form". The second feature is the double bump in the right hand X position at

the end of the sequence. This corresponds to the right hand motion in "Single Whip" and the motion of both hands to the right in "Brush Knee".

Figure 5-9 shows the tracking output from the bottom-up 3-D blob tracker for a slightly different execution of the sequence. To a human observer the motions are very similar except that the motions in Figure 5-9 are slightly more exaggerated. The tracking data looks significantly different due to the tracking failures associated with occlusions. The first salient feature shows significant clipping as the top of the gesture is lost when one or both of the hands occlude the face in one or both of the cameras. This is caused by the hands being raised higher than ideal (or, alternatively if the student is shorter than the master who taught the sensei). The second feature is lost because the hands are held too close together. This causes the 2-D blob trackers to fail in one or both cameras. Almost no tracking data is acquired after 14s due to this error.

Figure 5-10 is the result of tracking motion very similar to that represented by Figure 5-9. The difference is that in this case the motion is tracked by the recursive system described in Chapter 4. Both features are present and well-formed. This is true despite the fact that the hands were up high enough in opening form to cause occlusion with the face and close enough together to cause 2-D tracking failures at the end of the sequence. The recursive structure of the tracker allows the system to integrate information at all levels of modeling to produce good tracks in these ambiguous cases, and sensei is better able to evaluate the student's true departures from the ideal and provide more appropriate feedback.

5.2.2 Whack-a-Wuggle

Whacka is a virtual manipulation game. A virtual actor mirrors the motions of the human's upper body. The goal is for the human to touch objects in the virtual world vicariously through the virtual actor. The world contains static objects (Wuggles) and moving objects (Bubbles). Wuggles can always be whacked by returning to the same physical location since they do not move in the virtual environment, and the mapping between the physical and virtual worlds is fixed. Bubbles move through the virtual space, so they require hand-eye coordination to pop. In addition each Bubble is at a different depth away from the player in the virtual world. Due to poor depth perception in the virtual display, popping a Bubble requires the human to reach up to the approximate location of the bubble and then perform a limited search in depth.

There are a very small number of things that people do with their hands when playing Wacka. Hands are usually either resting, whacking, or popping. Whacking quickly becomes a ballistic action once the human learns where the Wuggles map into the physical world. Popping always involves performing the more complex visual-motor feedback loop required to find the Bubbles because they have random and only partially observable depth. The pace of the game is fast enough to discourage motivated players from wasting time performing extraneous gesticulations.

This context was used to test the method of alphabet selection described in Section 4.4.4. The player's motions were tracked and hand labeled for three types of behavior:

1. Whacking a Wuggle
2. Popping a Bubble
3. Tracker failure

Task 1 was to recognize these three classes of behavior. Task 2 was to be able to distinguish the playing styles of different people.

Figure 5-12 shows the results of searching the model parameter space for the best alphabet in the gesture recognition task. The best alphabet parameters for distinguishing the three players was 3 elements, with 10 states each and a base time scale of 32 frames (1s). Figure 5-13 shows alphabet traces for the three players over approximately one minute of play. These traces are the features used to do player identification. Player identification performance was 75% for three players.

Figure 5-14 illustrates the actions of the best HMMs in the intention identification task for a specific player. For this player the best intention alphabet parameters were 3 elements, with 9 states each and a base time scale of 8 frames (250ms). The plots show the data in grey, and the mean position and iso-probability contours in black. The left and right HMMs seem to be explaining salient motions for recognizing player's intention, while the middle HMM is modeling the outliers caused by tracker failures.

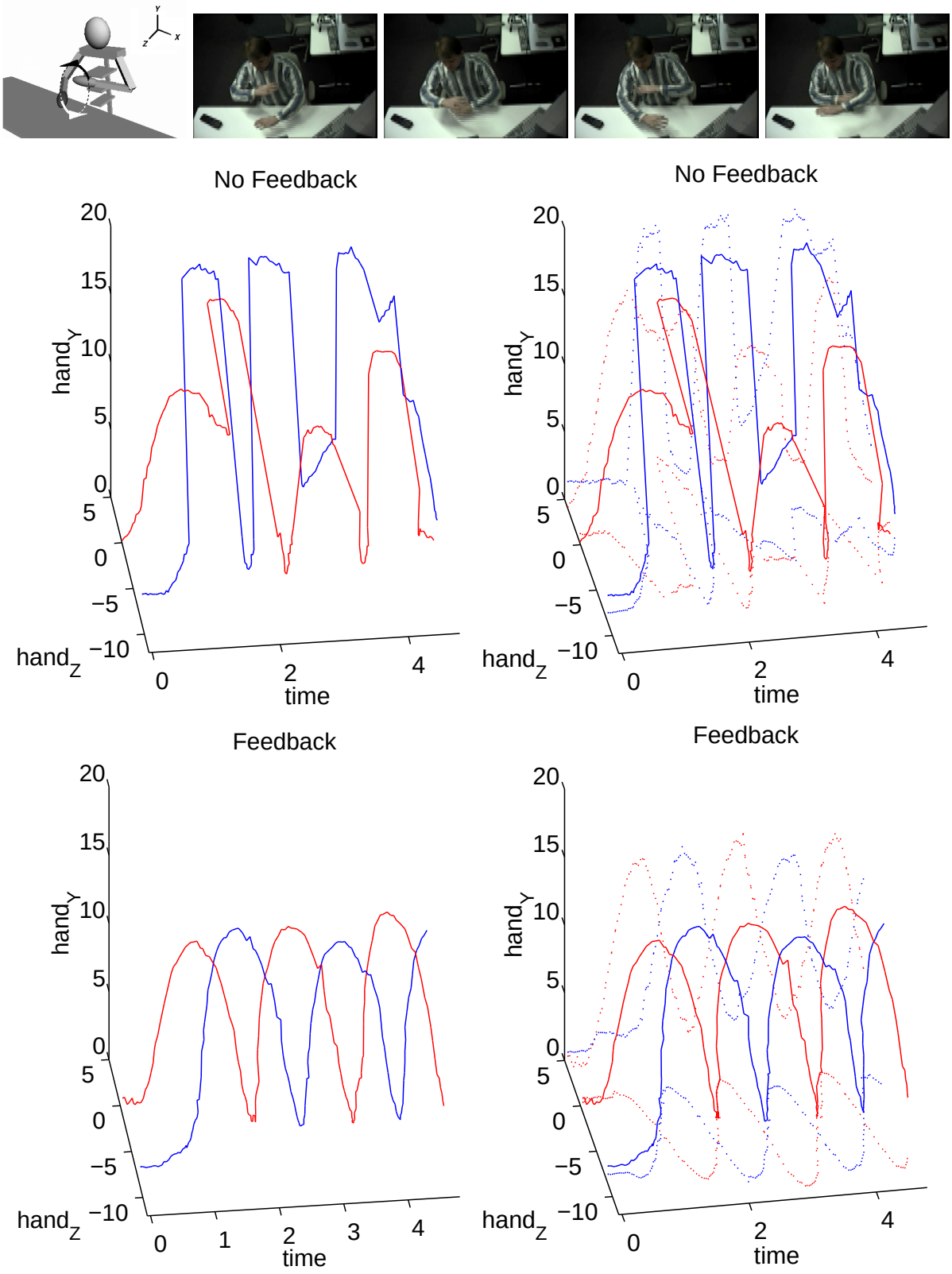

Figure 5-5: Tracking performance on a sequence with significant occlusion. **Top:** A diagram of the sequence and a single camera's view of **Middle:** A graph of tracking results without feedback (cross-eyed stereo pair). **Bottom:** Correct tracking when feedback is enabled (cross-eyed stereo pair).

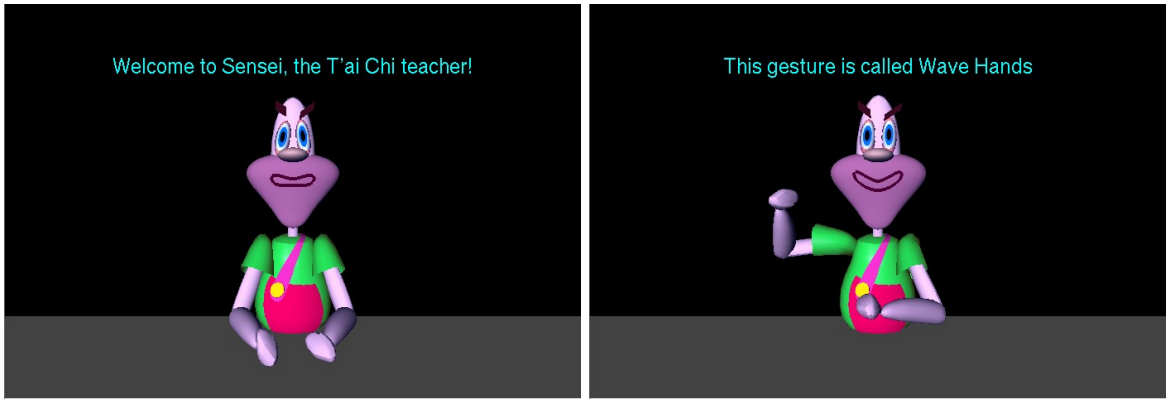

Figure 5-6: The T'ai Chi sensei shows gives verbal instruction and uses it's virtual body to show the student the T'ai Chi moves.

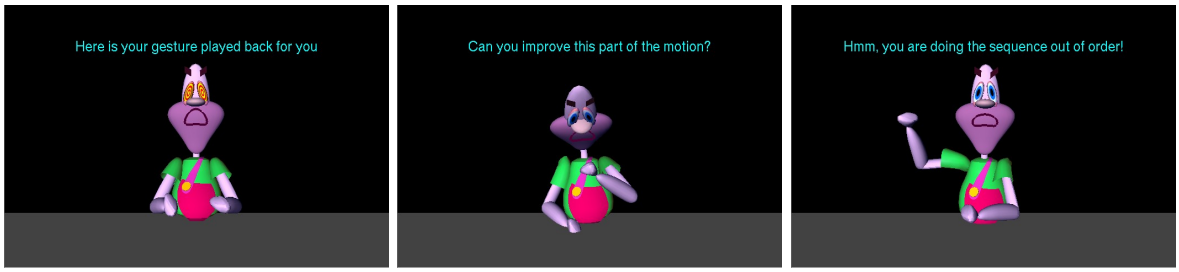

Figure 5-7: Visual and Auditory cues are used to give the student feedback. The sensei mimics the student motions and indicates problems to be worked on.

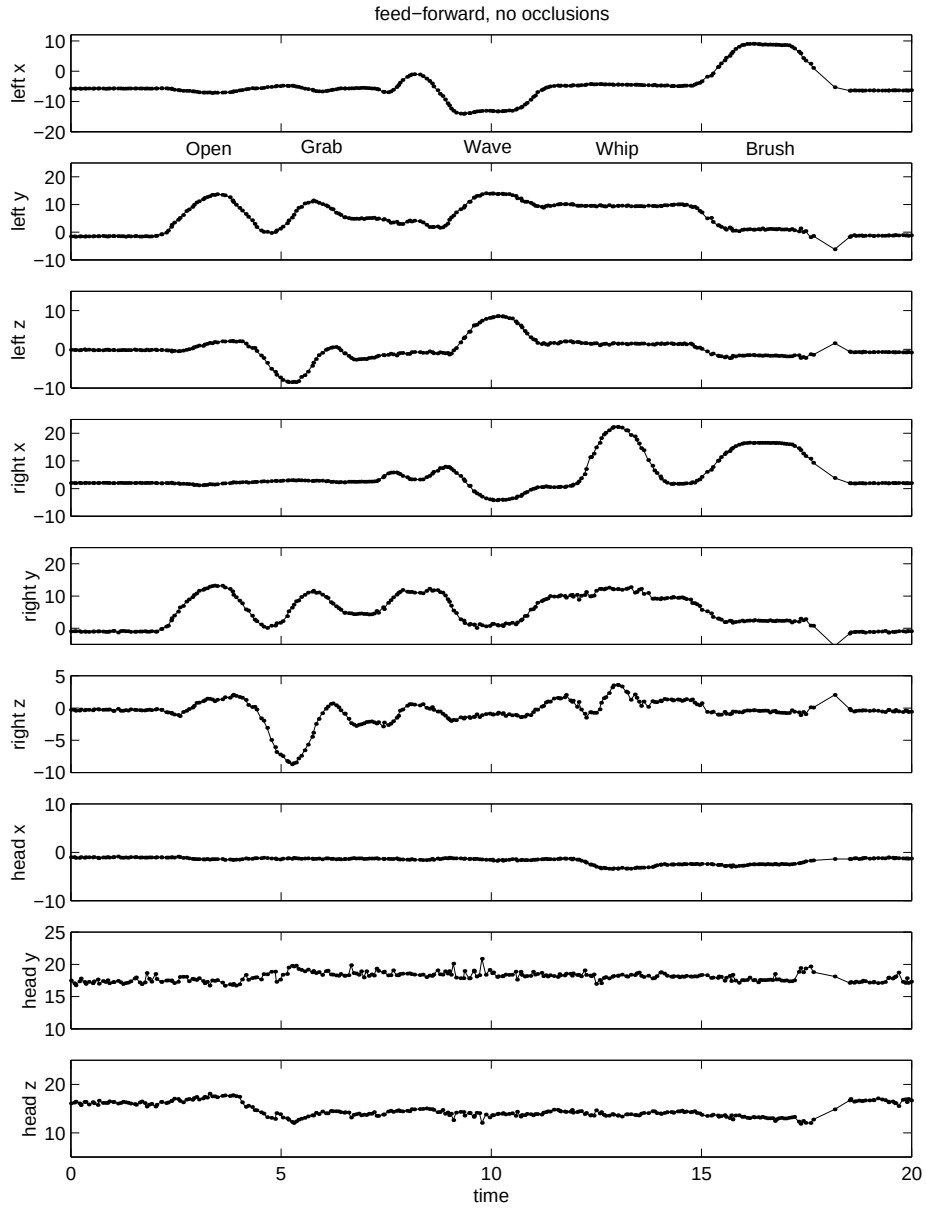

Figure 5-8: This is an example of a T'ai Chi sentence performed without feedback enabled. The gesture labels are at the top. The graphs show tracking data for left hand, right hand and head. The gestures can be performed without occlusions in the STIVE space.

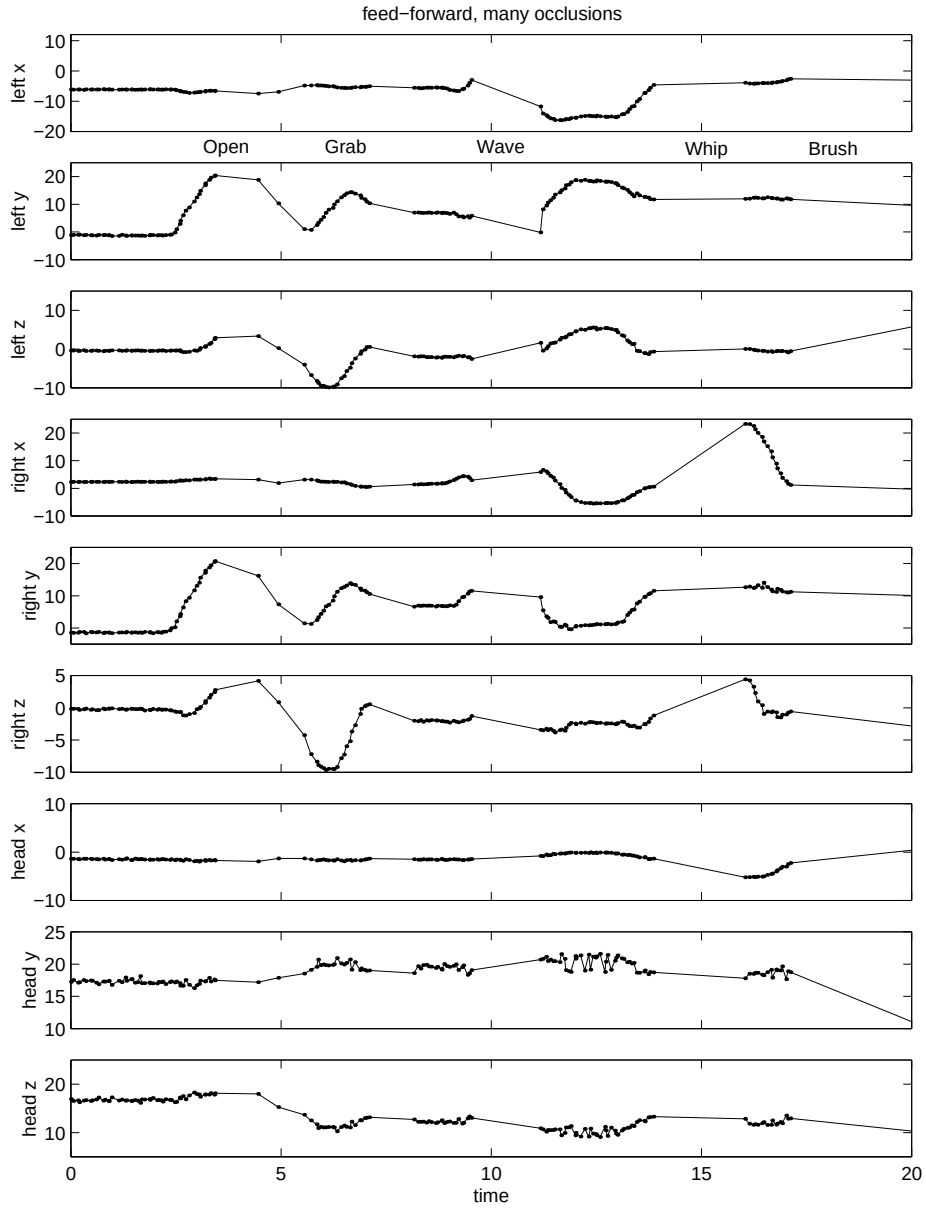

Figure 5-9: This is an example of a T'ai Chi sentence with significant occlusion failures. Compare to Figure 5-8. This example is intentionally pessimal to demonstrate all the possible failure modes. Missing dots indicate periods of severe tracking failure. See text for details.

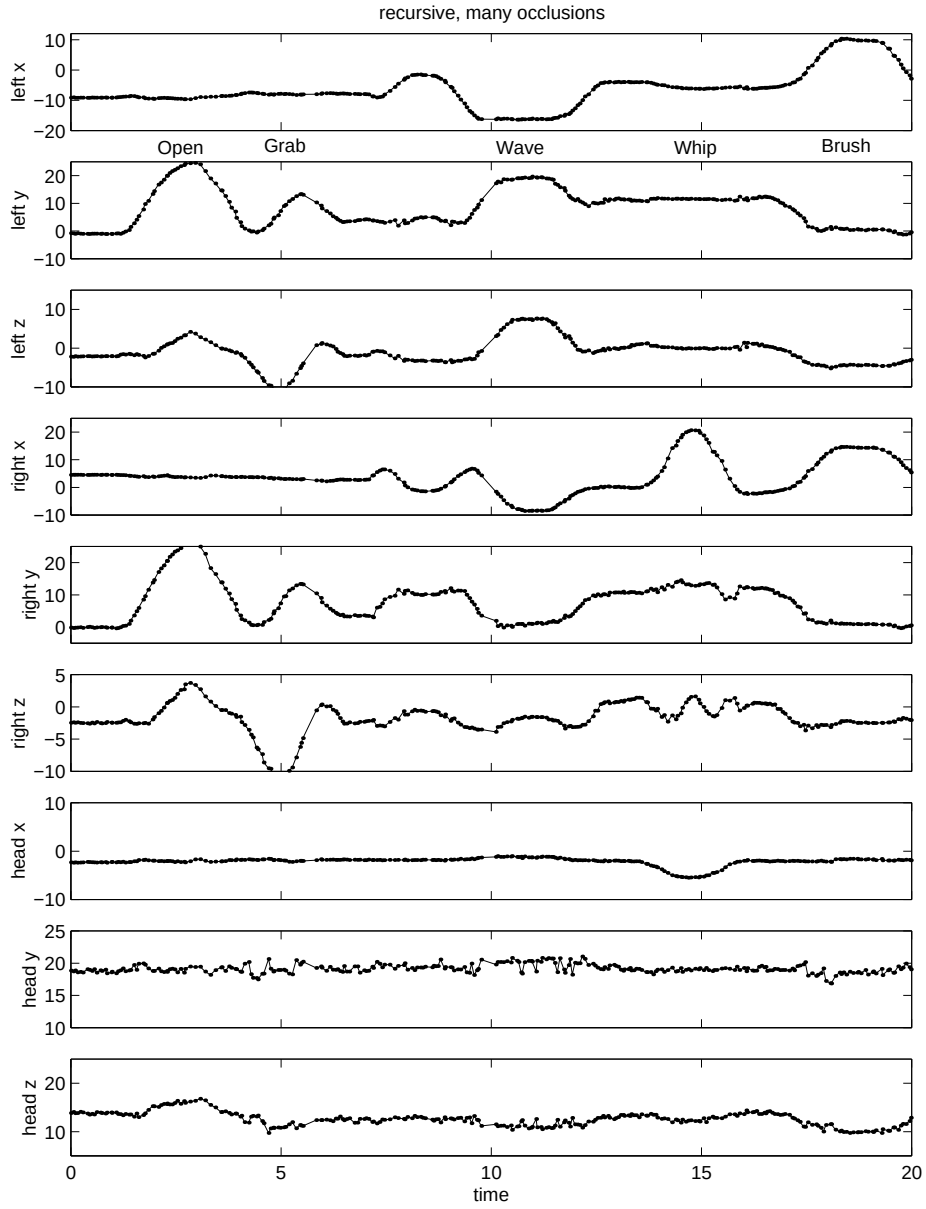

Figure 5-10: A T'ai Chi sentence with many of the same performance "mistakes" as in Figure 5-9, but this time the tracker is able to resolve ambiguities and provide good data to the sensei.

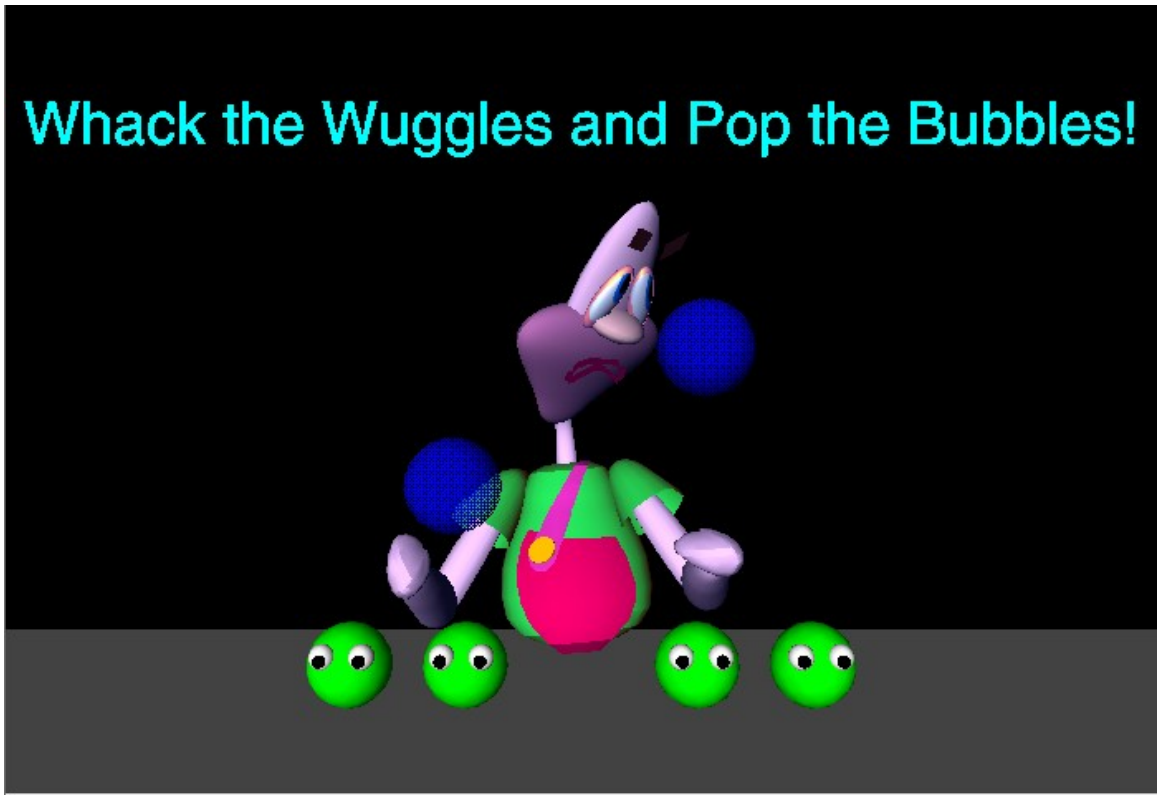

Figure 5-11: In Whack-a-Wuggle a clown mirrors the player's body motions. By guiding the clown's arms, the player can Whack Wuggles (the objects with eyes) and pop Bubbles (the objects floating in the air).

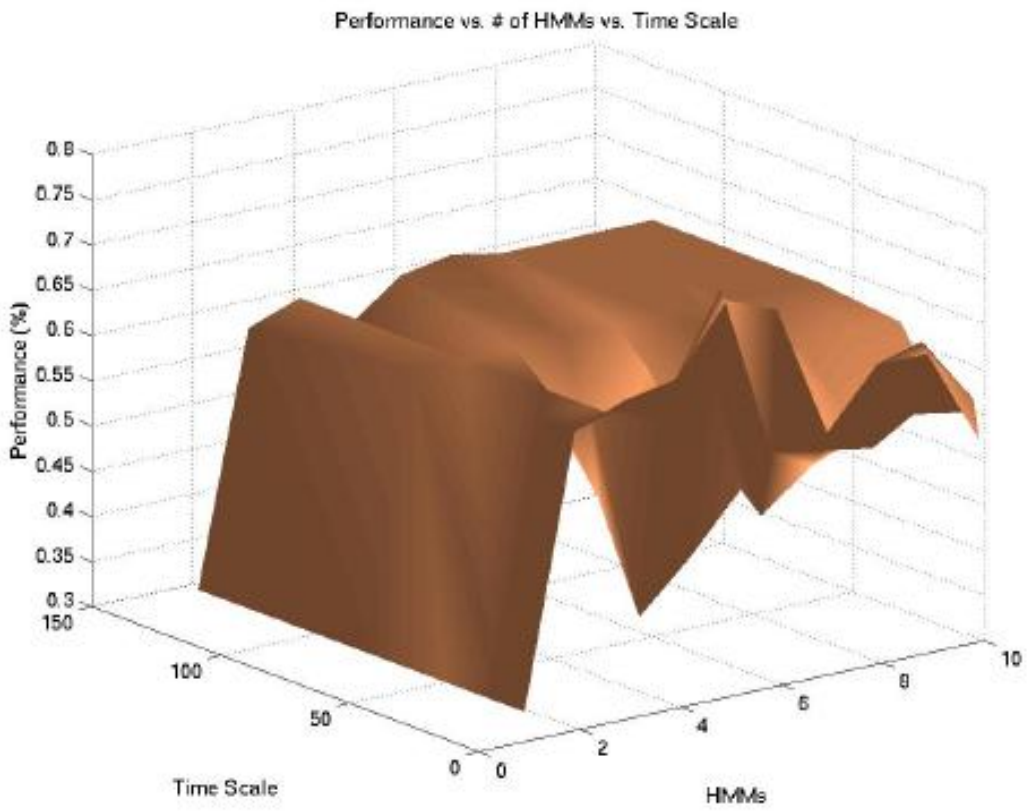

Figure 5-12: Plot showing gesture recognition performance as a function of model complexity.

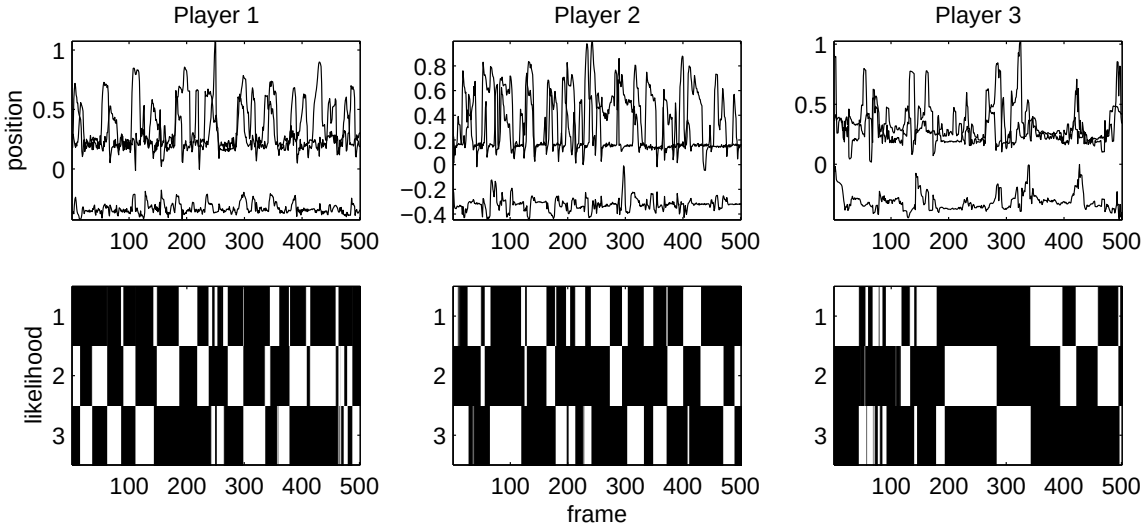

Figure 5-13: **Top:** Tracking data. **Bottom:** Corresponding likelihood trace of the identification alphabet.

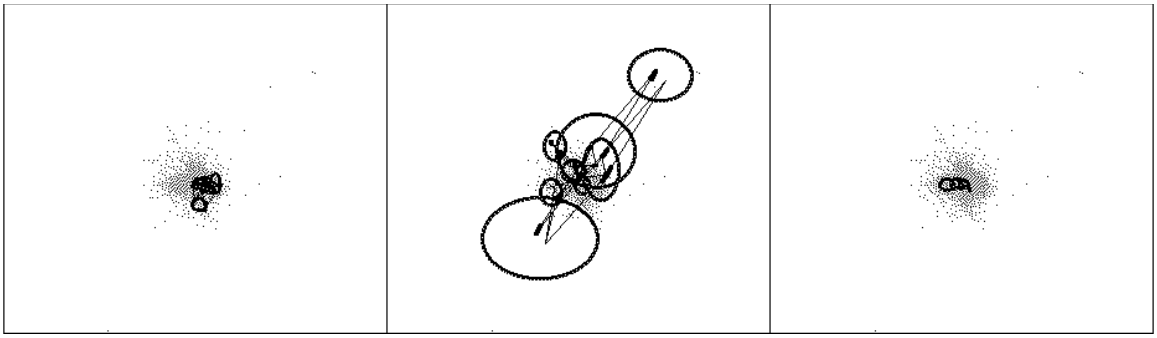

Figure 5-14: HMMs for the intentionality alphabet.

Chapter 6

Future Directions

In this chapter the game Netrek is proposed as a test-bed for perceptual user interfaces. The game Netrek provides a rich context for interfaces while retaining the closed world that makes a game environment tractable as a research platform. The next section will give an introduction to Netrek with an emphasis on the elements that make it particularly well suited to perceptual user interface work. Section 6.2 will detail the current state of the test-bed as embodied in *Ogg That There*.The last section will outline proposed additions that are intended to push perceptual user interface research and pave the way toward a truly novel human-computer interface.

6.1 The Netrek Domain

Netrek is a game of conquest with a Star Trek motif. The game is normally played by up to 16 players organized into two teams. A team wins by transporting friendly armies to each of the opposing team's planets. Figure 6-1 illustrates some of the basic elements of the game: planets, ships, and armies. The first benefit of Netrek as a test-bed for user interfaces is that it is a game: so it provides built in metrics for the success of a new interface design. If the interface allows a player to play the game more effectively (to win more), then the interface can be said to be successful.

Netrek is very much a team-oriented game. Winning requires a team that works together as a unit. This fact , in particular, provides a rich set of interface opportunities ranging from low-level tactics to high-level strategy. There has been some work on new tactical interfaces, but these interfaces were blocked by the Netrek community with an authentication system to keep games fair. We will concentrate on the opportunities for building interfaces for high-level communication regarding strategy since these provide the most room for novel interface design.

Netrek is usually played by groups of players on wide area networks spread over large geographic areas. The standard interface requires the player to directly control their ship. Communication with teammates is accomplished by type-written messages: this means that in order to send messages the player must temporarily give up control of their ship. So

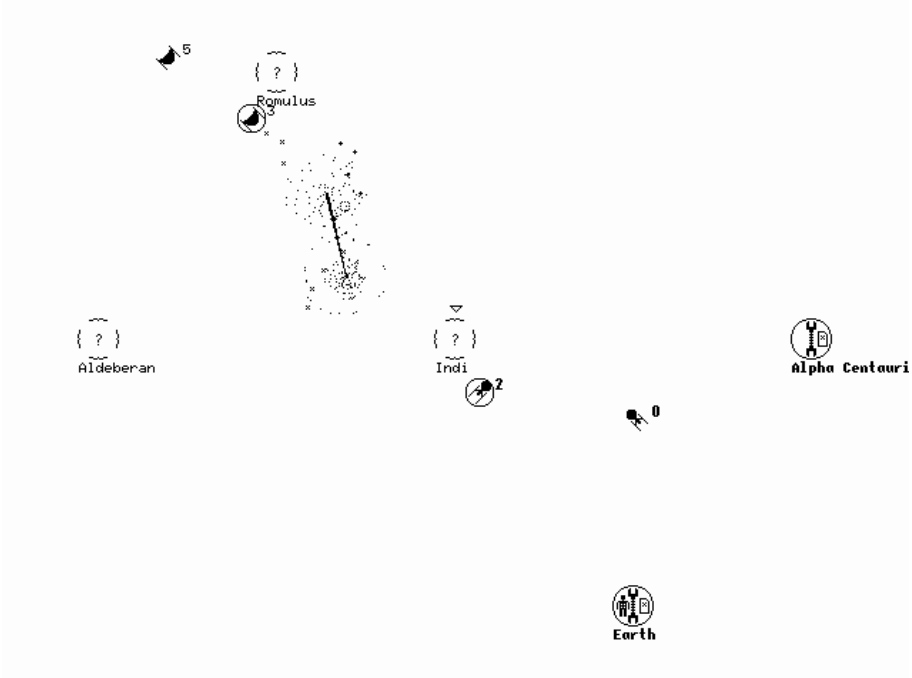

Figure 6-1: Netrek Screenshot: Federation ships $F0$ and $F2$ defend federation space near Earth and Alpha Centauri. Indi Romulus and and Aldeberan are unexplored by the Federation. Romulan ships $R3$ and $R5$ hold near Romulus. $R4$ and $F1$ have just destroyed each other near Indy.

players must communicate about strategy and complex maneuvers in an efficient manner. This necessity led to the creation of a set of staccato jargon and an associated set of what we will call programs. The programs are essentially little plays with a few roles that can be filled by appropriate players.

The existence of these programs is good for research in several ways. First there is pre-existing code, called a robot, that is capable of running a small set of these programs as well as game basics like tactics and navigation. These robots provide a solid base on which to build a research system. The jargon also represents a strict codification (enforced by the difficulty of communication) of human play that might indicate that recognition of plays and machine learning of plays through observation of human players would be tractable. This codification also means that there may be opportunities for novel, expressive interfaces to encourage the formation of new strategies.

One last aspect of Netrek is the virtual embodiment of the robots in the game. The ships obey a simple dynamic model and they have limited control. This is particularly interesting given the proposal to represent behaviors as control signals to dynamic systems. This creates a satisfying duality between the mode of expression and the target of that expression.

Figure 6-2: Netrek Collective Interface: the player uses a deictic gesture to select $F0$. Cameras on top of the display track the player's movements and a head-mounted microphone is used for speech recognition.

6.2 Initial Integration: *Ogg That There*

The first version of the Netrek Collective, entitled *Ogg That There*, is intended to perform in a manner similar to the classic interface demo "Put That There"[7]. Imperative commands with a subject-verb-object grammar can be issued to individual units. These commands override the robots internal action-selection algorithm, causing the specified action to execute immediately. Objects can either be named explicitly, or referred to with deictic gestures combined with spoken demonstrative pronouns. Figure 6-2 depicts a player selecting a game object with a deictic gesture.

Figure 6-3 illustrates the system architecture of *Ogg That There*. Thin, solid lines indicate standard socket-based Netrek communications. Thick, dashed lines indicate RPC based communication between our modules. The modules can be distributed across a cluster of machines to allow for future expansion of resource requirements. The following sections give details on the *Perception*, *Interpretation*, *Display* and *Robot* modules.

Perception: Deictic Gestures Deictics are the only form of gesture supported by *Ogg That There*. They are labeled by speech events, not actually recognized. Interpretation of deictics relies on interpolation over a set of calibration examples obtained off-line by asking the player to point at the four corners of the screen with both hands in turn. This results in four sets of measurements for each hand. Separate calibrations are maintained for each hand.

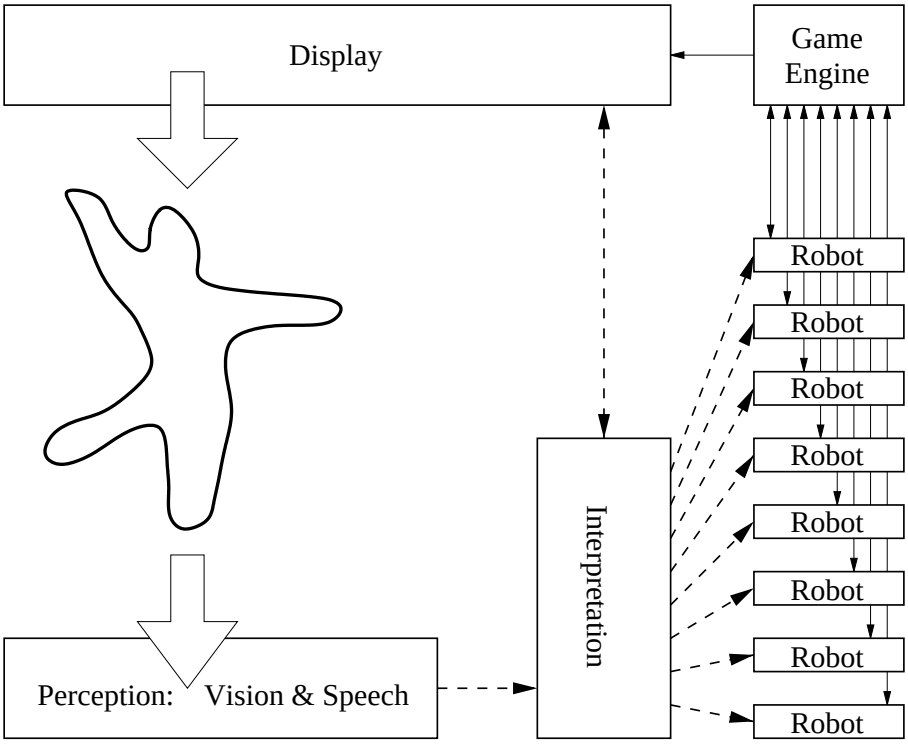

Figure 6-3: Netrek Collective System Diagram. Arrows indicate information flow. The *Display* module doubles as a database front-end for the *Interpretation* module so that no modification are needed to the *Game Engine*.

In general these four points will not form parallelograms in feature space, so linear strategies introduce unacceptable warping of the output space. *Ogg That There* employs a perspective warping to translate input features ((x, y) hand position in 3D space) to screen position:

$$\begin{bmatrix} XW \\ YW \\ W \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{6.1}$$

where X and Y are screen position and x and y are hand position. The parameters a, b, c, d, e, f, g, h are estimated from data. Interpolation of a novel deictic simply involves plugging the new (x, y) into Equation 6.1. The resulting screen coordinates, (X, Y), are then passed to the *Display* which does a pick action to convert those coordinates into a game object.

Display The state of the game is displayed on a rear-projection screen. The *Display* module generates the graphics for this screen. It is a standard Netrek client with several enhancements. An RPC interface allows remote access to standard display parameters such as viewport zoom and pan, plus addition features for the *Ogg That There* such as remote cursor display, highlighting of game objects, and textual feedback to the player. Some of these features can be seen in Figure 6-2.

The *Display* also provides a high-level interface for game information. For *Ogg That There* this is used by the interpreter to retrieve game objects that appear near a given screen location and satisfy a list of grammatical constraints.

Robot The *Robot* is instantiated several times in *Ogg That There*. There is one *Robot* for each player on the game. Not shown in Figure 6-3 are the eight players on the enemy team: the same code runs players on both teams.

The *Robot* contains a large amout of Netrek domain knowledge in the form of heuristic functions and implementation of several of the programs discussed above. There is also a collection of motor skills that generate commands to the *Game Engine* and allow the robot to perform all the primitive Netrek functions.

The heuristic functions are used by a primitive action-selection algorithm that selects targets and dispatches one of the tactical programs each tick. An RPC interface allows the *Interpreter* to override this system and explicitly execute a tactical program with supplied targets.

Interpretation The *Interpretation* module reads features from the vision system described above as well as an adaptive speech system developed by Deb Roy [42]. These feature streams must then be converted to game actions: commands to the robots or configuration of the display. The *Interpretation* module can query game state via the database engine built into the *Display* module, and can query individual robots regarding their internal state.

Currently this module is implemented as a finite state machine implementing the subject-

verb-object grammar used in *Ogg That There*. The adaptive speech system is pre-loaded with a set of verbs, literal nouns and demonstrative pronouns. During the game speech events advance the state of the interpreter. If a demonstrative pronoun is encountered the interpreter resolves the gesture features in screen coordinates as described above. Those screen coordinates are then combined with grammatical constraints on valid referents from the current state of the FSM to generate a query on the *Display* database. Once a full sentence is parsed and all referents are instantiated, a command is issued to the appropriate robot.

6.3 Next Generation

Current work focuses on learning patters of audio-visual behaviors in a constrained context. The Collective vignette supplies a constrained subset of the larger Netrek universe that covers a crucial aspect of game play: cooperation among ships to take a planet. The beginning game state is illustrated in Figure 6-4. The player begins with one planet populated with armies, two disposable guards, and one indispensable ship that can carry armies, labelled the quarterback in Figure 6-4. The player's task is to capture the contested planet planet in the center by using the quarterback to move armies there from the home planet in the upper left. The trick is that a frontal assault will result in the loss of the quarterback and failure. It is necessary to coordinate the actions of the friendly guards to dispose of the enemy guard before the quarterback arrives with the armies. A certain amount of timing is involved because the enemy guard regenerates at the enemy home in the lower-left and will proceed back to its defensive position post in the center.

6.3.1 Wizard of Oz

The details of the actions of the ships in the vignette are controlled by freeware software agents written by the internet netrek community. Modifications to the robot allows a remote process to control the robots actions by setting high-level goals via remote procedure calls (RPC). The highlevel goals are set by the operator during the experiment via a button box written in Java. The button box is illustrated in Figure 6-5. The buttonbox issues commands via RPC and also generates time-stamped tags to a file. There are also processes that record game state, gesture data, and speech data with similar timestamps.

Once the data is recorded and temporally aligned with the help of the timestamps, then the data can be analyzed with tools such as Cogno, descibed in Section 4.4.4. Current and future work includes using Cogno to discover gestural primitive, as in the Whakka example. The gesture primitives will then be combined with speech primitives (phonemes) and reclustered with Cogno in an attempt to discover audio-visual primitives.

6.4 Future Work

Ogg That There is very fragile. The FSM inside the *Interpreter* limits the robustness of the

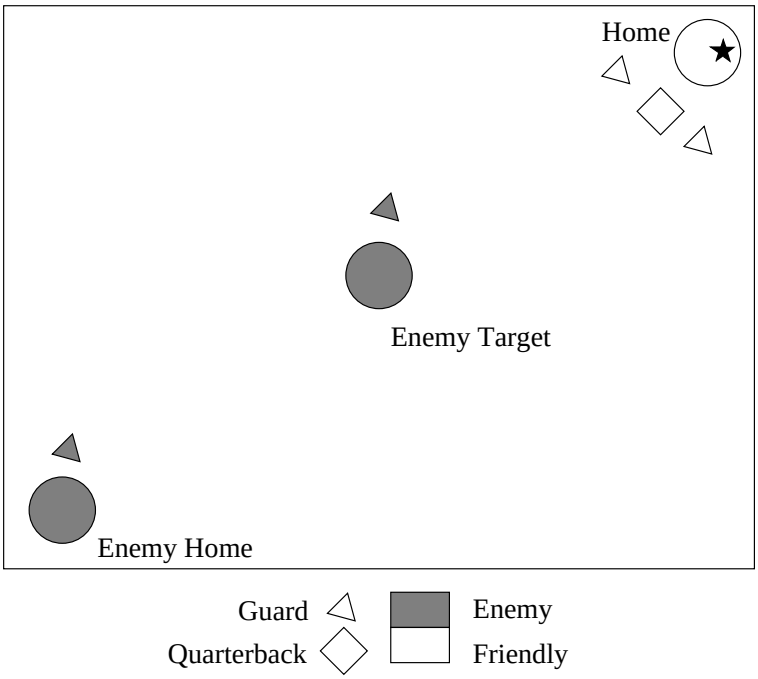

Figure 6-4: Set setup for the Collective vignette. The player conrols three ships and one planet in the upper left. The enemy controls the contested planet in the center, and some guards.

Quarterback	Wing 1	Wing 2
Override	Override	Override
Protect Earth	Protect Earth	Protect Earth
Protect Sirius	Protect Sirius	Protect Sirius
Take Sirius	Defend F0	Defend F0
Pickup At Earth	Escort F0 to Sirius	Escort F0 to Sirius
Ogg R4	Ogg R4	Ogg R4

Figure 6-5: This is the control panel that the operator uses during the Wizard of Oz experiemnt to control the robots in response to user directives.

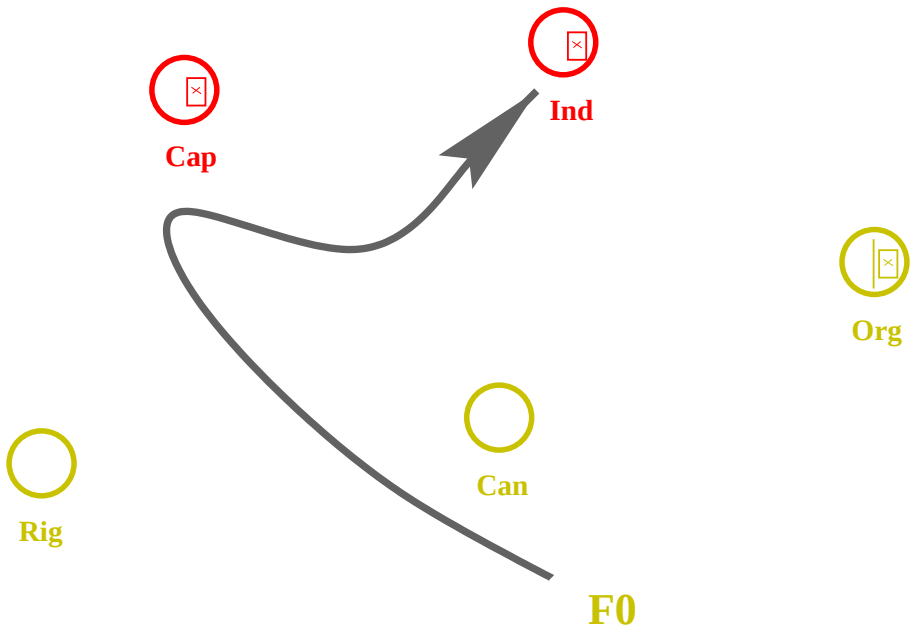

Figure 6-6: A situation where this feint path might be provided by the user gesturally even though the robot is not explicitly programmed to execute feints.

system as does the rigid grammar. Limiting the user to deictic gestures denies the potential expressiveness of a gestural interface. The reliance on imperative commands issued to specific robots doesn't keep up with the pace of the game, and leads to frustration as the situation changes faster than commands can be issued. *Ogg That There* succeeded in solving many integration issues involved in coupling research systems to existing game code, but it's now time to redesign the interface to more accurately match the flexibility of the perceptual technologies, the pace of play, and the need for a game interface to be fluid and fun.

Work is already underway to replace the FSM framework with a production system that can propagate time-varying constraints against recent perceptual events to generate parses. This should alleviate much of the brittleness of the *Ogg That There* implementation. Unfortunately the state-of-the-art for production systems falls short of what we would like to have for this project. However, even if we aren't able to break out of the need for grammars, it should be straightforward to support a wider array of possible grammars as well as to recover from simple speech recognition failures. Utterances and gestures that fall outside the scope of the implemented grammars will have to be handled outside the system. Some ideas for this special handling are explored below.

Ogg That There doesn't make use of the machinery in Chapter 4 except as a means to increase tracking performance. An interpreter that made more elaborate use of gesture could provide a much richer interface. Building a gestural language system is a popular interface technique, but it requires users to be trained and distances the user from the control task by interposing a symbolic layer.

The innovations-based representations for behavior described in Section 4.4.1 combined with the embodied, physics-based, nature of the Netrek robots presents a possibility for non-symbolic communication with the robots. Innovation steams, or parametric models of

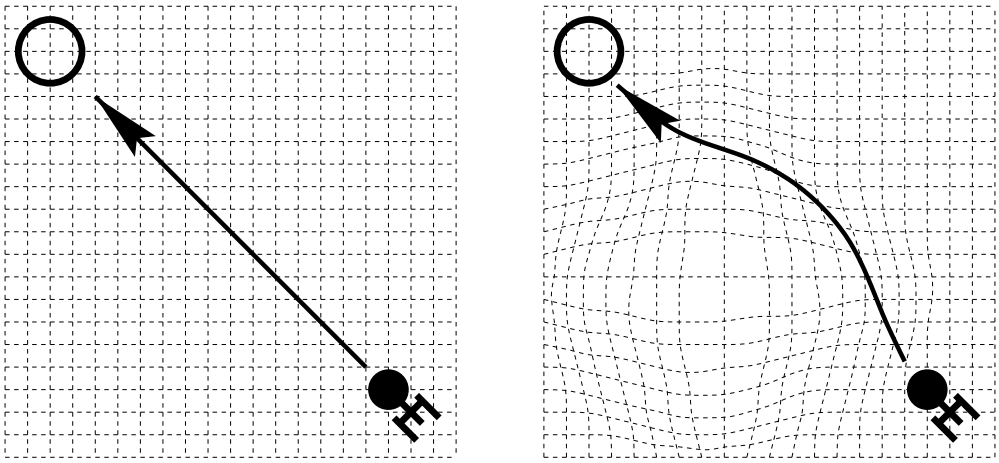

Figure 6-7: Modification of the navigation motor skill can affect a warping of space for any code that uses the low-level skill.

the innovations, could be provided to the robots as a control strategy to be layered on top of the current task. These controls can be thought of as non-verbal adverbs that would be difficult or impossible to convey verbally. Figure 6-6 illustrates a possible situation where the user may want a carrier, $F0$, to execute a feint toward Capella before hitting the real target, Indi. Human teammates might type to $F0$, "Take Ind, feint at Cap". If the robot isn't explicitly coded to execute feints (or if the human player doesn't know the word feint), then this symbolic strategy will fail.

Similar strategies, with appropriate intermediate representations, may also be possible for the audio modality. A command "F1, get in there!" said in a staccato, high-energy way might bias $F1$ toward higher speeds and maximal accelerations even if the system was only able to recognize the word "F1" (or maybe not even this if the viewport is on $F1$ or the user is gesturing toward $F1$). It seems that this may end up being somewhat more symbolic since the feature space of speech is so different from the control space of the robots. An analysis system might recognize agitated speech and generate a symbol representing agitation that the *Interpreter* could choose to pass on to one or more robots.

While the preceding discussion is hypothetical, one modification to the *Robot* motor skill code has already affected qualitative changes in the robots behavior without the need to modify the existing programs that use that skill. This functionality is not used in *Ogg That There*. It allows the *Interpreter* to specify a warping of space that affects how the low-level navigation skill expresses itself. Figure 6-7 illustrates an example of how a ship might avoid an area indicated as dangerous in its way to a target. This communication channel is probably more useful for global influences, as opposed to the more local, control-level example of the feint described above.

An even more indirect, global, non-symbolic influence involves the possibility of modifying the way that the robot decides what action to take. *Ogg That There* overrides the action-selection algorithm completely. So, either the robot is making its own decisions or it is following the imperative commands from the user. There is no possibility to simply bias the decisions of the robot in a certain direction. Figure 6-8 shows a possible scenario where a robot chooses a different target depending on the presence of a bias indicating a difference

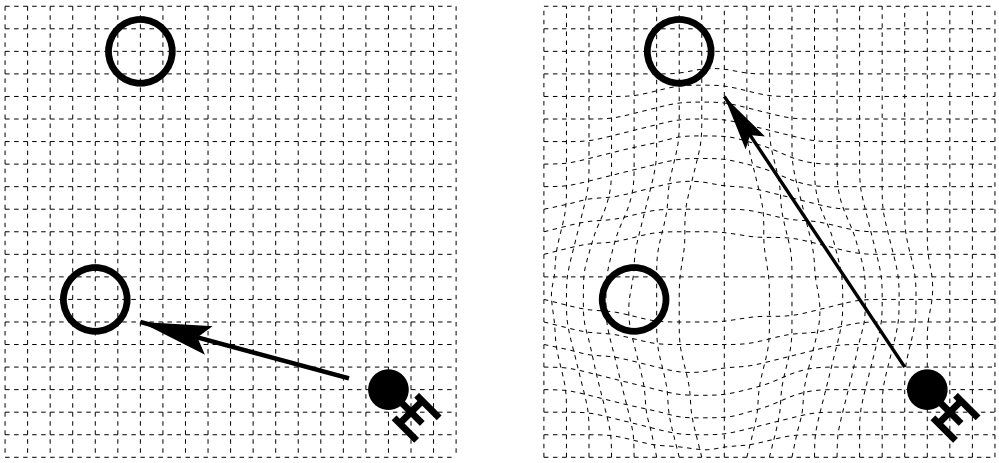

Figure 6-8: Modification of the action-selection algorithm to include user supplied weights. In this example the ship may choose to attack a different target because the most desirable target exists in an area deemed undesirable by the user.

in the value of the targets as perceived by the user.

The current action-selection implementation is the original code that came with the robots from the Netrek community. It is a rather brittle (not to mention obfuscated) collection of heuristics. The first step toward this sort of interaction with the robots will be the adoption of a more sophisticated action-selection implementation. An obvious choice is to use the current implementation of Bruce Blumberg's reactive behavior architecture, since it has proven itself flexible and is readily available [6].

While this mode provides the least direct control over individual robots, it is important to note that this is also a mechanism for specifying high-level strategic goals and hazards that can affect many robots at once. Moving away from the imperative command structure toward a method for specifying abstract goals will increase the ability of the interface to keep pace with the game. Deictics will undoubtedly be important for specifying these goals, but natural specification of the polarity and severity to be associated with the demonstrated region will probably rely on stylistic attributes of the deictic and accompanying voice events. That makes this class of communication another interesting challenge.

All of these possibilities have a theme in common: they are attempting to extract content from parts of the user input that are normally ignored by classic user interface techniques like those illustrated in *Ogg That There*. An extreme example is useful to illustrate: imagine that the user foresees imminent disaster. The user does not have time to communicate in a lucid fashion, but given the desperation of the situation, they are likely to try anyway. Classical interfaces would experience speech and gesture recognition failures, and would either give up or, in the most advanced case, would ask the user a leading question. This is exactly the wrong response. There are probably only a few bits of information present in the user's desperate squeaking, but they are *very* important bits: "**look out!**" The only kind of process that is going to recover these bits is one that is attending to the nature of the signals: the energy and pitch of the voice, and the style (in an innovations-based, statistical sense) of the gesticulations.

Chapter 7

Conclusion

This dissertation has examined a body of sophisticated perceptual mechanisms developed in response to the needs of human computer interface, as well as a selection of interface sketches. It has covered, in depth, the formulation of a fully recursive framework for computer vision called Dyna that improves performance of human motion tracking. The improvement in tracking performance is accomplished with the combination of a three-dimensional, physics-based model of the human body with modifications to the pixel classification algorithms that enable them to take advantage of this high-level knowledge. The result is a novel vision framework that has no completely bottom-up processes, and is therefore significantly faster and more stable than other approaches.

Every level of processing in the Dyna framework takes advantage of the constraints implied by the embodiment of the observed human. Higher level processes take advantage of these constraints explicitly while lower level processes gain the advantage of the distilled body knowledge in the form of predicted probability densities. These predictions enable more robust classification decisions. Systems which omit a model of embodiment entirely, or try to hallucinate physical constraints in the image plane will fail to capture important aspects of human motion and the overall system performance will suffer. Understanding embodiment is crucial to perceiving human motion.

Systems that rely on any completely bottom-up processing will incur unacceptable performance penalties to recover from inevitable low-level errors, if it is possible to recover at all. Recursive frameworks are also crucial to perceiving human motion.

Improvement of the interface between people and computation was the primary motivation for the design and implementation of Dyna. This work demonstrated the operation of the framework in the context of three applications. The first showcased enhanced performance for a pre-existing motion tutor called the T'ai Chi Teacher. The second application, called Wakka, involved fast-paced manipulation of virtual objects and served as a context for motion primitive discovery. The final application is a command and control interface built on top of the distributed, open-source game Netrek that points the way to the future of interface.

This dissertation contributes to the domain of machine perception. As these technologies

continue to develop an mature they will enable an environment rich with interactivity. Computers will cease to be boxes that necessitate unnatural, and sometimes painful or damaging interaction. Computers disappear into the environment, and the things we do naturally will become the primary interface. Our most important experiences are interactions with other people, and as machine perception advances, computation will finally begin to engage in that conversation.

Appendix A

Dynamics

This appendix will cover some of the basics of dynamic simulation including the formation of mass matrices, basic integration, and some simple constraint forms[25, 33, 51].

A.1 Introduction

The continuous motion of real objects is simulated digitally with a discrete model. Newtonian mechanics is used to determine the instantaneous relationship between applied forces and accelerations. Accelerations are integrated over time to obtain velocities. Velocities are integrated over time to obtain position. For a 1-D system with position x, mass m and applied force X, the discrete approximation to the system is:

$$\ddot{x}_t = \frac{1}{m} \cdot X_t \tag{A.1}$$

$$\dot{x}_t = \dot{x}_{t-\Delta t} + \Delta t \cdot \ddot{x}_t \tag{A.2}$$

$$x_t = x_{t-\Delta t} + \Delta t \cdot \dot{x}_{t-\Delta t} \tag{A.3}$$

This representation is an approximation. The time step Δt places a bound on the highest frequency in X that can be represented by the discrete signal. At best, the Nyquist limit for a sampled system states that only frequencies below $\frac{1}{2\Delta t}$ can be accurately represented. If there is significant energy in the system above that boundary, aliasing effects will severely degrade the accuracy of the model predictions.

A.2 Six Degree of Freedom Motion

For real objects the story is complicated by the fact that real motion has six degrees of freedom: three translational and three rotational. The state vector from the 1-D case above

becomes the 6-D vector:

$$\mathbf{x} = \begin{bmatrix} T_x \\ T_y \\ T_z \\ R_x \\ R_y \\ R_z \end{bmatrix}$$

Similarly, the vector of forces becomes the 6-Dvector or forces and torques:

$$\mathbf{X} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

In equation A.1, the main complication arises in the formation of m which becomes a matrix, called the *mass matrix*. For a box of material with uniform density, a mass of m, aligned with the world coordinate system, with length a on the edge parallel to the X-axis, length b on the edge parallel to the Y-axis, length c on the edge parallel to the Z-axis, and with the center of mass at the origin, the mass matrix looks like this:

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{m(b^2+c^2)}{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{m(a^2+c^2)}{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{m(a^2+b^2)}{12} \end{bmatrix}$$

The formula for the rotational inertia of various objects can be found in any physics textbook. The mass matrix for more complex forms can be estimated numerically. The lower-right 3x3 block is responsible for representing the rotational inertia of the object and is called the inertial tensor, denoted $\mathbf{I}$. When the object rotates away from alignment with the global coordinate system, the inertial tensor must be adjusted. This is accomplished by applying the object rotation to the inertial tensor. The matrix $\mathbf{R}$ is orthonormal rotation matrix specified by the Euler parameters $\{R_x, R_y, R_z\}$. It can be directly from the state vector:

$$\mathbf{R} = \begin{bmatrix} c_1c_3 - s_1s_2s_3 & c_1s_3 + s_1s_2c_3 & -s_1 * c_2 \\ -c_2s_3 & -c_2s_3 & s_2 \\ s_1c_3 + c_1s_2s_3 & s_1s_3 - c_1s_2c_3 & c_1c_2 \end{bmatrix}$$

where $s_1 = \sin(R_x)$, $c_1 = \cos(R_x)$, and so on. The updated inertial tensor is:

$$\mathbf{I}_g = \mathbf{R}\mathbf{I}_l\mathbf{R}^T$$

A.2.1 Force Conversion

The preceding discussion assumes that the forces and torques, $\mathbf{X}$, are applied at the center of mass of the object. This is often not the case. Forces will be applied to distal parts of objects through attachment or contact points on the periphery. An external force $\mathbf{f}$ applied at a point $\mathbf{p}$ on the object needs to be converted into a body-centered force, $\mathbf{F}$ and torque $\boldsymbol{\tau}$. The vector between the center of the object $\mathbf{c}$ and the point $\mathbf{p}$ defines the moment-arm of the force and is designated by the vector $\mathbf{a} = \mathbf{p} - \mathbf{c}$. The resulting body centered forces are then:

$$\mathbf{F} = \mathbf{f} \tag{A.4}$$

$$\boldsymbol{\tau} = \mathbf{a} \times \mathbf{f} \tag{A.5}$$

A.2.2 Representations for Rotational Motion

Over small displacements Euler parameters may be treated linearly, so the extension of integration to 6-D motion is straightforward. However, computation involving larger rotational displacements are best carried out using the quaternion representation since that representation is compact, supplies a ready mechanism for renormalization to correct numerical drift, and doesn't suffer from the singularities associated with Euler parameters.

The conversion from a set of Euler parameters to the quaternion representation involves a vector norm and some trigonometry. If we represent the Euler parameters $\{R_x, R_y, R_z\}$ as a vector $\theta\boldsymbol{\omega}$ where θ is the magnitude and $\boldsymbol{\omega}$ is the unit length directional vector, then the quaternion $\mathbf{q}$ that represents the same rotation as $\theta\boldsymbol{\omega}$ is:

$$\mathbf{q} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)\boldsymbol{\omega} \end{bmatrix}$$

The inverse relationship, mapping quaternion $\mathbf{q}$ into the Euler parameters represented by $\theta\boldsymbol{\omega}$ is accomplished by a similar algorithm.

It is also possible to integrate in a linear fashion over rotations represented as quaternions. Because the length of a quaternion is constrained to be 1, it is even possible to renormalize the state vector after an integration to eliminate numerical drift. The only difficulty with this approach is the formulation of the mass matrix. It would be necessary to calculate the inertial tensor relative to the quaternion representation. To avoid this problem we utilize the Euler parameter representation for physical simulation and constraint satisfaction. When it is necessary to do computations on large rotations, measuring the angular displacement between two objects for example, we convert to the quaternion representation.

A.3 Constrained Motion

The basics of constrained motion are covered in Chapter 4. In this appendix we will walk through a simple example. Returning to our flat box from Section A.2 we can turn this into

a gyroscope by adding a constraint between the box and some inferred ground.

The first step in the modularized solution to this constraint problem discussed in Section 4.2.1 is the formulation of the connector. The connector is a point of contact removed from the center of mass of the object. As a software object it presents an idealized external interface to an object. For example, the force/torque translation discussed above is implemented inside the connector. Connectors may have orientation, or may be simply ideal points. The connection between the gyro and the constraint at the origin will be insensitive to orientation, so we will formulate a position-only connector. The positional state of the gyro is the 6-D vector $\mathbf{x}$, and the state of the connector will be the 3-D, position only vector $\mathbf{a}$.

The most important function of the connector is to provide an abstraction barrier between constraints and simulated objects. Constraints implemented as relationships between idealized points and the connectors provide that idealized interface to the more complex simulated objects. The position and velocity of a connector can be computed from the position and velocity of the host object and information about the relationship between the connector and the host object's local coordinate frame. A critical part of this abstraction is the ability to compute the Jacobian that relates variations in object state to variation in connector state:

$$\frac{\partial \mathbf{a}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{d\mathbf{a}_1}{d\mathbf{x}_1} & \frac{d\mathbf{a}_1}{d\mathbf{x}_2} & \frac{d\mathbf{a}_1}{d\mathbf{x}_3} & \frac{d\mathbf{a}_1}{d\mathbf{x}_4} & \frac{d\mathbf{a}_1}{d\mathbf{x}_5} & \frac{d\mathbf{a}_1}{d\mathbf{x}_6} \\ \frac{d\mathbf{a}_2}{d\mathbf{x}_1} & \frac{d\mathbf{a}_2}{d\mathbf{x}_2} & \frac{d\mathbf{a}_2}{d\mathbf{x}_3} & \frac{d\mathbf{a}_2}{d\mathbf{x}_4} & \frac{d\mathbf{a}_2}{d\mathbf{x}_5} & \frac{d\mathbf{a}_2}{d\mathbf{x}_6} \\ \frac{d\mathbf{a}_3}{d\mathbf{x}_1} & \frac{d\mathbf{a}_3}{d\mathbf{x}_2} & \frac{d\mathbf{a}_3}{d\mathbf{x}_3} & \frac{d\mathbf{a}_3}{d\mathbf{x}_4} & \frac{d\mathbf{a}_3}{d\mathbf{x}_5} & \frac{d\mathbf{a}_3}{d\mathbf{x}_6} \end{bmatrix} = [\mathbf{I}|\mathbf{J}_R]$$

The translational component of this Jacobian in the identity, while the rotational component, $\mathbf{J}_R$, varies according to the specific relationship between the object and the connector. A similar velocity Jacobian is formed to model the relationship between object velocity and connector velocity.

Constraints are now simply relationships between idealized connectors. All constraints are defined to be satisfied when the states of the constrained connectors is equal. The constrain forms Jacobian similar to the one above to model how variations in connector state affect variations in constraint state. In the case of the gyro we want to implement a ball joint constraint. The form of the Jacobian is particularly simple since the constraint is only concerned with position:

$$\frac{\partial \mathbf{c}}{\partial \mathbf{a}} = \mathbf{I}$$

The same implementation can be used for ball joints in 3-D simulations, or pin joints in 2-D simulations.

A.4 Simulation Results

The gyro simulation starts with a one meter square, ten centimeter thick slab weighing one kilogram with a one meter moment arm suspended at the origin of the global coordinate system. This arm is massless and is implemented by a connector attached to the slab one

meter away from the center of mass along the X-axis. The gyro begins one meter out the X-axis of the global coordinate system, so the connector begins coincident with the global origin, and the constraint is thus satisfied at the start of simulation. The Y-axis is up, away from gravity. The initial state of the gyro is zero translational velocity and a 100 radian per second rotational velocity along the axis. The state of the gyro some time after the beginning of simulation is illustrated in Figure A-1.

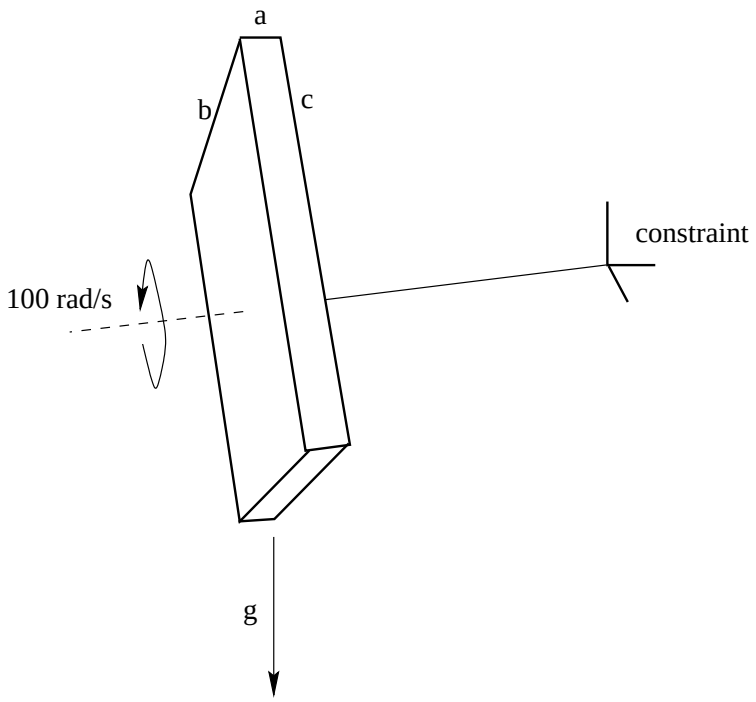

Figure A-1: The state of the gyro shortly after the beginning of simulation. $(a, b, c) = (10cm, 1m, 1m)$

As the simulation begins, gravitational force pulls downward on the gyro. This forces causes the gyro to precess around the point of constraint. This motion is not explicitly modeled. It arises naturally from the equations of motion. The X and Z position of the gyro are plotted in Figure A-2.

Other, higher order motions such as nutation are also captured by the system. Nutation is the wobbling of the gyro and is illustrated over the first nine seconds of simulation in Figure A-3.

On a 500MHz Alpha 21264 the gyro simulation is able to perform the constraint satisfaction plus forward integration loop at 9400Hz. The upper-body model runs at 600Hz. The gyro simulation runs significantly faster than the upper body model because the state vector is only 12-dimensional (not 60) and there is a single constraint (not 5).

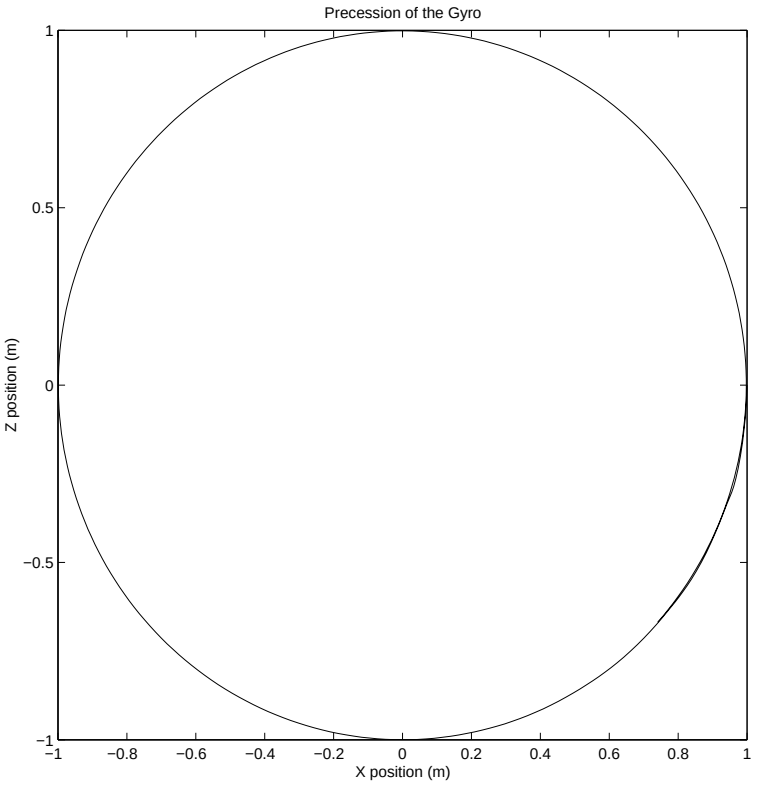

Figure A-2: Precession is not explicitly modeled, but It emerged from the equations of motion.

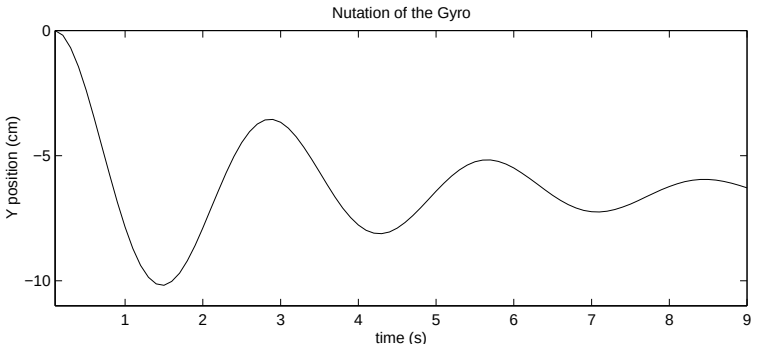

Figure A-3: Nutation, like precession, is a physically-realistic, higher-order motion that emerges from the simulation.

Appendix B

Classification

The classification techniques used in this work are derived from decision theory. This appendix will cover classification on the basis of decision theory, starting with the Bayes Risk Criterion[44, 50].

B.1 Definitions

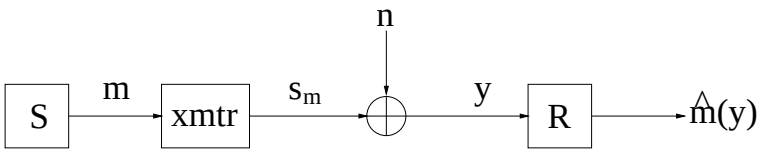

Figure B-1: Classification as a transmitter and receiver in noise.

Figure B-1 illustrates the basic theoretic model for classification. There is some sender, S that chooses a discrete message m. The message is sent with a transmitter that emits a signal S_m determined by the message. Gaussian noise is added to the signal during transmission producing the observation y. The receiver R must determine the best estimate of the original message given observation $\hat{m}(y)$.

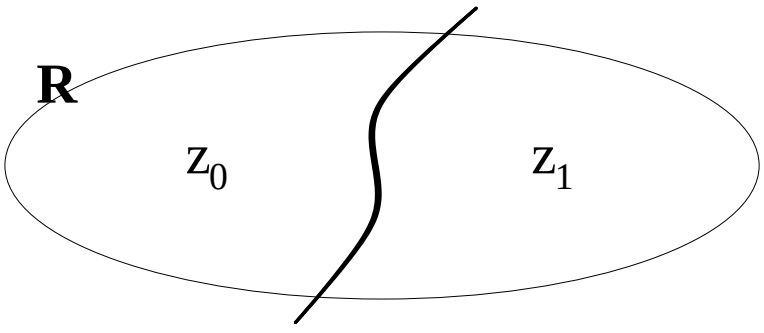

Figure B-2: Classification hypotheses cover the observations.

For simplicity we'll assume there are two messages, $m = 0$ and $m = 1$, although the

framework easily scales to multiple hypotheses. The receiver has only two hypotheses: hypothesis H_0 that the message is zero, and hypothesis H_1 that the message is one. The receiver knows the prior probabilities of these hypotheses being true, P_0 and P_1. The receiver also has access to the conditional densities of observations resulting from a message given $m = i$:

$$P_{y|H_i}(y|H_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-S_j)^2}{2\sigma^2}}$$

where σ^2 is the variance of the zero-mean transmission noise. The receiver must partition space of all observations into two domains: z_0 and z_1. When the observation falls in domain z_i then the receiver decides on hypothesis H_i:

$$H_i \Leftrightarrow y \in z_i$$

This situation is illustrated in Figure B-2. Finally, the receiver has a cost matrix $\mathbf{C}$ with elements c_{ij} that specifies the cost of deciding hypothesis H_i when the true hypothesis is actually H_j.

The Bayes Risk Criterion specifies that the expected risk of adopting a given classifier is:

$$E(c) = \sum_{j=0}^{1} \sum_{i=0}^{1} Pr(y \in z_i|H_j) c_{ij}$$

where the probability is computed over the domain:

$$Pr(y \in z_i|H_j) = \int_{z_i} P_{y|H_j}(y|H_j) \mathrm{d}y$$

The optimal classifier minimizes this risk.

B.2 Example

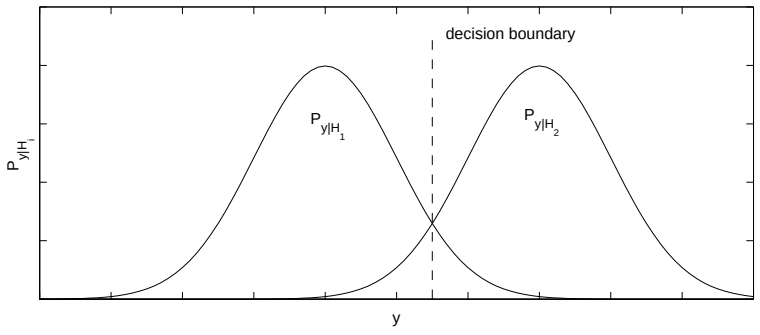

Figure B-3: Scalar classification example with two hypotheses.

For example, if the transmission noise is zero-mean with a variance σ^2, $S_0 = 0$ and $S_1 = 3\sigma$ then the relationship between the observation densities will be as illustrated in Figure B-3. The equal probability decision rule will occur half way between S_0 and S_1. To the left of this line $P_{y|H_0} > P_{y|H_1}$ while to the right of this line $P_{y|H_0} < P_{y|H_1}$.

The classifier makes a decision $\mathcal{L}(y)$ by weighing the probability of an observation against the prior information and the cost of making a wrong decision. The form of this decision is called the likelihood ratio test:

$$\mathcal{L}(y) = \frac{P_{y|H_1}(y|H_1)}{P_{y|H_0}(y|H_0)} \begin{array}{c} H_1 \\ > \\ < \\ H_0 \end{array} \frac{(c_{10} - c_{00})P_0}{(c_{01} - c_{11})P_1}$$

The combination of the cost matrix and the prior probabilities specify the classification threshold η:

$$\eta = \frac{(c_{10} - c_{00})P_0}{(c_{01} - c_{11})P_1}$$

If $\eta = 1$, then the classification boundary is as illustrated in Figure B-3. If the prior probability P_0 is greater than the prior probability P_1 then η will be biased toward the right, requiring more evidence in the observation to overcome the bias toward H_0. Alternately if c_{01} is greater than C_{10}, then the cost of mistakenly deciding H_0 is high, η will shift to the left, and more evidence evidence will be required to make the now risky decision H_0.

B.3 Specialized Classifiers

There are a few special cases of Bayesian classifier that are useful and widely applied. They represent simplifying assumptions about the problem structure.

B.3.1 Maximum *A Posteriori* Probability Classifier

The Maximum *A Posteriori* Probability (MAP) Classifier makes a minimum error probability decision. The simplifying assumption is that the cost matrix has the simple form of being 0 for all correct decisions and 1 for all incorrect decisions:

$$c_{ij} = 1 - \delta_{ij}$$

where δ_{ij} is the Dirac delta function:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

This simplifies the risk function to:

$$E(c) = P_0 Pr(y \in z_1|H_0) + P_1 Pr(y \in z_0|H_1)$$

and the likelihood ratio test becomes:

$$\Pr(H_1|y = Y) = \frac{P_1 P_{y|H_1}(Y|H_1)}{P_y(Y)} \begin{array}{c} H_1 \\ > \\ < \\ H_0 \end{array} \frac{P_0 P_{y|H_0}(Y|H_0)}{P_y(Y)} \frac{(c_{10} - c_{00})P_0}{(c_{01} - c_{11})P_1} = \Pr(H_0|y = Y)$$

B.3.2 Maximum Likelihood Classifier

The Maximum Likelihood Classifier makes a decision based solely on the likelihood of the observation. The simplifying assumption is that the prior probabilities are equal, The likelihood ration test becomes correspondingly simpler as the prior probabilities P_0 and P_1 are no longer necessary:

$$P_{y|H_1}(Y|H_1) \underset{H_0}{\overset{H_1}{\gtrless}} P_{y|H_0}(Y|H_0)$$

Appendix C

Classical Kalman Filtering

The Kalman filter is the optimal formulation of the recursive framework under a certain set of assumptions. This appendix covers the formulation of the discrete, multiple input, multiple output, time varying, linear Kalman filter. This derivation formulates the Kalman filter as a linear minimum mean squared error estimator (LMMSEE) of the system state[49, 55].

C.1 LMMSEE

Before we derive the Kalman filter, we begin with the definition of a linear minimum mean squared error estimator and some of its properties. Given a zero-mean random variable y, a vector of n measurements $\mathbf{x}$:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

the variance of y, σ_y^2, the covariance of $\mathbf{x}$ defined as:

$$\mathbf{\Lambda_x} \triangleq E\left[\mathbf{x}\mathbf{x}^T\right]$$

and the cross-covariance between $\mathbf{x}$ and y, defined as:

$$\mathbf{\Lambda}_{y\mathbf{x}} \triangleq E\left[y\mathbf{x}^T\right]$$

we wish to compute the best estimator $\mathbf{a}$ such that:

$$\hat{\mathbf{a}} = \arg\min E\left[y - \mathbf{x}^T\mathbf{a}\right]$$

This estimator is known as the LMMSEE and has the form:

$$\hat{\mathbf{a}} = \mathbf{\Lambda}_{y\mathbf{x}}\mathbf{\Lambda}_{\mathbf{x}}^{-1}$$

An important property of this estimator is that the error is perpendicular to the data:

$$(y - \mathbf{x}^T\mathbf{a}) \perp \mathbf{x}^T$$

This is due to the fact that $\mathbf{x}^T\mathbf{a}$ is the closest point to y in the subspace spanned by $\mathbf{x}$. Proofs of these claims are omitted, but are readily available in the literature.

Another important property of LMMSEE's that we will need is that the LMMSEE estimator of a random variable y based jointly on $\mathbf{x}$ and $\mathbf{z}$ is separable if $\mathbf{x} \perp \mathbf{z}$ that is:

$$\mathbf{x} \perp \mathbf{z} \Rightarrow \hat{y}(\mathbf{x}, \mathbf{z}) = \hat{y}(\mathbf{x}) + \hat{y}(\mathbf{z})$$

This fact derives directly from the definition of the LMMSEE:

$$\begin{aligned}
\hat{y}(\mathbf{x}, \mathbf{z}) &= \mathbf{\Lambda}_{y(\mathbf{xz})}\mathbf{\Lambda}_{(\mathbf{xz})}{}^{-1} \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} \\
&= \begin{bmatrix} E[y\mathbf{x}^T] E[y\mathbf{z}^T] \end{bmatrix} \begin{bmatrix} \mathbf{\Lambda}_{\mathbf{x}}^{-1} & \\ & \mathbf{\Lambda}_{\mathbf{z}}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} \\
&= \mathbf{\Lambda}_{y\mathbf{x}}\mathbf{\Lambda}_{\mathbf{x}}^{-1}\mathbf{x} + \mathbf{\Lambda}_{y\mathbf{z}}\mathbf{\Lambda}_{\mathbf{z}}^{-1}\mathbf{z} \\
&= \hat{y}(\mathbf{x}) + \hat{y}(\mathbf{z})
\end{aligned}$$

C.2 Definition

The filter observes a dynamic system that is defined as:

$$\begin{aligned}
\mathbf{x}_{t+1} &= \mathbf{\Phi}_t\mathbf{x}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{L}_t\boldsymbol{\xi}_t \\
\mathbf{y}_t &= \mathbf{H}_t\mathbf{x}_t + \boldsymbol{\theta}_t
\end{aligned}$$

where $\mathbf{x}_t$ is the current state of the system. The linear, time-varying dynamic model $\mathbf{\Phi}_t$ is used to compute the next state $\mathbf{x}_{t+1}$ in combination with the control signal $\mathbf{u}_t$, which is weighted by the time-varying matrix $\mathbf{B}_t$, and the system process noise $\boldsymbol{\xi}_t$ weighted by $\mathbf{L}_t$. Observations on the system $\mathbf{y}_t$ are made by a time-varying linear function of system state represented by $\mathbf{H}_t$ and include measurement noise $\boldsymbol{\theta}_t$. The random variables $\boldsymbol{\xi}_t$ and $\boldsymbol{\theta}_t$ have the following properties:

$$\begin{aligned}
E[\boldsymbol{\xi}_t] &= 0 \\
E[\boldsymbol{\theta}_t] &= 0 \\
E[\boldsymbol{\xi}_t\boldsymbol{\xi}_\tau^T] &= \mathbf{\Xi}_t\delta_{t,\tau} \\
E[\boldsymbol{\theta}_t\boldsymbol{\theta}_\tau^T] &= \mathbf{\Theta}_t\delta_{t,\tau} \\
E[\boldsymbol{\xi}_t\boldsymbol{\theta}_\tau^T] &= 0
\end{aligned}$$

The goal of the filter is to compute the estimate $\hat{\mathbf{x}}_{t|s}$ which is defined as the LMMSEE of $\mathbf{x}_t$ given the measurements $\mathbf{y}_\tau$ for $\tau \leq s$. Similarly $\hat{\mathbf{y}}_{t|s}$ is defined as the LMMSEE of $\mathbf{y}_t$ given

$\mathbf{y}_\tau$ for $\tau \leq s$. The estimation error to be minimized is:

$$\tilde{\mathbf{x}}_{t|s} \stackrel{\triangle}{=} \mathbf{x}_t - \hat{\mathbf{x}}_{t|s}$$

and the error covariance is a crucial entity in the formulation of the Kalman filter:

$$\mathbf{\Sigma}_{t|s} \stackrel{\triangle}{=} \mathbf{\Lambda}_{\tilde{\mathbf{x}}_{t|s}} = E[\tilde{\mathbf{x}}_{t|s}\tilde{\mathbf{x}}_{t|s}^T]$$

The filter starts with a prior estimate of systems state, $\hat{\mathbf{x}}_{0|0}$. The filter is called recursive because the computation of the new state estimate, $\hat{\mathbf{x}}_{t+1|t+1}$, only relies on the previous state prediction, $\hat{\mathbf{x}}_{t+1|t}$, and $\mathbf{y}_{t+1}$, the new observation:

$$\begin{aligned}
\hat{\mathbf{x}}_{t+1|t+1} &= \mathbf{x}_{t+1|t} + \mathbf{K}_{t+1}[\mathbf{y}_{t+1} - \hat{\mathbf{y}}_{t+1|t}] \\
\hat{\mathbf{y}}_{t+1|t} &= \mathbf{H}_{t+1}\hat{\mathbf{x}}_{t+1|t} \\
\hat{\mathbf{x}}_{t+1|t} &= \mathbf{\Phi}_t\hat{\mathbf{x}}_t + \mathbf{B}_t\mathbf{u}_t
\end{aligned}$$

The Kalman gain matrix $\mathbf{K}_{t+1}$ maps the error between the predicted observation and the actual observation into a state update. The current value of $\mathbf{K}_{t+1}$ is a combination of the predicted error covariance, the observation noise covariance and the observation model:

$$\mathbf{K}_{t+1} = \mathbf{\Sigma}_{t+1|t}\mathbf{H}_t^T \left[\mathbf{H}_t\mathbf{\Sigma}_{t+1|t}\mathbf{H}_t^T + \mathbf{\Theta}_{t+1}\right]^{-1}$$

The error covariance itself it estimated recursively starting from the prior covariance:

$$\mathbf{\Sigma}_{0|0} \stackrel{\triangle}{=} \mathbf{\Lambda}_{\tilde{\mathbf{x}}_{0|0}}$$

The estimated error covariance of the state prediction is a function of the estimated error covariance of the state estimate:

$$\mathbf{\Sigma}_{t+1|t} = \mathbf{\Phi}_t\mathbf{\Sigma}_{t|t}\mathbf{\Phi}_t^T + \mathbf{L}_t\mathbf{\Xi}_t\mathbf{L}_t^T$$

and the estimated error covariance of the state estimate is recursively defined in terms of the estimated error covariance of the previous state prediction:

$$\mathbf{\Sigma}_{t+1|t+1} = \left[\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H}_{t+1}\right]\mathbf{\Sigma}_{t+1|t}$$

Figure C-1 illustrated the recursive flow of information in the Kalman filter.

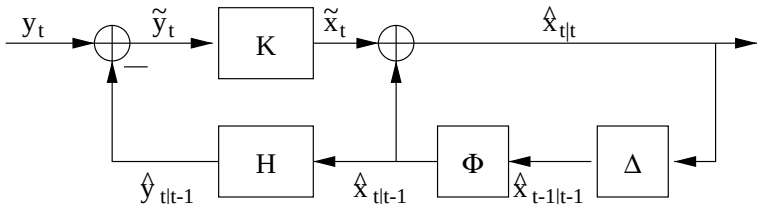

Figure C-1: Overall structure of the Kalman filter.

It is also useful to have a symbol to represent the observation error, also know as the

innovation:

$$\boldsymbol{\nu}_{t+1} = \mathbf{y}_{t+1} - \hat{\mathbf{y}}_{t+1|t}$$

and the sequence of innovations:

$$\mathbf{N}_{t+1} = \{\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_{t+1}\}$$

C.3 Derivation

The prediction steps of the Kalman filter are straightforward applications of linear algebra to random variables. Applying the system state dynamics:

$$\mathbf{x}_{t+1} = \boldsymbol{\Phi}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t + \mathbf{L}_t \boldsymbol{\xi}_t$$

to the state estimate yields:

$$\begin{aligned} \hat{\mathbf{x}}_{t+1|t} &= E[\boldsymbol{\Phi}_t \hat{\mathbf{x}}_{t|t} + \mathbf{B}_t \mathbf{u}_t + \mathbf{L}_t \boldsymbol{\xi}_t] \\ &= \boldsymbol{\Phi}_t \hat{\mathbf{x}}_t + \mathbf{B}_t \mathbf{u}_t \end{aligned}$$

because the expected value of the $\mathbf{L}_t \boldsymbol{\xi}_t$ term is 0 since $\boldsymbol{\xi}_t$ is defined to be a zero-mean random variable. The prediction of the error covariance proceeds similarly:

$$\begin{aligned} \boldsymbol{\Sigma}_{t+1|t} &= E\left[(\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1|t})(\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1|t})^T\right] \\ &= E\left[(\boldsymbol{\Phi}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t + \mathbf{L}_t \boldsymbol{\xi}_t - \boldsymbol{\Phi}_t \hat{\mathbf{x}}_{t|t} - \mathbf{B}_t \mathbf{u}_t)(\boldsymbol{\Phi}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t + \mathbf{L}_t \boldsymbol{\xi}_t - \boldsymbol{\Phi}_t \hat{\mathbf{x}}_{t|t} - \mathbf{B}_t \mathbf{u}_t)^T\right] \\ &= E\left[(\boldsymbol{\Phi}_t(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}) + \mathbf{L}_t \boldsymbol{\xi}_t)(\boldsymbol{\Phi}_t(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}) + \mathbf{L}_t \boldsymbol{\xi}_t)^T\right] \\ &= E\left[(\boldsymbol{\Phi}_t \tilde{\mathbf{x}}_t + \mathbf{L}_t \boldsymbol{\xi}_t)(\boldsymbol{\Phi}_t \tilde{\mathbf{x}}_t + \mathbf{L}_t \boldsymbol{\xi}_t)^T\right] \\ &= E\left[(\boldsymbol{\Phi}_t \tilde{\mathbf{x}}_t \tilde{\mathbf{x}}_t^T \boldsymbol{\Phi}_t^T + \mathbf{L}_t \boldsymbol{\xi}_t \boldsymbol{\xi}_t^T \mathbf{L}_t^T + \boldsymbol{\Phi}_t \tilde{\mathbf{x}}_t \boldsymbol{\xi}_t^T \mathbf{L}_t^T + \mathbf{L}_t \boldsymbol{\xi}_t \tilde{\mathbf{x}}_t^T \boldsymbol{\Phi}_t^T\right] \\ &= \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{t|t} \boldsymbol{\Phi}_t^T + \mathbf{L}_t \boldsymbol{\Xi}_t \mathbf{L}_t^T \end{aligned}$$

The expected value of the cross terms go to zero since the error is independent of the data in a LMMSEE.

The next step of the Kalman filter is the update, where the new observation is used to update the state estimate and the error covariance estimate. The current state is the state prediction plus the error:

$$\mathbf{x}_{t+1} = \hat{\mathbf{x}}_{t+1|t} + \tilde{\mathbf{x}}_{t+1|t}$$

The LMMSE estimate of $\mathbf{x}_{t+1}$ breaks down into two parts:

$$\begin{aligned} \text{LMMSEE}(\mathbf{x}_{t+1}|\mathbf{N}_t, \boldsymbol{\nu}_{t+1}) &= \text{LMMSEE}(\hat{\mathbf{x}}_{t+1|t}|\mathbf{N}_t) + \text{LMMSEE}(\hat{\mathbf{x}}_{t+1|t}|\boldsymbol{\nu}_{t+1}) + \\ &\quad \text{LMMSEE}(\tilde{\mathbf{x}}_{t+1|t}|\mathbf{N}_t) + \text{LMMSEE}(\tilde{\mathbf{x}}_{t+1|t}|\boldsymbol{\nu}_{t+1}) \\ &= \hat{\mathbf{x}}_{t+1|t} + \boldsymbol{\Lambda}_{\tilde{\mathbf{x}}_{t+1|t}\boldsymbol{\nu}_{t+1}} \boldsymbol{\Lambda}_{\boldsymbol{\nu}_{t+1}}^{-1} \boldsymbol{\nu}_{t+1} \end{aligned}$$

The middle two terms go to zero because $\hat{\mathbf{x}}_{t+1|t} \perp \boldsymbol{\nu}_{t+1}$ and $\tilde{\mathbf{x}}_{t+1|t} \perp \mathbf{N}_t$. The form of this

equation matches the form of the update equation sated above and implies that:

$$\mathbf{K}_{t+1} = \mathbf{\Lambda}_{\tilde{\mathbf{x}}_{t+1|t}\boldsymbol{\nu}_{t+1}}\mathbf{\Lambda}_{\boldsymbol{\nu}_{t+1}}^{-1}$$

The derivation of these values follows directly from the definitions. First we derive the cross-covariance between the error and the innovation:

$$\begin{aligned}
\mathbf{\Lambda}_{\tilde{\mathbf{x}}_{t+1|t}\boldsymbol{\nu}_{t+1}} &= E\left[\tilde{\mathbf{x}}_{t+1|t}\boldsymbol{\nu}_{t+1}^T\right] \\
&= E\left[(\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1|t})(\mathbf{H}_{t+1}[\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1|t}] + \boldsymbol{\theta}_{t+1})^T\right] \\
&= E\left[\tilde{\mathbf{x}}_{t+1|t}\tilde{\mathbf{x}}_{t+1|t}^T\mathbf{H}_{t+1}^T + \mathbf{x}_{t+1}\boldsymbol{\theta}_{t+1} - \hat{\mathbf{x}}_{t+1|t}\boldsymbol{\theta}_{t+1}\right] \\
&= \mathbf{\Sigma}_{t+1|t}\mathbf{H}_{t+1}^T
\end{aligned}$$

Then we derive the covariance of the innovations:

$$\begin{aligned}
\mathbf{\Lambda}_{\boldsymbol{\nu}_{t+1}} &= E\left[\boldsymbol{\nu}_{t+1}\boldsymbol{\nu}_{t+1}^T\right] \\
&= E\left[(\mathbf{y}_{t+1} - \hat{\mathbf{y}}_{t+1|t})(\mathbf{y}_{t+1} - \hat{\mathbf{y}}_{t+1|t})^T\right] \\
&= E\left[(\mathbf{H}_{t+1}\mathbf{x}_{t+1} - \mathbf{H}_{t+1}\hat{\mathbf{x}}_{t+1|t} - \boldsymbol{\theta}_{t+1})(\mathbf{H}_{t+1}\mathbf{x}_{t+1} - \mathbf{H}_{t+1}\hat{\mathbf{x}}_{t+1|t} - \boldsymbol{\theta}_{t+1})^T\right] \\
&= E\left[(\mathbf{H}_{t+1}\tilde{\mathbf{x}}_{t+1|t} - \boldsymbol{\theta}_{t+1})(\mathbf{H}_{t+1}\tilde{\mathbf{x}}_{t+1|t} - \boldsymbol{\theta}_{t+1})^T\right] \\
&= E\left[\mathbf{H}_{t+1}\tilde{\mathbf{x}}_{t+1|t}\tilde{\mathbf{x}}_{t+1|t}^T\mathbf{H}_{t+1}^T - \mathbf{H}_{t+1}\tilde{\mathbf{x}}_{t+1|t}\boldsymbol{\theta}_{t+1}^T - \boldsymbol{\theta}_{t+1}\tilde{\mathbf{x}}_{t+1|t}^T\mathbf{H}_{t+1}^T + \boldsymbol{\theta}_{t+1}\boldsymbol{\theta}_{t+1}^T\right] \\
&= \mathbf{H}_{t+1}\mathbf{\Sigma}_{t+1|t}\mathbf{H}_{t+1}^T + \mathbf{\Theta}_{t+1}
\end{aligned}$$

So now we can write down the derivation of the Kalman gain matrix as simply the expansion of the LMMSE estimator of $\tilde{\mathbf{x}}_{t+1|t}$ given $\boldsymbol{\nu}_{t+1}$:

$$\begin{aligned}
\mathbf{K}_{t+1} &= \mathbf{\Lambda}_{\tilde{\mathbf{x}}_{t+1|t}\boldsymbol{\nu}_{t+1}}\mathbf{\Lambda}_{\boldsymbol{\nu}_{t+1}}^{-1} \\
&= \mathbf{\Sigma}_{t+1|t}\mathbf{H}_t^T\left[\mathbf{H}_t\mathbf{\Sigma}_{t+1|t}\mathbf{H}_t^T + \mathbf{\Theta}_{t+1}\right]^{-1}
\end{aligned}$$

The only remaining derivation is the equation to update the error covariance:

$$\begin{aligned}
\mathbf{\Sigma}_{t+1|t+1} &= E\left[\tilde{\mathbf{x}}_{t+1|t+1}\tilde{\mathbf{x}}_{t+1|t+1}^T\right] \\
&= E\left[(\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1|t+1})(\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1|t+1})^T\right] \\
&= E\left[(\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1|t} - \mathbf{K}_{t+1}\boldsymbol{\nu}_{t+1})(\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1|t} - \mathbf{K}_{t+1}\boldsymbol{\nu}_{t+1})^T\right] \\
&= E\left[(\tilde{\mathbf{x}}_{t+1|t} - \mathbf{K}_{t+1}\mathbf{H}_{t+1}\tilde{\mathbf{x}}_{t+1|t})(\tilde{\mathbf{x}}_{t+1|t} - \mathbf{K}_{t+1}\mathbf{H}_{t+1}\tilde{\mathbf{x}}_{t+1|t})^T\right] \\
&= E\left[([\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H}_{t+1}]\tilde{\mathbf{x}}_{t+1|t})([\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H}_{t+1}]\tilde{\mathbf{x}}_{t+1|t})^T\right] \\
&= [\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H}_{t+1}]\,\mathbf{\Sigma}_{t+1|t}
\end{aligned}$$

C.4 Summary

So we see that the Kalman filter is a linear minimum mean squared error estimator of the system state given a stream of observations. The recursive nature falls out of the fact that LMMSE estimators generate estimates with error that is perpendicular to the data. This means that current estimate errors are uncorrelated with past data, and the Kalman filter can compute new estimates based only on the previous estimate and the new innovation.

Appendix D

Perspective Transform Estimation

Estimating perspective transforms from data is useful for mapping deictic gestures into screen coordinates. as a first step to them with virtual objects displayed on the screen. The formulation of a linear estimator is somewhat obtuse, and is presented here as an aide to the reader[13, 53].

The task is to find a transform that maps one arbitrary 3-D quadrilateral into another. One approach is use the perspective transform. Figure D-1 illustrates the situation: the first quadrilateral is shown on the Image Plane and the second quadrilateral is shown on the World Plane.

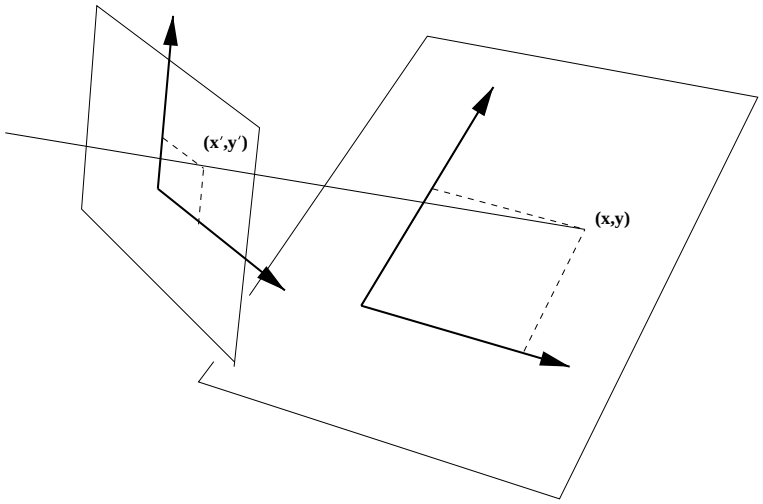

Figure D-1: The perspective transformation of a quadrilateral onto a plane. The original point (x, y) is projected into the plane as (x', y')

Why use this transform over other mappings? It has the interesting property that it maps straight lines to straight lines. Other transforms of this nature introduce distortions that cause some straight lines in one space to map to curves in the other space.

So the task is, given the coordinates of the four corners of the first quadrilateral, and

the coordinates of the four corners of the second quadrilateral, compute the perspective transform that maps a new point in the first quadrilateral onto the appropriate position on the second quadrilateral.

D.1 Deriving the Criminisi Equation

The solution is stated in Section 3.1 of [13], but the derivation is a bit lacking. I will briefly sketch the derivation here to make the solution more appetizing. We start with the claim that the camera model could be written as $\mathbf{I} = \mathbf{HW}$, where $\mathbf{W}$ is vector of world plane coordinates, $\mathbf{I}$ is the vector of image plane coordinates, and $\mathbf{H}$ is a matrix transform. We can write the this form out in more detail as:

$$\begin{bmatrix} \omega x' \\ \omega y' \\ \omega \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Then if we realize that ω is actually:

$$\omega = gx + hy + 1$$

we can rewrite the equation in a way that exposes its true non-linear form where the numerator supplies the parameters needed for affine transformation, and the denominator allows for the non-linear effect of perspective:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \frac{\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}{\begin{bmatrix} g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}$$

This is equivalent to the possibly more familiar, non-vector form of the perspective transform:

$$\begin{aligned} x' &= \frac{ax + by + c}{gx + hy + 1} \\ y' &= \frac{dx + ey + f}{gx + hy + 1} \end{aligned}$$

By multiplying each side of the equation by the denominator we get:

$$\begin{aligned} x'(gx + hy + 1) &= ax + by + c \\ y'(gx + hy + 1) &= dx + ey + f \end{aligned}$$

and multiplying through by x' and y' gives us:

$$gx'x + hx'y + x' = ax + by + c$$

$$gy'x + hy'y + y' = dx + ey + f$$

Now we isolate the naked x' and y' terms on the left:

$$x' = ax + by + c - gx'x - hx'y$$
$$y' = dx + ey + f - gy'x - hy'y$$

If we add in some zero terms:

$$\begin{aligned} x' &= ax + by + ac - 0d + 0e + 0f - x'xg - x'yh \\ y' &= 0a + 0b + 0c + xd + yd + af - y'xg - y'yh \end{aligned}$$

then it becomes more clear that this is the product of a matrix and a vector:

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \\ & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ \vdots \end{bmatrix}$$

and we have reached the previously mysterious point that Criminisi leaps to in a single step:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 \\ x_2 & y_2 & 2 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 2 & -y'_2x_2 & -y'_2y_2 \\ & & & & \vdots & & & \\ x_n & y_n & n & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n \\ 0 & 0 & 0 & x_n & y_n & n & -y'_nx_n & -y'_ny_n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

D.2 The Solution

The form of this equation is $\mathbf{A}\boldsymbol{\lambda} = \mathbf{b}$, and it can be solved by several methods. Given that the data will contain noise, we need the least squares estimate of the parameter vector $\boldsymbol{\lambda}$ that best satisfies the linear relationship between the matrix $\mathbf{A}$ and the vector of output coordinates $\mathbf{b}$. The simplest method, although not the most numerically stable, is the pseudo-inverse:

$$\begin{aligned} \mathbf{A}\boldsymbol{\lambda} &= \mathbf{b} \\ \mathbf{A}^T\mathbf{A}\boldsymbol{\lambda} &= \mathbf{A}^T\mathbf{b} \\ \boldsymbol{\lambda} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \end{aligned}$$

The rest of [13] discusses issues of measurement noise and estimation uncertainty that are important when choosing the right estimation method for a given application.

Acknowledgments

Life is a sequence of punctuated equilibria. Choices we make often lead to periods of relative stability as the consequences of our choices play out. Eventually these choices cease to have meaning and we must make new choices. We make little choices everyday, but occasionally we come to points in our lives where several momentous choices all expire at once. A friend of mine calls this situation a *node*. I hate nodes. I think most people do. They are periods of great opportunity, but often there is too little information and too little time to make right decisions. Life is not a game of chess. There is not perfect knowledge. We can only hope to make the optimal choices with imperfect data in less time than we would like. This dissertation is a testament to the complexity of a philosophically similar task.

Complex tasks cry out for procrastination, so people spend time reflecting on how good they had it only a couple of months ago. They write sappy reminiscences, and over-wrought farewells to all the people they should have spent more time appreciating along the way. It makes them feel better. It steels them for the coming node, like gathering up idols and talismans before a storm. I am no different, so brace yourself.

First and foremost, I want to thank my wife, Donna. Her love, support, and (sometimes not so) gentle encouragement have made this journey physically, mentally, and emotionally more healthy than it could have been. Sandy recently referred to Donna as my secret weapon. That is not far from the truth. I can only hope that I have the strength to be as patient and supportive during the coming fellowship years.

I also want to express my deepest gratitude to my advisor, Sandy Pentland. I am very fortunate that we crossed paths so many years ago, because the experiences I've had were so much more than anything that I had hoped for from graduate school. I'll count myself successful if I go on to create an environment that is half as stimulating and supportive as Vismod.

My committee members Eric Grimson and Trevor Darrell also deserve hearty thanks for comments and guidance that have helped make this dissertation a significantly better document than it was at first draft. Monica Bell has been an indispensable source of guidance for all matters of document format.

There are some prominent figures in my past who deserve mention here for their help along the way: Richard Bolt (thank you for hiring the all-too-excited freshman so many years ago), Peter Elias (thank you for your guidance: now the Ph.D's actually *mine*), and Marc Raibert (thank you for your help and guidance: maybe I don't need the letters after my name to do cool things, but now that I have them, it's a moot point).

During my time at the Media Lab, I've had the pleasure of interacting with some completely incredible people, and I'm better for knowing them. They've all had an impact on me: Dave Becker's exuberance, Michael "Particle" Johnson's erstwhile passion, Ali Azarbayejani's integrity, Lee Campbell's activism, Erik Trimble's existentialism, Martin Friedmann's perception, Yuri Ivanov's courage, and Andy Wilson's ineffable nature. The list goes on almost too long to believe: Sumit Basu, Dave Berger, Bruce Blumberg, Judy Bornstein, Brent Britton, Amy Bruckman, Bill Butera, Brian Clarkson, Jim Davis, Richard DeVaul, Lenny Foner, Jennifer Healey, Tony Jebara, Darrin Jewell, Matt Krom, Baback Moghaddam, Karen "Evil Queen" Navarro, Sourabh Niogi, Mary Obelnicki, Nuria Oliver, Egon Pasztor, Ali Rahimi, Carson Reynolds, Brad Rhodes, Deb Roy, Ken Russell, Robin Simone, Bernt Schiele, Stephen Schwartz, Flavia Sparacino, Thad Starner, Kris Thorisson, Bill Tomlinson, Josh Wachman, Yair Weiss, and Alan Wexelblat. There are also several groups in and around the Lab that have made graduate school a richer experience: the Thirsty Magic crew, the Diablo folk, and the many users of antiquated computing hardware (you know who you are).

Over the years I've had the honor of working with quite a few staff members and UROP students. I hope they have learned at least a little bit from me, as I've certainly learned quite a lot from interacting with them: William Abernathy, David Bailey, Matthew Belmonte, Cyrus Eyster, John Gu, Marc Lebovitz, Michael Li, and Theodore Weatherly.

Donna didn't keep me sane and healthy single handedly. I am fortunate to have a wonderfully supportive extended family: Rosalyn Foster, Randall Foster, Alexandria Foster, Colton Foster, Charlie Hackett, Sandie Hackett, Charles Hackett, Khodarahm Khodarahmi, Judith Khodarahmi, Susanne Khodarahmi, Stephen Uccello, and John Khodarahmi. I greatly appreciate all their love, support and, of course, the occasional reality checks (which often come as in hockey, rather than anything fitting the more gentle aspects of the word).

Finally, and most importantly, I would like to thank my parents Mary Louise and Richard Wren for their support and encouragement. I wouldn't have made it this far but for their belief in me, their sacrifices, their good example, and their love.

Bibliography

[1] M. Athans and C. B. Chang. Adaptive estimation and parameter identification using multiple model estimation algorithm. Technical Report 1976-28, Massachusetts Institute of Technology Lincoln Laboratory, Lexington, Massachusetts, USA, June 1976. Group 32.

[2] Ali Azarbayejani and Alex Pentland. Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features. In *Proceedings of 13th ICPR*, Vienna, Austria, August 1996. IEEE Computer Society Press.

[3] Ali Jerome Azarbayejani. *Nonlinear Probabilistic Estimation of 3-D Geometry from Images*. PhD thesis, Massachusetts Institute of Technology, February 1997. Media Arts and Sciences.

[4] A. Baumberg and D. Hogg. An efficient method for contour tracking using active shape models. In *Proceeding of the Workshop on Motion of Nonrigid and Articulated Objects*. IEEE Computer Society, 1994.

[5] David A. Becker. Sensei: A real-time recognition, feedback, and training system for t'ai chi gestures. Master's thesis, Massachusetts Institute of Technology Media Laboratory, 1997. also MIT Media Lab Perceptual Computing TR426.

[6] B. Blumberg. Action-selection in hamsterdam: Lessons from ethology. In *The Proceedings of the 3rd International Conference on the Simulation of Adaptive Behavior*, Brighton, August 1994.

[7] R. A. Bolt. 'put-that-there': Voice and gesture at the graphics interface. In *Computer Graphics Proceedings, SIGGRAPH 1980,*, volume 14, pages 262–70, July 1980.

[8] Christoph Bregler. Learning and recognizing human dynamics in video sequences. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, June 1997.

[9] Christoph Bregler and Jitendra Malik. Video motion capture. Technical Report UCB/CSD-97-973, University of California, Berkeley, 1997.

[10] Lee W. Campbell, David A. Becker, Ali Azarbayejani, Aaron Bobick, and Alex Pentland. Invariant features for 3-d gesture recognition. In *Second International Conference on Face and Gesture Recognition*, pages 157–62, Killington, VT, USA, 1996.

[11] Tat-Jen Cham and James M. Rehg. A multiple hypothesis approach to figure tracking. In *Workshop on Perceptual User Interfaces*, San Francisco, Calif., November 1998.

[12] Brian P. Clarkson and Alex Pentland. Unsupervised clustering of ambulatory audio and video. In *Proceedings of the International Conference of Acoustics Speech and Signal Processing*, Phoenix, Arizona, 1999.

[13] A. Criminisi, I. Reid, and A. Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999.

[14] Quentin Delamarre and Olivier Faugeras. 3d articulated models and multi-view tracking with silhouettes. In *Proceedings of the Seventh International Conference on Computer Vision*. IEEE, 1999.

[15] J. Deutscher, B. North, B. Bascle, and A. Bake. Tracking through singularities and discontinuities by random sampling. In *Proceedings of the Seventh International Conference on Computer Vision*. IEEE, 1999.

[16] Ernst D. Dickmanns and Birger D. Mysliwetz. Recursive 3-d road and relative ego-state recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):199–213, February 1992.

[17] Roy Featherstone. *Coordinate Systems and Efficiency*, chapter 8, pages 129–152. Kluwer Academic Publishers, 1984.

[18] Martin Friedmann, Thad Starner, and Alex Pentland. Synchronization in virtual realities. *Presence*, 1(1):139–144, 1991.

[19] Martin Friedmann, Thad Starner, and Alex Pentland. Device synchronization using an optimal linear filter. In H. Jones, editor, *Virtual Reality Systems*. Academic Press, 1993.

[20] D. M. Gavrila and L. S. Davis. Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. In *International Workshop on Automatic Face- and Gesture-Recognition*. IEEE Computer Society, 1995. Zurich.

[21] D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *CVPR96*. IEEE Computer Society, 1996.

[22] Luis Goncalves, Enrico Di Bernardo, Enrico Ursella, and Pietro Perona. Monocular tracking of the human arm in 3d. In *International Conference on Computer Vision*, Cambridge, MA, June 1995.

[23] Thanarat Horprasert, Ismail Haritaoglu, David Harwood, Larry S. Davis, Christopher R. Wren, and Alex P. Pentland. Real-time 3d motion capture. In *Workshop on Perceptual User Interfaces*, San Francisco, Calif., November 1998.

[24] Marcus J. Huber and Tedd Hadley. Multiple roles, multiple teams, dynamic environment: Autonomous netrek agents. In *Autonomous Agents '97*. ACM SIGART, 1997. http://sigart.acm.org:80/proceedings/agents97/.

[25] Peter C. Hughes. *Spacecraft Attitude Dynamics*. John Wiley & Sons, 1986.

[26] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. European Conference on Computer Vision*, pages 343–356, Cambridge, UK, 1996.

[27] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *Int. J. Computer Vision*, 1998. in press.

[28] Michael Isard and Andrew Blake. A mixed-state condensation tracker with automatic model-switching. In *Proc 6th Int. Conf. Computer Vision*, 1998.

[29] I. Kakadiaris, D. Metaxas, and R. Bajcsy. Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach. In *CVPR94*, pages 980–984, 1994.

[30] Ioannis Kakadiaris and Dimitris Metaxas. Vision-based animation of digital humans. In *Computer Animation*, pages 144–152. IEEE Computer Society Press, 1998.

[31] John MacCormick and Andrew Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proceedings of the Seventh International Conference on Computer Vision*. IEEE, 1999.

[32] Dimitris Metaxas and Dimitris Terzopoulos. Shape and non-rigid motion estimation through physics-based synthesis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993.

[33] Parviz E. Nikravesh. *Spatial Dynamics*, chapter 11, pages 289–300. Prentice-Hall, 1988.

[34] K. Oatley, G. D. Sullivan, and D. Hogg. Drawing visual conclusions from analogy: preprocessing, cues and schemata in the perception of three dimensional objects. *Journal of Intelligent Systems*, 1(2):97–133, 1988.

[35] J. O'Rourke and N.I. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2(6):522–536, November 1980.

[36] Vladimir Pavlović, James M. Rehg, Tat-Jen Cham, and Kevin P. Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. In *Proceedings of the Seventh International Conference on Computer Vision*. IEEE, 1999.

[37] A. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):730–742, July 1991.

[38] Alex Pentland and Andrew Liu. Modeling and predicition of human behavior. In *IEEE Intelligent Vehicles 95*, September 1995.

[39] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: the art of scientific computing*. Cambridge University Press, Cambridge, U.K., second edition, 1992.

[40] J.M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: An application to human hand tracking. In *European Conference on Computer Vision*, pages B:35–46, 1994.

[41] K. Rohr. Cvgip: Image understanding. *"Towards Model-Based Recognition of Human Movements in Image Sequences*, 1(59):94–115, 1994.

[42] Deb Roy and Alex Pentland. Multimodal adaptive interfaces. In *AAAI Spring Symposium on Intelligent Environments*, 1998. also Vision and Modeling Technical Report #438, MIT Media Lab.

[43] R. Shadmehr, F. A. Mussa-Ivaldi, and E. Bizzi. Postural force fields of the human arm and their role in generating multi-joint movements. *Journal of Neuroscience*, 13(1):45–62, 1993.

[44] Henry Stark and John W. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice Hall, 2 edition, 1994.

[45] Thad Starner and Alex Pentland. Real-time american sign language recognition from video using hidden markov models. In *Proceedings of International Symposium on Computer Vision*, Coral Gables, FL, USA, 1995. IEEE Computer Society Press.

[46] Charles W. Therrien. *Decision, Estimation, and Classification*. John Wiley and Sons, Inc., 1989.

[47] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[48] A. S. Willsky. Detection of abrupt changes in dynamic systems. In M. Basseville and A. Benveniste, editors, *Detection of Abrupt Changes in Signals and Dynamical Systems*, number 77 in Lecture Notes in Control and Information Sciences, pages 27–49. Springer-Verlag, 1986.

[49] Alan S. Willsky. *Recursive Estimation Supplementary Notes*. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, Spring 1994.

[50] Alan S. Willsky, Gregory W. Wornell, and Jeffery H Shapiro. *Stochastic Processes, Detection and Estimation*. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, Fall 1995.

[51] Andrew Witkin, Michael Gleicher, and William Welch. Interactive dynamics. In *ACM SIGGraph, Computer Graphics*, volume 24:2, pages 11–21. ACM SIGgraph, March 1990.

[52] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

[53] Christopher R. Wren. *Perspective Transform Estimation*. Massachusetts Institute of Technology Media Laboratory, Cambridge, MA, USA, December 1998. http://www.media.mit.edu/∼cwren/interpolator/.

[54] Christopher R. Wren and Alex P. Pentland. Dynamic models of human motion. In *Proceedings of FG'98*, Nara, Japan, April 1998. IEEE.

[55] John Wyatt. *Recursive Estimation Lecture Notes*. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, Spring 1997.