Vision-Based Recognition of Actions using Context

A DISSERTATION

Presented to

The Academic Faculty

By

Darnell Janssen Moore

In Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy in Electrical Engineering

Georgia Institute of Technology

Atlanta, Georgia USA

April 2000

Vision-Based Recognition of Actions using Context

A Dissertation by Darnell J. Moore

Approved:

Monson H. Hayes III, Chairman

Irfan A. Essa

Mark A. Clements

Date approved by Chairman

To my parents,

Gwendolyn W. and Darnell J. Moore

to my sister, Daphne J. Moore

to my beloved extended family and friends

Acknowledgments

First and foremost, I give all honor and praise to God for all of His many blessings and for giving me the strength, the perseverance, and the determination to fulfill this dream. To whom much is given, much is required. I pray that He will continue to use me according to His will.

I would like to thank my defense committee, specifically Professors Ron Schafer, Mark A. Clements, and Thad E. Starner, for their willingness to serve and provide me with extremely valuable comments that helped to improve the quality of this thesis.

My thesis advisor, Professor Monson Hayes, deserves special thanks for his guidance and support over the past several years. I appreciate his thoughtful direction and the freedom he gave me to explore by interests and express my creativity. I would also like to thank my technical advisor, Professor Irfan Essa, for his role in my development in the area of computer vision. He has been an outstanding mentor and has truly challenged me to strive for excellence. They both will always be tremendous sources of motivation for me and have helped me become a better engineer.

None of this would, likely, have been possible were it not for Dr. Fred Bacon of 3M's Fiber Optics Laboratory in St. Paul, Minnesota. He introduced me to research and challenged me to pursue the Ph.D. degree. Without his initial guidance and his confidence in my abilities, I would have never dreamed that I was capable of completing a Ph.D.

My special thanks and well wishes go out to the members of the Computational Perception Laboratory (CPL). I would especially like to thank Scott Stillman, Gabriel Brostow, Rawesak Tanawongsuwan, Amos Johnson, Wasinee Rungsarityotin, Arno Schöedl, Roberto Peon, Drew Steedley, Kayt Sukel, Dr. Aaron Bobick, and Tony Haro for their input and friendship over the years. They all helped to make graduate school fulfilling and even

fun, at times.

I would also like to thank all of the members of the Center for Signal and Image Processing (CSIP). A very special thanks goes to Drs. James DeBardelaben and Ara Nefian for their special friendships and invaluable advice over the years. Their camaraderie has made this experience an unforgettable one that I will always cherish. I also built some lasting friendships with several others including, but not limited to, Drs. Chris Lanciani, Ram Rao, Jonathon Su, Halûk Aydınoğlu, Joey and Jon Arrowood, Balu Santhanam, and Jeff Price.

Most importantly, I am eternally grateful for my family for their unconditional love, steadfast support and unselfish sacrifice throughout my life. I owe them the world for their faith in me and for teaching me, by example, how to love and to live. Thank God that I come from a family that believes in prayer. I love them endlessly. I would also like to thank my loved ones who have preceded me in death for the love and encouragement they showed me while they were here. I love them all dearly. Last but not least, I owe countless thanks to all of my friends for sharing their love and lives with me. Thanks for helping me to grow and enjoy life.

Contents

List of Tables

List of Figures

Summary

In this dissertation, we address the problem of recognizing human interactions with objects from video. Methods for recognizing these activities using human motion and information about objects are developed for practical, real-time systems. We introduce a framework, called ObjectSpaces, that sorts, stores, and manages data acquired using low-level vision techniques into intuitive classes. Our framework decomposes the recognition process into layers, i.e., a low-level layer for routine hand and object tracking and a high-level layer for domain-specific representation of activities. Segmenting recognition tasks and information in this way encourages model reuse and provides the flexibility to use a single framework in a variety of domains.

We present several ways of using context to aid in recognition problems. We exploit object context information (class, location, etc.) to help recognize hand-based actions and to enhance hand tracking. To classify unknown objects, we evaluate action context along with low-level image features and associations with other objects, where available. Throughout these approaches, we combine stochastic methods for classification of complex activities. Low-level hand actions associated with objects are recognized using the hidden Markov model. Markov chains are used to characterize a sequence of these low-level interactions so that single-tasked activities can be classified. At the very highest level, we use stochastic context-free grammar to represent the structure in activities that involve multiple objects and people over extended periods of time. We also provide extensions to the Earley-Stolcke parsing algorithm that enable error detection and recovery as well as adapt stochastic grammar to improve recognition. We also present methods of quantifying group and individual behavioral trends in activities with separable roles.

We show results of activity recognition in various domains, including an automobile,

a kitchen, and an office. Our approach is appropriate for locating and classifying both rigid and deformable objects under a variety of conditions including partial or full occlusion. We also provide results where both familiar and previously unseen objects are classified from action alone. From experiments with the card game, Blackjack, we produced high-level narratives of multi-player games and successful identification of player strategies and behavior.

CHAPTER 1

Introduction

"If you want to know the future, build it."

- Mark Weiser, [144]

Computer vision is a mechanism that will enable future generations of computers to be more perceptive of their surroundings and more capable of interacting intelligently with people. Vision-based technologies are posed to revolutionize the ways people relate to computers, and likely, the ways computers relate to people. The range of applications, which include automatic video surveillance and interactive spaces, is virtually unlimited.

In this dissertation, we focus on building computational awareness of human interactions with objects from video. Specifically, we are addressing the need for methods to recognize complex human activities as well as objects in the surroundings. Our research targets the range of activities that

(a) exhibit regular patterns that follow well-understood rules or structure,

(b) can be defined semantically, and

(c) possess measurable, salient attributes that can be characterized.

To capture many of these experiences, recognition methods are needed that can accommodate multiple users performing complex, coordinated tasks with several objects. As vision systems attempt to solve more sophisticated problems, formal approaches for utilizing and managing information will be required. Moreover, a pragmatic system design that recognizes interactions in real-time and over extended periods is also desirable.

To support computational perception of people and their interactions with objects, we have developed methods for extracting low-level features from video that can be used to

assemble higher-level representations of activity. The first step towards activity recognition requires a means of acquiring and analyzing motion-based events over time. Appearance-based features, like color and shape, along with information about the environment, such as the location of known objects, are assessed to track the hands during interactions. Our correspondence algorithm features estimates that improve linearly predicted positions of future hand locations using environmental information, and activity zones that use spatial context to assess hand traffic areas in the scene.

Hand motion is analyzed for recognition of low-level, pre-trained actions, which are characterized using hidden Markov models (HMMs). Higher-level, single-tasked activities, which are represented by Markov chains, are recovered by evaluating the sequence of low-level interactions generated by motion tracking and analysis. We demonstrate that recognition of interactions is easier and more reliable if we know the object's class and other contextual information.

We also prove that recognizing actions is useful for object discrimination. We offer a new approach to object classification that exploits detected actions, low-level appearance features, and object-based context to differentiate the class of unknown objects. A Bayesian classifier provides maximum likelihood classification by adapting how image-, object-, and action-based evidence is weighed as it is accumulated over time. Our approach is appropriate for locating and classifying objects under a variety of conditions including full occlusion.

We have also developed new techniques for adapting stochastic grammar designed to represent the structure of multitasked activities. Multitasked activities are complex interactions that involve multiple sub-tasks (single-tasked activities), objects, and people. High-level events of these activities are detected using heuristic models that take advantage of domain knowledge and low-level image-, object-, and action-based evidence. We offer new event parsing strategies that provide error detection and recovery. We also show novel methods of quantifying group and individual behavior in activities with separable roles. These extensions to conventional stochastic parsing provide real-time recognition of rule-based tasks.

To organize and exploit all of the information that is collected over the course of complex activities, we introduce an object-oriented framework, called *ObjectSpaces*. Ob-

jectSpaces is suitable for recognition tasks involving hand-based interactions with objects. This top-down, bottom-up framework serves as a backbone for all tracking and recognition processes by embracing intuitive policies for organizing, maintaining, and sharing data. Object classes are created for objects, people, and the activity domain. Classes, which are essentially containers for placing object-specific data and functions, provide hierarchical categorization of complex information. By exploiting these features, we show that vision applications can be decomposed into layers, which enable reuse of classes, provide design flexibility, and make recognition problems easier to solve. ObjectSpaces offer several advantages over an ad-hoc system design, including:

- a self-maintaining database of class models,
- a scaleable, modular, and efficient architecture appropriate for real-time implementation, and
- object-based event handling and focus-of-attention.

To demonstrate the efficacy of our approach, we have constructed a real-time vision system using *Visual C++*. We examine human activities in several domains, including an office, a kitchen, and an automobile, with average recognition rates exceeding 87%. We also demonstrate detection and recognition of both familiar and previously unseen objects using image-, object-, and action-based evidence. Similar experiments were also conducted using only action-based evidence to classify both rigid and deformable objects. Experiments for recognizing multitasked activities are also presented. Using stochastic context-free grammar to parse events from the card game Blackjack, we produced high-level narratives of multiplayer games and successfully identified player strategies and behavior.

Recognition of human interactions is a complex image and signal understanding problem that requires low-level image processing and pattern analysis as well as syntactic pattern classification to exploit high-level structure in activities. In this dissertation, we show that our approach offers a compelling solution to the problem of computational awareness of multitasked activities. Our work represents one of the first significant contributions to vision-based recognition of multitasked activities, so there is very little existing work with which to compare our results. In summary, the major contributions of our research are:

- methods for extracting and managing low-level information from video,
- action recognition from exploiting object and domain context,
- object recognition using action context, and
- recognition of multitasked activities using adaptive stochastic context free grammar with error detection and recovery.

1.1 Organization of Dissertation

The remainder of this dissertation is dedicated to describing the challenges involved in building perceptive, computer vision-based systems to recognize human interactivity.

Chapter 2 provides motivation for awareness of human activities by suggesting possible applications. We discuss human interactions and the fundamental components of vision-based systems. Key challenges related to activity recognition are mentioned. We conclude this chapter with a brief review of motion analysis, action recognition, and object classification methods.

Chapter 3 presents ObjectSpaces, our framework for facilitating recognition tasks. ObjectSpaces, which handles context management and organizes extracted information, serves as a backbone for various processes. We discuss aspects of the architecture, including object-oriented design principles as well as basic constructs for object class development and reuse. Layers for decomposing the recognition process and a class model database are also highlighted.

Chapter 4 describes how hands are identified and tracked using low-level image features and environmental information. We continue with a brief overview of the hidden Markov model and its applicability to human motion analysis. We also discuss issues related to implementation, such as initialization, topology, and computation.

Chapter 5 introduces evidence based on hand actions, image appearance, and objects association to recognize single-tasked activities or to classify objects. We present Bayesian decision theory and belief networks as the basis for recognition. We also provide experimental results from a variety of domains and testing conditions.

Chapter 6 defines "multitasked activity" and highlights its associated challenges. After a brief review of stochastic context-free grammar, we present an approach to parsing continuous, concurrent streams of events from multiple people and objects. Experiments deal with rule-based activities like *Blackjack* (card game).

Chapter 7 provides a summary of contributions in this dissertation. Future work and direction are also mentioned.

Appendix A describes the three key problems of the hidden Markov model in detail.

Appendix B is a reference for rules of the card game Blackjack/21.

Appendix C provides information on the Vision Action Recognition System (VARS), a real-time vision analysis system implemented in Visual C++.

Appendix D provides on review of grammar and syntactic pattern recognition.

CHAPTER 2

Background & Motivation

We hope to establish computer awareness of the environment, which is the first major step toward developing truly intelligent systems. Systems that are aware of their surroundings stand a much better chance of recognizing, learning and anticipating human activity. Signal processing and signal understanding represent key components of a solution to these issues. Throughout this dissertation, we will present approaches for signal representations that extend traditional signal- and image-based processing by leveraging knowledge-based resources like contextual information. In addition to quantitative models to represent motion analysis, methods are needed to capture and exploit dynamic information about the environment.

In this chapter, we begin by highlighting vision and several practical applications. We continue with a review of many of the fundamental issues with image-based computer vision systems, including feature extraction and motion analysis. We will also highlight several of the challenges that we address in this research. Related and existing work on action and object recognition will be surveyed.

2.1 An Overview of Computer Vision

Within the past few years, there has been a tremendous amount of interest in teaching computers to understand human behavior from images and video. Surprisingly, the initial impetus for today's work in human motion can likely be traced back over 112 years ago when Eadweard Muybridge published "Animal Locomotion" in 1887 [101]. Muybridge, regarded as the Father of Motion Pictures, was the first to capture photographic recordings of humans and animals in motion for examination purposes. His work is one of the most

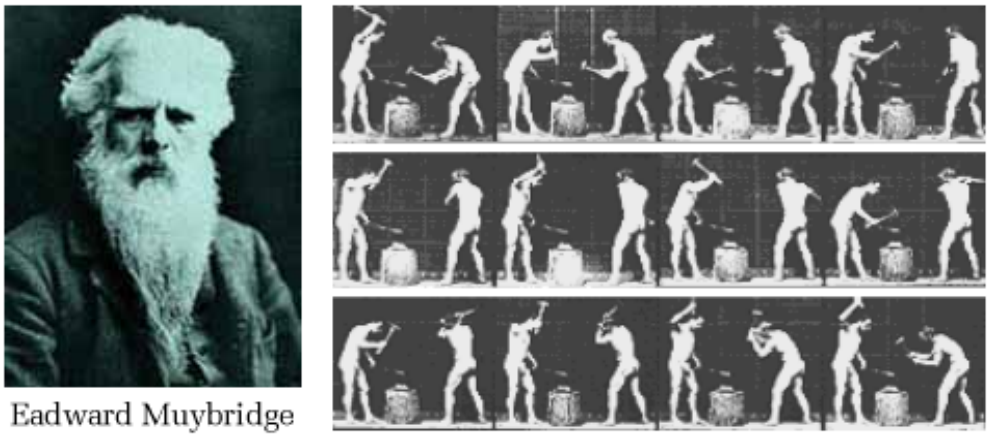


Figure 2.1: (left): Eadward Muybridge (1830-1904) (right): Photographs of *sledge hammering* taken by a process Muybridge invented that used a battery of still cameras [40].

comprehensive analysis of movement ever undertaken and is still widely used today as a source of illustration and for reference (see Figure 2.1).

However, it was Roberts' monumental work that began computer vision as it is known today. In the early 1960s, he was able to characterize the three-dimensional structure and arrangement of simple trihedral-vertex polyhedra from digital images [122]. Feeding on this modest but celebrated achievement, researchers continued to enjoy several episodes of moderate progress throughout the 1970s and 1980s, but not enough to satiate all of the hype promoting vision. Compromised by their incredible expense and painfully slow operation, most of the early vision systems lacked the robustness to do little more than recognize simple geometric shapes in low-resolution monochrome images. The spotlight quickly faded on those initial systems after the sobering realizations finally started to sink in: *vision is a very difficult problem* that can not be solved using simple, single-faceted strategies.

Although vision problems have not gotten any easier, researchers are much better equipped today to address its issues. The catalyst for the latest renaissance in image understanding has likely come from more powerful computers, inexpensive cameras, and a body of maturing knowledge. As proof of this trend, consider these few facts: standard desktop compute power has increased by 1,333% in the last ten years [97]; the average price of video capture hardware for the desktop has plummeted over 1,100% since 1989 [60, 143];

as of September 1998, over 5,724 journal articles have been published in computer vision since 1983 [68]. Moreover, the critical mass of complementary technologies needed to make perceptive systems not only a reality, but pervasive, are arriving and should extend the range of potential applications. These technologies include cheap but massive data storage, standardization of digital media and formats, multimedia architectures and operating systems, high-speed networks, distributed databases, and the Internet.

2.2 Applications

Arguably, there is no shortage of applications that can take advantage of human action perception. For starters, human-computer interfaces using this technology will permit people to interact with computers outside of the traditional desktop setting, paving the way for computer-embedded environments. Many futurists envision environments driven by human behavior and situational context, making the underlying technology appear to work effortlessly and transparently. Several potential application areas for vision include:

- automatic video surveillance [34, 52, 111];
- unsupervised annotation of video;
- content-based image storage and retrieval;
- interactive tutors that teach activities, such as cooking, by observing and responding to user behavior;
- action-model based coding for scene indexing and annotation, for driving animation, or for reconstructing scenes in a remote location;
- self-checkout stations in department stores and supermarkets to monitor customer merchandise scanning [105, 106];
- human activity analysis in retail, banking, and industrial environments [105];
- video game user interface driven by user actions or gestures [111];

- systems for monitoring and assisting people with disabilities, including the hearing impaired and the elderly [115, 130];
- autonomous mobile robots and agents with environmental perception [58];
- and smart rooms for smart living areas [47, 49, 55, 81, 99, 100, 124].

2.3 Recognition of Human Interactions

In this dissertation, we primarily consider physical interactions between people and other objects in the surroundings. We leverage what is already known about an object so that we only have to look for associated interactions instead of attempting to classify the full spectrum of human motion. In the following section, we review action recognition.

2.3.1 Action Recognition

Action recognition is closely related to gesture recognition, but typically involves a broader, less structured class of animated *body* motions. While both are inexplicably connected to the behavioral context in which they are used, most gestures that can be classified are reserved to augment communication between people. We concentrate on actions performed on some object by a person. These actions are not only intentional, but serve some objective or motive with respect to some object. Recognition of human activities poses several challenges that need to be addressed [45]:

- Repeated performances of the same activity by the same human vary even when all other factors are held constant;
- Similar activities are performed by different individuals in slightly different ways;
- Parsing an individual activity from a continuous performance of action is often difficult;
- Similar activities can have a variety of temporal durations;
- Different activities can have similar temporal durations or motion profiles;

- Occlusion and self-occlusion of body parts can occur during activity performance;
- The projection of moving body parts, and hence the interpretation of action, is dependent on the viewpoint;
- The distance between the camera and the human affect image-based measurements.

We deal specifically with common activities that take place in well-understood domains, i.e., office work, card games, driving, etc. The bulk of our research involves the characterization of hand-based motion although some full-body movement is studied.

There is a general tendency to use the words "action" and "activity" interchangeably, but this may cause some confusion in future discussions. We refer to *actions* as the basic building blocks of activities. An *activity* represents a family of actions that are coordinated to accomplish related tasks. For example, the "atomic" actions cut, stir, and slice all fit under the *eating* activity. People intentionally participate in a range of different activities to accomplish specific tasks. We divide the domain of activities into two camps: *single*-tasked activities, where there is an ordered sequence of actions generated by a single user, and *multitasked* activities, where multiple people and objects are involved.

2.3.2 Representations for Interactions

One of the predominant goals of this research is to exploit the relationship between an object and a finite set of possible interactions with that object. For example, the set of actions associated with a chair might include sitting in it, pushing it, or even standing in it. By constructing motion models for these three actions, we can appropriately describe interactions with the chair. Naturally, there is also a complementary set of actions that are not modeled, such as kicking or throwing the chair. Even without a motion model for these actions, we may still be able to describe the interaction by measuring the consequence of contact with the chair. Assessing the effects of contact will allow us to conclude that the chair was *moved* although we can not determine *how* it was moved. In the worst case scenario where there is neither a model nor adequate consequential information, we can only establish that some contact between a person and object has occurred. However, over time

we may be able to establish important contact patterns simply by observing the frequency of interaction.

In any case, we combine context with the mode of contact to form *interactive events* that represent interactions. These events use all available evidence to provide the most descriptive account of the interaction. Chapter 3 will introduce an architecture for a vision systems that handles this coupling of object and motion for representing interactions.

2.4 Computer Vision Systems

Computer vision problems are inherently difficult to solve because the units of observation are not the units of analysis [63]. The unit of observation is essentially the pixel, which by itself, offers virtually no information about the people, objects, and activities taking place in the image. Building transformations that can map these observation units into a viable space for analysis represents one of the primary objectives of computer vision research.

The human visual and cognitive system serves as a paragon for the vision community, even though there remains a great deal about its operation and complexity that are still vastly unknown. In attempting to mimic the human visual process, computer vision has been primarily rooted in image processing, computational intelligence, and pattern classification approaches [104]. In most cases, vision systems employ techniques for identifying and extracting information from monochrome or color image sequences. Data that are pulled from these sequences must be interrogated by a classification process that labels observations accordingly. We will discuss this process in more detail in the following sections.

2.4.1 Feature Extraction & Representation

To build representations that accurately capture the activities taking place, image features of people and objects must be carefully identified and extracted. A sequence of extracted features should be able to construct a meaningful representation of the action taking place in the original image sequence. Figure 2.2 shows many of the most commonly used methods for feature representation.

One of the most significant constraints placed on real-time vision systems is the com-

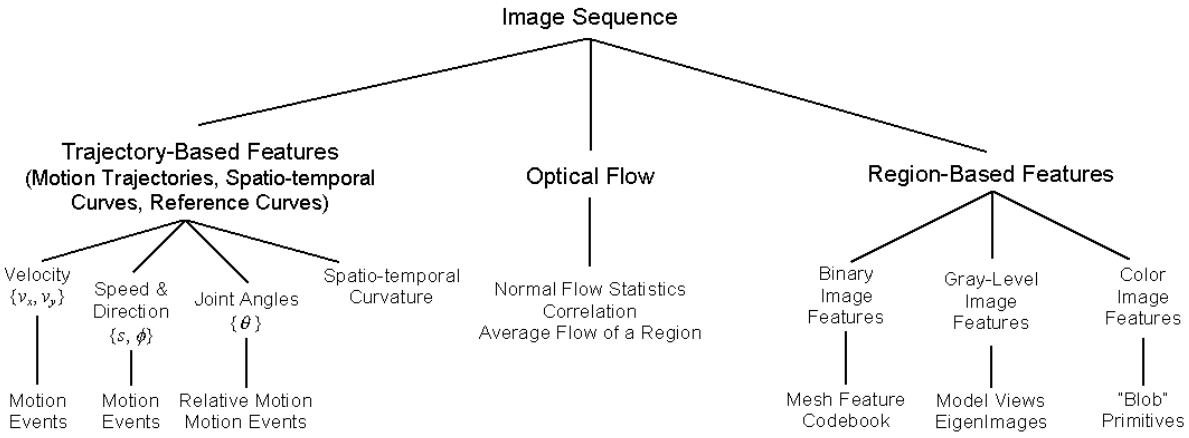


Figure 2.2: Methods of feature representation from a sequence of images.

putational overhead required to analyze an image and construct observations of important features. Additionally, many systems attempting to assemble high-level descriptions of human behavior must manage several other greedy processes simultaneously; in which case, there is ample motivation to utilize computational resources prudently. Such limitations encourage our use of region- and trajectory-based representations because they can often be processed in considerably shorter time than optical flow fields. Typically, optical flow patterns must be constrained by known 2D or 3D models before meaningful analysis can take place. Region-based features are also much less sensitive to sensor noise and fluctuation in lighting conditions than optic flow field vectors. We use representations that involve appearance-based features because they tend to be more stable in the presence of noisy data. Moreover, region-based tracking serves as an appropriate means for extracting information about human motion without supervision (after initialization).

2.4.2 Analysis Methods for Motion Characterization

Generally, there are two schools of thought that impact recognition of actions: *motion recognition from structure* and *motion recognition in the absence of structure*. The latter approach uses low-level features, such as the properties of colored blobs that represent skin or clothes, to target and track objects. With sufficient contextual information, this strategy can be efficient and reliable. However, without the benefit of structure to help

Figure 2.3: Simplified visualizations of Stoll's 3D model for a *bowling* motion.

resolve problems like self-occlusion, the use of motion without structure can prove to be a significant limitation.

Conversely, motion recognition from structure builds on top of low-level features to construct a 2D or 3D model of an object, such as a physics-based description of the human body. The sequence of recovered parameters, perhaps joint-angles of arms, legs, torso, etc. from the 3D model, are passed to a classifier for recognition. An example of motion from structure is provided by Stoll, who uses HMMs to match 3D models for recognition as seen in Figure 2.3. This high-level approach can offer more meaningful testimony of the action, but is computationally more expensive and highly dependent on the recovery of low-level primitives. Feature extraction directly influences the methods used for analyzing movement, whether they are probabilistic state-based representations, neural or fuzzy networks, or the like. To minimize computation, the smallest and richest feature representation is desired. Redundant features can help when uncertainty is high, but adding those that are prone to large noise levels can degrade performance. Finding the most significant measurements to extract are highly correlated with the nature of the motion and the view from which it is observed.

View-Based Representations

There is little consensus about which aspects of body movement and orientation are most important for motion analysis, so consequently, most system designers take an *ad hoc* approach that is motivated by the intended application. Stereo or multi-camera systems can provide unambiguous, three-dimensional object positions, but must contend with image correspondence issues and camera calibration. To develop algorithms that are view independent, a "world" model is required [25]. However, most vision systems rely on monocular

Recognition Methodology	Advantages	Disadvantages	Authors
Causal analysis of sequences using filters	Robust labeling even when undersampled	High-level approach requires supervision	Brand, Essa [32]
Phase Space Constraints (Motion Trajectories & Joint Angles from 3D Model)	Invariance to speed changes in body extension	Supervised learning	Campbell, Bobick [38]
Condensation	Auto. extraction of motion trajectory	Requires trackable icons & supervised initialization	Black et al. [20], Isard [71]
Recognition of Oscillatory Motion using Position Vector & Dynamic System Models	Prediction module allows real-time recognition w/ $> 85\%$ accuracy	Limited gesture vocabulary	Cohen et al. [41]
Multi-class, Multi-variate Discriminant Analysis to select Most Discriminating Features which are classified using recursive partition trees	Feature set is adaptive; Can identify 28 static hand signs from 805 total images with 93.1% accuracy	Segmentation against complex backgrounds takes 58.3 seconds per image	Cui, Weng [42]
Neural Networks	Decision available after every frame (real-time)	Can not process a sequence of frames	Schlenzig et al. [126]
Time-Delay Neural Networks	Learns space-time patterns; shift invar.	Need lots of training data	Yang [149]
Motion-History Images & Motion Energy Images	Compact & efficient templates for recognition	Sensitive to viewing perspective; does not handle spatial variation	Bobick [22]
Dynamic Time Warping (DTW)/Continuous Dynamic Programming	Time-normalization of space trajectories easy allow comparison to model	Behaves poorly with non-spatial features must be calculated at every frame	Gavrila Nagaya Nishimura [57, 102, 109]
Active Gesture Recognition using Partially Observable Markov Decision Processes	System can adaptively learn from experiences	Redundant detection	Darrell, Pentland [43]
Recognition from Orientation Histograms	Fast; translation invariant	Rotation invariant; Poor discrimination from similar gestures	Freeman, Roth [53]

Table 2.1: Several classification schemes used by action and gesture recognition systems. Continued in Table 2.2.

Recognition Methodology	Advantages	Disadvantages	Authors
Hidden Markov Models (continuous & discrete)	Handles space-time variation using DTW; highly scaleable	No decision available prior to observation sequence completion	Starner, Bobick [130, 21]
Model-Based Motion Recognition using Color & Rule-Based Classifier	Average recognition w/ 95% accuracy	Subject must wear colored glove/markers	Hienz et al. [64]
PNF Calculus for Handling Causal Propagation of the states of Time Intervals	Labeling (past, now) & future of events w/ loose structures	Deals poorly with time variation	Pinhanez, Bobick [115]
Recursive Learning using Rule-Based Induction	Fast recognition of 20 hand poses w/ 94% ; accuracy incremental learning possible	Difficult to generalize to other apps due to specific rules	Schlenzig et al. [126]
Range images	Uses velocity vectors; Easier than optic flw	only 5 gestures used	Umeda [141]

Table 2.2: *Continued from Table 2.1*: Several classification schemes used by action and gesture recognition systems.

image sequences, whose interpretation is directly tied to camera view (see [112] for a review of visual interpretation of hand gestures). Collapsing 3D motion into a 2D image representation presents special challenges for motion analysis such as:

- object occlusions caused by other objects or the object itself,
- scale or rotation variance,
- motion along lines of projection, and
- distortion caused by perspective foreshortening.

2.4.3 Classifiers for Motion Recognition

The goal of the classifier is to find the best match between image feature representations and a recognition methodology. Proper action representation must demonstrate time-shift invariance, i.e., time normalization, and order preservation [23]. Most importantly, this representation must be able to quantify the progression in space and time so that its motion signature can tolerate natural variation but still be distinguished from others.

First, however, we need methods to model human actions reliably. For a review of motion analysis, see [4, 40, 59]. There have been a number of classifiers that have become popular for distinguishing spatio-temporal data generated by human motion. Several of them are listed in Table 2.1 and Table 2.2. The temporal ordering of human motion motivates the use of state representations, such as the hidden Markov model. The motion models we consider are generally intended to characterize repeatable, individual, hand-based actions.

State-based representations are usually a finite state machine that attempts to segment and group motion into natural, ordered stages. One of the first state-based methods used Dynamic Time Warping (DTW) to align gestures and actions in time by warping. This process, which is similar to Continuous Dynamic Programming (CDP), works best when considering spatially significant data, such as motion recognition from structure parameters or frames containing subjects against a solid background. In fact, Gavrila and Davis apply dynamic time warping to joint angles taken from 3D models to force alignment of sequences

[57]. Nagaya et al. also use dynamic programming to judge the shapes of approximated motion trajectories by matching them against trajectory templates [102]. Bobick and Wilson [23] use a dynamic programming formulation to compute the minimum cost path between two nodes in a graph produced by motion trajectories. Nishimura and Oka [109] extract information from undersampled binary images in a real-time gesture recognition system that achieved 80% accuracy for eight different gestures using dynamic programming. However, their approach deals poorly with changes in the speed of gestures or the detection of fine motion. Most of these approaches use the dynamic programming algorithm to causally align a stream of spatially generated data so that its time normalized space trajectory can be equivocally compared to a prototype for recognition. While a powerful technique for comparing the physical displacement of hand motions, this same virtue is also a handicap, as dynamic programming behaves poorly when non-spatial features are supplied. In addition, dynamic time warping must be applied at every time instant since accurate segmentation at motion boundaries can be difficult without supervision. Black et al. [19, 20] recently introduced a method based on the Condensation algorithm [71, 72] to recognize gestures and human motion. Human motions are modeled as temporal trajectories of estimated parameters over time. Condensation is used to incrementally match human trajectory models to multi-variate input data.

A common thread in much of the recent work in action recognition has been the use of the hidden Markov model (HMM) as a means of modeling complex actions [22, 31, 95, 130, 148]. HMMs exploit the temporal ordering of natural gestures and actions using a probabilistic state-based representation. With proper training data, HMMs can efficiently characterize motion profiles in spite of the broad variation in the space and time domains in which actions are performed. In particular, the HMM has demonstrated a keen ability to provide robust recognition over extended motion sequences produced by multiple users [21, 31, 110]. With an impressive legacy in speech recognition [67], HMMs are well suited for recognition of human actions. However, probabilistic finite state machines, like HMMs, can not handle complex actions that involve concurrent or parallel events, multiple sequencing possibilities, and mutually exclusive intervals [5]. In Chapter 4, we take a closer look at the HMM and describe how we use it to represent low-level hand actions.

2.4.4 Object Recognition

As mentioned earlier, some of the first work done in computer vision was conducted in object detection and classification. The area continues to be a primary research focus. Ullman chronicles most of the seminal contributions in object recognition since Roberts in [140].

While several approaches for object classification have surfaced over the years, most of these proposals require models or templates as the basis for constraining and mapping extracted features. Generally, models for various classes of objects are defined using some three-dimensional parametric representation. Projections of these 3-D structures are rendered, often for a range of viewing conditions that take into account factors such scale and illumination, so that their image features can be realized. This step becomes the *prediction*. To describe 3-D objects from images, these features must be identified using some unsupervised segmentation process that also recovers 3-D surfaces using a range of approaches, including shading, texture, motion, stereo, or line drawings. This *description* stage produces a candidate class and attempts to match it to a known class.

There has only been a limited investigation into the use of action as the *primary* means of recognizing objects. In an example, Duric *et al.* determine an object's function from its motion using deductive reasoning [46]. In Chapter 5, we introduce the notion of recognizing an object from how a person interacts with it. We consider our work as one of the first contributions of object classification from action. However, our interest is not to explore the domain of problems in object recognition but to study interactions that may require the classification of unknown objects. Our work attempts to extend standard, low-level methods for object recognition.

CHAPTER 3

ObjectSpaces: A Framework for Vision Recognition Tasks

In this chapter, we discuss the problem of information management and process design. We argue that to provide high-level awareness of complex human behavior, methods are needed to handle the enormous amount of dynamic information that is extracted from video. All of this information requires facilities for organization, storage, and retrieval in addition to policies for modeling, segregating, and integrating. Moreover, as the activities that computers attempt to understand get more complicated and last longer, the complexity of modeling these activities also grows.

We begin by discussing context and its role in solving recognition problems. To address many of the issues mentioned above, we introduce a new object-oriented framework called *ObjectSpaces*, which facilitates data management for recognition tasks. ObjectSpaces sorts information by creating class models for objects, people, and activity domains. Object and hand tracking generates low-level evidence, which is collected and organized according to class, then passed up to another layer that analyzes interactions to recognize high-level activities and unknown objects.

The design of complex recognition systems benefits from careful planning that permits reuse and expansion, which are generally after-thoughts on *ad hoc* systems. Instead of using ad hoc design approaches to build task-specific systems, we present a methodology that accommodates a wide range of high-level vision applications by decomposing the activity domain and the entire recognition process into layers. Our design philosophy is pragmatic, scalable, modular, and efficient enough to deliver real-time performance and offer flexible implementation. In addition to the benefits generally available in an object-oriented

paradigm, features of our architecture include:

- base class templates for sorting context information
- layered framework for decomposing recognition problems
- activity zones for evaluating the spatial context of activity and to improve hand tracking,
- image-based methods for tracking objects
- object-based event handling and focus-of-attention,
- appearance-based templates for action and object recognition, and
- a database of class models for use in object classification.

As we reveal specific approaches to activity understanding over the next several chapters, we will build on top of the foundation established by ObjectSpaces. By adopting a protocol for representing and organizing information up front, we can leverage both information and processes to develop high-level awareness.

3.1 The Role of Context

So far in our discussion, there has been a frequent reference to context. Indeed, many ideas presented in this dissertation are motivated by using context to enhance the representation of information extracted from video. So exactly what is context?

Definition 3.1 *We define* **context** *as information that influences the interpretation of evidence. In essence, it captures the state of people, objects, and interactions by quantifying the circumstances in which they occur.*

Evidence refers to some set of values or data that is either provided beforehand or extracted from video. Classification involves the evaluation of evidence to make a decision. Any classifier, however primitive, must also have a model of context to properly evaluate candidate solutions. For example, if we have evidence (in the form of hand displacement) that a person

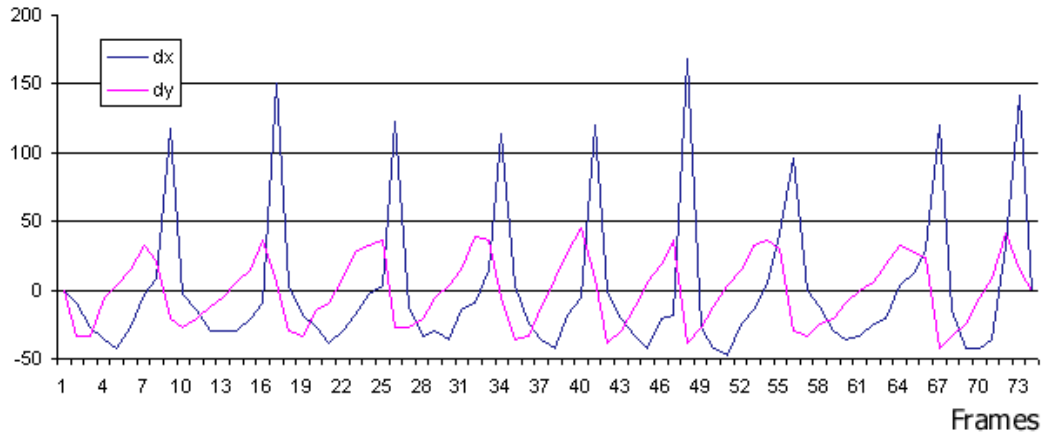


Figure 3.1: Displacement (dx, dy) during circular hand motion.

is interacting with an object in a clock-wise fashion as shown in Figure 3.1, there may be *several* plausible explanations for this behavior. We can reduce the field of possibilities if we also know that the object is a sauce pan, in which case, we can reasonably assume that a person is *stirring* the contents of the pan. To extend reasoning a step further, perhaps we can anticipate that other cooking-related activities will take place. Once we have established the relationship between motion patterns like this and corresponding object types, we can also use inference for classification purposes. Undoubtedly, to make sophisticated decisions like this, a rich body of diverse information about the people and the articles in the scene is needed. Utilizing information in such powerful ways is not possible without the intuitive management of context information and processes to handle data. Our research focuses on interactions with objects, which in turn, motivates our interest in developing context models for specific types of objects. In this way, we can encapsulate specific descriptions of an object's physical properties along with hand actions associated with it. Higher-level models of context, like task-specific grammar, are also needed to integrate context focused at the object level so that interactions taking place throughout the scene can be characterized. Domain-specific context is used for high-level classification of complex interactions, possibly between objects. Domain context also facilitates spatial context, i.e., exploiting the space where objects are and where activities take place, which is important for both activity and object recognition [11].

We not only stratify context from individual objects to the activity domain, but also with respect to the temporal ordering of activities. The temporal context of human interactions is extremely dynamic in terms of the environmental state changes occurring over time. Conventional data management approaches often assume that evidence collected at any moment is valid indefinitely. High-level classification will require mechanisms that can characterize complex temporal signatures of activities, even while our measures of "belief" change [21].

Strat and Fischler have shown that context can be instrumental in understanding images in complex domains [135]. We have also demonstrated its utility in activity recognition problems [95, 97]. Our use of object and domain context plays a major role in our ability to build awareness through observing and classifying human-object interactions. At this point, it should be clear that various incarnations of context are needed for high-level recognition tasks taking place over extended periods of time. Equally as important is a structure for organizing, storing, and maintaining context.

3.2 ObjectSpaces: An Architecture for Context Management

In the absence of mechanisms to emulate the breath and depth of human intellect, autonomous vision systems must be presented tightly defined problems. Even with constrained problems, the domain of eligible solutions can still approach infinity. It is common practice for developers to take an ad hoc approach to system design, which is typically engineered for specific applications [78]. However, as more advanced vision systems attempt to solve harder problems, the limitations of application-specific designs get more pronounced. For example, most vision systems designed specifically for object recognition can not be adapted for action classification without major configuration changes. Even changes in the activity domain, i.e., recognizing office interactions versus those in a kitchen, can result in having to retrofit significant parts of the system. Several aspects of design that should be avoided include:

- representations with no formal means of characterizing dynamic context,
- ad hoc organization of processes for data acquisition, analysis, and storage,

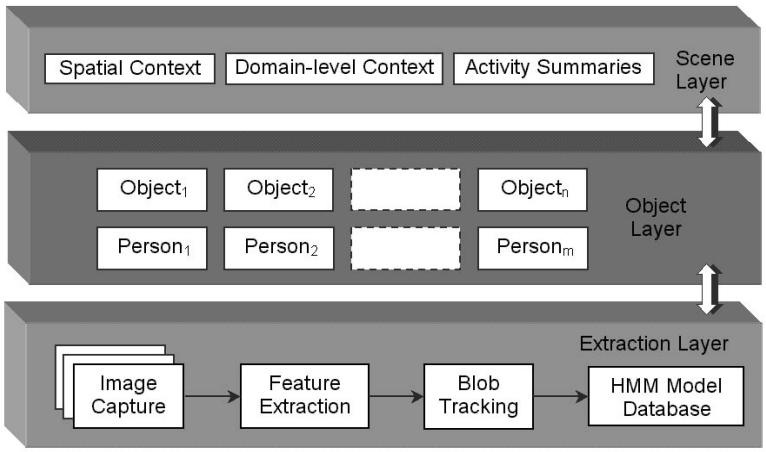


Figure 3.2: ObjectSpaces is a top-down, bottom-up framework that facilitates information sharing and knowledge discovery by passing information between layers using object-oriented constructs.

- no established protocol for sharing information between processes,
- poor code reuse and limited modularity, and
- long software development cycles required for each new activity domain.

To overcome these shortcomings, our framework segments processes into layers and data into objects so that context can be managed at graduated levels. Figure 3.2 illustrates the role of each layer. Object-oriented methods, which combine data with functions that operate on the data, provide the backbone needed for information management and analysis. To support several processes used for detecting, analyzing, and describing high-level interactions, we find and track all objects and people as they move throughout the scene. Tracking identifies the proper image features to extract with help from pre-defined class models and prior knowledge about the scene that reside in higher process layers. After low-level information is extracted, it is retained by the appropriate class model, which is designated for each person and article in the scene. These models use object-oriented constructs that are made available to an even higher layer that examines the relationships between people and objects throughout the sequence using domain-specific information, such as grammar.

To organize and leverage all of the class models and data operations, we develop a top-down, bottom-up infrastructure that permits vital sharing of complementary information between processes, people, articles, using layers. A layered framework allows us to divide complex processes into more manageable stages. For example, layers that handle low-level feature extraction perform the same tasks regardless of the activity domain, whereas layers where high-level classification takes place are often tailored to fit a particular domain. High-level tasks, which are more apt to use heuristic information, obey policies that are consistent across all domains. This is in contrast to ad hoc methods that do not follow any formal procedure for the use of heuristics. Process segmentation also allows us to substitute big, greedy algorithms with thinner, faster approaches (when available) to satisfy real-time constraints without having to modify the overall design process.

In addition, a dedicated repository is provided to sort and store all of the newly discovered information that the vision system generates. Complex, multitasked activities can involve a variety of objects and people over long periods of time, so our architecture is designed to be scaleable as well as discerning of data that is warehoused, i.e., only keeping information that can be used later. ObjectSpaces has logical containers that make information retrieval reliable and fast. This careful organization permits easier integration and evaluation of information for recognition tasks as opposed to conducting global searches through all the data every time a piece of information is needed.

3.2.1 Other Frameworks for Recognition

Although context is broadly used to enhance analysis tasks [1, 123], no universal framework for it has emerged. However, from our evaluation of other vision systems, we see evidence that high-level reasoning can benefit from exploiting graduated levels of context. To site a few example of this, Oliver et al. have proposed a system for assessing interactions between people using statistical Bayesian approaches, HMMs, and coupled HMMs [110]. Their system combines top-down with bottom-up information in a closed-feedback loop to model behaviors and interactions. Using a similar hybrid approach for analysis, Bregler evaluates motion at various levels of abstraction by using a four-level decomposition framework that learns and recognizes human dynamics in video sequences [33]. Also recognizing the virtues

Scene	*Kitchen*	*Office*	*Game room*	*Store*
Activities	cooking, clean dishes, eat	computer work, meeting, study	play cards, play pool	shopping, pick up goods, self-check out
Actions	stir, slices, chop, wash, eat, drink, pour, shake	flip forward, flip backward, open, close, pick-up phone	deal card, flip card, shuffle, pick-up card	place obj in cart, place obj on shelf, scan object
Objects	sauce pan, cup bowl, plate, stove, refrig, toaster, sink	phone, chair, keyboard, book, monitor, CPU, desk, mouse	card, deck, stack, pot, card cradle	cart register, scanner

Table 3.1: Chart demonstrates hierarchical organization of information.

of hierarchical information strategies, Mann and Jepson integrate information about object properties and abilities (primarily force/dynamic) in order to develop representations of activity [89]. They attempt a bottom-up approach that infers physical descriptions of the actions depicted in image sequences. In a vastly different effort, Pinhanez and Bobick have provided a PNF-network, based on interval algebra, to describe the temporal structure of actions, subactions, and events [115]. In this approach, actions or events are constrained as *past, now*, or *future* occurrences. PNF propagation is a type of logical description of action that enhances the discriminatory power of sensors and leads to the detection and removal of inconsistent situations. In a survey of more general attempts at analyzing people, Jain provides evidence of wide-spread experimentation with a variety of frameworks that couple complementary processes and algorithms to solve image understanding problems [77].

While these approaches are motivated by different applications and driven by various detection and recognition algorithms, all of them seek to leverage complementary information provided by (a) combining different classifiers and (b) creating hierarchical layers of context. Architectures that leverage hierarchical inheritance have demonstrated their utility for managing and cataloging information [51]. Hierarchical distribution of information is also compelling because it provides a structure for context that is more easily referenced, managed, and updated. Hierarchical representations provide intuitive organization of various granularities of information, such as those listed in Table 3.1.

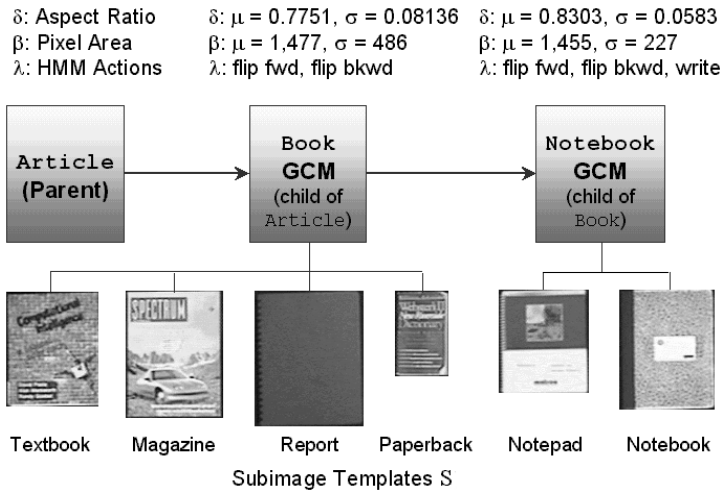


Figure 3.3: Generalized Class Models (GCMs) for the `book` and `notebook` class with actual templates from examples. The GCM offers hierarchical organization of object types.

3.2.2 Using Object-Oriented Principles

Object-orientation is a powerful paradigm for designing, programming, and implementing complex software engineering challenges such as vision systems. Object-oriented (OO) design attempts to model the world through powerful abstractions called *objects*. An object is simply a container for encapsulating context, which takes the form of properties and other representations of state or behavior. Unlike conventional software techniques, data and functions that operate on the data are not separated, but are combined in the same structure. The context model for a particular type of object is called a *class*. It supports type-specific data and corresponding functions. An object is a particular instance of an object class. Typically, an object represents a real-world entity, such as a book or a person. For example, an image template of a book represents data while a procedure that determines an image's pixel count represents a data function.

We use classes and objects as building blocks to assemble representations for articles, people, and the scene itself. Three parent classes, `Article`, `Person`, and `Scene`, act as basic templates for describing all objects representing scene articles, people, and the scene, respectively. New classes inherit the same properties and methods of a parent class, but can be extended by adding additional attributes and methods. For example, we can derive

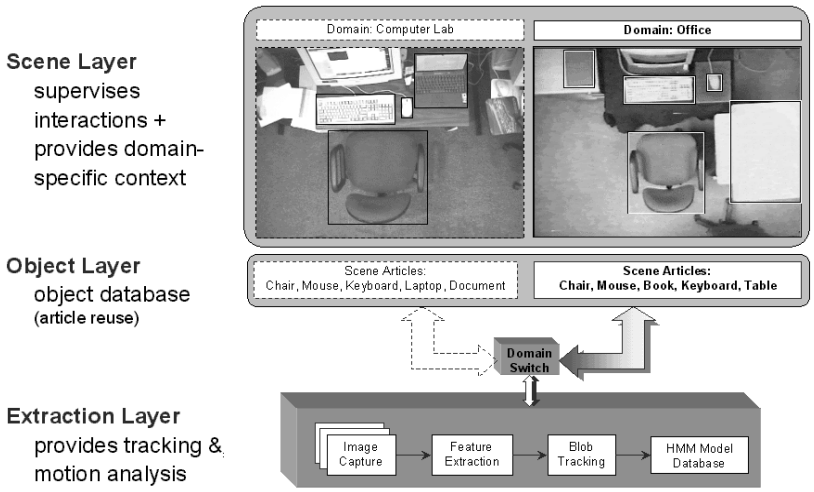


Figure 3.4: ObjectSpace's layered, object-oriented framework allows multiple activity domains to be processed by a single system. Class model reuse is also encouraged.

classes such as `notepad`, `phone`, and `mouse` from `Article` to describe objects in an office. Moreover, `notepad` attaches a model for the action "write," whereas `mouse` may not have any associated action models. Inheritance is used to form a natural hierarchy that helps to organize or disambiguate classes in our class database. Object and action recognition benefit from having properties and functions sorted according to class, as we will see later. Figure 3.3 illustrates the hierarchy formed by derived classes and its usefulness in recognition tasks.

Object-oriented development lends itself to efficient software engineering, reuse, maintenance, and abstraction. Reuse is a powerful mechanism that allows OO systems to be modular and convenient, often shortening the development cycle. Once a class is developed, it can be reused in different domains without having to retrain action models or re-specify the same properties or behaviors. In practice, however, it is common to "tune" the class so that it better models the properties or behaviors for a particular domain, i.e., retraining or adding a few actions or simply adjusting a few parameters. In essence, class models should improve the more they are used. By using multiple, integrated layers, our approach can be easily extended for use in multiple domains without retrofitting the entire framework for a specific application. As illustrated in Figure 3.4, several objects include a chair, mouse, and keyboard appear in both scene. However, after developing a model for each of these

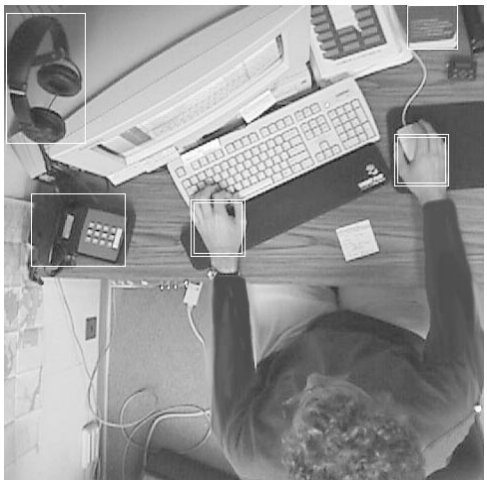


Figure 3.5: Highlighted articles: headphones, mouse, telephone, etc. from ceiling-mounted camera. The hands are also highlighted.

objects, it can be reused in new domains.

ObjectSpaces utilizes an object-oriented design because it provides a hierarchical structure that is intuitive, scalable, and reusable. Moreover, an OO framework facilitates the design and implementation of real systems. Unlike many other complex processes that are planned and conceived on paper only to undergo lengthy and often extensive modification before they can actually be implemented, ObjectSpaces was developed from inception to offer seamless integration between design and implementation. Appendix C introduces the *Vision Action Recognition System* (VARS), a system developed entirely in *C++*, a popular object-oriented programming language. VARS implements our ObjectSpaces framework to handle all of the recognition tasks presented in this thesis. Because object-oriented development represents a significant departure from conventional system design, it is not uncommon for the true essence of OO to remain a mystery for the casual reader. For a more thorough introduction into object-oriented, principles, see [128].

3.3 Building Context Models for Representation

Before we can analyze video, we have to construct models for the articles, people, and the particular domain where interactions will take place. The procedure starts with an initial

snapshot of the scene as shown in Figure 3.5. We manually label known articles in the image using classes derived from `Article` to represent the different types of objects. Using appearance-based parameters, HMMs, and rules, article classes represent object properties and behaviors. Because the `Person` class provides a generic model of a person that is generally sufficient for most view-based applications, we rarely need to derive new classes from it. Instead, we invoke instances directly from the parent class to represent particular people. Each person object maintains a unique skin color distribution and anatomical parameters, like arm span, which help to track and analyze hand interactions with articles. For each new domain, we derive a new class from `Scene` to maintain domain-specific context, to monitor physical contact between people and articles, as well as to examine these interactions for patterns of behavior. Although only these three parent classes are described in the next several sections, we remind the reader that our implementation of ObjectSpaces includes several additional OO constructs. However, only the aforementioned classes are fundamental to our framework.

3.3.1 The `Article` Class

The `Article` class is the parent class from which all inanimate objects are derived. The idea is to equip this class with variables and functions that are appropriate for any type of object. Since all articles inherit the attributes of this class, we can ensure a level of consistency in our representation and organization of context that facilitates data sharing between objects and processes. Some of the variables that describe basic properties or states of an article are listed in Table 3.2, and some of the functions that are used to manipulate these variables are listed in Table 3.3. For example, Figure 3.6 highlights some of attributes of the `book` class.

An article records every episode of hand contact as a separate event, which is stored in an event array. Each event is time-stamped and include the person and hand involved in the contact (as shown in Figure 3.6. It also maintains a list of human actions (HMMs) associated with its use. Additional variables and functions can be defined as required.

Variables	Description
bounding box	two Cartesian points defining a box that encloses the object
label	object's type label, i.e., book
focus	level of attention, i.e., inactive, tentative, or active
frequency	frequency of hand contact
image template	image determined by bounding box
edge image template	result of passing image template through Sobel edge detector
moveable	(Boolean) true if article can be moved
rigid	(Boolean) true if article always maintains rigid shape
occluded	(Boolean) true if article has been occluded
events array	list of recorded hand interactions with object
action models	list of pre-trained action HMMs associated with object class

Table 3.2: `Article` class data variables for representing context, class properties, and behaviors.

GetActionName	SetActionName
GetArticleClass	SetArticleClass
GetOtherAssocClasses	SetOtherAssocClasses
GetNewToScene	SetNewToScene
GetArticleName	SetArticleName
GetBoundingBox	SetBoundingBox
GetImageTemplate	SetImageTemplate
GetSobelEdgeTemplate	ComputeSobelEdges
GetEvent	GetNumEvents
GetExitByID	SetExitByID
GetFocus	GetDurationOfFocus
GetIntroducedByID	SetIntroByID
GetActionModels	SetModelList
GetMoveable	SetMoveable
GetNumModels	SetModel
GetProbability	SetProbability
GetFirstSightingTime	SetFirstSightingTime
GetLastSeenTime	SetLastSeenTime
GetUpdateLocation	UpdateState
GetValue	SetValue
CheckPosition	FindNewPosition
RemoveArticle	CheckBBoxOverlap
CalcFrameDiff	BuildConnectedComponents

Table 3.3: `Article` class functions for manipulating data variables.

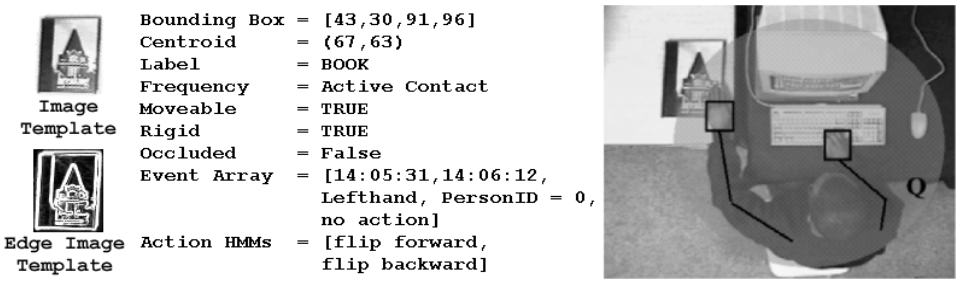


Figure 3.6: Structure for a `book` article. Note arm span (circular shaded area) and hand regions of person are superimposed.

3.3.2 The `Person` Class

The `Person` class works in tandem with the extraction layer to locate people based on a model of a person. A simple view-based representation is used to model the arm/hand components as well as the head/torso component, as seen in Figure 3.7. The former component is characterized by physical properties, such as hand size (in pixels) and skin color, as well as physiological considerations, like arm span (the maximum distance between the head/torso centroid and hand centroid). Skin color is described by an array $\mathbf{C} = [\mathbf{Y}\ \mathbf{U}\ \mathbf{V}]$ containing all of the flesh tones in the person's hands. This color distribution is used to assist in the segmentation of the hands. We elect to use the YUV color space instead of RGB because of its separation of the luminance and chrominance bands. In practice, we generate $\mathbf{C}$ by manually sampling pixel values taken from training footage of a subject. Other parameters are also specified manually.

The `Person` class stores all hand positions so that future locations can be estimated and supplied to the extraction layer to assist in tracking. The `Person` class is also equipped with methods for summarizing current activities by creating a log that contains the most current actions performed and articles handled.

3.3.3 The `Scene` Class

The `Scene` class "pays attention" to interactions between people and articles while measuring all dynamic changes in the state of the scene. Methods for establishing high-level

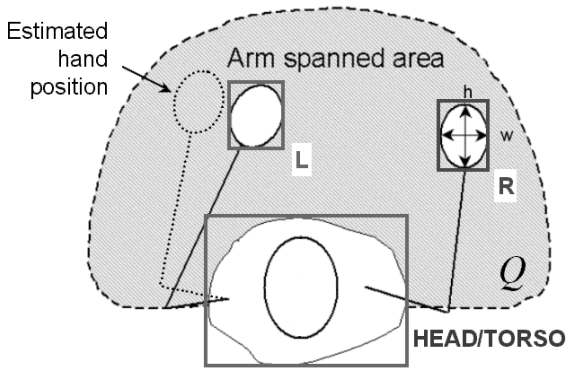


Figure 3.7: View-based model for a `Person` object is inspired by anatomical models. Here, Q represents the area spanned by the hands.

recognition from evidence collected in lower layers must be embedded in this class[1]. An instantiation of this class produces the only object in the scene layer.

To configure the scene, an initial snapshot of the background scene I is taken from the ceiling, looking downward. Using a mouse, n non-overlapping regions for each article are partitioned from I manually, such that $I_i(t) \subset I,\ I_i(t) \cap I_j(t) = \emptyset$, for all $i \neq j,\ 1 \leq i, j \leq n$. Figure C.4 shows the actual dialog used by ObjectSpaces to configure objects. $I_i(t)$ indicates the i^{th} partition of I taken at time t and also defines the boundaries of the corresponding bounding box. During initialization *(with no people in the scene and $t = 0$)*, each subimage template $I_i(0)$ and its edge image[2], $E_i(0)$, are assigned to an object of appropriate class type. For example, Figure 3.5 shows separately derived classes for the keyboard, mouse, phone, etc. `Scene` maintains a list of n scene articles $\mathbf{a} = \{a_1, a_2, ...a_n\}$ and m person objects $\mathbf{p} = \{p_1, p_2, ...p_m\}$.

3.4 Using Multiple Layers for Context Abstraction

We attempt to decompose context into various levels of granularity using multiple, integrated layers, as illustrated in Figure 3.2. The *extraction layer* is responsible for finding, extracting, and tracking people and articles in the scene. It also provides facilities for char-

[1] In Chapter 6, grammar is introduced as a means of generating high-level awareness. Facilities for symbol generation and grammar parsing are added to this class.

[2] From passing $I_i(0)$ through a Sobel edge filter [85].

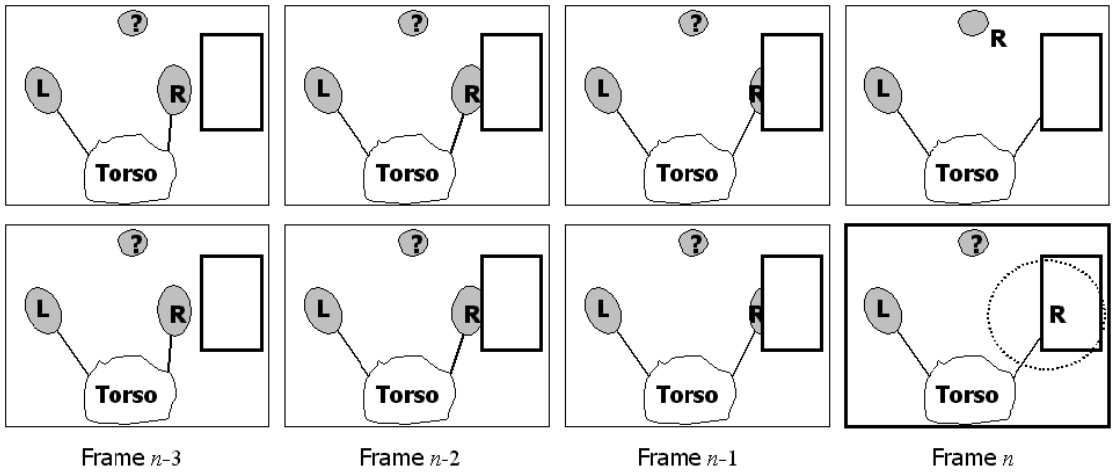


Figure 3.8: Schematic Diagram: Hand tracking algorithm labels colored regions to fit person model: (top sequence) With only knowledge of the input image, tracking mislabels region in frame n as hand is occluded by object; (bottom sequence) With scene and person information, tracking determines a possible perimeter where right hand can be even though it is occluded.

acterizing motion using hidden Markov models (HMMs). This layer generates context at the lowest level from raw images. The *object layer* represents the database of all object classes for a particular domain. The *scene layer*, which maintains domain-specific context, monitors physical contact between people and articles, as well as examines these interactions for patterns of behavior. It also keeps track of all people and articles as they are added or removed from the scene. All three layers create the backbone of an architecture for managing information at different levels of granulatity, i.e., low-level to high-level context.

Our approach features an bottom-up and top-down strategy with integrated layers, allowing mutual information to be shared when needed. For example, a person model in the object layer supplies the extraction layer with a color distribution and region shape parameters that help guide color blob segmentation and tracking. Likewise, blob trajectories that represent hand regions are passed back to the corresponding person object from the extraction layer after they have been properly labeled. This allows each person object to maintain a buffer of all of its hand positions. The liberal exchange of information throughout the layers using OO messaging[3] helps to offset limitations caused by strictly bottom-up or

[3]Messaging is the method of requesting or receiving information between objects in an object-oriented

top-down approaches. The complementary usage of context will enable this architecture to recover from failures or inconsistencies that occur at either the lowest or highest levels of abstraction in the system. Figure 3.8 (top sequence) depicts a scenario where the right hand is occluded, causing the hand tracking algorithm to incorrectly select another similarly colored candidate region. By sharing information about objects in the scene that occlude hands and the history of hand positions with an, otherwise, blind low-level correspondence tracker, we can resolve some occlusions, as illustrated in Figure 3.8 (bottom sequence).

Managing context in layers of abstraction also allows objects (articles and people) and their behaviors to be defined intrinsically, without regard for the domain in which they will appear. For example, a book will appear and be used in the same way regardless as to the current domain. Once developed, a class in our class database is available for reuse in multiple domains without the need to retrain actions associated with it. Behaviors between objects can be better specified at the domain-level, placing task dependencies at the highest level of abstraction. Moreover, the framework will require little or no retrofitting to accommodate a specific application's needs because domain-level context can be neatly compartmentalized in the scene layer. In Figure 3.4, several objects such as chairs, only have to be defined once, but can be reused in several domains.

3.4.1 The Extraction Layer

As mentioned, the extraction layer provides various facilities for tracking hand and article movement. Chapter 4 is dedicated to describing hand tracking and action model analysis. Here we present methods for tracking article displacement.

Contact between an article and a person's hand is established by checking for any overlap between the region bounding the article and the one surrounding the hand. The location of the bounding box is not updated while contact is established. Instead, we maintain a policy of waiting until the hands leave the last known bounding region before we check to see if the article has been moved.

If article i is moveable, its image template, I_i, is used by template-matching methods

paradigm.

to detect movement or occlusion[4]. The subimage defined by the current bounding box is compared with the image template captured during initialization by image subtraction and thresholding, i.e., $|I_i(x,y,t) - I_i(x,y,0)| > \alpha$, where α is a threshold found experimentally. To determine the object's new location, template-matching is invoked [138]. To match an $N_1 \times N_2$ pixel template, we use the minimum *mean square error* (MSE), which is defined as

$$MSE(d_1,d_2) = \frac{1}{N_1 N_2} \sum_{(u,v)\in\mathcal{B}} [I_i(u,v,0) - I_i(u+d_1, v+d_2, t)]^2, \tag{3.1}$$

where $\mathcal{B}$ denotes a $N_1 \times N_2$ block for a set of candidate motion vectors (d_1, d_2). The estimate of the motion vector is taken to be the value of (d_1, d_2), that minimizes the MSE, i.e.,

$$[\hat{d_1}\ \hat{d_2}]^T = \arg\min_{(d_1,d_2)} MSE(d_1,d_2). \tag{3.2}$$

Alternatively, we can substitute the MSE for the maximum *matching pixel count* (MPC), where each pixel within the block $\mathcal{B}$ is classified as either a matching or a mismatching pixel according to

$$T(u,v;d_1,d_2) = \begin{cases} 1 & \text{if } |I_i(u,v,0) - I_i(u+d_1,v+d_2,t)| \le \alpha \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where α is a predetermined threshold found from experiments. The motion vector (d_1, d_2) that produces the highest number of matching pixels is selected, i.e.,

$$MPC(d_1,d_2) = \sum_{(u,v)\in\mathcal{B}} T(u,v;d_1,d_2) \tag{3.4}$$

with

$$[\hat{d_1}\ \hat{d_2}]^T = \arg\max_{(d_1,d_2)} MPC(d_1,d_2). \tag{3.5}$$

The search for the object's new location is guided, to some extent, by our knowledge of the past hand locations, which helps to minimize exhaustive searches. Normally, however, we can get a good estimate of location by performing background subtraction against the current frame $I(x,y,t)$, revealing a region $\mathcal{Q} \in I$.

[4]The edge template E_i is used in lieu of I_i when prominent sensor noise or lighting fluctuations are present in images.

While this approach accommodates rigid objects that are translated with respect to the camera, recovering the location of rotated objects will likely fail without applying affine transformations to the template. To determine the angle of rotation, we determine the angle of the principle axis θ of $\mathcal{Q}$ given by

$$\theta = \frac{1}{2}\arctan\left(\frac{2\sum\limits_{(x,y)\in\mathcal{Q}} xy}{\sum\limits_{\mathcal{Q}} x^2 - \sum\limits_{\mathcal{Q}} y^2}\right), \tag{3.6}$$

where (x, y) are Cartesian points in region $\mathcal{Q}$. Application of a rotation transform $\mathbf{R}(\theta)$,

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \tag{3.7}$$

to I_i should produce a new template $\widehat{I}_i$, i.e., $\widehat{I}_i = \mathbf{R}(-\theta)I_i$, that can be used with the aforementioned block-matching methods to verify the object's new position. There is no treatment for recovering the orientation of objects with nonrigid shape.

3.4.2 The Object Layer

The object layer maintains the database of all currently instantiated objects as well as those that are stored. This layer assists the scene layer when detecting and classifying unknown objects. The model database is more carefully discussed in Section 3.5. Chapter 5 describes methods contained in this layer that are used for evaluating evidence for object classification.

3.4.3 The Scene Layer

This layer searches for relations between object interactions in order to classify particular activities or to identify certain human behaviors. For well-understood multitasked activities, we can specify the sequence of expected tasks using grammars. Grammar provides syntax and ordering so that a sequence of interactive events can be parsed and interpreted. We devote much of Chapter 6 to grammars and multitasked action interpretation.

Spatial context regarding the location of articles in the surroundings can be embedded into *activity zones* to facilitate tracking and recognition. Consider l non-overlapping zones

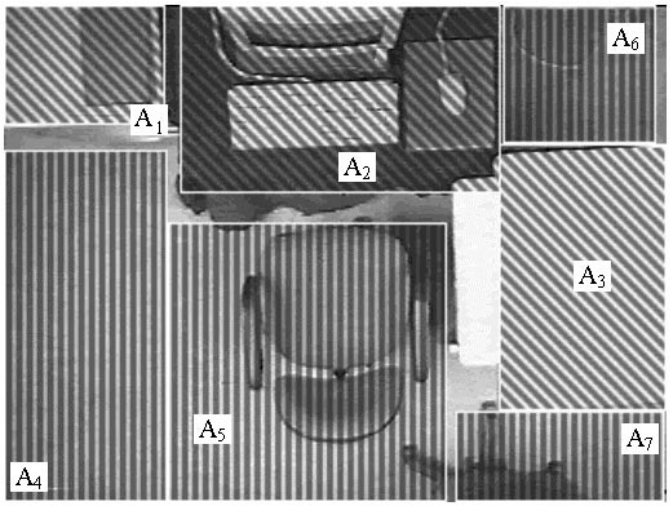


Figure 3.9: *Active Hand Zones:* Zones with vertical lines indicate areas less likely to find hand regions while diagonal zones suggests more likely zones.

within I, each specified by the top-left and bottom-right points of a bounding box given by the 4-element vector $\mathbf{z}_i$, such that

$$\mathbf{z}_i = [x_l \ y_t \ x_r \ y_b]^T.$$

The probability w_i reflects the likelihood of finding a hand in the zone. The probability w_i does not explicitly weigh the frequency of hand penetration in the zone relative to other zones, but suggests the likelihood of interactions taking place in zone $\mathbf{z}_i$. Each probability is determined independently. The $4 \times l$ matrix $\mathbf{Z}$ contains the set of all l zones, i.e., $\mathbf{Z} = [\mathbf{z}_1 \ \mathbf{z}_2 \ldots \mathbf{z}_l]$, while the vector $\mathbf{w}_z$ holds the corresponding probabilities. For more clarity, consider Figure 3.9, which illustrates the likelihoods of recovering hand blobs in different areas within the scene. Because there are articles located in zones marked A_1, A_2, and A_3 where we expect more hand traffic, the corresponding likelihoods are higher than those in zones A_4, A_5, A_6, and A_7 where traffic is not as likely, i.e., $w_1, w_2, w_3 > \frac{1}{2}$ and $w_4, w_5, w_6, w_7 < \frac{1}{2}$. All other areas not covered by zones remain equiprobable, i.e., $p = \frac{1}{2}$. Note in this case that the presence or absence of articles affects how the extraction layer and the person models use information when tracking hands.

3.4.4 Object-based Event Handling

Article bounding boxes provide a means to detect and focus on motion near a specific article. Associated with each object is a finite state machine driven by contact with a person. Initial contact with an object forces a transition from the *inactive* state to the *tentative* state. If the contact turns out to be transient, the object returns to the inactive state; otherwise, it progresses to the *active* state, which initiates a *contact event*. A contact event is a record that maintains information about the interaction, such as the duration, the person performing the action, and a description of the action (provided via HMMs), if available. The record also records the last known position and orientation of the object so that any displacement can be detected later.

To gather a better description of hand motion, the article retrieves hand positions from each person that interacts with it. It passes a buffer containing these positions to the extraction layer which attempts to match motion trajectories against pre-trained actions associated with the article class. For example, consider a `book` object with two associated actions described by HMMs, i.e., *left-to-right* motion $\rightarrow \lambda_{ff}$: "flip forward" and *right-to-left* motion $\rightarrow \lambda_{fb}$: "flip backward". After the hands penetrate the image area where the book is located (establishing contact), sweeping the hand from left to right will indicate flipping a page forward. More will be said about the use of HMM models for motion analysis in Chapter 4.

Object-centered interaction allows objects to focus on themselves, freeing the scene layer to watch for events requiring higher-level information. For example, newly introduced articles can query the scene for all person objects. In this request, they can ask for the most recent hand locations to determine which person was responsible for placing it in the scene. This benefit can only be provided by an object-oriented framework. In Chapter 5, we cover this technique along with other aspects of focusing attention on action and scene change.

3.5 Generalized Class Models

A repository is required to manage and categorize all of the article classes. This database contains models of class types that are used to classify unknown objects that enter the scene. These models, called *generalized class models* (GCMs), contain appearance-based descriptions that are representative of all examples of that particular class. A GCM is created for every child level within the class hierarchy. For example, the `book` GCM is created from several examples of actual `book` objects, as Figure 3.3 illustrates. Descriptions of associated class actions, in this case flip forward λ_{ff} and flip backward λ_{fb}, are maintained as a list of HMMs. Appearance-based descriptions are maintained by a set of Gaussian parameters that represent typical values of region features, i.e., $\mathbf{P} = \{(\mu_{pixel\ area}, \sigma^2_{pixel\ area}), (\mu_{aspect\ ratio}, \sigma^2_{aspect\ ratio}), (\mu_{perimeter\ edges}, \sigma^2_{perimeter\ edges})\}$. The Gaussian parameters for these distributions, mean μ and variance σ^2, are calculated in the normal way, such that

$$\mu = \frac{1}{n}\sum_{i=1}^{n} f \qquad \sigma^2 = \frac{n\sum_{i=1}^{n} f^2 - \left(\sum_{i=1}^{n} f\right)^2}{n^2}, \tag{3.8}$$

where f is some feature value, such as pixel area, and n is the number of actual examples used to model the GCM. GCMs of child classes are independent of GCMs of the parent class. The notebook class extends book by adding an action model for "write," λ_{wrt}. Because the book GCM is a parent, $\mathbf{P}_{book}$ is influenced by the contribution of its children, which includes notebook, although $\mathbf{P}_{notebook}$ bears no dependence on $\mathbf{P}_{book}$. These distributions allow us to represent appearance-based features, such as bounding box aspect ratio and pixel area, using a Gaussian parameters. Our class database is composed of the entire set of GCMs, which forms model space $\mathbf{M}$, such that $\mathbf{M} = [M_{book}\ M_{notebook}\ \ldots]^T$.

In this chapter, we have given a broad overview of our framework for information management and process decomposition. In Chapter 4, we introduce methods for recognizing different interactions which are staged in the extraction layer. When classification of objects and activities is discussed in Chapters 5 and 6, we will discuss techniques that take place within the scene layer.

CHAPTER 4

Hand Motion Analysis

In this chapter, we address the problem of detecting, tracking, and characterizing hand motion. Determining *where* a person's hand are and *when* some meaningful motion pattern occurs is a critical step in solving action recognition problems. By also exploiting prior knowledge regarding objects that are in the surroundings, we can establish a reasonable basis for modeling a person's interactions with objects. Moreover, the recovery of contact events and actions supplies us with primitives that can be assembled to describe complex activities and behavior.

To detect hands, we rely on stable, appearance-based features, such as color, shape, and size, and heuristics to segment candidate hand regions from images. We introduce an efficient technique that uses block-based image sampling to reduce grouping and labeling operations in connected component labeling of sparse binary pixels. Hand tracking takes advantage of environmental context and Bayesian analysis to enhance linear predictions. Scene and domain context are also leveraged, enabling real-time hand tracking to recover from temporary failures. Hidden Markov models are employed to recognize trained hand actions under a variety of conditions including spatial and temporal variation. We begin with an overview of our strategy to track people and their hands.

4.1 Tracking People

One of our objectives is to capture interactions between people and objects in the environment. The great majority of these interactions involve hand contact. While there may be an occasional need to perform whole-body tracking as people move throughout the scene, our tracking algorithms will emphasize hand-based methods. We assume that our view of

the scene is provided by a downward-pointing, ceiling-mounted camera which offers several advantages to hand and object tracking, such as a less unobstructed perspective of the interactions.

The facilities to segment, group, label, and analyze hand features from video reside in the extraction layer of the framework. While feature parameters and object locations must be specified for each scene, these necessary operations are procedural, requiring little, if any, modification from scene to scene. This is a primary motivation for having a low-level layer in our framework for routine tasks like hand tracking and a high-level layer that can be tailored to fit a specific domain. Moreover, the segregation of low-level and high-level operations encourages model reuse and deployment for various recognition tasks.

There are a vast array of sophisticated techniques for tracking that have enjoyed widespread adoption, such as template matching, Kalman filtering, and Condensation. Tracking hands *as they interact with objects* can not be performed effectively using image templates due to the degrees of freedom of both hand and object movement. Deformable templates, perhaps that use energy-minimizing contours, work better, but often require hand-drawn initialization. When combined with classifiers like 3D splines, the Kalman filter is a powerful method for tracking hands even with noisy measurements, but can be easily confused when hands collide or other hand-like regions appear in the image. Condensation overcomes this problem by using multimodal Gaussian distributions to maintain multiple hypotheses, but requires that prior densities be estimated from training exercises. Moreover, the computation and memory requirements of these methods are demanding, making them less attractive for real-time systems. Motivated, in part, by the Kalman gain's corrective adjustment to a deterministic motion model, we introduce a simpler, direct (non-iterative) approach that exploits scene context and user intentions to enhance a linear tracking algorithm. Using primarily appearance-based features, our tracking algorithm is appropriate for efficient tracking in real-time and scales easily as multiple objects and hands are added to the scene.

4.1.1 Finding People

After classes have been defined and the scene configured, tracking begins by looking for the people. Background subtraction is a basic technique used to identify areas of change in an image. It is easily performed by subtracting the background frame $I(x, y, 0)$[1] from the current image frame $I(x, y, t)$. To make this procedure less expensive, this operation takes place only in the portions of the image where foot traffic is permissible, denoted $\mathcal{W}$. Using α_{bkg} as a preset threshold to assess change in each pixel, the scene's binary background difference $I_{BD}(x, y, t)$, i.e.,

$$I_{BD}(x, y, t) = \begin{cases} 1 & \text{if } |I(x, y, t) - I(x, y, 0)| > \alpha_{bkg} \\ 0 & \text{otherwise} \end{cases}, \tag{4.1}$$

detects a person walking into the scene by comparing the sum of the difference pixels to a threshold α_p, i.e.,

$$\text{Person detected} \approx \begin{cases} true & \sum_x \sum_y I_{BD}(t, x, y) > \alpha_p \ \forall \ x, y \in \mathcal{W} \\ false & \text{otherwise} \end{cases}.$$

Both α_{bkg} and α_p are established empirically depending on sensor noise, lighting conditions, head/torso parameters, and the size of region $\mathcal{W}$. Although I represents a color image, only the intensity channel (Y in YUV) is used.

4.1.2 Finding and Tracking the Hands

Once a person is detected in the scene, color is the basis for the recovery of the hands. Generally, we assume that the subject is wearing a long-sleeve shirt that offers sufficient contrast with skin color. We offer no solution for scenarios when subjects wear short-sleeve shirts or when exposed skin on the head causes confusion with hands. We segment colored blobs from the image using the color distribution $\mathbf{C}$, which was mentioned in Section 3.3.2. Skin colors are segmented from $I(x, y, t)$[2] by rifling through every pixel in the image looking

[1] Taken at initialization when $t = 0$ with no people in the scene.
[2] Each image pixel in I represents a color in YUV Space.

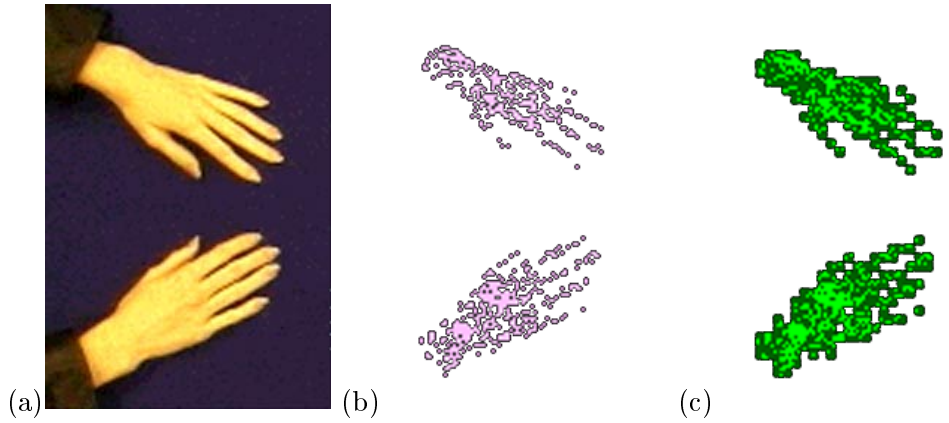


Figure 4.1: *Connected component labeling:* (a) Original image I, (b) Color-based segmentation using an under-specified $\mathbf{C}$ produces B with many sparse pixels and several potential groups (c) Sub-sampling with $\alpha_m = 1$ and $m = 3$ of B generates $\widehat{B}$ and two desirable groups.

for members of $\mathbf{C}$, which produces a binary image B given by

$$B(x, y, t) = \begin{cases} 1 & \text{if } I(x, y, t) \in \mathbf{C} \\ 0 & \text{otherwise} \end{cases} . \tag{4.2}$$

Connected Component Labeling (CCL) is a well known procedure that groups all neighboring binary pixels with value 1 into individual regions and attaches a unique label to each [63]. Because the colors in $\mathbf{C}$ are manually selected from a few sample images, there may be some skin colors that are not represented in the distribution. Additionally, lighting fluctuations and shadows prevent $\mathbf{C}$ from entirely modeling all skin colors, so binary image B often contains many outlying pixels and fragments of broken pixel groups. Under these conditions, merging and managing many small pixel groups by iterative clustering approaches, such as the K-means algorithm, is computationally expensive and slow. Moreover, these methods tend to perform poorly without supervision or well-specified linear decision functions[3].

To cohere sparsely distributed pixels, we invoke a preprocessing step that uses block-based sampling before CCL. We generate $\widehat{B}$, a sub-sampled version of binary image B, using an $m \times m$ block, where m is an odd number such as $3, 5, 7, \ldots$. If B is an $N_x \times N_y$ matrix,

[3] A linear decision function is typically a line or surface used as a decision boundary when attempting to classify cluster data. See Tou for more information on minimum-distance pattern classification concepts [139].

consider two sets of integers denoted as $L_1 = \{\tau, \tau + m, \tau + 2m, \tau + 3m, \ldots\} \leq N_x - 1$ and $L_2 = \{\tau, \tau + m, \tau + 2m, \tau + 3m, \ldots\} \leq N_y - 1$ where $\tau = \frac{m-1}{2}$. We define the $\frac{N_x}{m} \times \frac{N_y}{m}$ matrix $\widehat{B}$ as

$$\widehat{B}(i,j,t) = \begin{cases} 1 & \text{if } \sum_{u=i-\tau}^{i+\tau} \sum_{v=j-\tau}^{j+\tau} B(u,v,t) \geq \alpha_m \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, i \in L_1, j \in L_2 \tag{4.3}$$

where α_m is a threshold that we determine empirically based on the coherence of pixels in B. For example, the distribution $\mathbf{C}$ used to segment the hands in Figure 4.1-a results in many poorly connected pixel groups, as shown in Figure 4.1-b. Using a 3×3 block to sample B and a low threshold of $\alpha_m = 1$, which turns "on" a block containing even one pixel, the sub-sampled image $\widehat{B}$ produces only two regions. By adjusting the block size and the density of pixels, i.e., manipulating m and α_m, respectively, cohesive pixel groups can be generated. Sub-sampling also reduces the computation and memory requirements of the connected components algorithm. Figure 4.2 shows the number of operations involved in grouping and labeling operations of CCL for a 500 frame sequence using various block sizes. Although the bulk of the operations take place during initial thresholding[4], switching to larger block size reduced operations by almost 90%.

Connected component labeling produces a set of k binary regions r_i from $\widehat{B}$. Due to the presence of similarly colored objects in the room, many of these blobs are not hands, as shown in Figure 4.3. We use context information maintained by the appropriate objects in our framework to help us classify regions. Let $\Psi = \{area_{min}, area_{max}, ratio_{min}, \mathbf{w}_z, \mathbf{Z}, \Omega_e\}$ represent a set of parameters supplied by the person class and the scene layer to provide an idea of what a hand should look like and where it should be located in the scene. These parameters are used by heuristic filters that measure region features, such as pixel area, elongation, overlap with activity zones, and the distribution of edges in the hand. In comparison to deformable templates and splines, these features are stable, reliable, and moderately invariant to a variety of imaging conditions, i.e., illumination changes, distortion from perspective foreshortening, etc., especially over extended periods of time. From

[4]Thresholding always involves every image pixel, regardless if block samples takes place, i.e., no savings in computation, memory, etc. However, many vision systems design assembly code or special hardware to perform this operation.

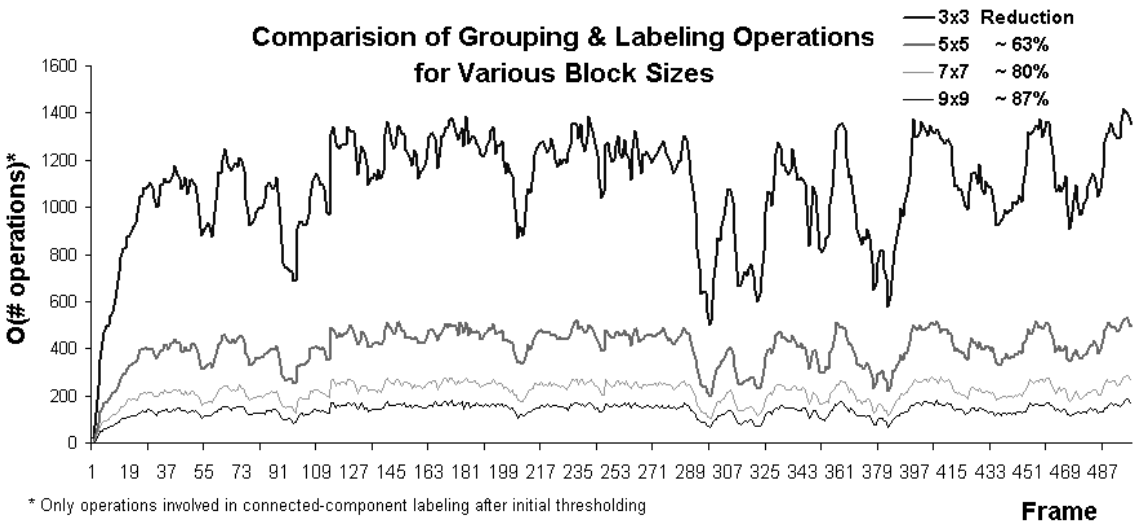


Figure 4.2: Comparison of the number of operations used in Connected Component Labeling for various block sizes. For a 500 frame sequence of a pair of hands, increasing block size from 3×3 to 5×5, 7×7, or 9×9 reduces operations by 63%, 80%, and 87%, respectively.

experimentation, we argue that they are sufficient for distinguishing hands from non-hands, barring failures with color segmentation[5]. Such basic features are also attractive because they can be consistently recovered over several frames, which is critical to establishing region correspondence and tracking.

For each candidate region r_i, we seek to multiple the output scores of each heuristic filter of Ψ. Scores are determined using heuristics developed from empirical examination. Parameters of Ψ are assumed to be independent so that we can evaluate each feature separately, as described below:

- Pixel area: p_{area}, i.e.,

$$p_{area} = \begin{cases} 1 - \frac{area_{min} - area_{r_i}}{area_{min}} & area_{r_i} < area_{min} \\ 1 & area_{min} \leq area_{r_i} \leq area_{max} \\ 1 - \frac{area_{r_i} - area_{max}}{area_{max}} & area_{r_i} > area_{max} \end{cases}, \tag{4.4}$$

where $area_{min}$ is the minimum number of pixels in r_i and $area_{max}$ is the maximum.

[5]Isolating the hands using color-based segmentation can fail when the hands share similar colors with the environment or when other fresh-baring parts of the body are exposed.

(a)  (b) 

Figure 4.3: (a) Original image, (b) Color-based segmentation and grouping of regions.

- Elongation of bounding box: $p_{elongation}$, i.e.,

$$p_{elongation} = \begin{cases} 1 & ratio_{r_i} \geq ratio_{min} \\ \left[1 + \dfrac{ratio_{r_i}^2}{ratio_{min}^2}\right]^{-1} & ratio_{r_i} < ratio_{min} \end{cases}, \text{ where} \tag{4.5}$$

$$ratio_{r_i} = \frac{\min(x_r - x_l, y_b - y_t)}{\max(x_r - x_l, y_b - y_t)}$$

and $ratio_{min}$ is the smallest aspect ratio between the short and long sides of r_i's bounding box $\mathbf{z}_{r_i} = [x_l \ y_t \ x_r \ y_b]^T$ as defined in Section 3.4.3. In practice, $ratio_{min} \geq 0.25$, depending on the size of hands. The probability of region elongation smaller than $ratio_{min}$ experiences a second-order roll-off because the likelihood becomes exceedingly lower.

- Overlap with activity zone bounding box: p_{zone}, i.e.,

$$\begin{aligned} p_{zone} = & \sum_{j=1}^{l} w_j \frac{area(\mathbf{z}_{r_i} \cap \mathbf{z}_j)}{area(\mathbf{z}_{r_i})} \\ = & \sum_{j=1}^{l} w_j \frac{[\min(x_r, x_r(j)) - \max(x_l, x_l(j))][\min(y_b, y_b(j)) - \max(y_t, y_t(j))]}{(x_r - x_l)(y_b - y_t)} \end{aligned} \tag{4.6}$$

where $\mathbf{z}_j = [x_l(j) \ y_t(j) \ x_r(j) \ y_b(j)]^T$ is the bounding box of zone j. p_{zone} factors the percentage overlap between r_i's bounding region and zone $\mathbf{z}_j$ and the corresponding likelihood w_j, where $0 \leq w_j \leq 1$. Using this feature, the proportional contribution of

zone probabilities is used to determine hand likelihood based on the location of r_i's bounding box in the image.

- Edge pixel ratio between the inner and outer regions: p_{edge}

$$edge_{r_i} = \frac{\sum_{v=y_t+\tau_y}^{y_b-\tau_y} \sum_{u=x_l+\tau_x}^{x_r-\tau_x} E_{r_i}(u,v)}{\sum_{v=y_t}^{y_b} \sum_{u=x_l}^{x_r} E_{r_i}(u,v) - \sum_{v=y_t+\tau_y}^{y_b-\tau_y} \sum_{u=x_l+\tau_x}^{x_r-\tau_x} E_{r_i}(u,v)} \tag{4.7}$$

where E_{r_i} represents the edge map of r_i produced by a Sobel edge filter. Given enough contrast between the hand r_i and background, there should be substantial edge information around the periphery of the hand with minimal edge pixels in the interior. We also calculate the edge pixel count along the bounding box perimeter as

$$perimeter_{r_i} = \frac{\sum_{v=y_t}^{y_b} \sum_{u=x_l}^{x_r} E_{r_i}(u,v) - \sum_{v=y_t+\tau_y}^{y_b-\tau_y} \sum_{u=x_l+\tau_x}^{x_r-\tau_x} E_{r_i}(u,v)}{2(x_r - x_l) + 2(y_b - y_t)}. \tag{4.8}$$

p_{edge} offers a measure of hand likelihood given edge information and is defined as

$$p_{edge} = \begin{cases} 1 & edge_{r_i} < \omega_4, perimeter_{r_i} \geq \omega_5 \\ \omega_1 & edge_{r_i} < \omega_4, perimeter_{r_i} < \omega_5 \\ \omega_2 & edge_{r_i} \geq \omega_4 \\ \omega_3 & perimeter_{r_i} = 0 \end{cases}. \tag{4.9}$$

The variables of $\Omega_e = \omega_1, \omega_2, \ldots \omega_5$ are determined after substantial testing, but are not necessarily optimal. In practice, their values depend on the amount of contrast in intensity between hands and surrounding surfaces. Typical values are $\omega_1 = 0.66, \omega_2 = 0.60, \omega_3 = 0.50, \omega_4 = 0.1$, and $\omega_5 = 1.9$.

The total appearance-based score for region r_i given Ψ is calculated as

$$p_{r_i \to \Psi} = p_{area} \cdot p_{elongation} \cdot p_{zone} \cdot p_{edge}. \tag{4.10}$$

The process of color segmentation, connected component labeling, and feature analysis of candidate regions is illustrated in Figure 4.4. Unfortunately, Equation 4.10 only helps to remove some of the ambiguity of mapping regions to actual hands. In Section 4.1.3, we

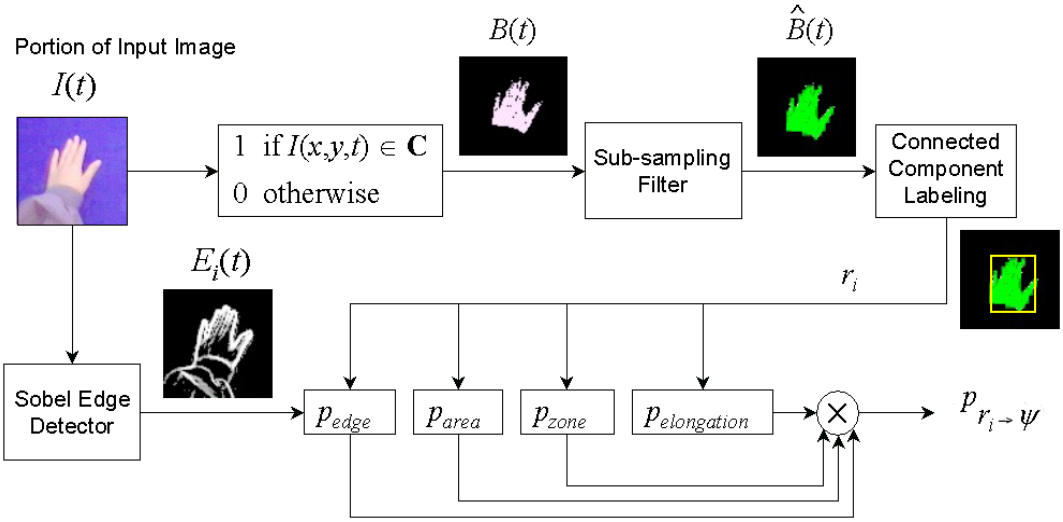


Figure 4.4: Diagram illustrates heuristic filters used to determine likelihood of a colored blob given pre-defined appearance-based parameters.

introduce hand centroid estimates that are provided by context-enhanced linear prediction. Estimates for the left and right hands of person p are denoted as $\hat{\mathbf{x}}_{l,p}$ and $\hat{\mathbf{x}}_{r,p}$, respectively. To establish region correspondence from frame-to-frame, we measure the distance between each region centroid and the the estimated hand centroid of each person, which is written as $d(\mathbf{x}_{r_i}, \hat{\mathbf{x}}_{l,p})$ for the left hand and $d(\mathbf{x}_{r_i}, \hat{\mathbf{x}}_{r,p})$ for the right. These distances form the sets given by

$$\mathbf{d}_{l,p} = \{d(\mathbf{x}_{r_1}, \hat{\mathbf{x}}_{l,p}), d(\mathbf{x}_{r_2}, \hat{\mathbf{x}}_{l,p}), \ldots, d(\mathbf{x}_{r_k}, \hat{\mathbf{x}}_{l,p})\} \tag{4.11}$$

$$\mathbf{d}_{r,p} = \{d(\mathbf{x}_{r_1}, \hat{\mathbf{x}}_{r,p}), d(\mathbf{x}_{r_2}, \hat{\mathbf{x}}_{r,p}), \ldots, d(\mathbf{x}_{r_k}, \hat{\mathbf{x}}_{r,p})\}. \tag{4.12}$$

Selecting the minimum-distance pair for each hand works towards solving the correspondence problem. This is a linear regression process that selects the pairings based on minimizing the least-square error. However, the minimum-distance pair is not always the correct choice, especially if there is evidence that our estimates are poor. We calculate time-weighted error to assess our confidence in the accuracy of our estimates. Confidence is established by adding the normalized minimum-distance error with prior confidence scores, such that

$$c_{l,p}(i,t) = \beta_c \frac{\max\{d_Q - d(\mathbf{x}_{r_i}, \hat{\mathbf{x}}_{l,p})\}}{d_Q} + (1-\beta_c) c_{l,p}(i, t-1) \tag{4.13}$$

$$c_{r,p}(i,t) \quad = \quad \beta_c \frac{\max\{d_Q - d(\mathbf{x}_{r_i}, \hat{\mathbf{x}}_{r,p})\}}{d_Q} + (1-\beta_c)c_{r,p}(i,t-1), \tag{4.14}$$

where d_Q is the maximum allowable distance from the head/torso center and β_c is a weighing constant. Note that estimation error is normalized relative to arm span d_Q. Typically, $\beta_c \geq 0.85$ so that estimation error is not propagated excessively. We combine appearance-based probability and our confidence in estimates with minimum-distance pairs to make the final correspondence decisions given by

$$\arg\min_{i \neq j} \left\{ \sum_{p=1}^{m} c_{l,p}(i,t) d(\mathbf{x}_{r_i}, \hat{\mathbf{x}}_{l,p}) p_{r_i \to \Psi} + c_{r,p}(j,t) d(\mathbf{x}_{r_j}, \hat{\mathbf{x}}_{r,p}) P(r_j|\Psi) \right\}, \; 1 \leq i,j \leq k. \tag{4.15}$$

These arguments return the region indices for each hand. A region is typically paired with only one hand, but can be matched to more in special cases. In our implementation, we have added heuristics that examine features such as area to test for blobs that are the likely result of two merging regions, i.e., the person has placed both hands together, and hence, has two hands corresponding to the same blob. These heuristics also evaluate motion estimates and scene context to detect occlusions or episodes when the hands move outside the field of view of the camera. The evaluations above are used in an adaptive, tracking algorithm that maximizes the likelihood of matching a candidate to the respective hand, depending on the number of people that are believed to be in the scene.

4.1.3 Context-Enhanced Linear Prediction of Hand Centroids

To estimate future hand centroids, we use linear prediction along with context from the environment. The linear, estimated hand centroid is given by

$$\bar{\mathbf{x}}_{t+1} = \mathbf{x}_t + d\mathbf{x}, \tag{4.16}$$

where $d\mathbf{x} = \mathbf{x}_t - \mathbf{x}_{t-1}$. When combined with our minimum-distance correspondence algorithm, this simple discrete-time filter performs surprisingly well during gradual, well-behaved hand motion. At present, higher-order motion models are avoided because they introduce lag. We attempt to reinforce simple estimates using knowledge of the surroundings.

Although hand motion is generally unconstrained, we assume that people tend to be explicit with regard to their interactions with articles. Scene-level context is leveraged

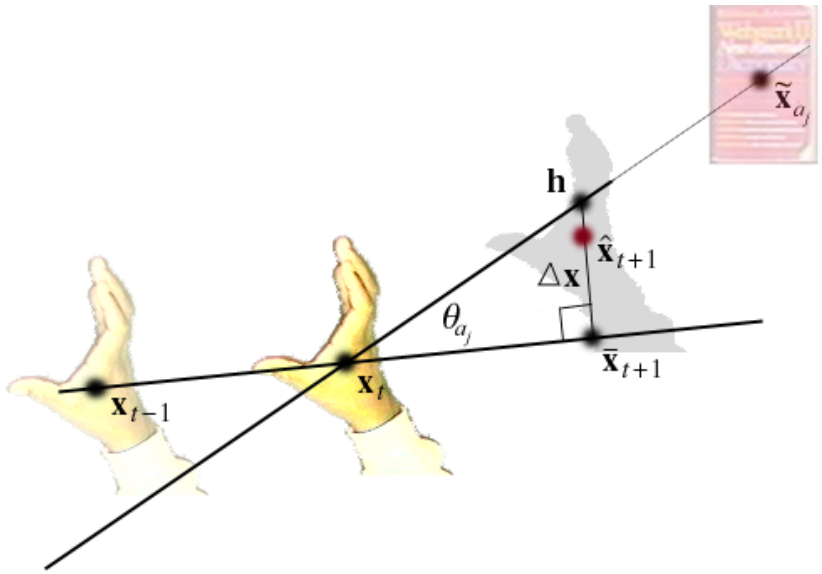


Figure 4.5: *Context-enhanced estimation*: Enhanced predictor $\hat{\mathbf{x}}_{t+1}$ slides between $\bar{\mathbf{x}}_{t+1}$ (no context, i.e., $\zeta = 0$) to $\mathbf{h}$ (max. context, $\zeta = 1$).

to extract information that may reflect a user's intentions, such as for which object he or she may be reaching. We have identified three principal factors that can be measured to quantify intention:

- the direction of hand motion,
- the distance between the hand and neighboring articles, and
- the likelihood of moving from one article to another.

Figure 4.5 illustrates the concept of providing enhancements to the linear prediction given by $\bar{\mathbf{x}}_{t+1}$. Using geometry to model hand trajectories, we calculate the line-of-sight estimate $\mathbf{h} = [x_h \ y_h]^T$, the point located along the line-of-sight path between the current hand location position $\mathbf{x}_t$ and the centroid $\tilde{\mathbf{x}}_{a_j}$ of article a_j. The line established by linear prediction contains $\mathbf{x}_t$ and $\bar{\mathbf{x}}_{t+1}$. To determine $\mathbf{h}$, we solve for the intersection of the line-of-sight path and the line perpendicular to the linearly predicted path. This perpendicular line, which also intersects the linearly predicted path at $\bar{\mathbf{x}}_{t+1}$, contains the segment $\triangle\mathbf{x}$.

We begin by defining $\mathbf{h}$ as

$$\mathbf{h} = \begin{bmatrix} x_h \\ y_h \end{bmatrix} = \begin{bmatrix} \dfrac{\bar{y}_{t+1} - m_1\bar{x}_{t+1} - y_t + m_2 x_t}{m_2 - m_1} \\ \dfrac{m_2(\bar{y}_{t+1} - m_1\bar{x}_{t+1} - y_t + m_2 x_t)}{m_2 - m_1} + y_t - m_2 x_t \end{bmatrix} \tag{4.17}$$

where

$$m_1 = -\frac{\bar{x}_{t+1} - x_t}{\bar{y}_{t+1} - y_t} \text{ and } m_2 = \frac{\tilde{y}_{a_j} - y_t}{\tilde{x}_{a_j} - x_t},\ \forall\ m_1 \neq m_2,\ \bar{y}_{t+1} \neq y_t,\ \tilde{x}_{a_j} \neq x_t.$$

Wherever $\mathbf{h}$ is undefined, i.e., $m_1 = m_2$, the enhancement factor added to the simple estimate $\bar{\mathbf{x}}_{t+1}$ is zero. Otherwise, we add a context factor ζ to the simple estimate. The linear, context-enhanced estimate $\hat{\mathbf{x}}$ becomes

$$\hat{\mathbf{x}}_{t+1} = \begin{bmatrix} \hat{x}_{t+1} \\ \hat{y}_{t+1} \end{bmatrix} = \begin{bmatrix} \zeta(i)\triangle x + \bar{x}_{t+1} \\ \zeta(i)\triangle y + \bar{y}_{t+1} \end{bmatrix}, \tag{4.18}$$

where $\triangle\mathbf{x} = [\triangle x\ \triangle y]^T$ denotes the displacement between the simple estimate $\bar{\mathbf{x}}_{t+1}$ and the line-of-sight estimate $\mathbf{h}$, and $\zeta(i)$ weighs the likelihood that a_j is the destination article given that a_i was the last article touched. We establish this likelihood as

$$\zeta(i) = \max_{1 \leq j \leq n} \left\{ k_{ij} \left[1 - \frac{\theta_{a_j}}{360^\circ} \right] \left[1 - \frac{\|\bar{\mathbf{x}} - \tilde{\mathbf{x}}_{a_j}\|}{d_Q} \right] \right\}. \tag{4.19}$$

The first term of Equation 4.19, k_{ij}, represents the probability of a contact event with the j^{th} article given that the previous contact event was with the i^{th} article, i.e., $k_{ij} = P(a_j(t+1)|a_i(t))$ where time $t+1$ is not necessarily the next time instant but the next future event. More is discussed about k_{ij} in Section 5.1.2. The second term quantifies the direction of hand motion relative to nearby article a_j by calculating θ_{a_j}, the angle between the line-of-sight estimate and the linear prediction, which is given by

$$\theta_{a_j} = \arctan\left(\frac{\|\triangle\mathbf{x}\|}{\|d\mathbf{x}\|} \right). \tag{4.20}$$

The final term in Equation 4.19 is proportional to the distance the hand is to potential target a_j. So in Equation 4.18, as more evidence that a person's hand is moving towards a_j becomes available, the more the final estimate moves from its linear prediction at $\bar{\mathbf{x}}_{t+1}$ to the line-of-sight prediction $\mathbf{h}$.

Context-enhanced estimation offers the greatest improvement when articles have already been placed in the environment. Figure 4.6-a shows the per frame estimation error,

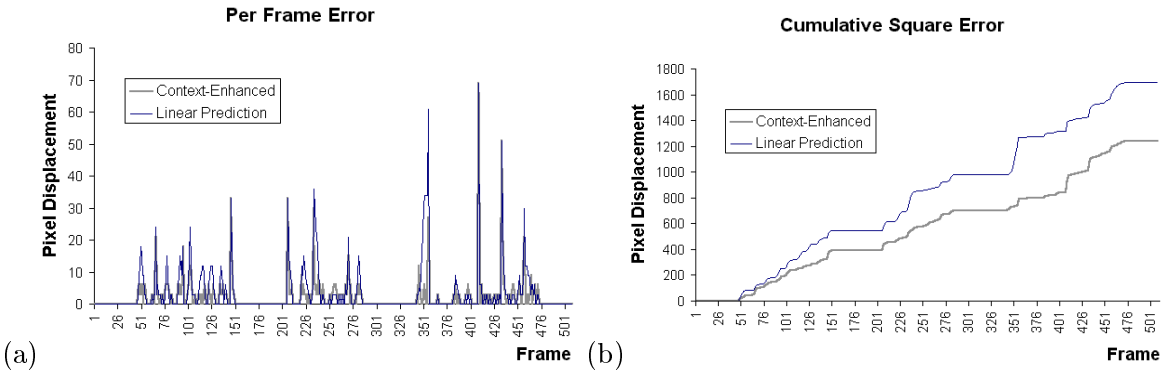


Figure 4.6: *Linear Prediction versus Context-Enhanced Estimation*: (a) Per Frame Error (b) Context-Enhanced estimation reduces cumulative square error by 26.68% over Linear Prediction.

i.e., $\mathbf{x} - \hat{\mathbf{x}}$, of the simple estimate versus the context-enhanced estimate. The improvement provided by context-enhanced estimation is shown more clearly in Figure 4.6-b, which shows that the cumulative square error over the entire sequence is reduced by almost 27%. Informal experiments using context-enhanced estimates in a variety of scenes show cumulative square error can be reduced by as much as 63%. Factors affecting performance include the size, location, and number of articles as well as arm span and hand velocity.

Context-enhanced estimation is an example of how ObjectSpaces facilitates information sharing, i.e., providing the location of near-by objects, which improves other processes, in this case, low-level tracking. Context-enhanced estimation combined with appearance-based correspondence can select the proper regions from a set of potentially confusing candidates as shown Figure 4.3-b. When confidence is high and good hand features are available, tracking can be maintained even if the hands wander into activity zones with a low likelihood for hand traffic. However, if only weak image features are available, i.e., $P(area_{r_i}|area_{min}, area_{max}) \rightarrow 0$, the value of the information provided by the activity zones is more heavily considered. This kind of "reasoning" is possible because of several confidence functions that measure uncertainty and our top-down, bottom-up framework.

Using context and appearance-based measurements, hand tracking can be sustained in the presence of pronounced movement, variations in the lighting conditions, or temporary occlusions. Automatic tracking of multiple hands using Kalman filtering requires instantiating a separate Kalman filter for each candidate hand region. Although Isard et

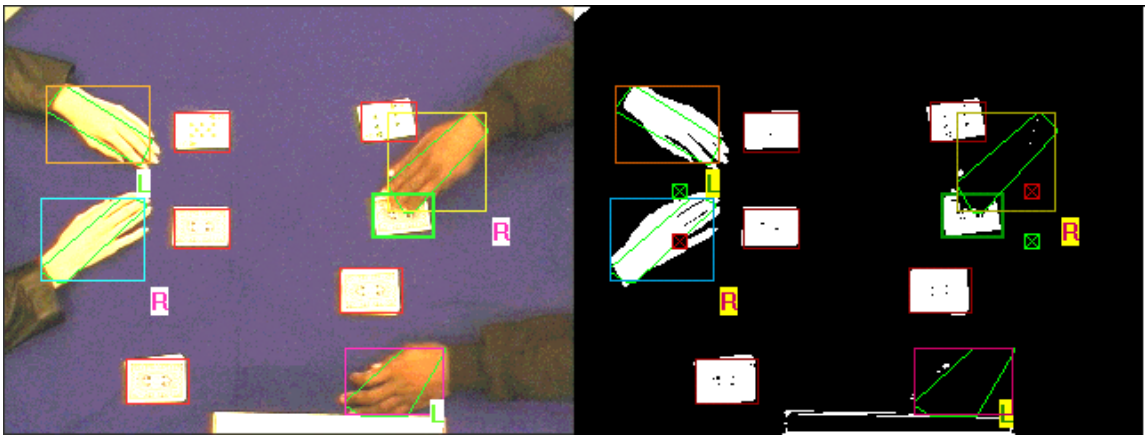


Figure 4.7: Hand-object contact: Extremal points provide better region perimeters than bounding boxes for various hand orientations.

al. claim that Condensation provides simultaneous tracking of multiple hypotheses [71, 72], it does not solve the correspondence problem. Using confidence scores from Equation 4.13 to exploit physiological structure, our approach weighing the likelihood of regions to establish correspondence. Moreover, we achieve good performance using context-enhanced estimation and tracking without the computational resources required by these methods.

4.1.4 Detecting Interaction with Articles

Throughout hand tracking, the scene layer maintains an inventory of people **p** and articles **a** in the scene. We use a fixed, overhead color CCD camera that is pointed downward to provide a view where the location of people and objects can be clearly determined. In comparison to a lateral view of the scene, this perspective is less obstructive and less prone to occlusions caused by people moving in front of the camera (shown in Figure 3.5). "Contact" between a person's hand and an article is determined when there is any measurable overlap between bounding regions.

Up to now, we have assumed a rectangular bounding region, denoted as $\mathbf{z} = [x_l\ y_t\ x_r\ y_b]^T$. However, to more accurately determine the perimeter of the hand, we define the bounding region using 8 extremal points versus the two points for rectangular bounding boxes. Figure 4.7 illustrates the advantage of using extremal points over bounding rectangles to establish contact. In this example, the right hand bounding box (of the person on the right) over-

Point Name	Coord.	Point	Point
Topmost left	(r_1, c_1)	$r_1 = rmin$	$c_1 = \min\{c \mid (rmin, c) \in r_i\}$
Topmost right	(r_2, c_2)	$r_2 = rmin$	$c_2 = \max\{c \mid (rmin, c) \in r_i\}$
Rightmost top	(r_3, c_3)	$r_3 = \min\{r \mid (r, cmax) \in r_i\}$	$c_3 = cmax$
Rightmost bottom	(r_4, c_4)	$r_4 = \max\{r \mid (r, cmax) \in r_i\}$	$c_4 = cmax$
Bottommost right	(r_5, c_5)	$r_5 = rmax$	$c_5 = \max\{c \mid (rmax, c) \in r_i\}$
Bottommost left	(r_6, c_6)	$r_6 = rmax$	$c_6 = \min\{c \mid (rmax, c) \in r_i\}$
Leftmost bottom	(r_7, c_7)	$r_7 = \max\{r \mid (r, cmin) \in r_i\}$	$c_7 = cmin$
Leftmost top	(r_8, c_8)	$r_8 = \min\{r \mid (r, cmin) \in r_i\}$	$c_8 = cmin$

Table 4.1: The eight extremal points. [63]

laps two cards while the region enclosed by the extremal points voids the neighboring card. Polygonal regions defined by extremal points are particularly useful when the hands are held at more acute angles relative to the camera orientation. Table 4.1 shows how extremal points are computed.

Extremal Coord. Name	Coord. Representation
Topmost row	$rmin = \min\{r \mid (r, c) \in r_i\}$
Bottommost row	$rmax = \max\{r \mid (r, c) \in r_i\}$
Leftmost column	$cmin = \min\{c \mid (r, c) \in r_i\}$
Rightmost column	$cmax = \max\{c \mid (r, c) \in r_i\}$

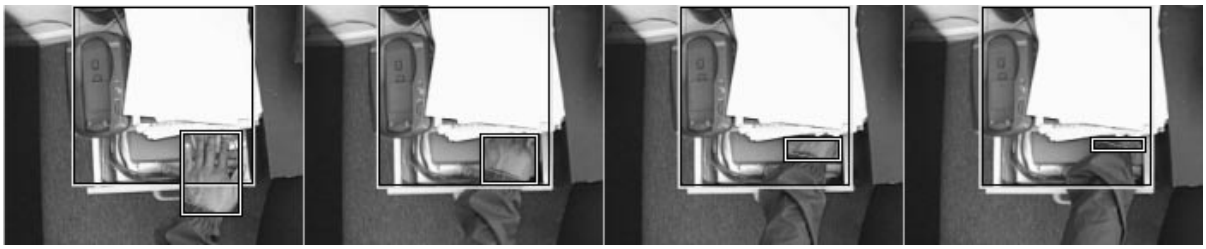


Figure 4.8: Portion of key frames illustrate blob tracking around article perimeter.

Our hand tracking algorithm also has a simple occlusion model that maintains awareness of hands that "disappear" under obstacles that lie closer to the image plane of the camera, as shown in Figure 4.8. The detection of an occlusion event starts once there has been some overlap between the hand perimeter and the article's bounding box (similar to a regular contact event). The extraction layer tracks all hand-colored regions around the article's bounding box by relaxing appearance-based parameters, i.e., $area_{min} \rightarrow 0$ and

$ratio_{min} \rightarrow 0$. If the hand region disappears after initial contact, it is assumed to lie under the occluding object until it re-emerges. If the hand region is fragmented into more than one blob, the larger blob is selected.

In some cases, an unanticipated occlusion will occur. For example, in the process of handling an article, the hand can temporarily become occluded. An illustration was presented earlier in Figure 3.8 where the right hand of a person slides under an object. To detect this type of occlusion, we first determine that there are no unmatched blobs with a hand score that satisfies a threshold, i.e., for a left hand with no corresponding region, we search all unlabeled regions such that $d(\mathbf{x}_{r_i}, \hat{\mathbf{x}}_{l,p}) p_{r_i \rightarrow \Psi} > \alpha_{min}$, where α_{min} is a preset threshold that represents the minimum allowable likelihood. We also check the estimated hand centroid $\hat{\mathbf{x}}$ to see that no hand may have moved outside of the camera view. After satisfying these two conditions for an occlusion, we set up a circular perimeter centered at $\hat{\mathbf{x}}$ with radius d_Q, the maximum hand displacement. We assume the hand has been occluded by some article within this region. The centroid estimate for the hand remains fixed at $\hat{\mathbf{x}}$ until the hand emerges within the perimeter.

4.2 Characterizing Actions using Hidden Markov Models

The hidden Markov Model (HMM) has emerged as a widespread statistical technique for recognizing sequential patterns in data. The majority of this section is dedicated to the description of the hidden Markov model. We begin with a brief look at the history of the HMM.

4.2.1 Background & Motivation

The fundamental theory of hidden Markov models was first published in a series of seminal publications by Baum and others in the late 1960s and early 1970s [12, 13, 14, 15]. Baker [8] and Jelinek et al. [79] first popularized the method for use with speech processing applications in the 1970s. The HMM is now a broadly used statistical technique for characterizing sequential processes ranging from the alignment of gene transcriptions for deoxyribonucleic acid (DNA) analysis [120] to handwriting recognition [66].

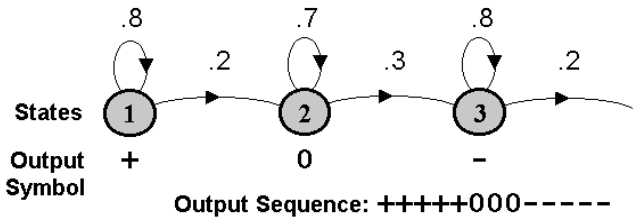


Figure 4.9: *Markov chain* - States with transition probabilities and deterministic output. State sequence can be determined uniquely from output [130].

Yamato et al. developed one of the first HMM-based gesture recognition systems in 1991 using binary mesh features to distinguish between 6 tennis strokes [148]. Since then, there has been a substantial following of researchers in the vision community who have adopted the HMM as their method of choice for characterizing human gestures and actions [21, 130]. Over the years, there have also been several modifications to the basic HMM, motivated by desires to expand its applicability to a broader range of problems. Examples include parametric HMMs [146], coupled HMMs [31], factorial HMMs [61], parallel HMMs [94], embedded HMMs [107], along with a host of hybrid HMM methods.

One reason for the popularity of the HMM has been its ability to accurately characterize data exhibiting sequential structure in the presence of noise and mild variation. State-based representations for motion, like the HMMs, are well-suited for actions and gestures, which have motion signatures with some natural ordering. We begin our discussion of the HMM by offering a complete definition.

4.2.2 Definition

HMMs can be described as a finite-state machine characterized by two stochastic processes: one process determines state transitions and is unobservable. The other produces the output observations for each state. The states can not be directly determined from the observations; hence, the states are *hidden*. For example, consider three states with transition probabilities and output responses, as depicted in Figure 4.9. Notice that each state generates a unique output, i.e., deterministic output, so it is possible to recover the state sequence directly from the output sequence. This is an example of a Markov chain. However, if each state

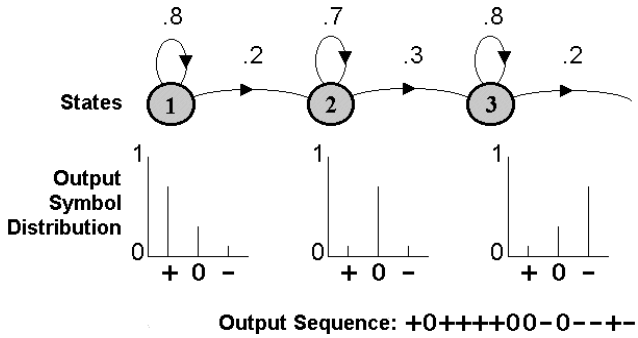


Figure 4.10: *HMM* - States with transition probabilities and probabilistic output. State sequence can not be determined uniquely from outputs.

can generate any one of the three outputs, then it is no longer possible to unambiguously recover the state sequence from the observations as Figure 4.10 illustrates; thus, the states are hidden. This example typifies an HMM. The beauty of the HMM lies in its ability to find the most likely sequence of states that may have produced a given sequence of observations.

We formally define the elements of a hidden Markov model with discrete observations using the following declarations:

- N, number of pre-defined, fixed states of the model.
- M, number of observation symbols in the symbol alphabet $\mathbf{V}$, such that $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M\}$; if the observations are continuous, then $\mathbf{V}$ is an infinite set.
- T, length of the observation sequence $\mathbf{O}$, which is given by $\mathbf{O} = (\mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_T)$.
- $\mathbf{s}$, the state space, which is partitioned into N states, such that $\mathbf{s} = \{s_1, s_2, \ldots, s_N\}$. The state of the model at time t is given by $q_t \in \mathbf{s},\ 1 \leq t \leq T$.
- $\mathbf{A}$, the state transition probability matrix, i.e., $\mathbf{A} = \{a_{ij}\}$ where

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i),\ 1 \leq i, j \leq N \tag{4.21}$$

  and q_t denotes the current state. The transition probabilities should satisfy the normal stochastic constraints:

$$a_{ij} \geq 0 \text{ and } \sum_{j=1}^{N} a_{ij} = 1,\ 1 \leq i, j \leq N.$$

- $\mathbf{B}$, the probability distribution matrix for the observation symbols, i.e., $\mathbf{B} = \{b_j(k)\}$ where,

$$b_j(k) = P(\mathbf{o}_t = \mathbf{v}_k | q_t = s_j),\ 1 \leq j \leq N,\ 1 \leq k \leq M \tag{4.22}$$

$\mathbf{o}_t$ is the observation at time t, and $\mathbf{v}_k$ denotes the k^{th} observation symbol in $\mathbf{V}$. Adherence to the stochastic constraints must again be satisfied:

$$b_j(k) \geq 0 \text{ and } \sum_{k=1}^{M} b_j(k) = 1.$$

If the observations are not discrete, then continuous probability density functions are employed, usually approximated by a weighted sum of M Gaussian distributions $\mathcal{N}$ such that

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{M} \omega_{jk} \mathcal{N}(\mathbf{o}_t, \mu_{jk}, \mathbf{C}_{jk}) \tag{4.23}$$

where ω_{jk} is the weighting coefficient for the k^{th} mixture in state s_j and $\mathcal{N}(\mathbf{o}_t, \mu_{jk}, \mathbf{C}_{jk})$ is a Gaussian pdf with mean vector μ_{jk} and covariance matrix $\mathbf{C}_{jk}$. The weighting coefficients should satisfy the constraints

$$\omega_{jk} \geq 0 \text{ and } \sum_{k=1}^{M} \omega_{jk} = 1.$$

- Π, the initial state distribution, $\Pi = \{\pi_i\}$, where

$$\pi_i = P(q_1 = s_i). \tag{4.24}$$

For convenience, an HMM with discrete probability distributions can be represented using the compact notation,

$$\lambda = (\mathbf{A}, \mathbf{B}, \Pi), \tag{4.25}$$

or

$$\lambda = (\mathbf{A}, \omega_{jk}, \mu_{jk}, \mathbf{C}_{jk}, \Pi), \tag{4.26}$$

if continuous probability distributions are adopted.

Generally, there are three key problems associated with the hidden Markov model: *evaluation*, *training*, and *decoding*. A detailed review of these procedures is provided in Appendex A. For an exhaustive discussion on HMMs, see [67, 118].

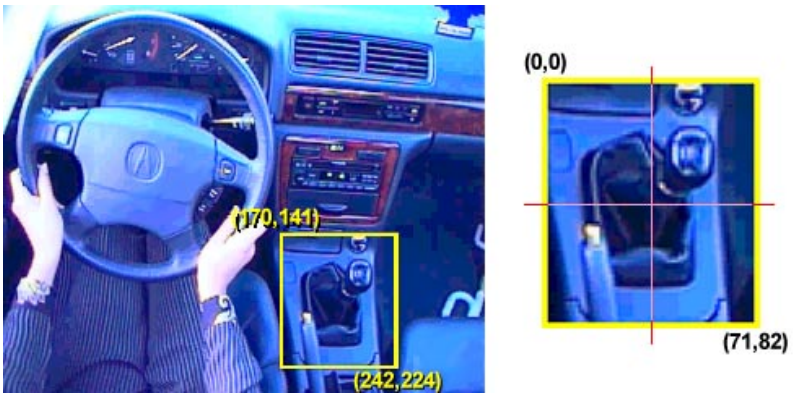


Figure 4.11: Gearbox uses translation matrix **T** to map bounding box coordinates back to origin. No rotation is required, i.e., $\theta = 0$.

4.3 HMM Implementation Issues

We continue with our discussion by examining issues related to their implementation and usage by describing the observation vectors used by our hidden Markov model. Our implementation of Viterbi parsing of continuous hand motion is also highlighted.

4.3.1 Observation Vectors

As we mentioned in Section 2.4.2, we are primarily concerned with finding the smallest and richest feature set that will allow us to properly characterize hand-based motion. Using color-based segmentation, we generate regions that represent the hands. While there are several metrics that can be used, such as pixel area, bounding box ratio, elongation, color (YUV value), velocity, eccentricity, angle of least inertia, etc., the only feature that we use is the two-dimensional hand centroid $(x, y$ position). For modeling the majority of hand actions, the centroid is generally sufficient, especially when the lexicon of actions is distinct and small, i.e., less than twenty. Moreover, because our framework was designed to reuse classes, which includes pre-trained actions, the general trajectory of motion is the only feature that is invariant to scene lighting conditions, scale variation, or the hand size, shape, or color of independent subjects.

To facilitate class reuse of action models, articles provide spatial context to normalize hand centroids. When modeling an action, we train the HMM using specific examples of

the interaction, usually conducted with a specific object. However, the HMM for this action is intended to be used by other instances of the class. For example, if we train the "flip forward" action using a specific *textbook*, we also want to use the same HMM to recognize the action when a different book is used. To use position information as observations of the model, we normalize hand movement relative to the article's bounding box. A second transformation is required to normalize hand movement relative to the dimensions of the article's generalized class model. Using these transformations, actions can be recognized no matter where an article is located in the scene. In other words, all hand positions are normalized with respect to the class model. Moreover, a single action model can be applied by articles with different physical dimensions.

When people interact with articles, the scene layer captures the raw coordinates of hand centroids. The article bounding box $\mathbf{z} = [x_l \ y_t \ x_r \ y_b]^T$ supplies the spatial context to translate and rotate raw coordinates for normalization. So, for a sequence of observation features $\mathbf{O} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$, we apply a 2×2 rotation matrix about θ, the angle of least inertia, i.e., angle of primary axis, to account for any rotation, and a translation vector $\mathbf{x}_{trans}$, which places the top-left corner of the bounding box at the origin. We also apply two scale factors to normalize the dimensions of the bounding box to match those of the generalized class model. For each object class, these affine transformations map hand movement to a single model (the GCM). The normalized observation feature $\widehat{\mathbf{O}}$ becomes

$$\widehat{\mathbf{O}} = \mathbf{S}[\mathbf{R}(\theta)\mathbf{O} + \mathbf{x}_{trans}] \tag{4.27}$$

where

$$\mathbf{S} = \begin{pmatrix} \frac{\mu_{width}}{x_r - x_l} & 0 \\ 0 & \frac{\mu_{height}}{y_b - y_t} \end{pmatrix}, \mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \mathbf{x}_{trans} = \begin{pmatrix} -x_l \\ -y_t \end{pmatrix},$$

μ_{width} and μ_{height} are the mean width and height dimensions of the GCM bounding box.

The states of an HMM represent characteristic stages of an action sequence. Because position is the only metric used to model the action, states of the HMM actually quantize positions of the sequence. In other words, if hand centroids are given as raw x and y coordinates, then the N states of the HMM correspond to N key Cartesian points in the image that hand centroids pass through during the action's execution. Each state is represented

as a Gaussian density with the normal mean and variance learned from training data. In our case, we elect to use single mixture, continuous HMMs for our models to properly characterize actions with a moderate amount of training data. We say more about setting up our continuous HMM in Section 4.3.4 below.

When only position information is used, it is interesting to ask "is the hidden portion of our HMM (the model's states) still hidden or does our HMM degenerate to a Markov chain?" Because of the direct relationship between observable positions in the image and hand locations, observations can be more intuitively related to the states that generated them; however, the states are, technically, still hidden.

4.3.2 Training Sets

In general, the more training data, the higher the recognition accuracy, provided that the supplied examples fully capture the nature of the action and its spatio-temporal variation. We assume that all actions are distinct and repeatable. Training sets for user independent action recognition systems tend to profit from observations constructed from a variety of subjects. However, the use of centroid-based observations allows models trained using a single user to be used effectively for person-indepedent testing. Unless specified otherwise, all actions are one-handed interactions.

Because of our unique, view-based approach, no existing action footage could be utilized for training. Therefore, all sequences used for training and testing were acquired expressly for our needs. The training set for each action roughly consists of 10-20 examples performed by the same person, and are manually segmented from video. More detail relevant to training and test conditions will be discussed in latter sections of Chapter 5, when some of our experimental results are highlighted.

4.3.3 HMM Topologies

Although there are elaborate techniques for defining optimal HMM topologies, we conducted empirical evaluations to select our topological structures. Using the same data for each experiment, we ran several trials with different model structures running Entropic's Hidden Markov Toolkit (HTK). The topology that maximized $P(\mathbf{O}|\lambda)$ over all of the heterogeneous

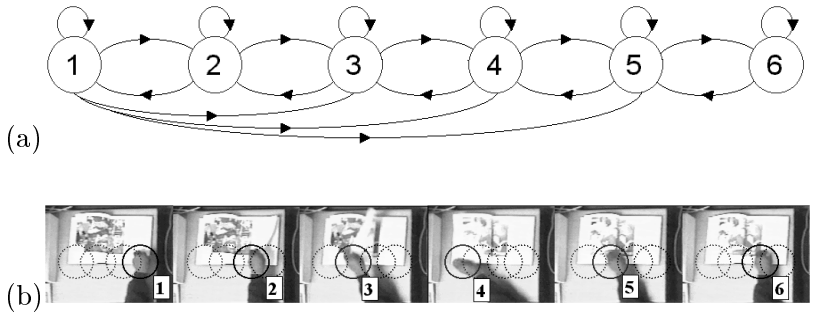


Figure 4.12: (a) Our empirically derived 6 state, semi-ergodic HMM with skip transitions; (b) image key frames that are representative of the 6 corresponding states of the "flip-forward" action.

action sequences was selected and is shown in Figure 4.12-a. Through our examinations, we found this structure to perform better than strictly left-to-right structures. This topology is used when training begins but is likely to be reduced to a more simple structure, depending on the actual training data, i.e., parameter estimation during training can use skip transitions to reduce the total number of effective states to less than six. Fully ergodic structures also performed worse, as they tend to require much more training data than we had available to properly characterize data. Figure 4.12-b shows that the six key frames that are representative of the six states used in our model topology during the "flip forward" action. Notice that states three and five as well as two and six appear to be located in the same area in the image. Likewise, they share similar probabilities of generating a hand location within the respective image regions. However, for this model, the probability of transition from s_4 to s_3 is pruned, i.e., $a_{43} \rightarrow 0$.

While the topology given in Figure 4.12-a works well as a general structure, there are many cases where a topology must be tuned for a particular action. Some of these models are shown in Figure 4.13. The top structure has a typical left-to-right topology for ordered motion paths, which is typical of many actions such as "grab book" (bookcase), "open door" (refrigerator), "take drink" (cup), and "pull parking brake" (automobile). The middle topology has a structure for supporting repetitive loops, such as the action "shake" (salt shaker). The bottom HMM offers a similar structure for redundant movements and is used to model actions like "wash dish" (sink).

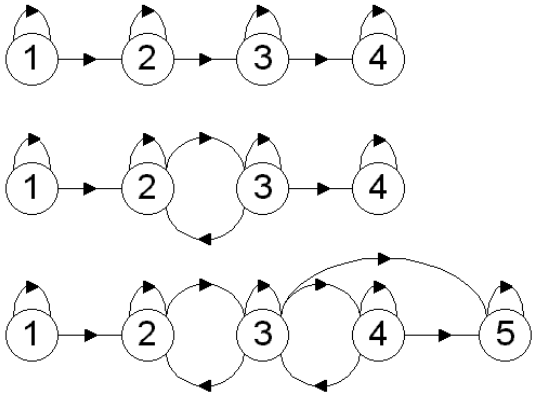


Figure 4.13: Other topological structures used to model actions.

4.3.4 Model Initialization

Once we have determined the number of states and a candidate topology for their arrangement, we can begin to model action sequences. The probability distribution for the continuous HMM can be expressed using a unimodal Gaussian given by

$$b_j(\mathbf{o}_t) = \frac{1}{\sqrt{(2\pi)^2|\mathbf{C}_j|}} e^{\frac{1}{2}(\mathbf{o}_t-\mu_j)^T\mathbf{C}_j^{-1}(\mathbf{o}_t-\mu_j)}, \tag{4.28}$$

where the mean μ_j and the covariance matrix $\mathbf{C}_j$ of state j are determined by

$$\mu_j = \frac{1}{T_j}\sum_{t=1}^{T_j}\mathbf{o}_t \quad \text{and} \tag{4.29}$$

$$\mathbf{C}_j = \frac{1}{T_j}\sum_{t=1}^{T_j}(\mathbf{o}_t-\mu_j)(\mathbf{o}_t-\mu_j)^T, \tag{4.30}$$

respectively, and T_j is the number of samples associated with state j. Usually, initial estimates for μ and $\mathbf{C}$ can be obtained by simply dividing the observation evidence evenly among the states, then using Equations 4.29 and 4.30. Initial transition probabilities for $\mathbf{A}$ are normally taken to be uniform. However, in many cases, we achieved better results from exploiting using hand-marked data from training examples instead of resorting to flat densities. Unfortunately, Baum-Welch re-estimation only provides a local maximum for the likelihood function. Whenever possible, initial estimates are also empirically selected so that the local maximum is likely to be the global maximum.

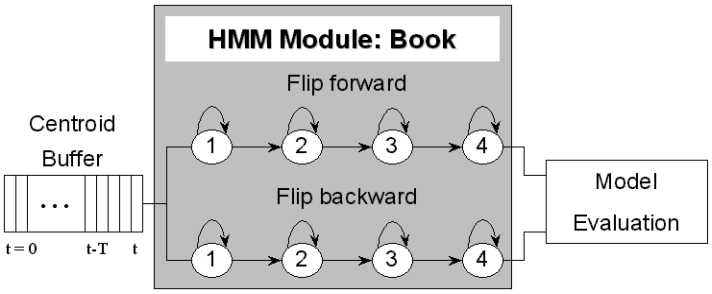


Figure 4.14: Buffer containing hand positions feeds observation vectors of variable length to class-related HMMs, which are evaluated in parallel.

The reestimation formulas for our Gaussian model are given by

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} \gamma_t(j) \cdot \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{4.31}$$

$$\bar{\mathbf{C}}_j = \frac{\sum_{t=1}^{T} \gamma_t(j) \cdot (\mathbf{o}_t - \mu_j)(\mathbf{o}_t - \mu_j)^T}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{4.32}$$

The reestimation equation for a_{ij} is unchanged from Equation A.28.

4.3.5 Viterbi Parsing of Observations

A buffer that records the centroid of each hand in each frame is maintained during tracking. After hand contact with an article is established, positions from the buffer are fed to an HMM Viterbi recognition module. The module contains a bank of all the action HMMs associated with the article class, which are evaluated in parallel, as shown in Figure 4.14.

Each HMM evaluates the likelihood of a particular action using the Viterbi algorithm to parse the observation sequence over a variable length window. As each new observation is added, the oldest sample at the other edge of the window is removed. The Viterbi trellis search is then conducted *backwards* over the input observations to find the maximum probability parse. The window length T is based on the average sequence length used to train the particular model. In practice, however, this length is adjusted over a small range

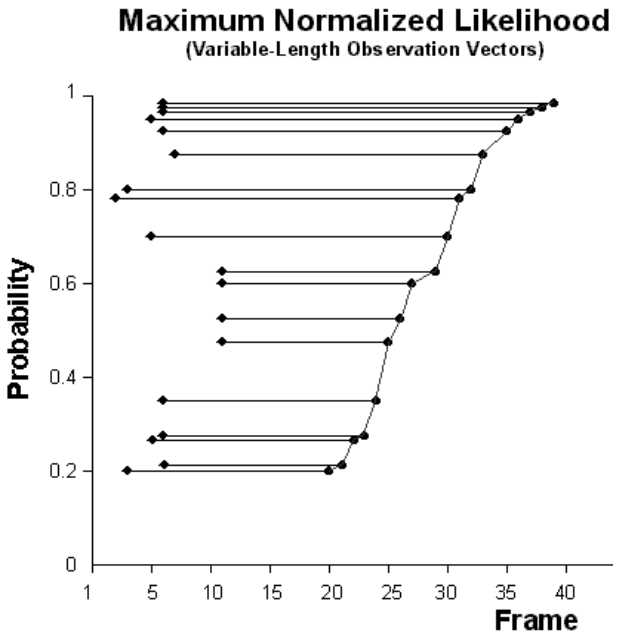


Figure 4.15: Maximum normalized probability per frame (*smooth curve*) with the corresponding length T of the observation vector.

in the neighborhood of length T. As the window's length increases, the sequence likelihood decreases due to the multiplication of probabilities less than one. We normalize by averaging over the length of the sequence. By executing all relevant HMMs in parallel, we can provide the maximum normalized probability and the corresponding optimal sequence length at each time step. Knowing the sequence length helps to avoid timing conflicts so that the same action isn't reported multiple times (at each frame-wise evaluation). Determining the start-finish boundaries for an action helps to disambiguate potentially confusing actions. For example, if only position information is available repeated occurrences of the right-left-right motion of the "flip forward" action and the left-right-left motion of "flip backward" are easily confused. However, since we can determine when the motion starts (from the left or from the right), we can decide between the two. Figure 4.15 illustrates the window length associated with the maximum normalized probability.

We use manually set thresholds to discard low scores, then select the model that maximizes $P(\mathbf{O}|\lambda)$. Figure 4.16 illustrates the output of the "flip forward" HMM in for the `book` class.

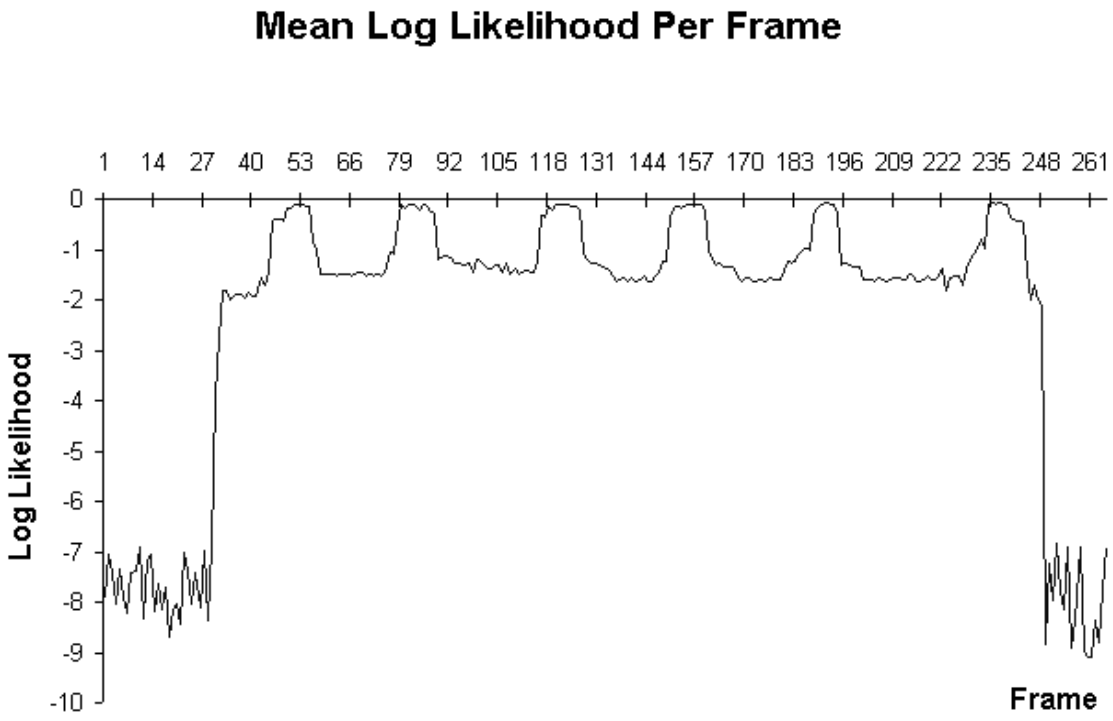


Figure 4.16: *"Flip Forward" HMM output*: Mean log likelihood per frame (action is repeated).

CHAPTER 5

Evaluating Evidence for Recognition Tasks

Poole calls evidence "the knowledge directly acquired through interacting with the world" [116]. In Chapter 3, we introduced ObjectSpaces, a framework for managing and organizing evidence extracted from video. We also discussed processes for detecting contact and analyzing hand interactions with articles in Chapter 4. In this chapter, we present techniques to analyze evidence or information collected by our framework to recognize single-tasked activities as well as to classify unknown objects.

To address these problems, we extend appearance-based representations by searching for low-level information that indicates an object's image appearance as well as how people appear to interact with the object over time. Bayesian classification techniques are employed to identify and weigh contextual patterns in evidence that can be used to recognize single-tasked activities or to classify unknown objects. Our approach relates hand motion and object context to achieve:

- object recognition using action context, and
- action recognition using object context.

Contributions in this chapter include:

- an extension to appearance-based representations for recognition
  - object classification using action context, i.e., object classification based on how people appear to interact with the object, and
  - action recognition using object context, i.e., action recognition based on an object's appearance

- adaptive Bayesian classification techniques to identify and weigh dynamic patterns in evidence over time.

5.1 Extracting Evidence

We concentrate on collecting heterogeneous, low-level information that can be recovered reliably and can provide a basis for classification. We look for complementary information so that decisions regarding recognition are more robust against evidence that may be weak in some areas or stronger in others. By sampling the state of the environment over time, i.e., capturing object features, interactions, etc., we accumulate dynamic changes in object and action context, which provide much richer information for recognition than static observations. There are three categories of evidence that we will use to establish maximum likelihood classification:

Definition 5.1 Image-based evidence *consists of appearance-based features that quantify an object's size, shape, edges, similarity to known image templates, etc.*

Definition 5.2 Object-based evidence *captures the likelihood of sequential object-to-object contact between articles classes.*

Definition 5.3 Action-based evidence *deals with the recognition of known actions.*

All three categories are weighed and assessed for object classification while activity recognition primarily relies on action-based evidence (since the object is assumed to be known). Only image- and action-based evidence are used to detect new articles.

5.1.1 Image-based Evidence

We consider a small set of appearance-based features that are generally sufficient enough for object classification when considering a relatively small set of object classes [65]. We detect new objects and people in the scene using conventional background segmentation techniques. During initialization, a snapshot of the background is taken to capture its initial state, $I(x, y, 0)$, then known articles are labeled. At run-time, changes in the appearance

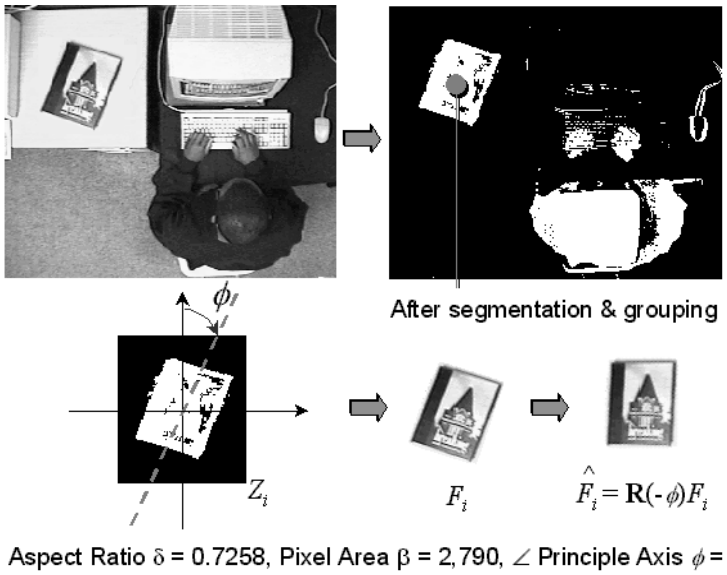


Figure 5.1: *Image-based Evidence:* Background segmentation reveals newly introduced objects that can be analyzed to recover appearance-based features, such as aspect ratio, pixel area, orientation, the image template, and bounding region.

of the background can be caused by movement from hands or existing articles as well as by the introduction of new, unknown articles. In any case, we segment these differences by subtracting the current frame $I(x, y, t)$ from $I(x, y, 0)$, then apply an empirically determined threshold α, i.e.,

$$\tilde{B}(x, y, t) = \begin{cases} 1 & \text{if } |I_Y(x, y, 0) - I_Y(x, y, t)| \geq \alpha \\ 0 & \text{otherwise} \end{cases} . \tag{5.1}$$

Here I_Y represents only the *intensity component*[1] of YUV image I while $\tilde{B}(x, y, t)$ is the binary image produced by the segmentation. Since we avoid overlapping or stacking articles on top of each other, i.e., intentional occlusion of an article by another, there is little need for a background adaptation process to update $I_Y(x, y, 0)$. We invoke Connected Component Labeling (CCL) as described earlier (in Equations 4.3) to group and label binary regions.

Our correspondence algorithm analyzes and tracks each blob over consecutive frames to determine if the region represents a new, unknown object or is part of a known article or

[1]Implementing image subtraction using only 1 image channel versus 3 shortens processing time and reduces memory storage by 66%.

hand part. During each frame, we check for overlap between the known bounding boxes of articles and hands and the bounding box of each blob. This comparison reveals changes in the background for which we can not account. To determine if the blob represents a new object or an image artifact, such as a shadow, we track it over consecutive frames while a region analysis procedure examines motion and pixel area. Those unknown regions with inconsistent size and position are pruned from further consideration[2]. The remaining n regions that appear to be unlabeled articles are examined to extract evidence that can be used for recognition. The foreground subimage $F_i(x, y, t)$ of unknown article Z_i is given by

$$F_i(\mathbf{x}, t) = I(\mathbf{x}, t), \ \forall \ \mathbf{x} \in Z_i, \ 1 \leq i \leq n \tag{5.2}$$

where $\mathbf{x} = [x \ y]^T$. To determine Z_i's orientation with respect to the angle of the principle axis, we calculate ϕ given by

$$\phi = \frac{1}{2} \arctan \left(\frac{2 \sum\limits_{(x,y) \in Z_i} xy}{\sum\limits_{Z_i} x^2 - \sum\limits_{Z_i} y^2} \right). \tag{5.3}$$

We apply a rotation matrix $\mathbf{R}$ to obtain the rotationally normalized foreground subimage $\widehat{F_i}$, i.e., $\widehat{F_i} = \mathbf{R}(-\phi)F_i$ (illustrated in Figure 5.1). Using $\widehat{F_i}$, the unknown can be compared to stored image templates from our class database. We also recover other features[3] of Z_i, including pixel area β_i, aspect ratio δ_i, and perimeter edge count ϵ_i. A bounding box $\mathbf{z}_i$ is constructed around Z_i, which helps to detect future hand overlap and to recover actions performed around this region. To minimize noisy measurements, these parameters are averaged over several frames and used to formally create a new `Article` object (though still unknown).

To leverage prior knowledge, we first attempt to identify each unknown Z_i by comparing $\widehat{F_i}$ to each stored subimage template S_j that shares a similarly-sized bounding box[4], such as those shown in Figure 3.3. The *mean square error* (MSE) η, used to quantify

[2]We assume movement or shape inconsistency to be a trademark of an image artifact.

[3]Note that these are the same appearance features used to define any `Article` class.

[4]Recall that we assume scale variation is negligible since the distance between the action plane and camera is generally fixed.

matching of an $N_1 \times N_2$ pixel template, is defined as

$$\eta(i,j) = \frac{1}{N_1 N_2} \sum_{\mathbf{x} \in S_j} [S_j(\mathbf{x}) - \widehat{F_i}(\mathbf{x})]^2. \tag{5.4}$$

In summary, the parameters $\beta_i, \delta_i, \epsilon_i$, and η form the vector Υ_i, which represents image-based evidence. We also have created a new, unlabeled `Article` object, which we can use to manage context, properties, and interactive events associated with Z_i.

5.1.2 Object-based Evidence

Sequential contact is the order in which one touches different objects. For example, in a car, the right hand touches the stick shift while changing gears, *then* grabs the steering wheel. The statistical history of sequential object-to-object contact can be helpful for revealing relationships between article classes. One of the objectives of object-based evidence is to characterize probabilistic relationships between article classes, i.e.,

$$P(\textit{currently} \text{ touching steering wheel} | \textit{previously} \text{ touched stick shift}).$$

These 1^{st}-order relationships can be exploited when encountering contact between a known and an unknown article.

Before the scene is initialized, we select the domain $\mathcal{C}$ that best describes the environment where interactions are to take place. We know *a priori* which generalized class models can appear in $\mathcal{C}$. For example, in the office domain, we anticipate several article types, including a keyboard, book, phone, table, etc. Let $n_{\mathcal{C}}$ represent the number of unique GCMs in domain $\mathcal{C}$ and $\mathbf{M}_{\mathcal{C}}$ denote the subset of class models relevant to domain $\mathcal{C}$, e.g., $\mathbf{M}_{\mathcal{C}} \in \mathbf{M}$.

While a person is interacting with objects in the environment, the scene layer observes sequential contact between every two articles a_i and a_j, storing the number of transitions from class type to class type in a $n_{\mathcal{C}} \times n_{\mathcal{C}}$ matrix $\mathbf{K} = \{k_{ij}\}$. Here k_{ij} denotes contact order starting from an article with the class model M_i and terminating with an article of class M_j. Instead of blindly considering any pair of object-to-object transitions, we look for rest states in hand motion, i.e., $d\mathbf{x} = 0$, that might indicate a break in deliberate hand-to-object interaction. We initialize matrix $\mathbf{K}$ to zero, i.e., $k_{ij} = 0$. During training exercises,

	cup	dr lock	pkg brake	radio	stick	str wheel	temp ctrl	win handle
cup	0.081	0.000	0.041	0.041	0.081	0.715	0.041	0.000
door lock	0.000	0.000	0.000	0.000	0.000	0.900	0.000	0.100
parkg.brake	0.050	0.000	0.050	0.100	0.033	0.400	0.167	0.000
radio	0.040	0.000	0.060	0.175	0.137	0.545	0.043	0.000
gear stick	0.035	0.000	0.105	0.058	0.000	0.745	0.057	0.000
steer wheel	0.062	0.038	0.030	0.094	0.517	0.097	0.088	0.074
temp ctrl	0.051	0.000	0.000	0.117	0.072	0.676	0.084	0.000
wind handle	0.000	0.100	0.000	0.000	0.000	0.900	0.000	0.000

Table 5.1: Object-to-object hand transition matrix **K** for automobile domain.

we populate the scene with articles from all $n_{\mathcal{C}}$ class models so that we can construct a well-balanced distribution of transitions. As hands are tracked, we update k_{ij} by counting the transitions when hand contact moves between articles. To determine the probability that the j^{th} article is handled given that the i^{th} article was previously handled, we solve for $P(a_i|a_j)$ such that

$$P(a_i|a_j) = \begin{cases} \dfrac{k_{ij}}{n_i} & \text{if } n_i > 0 \\ 0 & \text{otherwise} \end{cases}, \text{ where } n_i = \sum_{j=1}^{n_{\mathcal{C}}} k_{ij} \tag{5.5}$$

Counting transitions proves to be advantageous because it makes it convenient to normalize transition probabilities based on the classes represented in the scene. This is easily accomplished by excluding terms of missing article classes in the summation in Equation 5.5 or by adding others as new object classes are introduced to the scene. Table 5.1 shows matrix **K** for the automobile domain after initial training. In summary, contact transition matrix **K** serves as a measure of object-based evidence by inferring the class of an unknown object through its relationship with known articles.

5.1.3 Action-based Evidence

Action-based evidence plays an important role in both object classification and activity recognition. When a person interacts with an unknown object, we search for hand actions to infer the object's class, e.g., object recognition from action context. Naturally, if we already know what article a person is interacting with, we limit the range of possible actions to those that we directly associate with that object's class, e.g., action recognition from object context. The background subtraction method used to extract image-based evidence

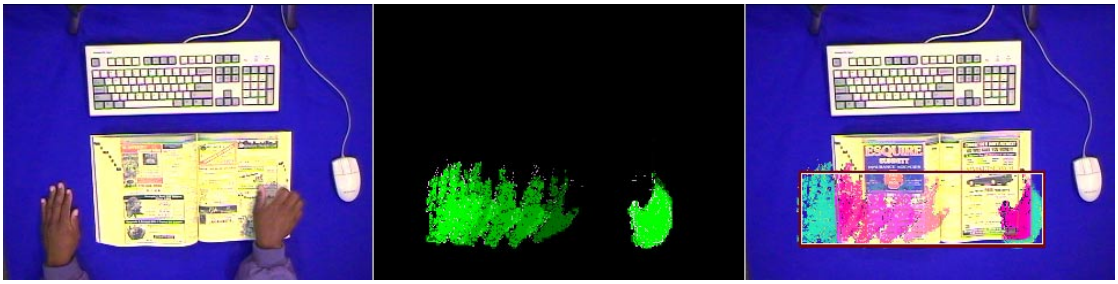


Figure 5.2: The area spanned by the hands during the "flip forward" action helps to define the bounding box of a book that is initially part of the background.

is the best way to detect newly introduced objects. When this method fails, we show that recognizing actions can also provide a powerful method for detecting unknown objects, especially those that are part of the background.

In cases where the article's class is unknown, all HMMs that can be associated with the known domain $\mathcal{C}$ are placed in parallel as mentioned in Section 4.3.5. Let the entire set of HMMs used by all GCMs appearing in domain $\mathcal{C}$ form action model space $\mathbf{\Gamma}_{\mathcal{C}}$ of dimension v, such that $\mathbf{\Gamma}_{\mathcal{C}} = \{\lambda_{ff}, \lambda_{fb}, \lambda_{wrt}, ...\} = \{\lambda_1, \lambda_2, ..., \lambda_v\}$. We determine the most likely sequence of states given a model using the Viterbi Algorithm [67]. We maximize $P(\mathbf{O}|\lambda)$ over every action model in $\mathbf{\Gamma}_{\mathcal{C}}$, i.e.,

$$p_\gamma = \max_{\mathbf{\Gamma}_{\mathcal{C}}} \{P(\mathbf{O}, \mathbf{q}|\lambda)\}, \tag{5.6}$$

which we refer to as a measure of action-based evidence. To model interactions with known articles that have associated actions, the same process is used, except that only the set of HMMs associated with the object are assembled in parallel to operate on incoming observations.

To detect unknown articles that may be fully occluded or merged with the background, we conduct HMM analysis when hands are in undesignated areas (considering all action models in $\mathbf{\Gamma}_{\mathcal{C}}$). We examine where hand-based actions take place, relative to the location of known articles. If actions are repeatedly recognized outside of known bounding regions, we infer that an unknown article must exist in the undesignated area. The bounding box for the article $\mathbf{z}_i = [x_l \; y_t \; x_r \; y_b]^T$ is initially defined by the space between existing zones and the area spanned by the hand's bounding box during an action (upper-left and lower-right

corners of hand motion). The bounding coordinates of the unknown article are modified as we track the extremities of the hand's bounding region during each recognized action, i.e.,

$$\mathbf{z}_i = \begin{pmatrix} x_l \\ y_t \\ x_r \\ y_b \end{pmatrix} = \begin{pmatrix} \min_{t=1...T}\{x_l(t), x_l\} \\ \min_{t=1...T}\{y_t(t), y_t\} \\ \max_{t=1...T}\{x_r(t), x_r\} \\ \max_{t=1...T}\{y_b(t), y_b\} \end{pmatrix}, \tag{5.7}$$

where $x_l(t), y_t(t), x_r(t)$, and $y_b(t)$ are bounding coordinates in frame t of a T-length sequence.

5.2 Evaluating Evidence

In the following sections, we describe Bayesian approaches to weigh and assess image-, object-, and action-based evidence for maximum likelihood classification. We begin with a brief motivation for using Bayesian methods before we unveal our strategies for recognizing single-tasked activities and classifying unknown objects.

5.2.1 Bayesian Decision Theory

In his 1763 essay entitled, "An essay towards solving a problem in the doctrine of chances," the Reverend Thomas Bayes introduced a mathematical relationship that remains one of the methods of choice for handling uncertainty [16]:

$$P(model|data) = \frac{P(data|model) \cdot P(model)}{P(data)}.$$

Bayes' theorem states that the posterior is proportional to the likelihood times the prior. It weighs the strength of belief in a model, or hypothesis, against prior knowledge and observed evidence. In addition, it provides attractive features, including: (1) its ability to pool evidence from various sources while making a hypothesis, and (2) its amenability to recursive or incremental calculations, especially when evidence is accumulated over time [3]. These features motivate our application of Bayesian classifiers to summarize activities and resolve unknown objects.

Bayesian methods are widely used to provide a formal means to reason about partial beliefs under conditions of uncertainty [113]. In particular, there is a substantial amount of

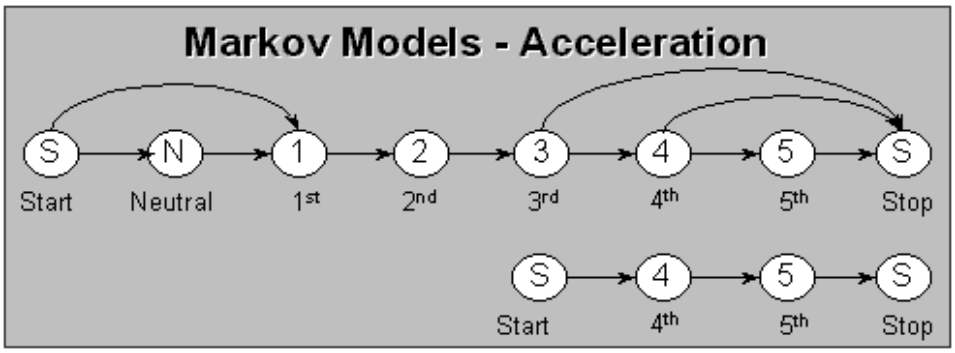


Figure 5.3: To accommodate various scenarios, multiple Markov models are used to represent some high-level activities, such as acceleration.

existing work in the literature that exploits Bayesian approaches for solving action recognition problems. Examples include Nagel [103] and Neumann [108], who conducted several pioneering studies that describe behavior from image sequences. More recently, Bayesian belief networks surface as a viable method for detecting and describing interactions. The framework proposed by Buxton et al. uses Bayesian Networks to perform surveillance and evaluate evidence in well understood scenes [35]. In their approach, static and dynamic Bayesian networks provide robust tracking and segmentation. Using rules about the scene and human activities as constraints, these networks can be use to analyze user behaviors. Bayesian inference methods are also useful for making conclusions based on the presentation of adequate evidence. Yi and Chelberg use Bayesian belief networks for selecting probable objects based on discriminating features and domain-specific knowledge [150]. In another example, Brand combines low-level image features of hand interactions with objects along with an action grammar to interpret video of manipulation tasks such as disassembly [30]. In this work, causal constraints are used to detect meaningful changes in the integrity and movement of foreground-segmented blobs. A model of manipulation is used to disambiguate, parse, and produce a script of actions taking place in the scene.

5.2.2 Evaluating Evidence for Single-Tasked Activities

High-level, single-tasked activities are characterized by a sequence of lower-level hand actions or events that we represent using a Markov chain. For example, in the automobile

domain, the high-level activity "accelerating" is recognized when an ordered sequence of lower-level actions, such as "Neutral-to-1$^{\text{st}}$," "1$^{\text{st}}$-to-2$^{\text{nd}}$," and "2$^{\text{nd}}$-to-3$^{\text{rd}}$." In some cases, we create multiple Markov models to represent the same activity as illustrated in Figure 5.3. The single-tasked activity model Λ is defined similarly to an HMM:

- N, number of fixed states of the model.
- M, number of observation symbols in the symbol alphabet $\mathbf{v}$, such that $\mathbf{v} = \{v_1, v_2, \ldots, v_M\}$; Symbol alphabet $\mathbf{v}$ is composed of all possible hand action HMMs and contact events in domain $\mathcal{C}$, i.e., $\mathbf{v}_\mathcal{C} = \{\lambda_1, \lambda_2, \ldots, \varepsilon_1, \varepsilon_2, \ldots\}$.
- T, length of the observation sequence $\mathbf{o}$, which is given by $\mathbf{o} = \{o_1\ o_2 \ldots o_T\}$.
- $\mathbf{s}$, the state space, which is partitioned into N states, such that $\mathbf{s} = \{s_1, s_2, \ldots, s_N\}$. The state of the model at time t is given by $q_t \in \mathbf{s},\ 1 \leq t \leq T$.
- $\mathbf{A}$, the state transition probability matrix, i.e., $\mathbf{A} = \{a_{ij}\}$ where

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i),\ 1 \leq i, j \leq N \tag{5.8}$$

  and q_t denotes the current state. The transition probabilities should satisfy the normal stochastic constraints:

$$a_{ij} \geq 0 \text{ and } \sum_{j=1}^{N} a_{ij} = 1,\ 1 \leq i, j \leq N.$$

- Each state s_j generates a unique output symbol v_k, i.e., $P(v_k | s_j) = 1,\ \forall\ v_k \in \mathbf{v}_\mathcal{C},\ 1 \leq j \leq N,\ 1 \leq k \leq M$.
- Π, the initial state distribution, $\Pi = \{\pi_i\}$, where

$$\pi_i = P(q_1 = s_i). \tag{5.9}$$

The Markov chain can be represented using the compact notation,

$$\Lambda = (\mathbf{A}, \mathbf{v}_\mathcal{C}, \Pi). \tag{5.10}$$

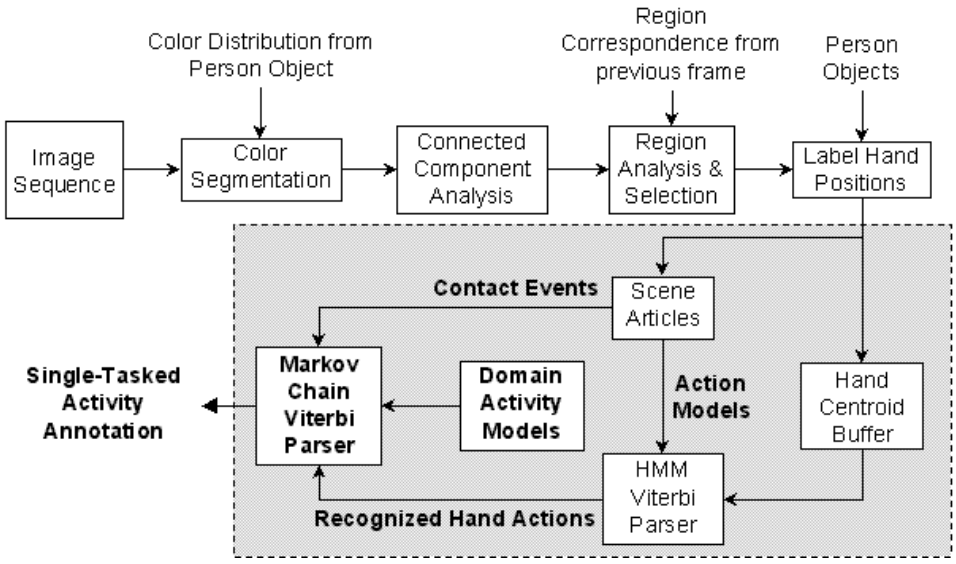


Figure 5.4: Contact events and low-level hand actions feed into a bank of Markov models for recognizing single-tasked activities.

Since state output is deterministic in a Markov chain, only state transitions are probabilistic. State transition parameters can generally be specified by hand for simple models[5], but benefit from training data for more complex Markov models.

To recognize single-tasked activities, we construct a second tier of Viterbi parsers that accept, as input, the sequence of low-level interactive hand events generated by hand motion analysis in the extraction layer. Recall that the extraction layer attempts to recover actions associated with object context by passing hand centroids through a bank of Viterbi parsers for each action HMM λ. If the article class has no associated action models or hand motion can not be characterized, a "contact" event ε is reported. This is illustrated in Figure 5.4.

For a given domain $\mathcal{C}$, we consider the set $\mathbf{\Omega}_{\mathcal{C}}$, which contains every single-tasked activity model Λ, such that $\mathbf{\Omega}_{\mathcal{C}} = \{\Lambda_1, \Lambda_2, \ldots\}$. In order to compute the likelihood of an activity given a sequence of observations, we solve the relation

$$\hat{\Lambda} = \max_{\mathbf{\Omega}_{\mathcal{C}}} \{P(\mathbf{q}|\Lambda)\}, \tag{5.11}$$

[5]Simple models typically have left-to-right topologies with no return-state loops or skip states.

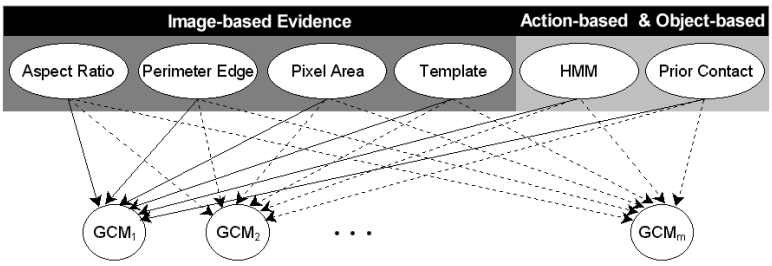


Figure 5.5: Belief network corresponding to a naïve Bayesian classifier for selecting the most likely generalized class model (GCM).

where $\mathbf{q} = \{v_1 \; v_2 \ldots v_T\}$ represents a sequence of actions and events using the Viterbi algorithm.

5.2.3 Evaluating evidence for classifying unknowns

For recognizing unknown objects, we construct a naïve Bayesian classifier to select the most likely Generalized Class Model (GCM) associated with unknown Z_i (Figure 5.5). The classifier assesses each component of image-, object-, and action-based evidence independently. We allow M_k to represent the k^{th} GCM in $\mathbf{M}$. To measure appearance similarity, we express the likelihood of Z_i's image-based evidence Υ_i given a particular class model M_k as

$$\begin{aligned} P(\Upsilon_i|M_k) &= \frac{P(\beta_i, \delta_i, \epsilon_i, M_k)}{P(M_k)} \\ &= \frac{P(\beta_i|M_k)P(\delta_i|M_k)P(\epsilon_i|M_k)}{P(M_k)} \\ &= e^{-\frac{1}{2}\left(\frac{(\beta_i - \mu_{\beta,k})^2}{\sigma^2_{\beta,k}} + \frac{(\delta_i - \mu_{\delta,k})^2}{\sigma^2_{\delta,k}} + \frac{(\epsilon_i - \mu_{\epsilon,k})^2}{\sigma^2_{\epsilon,k}}\right)}, \\ &= e_1(i,k) \end{aligned} \tag{5.12}$$

where $\mu_{\beta,k}$, $\mu_{\delta,k}$, and $\mu_{\epsilon,k}$ are mean values and $\sigma^2_{\beta,k}$, $\sigma^2_{\delta,k}$, and $\sigma^2_{\epsilon,k}$ are variances determined from example image templates for each components of appearance, β_i, δ_i, and ϵ_i, respectively[6]. In Equation 5.12, we use an unnormalized Gaussian so that we do not dilute the likelihood score.

We also want to consider the contribution from any of the stored image templates of previously seen articles. The reader is reminded of the possibility that template S_j can

[6]Recall that Equation 3.8 is used to estimate the Gaussian parameters for each component.

belong to more than one class model M_k. We consider the likelihood of the unknown given the joint likelihood that template S_j is a member of class model M_k, such that

$$\begin{aligned} P(Z_i|S_j, M_k) &= \max_j \left\{ \frac{P(Z_i, S_j, M_k)}{P(S_j, M_k)} \right\} \\ &= \max_j \left\{ \frac{P(Z_i, S_j)P(M_k|S_j)}{P(S_j|M_k)P(M_k)} \right\} \\ &= \max_j \left\{ \frac{P(Z_i|S_j)P(M_k|S_j)P(S_j)}{P(S_j|M_k)P(M_k)} \right\} \\ &= e_2(i,k) \end{aligned} \tag{5.13}$$

We determine the priors $P(S_j)$ and $P(M_k)$ along with the posteriors $P(M_k|S_j)$ and $P(S_j|M_k)$ during initialization of our model database $\mathbf{M}$. Note that $P(S_j)$ reflects the frequency with which S_j occurs over the sample space of all models M_k, i.e.,

$$P(S_j) = \frac{\text{no. occurrences of } S_j \ \forall \ \mathbf{M}}{\sum_{\forall\ k} \text{no. templates in } M_k}. \tag{5.14}$$

Recall that we calculate the mean square error in Equation 5.4 to get a measure of similarity between two templates. If cameras are mounted roughly at the same distance from the image plane where most activities occur, i.e., same distance from table tops, scale variation within templates is insignificant. Typically, this distance is large enough to keep perspective projection distortion relatively small. Hence, the location of the template in the scene is of little consequence. To use the MSE to determine the likelihood that Z_i appears as S_j, we approximate $P(Z_i|S_j)$ by

$$P(Z_i|S_j) \approx e^{\frac{-\eta}{\tau}}, \tag{5.15}$$

where τ is discovered empirically. We elect to use this exponential distribution because it is well-behaved on the range [0:1] for appropriate τ (monotonically decreasing). Additionally, since the MSE is a relatively noisy measurement, we can design τ to be more tolerable of effects caused by poor alignment, reflections, lighting conditions, and small occlusions.

Next, we collect object-based evidence that may allow us to determine the class of the unknown by identifying how people behave while using it with known objects. We are able to assess this evidence by considering the interactive relationship between article pairs. Recall matrix $\mathbf{K}$, which contains the number of hand transitions between respective article classes. During interactivity, we maintain a count of the transitions between Z_i and other

articles in the scene using vector $\mathbf{g}$ of dimension $n_{\mathcal{C}}$, i.e., $\mathbf{g}^T = [g_1\ g_2\ \ldots g_{n_{\mathcal{C}}}]$ (where g_j is the count of transitions from Z_i to some article from model class M_j). To compare Z_i's transition behavior (described by $\mathbf{g}_i$) to behavior collected from training exercises (described by $\mathbf{K}$), we subtract $\mathbf{g}_i^T$ from each row in $\mathbf{K}$, denoting the resulting $n_{\mathcal{C}} \times n_{\mathcal{C}}$ matrix as $\widehat{\mathbf{K}}$. The highest correlation between $\mathbf{g}_i$ and $\mathbf{M}$ will produce the minimum row sum in $\widehat{\mathbf{K}}$. Each element $\hat{k}_{uv}$ of $\widehat{\mathbf{K}}$ is defined as

$$\hat{k}_{uv} = \begin{cases} 2|k_{uv} - g_v| & \text{if } k_{uv} = 0 \\ |k_{uv} - g_v| & \text{otherwise} \end{cases} . \tag{5.16}$$

We assume that $\mathbf{K}$ is well-developed from extensive training footage. We also assume that any zero probabilities in $\mathbf{K}$ reflect highly unlikely transitions, therefore, we amplify those differences. In each row u of $\widehat{\mathbf{K}}$, we compute $\zeta(u)$ to describe the total matrix sum minus the sum of differences between the likelihoods of the unknown and each class model, i.e.,

$$\zeta(u) = \chi - \sum_{v=1}^{n_{\mathcal{C}}} \hat{k}_{uv},\ 1 \leq u \leq n_{\mathcal{C}}, \tag{5.17}$$

where the total matrix sum χ is given by

$$\chi = \sum_{u=1}^{n_{\mathcal{C}}} \sum_{v=1}^{n_{\mathcal{C}}} \hat{k}_{uv}. \tag{5.18}$$

Finally, we can determine the probability of $\mathbf{g}_i$ given model M_k by

$$\begin{aligned} P(\mathbf{g}_i|M_k) &= \frac{\zeta(k)}{\sum_{u=1}^{n_{\mathcal{C}}} \zeta(u)} \\ &= e_3(i,k) \end{aligned} . \tag{5.19}$$

To assess the contribution of action-based evidence, we examine hand position $\mathbf{O}$ for patterns that map to a known action model λ, which we also associate with class models. To calculate the probability of a hand interactions with Z_i given both an action model and a generalized class model, we solve the relation

$$\begin{aligned} P(\mathbf{O}|M_k) &= \max_{\mathbf{\Gamma}_{\mathcal{C}}} \{P(\mathbf{O}|\lambda_\gamma) P(\lambda_\gamma|M_k)\} \\ &= e_4(i,k) \end{aligned} . \tag{5.20}$$

Recall that $\mathbf{\Gamma}_{\mathcal{C}}$ contains all action models in domain $\mathcal{C}$.

We compose four-dimensional vector $\mathbf{e}^T(i,k)$ from all image-, object-, and action-based evidence components associated with unknown Z_i, such that

$$\begin{aligned} \mathbf{e}(i,k) &= [P(\Upsilon_i|M_k)\ P(Z_i|S_j,M_k)\ P(\mathbf{g}_i|M_k)\ P(\mathbf{O}_i|M_k)]^T \\ &= [e_1(i,k)\ e_2(i,k)\ e_3(i,k)\ e_4(i,k)]^T \end{aligned} . \tag{5.21}$$

Naturally, the quantity and quality of this evidence changes over time in response to interactions taking place. Likewise, our model of belief should be able to capture the dynamic contribution of the different bodies of evidence as well as the overall level of uncertainty in our probabilistic model. The dynamic effects of changing context can be ameliorated by accumulating evidence over time. For compactness in our notion, we drop the arguments i and k when convenient, but include t to signify the time when the evidence is sampled[7]. We determine the sample average of the evidence $\tilde{\mathbf{e}}$ over a sliding window of length L such that

$$\tilde{\mathbf{e}}(t) = \frac{1}{L}\sum_{s=t-L}^{t}\mathbf{e}(s). \tag{5.22}$$

We can collect evidence at every frame or only when there is some measurable change in one of the vector elements of $\mathbf{e}$. Likewise, we adjust the length L to mirror the sampling rate, opting for longer windows for more frequent sampling. To avoid redundant computations, we can also express Equation 5.22 recursively as

$$\tilde{\mathbf{e}}(t) = (1-\alpha_L)\tilde{\mathbf{e}}(t-1) + \alpha_L\mathbf{e}(t), \tag{5.23}$$

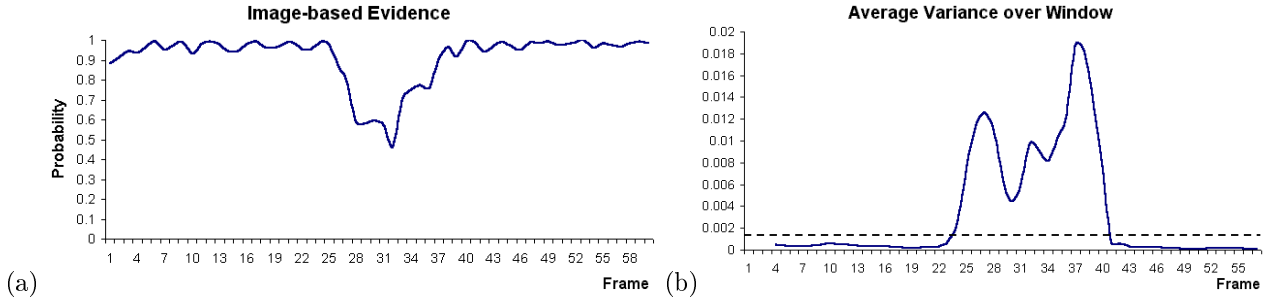
where α_L is a sample-weighted constant. For uniform contribution of evidence over the length of the window, we set $\alpha_L = \frac{1}{L}$. To assess the probability of Z_i given the evidence from class model M_k, we solve

$$P(Z_i|\tilde{\mathbf{e}}(i,k)) = \mathbf{w}^T\tilde{\mathbf{e}}(i,k), \tag{5.24}$$

where $\mathbf{w}$ represents a weight vector whose components are adjusted to reflect the strongest beliefs among available evidence. Each weight is defined as

$$w_j = \frac{\tilde{e}_j^2}{\tilde{\mathbf{e}}^T\tilde{\mathbf{e}}},\ \text{subject to}\ \sum_{j=1}^{4} w_j = 1. \tag{5.25}$$

[7]We emphasize here that the notion of time is discrete in nature. Future references to time are always implied, but we will only include t in equations when particularly relevant.



(a) (b)

Figure 5.6: (a) Probability of image evidence given class model $P(\Upsilon_i|M_k)$ (b) corresponding mean variance over window.

For each unknown Z_i, the most likely generalized class model ψ is the model in $\mathbf{M}$ producing the highest score, such that

$$\psi = \arg\max_{\mathbf{M}}\{\mathbf{w}^T\tilde{\mathbf{e}}(i,k)\}. \tag{5.26}$$

Additional biases are also placed on $\mathbf{w}$, which have the effect of dynamically adjusting the contribution of image-, object-, or action-based influences. For the j^{th} component of the evidence vector, we calculate the variance of the component's contribution over the length of the window, $\sigma^2_{e_j}$, as

$$\sigma^2_{e_j}(t) = \frac{L\sum_{s=t-L}^{t} e_j^2(s) - \left(\sum_{s=t-L}^{t} e_j(s)\right)^2}{L^2}. \tag{5.27}$$

We also take the sample mean of this variance measurement over a window of length L, i.e.,

$$\tilde{\sigma}^2_{e_j}(t) = \frac{1}{L}\sum_{s=t-\xi}^{t}\sigma^2_{e_j}(s). \tag{5.28}$$

By setting thresholds that are best discovered empirically, we can decide when large increases in the mean variance suggest dynamic context and, hence, potentially unstable evidence. For example, Figure 5.6-a shows the image-based likelihood $P(\Upsilon_i|M_k)$ of an unknown where major changes in evidence are detected during the middle of the sequence. The change excites an increase in mean variance measurements, which are shown in Figure 5.6-b. The corresponding weights for image-based evidence, w_1 and w_2, are set to zero when the mean variance exceeds the empirical threshold represented by the dashed-line. The remaining weights are re-normalized to satisfy the condition specified in Equation 5.25. Again, we

emphasize the importance of choosing an appropriate window length. Selecting L too long can introduce enough latency to compromise some of the benefits of dynamic, context-based weights.

Our naïve Bayesian classifer allows us to collect evidence over time. While Equation 5.26 guarantees that we will select the most likely class of the unknown, we require that belief assessment possess greater likelihood than the equi-probable condition of selecting any model (assuming each is equally likely to occur) for the domain $\mathcal{C}$, i.e., $P(Z_i|\tilde{\mathbf{e}}(i,\psi)) > \frac{1}{n_{\mathcal{C}}}$. Because belief is accumulated, we do not want to consider weak evidence, which may compromise more legitimate beliefs. In some cases, there are substantial similarities between GCMs that may cause our classifier to select multiple classes for an unknown. For example, classes with derived children often share the same likelihood scores unless there is additional information that can be used to disambiguate between them. As evidence is acquired, it is also possible for the maximum likelihood selection to change. Naturally, fluctuations in classification can cause problems in higher-level processes which make decisions from amalgamating low-level evidence. To insure stability with recognition, we can invoke an additional restriction that guarantees the selection of ψ will be distinguishable from neighboring models, i.e.,

$$|P(Z_i|\tilde{\mathbf{e}}(i,\psi)) - P(Z_i|\tilde{\mathbf{e}}(i,k))| \geq \tau_\psi \; \forall \; \psi \neq k \in \mathbf{M}, \tag{5.29}$$

where τ_ψ is a confidence threshold that we determine by experimentation. In the following sections, we consider results from several experiments designed to illustrate our approach to activity and object recognition.

5.3 Experiments

We conducted several experiments to evaluate our methodology for recognition. Experiment I identifies three different domains where people naturally interact with their surroundings. Common articles and activities in each domain were also selected. We attempt to capture low-level interactions with domain articles over extended periods of time, demonstrating event detection as well as hand tracking and recognition. High-level single-tasked activities are also recognized in each domain.

Domain	Activity	Recognition
office	reading	95%
	coffee break	90%
	computer work	87%
	taking phone message	60%
	counting documents	93%
kitchen	washing dishes	95%
	cooking stir fry	87%
	cleaning kitchen	60%
auto	accelerating (shifting up)	95%
	drinking-and-driving	87%
	winding road	93%
	roll window down	100%
	roll window up	100%
	parking car	85%

Table 5.2: Summary of single-tasked activities and corresponding recognition rates in three domains.

Experiment II exhibits ObjectSpaces' ability to classify unknown objects using image-, object-, and action-based information in office and kitchen domains. Similar experiments were conducted in Experiment III when only action-based information was used for classification of unknowns (demonstrating object recognition from action context). Experiment IV illustrates object classification of articles that are initially merged with the background. All of the experiments were conducted using the Vision Action Recognition System (see Appendix C).

5.3.1 Experiment I: Capturing Experiences

Office, kitchen, and automobile (interior) domains were configured with articles and a person was invited to perform many common tasks and activities. Performance scripts, i.e., sequence of anticipated actions like "start in neutral gear, then shift to first, then to second, etc.," for several activities in each domain were written first, then performed by a single user. We manually segmented and labeled 597 action examples of these video sequences for training the HMMs. Accuracy of interactive events captured by the system are shown in Table 5.3. Percentages for objects with no associated actions were based on detectibility alone; otherwise, recognition precision is shown. These events are the primitives used to construct descriptions for more involved single-tasked activities. Table 5.2 shows the

Figure 5.7: (a) Initial background snapshot of scene includes known articles: chair and keyboard. (b) Background after book, notebook, mouse, and printer articles are introduced. (c) Background subtraction reveals newly introduced articles.

recognition rates of several activities in various domains.

Throughout these interactions, 90% of all actions were recognized (assessed by hand-generated ground-truth observations). Using time-stamped logs, the system was also able to measure duration of hand-object contact. For the auto domain, a different person was used for testing than for training, illustrating person independent recognition of action since only hand position is used.

5.3.2 Experiment II: Object Recognition from Available Evidence

To demonstrate detection and recognition of newly introduced objects, several various objects (*book, notebook, printer*, and *mouse*) were carried into an office scene after the background was acquired. Table 5.3 shows the actions or events associated with each article. The system was already aware of other objects in the room, including a keyboard and chair. Segmentation began immediately as initial image-based evidence of the unknown objects was acquired and initial beliefs were forged. The background subtraction process for establishing appearance-based features of the unknown objects is shown in Figure 5.7.

As a person interacted with both known and unknown objects over several minutes, the strength of belief grew in proportion to the number of actions identified, as shown in Figure 5.8-a. The units of the horizontal axis are in *events*, representing when new image-, object-, and action-based evidence is contributed from one of the four unknowns. Appearance-based information was sufficient for establishing Z_2 and Z_3 early on (event

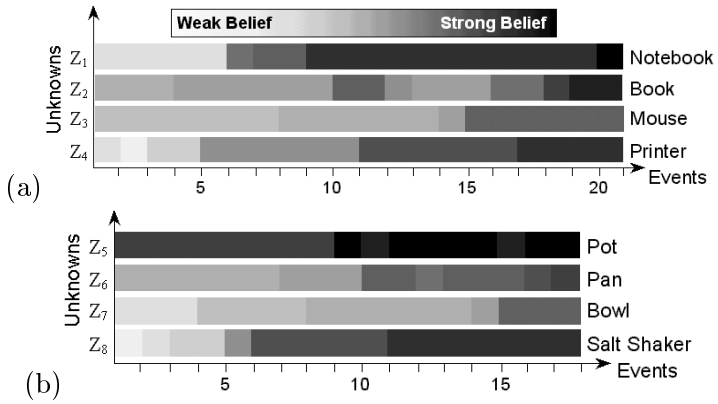


Figure 5.8: From experiment II, as evidence from one of four unknowns Z_i is collected, the strength of belief is shown in proportional to horizontal grayscale bars: (a) office environment (b) kitchen environment

1). While relevant actions were able to classify Z_1 as a notebook by event 9, conflicting actions registered to Z_2 during events 12 and 13 compromised belief testimony. Although Z_3 (mouse) has no actions associated with it, moderate belief can still be established by monitoring its interaction with a known article, in this case a keyboard (object-based evidence). In general, however, articles such as Z_3 stand a greater chance of being mis-labeled if actions associated with other GCMs are performed while interacting with it. Classification probability for Z_1 through Z_4 after 21 events (acquired over 1500 frames) was 97%, 94%, 80%, and 91%, respectively. Closer inspection reveals that 5, 8, 4, and 6 events for Z_1 through Z_4, respectively, were needed to achieve this degree of classification.

A similar experiment was conducted in the kitchen domain with a *pot, pan, bowl*, and *salt shaker* added to the scene already furnished with a stove, a pan, and cabinets. Classification probability of these 4 unknowns after 18 events was 98%[8], 85%, 77%, and 93%, respectively. A graph illustrating the strength of belief versus event-based episodes over time is given in Figure 5.8-b.

[8]Resulting from a template match of the same object stored in the database.

5.3.3 Experiment III: Object Recognition from Action

To evaluate the strength of action-based evidence, 11 action events that were acquired over 583 frames. Image evidence assisted in action recognition, but was not used to score GCMs during evaluation[9]. Figure 5.9-a shows the mean log probability of the candidate GCMs. Note that belief was shared between the GCM for notebook and book until event 7, when "write" was the most probable action observed, consequently rejecting book. Figure 5.9-b shows the accumulated likelihoods of several actions as they occurred throughout this sequence. It also reveals the potential for actions to be confused. Note that some actions that never actually occurred, such as "erase," have high, accumulated probabilities, suggesting that it is similar to several of the gestures performed. Also note that recognition is not affected by an object's deformable structure. As a testament to inferring classification from action, adding initial image-based evidence to this action-based information yields results that were only 3% higher.

5.3.4 Experiment IV: Object Recognition of Background Objects

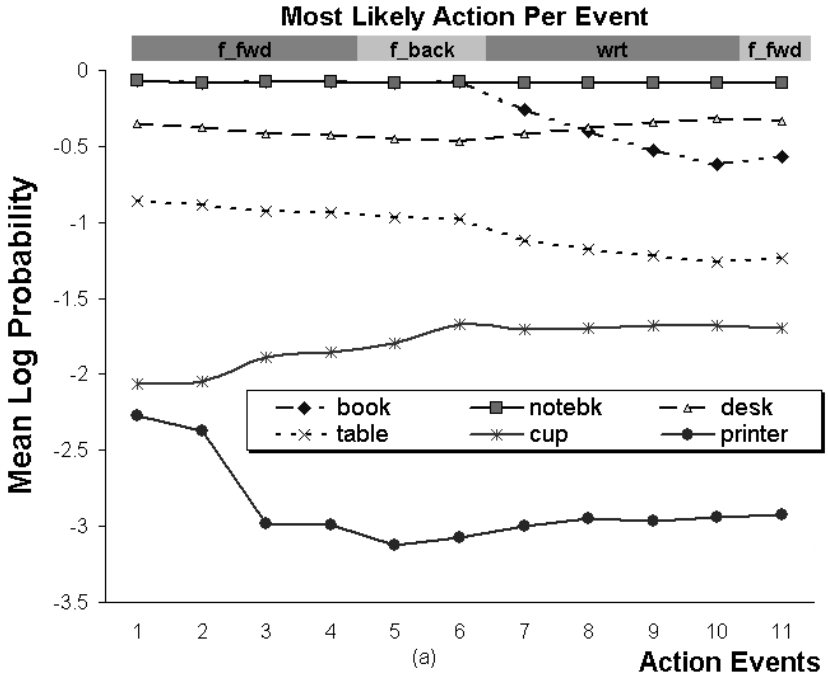
To demonstrate detection and recognition of objects initially merged with the background (full occlusion), we performed several eating actions (*stir, cut, feed*) in an undeclared space in the scene. When actions, such as stir, occur for more than one GCM, belief is shared. Without image-based segmentation, motion normalization suffers, resulting in lower action recognition rates and occasional mis-labeling. Notice in events 8 and 9 in Figure 5.9-c, "open" and "erase" were mistaken for "feed" and "cut". The table GCM exhibits the strongest belief, as shown in Figure 5.9-c.

Over many of these tests, belief ranged from 5%-17% lower over the same number of action events when image information was acquired, but not used for scoring GCMs. With no image evidence acquired, action recognition suffered even more and belief estimates dropped 14%-33%. Without an accurate assessment of region boundaries, which is typically provided by image-based evidence, motion normalization suffered.

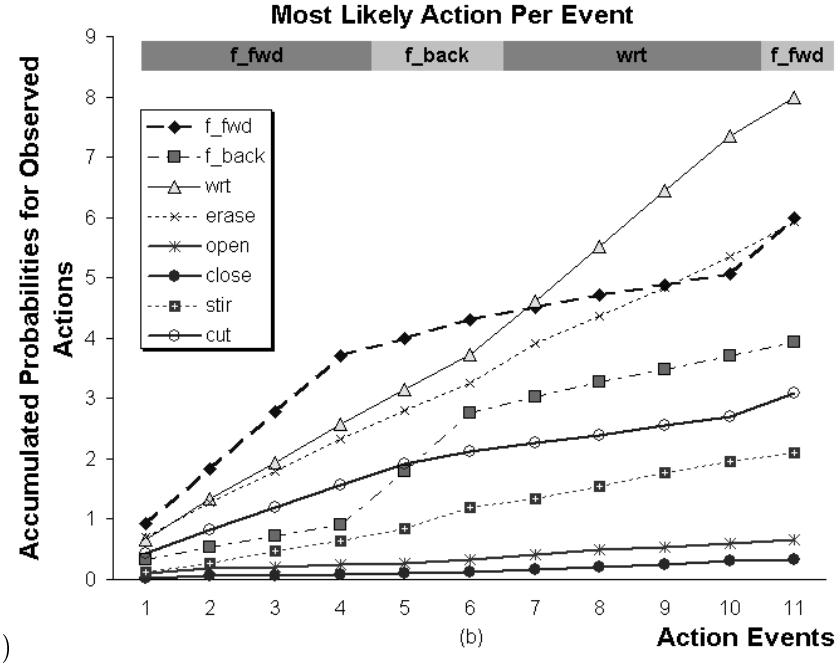
[9]Image-based evidence was used to the extent that region boundaries were recovered to normalize hand motion. However, we set appearance-based weights w_1 and w_2 to zero used in Equation 5.26 to select the most likely GCM.

Article	Action Recognition Accuracy
	Office Environment
bookcase	grab book - 100%, return book - 90%
book†	flip forward - 94%, flip backward - 90%
chair	no action, event only - 100%
cup	drinking - 100%, stir - 90%
desk	drawing - 93%, erasing - 86%, open drawer - 100%, close drawer - 80%
keyboard	no action, event only - 97%
mouse	no action, event only - 95%
notebook†	flip forward - 94%, flip backward - 92%, write - 80%
phone (rotary)	pick up - 90%, put down - 60%, dial number - 80%
printer†	open cover - 95%, close cover - 92%
table	feeding - 88%, stirring - 93%, cut - 85%
	Kitchen Environment
bowl	stir - 90%
cabinet	no action, event only - 100%
cut board	cut - 88%, scrape off - 93%
can opener	no action, event only - 100%
disposal	no action, event only - 100%
microwave†	open door - 100%, close door - 90%
pot/pan	stir - 85%
refrigerator	open door - 100%, close door - 90%
stove†	clean surface - 79%, open oven - 90%, close oven - 77%, adjust temp controls (no action, event only) - 95%
salt shaker	shake - 72%
sink	adjust water flow (no action, event only) - 100%, wash dish - 85%, grab rinse nozzle (no action, event only) - 100%
toaster	no action, event only - 95%
	Automobile Environment
cup	drink - 85%
stick shift†	gear changes: neut. $\rightarrow$ 1 - 100%, 1 $\rightarrow$ 2 - 90%, 2 $\rightarrow$ 3 - 100%, 3 $\rightarrow$ 4 - 90%, 4 $\rightarrow$ 5 - 100%, neutral $\rightarrow$ reverse - 100%
lock	no action, event only - 100%
parking brake	pull bake - 80%
radio	adjust (no action, event only) - 100%
steering wheel	turn left - 100%, turn right - 95%
temp control	adjust (no action, event only) - 100%
window handle	roll up - 100%, roll down - 100%

Table 5.3: *Experiment I*: Office, kitchen, & automobile objects with associated actions and recognition accuracy, respectively. †Class with multiple appearance descriptions stored in class.



(a)



(b)

Figure 5.9: (a) from experiment III, mean log probability of GCM classification over several action events; (b) from experiment III, shows the accumulated likelihoods of several actions as they occurred throughout the corresponding sequence, with the most probable action per event highlighted (top)

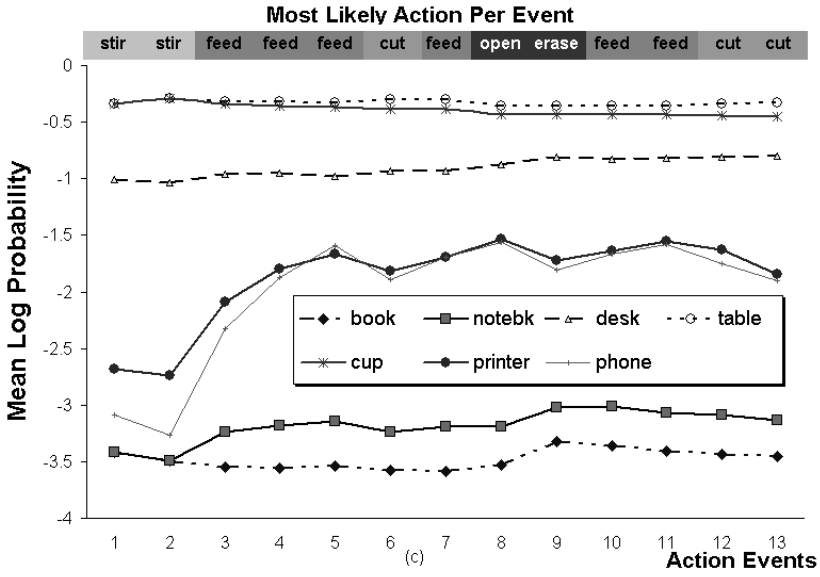


Figure 5.10: from experiment IV, GCM Mean log probability of unknown object without image-based segmentation

CHAPTER 6

Recognizing Multitasked Activities

In Chapter 5, techniques for integrating low-level information for object classification and recognition of single-tasked activities were introduced. In this chapter, we building on top of the foundation provided by those techniques to address the problem of representing complex, multitasked activities. Figure 6.1 shows how our ObjectSpaces framework is

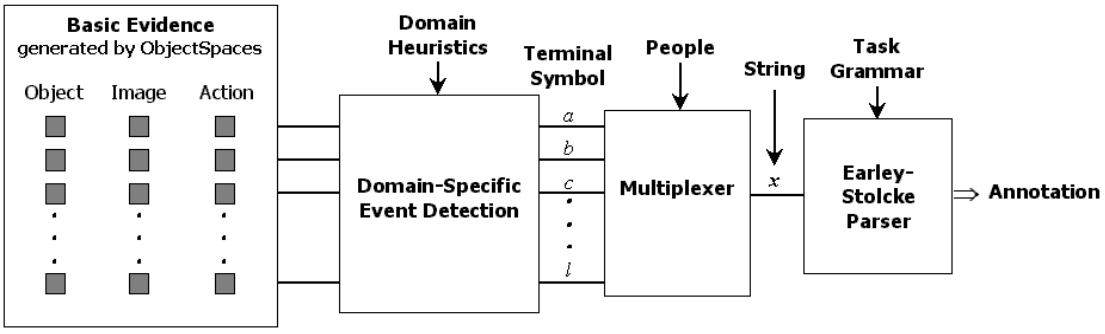


Figure 6.1: Framework for SCFG-based classification is an extension of ObjectSpaces.

leveraged by combining object-, image-, and action-based evidence with domain heuristics to label specific events. With each event labeled uniquely, we represent the sequence of these events as an ordered symbolic string. Stochastic context-free grammar is developed based on underlying rules of the activity to encode task-specific semantics. The Earley-Stolcke algorithm is employed to parse the string so that the most likely semantic derivation can be identified. Parsing also allows us to recognize substring patterns that provide high-level annotation of the video sequence. We introduce several novel approaches, including:

- adaptive stochastic grammar that improves recognition of structured activities,
- new parsing strategies that enable error detection and recovery in stochasitic context-

free grammar, and

- methods of quantifying group and individual behavior in activities with separable roles.

Experiments with the card game, Blackjack, produced high-level narratives of multi-player games and successful identification of player strategies and behavior. Our methods demonstrate recognition of extended, complex multitasked activities in real-time.

6.1 Characteristics of Multitasked Activities

So far, we have described approaches for handling single-tasked activities that typically involve only one person working towards a singular task or objective. To provide high-level awareness for a broader range of human activities, we also need to identify more complicated interactions, which can involve multiple people and objects. Many of these interactions are difficult to represent due to the loosely constrained context between multiple events, especially if these events are scattered over long periods of time. Naturally, the name "multitasked" suggests that several, single-tasked events are taking place around the same time. What may not be as clear is whether these events are participating as separate, independent processes or as an individual, coordinated affair, or even both. Intentionality can be very difficult, if not impossible, to model when a process is composed of many smaller, and perhaps disjoint, sub-tasks. To complicate matters further, the number of objects and people involved can change dynamically over the duration of the activity. Moreover, variations in the timing and execution of many events can be in reaction to other semantically significant events or simply improvised from one performance to another. Our ability to accurately measure and relate all of these interactions over time and space makes characterizing multitasked activities extremely challenging.

We define multitasked activities as *the intermittent concurrence of events involving multiple people, objects, and actions, typically over an extended period of time.* To use grammar effectively to represent these types of activities, we will only consider *rule-based, event-driven* multitasked activities, like juggling and playing card games. This subset represents activities that are constrained by unambiguous rules that completely specify task

semantics. Each rule has a corresponding event that, when detected, allows semantically meaningful behavior to be recognized. For convenience, any future mention of activities refers only to these types of rule-based, event-driven activities unless otherwise mentioned.

6.2 Modeling Multitasked Activities

Only a limited investigation in the areas of action recognition and activity understanding has been conducted to identify representations for multitasked activities. Our previous discussion of single-tasked activities primarily focused on methods for aggregating low-level actions and contact events over short periods of time, i.e., the length of time it takes to perform a few gestures. When attempting to characterize multitasked activities, we desire models that preserve semantic structures and tolerate interactions over longer periods.

The HMM fails to be an appropriate method because characterizing the joint states of the model (as shown in Figure 6.2) will likely introduce prohibitive computational costs, require an inordinate amount of training data, and still may produce results that are unsatisfactory. Moreover, many multitasked processes cannot be neatly dissembled without violating the integrity and character of the process itself. To overcome the limitations of the HMM, Brand proposes coupled hidden Markov models (CHMMs) to model interactions between processes that may have different time scales, state structures, and degrees of influence on each other [31]. Brand offers some limited evidence of the CHMM's ability to model highly structured multitasked processes using synthetic data as well as real, 3D hand tracking data of T'ai chi gestures. However, the rigid finite-state framework is simply inappropriate, by itself, to represent dynamic activities that occur over extended periods of time. Moreover, as the number of sub-tasks are combined, parameter estimation becomes NP-hard[1].

We believe that syntactical pattern classifiers that use grammar are needed to augment the limitations of finite state machines like the HMM. In general, finite state machines are appropriate for modeling a single hypothesis. As variation becomes more pronounced,

[1]For NP or *nondeterministic polynomial*-hard problems, no known algorithms are capable of generating an optimal solution in an amount of time that grows only as a polynomial function of the number of elements in the problem.

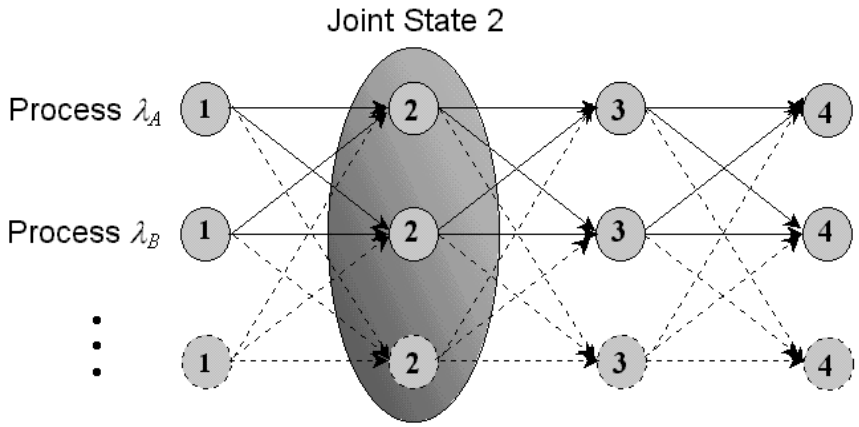


Figure 6.2: Joint state of HMM generated by the cross-product of all possible component states [31].

it becomes exceedingly difficult to collapse additional hypotheses into a single finite state model. In contrast, the generative process associated with grammar is non-deterministic, allowing the derivation of a much longer and elaborate sequence of events [137]. In other words, grammar allows us to use a single, compact representation for well-understood interactive events that also accommodates the natural generalizations that occur during their performance.

Bobick and Ivanov combine syntactic pattern recognition with statistical approaches to recognize activity taking place over extended sequences [24]. In their work, HMMs are used to propose candidates of low-level temporal features. These outputs provide the input stream for a stochastic context-free grammar parsing mechanism that enforces longer range temporal constraints, disambiguates or corrects uncertain or mislabeled low-level detections, and allows the inclusion of prior knowledge about the structure of temporal events in a given domain. Our approach extends the work of Bobick and Ivanov by exploiting the image-, object-, and action-based evidence generated by our object-oriented framework instead of using just HMM outputs. We also offer techniques to improve parsing with adaptive grammar and error recovery.

In the next section, we briefly discuss grammar and its utility for understanding multitasked activities.

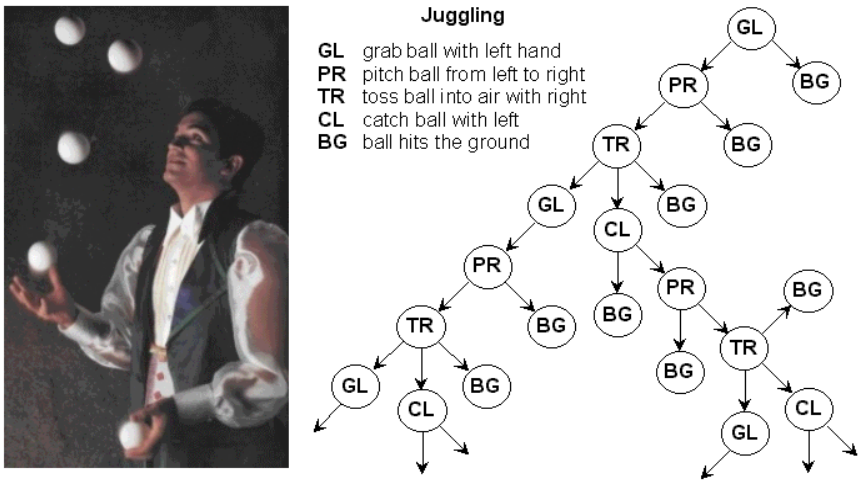


Figure 6.3: Grammar derivation tree describes a single person juggling balls into the air.

6.3 Grammar

Grammar is a mechanism that uses a system of rules to generate semantically meaningful sentences in a language. Specifically, grammar is the underlying representation of structure that we will eventually use to describe the interrelationships between patterns of evidence in activities. The language that is generated by this grammar should be capable of expressing all legitmate interactions of the activity.

As a motivating example of how grammar can be used to describe the structure of a multitasked activity, consider the art of juggling several balls in the air simultaneously. A person starts this activity by grabbing a ball with the left hand, then pitching it to the right. If the right hand does not drop the ball, the right tosses it into the air. While the ball is in the air, the left can prepare to catch it or grab another ball to put into motion. If no ball is dropped, the process repeats in a similar fashion, perhaps with several balls eventually being juggled simultaneously. Figure 6.3 shows an example of juggling along with a derivation tree that describes all possible event transitions. To represent this complex, multitasked activity using grammar, we define a small "alphabet" of basic events to describe the process, which is constrained by rules that govern how juggling is executed. A well-designed grammar provides a convenient way of accommodating redundancy in addition to handling events that can switch the order in which they occur but still represent a valid

interpretation of the activity.

Definition 6.1 Grammar

We define an *alphabet* to be any finite set of symbols. A *string* over an alphabet V is a finite sequence of elements from V. A *sentence* is any syntactically correct string of finite length composed of symbols from the alphabet. Formally, we define a *grammar* G as the quadruple $G = \{V_N, V_T, \mathcal{P}, S\}$, where V_N is a set of *nonterminals* or variables, V_T is a set of *terminals* or constants, $\mathcal{P}$ is a set of *productions* or rewriting rules, and S is the *start* or root symbol. The alphabet V_G is the union of the nonterminal and terminal alphabets, i.e., $V_G = V_N \cup V_T$, and S is assumed to belong to the nonterminal set, i.e., $S \in V_N$. The sets V_N and V_T are assumed to be disjoint. The *language* generated by G, denoted by $L(G)$, is any set (not necessarily finite) of sentences over alphabet V_G that satisfies the conditions: (1) each string is composed of only terminals, i.e., a *terminal string*, and (2) each string can be derived from S by suitable application of productions from the set $\mathcal{P}$. We refer to the table below regarding our notation:

Symbol	**Description**	**Example**
Capital Roman letter	Single nonterminal	$A, B, \ldots$
Small Roman letter	Single terminal	$a, b, \ldots$
Small Greek letter	String of terminals and nonterminals	$\lambda, \mu, \ldots$
Epsilon ϵ	Empty (null) string	ϵ

Let V_G^* denote the set of all strings composed of symbols from V_G, including the null string ϵ, and V_G^+ is the entire set of strings from V_G minus the null string, i.e., $V_G^+ = V_G^* - \epsilon$. The set of productions $\mathcal{P}$ consists of expressions of the form $\lambda \rightarrow \mu$, where the symbol "$\rightarrow$" indicates replacement of the string λ by the string μ. The symbol "$\Rightarrow$" will be used to denote operations of the form $\alpha\lambda\beta \Rightarrow \alpha\mu\beta$, where λ is replaced by μ by means of the production $\lambda \rightarrow \mu$, and α and β are left unchanged. For more information on syntactic pattern recognition and the different types of grammar, see Appendix D.

6.3.1 Representing Multitasked Activities using Stochastic Context-Free Grammars

Although stochastic context-free grammars (SCFG) are generally less popular than other grammars in computational linguistics, they are very appropriate for describing multitasked activities. The SCFG formalism has been likened to the HMM, as have non-probabilistic context-free grammar to finite-state grammars [132]. Standard algorithms for implementing probabilistic finite-state models, like HMMs, also have generalized versions for SCFGs. Unfortunately, they are computationally more demanding than simpler language models that employ finite-state and n-gram approaches. More specifically, parameter estimation for SCFG is often intractable in practice [132]. Due to the conditional independence assumptions used by SCFGs, finite-state models may perform better when quantifying first-order Markovian contingencies between words. However, Jurafsky and others argue that SCFGs can be applied successfully when using "very specific, semantically oriented categories and rules," especially with well-behaved grammars with small alphabets and precisely formalized combinatorial properties [80, 132]. Rule-based activities, in particular, make good candidates because they can be described using a relatively small lexicon of primitive events.

There are also several other advantages of SCFGs over more simple regular grammars. An n-gram grammar is a set of probabilities given by $P(x_n|x_1, x_2 \ldots x_{n-1})$, i.e., the probability that terminal symbol x_n follows the string $x_1 x_2 \ldots x_{n-1}$ for each possible combination of symbols in the alphabet. Typically n-gram grammars, like bigrams ($n = 2$), require a substantial amount of training data, and often smoothing techniques, to produce reliable estimation [134]. For example, when each $P(x_i|x_{i-1})$ is considered a parameter, a bigram grammar with a 20 symbol alphabet would have approximately $20 \times 20 = 400$ free parameters, and a trigram would have 8,000. As the size of the alphabet increases, the number of parameters increases exponentially. In contrast, a SCFG can have considerably fewer, depending on the rules of the grammar. In practice, these n-gram probabilities are usually pre-computed and made available at run-time in a lookup table. However, the parser still requires some searching mechanism to find the most likely transitions from one symbol to another. While there are several different "back-off" and pruning strategies designed to limit the number of choices, these algorithms can skip candidates, resulting in suboptimal

performance.

Stochastic context-free grammars can also be extended in a straight-forward fashion, based on existing knowledge embodied by the grammar. Extensibility is especially important for rule-based activities because it allows generically specified task grammars to be tailored for specific applications. For example, the card game Blackjack has several rules of play that change from casino to casino. These rules can be easily modified by proper section of a few productions without affecting other rules. Regular grammars are not easily extensible, but require recomputing the entire set of probabilities. Moreover, SCFG is more appropriate for modeling multitasked actions than regular grammar because it can capture semantic generalizations using more complex production rules.

The basic events of an activity become the terminal constants of the task grammar while the ordering of events is represented by nonterminal variables. It is convenient to specify the grammar by hand as opposed to "learning," which can be a difficult process that requires good initial conditions, a prodigious amount of training data and, typically, a fair amount of experimentation. More information on specifying grammar can be found in Appendix D.4.

Definition 6.2 *A* **stochastic context-free grammar** *is an extention of context-free grammar where a probability is added to every production rule:*

$$A \to \lambda \qquad [p].$$

We can also express the rule probability p as $P(A \to \lambda)$, which essentially gives the conditional likelihood of the production $A \to \lambda$. SCFGs are superior to non-stochastic context-free grammar because the probability attached to each rule provides a quantitative basis for ranking and pruning parses as well for exploiting dependencies in a language model. The reader is directed to Appendix D.3 for more informtion on rule probability estimation.

Formally, we represent the SCFG G as the quintuple $G = \{V_N, V_T, \mathcal{P}, S, P_r\}$, where P_r represents rule probabilities $P(r)$ for all $r \in \mathcal{P}$. In a SCFG, the probability of the complete derivation of a string is determined by the product of the rule probabilities in the derivation. The notion of context-freeness is extended to include probabilistic conditional independence

of the expansion of a nonterminal from its surrounding context [132]. There are a few basic definitions associated with SCFG G that will be important for future discussion:

Definition 6.3 *Associated terms of stochastic context-free grammar*

1. The probability of partial derivation $\nu_1 \Rightarrow \nu_2 \Rightarrow \ldots \nu_k$ is given using the inductive procedure:

   (a) $P(\nu_1) = 1$

   (b) $P(\nu_1 \Rightarrow \ldots \Rightarrow \nu_k) = P(A \rightarrow \omega)P(\nu_2 \Rightarrow \ldots \Rightarrow \nu_k)$, where production $A \rightarrow \omega$ is a production of G, ν_2 is derived from ν_1 by replacing one occurrence of A with ω, and $\nu_1, \nu_2, \ldots, \nu_k$ are strings of terminals and nonterminals in V_G^*.

2. The *string probability* $P(A \Rightarrow^* \lambda)$ is the sum of all the leftmost derivations $A \Rightarrow \ldots \Rightarrow \lambda$, i.e.,

$$P(\lambda|A) = \sum_{A \Rightarrow \ldots \Rightarrow \lambda} P(A \Rightarrow \ldots \Rightarrow \lambda), \tag{6.1}$$

   where the summation is over all string derivations that yield λ.

3. The *sentence probability* $P(S \Rightarrow^* \lambda)$ is the string probability of S, i.e.,

$$P(\lambda|G) = \sum_{S \Rightarrow \ldots \Rightarrow \lambda} P(S \Rightarrow \ldots \Rightarrow \lambda), \tag{6.2}$$

   again where the summation is over all string derivations that yield λ.

4. The *prefix probability* $P(S \Rightarrow_L^* \lambda)$ (of λ given G) is the sum of the strings having λ as a prefix, such that

$$P(S \Rightarrow_L^* \lambda) = \sum_{\omega \in V_G^*} P(A \Rightarrow^* \lambda\omega). \tag{6.3}$$

   In particular, $P(S \Rightarrow_L^* \epsilon) = 1$. Here we denote leftmost derivations of λ from S as $S \Rightarrow_L^* \lambda$. To reduce the number of derivations that must be considered by the parser, every string in a language is derivable in a leftmost manner [136]. The possibility of more than one distinct leftmost derivation points to possible ambiguity in the grammar, and hence, in the language.

6.4 Parsing SCFGs

Our motivation for using stochastic context-free grammar is to aggregate low-level evidence generated by our framework so that we can construct higher-level models of interaction. In this section, we discuss aspects dealing with terminal string generation and parsing.

6.4.1 String Generation from Event Detection

ObjectSpaces provides facilities for managing prior as well as newly discovered information about people, objects, and their interactions. We have presented this information as image-, object-, or action-based evidence that is collected from object-oriented "containers" as described earlier. Table 6.1 shows some of the raw object-oriented information contained in the scene layer of the framework, including object transition matrix **K** as mentioned in Section 5.1.2 . The corresponding actual screen capture representing this data is depicted in Figure C.5. Raw, low-level evidence represents measurements that, by themselves, say very little about what is happening. However, combining these measurements with domain-specific heuristics helps to detect patterns that represent meaningful events. When a particular event is observed, its corresponding symbolic terminal is appended to the end of the activity string x, which is parsed by the Earley-Stolcke algorithm. This process of representing low-level evidence using domain-specific symbols is called *tokenization*.

As an example, consider the event "person dealt card" in the domain of card games to be detected by detector D_{pdc}. The corresponding token symbol for this event should be generated every time a person places a new card on the table. Using background-subtraction methods mentioned earlier, the scene layer detects and classifies the new object entering the scene as a card. We assume that every article is either predefined during initialization or brought into the scene by one person. The scene attempts to match the location of each newly introduced article against the n most recent hand locations to determine which person was responsible for placing it in the scene, i.e., in our example, to determine who dealt the card. The range of frames involved in the search, empirically set to n, is influenced by the sequence frame rate. From current frame t to frame $t - n$, the minimum square distance

People(2)		**Articles**(9)	
name	"dealer"	name	"card"
ID	0	class	PlayingCard
color distribution	$135_Y 56_U 68_V, \ldots$	value	facedown
maxHandMovement	200	focus Of Attention	false
LH confidence	0.915	moveable	true
RH confidence	0.742	recognition	0.921
LH Position	(54,44)	bounding box	(179,180,211,204)
RH Position	(137,119)	introduced by ID	0
est. LH Position	(54,44)	*Edgemap*	32×24 binary,...
est. RH Position	(116,98)	*BG Frame*	32×24 YUV,...
prev. LH Positions	(54,44),(54,4...	*FG Frame*	32×24 YUV,...
prev. RH Positions	(154,140),(15...	no. events	0
LH Region	(33,30,89,152...	*event history*	...
RH Region	(108,103,170,...	*HMM database*	...
headtorso Region	...	*current event*	...
name	"player"	name	"deck"
ID	1	class	CardDeck
color distribution	$145_Y 66_U 80_V, \ldots$	value	facedown
maxHandMovement	150	focus Of Attention	false
LH confidence	1.0	moveable	true
RH confidence	0.975	recognition	1.0
LH Position	(254,191)	bounding box	(118,68,156,88)
RH Position	(251,74)	introduced by ID	-1
est. LH Position	(251,191)	*Edgemap*	38×20 binary,...
est. RH Position	(251,74)	*BG Frame*	38×20 YUV,...
prev. LH Positions	(254,191),(25...	*FG Frame*	38×20 YUV,...
prev. RH Positions	(251,71),(25,...	no. events	2
LH Region	(226,173,281,...	*event history*	3.14 sec, ID(0),...
RH Region	(229,63,277,9...	*HMM database*	...
headtorso Region	...	*current event*	...

$$\mathbf{K} = \begin{matrix} 0.321 & 0.058 & \ldots \\ 0.125 & 0.895 & \ldots \\ \vdots & & \ddots \end{matrix}$$

Scene BG Frame	"BJ21-bkg.bmp"
FrameTimeStamp	00h:00m:34s
NumHandZones	0

Table 6.1: Snapshot of unprocessed, low-level object-oriented evidence collected by ObjectSpaces during a Blackjack card game. Items in *italics* are objects containing more embedded data. Corresponding screen capture illustrates actual VARS data in Figure C.5.

between each hand centroid and the centroid of article i is calculated as

$$d(i,n,t,\mathbf{p}) = \min_{\mathbf{p}}\{(\mathbf{x}_{l,p}(j)-\mathbf{x}_i)^T(\mathbf{x}_{l,p}(j)-\mathbf{x}_i),(\mathbf{x}_{r,p}(j)-\mathbf{x}_i)^T(\mathbf{x}_{r,p}(j)-\mathbf{x}_i)\} \;\; \forall\, j \in [t-n,t], \tag{6.4}$$

where $\mathbf{x}_i$ is the centroid of article i (assumed stationary), in this case a card, $\mathbf{x}_{l,p}(j)$ and $\mathbf{x}_{r,p}(j)$ denotes the left and right hand centroids in frame j, respectively, and $\mathbf{p} = \{p_1, p_2, ...p_m\}$ represents the m people in the scene. An illustration of this procedure is available in Figure 6.4, where the shaded boxes represent previous hand locations. To associate hand contact with article i, the distance $d(i,n,t,\mathbf{p})$ must be within a minimum distance specified by $d_{min}(i)$, which is given by

$$d_{min}(i) = \alpha_i \left((x_r - x_l)^2 + (y_b - y_t)^2\right). \tag{6.5}$$

Recall that $\mathbf{z}_i = [x_l \;\; y_t \;\; x_r \;\; y_b]^T$ specifies the bounding region surrounding an article. Empirical values established for α_i are motivated by the dimensions of the hand's bounding region relative to the bounding region of the article. In practice, for small articles, $\alpha_i \geq 1$. Finally, to determine the last person $\hat{p}$ to touch an object, we solve

$$\hat{p} = \begin{cases} \arg d(i,n,t,\mathbf{p}) & \text{if } d(i,n,t,\mathbf{p}) \leq d_{min}(i) \\ -1 & \text{otherwise} \end{cases}, \tag{6.6}$$

where -1 is a special ID reserved for the scene, i.e., person can not be determined. This operation is also triggered by the disappearance of an article in order to determine who removes an article from the scene. Needless to say, determining who deals or removes cards in this domain enhances our ability to detect semantically important events.

We would also like to attach some measure of confidence about the likelihood of events. Continuing with our example of person $\hat{p}$ dealing a card, we express the likelihood of this event as

$$P(\text{Person } \hat{p} \text{ deals a card}) = P(\text{card} \mid \text{new object})P(\text{Person } \hat{p} \text{ introduced object}).$$

Recall from Equation 5.24 that the first term is expressed as the probability of the generalized class model M_k given the available image-, object-, and action-based evidence related to the unlabeled object Z_i, i.e., $P(Z_i|\tilde{\mathbf{e}}(i,k))$. The second term is influenced by those hand

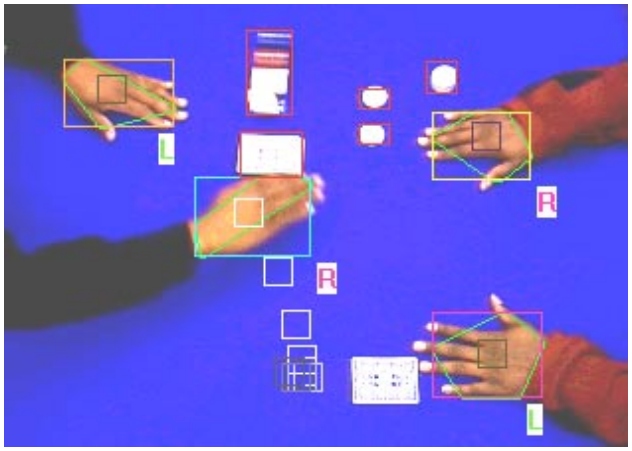


Figure 6.4: Shaded from white to black, small boxes indicate previous n hand positions (white indicating the most recent). The minimum square distance between each hand centroid and object centroid is found to determine the last person to touch an article.

centroids identified within the circular region centered at the article's centroid, extending with radius $d_{min}(i)$, i.e.,

$$\begin{aligned} P(\hat{p}) &= P(\text{Person } \hat{p} \text{ introduced object}) \\ &= \frac{d_{min}(i) - (\tilde{\mathbf{x}}_{\hat{p}} - \mathbf{x}_i)^T(\tilde{\mathbf{x}}_{\hat{p}} - \mathbf{x}_i)}{\sum_{p \in \tilde{\mathbf{p}}} d_{min}(i) - (\tilde{\mathbf{x}}_p - \mathbf{x}_i)^T(\tilde{\mathbf{x}}_p - \mathbf{x}_i)}, \end{aligned} \tag{6.7}$$

where $\tilde{\mathbf{p}}$ represents the set of people with hand locations inside the region specified by $d_{min}(i)$, $\tilde{\mathbf{x}}_p$ denotes the hand location with minimal distance to the card's centroid $\mathbf{x}_i$ for each person $p \in \tilde{\mathbf{p}}$, and $\tilde{\mathbf{x}}_{\hat{p}}$ is the location that minimized Equation 6.4. Equation 6.7 generates a likelihood based on how close the hands of person $\hat{p}$ came to the card's centroid, relative to any other person that may have also had hands near $\mathbf{x}_i$. So using Equations 5.24 and 6.7, the likelihood of the event is given by

$$P(D_{pdc}) = \{\max_{\mathbf{M}} P(Z_i|\tilde{\mathbf{e}}(i,k))\}P(\hat{p}). \tag{6.8}$$

We design additional heuristic detectors in a similar fashion to provide rich, meaningful information about interactions. For example, in the card game domain we construct similar detectors for determining when betting chips are added or removed from the scene. Where applicable, we also consider an articles's location relative to the entire scene as well as in respect to other objects.

To the extent possible, however, we refrain from constructing application-specific detectors that generate high-level information. Rather, we attempt to construct generic, low-level indicators that can be utilized with a broad range of applications. Our goal is to use various syntactic pattern recognition tools to assemble relevant low-level events so that important domain semantics can be explained. For some domain $\mathcal{C}$, we let $\mathbf{D}_{\mathcal{C}} = \{D_1, D_2, \ldots\}$ represent the set of detectors for generating the set of terminal symbols V_T. For convenience, the likelihood of a detected event, i.e., the likelihood of generating the terminal symbol x_i corresponding to detector D_i, is given by

$$P_{\mathbf{D}}(x_i) = P(D_i),$$

where $P(D_i)$ is defined on a case by case basis, similarly to Equation 6.8. By processing an activity sequence in domain $\mathcal{C}$, we use $\mathbf{D}_{\mathcal{C}}$ to generate symbolic string $x = x_1 x_2, \ldots, x_l$, where $l = |x|$).

When parsing is discussed in the next section, we will show that it is possible to compute the syntactical likelihood of a sequence of events $P(x)$ (assuming perfect detection). Such a likelihood offers a measure of how much semantic merit a sequence has. We have also shown that it is possible to develop a measure of confidence regarding the detection of individual events. Using the independence assumption guaranteed by the use of context-free grammar, we can also describe the likelihood of string formation based on detection alone, i.e.,

$$P_{\mathbf{D}}(x) = \prod_{i=1}^{l} P_{\mathbf{D}}(x_i). \tag{6.9}$$

Unfortunately, as the length l increases, the overall likelihood of the string decreases from repeated multiplication of values less than unity. A better measure of confidence normalizes the likelihood according to l, which we describe by simply calculating the sample mean likelihood of the string, i.e.,

$$\tilde{P}_{\mathbf{D}}(x) \frac{1}{l} \sum_{i=1}^{l} P_{\mathbf{D}}(x_i). \tag{6.10}$$

6.4.2 The Earley-Stockle Parsing Algorithm

In this section, we introduce the Earley-Stolcke algorithm, a parser originally developed by Jay Earley for efficient parsing of CFG and later modified by Andreas Stolcke to ac-

commodate SCFG[2]. The Earley-Stolcke algorithm uses a top-down parsing approach and context-free productions to build strings that are derivable from left to right. Starting at the top of a semantic derivation tree, much like the one presented earlier in Figure 6.3, top-down parsing begins with the starting symbol and attempts to build semantic sentences as it works its way down the tree. In contrast, other parsers operate in a bottom-up approach, which starts with the "sentence" and attempts to work its way up the tree to the starting symbol.

The Earley-Stolcke algorithm maintains multiple hypotheses of all possible derivations that are consistent with the input string up to a certain point. Scanning input from left to right, the number of hypotheses increases as new options become available or decrease as ambiguities are resolved. An additional benefit is that the task grammar does not have to be in Chomsky Normal Form (CNF)[3] to be accepted by an Earley-Stolcke parser. However, a grammar is more extensible, is easier to read, and expresses semantic relationships in greater detail if productions limit the number of nonterminals on the right hand side of the arrow.

A set of states, determined by the length of the string, is created for each position in the input. This state describes all pending derivations. The entire set of states forms the *Earley chart*. We preserve notation[4] introduced by Earley to represent a state, which is given by

$$i : \ {}_kX \rightarrow \lambda . \mu \ , \tag{6.11}$$

where i is the index of the current position in the input stream and k is the *starting* index of the substring given by nonterminal X. Nonterminal X contains substring $x_k \ldots x_i \ldots x_l$, where x_l is the last terminal of the substring μ. When we are in position i, the substring $x_0 \ldots x_{i-1}$ has already been processed by the parser. *State set* i represents all of the states that describe the parsing at this position. There is always one more state set than input symbols, i.e., set 0 describes the parser before any input is processed while set l depicts the parser after all processing.

[2]See Appendix D.6 for more information on the complexity of the Earley-Stolcke algorithm.

[3]Chomsky Normal form has rules of the form: i) $A \rightarrow BC$, ii) $A \rightarrow a$, or iii) $S \rightarrow \lambda$.

[4]Earley notation uses **only one** "." (index point) when defining states. The reader is encouraged to pay close attention to the location of the index, which is easily confused with a period.

Parsing proceeds iteratively through three steps: *prediction, scanning,* and *completion.* States produced by each of these steps are called *predicted, scanned,* and *completed*, respectively. A state is called *complete* (not to be confused to *completed*) if the substring $x_j \dots x_i$ has been fully expanded and is syntactically correct (which is written with the dot located in the rightmost position of the state, i.e., $i :\ {}_jY \to \nu.$).

Forward & Inner Probabilities

Recall that the string, sentence, and prefix probabilities as defined in Definition 6.3 are determined by summing derivation probabilities. To exploit the iterative steps in the parsing process, we introduce two probabilities that are somewhat analogous to the terms commonly used with HMMS.

Definition 6.4 *To determine the likelihood of a string at the current index i, the* **forward probability** α *gives the likelihood of selecting the next state at step i, along with probability of selecting previous states, i.e., $x_1 \dots x_{i-1}$.*

Definition 6.5 *The* **inner probability** γ *measures the likelihood of generating a substring of the input from a given nonterminal using a particular production.*

Unlike the forward probability, which starts from the beginning of the string, the inner probability starts at position k in the string.

Prediction

The prediction step is used to hypothesize the possible continuation of the input based on the current position in the derived string. We expand one branch of the derivation tree down to the set of its leftmost term to predict the next possible input terminal. So, given state $i :\ {}_kX \to \lambda.Y\mu$ with the production $Y \to \nu$, we can predict state

$$i :\ {}_iY \to .\nu\ [\alpha, \gamma]. \tag{6.12}$$

Mandated by Definition 6.3(a), the parser starts out in the *dummy state*

$$0 :\ {}_0 \to .S\ [1, 1],$$

where S is the sentence nonterminal or start symbol. Recall that in a SCFG, there are rule probabilities associated with choosing a predicted state.

Scanning

During prediction, we create a list of all the states that are syntactically possible based on prior input. These states provide the candidate terminal symbols that we can anticipate at the next position in the input string. The scanning step is where we read the next input symbol and match it against all states under consideration. For each state generated by the previous prediction stage, i.e.,

$$i : \ {}_kX \rightarrow \lambda.a\mu \ [\alpha, \gamma],$$

where a is a terminal symbol that matches the current input x_i, we add the next state given by

$$i+1 : \ {}_kX \rightarrow \lambda a.\mu \ [\alpha, \gamma].$$

Note that the dot has moved to the right, over the *current* symbol a to indicate the position change from state i to $i+1$. Scanning ensures that the terminal symbols produced in a derivation match the input string. No terminals are generated at this stage, so the forward and inner probabilities are unchanged. The scanning step promotes the states for the next iteration.

Completion

Given a set of states which have just been confirmed by scanning, the completion step updates the positions in all pending derivations throughout the derivation tree. Each completion corresponds the end of a particular nonterminal expansion which was initiated by an earlier prediction step. For each complete state

$$i : \ {}_jY \rightarrow \nu. \ [\alpha'', \gamma''],$$

and each state in the set j, such that $j < i$, that has nonterminal Y to the right of the dot, i.e.,

$$j : \ {}_kX \rightarrow \lambda.Y\mu \ [\alpha, \gamma],$$

generates

$$i: \ {}_kX \rightarrow \lambda Y.\mu \ [\alpha^{'}, \gamma^{'}],$$

where the dot is moved over the current nonterminal and

$$\begin{aligned} \alpha^{'} &= \sum_{\forall \ \lambda,\mu} \alpha(j: \ {}_kX \rightarrow \lambda.Y\mu)\gamma^{''}(i: \ {}_jY \rightarrow \nu.) \\ \gamma^{'} &= \sum_{\forall \ \lambda,\mu} \gamma(j: \ {}_kX \rightarrow \lambda.Y\mu)\gamma^{''}(i: \ {}_jY \rightarrow \nu.) \ . \end{aligned} \tag{6.13}$$

We take note that $\alpha^{''}$ is not used because $\gamma^{''}$ weighs the probabilities of all paths expanding $Y \rightarrow \nu$. After processing the last symbol in the string, the parser confirms that the sentence has been completed

$$l: \ {}_0 \rightarrow S. \ , \tag{6.14}$$

where l is the length of the input string x. At any position in the parsing process, if a state remains incomplete because no states from the previous stage permit scanning, the entire process can be aborted, signifying a syntax error in the string.

Parsing Example using Non-probabilistic Context-Free Grammar

To illustrate the three steps of the Earley-Stolcke parsing algorithm, we first consider non-probabilistic parsing of a grammar, as shown in Table 6.2. Conventional Earley parsing of CFG is essentially the same as Earley-Stolcke parsing of SCFG without considering the forward and inner probabilities. The parser will process the short sentence ab. We begin in state set 0, with the dummy state ${}_0 \rightarrow .S \ \ .$ The first step is always prediction, which allows us to generate hypotheses that are syntactically consistent with our grammar. The parser considers all rules that S can generate that will eventually produce a terminal symbol. We consider all three productions of starting symbol S. In prediction, the dot is always to the left of both terminals and nonterminals. If the symbol to the right of this dot is a nonterminal, then we expand the nonterminal further, until we arrive at a terminal candidate. The state set where the prediction takes place is always designated by the subscript on the left. Note that predicted state ${}_0A \rightarrow .AB$ has recursive properties that may cause it to regenerate itself indefinitely. For CFG, there is no additional information gained by adding a redundant state, in which case the parser can simply stop. However,

(a)

S	$\rightarrow$	AB
	$\rightarrow$	C
	$\rightarrow$	d
A	$\rightarrow$	AB
	$\rightarrow$	a
B	$\rightarrow$	bB
	$\rightarrow$	b
C	$\rightarrow$	BC
	$\rightarrow$	B
	$\rightarrow$	c

(b)

	a	b
${}_0 \rightarrow .S$	*scanned*	*scanned*
predicted	${}_0A \rightarrow a.$	${}_1B \rightarrow b.$
${}_0S \rightarrow .AB$	*completed*	${}_1B \rightarrow b.B$
${}_0S \rightarrow .C$	${}_0A \rightarrow A.B$	*completed*
${}_0S \rightarrow .d$	${}_0S \rightarrow A.B$	${}_0S \rightarrow AB.$
${}_0A \rightarrow .AB$	*predicted*	${}_0 \rightarrow S.$
${}_0A \rightarrow .a$	${}_1B \rightarrow .bB$	*predicted*
${}_0C \rightarrow .BC$	${}_1B \rightarrow .b$	${}_2B \rightarrow .bB$
${}_0C \rightarrow .B$		${}_2B \rightarrow .b$
${}_0C \rightarrow .c$		
${}_0B \rightarrow .bB$		
${}_0B \rightarrow .b$		
State set 0	1	2

Table 6.2: *Earley parsing*: (a) Example grammar (b) Parsing steps of the string *ab*.

for SCFG, the probability of each redundant state adds to the probability of the previously predicted state, generating an infinite sum. In the next section, we offer a remedy for left-recursive expansions.

State set 1 is initiated by scanning the input. The predicted states: ${}_0S \rightarrow .d$, ${}_0A \rightarrow .a$, ${}_0C \rightarrow .c$, ${}_0B \rightarrow .bB$, and ${}_0B \rightarrow .b$ offer choices as to the next terminal that can be generated from various nonterminal expansions. Terminal symbol a is sampled, and a search of all predicted states begins to match candidate terminals to the symbol. Only one predicted state matches the current input, which we denote by moving the dot to the right of the matched terminal to get ${}_0A \rightarrow a.$. We also preserve other "paths" that generated predicted state ${}_0A \rightarrow .a$, i.e., ${}_0A \rightarrow .AB$, ${}_0S \rightarrow .AB$, and ${}_0 \rightarrow .S$. We define a *path* as a sequence of states linked by prediction, scanning, or completion. All other predicted states from set 0 are pruned.

During completion, we search through the pending states that remain, looking for nonterminals that have completed their expansion. Anytime a scanned state has a dot in the right-most position on the RHS of the production, the nonterminal expansion is complete. States that also predicted this nonterminal are also promoted towards completion by moving the dot to the right, i.e., ${}_0A \rightarrow .AB$ becomes ${}_0A \rightarrow A.B$ and ${}_0S \rightarrow .AB$ becomes ${}_0S \rightarrow A.B$.

Prediction begins as before, but instead of starting from ${}_0 \rightarrow .S$, the parser looks at

scanned and completed states that have remaining symbols to the right of the dot. In this case, we have states ${}_0A \rightarrow A.B$ and ${}_0S \rightarrow A.B$, which predict states ${}_1B \rightarrow .bB$ and ${}_1B \rightarrow .b$.

In the final state set, completed states include ${}_0 \rightarrow S.$, which confirms that our sentence checks out. However, we can still continue with prediction in the event that another symbol is amended to the sentence. If any other symbol besides b were to be scanned, the sentence could not be parsed and would be marked as syntactically invalid.

6.4.3 Calculating Forward and Inner Probabilities

We arrive at prefix and string probabilities, given by Equations 6.3 and 6.1, respectively, by summing the probabilities associated with each state generated by a production derivation. One such probability associated with each state is the forward probability $\alpha_i({}_kX \rightarrow \lambda.\mu)$, which refers to the likelihood of an Earley-Stolcke parser generating the prefix of the input up to position $i-1$ while passing through the state ${}_kX \rightarrow \lambda.\mu$ at position i. In other words, it is the sum of the probabilities of all constrained paths of length i that end in state ${}_kX \rightarrow \lambda.\mu$.

The other is the inner probability $\gamma_i({}_kX \rightarrow \lambda.\mu)$, which represents the sum of the probabilities of all paths of length $i-k$ that start in state $k:\ {}_kX \rightarrow .\lambda\mu$ and end in $i:\ {}_kX \rightarrow \lambda.\mu$, and generate the input symbols $x_k \ldots x_{i-1}$. It represents the probability of generating a substring of the input from a given nonterminal using a particular production. Inner probabilities are conditional on the presence of a given nonterminal X with expansion starting at position k. They differ from the forward probability, which include the entire history starting with the initial state.

A left recursion occurs when productions of the form

$$\begin{aligned} A &\rightarrow Aa \\ &\rightarrow a \end{aligned}$$

appear in grammar during the prediction stage. A unit production, which is identified by

the productions,

$$\begin{aligned} A &\rightarrow B \\ &\rightarrow a \\ B &\rightarrow A \end{aligned}$$

and the state $i: \ {}_jA \rightarrow a.$ during the completion stage of set j, which contains

$$\begin{aligned} j: \ {}_jA &\rightarrow .B \\ j: \ {}_jA &\rightarrow .a \\ j: \ {}_jB &\rightarrow .A \end{aligned}$$

represent examples of recursive grammar that can throw the parser into an infinite loop[5]. If it were not for the left recursions and unit productions, calculating these probabilities would be trivial. However, we summarize the corrections for these special cases in the chart below. For convenience, we copy Stolcke's notation[6] to represent $\alpha_i({}_kX \rightarrow \lambda.\mu)$ as α, $\gamma_i({}_kX \rightarrow \lambda.\mu)$ as γ, $R(Z \Rightarrow_U^* Y)$ as R_U, and $R(Z \Rightarrow_L^* Y)$ as R_L. The single prime over symbols signifies the corrected representation.

Prediction	**Scanning**	**Completion**
$\alpha' += \alpha R_L P(Y \rightarrow \nu)$	$\alpha' = \alpha$	$\alpha' += \alpha\gamma'' R_U$
$\gamma' = P(Y \rightarrow \nu)$	$\gamma' = \gamma$	$\gamma' += \gamma\gamma'' R_U$

The careful reader will notice that the probabilistic left-corner and unit production matrices R_L and R_U, respectively, are required to provide the corrections in the prediction and completion stages. Booth and Thompson provide a rigorous proof that guarantees the existence of these matrices if the grammar is well-behaved according to axioms provided by [27]. For further clarity, Stolcke also provides a simple example in Table D.3.

6.4.4 Viterbi Parse

Motivated by the use of the Viterbi parsing in the HMM, we can also apply a generalization of the Viterbi method for parsing a string x to retrieve the most likely probability among

[5]For more information on recursive grammar, see Appendix D.5.

[6]The notation $x+=y$, which is borrowed from the C language, means that x is computed incrementally as a sum of various y terms.

all possible derivations for x. For SCFG, a Viterbi parse yields the most likely derivation path of the string. In our case, this will give the most likely interactive summary of events over the duration of the sequence. Path probabilities are recursively multiplied during completion steps using the inner probabilities as accummulators [132]. To compute the Viterbi parse, each state set must maintain the maximal path probability leading to it as well as the predecessor states associated with that maximum likelihood path. By familiar backtracking along the maximal predecessor states, the maximum probability parse can be recovered.

To implement this Viterbi computation, we modify the parser such that:

- Each state computes its *Viterbi probability* v.
- v is propagated similarly to γ, except that during completion the summation is replace by maximization, such that $v_i({}_kX \rightarrow \lambda Y.\mu)$ is the maximum of all products $v_i({}_jY \rightarrow \nu.)v_j({}_kX \rightarrow \lambda.Y\mu)$ along paths that lead to the completed state ${}_kX \rightarrow \lambda Y.\mu$, i.e.,

$$v_i({}_kX \rightarrow \lambda Y.\mu) = \max_{\lambda,\mu}\{v_i({}_jY \rightarrow \nu.)v_j({}_kX \rightarrow \lambda.Y\mu)\}. \tag{6.15}$$

  The state associated with the maximum is listed as the Viterbi path predecessor of ${}_kX \rightarrow \lambda Y.\mu$, i.e.,

$${}_kX \rightarrow \lambda.Y\mu = \arg\max_{\lambda,\mu}\{v_i({}_jY \rightarrow \nu.)v_j({}_kX \rightarrow \lambda.Y\mu)\}. \tag{6.16}$$

- The inner probability γ used in these calculate uses the original recursion in lieu of the corrective recursion formula given by Equation D.12. Unit production loops are avoided since they lower a path's probability. Moreover, we need the litany of predecessor states in order to perform backtracking. However, we can safely keep corrective formulas used in the prediction phase.

A familiar recursive procedure is required to recover the maximum likelihood (ML) derivation tree associated with the Viterbi parse. During the normal parsing operation described earlier, state ${}_kX \rightarrow \lambda.Y\mu$ maintains a pointer to the state ${}_jY \rightarrow \nu.$ that completes it, providing a path for backtracking. After arriving in the final state, the ML tree is reproduced by visiting predecessor states as identified by pointers.

6.5 Parsing and the ObjectSpaces Framework

In this section, we describe parsing methods that leverage our ObjectSpaces framework which, allow easy segmentation of events relative to individual people or objects as required. Methods for handling uncertainty in the input as well as adapting grammar to improve recognition are also surveyed.

6.5.1 Parsing in Uncertainty

There has been a presumption that all detected sentences are syntactically agreeable by the parser, but in practice, human and system error can produce activity sentences that are semantically meaningless. In Equation 6.4.1, we introduced $P_{\mathbf{D}}(x_i)$, which is the probability of the detectors producing symbol x_i. We factor in the likelihood of the input into the parsing mechanism by multiplying $P_{\mathbf{D}}(x_i)$ by the forward and inner probabilities during the scanning step, i.e.,

$$\alpha' = \alpha(i : {}_kX \rightarrow \lambda.a\mu)P_{\mathbf{D}}(a) \tag{6.17}$$

$$\gamma' = \gamma(i : {}_kX \rightarrow \lambda.a\mu)P_{\mathbf{D}}(a), \tag{6.18}$$

where a is the terminal sampled from the input in state set i. The revised values for α' and γ' reflect the weight of the likelihood of competing derivations as well as the certainty associated with the scanned input symbol.

6.5.2 Parsing Separable Activities

In our event-driven framework, we assume that some agent, or person, is responsible for inducing interactions. As more people engage in an activity, it becomes exceedingly difficult to parse high-level behavior from a single string because event symbols are interlaced. Parsing the actions of several people from one string is likened to listening to multiple speakers. If interactions are choreographed in a manner that can be unambiguously represented by the task grammar, i.e., no two speakers ever talk at the same time, de-interlacing terminals in the string can proceed without much confusion. On the other hand, if unscheduled interactions occur intermittently during an activity, i.e., multiple speakers talk at the same time, segmenting the events generated by an individual becomes nontrivial.

One treatment for handling interlaced strings is to create a separate grammar to describe the task semantics relative to the number of people participating. However, the string could be generated by multiple, perhaps, disjoint grammars as the number of participants vary throughout the course of an activity, making parsing impossible. Another is to modify the grammar to accommodate every potential arrangement of events as they are generated by each agent. However, the resulting grammar may be too confusing to parse or complicated to represent succinct, semantic expressions. While the list of alternatives is long, we feel the most reasonable approach is to exploit our object-oriented framework, which provides automatic segmentation of interactions.

Bobick and Ivanov deal with multi-agent interactions by assigning a class label to each production rule of the grammar [75]. After performing filtering operations to enforce temporal consistency of an interlaced string, they can segment tracks that describe interactions for each object type. ObjectSpaces provides such segmentation automatically without filtering because each object (whether a person or article) maintains a list of its interactions. As we showed earlier in the example in Section 6.4.1, every event identifies a person responsible for its detection. We go a step further by examining interactive relationships between groups of people. We recognize that individuals can have roles that influence how they interact with objects and other people in a process. Activities with **separable groups** are characterized by wholly independent interactive relationships between two or more agents, i.e., multiple speakers, but separate speeches. Conversely, **non-separable roles** occur when multiple agents take on collaborative, inter-dependent behavior, i.e., dialogue (or argument) when speakers talk at the same time concerning the same topic.

We need ways of assessing overall task interactions while preserving individual behaviors. Our approach is to divide activities into separable groups, then develop grammars that describe the non-separable interactions in each. In the card game of Blackjack[7] or "21," for example, a player's conduct is motivated by how the dealer is expected to behave. While there can be many players in a single game, each shares a similar yet independent relationship with the dealer. Since there is rarely any correlation between player interactions, each player-dealer pair represents a separable group. Interactions between player and dealer are

[7]For more information of the game of Blackjack, see Appendix B.

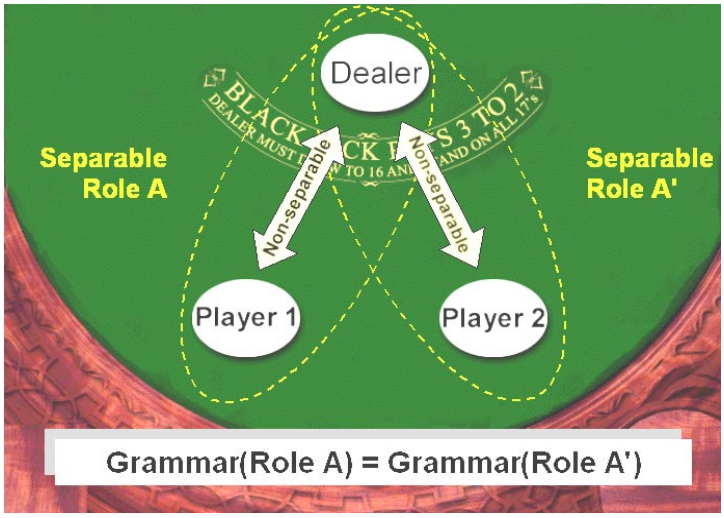


Figure 6.5: Each dealer-player group represents separable (*independent*) roles. Within each group, individual roles are non-separable (*dependent*) and share the same grammar.

non-separable. Consequently, with only one type of non-separable role to characterize, we can capture the exchanges in the activity using a singular grammar. See Figure 6.5 for an illustration. The production rules for this grammar are listed in Table 6.3. Terminal symbols used in alphabet V_{21} are based on primitive events detected. This grammar generates a language that can describe the role between any deal-player couple.

In our framework, the scene layer monitors all interactions taking place and assists in the labeling of detected events, as mentioned earlier. Each of the m person objects in $\mathbf{p} = \{p_1, p_2, ...p_m\}$ maintains a separate symbolic string that represents interactive events in which it participated. In our case, the relation between any event and person object is established by two measures: a) the person in contact with an article, and b) the "owner" of the article. These tags are important in Blackjack because they help the scene object associate an article with a respective player. Moreover, the tags remove potential ambiguity that can confuse the parser and, consequently, lead to cloudy semantic expressions. The first measure is easily established when we detect an overlap between the regions bounding the hand and the object. The second measure is largely determined by proximity zones $\mathbf{Z}_p$ placed around each person. The boundary zone $\mathbf{z}_{p_i} = [x_l \ y_t \ x_r \ y_b]^T$ for each person is defined manually when the scene is initialized, then labeled with the same ID as the

Production Rules				Description
S	$\rightarrow$	AB	[1.0]	Blackjack $\rightarrow$ "play game" "determine winner"
A	$\rightarrow$	CD	[1.0]	play game $\rightarrow$ "setup game" "implement strategy"
B	$\rightarrow$	EF	[1.0]	determine winner $\rightarrow$ "evaluate strategy" "cleanup"
C	$\rightarrow$	HI	[1.0]	setup game $\rightarrow$ "place bets" "deal card pairs"
D	$\rightarrow$	GK	[1.0]	implement strategy $\rightarrow$ "player strategy"
E	$\rightarrow$	LKM	[0.6]	evaluate strategy $\rightarrow$ "flip dlr down-card" "dlr hits" "flip plyr down-card"
	$\rightarrow$	LM	[0.4]	evaluate strategy $\rightarrow$ "flip dealer down-card" "flip player down-card"
F	$\rightarrow$	NO	[0.5]	cleanup $\rightarrow$ "settle bet" "recover card"
	$\rightarrow$	ON	[0.5]	$\rightarrow$ "recover card" "settle bet"
G	$\rightarrow$	J	[0.8]	player strategy $\rightarrow$ "Basic Strategy"
	$\rightarrow$	Hf	[0.1]	$\rightarrow$ "Splitting Pair"
	$\rightarrow$	$bfffH$	[0.1]	$\rightarrow$ "Doubling Down"
H	$\rightarrow$	l	[0.5]	place bets
	$\rightarrow$	lH	[0.5]	
I	$\rightarrow$	ffI	[0.5]	deal card pairs
	$\rightarrow$	ee	[0.5]	
J	$\rightarrow$	f	[0.8]	Basic strategy
	$\rightarrow$	fJ	[0.2]	
K	$\rightarrow$	e	[0.6]	house hits
	$\rightarrow$	eK	[0.4]	
L	$\rightarrow$	ae	[1.0]	Dealer downcard
M	$\rightarrow$	dh	[1.0]	Player downcard
N	$\rightarrow$	k	[0.16]	settle bet
	$\rightarrow$	kN	[0.16]	
	$\rightarrow$	j	[0.16]	
	$\rightarrow$	jN	[0.16]	
	$\rightarrow$	i	[0.18]	
	$\rightarrow$	iN	[0.18]	
O	$\rightarrow$	a	[0.25]	recover card
	$\rightarrow$	aO	[0.25]	
	$\rightarrow$	b	[0.25]	
	$\rightarrow$	bO	[0.25]	

Symbol	Domain-Specific Events
a	dealer removed card from house
b	dealer removed card from player
c	player removed card from house
d	player removed card from player
e	dealer added card to house
f	dealer dealt card to player
g	player added card to house
h	player added card to player
i	dealer removed chip
j	player removed chip
k	dealer pays player chip
l	player bets chip

Table 6.3: SCFG G_{21} for Blackjack/"21" card game: Production rules, probabilities, and descriptions. Detectable domain-specific events make up the terminal alphabet V_T of G_{21}.

$0+1+2$	$l_{10}l_{20}l_{20}l_{10}l_{10}f_{01}f_{01}l_{20}f_{02}f_{02}e_{00}e_{00}f_{01}a_{00}e_{00}d_{11}d_{22}h_{11}h_{22}k_{02}k_{02}j_{20}j_{20}k_{02}\ldots$ $j_{20}j_{20}i_{00}j_{20}i_{00}j_{20}i_{00}a_{00}a_{00}b_{01}b_{01}b_{01}b_{02}b_{02}b_{02}$
$0+1$	$l_{10}l_{10}l_{10}f_{01}f_{01}e_{00}e_{00}f_{01}a_{00}e_{00}d_{11}h_{11}i_{00}i_{00}i_{00}a_{00}a_{00}b_{01}b_{01}b_{01}$
$0+2$	$l_{20}l_{20}l_{20}f_{02}f_{02}e_{00}e_{00}a_{00}e_{00}d_{22}h_{22}k_{02}k_{02}j_{20}j_{20}k_{02}j_{20}j_{20}i_{00}j_{20}i_{00}j_{20}i_{00}a_{00}\ldots$ $a_{00}b_{02}b_{02}b_{02}$

Table 6.4: Example strings from a game of Blackjack. Subscripts on each terminal denotes i) ID of person making contact with object, and ii) ID of owner of object, respectively. ID numbers: dealer(0), player A(1), and player B(2).

respective person object. During initialization, we also have the option of establishing the role for each person, i.e., labeling each as a dealer or a player. Objects that are placed in these zones inherit the same ID as the zone. These tags are attached during the scanning stage when the next state is added, such that

$$i+1: \ {}_kX \rightarrow \lambda a.\mu \ [\alpha, \gamma, p_j, o(\mathbf{z}_{p_i})],$$

where $o(\mathbf{z}_{p_i})$ returns the ID p_j corresponding to the zone defined by $o(\mathbf{z}_{p_i})$. Table 6.4 provides an actual example of strings from a game of Blackjack with a dealer (ID $\equiv 0$) and two players (IDs 1 and 2). For emphasis, the terminals are labeled with subscripts that indicate the two measures mentioned above. The first row shows the single, interlaced string involving all agents (IDs 0+1+2), which would be ambiguous to parse, even for the human observer, were it not for the subscript tags. Instead of wrestling with this arduous string, ObjectSpaces extracts separable groups from this string representing the dealer-player pairs (0+1) and (0+2). Only terminals with at least one subscript that matches an ID in the grouping is retained in the string.

Separable groups are not restricted to pairwise combinations of non-separable roles, but to any number that can be efficiently specified by grammar. For example, in the popular card game of *Spades*, which typically involves four players, there are no separable groups because the strategy of each player is inextricably tied to the interactions of the other players. Therefore, the required grammar must attempt to explain all four non-separable roles. Conversely, when the number of separable groups equals the number of participants in the activity, an appropriate grammar (not necessarily unique) is needed for

each independent agent. Naturally, we are motivated to consider separable activities that limit the dimension of non-separable roles.

Exploiting separability allows us to assess the probabilistic behavior of individuals in the scene by isolating events that occur within a non-separable relationship. To model a particular behavior, we manually select a subset of production rules, i.e., $\mathcal{P}_\varsigma \in \mathcal{P}$, that offer a basis for characterizing interactions. Here $\mathbf{b}_\varsigma$ is a vector that represents all n production probabilities in subset $\mathcal{P}_\varsigma$. Each person object maintains a unique set of production likelihoods $\hat{\mathbf{b}}_\varsigma$, which are reset initially to reflect a uniform distribution[8]. Using Equation D.1, rule probabilities for each individual are "tuned" based on observations of selected productions over the course of several trials. For example in Blackjack, the use of certain strategies designed to improve the odds of winning are more likely to be used by a more experience player versus a novice. Comparisons between individually tuned rule probabilities $\hat{\mathbf{b}}_\varsigma$ and pre-trained models $\mathbf{b}_\varsigma$ can be made using the mean sum of the square differences or mean square error, i.e.,

$$err(\mathbf{b}_\varsigma - \hat{\mathbf{b}}_\varsigma) = \frac{1}{n}\sum_{i=1}^{n}(\beta_i - \hat{\beta}_i)^2. \tag{6.19}$$

Values in $\mathbf{b}_\varsigma$ are determined from training data data and represent our established models of particular behavior. The MSE is used to measure the pairwise distance so that the likelihood of a behavior given a model for it can be established by

$$P(\hat{\mathbf{b}}_\varsigma|\mathbf{b}_\varsigma) = 1 - \sqrt{err(\mathbf{b}_\varsigma - \hat{\mathbf{b}}_\varsigma)}. \tag{6.20}$$

Using individually tuned grammars, production probabilities are assessed to disambiguate player behavior. As the number of trials increases, we expect better characterization of behavior.

6.5.3 Error Detection & Recovery

Another presumption that we have maintained throughout our discussion of grammar has been the expectation of a syntactically agreeable input string. Naturally, there are several

[8]For a nonterminal X that generates n other strings of terminals and nonterminals, i.e., $X \rightarrow \mu_1|\mu_2|\ldots|\mu_n$, all respective likelihoods $\mathbf{b}_X = \beta_1, \beta_2, \ldots, \beta_n$ are set identically to $\frac{1}{n}$. During separable role characterization, each individual shares the same initial set of rule likelihoods $\mathbf{b}_X$ over $\mathcal{P}_\varsigma$.

cases where ungrammatical input may be encountered. A **substitution error** occurs when the wrong terminal symbol is generated because the actual event is not detected as the most likely. **Insertion errors** take place when spurious symbols that do not correspond to actual events are added to the input. Finally, **deletion errors** are the result of failures in detecting events that actually occurred.

Domain-specific detectors often benefit from empirically derived methods which help to improve their ability to measure anticipated changes in the environment. As a result of using such heuristic models to detect meaningful events, substitution and insertion errors are rare. However, it is challenging to detect events that deviate significantly from these same heuristic models. The trade off is generally a higher rate of deletion errors. Ivanov and Bobick handle substitution and insertion errors by employing multi-valued strings and by modifying the grammar so that it accepts input that could, otherwise, terminate parsing [75]. However, for rule-based activities, any attempt at correcting an error compromises the benefit of actually detecting when a rule is violated. In fact, we have a vested interest in determining how, when, where, and by whom errors occur. At the same time, we wish to make parsing robust enough to tolerate erroneous input.

While no treatment for correcting errors is suggested, we attempt to recover from parsing failures by taking advantage of grammatical structure. Although the arrangement of terminals in the input is non-deterministic, it is constrained by *a priori* known rules that we leverage to anticipate future input. Parsing errors occur in the scanning stage when the symbol sampled from the input does not match any of the terminals from the prediction stage. This invariably happens during a nonterminal expansion. We revisit the prediction stage during the expansion of nonterminal X, which creates a list of productions $Y \rightarrow \nu$ that are syntactically consistent with the expansion, i.e.,

$$i : {}_{k}X \rightarrow \lambda . Y \mu \; [\alpha, \gamma] \;\; \Rightarrow \;\; i : {}_{i}Y \rightarrow . \nu \; [\alpha^{'}, \gamma^{'}].$$

Every nonterminal Y is also expanded until the next terminal is predicted, i.e.,

$$i : {}_{i}Y \rightarrow . a \xi \; .$$

Solutions to a parsing failure are motivated by the nature of the error. We offer three scenarios below:

- *If the failure is caused by an insertion error*, we simply ignore the scanned terminal, and return the state of the parser to the point prior to scanning. The same pending expansions for prediction are maintained.

- *If the failure is caused by a substitution error*, we promote each pending prediction state as if it were actually scanned, creating a new path for each hypothetical terminal. (At this point, there are no real paths). We proceed with normal parsing, but instead of maintaining paths that spawn from a single scanned terminal, we accommodate all paths from each hypothetical terminal appearing as a result of a simulated scan. A hypothetical path is terminated if another failure occurs in the next real scanning step. The actual likelihood of the event associated with the hypothetical terminal $P_{\mathbf{D}}(a)$ is recovered, then multiplied to prediction values of α and γ such that

$$\alpha^{'} = \alpha(i : {}_{i}Y \rightarrow .a\xi)P_{\mathbf{D}}(a) \tag{6.21}$$

$$\gamma^{'} = \gamma(i : {}_{i}Y \rightarrow .a\xi)P_{\mathbf{D}}(a), \tag{6.22}$$

  as before.

- *If the failure is caused by a deletion error*, again we promote each pending prediction state and create a separate path for each hypothetical symbol. We proceed through the completion stage, then to prediction to generate the next state terminal. During a simulated scan, hypothetical paths that are inconsistent with the symbol that caused the first failure are terminated. When a deletion error is assumed, there is no detection likelihood to recover for the missing symbol. We approximate this likelihood, denoted as $\widetilde{P}_{\mathbf{D}}(a)$, using empirical values that we select by hand, which are influenced by historical probability values for the detection of symbol a. Modified forward and inner probabilities in the first scanning step are given as

$$\alpha^{'} = \alpha(i : {}_{i}Y \rightarrow .a\xi)\widetilde{P}_{\mathbf{D}}(a) \tag{6.23}$$

$$\gamma^{'} = \gamma(i : {}_{i}Y \rightarrow .a\xi)\widetilde{P}_{\mathbf{D}}(a), \tag{6.24}$$

  while those of the second simulated scanning step can be recovered from the original scan likelihood, i.e.,

$$\alpha^{'} = \alpha(i+1 : {}_{i+1}Y \rightarrow .b\xi)P_{\mathbf{D}}(b) \tag{6.25}$$

$$\gamma' \quad = \quad \gamma(i+1 : \ {}_{i+1}Y \rightarrow .b\xi) P_{\mathbf{D}}(b). \tag{6.26}$$

Using these methods, the parser is guaranteed to generate a syntactically legal interpretation but provides no warranty of its semantic legitimacy. The parser supplies the framework with the erroneous symbol and its corresponding likelihood so that records of potential failures can be attached to the appropriate person object. In this way, substrings with bad syntax can be more closely scrutinized to determine when an illegal action takes place.

Table 6.5 illustrates how the parser attempts to recover from failures using the three error scenarios mentioned above. We maintain every recovered path, even if multiple tracks (each representing one of the three error scenarios) are formed from a single failure. For each of the error scenarios, we tolerate only two consecutive failures before terminating the parse of that path. However, our approach can be applied iteratively so that more consecutive failures can be tolerated. A consequence of accepting more failures must be reflected by increasing the uncertainty in our approximation of $P_{\mathbf{D}}(a)$, denoted as $\widehat{P}_{\mathbf{D}}(a)$. We rely on the exponential to serve as a penalty function that is applied by multiplication to the historical mean likelihood $\widetilde{P}_{\mathbf{D}}(a)$, i.e.,

$$\widehat{P}_{\mathbf{D}}(a) = e^{\frac{-n}{\rho}} \widetilde{P}_{\mathbf{D}}(a), \tag{6.27}$$

where n is the number of consecutive failures and ρ is empirically derived.

The algorithmic complexity of tracking multiple paths, which is a function of the production rules involved, tends to grow linearly for grammars with small terminal and nonterminal alphabets but can expand exponentially for larger grammars or for very long terminal strings. When computation and memory resources must be delicately managed, we prune recovered paths that have low overall likelihoods. Unlike the example provided in Table 6.5, we can also entertain hybrid error scenarios in order to generate the most likely parse, i.e., instead of treating each consecutive error by the same type of scenario, all three alternatives can be consider for each bad input symbol.

6.5.4 Parsing with Adaptive Grammar

As we have already seen, the parser makes the most likely parse of the input based on the prior input received. In essence, the parser offers the most likely interpretation of a

(a)

Grammar

S	$\rightarrow$	AB
A	$\rightarrow$	aa
	$\rightarrow$	aaA
B	$\rightarrow$	bc
	$\rightarrow$	bcB

(b)

Earley Chart

predicted

$0:\ {}_0 \rightarrow .S$
$0:\ {}_0S \rightarrow .AB$
$0:\ {}_0A \rightarrow .aa$
$0:\ {}_0A \rightarrow .aaA$

scanned "a"

$1:\ {}_0A \rightarrow a.a$
$1:\ {}_0A \rightarrow a.aA$

none completed

predicted

$1:\ {}_1A \rightarrow a.a$
$1:\ {}_1A \rightarrow a.aA$

(c)

Insertion

scanned "b"
failure - expecting "a"
ignore "b"
predicted
$1:\ {}_1A \rightarrow a.a$
$1:\ {}_1A \rightarrow a.aA$
scanned "c"
2^{nd} failure - expecting "a"
TERMINATED

Substitution

scanned "b"
failure - expecting "a"
*scanned "a"
$2:\ {}_1A \rightarrow aa.$
$2:\ {}_1A \rightarrow aa.A$
completed
$2:\ {}_1A \rightarrow aa.$
$2:\ {}_0S \rightarrow A.B$
predict
$2:\ {}_2A \rightarrow .aa$
$2:\ {}_2A \rightarrow .aaA$
$2:\ {}_2B \rightarrow .bc$
$2:\ {}_2B \rightarrow .bcB$
scanned "c"
2^{nd} failure - expecting "b"
TERMINATED

Deletion

scanned "b"
failure - expecting "a"
*scanned "a"
$2:\ {}_1A \rightarrow aa.$
$2:\ {}_1A \rightarrow aa.A$
completed
$2:\ {}_1A \rightarrow aa.$
$2:\ {}_0S \rightarrow A.B$
predict
$2:\ {}_2A \rightarrow .aa$
$2:\ {}_2A \rightarrow .aaA$
$2:\ {}_2A \rightarrow .bc$
$2:\ {}_2A \rightarrow .bcB$
*scanned "b" (retry)
$3:\ {}_2A \rightarrow b.c$
$3:\ {}_2A \rightarrow b.cB$
none completed
predicted
$3:\ {}_3A \rightarrow b.c$
$3:\ {}_3A \rightarrow b.cB$
scanned "c"
$3:\ {}_2A \rightarrow b.c$
$3:\ {}_2A \rightarrow b.cB$
$3:\ {}_2A \rightarrow b.cB$

Table 6.5: a) Simple CFG grammar. A deletion error occurs in the detection of events $aabc\ldots$; input only contains $abc\ldots$ b) Shows the Earley chart after the first symbol is scanned. The next scanned symbol, b, will cause parsing to fail under normal conditions. c) Continuation of Earley Chart shows parser recovery attempts under different error assumptions. *Scan of hypothetical symbol is simulated to promote parsing step.

sequence of events based on measurable evidence. We look to exploit this evidence along with the grammatical structure of an activity to improve our expectation of future input.

By identifying rules that are explicitly correlated and by observing prior productions, we can generate a reliable forecast of events that are likely to appear in the input. We modify the likelihood of these anticipated production rules to reflect our confidence in an activity's topology. This is equivalent to placing "if-then" branches in the structure so that we can tailor productions for specific scenarios encountered in the input. Our approach is motivated by the work of Langley, who demonstrates how an agent can learn and anticipate grammatical expressions over time by adjusting the rule probabilities to maximize recognition [84].

In many rule-based activities, certain events typically repeat a number of times. For instance, if a Blackjack player wishes to bet \$5 (using \$1 chips), the event "player bets chip" will occur *five* times. While the number of repeats is initially arbitrary, this number becomes a potentially important indicator of how many times we can expect to see future events, i.e., if the house wins, then it will collect exactly *five* chips from the player. Recall that a context-free grammar can form the language $L(G_{free}) = L_f : a^n b^m \ldots$, where correlations between n and m can be counted. However, these correlations have to be explicitly defined by rules in the grammar. For example, to recognize a string containing exactly three occurrences of event a, i.e., $aaa = a^3$, the production $X \rightarrow a^3$ must be explicitly defined in the production set $\mathcal{P}$.

To describe the sequential repetition of event a, which can loop arbitrarily many times, we employ productions of the form

$$\begin{aligned} A &\rightarrow aA \\ &\rightarrow a \ . \end{aligned}$$

We record the state k where terminal a is first predicted in the expansion of A, i.e.,

$$\begin{aligned} k: \ {}_kX &\rightarrow \lambda . A\mu \\ k: \ {}_kA &\rightarrow .a \\ k: \ {}_kA &\rightarrow .aA \end{aligned}$$

and allow parsing to proceed as usual. After the final symbol a has been scanned and state $i: \ {}_kX \rightarrow \lambda A.\mu$ is completed, the number of times A generates a is given by $n_A = i - k$. In

other words, n_A is equal to the length of the substring containing all of the a symbols. The mean likelihood of detecting the substring $a^{(n_A)}$ is computed using Equation 6.10 as

$$\widetilde{P}_{\mathbf{D}}(a^{(n_A)}) = \frac{1}{n_A} \sum_{j=k+1}^{i} P_{\mathbf{D}}(x_j), \tag{6.28}$$

where x is the input string and j covers all states sets where input scanning recovered terminal a during the generation of A.

To make parsing more context sensitive, we exploit the semantic relation between the high-level event represented by A and another high-level, recursive event given by B, i.e.,

$$\begin{array}{rcll} B & \rightarrow & bB & [\beta_1] \\ & \rightarrow & b & [\beta_2] \\ & & \vdots & \end{array},$$

where β_1 and β_2 represent $P(B \rightarrow bB)$ and $P(B \rightarrow b)$, respectively. Notice that B can count any number of events b from the input as well as generate other strings. However, since A influences B, the expansion of B should generate a specific terminal string rather than one of arbitrary length. The most probable number of events n_B is defined by a relation function, i.e., $n_B = f_{A \mapsto B}(n_A)$, using task rules and semantics. This heuristic function provides a mapping between the number of loops in A and the number of loops in B and is specified manually when the production rules are designed. To increase the likelihood of the correct parse, we add the stochastic production $B \rightarrow b^{(n_B)}$ $[\widetilde{P}_{\mathbf{D}}(a^{(n_A)})]$ to $\mathcal{P}$ and decrement the probabilities of other rules generated by B. Because we rely on unambiguous rules to dictate the mapping between A and B, the likelihood of expected production $B \rightarrow b^{(n_B)}$ should be unity. However, we adjust this value to reflect our confidence in the detected string $a^{(n_A)}$. When determining the likelihood of $b^{(n_B)}$, we ignore the syntactical likelihood of $a^{(n_A)}$, which is given by the inner probability $\gamma({}_kX \rightarrow \lambda A.\mu)$. We replace B with $\widehat{B}$, which is written as

$$\begin{array}{rcll} \widehat{B} & \rightarrow & b\widehat{B} & [\hat{\beta}_1] \\ & \rightarrow & b & [\hat{\beta}_2] \\ & \rightarrow & b^{(n_B)} & [\widetilde{P}_{\mathbf{D}}(a^{(n_A)})] \\ & & \vdots & \end{array},$$

where $\hat{\beta}_1 = \beta_1(1-\widetilde{P}_{\mathbf{D}}(a^{(n_A)}))$ and $\hat{\beta}_2 = \beta_2(1-\widetilde{P}_{\mathbf{D}}(a^{(n_A)}))$. As usual, all production probabilities of $\widehat{B}$ sum to unity. Because this genre of production does not involve unit productions or left recursions, there is no need to recalculate R_U and R_L. Modified nonterminal $\widehat{B}$ persists in $\mathcal{P}$ until the entire string is parsed before reverting back to B.

In a similar fashion, we generalize the treatment used to handle sequential repetitions so that various other types of dependencies between high-level nonterminals can be accepted. Without loss of generality, we allow the mapping between Y and Z, denoted $Y \mapsto Z$, which results in some new or pre-existing production $Z \rightarrow \nu$ with corresponding probability $\widetilde{P}_{\mathbf{D}}(w)$. Here, w represents a terminal substring generated by the production in Y. As before, the modified nonterminal $\widehat{Z}$ replaces Z and the probabilities of other productions are normalized accordingly. In cases where the production $Z \rightarrow \nu$ already exists in $\mathcal{P}$, we simply modify the production probabilities as we have demonstrated above.

Multiple semantic dependencies among productions in the grammar result in mappings that are not necessarily unique. The set of all v nonterminals that influence, or map to Z is given by $V_{\mapsto Z} = \{Y_1, Y_2, \ldots, Y_v\}$. Each nonterminal Y_i generates some string w_i which invokes the production $Z \rightarrow \eta_i \ [\widetilde{P}_{\mathbf{D}}(w_i)]$ in $\mathcal{P}$. Here the subscript i denotes the i^{th} nonteminal Y_i. Considering the other u independent productions of Z, i.e., $Z \rightarrow \alpha_i \ [\beta_i]$, the modified nonterminal $\widehat{Z}$ is given by

$$\begin{array}{rcll} \widehat{Z} & \rightarrow & \eta_1 & [\tau\widetilde{P}_{\mathbf{D}}(w_1)] \\ & \rightarrow & \eta_2 & [\tau\widetilde{P}_{\mathbf{D}}(w_2)] \\ & & \vdots & \\ & \rightarrow & \eta_v & [\tau\widetilde{P}_{\mathbf{D}}(w_v)] \\ & \rightarrow & \alpha_1 & [\hat{\beta}_1] \\ & \rightarrow & \alpha_2 & [\hat{\beta}_2] \\ & & \vdots & \\ & \rightarrow & \alpha_u & [\hat{\beta}_u] \end{array} , \tag{6.29}$$

where $\tau = \frac{1}{v}$ and $\hat{\beta}_i = \beta_i(1-\tau) \ \forall \ 1 \leq i \leq u$. To distribute likelihood appropriately among the productions, the probability of each dependent production is in proportion to its detected likelihood while the probability of each independent production is proportional to its original albeit diminished likelihood.

We avoid adapting productions that do not have a strictly dependent relationship with other rules. For example, if productions with nonterminal B are influenced by those of A, as well as by several other nonterminals, adjusting its probability can erode likelihoods based on detectibility versus based on observation. Moreover, some adaptation may require recompilation of the R_L and R_U matrices[9].

6.6 Experimental Results

In this section, we provide real examples of our approach in the domain of the card game, Blackjack. One of the motivations for using SCFG to specify this activity is the convenience by which a rule of the game can be used, virtually unmolested, as a production rule in the grammar (a list of the rules of Blackjack/"21" is available in Appendix B). We examine card games as case studies for understanding multitasked activities because they involve processes that are dependent within the context of each player but are wholly separate within the context of the game, i.e., contain separable groups and non-separable roles. They involve multiple people, objects, actions, and player strategies that are difficult to model accurately using strictly stochastic or deterministic approaches.

In general, the card game domain has rich semantics, several activities that can be reasonably specified using grammar, and primitive events that can be measured using computer vision. Moreover, ground-truth observations for measuring accuracy are also readily available for these activities. By closely examining multitasked, collaborative tasks such as card games, we develop methods that are appropriate for treating other highly complicated human experiences.

All experiments conducted in this section were implemented using the Vision Action Recognition System[10] (VARS). Each activity sequence we mention below is a 1/4 frame, color YUV video sequence of an entire Blackjack game. Recall Table 6.3 which shows the stochastic grammar G_{21} used to model Blackjack.

[9] For all but the most sparsely populated matrices, recalculation of these matrices in nontrival.
[10] To reference more information about VARS, see Appendix C

Symbol	Domain-Specific Events	Detect Rate	Error Rate		
			Insert	*Sub*	*Del*
a	dealer removed house card	100.0%	0.0%	0.0%	0.0%
b	dealer removed player card	100.0%	0.0%	0.0%	0.0%
c	player removed house card†	100.0%	0.0%	0.0%	0.0%
d	player removed player card	100.0%	0.0%	0.0%	0.0%
e	dealer added card to house	94.6%	0.0%	0.0%	5.4%
f	dealer dealt card to player	92.2%	0.0%	0.0%	7.8%
g	player added card to house†	100.0%	0.0%	0.0%	0.0%
h	player added card to player	89.3%	3.6%	0.0%	7.1%
i	dealer removed chip	93.7%	0.0%	0.0%	6.3%
j	player removed chip	96.9%	0.0%	0.0%	3.1%
k	dealer pays player chip	96.9%	0.0%	0.0%	3.1%
l	player bet a chip	90.5%	1.1%	1.1%	7.4%

Table 6.6: *Experiment V*: Detection rate of domain-specific events which make up the terminal alphabet V_T of G_{21}. Errors are categorized as insertion, substitution, and deletion, respectively. †Denotes events with no significance to legitimate Blackjack play, but can be used to detect illegal occurrences.

6.6.1 Experiment V: Low-level Event Detection

In this experiment, we are interested in determining the accuracy of terminal symbol generators, which use image- and object-based evidence as well as heuristics to model detectable events. No action-based evidence was used to classify objects or detect events. An action-based gesture[11] was attempted, however, but could not be consistently recognized due to limitations in our ability to characterize and distinguish the action in the presence of motion noise and meaningless, random hand movement.

Twenty-eight sequences were used to generate 700 example events, which were compiled to determine the detection rate of each detector. Each sequence consisted of a full game of Blackjack with at least one player. For example, a sequence might generate six examples of the event "player bet a chip," five examples of "dealer removed player card," etc. The overall detection rate for all events is 96.2%. The error rates, which were assessed manually, for insertion, substitution, and deletion errors were 0.4%, 0.1%, and 3.4%, respectively. As expected, deletion errors dominate primarily because several newly introduced objects could not be classified. For example, attempts to classify an unknown as a card

[11] In Blackjack, a player can request an additional card by tapping on the table with his/herhand.

Symbol	Detect Rate		Insert Err		Sub Err		Del Err	
	on	*off*	*on*	*off*	*on*	*off*	*on*	*off*
a	98.8%	92.5%	0.0%	0.0%	0.0%	0.0%	1.2%	7.5%
b	97.8%	90.8%	0.0%	0.0%	0.0%	0.0%	2.2%	9.2%
c	100.0%	80.0%	0.0%	0.0%	0.0%	20.0%	0.0%	0.0%
d	100.0%	91.7%	0.0%	0.0%	0.0%	0.0%	0.0%	8.3%
e	94.0%	74.9%	1.2%	5.0%	1.2%	7.5%	3.6%	12.5%
f	95.6%	70.3%	0.0%	2.3%	0.0%	9.2%	4.4%	18.3%
g	100.0%	50.0%	0.0%	0.0%	0.0%	50.0%	0.0%	0.0%
h	80.0%	41.7%	4.0%	8.3%	8.0%	33.3%	8.0%	16.7%
i	92.9%	88.9%	0.0%	0.0%	0.0%	0.0%	7.1%	11.1%
j	96.5%	92.7%	0.0%	0.0%	0.0%	0.0%	3.5%	7.3%
k	79.0%	12.5%	10.5%	36.5%	10.5%	43.8%	0.0%	7.3%
l	90.6%	55.8%	4.7%	17.2%	2.4%	9.8%	2.4%	17.2%

Table 6.7: *Experiment V*: Detection and error rates for Corpus A with error recovery turned on and off. Error recovery improves overall detection rate by 33.8%.

may fail if there is any overlapped between the card and neighboring cards or objects. To improve detection and classification of unknowns, we control the environment by using a solid, colored surface as the Blackjack table. Table 6.6 shows the results of this examination.

6.6.2 Experiment VI: Error Detection & Recovery

The Earley parser is complete, which means that it generates all possible derivations of a string[132]. When combined with a grammar of highly constrained, rule-based activities, the parser is guaranteed to parse an input string, barring semantic violations or errors. As expected, when a semantically legitimate sequence is presented to VARS with no insertion, substitution, or deletion errors, we are able to parse the activity with 100% accuracy.

To provide a more diverse sample of sequences, two testing corpa were compiled from several minutes of video where Blackjack was played, primarily with two people (one dealer and one player). Corpus A contained 28 legitimate sequences with at least one detection error per sequence (either a deletion, substitution, or insertion error). Corpus B represents a family of 10 illegitimate sequences with various errors. Sequences in Corpus B often contained illegal moves, dealer mistakes, cheating, etc. With error recovery disabled, only 12 of the 28 sequences (42.9%) in Corpus A could be entirely parsed without terminating in failure. Of those that could be parsed, the mean detection rate for 320 events was 70.1%.

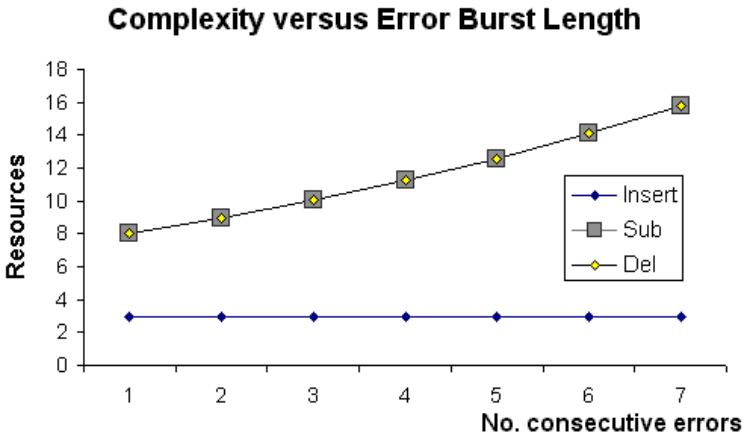


Figure 6.6: Using Corpus C, the complexity versus the length of the error burst is illustrated. Here complexity refers to the amount of system resources used, including computation and memory.

The error rates for insertion, substitution, and deletion errors were 5.8%, 14.5%, and 9.6%, respectively. None of the sequences in Corpus B could be entirely parsed. With error recover enabled (accepting up to 2 consecutive failures), 25 of the 28 sequences (85.7%) from Corpus A could be parsed with 93.8% of all 625 events detected accurately. Insertion, substitution, and deletion errors were reduced by 70.5%, 87.3%, 71.9%, respectively. Parsing improved by 40% for Corpus B sequences with error recovery turned on, with an average 85.4% of high-level events recognized accurately. This improvement in the parsing rate is attributable to our ability to recover from insertion errors, which simply skipped over rule violations encountered during the sequence. We assessed that 22.5%, 17.5%, and 60.0% of errors were caused by insertion, substitution, and deletion errors. respectively.

To provide more exhaustive testing of the parser, a third testing corpus was developed from 113 simulated terminal strings representing legal plays with various errors. Using simulated data, the probability of detection for each event $P(D_i)$ is estimated using the average determined in Table 6.6. We justify the use of simulated data because the parser only needs a string to generate the most likely derivation. In other words, it is of no consequence to the parser if the string is generated from real video or simulated. Using Corpus C, we can measure the performance of the parser under a variety of conditions, including consecutive error bursts. Figure 6.6 shows complexity versus the length of a homogeneous error burst. Here we define complexity as a measure of resources required in

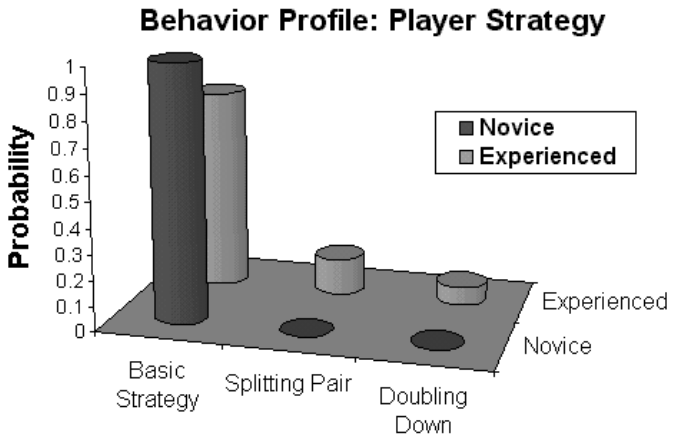


Figure 6.7: Trained behavior profiles of player strategy for novice and expert.

the computation of the algorithm, which include the number of pending states, memory, etc. Homogeneous error types present the worst-case system complexity due to contiguous blocks of substitution and deletion errors. Heterogeneous error scenarios benefit from the treatment used for insertion errors, which only need to maintain the same set of pending states, effectively lowering overall system complexity. We also learn empirically that to recover from an error burst of length n, we must accept at least n consecutive failures to recover.

6.6.3 Experiment VII: High-level Behavior Assessment

We examine non-separable roles between a player and the dealer to assess patterns of behavior. The conduct of the dealer is strictly regulated by the rules of Blackjack, but the player is permitted to execute a range of different strategies to improve his/her chance of winning. We define a *novice* as a player whose moves are limited to *basic strategy*[12] where *experts* employ more advanced strategies, such as "splitting pairs" and "doubling down." The profiles for these two behaviors is shown in Figure 6.7.

We can also assess other behaviors, such as whether a player is a low-risk or high-risk player, using heuristically motivated measures. After tuning several behaviors with actual and synthetic training data, roughly 10 trials per individual were conducted to assess their

[12]When no extra cards are dealt to the player after the initial card pair, basic strategy is assumed.

behavior. Results are posted in the chart below:

Behavior	Detection Accuracy
Low-risk	92%
High-risk	76%
Novice	100%
Expert	90%

6.6.4 Experiment VIII: Adaptive Grammar

We conducted experiments to assess the benefit of using adaptive grammar under a variety of conditions. Adaptive grammar exploits structure to enhance the probability of the most likely path. We use simulated data to examine the likelihood of the most likely parse of 200 sequences. When considering error-free strings, the mean improvement of likelihood using adaptive grammar was 11.3%. However, if errors occur in symbols where adaptive grammar has artificially inflated the likelihood (and consequently diminished the likelihood of other productions), the benefit of adaptive grammar is minimized.

6.7 Comments

We have demonstrated that stochastic grammar is a very appropriate method for representing multitasked activities. A significant secondary benefit in this regard is the level of compression offered in a terminal string. A legitimate terminal string contains enough meaningful information to accurately represent a wide variety of interactions. For example, consider that the average game of Blackjack (with 1 dealer and 1 player) lasts for 64 seconds and generates 1687.5MB of data (assuming full framerate, uncompressed color video). Albeit not with the same fidelity, a 26-character terminal string offers a compelling alternative representation, with an astounding 5,507,076:1 compression ratio over the video sequence. With this level of efficiency, several hours of interaction can be captured and stored using a terminal string. Additionally, substrings that express interactive patterns can also be conveniently indexed and searched by virtually any conventional text parser or editor.

CHAPTER 7

Conclusions

In this dissertation, we presented methods for recognizing complex activities using vision. To handle data management and facilitate recognition processes, we have introduced a multi-layer, object-oriented framework. We argue that policies are required to organize, store, and distribute a wide range of dynamic, heterogeneous information in order to solve sophisticated vision problems. We have provided an intuitive architecture that decomposes the activity domain into objects and layers, which encourages reuse of data and models. Decomposing a complex recognition process also and systems for more efficient, flexible, and scaleable vision systems. The implementation of ObjectSpaces, which contains over 93,000 lines of C++ code, is the best demonstration that our framework can provide high-level recognition of complex activities in a real-time system.

Using our framework, we have been able to exploit the relationship between actions and objects to improve recognition of objects and single-tasked activities. Our novel contributions demonstrate that objects can be detected and classified through recognition of associated actions. Using image-, object-, and action-based evidence together, we show classification accuracy of 89.7% of 8 unknown objects. We have also shown that object context provides powerful clues about with actions to anticipate. Using the hidden Markov model, we recognize 46 different hand-based actions with mean recognition rate of 90.2%. Using Markov models to characterize single-tasked activities, we show the mean recognition accuracy of 14 activities to be 87.6%. These results were achieved using an adaptive Bayesian classifier that leverages the strongest available evidence over time.

For recognition in complex tasks, we concentrate on rule-based, event-driven multi-tasked activities. By developing a grammar to describe meaningful interactions between

non-separable roles, our implementation detects 96.2% of all events in Blackjack. Using simulated data of over 200 Blackjack games, we show that adapting stochastic context-free grammar can improve recognition accuracy by 11.3%. Our novel extensions to the Earley-Stolcke algorithm provides error detection and recovery of parsing failures. When techniques for handling insertion, deletion, and substitution errors are enabled (accepting up to 2 consecutive failures), 88.1% more sequences could be parsed without failing. Even with errors present in the sequences, 93.8% of all 625 events detected accurately and insertion, substitution, and deletion errors were reduced by 70.5%, 87.3%, 71.9%, respectively. We also demonstrated quantitative techniques to evaluate human behavior by exploiting production rule probabilities.

7.1 Summary of Contributions

A summary of our contributions include:

- a layered framework for decomposing recognition problems,
- base class templates for categorizing context information,
- object-based event handling and focus-of-attention,
- block-based image sampling (preprocessor to connected component labeling),
- a hand tracking scheme that takes advantage of environmental context to enhance linear predictions of future hand locations,
- person-independent recognition of hand actions using scene context and HMMs.
- an extension to appearance-based representations for recognition
  - object classification using action context, i.e., object classification based on how people appear to interact with the object,
  - action recognition using object context, i.e., action recognition based on an object's appearance

- adaptive Bayesian classification techniques to identify and weigh dynamic patterns in evidence over time.
- techniques for adapting grammar to improve recognition of structured activities,
- new parsing strategies that enable error detection and recovery in stochasitic context-free grammar, and
- methods of measuring behavior in non-separable roles using production rule scoring.

7.2 Caveats

Although the single-camera approach is appropriate most of the time, non-planar, complex motion can become ambiguous from one perspective. Although several scene articles have no actions associated with them or no hand-based motion can be adequately modeled, we have shown that exploiting the relationship between object pairs can be helpful for summarizing activities and discovering articles. Despite the scalability problems that can be caused by associating all of the possible actions with each object, we hope to limit these by working with well defined and constrained domains. Additionally, many interesting activity domains can not be adequately modeled using grammar because of insufficient structure or rules. We hope to investigate methods that provide recognition of events while relaxing the structural constraints.

7.3 Future Work

There are several opportunities to extend our present work into exciting new areas. Since recognition of multitasked activities is still very new, more training data from different domains would be valuable to develop a more extensive database of actions, articles, and models of high-level behavior. Specifically, more work should be done on developing adaptive grammar approaches that can accommodate less structured multitasked activities. In order for vision-based action recognition to enjoy commercial success and wide-spread implementation, real-time systems that can accommodate a large dictionary of both simple and complex gestures and actions must be created [40]. Additionally, these systems should

be flexible enough to learn new activities while preserving the ability to discriminate existing action families. For most applications, action recognition systems will need to maintain low error rates and user independence while recognizing continuous gestures and body actions.

In general, we have examined human interactions using only image information. A natural extension to this approach is to consider hybrid approaches that combine information from several heterogeneous sensors, such as audio or tactile. One of the biggest barriers that must be addressed in this regard is sensor fusion [127]. Building models that use data that can have non-linear temporal or dynamic characteristics can be challenging, especially if finite state representations are used. There are also more opportunities for confusion and conflict if sensor data is married together incorrectly. Additionally, future systems will likely incorporate face and speech recognition with hand gestures to fuse intelligent, perceptual systems. A great deal of work lies ahead to solve issues such as automatic selection of optimal feature extraction and tracking in the presence of challenging environmental conditions.

While there are huge opportunities to immerse vision-based technology in various environments, developers of vision systems must still address other issues such as privacy [90, 98] and deployment in mission-critical installations [114]. In today's information society, several controversies are brewing regarding the collection, processing, and dissemination of information collected about people, especially in the form of audio, images, and video [92]. A consequence of implementing any information-gathering technology is controlling its usage. While Lunheim and Sindre argue that "privacy is a cultural construct, rather than a genuine human need [87]," the acceptance of vision-based technologies is likely to come under fire if there is any likelihood than an individual's privacy will be compromised or breached all together. Moreover, it may be difficult, if not impossible, to form a communal consensus of how best to deploy such technologies, much less a universal one. Like other revolutionary inventions that preceded it, such as the television and telephone, vision will also have to establish a clear perceived benefit as well as a comfortable rapport before it can be accepted by the community at large [87]. Establishing this relationship goes beyond technological or political solutions and can be addressed only through increased awareness of cultural and personal privacy concerns.

7.4 Published Work

- Darnell J. Moore, Antai Peng, and Monson H. Hayes, "A Genetic Algorithm for Synthesizing Lip Movements from Speech," *Proceedings of the 4th Bayona Workshop on Intelligent Methods in Signal Processing and Communications*, Bayona-Vigo, Spain. June 24-26, 1996.

- Darnell J. Moore and Monson H. Hayes III, "Tracking 3D Position and Orientation from 2D Sequences Using Simple Geometry," *Proceedings of the 30th Asilomar Conference on Signals, Systems, and Computers,* Monterey, California, November 3-6, 1996.

- Darnell J. Moore, Irfan A. Essa, and Monson H. Hayes III, "ObjectSpaces: Context Management for Activity Recognition," *Proceedings of the 2nd Annual Audio-Visual Biometric Person Authentication Conference*, Washington, D. C., March, 1999.

- Darnell J. Moore, Irfan A. Essa, and Monson H. Hayes III, "Exploiting Human Actions and Object Context for Recognition Tasks ," *Proceedings of the 7th IEEE International Conference on Computer Vision*, Vol. 1, pp. 80-86, Corfu, Greece, September 20-26, 1999.

- Darnell J. Moore, Roy Want, Beverly Harrison, Anuj Gujar, Ken Fishkin, "Implementing PHICONS: Combining Computer Vision with InfraRed Technology for Interactive Physical Icons," *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology* (UIST Technote), Asheville, NC, November 7-10, 1999.

APPENDIX A

Three Key Problems of the Hidden Markov Model

A.1 Model Assumptions

Several assumptions are generally made to limit the number of free parameters for the sake of mathematical and computational tractability. However, if these conditions are violated, the model representation can be weakened.

The Markov Assumption : The Markov assumption purports that the next state is dependent only upon the current state, resulting in a first order HMM. This assumption is implicit in Equation 4.21 since the probability of being in state S_j at time $t+1$ depends only on the state s_i at time t. It is also possible, however, to allow the next state to depend on the past k states, which specifies a k^{th} order HMM given by

$$a_{i_1 i_2 ... i_k j} = P(q_{t+1} = s_j | q_t = s_{i_1}, q_{t-1} = s_{i_2}, ..., q_{t-k+1} = s_{i_k}),\ 1 \leq i_1, i_2, ...i_k, j \leq N \tag{A.1}$$

Although first order HMMs are generally appropriate, higher order HMMs can better characterize some actions, but at the added expense of more complexity.

The Stationarity Assumption : The stationarity assumption considers state transition probabilities to be independent of the actual time at which the transition takes place. Mathematically, we express this as

$$P(q_{t_1+1} = s_j | q_{t_1} = s_i) = P(q_{t_2+1} = s_j | q_{t_2} = s_i),\ \forall\ t_1, t_2. \tag{A.2}$$

The Output Independence Assumption : The output independence assumption states that the current output observation is statistically independent of the previous output

observation. This can be formulated mathematically for a sequence of observations $\mathbf{O}$ as

$$P(\mathbf{O}|q_1, q_2, ..., q_T, \lambda) = \prod_{t=1}^{T} P(\mathbf{o}_t|q_t, \lambda). \tag{A.3}$$

A.2 The Evaluation Problem

Each action targeted for recognition will be represented by an HMM, λ. The evaluation problem is concerned with finding the probability of a sequence of observations $\mathbf{O} = (\mathbf{o}_1\mathbf{o}_2...\mathbf{o}_T)$, given a model λ. must be evaluated for every model. The model with the highest probability is selected. The most direct way of calculating $P(\mathbf{O}|\lambda)$ is to begin by enumerating every possible state sequence of length T. We should end up with N^T sequences (each state is equally likely to occur for each of the T observations, such that $N \cdot N \cdot N \cdot \ldots$) similar to

$$\mathbf{q} = (q_1 q_2 ... q_T), \tag{A.4}$$

where q_1 is the initial state. The probability of the observation sequence $\mathbf{O}$ given the state sequence defined above in Eq. A.4 is

$$P(\mathbf{O}|\mathbf{q}, \lambda) = \prod_{t=1}^{T} P(\mathbf{o}_t|q_t, \lambda). \tag{A.5}$$

Invoking the output independence assumption, Eq. A.5 can also be expressed as

$$P(\mathbf{O}|\mathbf{q}, \lambda) = b_{q_1}(\mathbf{o}_1) \cdot b_{q_2}(\mathbf{o}_2) \ldots b_{q_T}(\mathbf{o}_T). \tag{A.6}$$

This is the probability of producing some output $\mathbf{o}_1$ while in state q_1 at time $t = 1$ times the probability of producing some output $\mathbf{o}_2$ while in state q_2 at time $t = 2$, and so on. The probability of state sequence $\mathbf{q}$ for this model λ is

$$P(\mathbf{q}|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T}. \tag{A.7}$$

Now we can easily express the joint probability that $\mathbf{O}$ and $\mathbf{q}$ occur simultaneously by multiplying Eqs. A.6 and A.7, giving

$$P(\mathbf{O}, \mathbf{q}|\lambda) = P(\mathbf{O}|\mathbf{q}, \lambda) P(\mathbf{q}|\lambda). \tag{A.8}$$

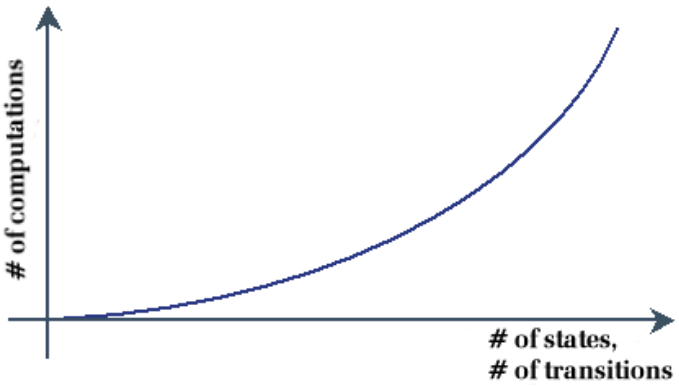


Figure A.1: Computation complexity grows exponentially as the number of states or state transitions are increased.

Finally, we can compute the probability of the observation given the model $P(\mathbf{O}|\lambda)$ by summing the joint probability over all N^T possible state sequences $\mathbf{q}$, such that

$$P(\mathbf{O}|\lambda) = \sum_{all\ \mathbf{q}} P(\mathbf{O}|\mathbf{q},\lambda)P(\mathbf{q}|\lambda) \tag{A.9}$$

$$= \sum_{all\ \mathbf{q}} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\mathbf{o}_t), \text{ where } a_{q_0 q_1} = \pi_{q_1} \tag{A.10}$$

$$= \sum_{q_1,q_2\ldots q_T} \pi_{q_1} b_{q_1}(\mathbf{o}_1) a_{q_1 q_2} b_{q_2}(\mathbf{o}_2) \ldots a_{q_{T-1}q_T} b_{q_T}(\mathbf{o}_T) \tag{A.11}$$

However, this calculation involves an exponential number of operations on the order of $2T \cdot N^T$, since at every $t = 1, 2, \ldots, T$, there are N possible states that can be reached and for each such state sequence about $2T$ calculations are needed for each term in the sum of Eq. A.11 (illustrated in Figure A.1). There is ample motive for a more efficient procedure, which we find in the *Forward-Backward algorithms*. These algorithms generate the same calculation, but on an order proportional to N^2T.

A.2.1 The Forward Algorithm

The Forward algorithm is characterized by a three-step procedure. We begin by introducing the forward variable $\alpha_t(i) = P(\mathbf{o}_1\mathbf{o}_2 ... \mathbf{o}_t, q_t = s_i|\lambda)$ as the probability of the partial observation sequence $\mathbf{O} = (\mathbf{o}_1\mathbf{o}_2 \ldots \mathbf{o}_t)$ when it terminates in state s_i. We solve for $\alpha_t(i)$ inductively, as follows:

1. Initialization

$$\alpha_1(i) = \pi_i b_i(\mathbf{o}_1), \quad 1 \leq i \leq N \tag{A.12}$$

2. Induction

$$\alpha_{t+1}(j) = b_j(\mathbf{o}_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}, \ 1 \leq j \leq N, \ 1 \leq t \leq T-1 \tag{A.13}$$

3. Termination

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{A.14}$$

Step 1 initializes the forward probabilities as the joint probability of state s_i and the first observation $\mathbf{o}_1$. In Step 2, $\alpha_t(i)a_{ij}$ is the probability of the joint event that $\mathbf{o}_1\mathbf{o}_2...\mathbf{o}_t$ is observed and that state s_j is reached at time $t+1$ through state s_i at time t. Summing over all the N possible states gives the likelihood of state s_j at time $t+1$ with all the previous partial observations. After s_j is determined, we arrive at $\alpha_{t+1}(j)$ by considering the observation $\mathbf{o}_{t+1}$ in state s_j. Step 3 sums all of the terminal forward variables $\alpha_T(i)$.

A.2.2 The Backward Algorithm

Similarly, the Backward algorithm is characterized by a recursive procedure over three stages. Consider the backward variable $\beta_t(i) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2}...\mathbf{o}_T, q_t = s_i|\lambda)$ as the probability of the partial observation sequence from $t+1$ to the end, given state s_i at time t and the model λ. Again, we solve for $\beta_t(i)$ inductively, as follows:

1. Initialization

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \tag{A.15}$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j), \ 1 \leq j \leq N, \ 1 \leq t \leq T-1 \tag{A.16}$$

3. Termination

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \pi_i b_i \mathbf{o}_1 \beta_1(i) \tag{A.17}$$

Step 1 arbitrarily initializes $\beta_T(i)$ to be 1 for all states. Step 2 first considers the transition from state s_i to state s_j, the observation $\mathbf{o}_{t+1}$ in s_j, as well as the remaining partial observation sequence from s_j (denoted as $\beta_{t+1}(j)$). This is all taken over all possible states s_j at time $t+1$ in order to determine the likelihood of being in state s_i at time t. We continue, backing up in time $t = T-1, T-2, \ldots, 1$ to the beginning. We arrive at the final probability $P(\mathbf{O}|\lambda)$ in step 3 by summing over all N states. By identifying the relation

$$P(\mathbf{O}, q_t = s_i|\lambda) = \alpha_t(i)\beta_1(i),\ 1 \leq i \leq N,\ 1 \leq t \leq T-1, \tag{A.18}$$

both forward and backward variables can also be used to calculate $P(\mathbf{O}|\lambda)$ as

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} P(, q_t = s_i|\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_1(i). \tag{A.19}$$

A.3 Training HMMs

Considered the most difficult aspect of an HMM, "training" entails developing a parameterized model that properly characterizes a sequential process. Observations that represent typical actions, called the training set, are used to train an HMM by optimally adjusting model parameters $\{\mathbf{A}, \mathbf{B}, \Pi\}$ to maximize $P(\mathbf{O}|\lambda)$. Although there is no known way to solve for a maximum likelihood model analytically, an iterative procedure called the Baum-Welch (BW) Algorithm is regularly used for this optimization process because convergence is guaranteed.

A.3.1 The Baum-Welch Algorithm

The Baum-Welch method is an Expectation-Maximization (EM) algorithm for reestimating HMM parameters. In unsupervised learning, *incomplete* data contains only observable data, such as a data sample $\mathbf{x}$. *Complete* data represents both hidden data, such as from which state data sample $\mathbf{x}$ comes, as well as observable data [67]. The purpose of any EM algorithm is to maximize the likelihood from incomplete data by iteratively maximizing the expectation of likelihood from complete data. In other words, EM methods iteratively update and improve model parameters using training data until optimal values are reached. This process is also called reestimation.

We begin by establishing the probability of being in state s_i at time t then in state s_j at time $t+1$, given the model and the observation sequence as

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | \mathbf{O}, \lambda). \tag{A.20}$$

Recalling from the previous section when the forward and backward variables were introduced, we rewrite Eq. A.20 as

$$\begin{aligned} \xi_t(i,j) &= \frac{P(q_t = s_i, q_{t+1} = s_j, \mathbf{O}|\lambda)}{P(\mathbf{O}|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O}|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}. \end{aligned} \tag{A.21}$$

We define the *a posteriori* probability of being in state s_i at time t, given the entire observation sequence and the model as $\gamma_t(i)$ such that

$$\begin{aligned} \gamma_t(i) &= P(q_t = s_i | \mathbf{O}, \lambda) \\ &= \frac{P(q_t = s_i, \mathbf{O}|\lambda)}{P(\mathbf{O}|\lambda)} \\ &= \frac{P(q_t = s_i, \mathbf{O}|\lambda)}{\sum_{i=1}^{N} P(q_t = s_i, \mathbf{O}|\lambda)}. \end{aligned} \tag{A.22}$$

Noticing that $P(q_t = s_i, \mathbf{O}|\lambda) = \alpha_t(i)\beta_t(i)$, we can rewrite Eq. A.22 in terms of the forward and backward variables as

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}, \tag{A.23}$$

where $\alpha_t(i)$ represents the partial observation sequence $\mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_t$ and state s_i at t, while $\beta_t(i)$ accounts for the remainder of the observation sequence $\mathbf{o}_{t+1} \mathbf{o}_{t+2} \ldots \mathbf{o}_T$, given state s_i at t. Even closer scrutiny will reveal a relationship between $\gamma_t(i)$ and $\xi_t(i,j)$ by summing over j, giving

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j), \ 1 \le i \le N, \ 1 \le t \le T-1. \tag{A.24}$$

If we sum $\gamma_t(i)$ over the time index t, we get a quantity that is interpreted as the expected (over time) number of times that state s_i is visited, or equivalently, the expected number of transitions made from state s_i, (if we exclude the final contribution at time $t = T$ from the summation) [67]. Likewise, if we sum $\xi_t(i,j)$ from $t = 1$ to $t = T - 1$, we can quantify the expected number of transitions from s_i to s_j. In other words,

$$\sum_{t=1}^{T} \gamma_t(i) = \text{expected number of transitions from } s_i \text{ in } \mathbf{O} \tag{A.25}$$

$$\sum_{t=1}^{T} \xi_t(i,j) = \text{expected number of transitions from } s_i \text{ to } s_j \text{ in } \mathbf{O} \tag{A.26}$$

By exploiting these relationships, reestimation formulas for HMM parameters $\lambda = \{\pi, \mathbf{A}, \mathbf{B}\}$ are given by

$$\begin{aligned} \bar{\pi} &= \text{expected number of times in } s_i \text{ at time } t = 1 \\ &= \gamma_1(i) \qquad \text{(A.27)} \\ \bar{a}_{ij} &= \frac{\text{expected number of transitions from } s_i \text{ to } s_j}{\text{expected number of transitions from } s_i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \text{(A.28)} \\ \bar{b}_j(k) &= \frac{\text{expected number of times in } s_j \text{ and observing symbol } \mathbf{v}_k}{\text{expected number of times in } s_j} \\ &= \frac{\sum_{t=1,\ s.t.\ \mathbf{o}_t = \mathbf{v}_k}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}. \qquad \text{(A.29)} \end{aligned}$$

If we define the reestimated model as $\bar{\lambda} = \{\bar{\pi}, \bar{\mathbf{A}}, \bar{\mathbf{B}}\}$ from Eqs. A.27-A.29, Baum et al. guarantees that $P(\mathbf{O}|\bar{\lambda}) \geq P(\mathbf{O}|\lambda)$ as the reestimation calculations are repeated. This is to say that by iteratively replacing λ with $\bar{\lambda}$, we can improve monotonically the probability of $\mathbf{O}$ being observed from the model until some threshold is reached, resulting in a maximum likelihood estimate of the HMM. We remind the reader that the forward-backward algorithm does not provide global maximization over the entire likelihood function, but rather, only guarantees a local maxima is reached. The reestimation derivation presented above holds

for the discrete HMM, while the continuous HMM requires a similar, but slightly different treatment to accommodate continuous density functions. For an exhaustive description of this process, see [67].

A.4 Decoding HMMs

Decoding an HMM refers to finding the "optimal" state sequence associated with a given observation sequence. One possible criterion for optimality is to maximize the expected number of correct individual states, where each state q_t at time t represents the most likely. However, this measure determines the most likely state at every instant without considering the probability of occurrence of the entire sequence of states. With this approach, we can possibly generate an invalid sequence or one with zero probability (if $a_{ij} = 0$ for some i and j) . We define optimality in terms of recovering the single most likely state sequence $\mathbf{q}$ for a given sequence of observations $\mathbf{O}$ and a model λ, i.e., maximizing $P(\mathbf{q}|\mathbf{O}, \lambda)$, which is the same as maximizing $P(\mathbf{q}, \mathbf{O}|\lambda)$. The Viterbi algorithm is commonly used to calculate optimal $\mathbf{q}$.

A.4.1 The Viterbi Algorithm

To find the optimal $\mathbf{q} = (q_1 q_2 \dots q_T)$, we calculate the maximum probability that a partial observation sequence and state sequence up to time t can have when in the current state is s_i as

$$\delta_t(i) = \max_{q_1 q_2 \dots q_{t-1}} P(q_1 q_2 \dots q_{t-1}, q_t, \mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_t|\lambda). \tag{A.30}$$

By induction, we have

$$\delta_{t+1}(j) = [\max_{1 \le i \le N} \delta_t(i) a_{ij}] b_j(\mathbf{o}_{t+1}),\ 1 \le i \le N,\ 1 \le t \le T-1. \tag{A.31}$$

To recover the actual state sequence, we record the arguments that maximize Eq. A.31, for each t and s_j using the array $\psi_t(j)$. We outline the Viterbi procedure as follows:

1. Initialization

$$\delta_t(i) = \pi_i b_i(\mathbf{o}_1),\ 1 \le i \le N \tag{A.32}$$

$$\psi_t(i) = 0. \tag{A.33}$$

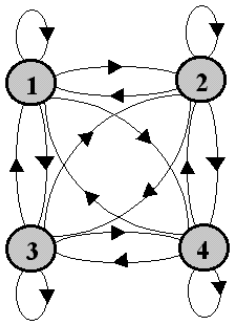


Figure A.2: Example topological structure: Fully-connected ergodic HMM

2. Recursion

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(\mathbf{o}_t), 1 \le j \le N,\ 2 \le t \le T \tag{A.34}$$

$$\psi_t(j) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], 1 \le j \le N,\ 2 \le t \le T \tag{A.35}$$

3. Termination

$$P^* = \max_{1 \le i \le N} [\delta_T(i)] \tag{A.36}$$

$$q_T^* = \arg \max_{1 \le i \le N} [\delta_T(i)] \tag{A.37}$$

4. State sequence backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*),\ t = T-1, T-2, \ldots, 1 \tag{A.38}$$

The Viterbi calculations are on the order of N^2T multiplications. The Viterbi algorithm can also be used for evaluating an HMM, i.e., calculating $P(\mathbf{O}|\lambda)$. Using it is viewed as a specific case of the forward-backward algorithm where the most likely path at each time step is selected. Recall that the forward-backward algorithm obtains $P(\mathbf{O}|\lambda)$ through the summation of $P(\mathbf{O}, \mathbf{q}|\lambda)$ over all possible state sequences $\mathbf{q}$, while the Viterbi algorithm finds the maximum of $P(\mathbf{O}, \mathbf{q}|\lambda)$ over all $\mathbf{q}$. While less robust than the forward-backward algorithm, the Viterbi algorithm is a substantially more efficient alternative.

A.4.2 HMM Topologies

Like any other state machine, the HMM's topological structure is influenced by the training data and by the nature of the gestures or actions the model is attempting to represent

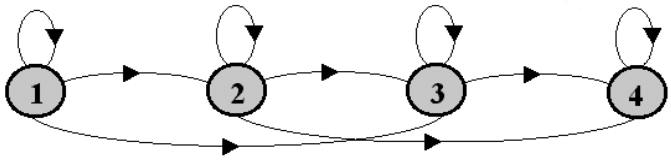


Figure A.3: Example topological structure: Left-to-Right HMM with Skip Transitions

[67, 118]. Ergodic models are characterized by a full state transition matrix **A**, suggesting that transitions can be made from any state to any other state (illustrated in Figure A.2). Left-to-right or causal models, which have upper triangular transition matrices, impose a temporal order on the model because the lower numbered states account for observations that occur prior to those for higher numbered states (shown in Figure A.3). Besides the two topologies mentioned above, there are many other possible variations and combinations. While ergodic models with several states may improve modeling robustness, in practice, most implementations resort to non-ergodic left-to-right structures with the minimum number of states to ease the computational burden.

APPENDIX B

Blackjack "21"

B.1 Introduction

Blackjack, also known as "21," is a popular card game played in casinos throughout the United States and the world. While there are many basic strategies for Blackjack, in this section we outline the rules that were used to conduct experiments in Chapter 6. Blackjack is a contest with any number of players who bet against the dealer, also called the *house*.

The basic premise of the game is for each player to have a hand value that is closer to 21 than that of the dealer, without going over 21. Although only one deck was used in our experiments, we assume strategies employed by one player are independent of strategies used by others. In other words, a player's hand is strictly played out against the hand of the dealer. In this game, the rules of play for the dealer are set, so the dealer cannot make strategic decisions that may increase the house's advantage.

B.2 Rules of the Game

Blackjack is traditionally played on a special table where the dealer stands on one side and all players stand on the other. This table is typically covered in green felt and is labeled with house rules, special places for wagers, etc. To minimize tracking errors during our experiments, the table was covered with a solid blue canvas without labels.

B.2.1 Making bets

Before any cards are dealt, each player must wager a bet using chips of a single denomination (only white chips). The player takes the desired number of chips from her *pot*, where all

chips are stacked on top of each other, and scatters them in front on the house's pot so that they do not touch or are not stacked on top of each other. This ensures that chips can be segmented and recognized. Once the cards have been dealt, players are not allowed to touch their bets.

Once the hand is over, the dealer will move around the table to each position in turn, paying winners and collecting the chips from losing hands. Chips collected by the dealer are returned to the house pot. After the dealer has paid the player, chips can be removed and returned to the player's pot. Players are not permitted to let winnings ride from game to game, i.e., they must remove all chips at the end of each game.

B.2.2 Value of Cards

The value of a hand is simply the sum total of card evaluations in the hand. In a 52-card deck, an ace can be either eleven or one, any face card (jack, queen, or king) counts as ten, and all other cards retain their face value. If an ace is drawn, it is assumed that its value changes to reflect the best hand. For example, if the initial hand contains an ace and a six, the hand is value is 17. However, if another card is drawn, say an eight, the ace's value becomes one to avoid exceeding 21 and total hand values is now 15. A hand that contains an Ace is called a *soft* total if the Ace can be counted as either 1 or 11 without the total going over 21. For example (ace, 6) is a soft 17. The description stems from the fact that the player can always draw another card to a soft total with no danger of *busting* by going over 21. On the other hand, the hand (ace, 6, 10) is a *hard* 17, since the ace must be counted as only one, again because counting it as 11 would make the hand go over 21.

Dealing the Cards

Once all the bets are made, the dealer will deal two cards to each player, then to himself. In a *shoe game*, all player cards are dealt face up; however, the dealer's hand always has one card dealt facing down and the other facing up. Once the cards are dealt, play proceeds around the table, starting at the first seat to the dealer's left, also called *first base*. Players are not allowed to remove the cards from the table, but can peek under the face-down card to check its value (if not playing a shoe game). Each player indicates to the dealer how he

wishes to play the hand. The player strategies which were available for our experiments are covered in detail in the sections below. After each player has finished his hand, the dealer will complete his hand, and then pay or collect the player bets.

B.3 How the dealer plays his hand

The dealer's play is restricted to a common rule in many casinos: "Dealer stands on all 17s." In this case, the dealer must continue to draw cards, i.e., *hit*, until his hand value totals 17 or greater. An ace in the dealer's hand is always counted as 11 if possible without the dealer going over 21. For example, (ace, 8) would be 19 and the dealer would stop drawing cards, i.e., *stand*. An (ace, 5) hand is only 16, so the dealer would continue to draw cards until the hand's value is 17 or more. Again, the dealer has no choices to make in the play of his hand. He cannot split pairs, but must instead simply hit until he reaches at least 17 or busts by going over 21.

B.4 Player Strategy

In addition to the rules listed earlier, their are particular strategies that more skill Blackjack players can use to improve their odds of winning. Those strategies we consider listed below.

B.4.1 Hitting/Standing

The most common decision a player must make during the game is whether to draw another card to the hand ("hit") or stop at the current total ("stand").

B.4.2 Doubling Down

The player is allowed to double the initial bet on his first two cards and receive exactly one more card to improve his hand. The option to double is often permitted on the player's first two cards only, although some casinos allow doubling after splitting a pair.

Splitting Pairs

If the first two cards a player is dealt are a pair, he may split them into two separate hands, bet the same amount on each and then play them separately. Aces receive only one additional card. After splitting, A-10 counts as 21 and not as Blackjack.

APPENDIX C

Vision Action Recognition System (VARS)

C.1 Introduction

To implement the ideas presented in this dissertation, we have developed the *Vision Action Recognition System* (VARS), a multi-application environment written in Microsoft Visual C++. VARS runs on the Win9x/NT platform with Intel compatible processors. Using the standard conversion of 62 lines of code per kilobyte, VARS is estimated to contain over 93,000 lines of code. In particular, the framework for ObjectSpaces has been duplicated in software so that VARS executes in real-time or near real-time, depending on the availability of systems resources, i.e., processor speed, memory, etc. With the exception of Matrox drivers required for frame capture, all software written is original.

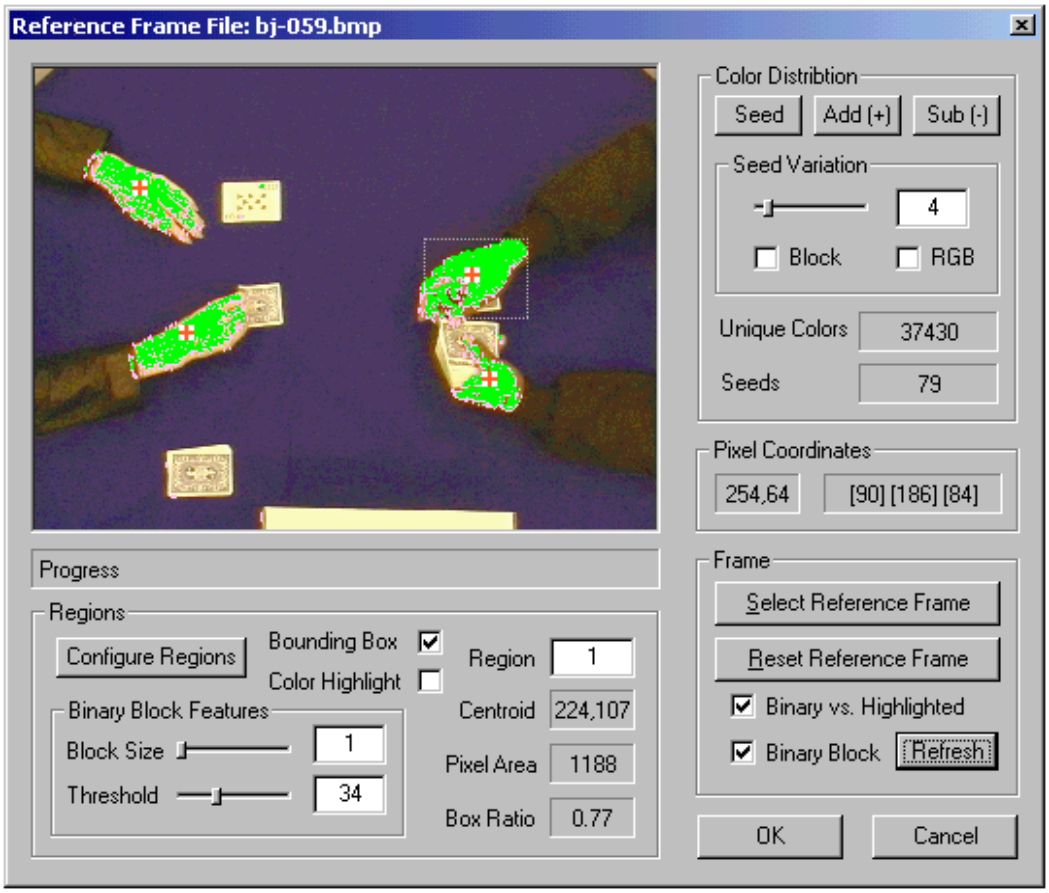


Figure C.1: VARS dialog for configuring hand color distribution **C**: Using a manually placed cross-hair cursor over the pixel of interest, the distribution adds all colors within the standard deviation specified by *seed variation*.

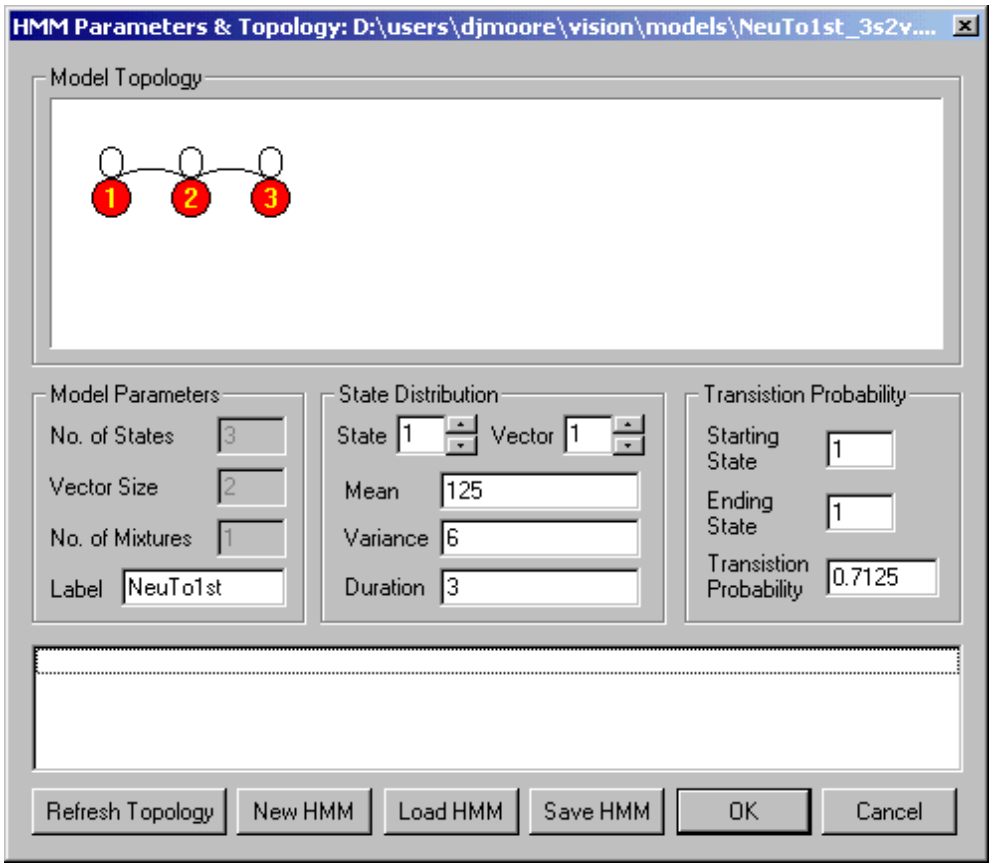


Figure C.2: VARS dialog for configuring HMM: Model topology and initial parameters are set here.

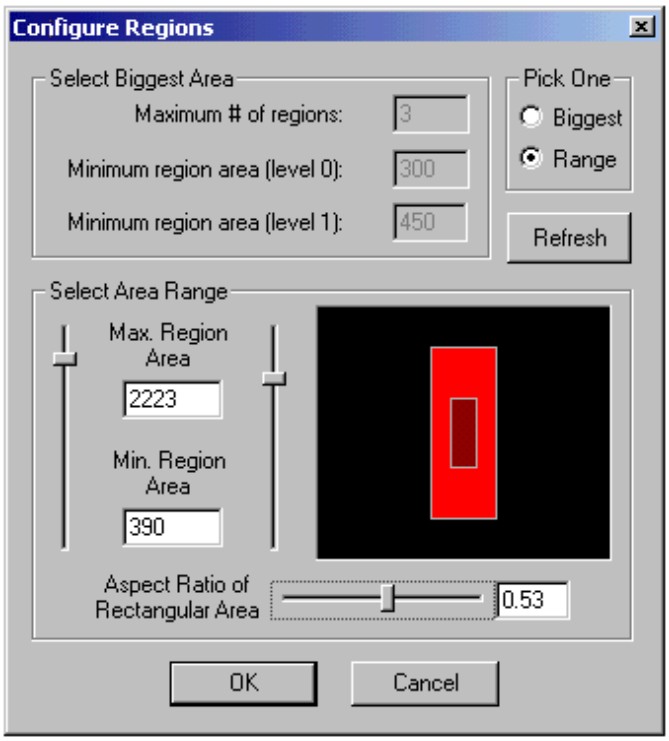


Figure C.3: VARS dialog for configuring hand region area (max. and min.) and ration of bounding box sides.

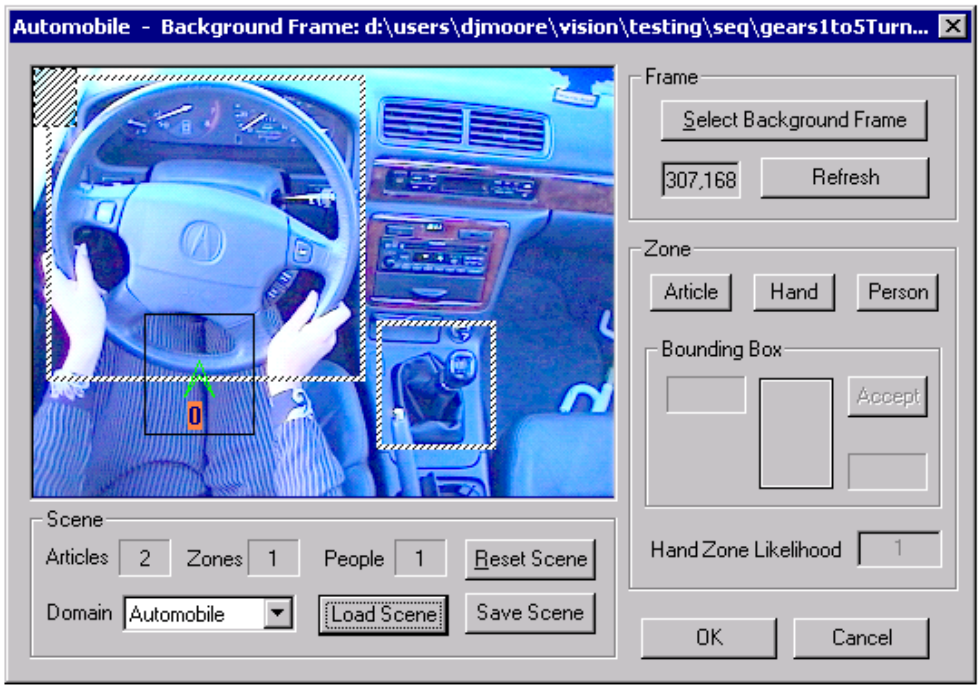


Figure C.4: VARS dialog for configuring scene: Know articles, activity zones, people, initial background, etc. are set up here.

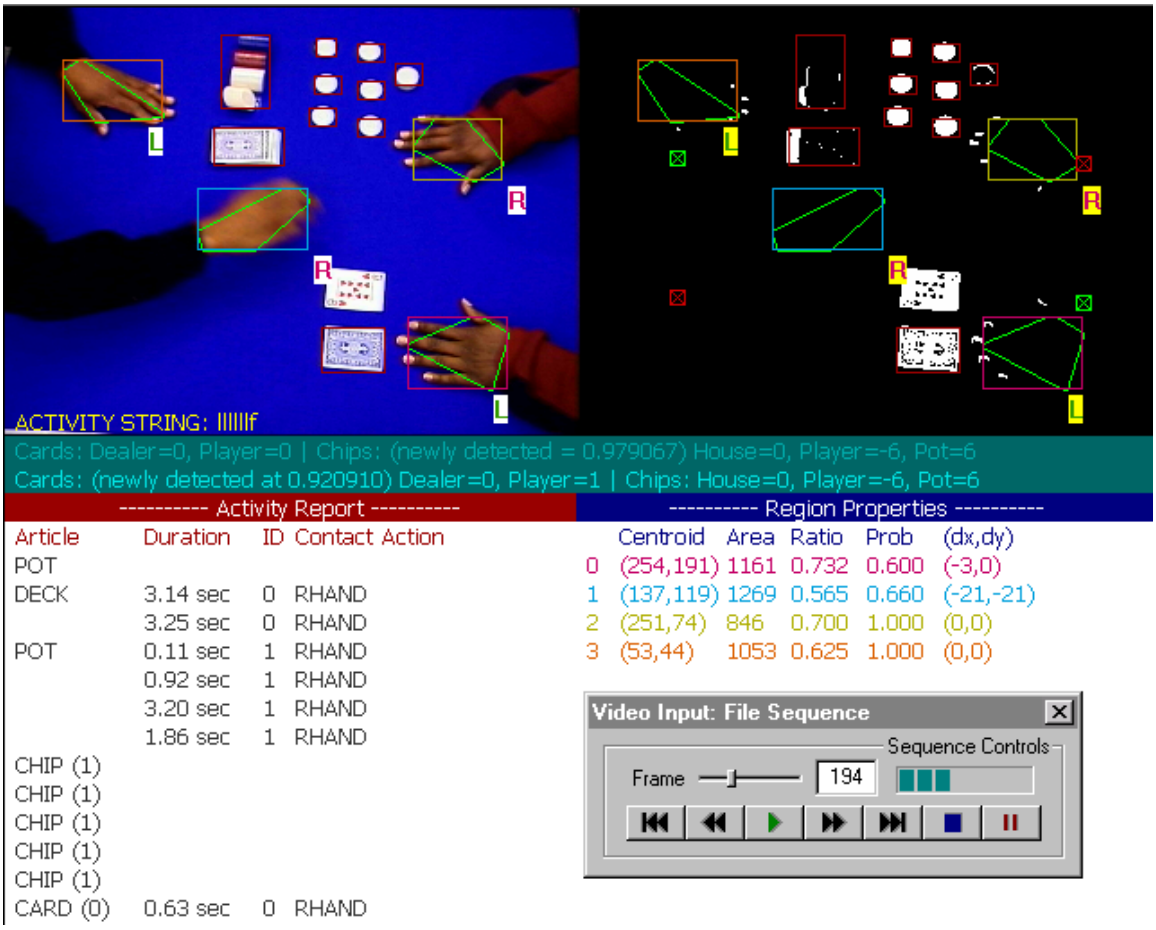


Figure C.5: Actual VARS screen capture shows evidence corresponding to data appearing in Table 6.1.

APPENDIX D

A Grammar Overview

Through the pioneering efforts of Noam Chomsky in the middle 1950s, several mathematical models of grammar have been proposed. We begin with a brief review of grammar and other concepts from formal language theory and computational linguistics.

D.1 Overview of Syntactic Pattern Recognition

Inspired by concepts from formal language theory and natural language processing in the early 1960s, *syntactic pattern recognition* exploits the *structure* of patterns during classification. This is in contrast to more conventional, analytical approaches that deal with patterns on a more quantitative basis, often ignoring correlations between the components of a pattern. For example, Figure D.1 shows a spiral pattern composed on crosses ("x") and naughts ("o") with a single, unlabeled symbol represented by a question mark. To classify the unlabeled symbol, conventional pattern classification strategies might look toward the nearest neighbor or may offer some likelihood based on prior observations, either of which might incorrectly label it as a naught. On the other hand, syntactic pattern approaches attempt to capitalize on the apparent structure for labeling and classification.

The problem of representing extended, complex activities is similar to the challenges researchers have faced in other areas, such as speech recognition, computational linguistics, and natural language processing. The HMM has a well documented history as a tool for isolated word recognition. However, sentence and phrase recognition require additional tools like grammar for aggregating isolated words. Similarly, syntactic approaches show remarkable promise for addressing problems in understanding multitasked activity.

Syntactic pattern recognition has also been a valuable tool for characterizing multi-

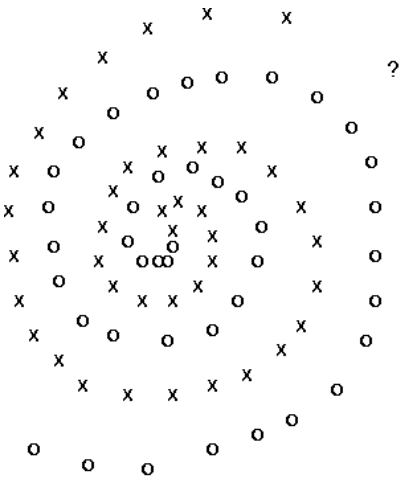


Figure D.1: From inspection, we expect the ? to be a cross, despite the fact that its closest neighbors are mostly naughts. Syntactic pattern classifier utilize structure whereas conventional pattern recognition approaches tend to rely on quantitative measures like distance. Provided courtesy of Michael Alder.

dimensional structure and shape, such as classification of chromosomes and alphanumeric characters [139].

D.2 Types of Grammars

There are generally four types of grammars with each defined, in part, by how the rewriting rules are designed. Naturally, the complexity and restrictions on the grammar, and consequently, of the language generated by it are all affected by a grammar's production set $\mathcal{P}$.

Definition D.1 Unrestricted Grammars

An *unrestricted grammar* has rewriting rules of the form $\alpha \rightarrow \beta$, where $\alpha \in V_G^+$ and $\beta \in V_G^*$. The only restriction on a production of an unrestricted grammar is that the left-hand side (LHS) not be null.

Definition D.2 Context-Sensitive Grammars

A *context-sensitive grammar* (CSG) has productions of the form $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$, where α_1 and α_2 are in V_G^*, $\beta \in V_G^+$, and $A \in V_N$. This is to say that it is possible for α_1 or α_2 to

be the null string. CSG allows replacement of the nontermial A by the string β only when A appears *in the context* $\alpha_1 A \alpha_2$ of strings α_1 and α_2. The length of the derived string remains the same or increases with each rule application, i.e., CSG productions cannot yield the null string ϵ. A grammar is said to be *positive* if there are no null productions. Context-sensitive grammars represent an intermediate step between the context-free and the unrestricted grammars.

Definition D.3 Context-Free Grammars

A *context-free grammar* (CFG) has productions of the form $A \rightarrow \beta$, where $A \in V_N$ and $\beta \in V_G^*$. In a stochastic framework, context-freeness of a grammar translates into statistical independence of its nonterminals. The name context-free arises from the fact that the variable A may be replaced by a string β regardless of the context in which A appears. When CFG has at most one nonterminal symbol on the right hand, that is, if $|\beta|_N \leq 1$, it is called *linear*.

A context-free grammar can form the language $L(G_{free}) = L_f : a^n b^m \ldots$, where correlations between n and m can be counted. Keep in mind that *counting*, in the linguistic sense, refers to symbolic *matching* as opposed to counting in the mathematical sense. To say that correlations between n and m can be counted means that we can enforce a relation between n and m in L_f without requiring fixed values for n and m, i.e., $a^n b^m = a^{3m} b^m$. Moreover, counting is performed recursively so that each symbol b matches symbol a in reverse order. Every positive CFG is also context-sensitive. However, any grammar that is not CFG cannot be CSG.

Context freeness in grammar places no limitations on the applicability of a rule with regard to task semantics, i.e., nonterminals generate the substring without regard to surrounding symbols. This assumption, while convenient, is not strictly true in most cases as there is likely some dependence between productions in a grammar. Adding the stochastic component to context-free grammar can ameliorate this limitation so that parsing is more "sensitive" to the context of surrounding symbols.

Definition D.4 Regular Grammars

Finally, a *regular* (or *finite-state*) *grammar* is actually a context-free grammar in which each rule has one of the following forms: (1) $A \rightarrow a$, (2) $A \rightarrow aB$, (3) $A \rightarrow \lambda$, where A and B are nonterminal variables in V_N, a is a terminal constant in V_T, and λ is a string in V_N. Regular grammar is used to form the simplest language models, generally of the form $L_r : a^n b^m \ldots$, where n and m are independent.

The four grammars mentioned above provide a formal system for generating strings over an alphabet. They also make up the *Chomsky hierarchy*, named after Noam Chomsky, who proposed them as syntactic models of natural language [136]. Unrestricted, context-sensitive, context-free, and regular grammars are referred to as type 0, type 1, type 2, and type 3 grammars, respectively. The restrictions placed on the production rules increase with the number of the grammar.

D.2.1 Machines

Procedures that check the syntax of a string to make sure it is derivable from the start symbol using rules of the grammar are implemented by *parsing algorithms* or simply *parsers*. Parsers requiring only a fixed (finite) amount of memory for arbitrary input are called *finite-state machines* or *finite-state automata*. In practice, strings of a regular language L_r can be generated by finite state machines which do not need extra stack memory. However, the absence of stack memory in the parsing and generating mechanism of regular grammars is one of its limiting factors [74]. Recursion is necessary for a finite set of rules to generate infinite languages and strings of arbitrary length, i.e., the language given by $\{a^i b^i | i \geq 0\}$. Recursive relationships cannot be captured by regular grammar because extra memory is required to represent the states of the parsing mechanism before and after the recursive invocation. In other words, to accept the language $\{a^i b^i\}$, a machine needs to record the processing of any finite number of a's. The restriction of having finitely many states does not allow the automaton to "remember" the number of a's in the input string, i.e., finite state machines can only count, or match, a finite number symbols in the alphabet.

A *push-down automaton* (PDA) is a finite-state machine augmented with external stack memory, which provides the PDA with last-in, first-out memory management. By utilizing stack and states, the PDA can accept the language given by $\{a^i b^i | i \geq 0\}$, which

cannot be handled by a deterministic finite automaton. A context-free grammar is typically realized as a push-down automaton, allowing for recursive invocations of the grammar nonterminals. However, a PDA cannot model dependencies between more than two elements of the alphabet, i.e., $a^i b^i c^i$ are not realizable by a PDA.

A Turing machine is a finite-state machine in which a transition prints a symbol on tape, i.e., stores a symbol in memory. A Turing machine has no limitation on the amount of time or memory available for computation. The "tape head" can move either left or right in one direction, allowing the machine to read and manipulate the input as many times as desired. Unlike finite-state and push-down automata, a Turing machine need not read the entire input string to accept the string. For this reason, unrestricted grammars are accepted by Turing machines.

A linearly-bounded automaton (LBA) is a Turing machine in which the amount of available tape is determined by the length of the input string. In other words, a LBA is Turing machine with finite tape. A Turing machine writes the rules of an unrestricted grammar onto tape that is infinite, i.e., infinite memory. However, an LBA encodes the rules in states and transitions due to the finite amount of tape available. Due to computational complexity of LBA, which are typically used to realize a context-sensitive language, CSGs are much more difficult to realize than a CFG. The table below describes the machines that are used to handle the respective grammar and language [136]. We will say more about parsing in Section 6.4.2.

Grammars	**Languages**	**Accepting Machines**
Type 0	recursively enumerable	Turing machines, nondeterministic Turing machines
Type 1	context-sensitive	Linear-bounded automata
Type 2	context-free	Push-down automata
Type 3	regular	deterministic finite state automata, non-deterministic finite state automata

D.3 Rule Probability Estimation for SCFGs

While rule probabilities can be set with empirically selected values initially, they benefit from more formal training exercises where likelihoods are adjusted to be more consistent with actual observations. In estimating these probabilities, we are reminded of the similarities between SCFG and the HMM. In the HMM, the process that determines state transitions, i.e., state derivation, is hidden, but can be solved using the Baum-Welch method. Similarly, for stochastic context-free grammar, rule derivations are unobservable and can be solved using familiar Expectation-Maximization (EM) approaches. Baker provides an EM procedure for obtaining maximum-likelihood (ML) estimates [9]. However, for small, task grammars that have small symbol alphabets and productions that generate only a few nonterminals, an easier procedure for estimating rule probabilities based on bigram probability estimation is generally sufficient.

We estimate rule probabilities by calculating the average production likelihood. The average simply divides the *count*, the number of times a rule is applied, denoted $c(X \to \lambda)$ for production $X \to \lambda$, by the count of all rules with X on the LHS, i.e.,

$$P(X \to \lambda) = \frac{c(X \to \lambda)}{\sum_{\mu} c(X \to \mu)}, \tag{D.1}$$

where μ represents all nonterminals generated by nonterminal X.

D.4 Specifying Activities with Grammar

Rule-based activities, which are inherently structured and have defined sub-tasks, can be represented appropriately using SCFG. In developing a grammar for a particular multi-tasked activity, it is necessary to have a complete understanding of the component tasks and events that are involved. To represent the overall structure of an activity in all of its possible variations, the grammar's production rules must describe the syntactic dependencies between these components. In other words, the basic events of a process or activity become the terminal constants of the task grammar while the ordering between events is represented by nonterminal variables. Since we are considering well-understood activities, it is convenient to specify the underlying grammar by hand using known task rules or

constraints. Attempting to "learn" the grammar is a difficult process that requires good initial conditions, a prodigious amount of training data and, typically, a fair amount of experimentation.

In practice, manually specified grammar should be tested with actual training data to make sure that it does not generate semantically illegitimate predictions or drive the parser into an infinite loop. Of particular concern are null productions, i.e., $\lambda \rightarrow \epsilon$, which can confuse the parsing engine [132]. In our approach, the detection of events, and hence string generation, is driven by evidence collected from the environment. Since we only generate strings from evidential measurements, there is no acceptable means of producing a null string, aside from complete detecting inactivity, i.e., no evidence to collect. To avoid this problem, we simply eliminate all null productions in the grammar, which may require massaging the production set by adding additional nonterminal variables, but generally can be accomplished without violating the semantic expressiveness of the grammar. We refer the reader to Stolcke [133], who provides a formal procedure for eliminating null-productions in SCFG. He also describes an alternative treatment that tolerates null productions. With null productions eliminated, we are reminded that such a SCFG now becomes a stochastic context-sensitive grammar as well.

D.5 Recursive Grammar

A left recursion occurs when productions of the form

$$\begin{aligned} A &\rightarrow Aa \\ &\rightarrow a \end{aligned}$$

appear in grammar. The parser will consider $A \rightarrow .Aa$ and $A \rightarrow .a$ during the first prediction step. The first production will generate the same two again, and so forth. Left recursions in SCFG can significantly affect the total probability of a predicted state because it will be added infinitely many times. The treatment for a left-recursive expansion of this nature requires a parsing concept called *left corner* remediation that collapses all repeated prediction steps into a single, modified prediction step and computes the corresponding sums in closed form.

S	$\rightarrow$	AB	$[p_1]$
	$\rightarrow$	C	$[p_2]$
	$\rightarrow$	d	$[p_3]$
A	$\rightarrow$	AB	$[p_4]$
	$\rightarrow$	a	$[p_5]$
B	$\rightarrow$	bB	$[p_6]$
	$\rightarrow$	b	$[p_7]$
C	$\rightarrow$	BC	$[p_8]$
	$\rightarrow$	B	$[p_9]$
	$\rightarrow$	c	$[p_{10}]$

P_L	S	A	B	C
S		p_1		p_2
A		p_4		
B				
C			$p_8 + p_9$	

R_L	S	A	B	C
S	1	$\frac{-p_1}{-1+p_4}$	$p_2p_8 + p_2p_9$	p_2
A		$\frac{-1}{-1+p_4}$		
B			1	
C			$p_8 + p_9$	1

Table D.1: Left Corner P_L and Reflexive Transitive Closure R_L matrices for a simple SCFG.

Definition D.5 *Two nonterminals X and Y are said to be in a* **left corner relation** $X \rightarrow_L Y$ *iff there exists a production for X that has a right hand side starting with Y, i.e., $X \rightarrow Y\lambda$.*

The *probabilistic left-corner relation*[1] $P_L = P_L(G)$ is the matrix of probabilities $P(X \rightarrow_L Y)$, defined as the total probability of choosing a production for X that has Y as a left corner:

$$P(X \rightarrow_L Y) = \sum_{X \rightarrow Y\lambda \in G} P(X \rightarrow Y\lambda). \tag{D.2}$$

The left-corner relationship matrix P_L is essentially the adjacency matrix that describes the production probability associated with generating any pair of nonterminals in the grammar. Because the production direction is one way, this matrix is intended to be read horizontally. The only elements in the matrix represent the probabilities of productions starting with nonterminals. For clarity, consider the example given in Table D.1. The only elements in the matrix represent probabilities of nonterminal productions. The reflexive, transitive closure of $X \rightarrow_L Y$, i.e., $X \Rightarrow_L^* Y$, exists iff $X = Y$ or there is a nonterminal Z such that $X \rightarrow_L Z$ and $Z \Rightarrow_L^* Y$. We can compute a total recursive contribution of each left-recursive production rule using the *probabilistic reflexive, transitive left-corner relation* matrix $R_L = R_L(G)$. This matrix is composed of probability sums $R(X \Rightarrow_L^* Y)$, where

[1] If a *probabilistic relation* R is placed by its set-theoretic version R', i.e., $(x, y) \in R'$ iff $R(x, y) \neq 0$, then the closure operations used here reduce to their traditional discrete counterparts; hence the choice of terminology

each $R(X \Rightarrow_L^* Y)$ is defined as a series

$$\begin{aligned} R(X \Rightarrow_L^* Y) &= P(X = Y) \\ &\quad + P(X \rightarrow_L Y) \\ &\quad + \sum_{Z_1} P(X \rightarrow_L Z_1) P(Z_1 \rightarrow_L Y) \\ &\quad + \sum_{Z_1, Z_2} P(X \rightarrow_L Z_1) P(Z_1 \rightarrow_L Z_2) P(Z_2 \rightarrow_L Y) \\ &\quad + \ldots \end{aligned} \tag{D.3}$$

To express this a different way, consider the productions $X \rightarrow Y\nu$ and $Y \rightarrow a\eta$. If there is a path between X and Y, we can predict the likelihood of emitting terminal a by summing the probabilities along all of the paths connecting X with Y. Then we multiply this sum by the probability of directly emitting a from Y, i.e., $Y \rightarrow a\eta$. We write this as

$$\begin{aligned} P(a) &= P(Y \rightarrow a\eta) \sum_{\forall X} P(X \Rightarrow^* Y) \\ &= P(Y \rightarrow a\eta) \{ P_0(X \Rightarrow^* Y) \\ &\quad + P_1(X \Rightarrow^* Y) \\ &\quad + P_2(X \Rightarrow^* Y) \\ &\quad + \ldots \}, \end{aligned} \tag{D.4}$$

where $P_k(X \Rightarrow^* Y)$ is the probability of a path from X to Y of length $k = 1, \ldots, \infty$. The complete reflexive transitive closure R_L can be presented as a geometric series, which offers a nice closed form solution, i.e.,

$$R_L = P_L^0 + P_L^1 + P_L^2 + \ldots = \sum_{k=0}^{\infty} P_L^k = (I - P_L)^{-1}. \tag{D.5}$$

The significance of the matrix R_L is that its elements are the sums of the probabilities of the potentially infinitely many prediction paths leading from state ${}_kX \rightarrow \lambda.Z\mu$ to a predicted state ${}_iY \rightarrow .\nu$ by way of any number of intermediate states. During the prediction stage,

$$i : \ {}_kX \rightarrow \lambda.Z\mu \ [\alpha, \gamma] \implies {}_iY \rightarrow .\nu \tag{D.6}$$

for all productions $Y \rightarrow \nu$ such that $R(Z \Rightarrow_L^* Y) \neq 0$. The recursive correction results in more reliable forward and inner probabilities, now given by

$$\alpha' = \sum_{\forall \ \lambda, \mu} \alpha(i : \ {}_kX \rightarrow \lambda.Z\mu) R(Z \Rightarrow_L^* Y) P(Y \rightarrow \nu) \tag{D.7}$$

$$\gamma' = P(Y \rightarrow \nu). \tag{D.8}$$

$$
\begin{array}{lcll}
S & \rightarrow & AB & [p_1] \\
 & \rightarrow & C & [p_2] \\
 & \rightarrow & d & [p_3] \\
A & \rightarrow & AB & [p_4] \\
 & \rightarrow & a & [p_5] \\
B & \rightarrow & bB & [p_6] \\
 & \rightarrow & b & [p_7] \\
C & \rightarrow & BC & [p_8] \\
 & \rightarrow & B & [p_9] \\
 & \rightarrow & c & [p_{10}]
\end{array}
$$

P_U	S	A	B	C
S				p_2
A				
B				
C			p_9	

R_U	S	A	B	C
S	1		p_2p_9	p_2
A		1		
B			1	
C			p_9	1

Table D.2: Unit Production P_U and Reflexive Transitive Closure R_U matrices for a simple SCFG.

There are other production relationships that can throw the parser into an infinite loop. Consider the productions,

$$
\begin{array}{lcl}
A & \rightarrow & B \\
 & \rightarrow & a \\
B & \rightarrow & A
\end{array}
$$

and the state $i:\ {}_jA \rightarrow a.$ during the completion stage of set j, which contains

$$
\begin{array}{lcl}
j:\ {}_jA & \rightarrow & .B \\
j:\ {}_jA & \rightarrow & .a \\
j:\ {}_jB & \rightarrow & .A
\end{array}
$$

This operation will generate several states including the *unit production*, $i:\ {}_jA \rightarrow a.$, which will cause the parser to dive into an infinite loop.

Definition D.6 *Two nonterminals are said to be in a* **unit production relation** $X \rightarrow_U Y$ *iff there is a production for* X *of the form* $X \rightarrow Y$.

The *probabilistic unit-production relation* $P_U = P_U(G)$ is the matrix of probabilities $P(X \rightarrow Y)$. Similarly to the case with prediction, we seek to compute the closed form for the *probabilistic reflexive, transitive unit production relation* matrix $R_U = R_U(G)$, which is given in closed form by

$$R_U = (I - P_U)^{-1}. \tag{D.9}$$

An example is given in Table D.2. Likewise, we seek estimates for the forward and inner probabilities. We modify the completion stage such that

$$\left.\begin{array}{l} i:\ {}_jY \rightarrow \nu.\ [\alpha^{''}, \gamma^{''}] \\ j:\ {}_kX \rightarrow \lambda.Z\mu\ [\alpha, \gamma] \end{array}\right\} \Longrightarrow\ i:\ {}_kX \rightarrow \lambda Z.\mu\ [\alpha^{'}, \gamma^{'}] \tag{D.10}$$

where the revised forward and inner probabilities are given, respectively, by

$$\alpha^{'} = \sum_{\forall\lambda,\mu} \alpha(i:\ {}_kX \rightarrow \lambda.Z\mu) R(Z \Rightarrow_U^* Y) \gamma^{''}(i:\ {}_jY \rightarrow \nu.) \tag{D.11}$$

$$\gamma^{'} = \sum_{\forall\lambda,\mu} \gamma(i:\ {}_kX \rightarrow \lambda.Y\mu) R(Z \Rightarrow_U^* Y) \gamma^{''}(i:\ {}_jY \rightarrow \nu.) \tag{D.12}$$

In summary, we use matrix R_U to collapse all unit completions into a single step. However, it is still necessary to go through the normal procedure when encountering iterative completion of non-unit productions.

D.6 Complexity of the Earley-Stolcke Algorithm

Most probabilistic parsers employ algorithms based on bottom-up parsing, such as the CYK algorithm [132]. On the other hand, the Earley-Stolcke algorithm is more efficient than bottom-up approaches because of its top-down prediction, which constrains the number of potential continuations of the string. In the worst-case scenario, where every production in the set is considered, the computational expense of the Earley-Stolcke algorithm performs as well as other known specialized algorithms [133].

Naturally, the size of the grammar and the length of the input string are the two biggest contributors to algorithm complexity. During parsing, the prediction and completion step are responsible for the bulk of the computation, especially if matrix inversions required for R_U and R_L have to be computed iteratively. In most cases, matrix inversion can be completed off-line and stored in memory for later use. For an string of length l, the complexity of the Earley-Stolcke algorithm is on the order of l^3. However, the average algorithm complexity of sparse grammars, which are more likely given the structure activities we consider, approaches the order of l [133].

$S \to a\ [p]$	$P_L = q$	$P_U = 0$
$S \to SS\ [q]$	$R_L = (1-q)^{-1} = p^{-1}$	$R_U = (1-P_U)^{-1} = 1$

	α	γ
State set 0		
${}_0 \to .S$	1	1
predicted		
${}_0S \to .a$	$1 \cdot p^{-1}p = 1$	p
${}_0S \to .SS$	$1 \cdot p^{-1}q = p^{-1}q$	q
State set 1		
scanned		
${}_0S \to a.$	$p^{-1}p = 1$	p
completed		
${}_0S \to S.S$	$p^{-1}q \cdot p = q$	$q \cdot p = pq$
predicted		
${}_1S \to .a$	$q \cdot p^{-1}p = q$	p
${}_1S \to .SS$	$q \cdot p^{-1}q = p^{-1}q^2$	q
State set 2		
scanned		
${}_1S \to a.$	q	p
completed		
${}_1S \to S.S$	$p^{-1}q^2 \cdot p = q^2$	$q \cdot p = pq$
${}_0S \to SS.$	$q \cdot p = pq$	$pq \cdot p = p^2q$
${}_0S \to S.S$	$p^{-1}q \cdot p^2q = pq^2$	$q \cdot p^2q = p^2q^2$
${}_0 \to S.$	$1 \cdot p^2q = p^2q$	$1 \cdot p^2q = p^2q$
predicted		
${}_2S \to .a$	$(q^2 + pq^2) \cdot p^{-1}p = (1+p)q^2$	p
${}_2S \to .SS$	$(q^2 + pq^2) \cdot p^{-1}q = (1+p^{-1})q^3$	q
State set 3		
scanned		
${}_2S \to a.$	$(1+p)q^2$	p
completed		
${}_2S \to S.S$	$(1+p^{-1})q^3 \cdot p = (1+p)q^3$	$q \cdot p = pq$
${}_1S \to SS.$	$q^2 \cdot p = pq^2$	$pq \cdot p = p^2q$
${}_1S \to S.S$	$p^{-1}q^2 \cdot p^2q = pq^3$	$q \cdot p^2q = p^2q^2$
${}_0S \to SS.$	$pq^2 \cdot p + q \cdot p^2q = 2p^2q^2$	$p^2q^2 \cdot p + pq \cdot p^2q = 2p^3q^2$
${}_0S \to S.S$	$p^{-1}q \cdot 2p^3q^2 = 2p^2q^3$	$q \cdot 2p^3q^2 = 2p^3q^3$
${}_0S \to S.$	$1 \cdot 2p^3q^2 = 2p^3q^2$	$1 \cdot 2p^3q^2$

Table D.3: Stolcke's example: (top) A simple SCFG with R_L and R_U. (lower) The left column represents the parsing chart while the two right-most columns represent the forward and inner probabilities, respectively, for each state. In both α and γ columns, the "$\cdot$" separates old factors from new ones. "+" indicates multiple derivations of the same state.

Bibliography

[1] G.D. Abowd, A.K. Dey, R. Orr, and J. Brotherton, "Context-awareness in wearable and ubiquitous computing," *First International Symposium on Wearable Computers*, pp. 179-80, Cambridge, MA, Oct. 13-14, 1997.

[2] M.D. Alder, *An Introduction to Pattern Recognition: Statistical, Neural Net and Syntactic methods of getting robots to see and hear.*, URL: *http://ciips.ee.uwa.edu.au/ mike/PatRec/PatRec.html* September 19, 1997.

[3] J.K. Aggarwal, S. Shah, R. Chin, and T. Pong, "Bayesian Paradigm for Recognition of Objects - Innovative Applications," *ACCV Proceedings: Third Annual Asian Conference on Computer Vision*, Vol. 2, pp. 275-82, Hong Kong, 1997.

[4] J.K. Aggarwal and Q. Cai, "Human motion analysis: a review," *Computer Vision and Image Understanding*, Vol. 73, No. 3, pp. 428-40, March 1999.

[5] J.F. Allen and G. Ferguson, "Actions and Events in Interval Temporal Logic," *Journal of Logic and Computation*, Vol. 4, No. 5, pp. 531-579, 1994.

[6] D. Ayers and M. Shah, "Monitoring Human Behavior in an Office Environment," *Interpretation of Visual Motion Workshop* (CVPR-98), June 1998.

[7] D. Ayers and M. Shah,"Recognizing Human Actions in a Static Room", *4th IEEE Workshop on Applications of Computer Vision* (WACV'98), October 19-21, 1998, Princeton, NJ.

[8] J.K. Baker, "The Dragon System - An Overview," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 23, No. 1, pp. 24-29, February 1975.

[9] J.K. Baker, "Trainable grammars for speech recognition," *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, ed. by J.J. Wolf and D.H. Klatt, pp. 547-550, MIT Press, cambridge, MA, 1979.

[10] D.S. Banarse and A.W.G. Duller, "Deformation Invariant Pattern Classification for Recognizing Hand Gestures," *IEEE International Conference on Neural Networks*, pp. 1812-1817, Vol. 3, 1996.

[11] M. Bar and S. Ullman, "Spatial Context in Recognition," *Masters Thesis*, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, Israel, December 1993.

[12] L.E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Stat.*, Vol. 37, pp. 1554-1563, 1966.

[13] L.E. Baum and J.A. Egon, "An equality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology," *Bulletin of American Meteorological Society*, Vol. 73, pp. 360-363, 1967.

[14] L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statisitical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, Vol. 41, No. 1, pp. 164-171, 1970.

[15] L.E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, Vol. 3, pp. 1-8, 1972.

[16] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Philosophical Transactions of the Royal Society of London*, Vol. 53, pp. 370-418. Published posthumously, then reprinted in *Bayesian Statistics*, pp. 189-217, S.J. Press, Wiley, New York, 1989.

[17] M. Black and Y. Yacoob, "Recognizing Facial Expressions under Rigid and Non-Rigid Facial," *International Workshop on Automatic Face and Gesture Recognition*, Zurich, pp. 12-17, 1995.

[18] M. Black and Y. Yacoob, "Tracking and Recognizing Rigid and Non-Rigid Facial Motions using Local Parametric Models of Image Motion," *IEEE 5th International Conference on Computer Vision*, pp. 374-381, 1995.

[19] M. Black and A.D. Jepson, "A probabilistic framework for matching temporal trajectories: CONDENSATION-based recognition of gestures and expressions," *5th European Conference on Computer Vision*, Vol. 1, pp. 909-24, 1998.

[20] M. Black and A.D. Jepson, "Recognizing temporal trajectories using the condensation algorithm," *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp.16-21 Nara, Japan, April 14-16, 1998.

[21] A. Bobick, "Movement, Activity, and Action: The Role of Knowledge in the Perception of Model," *Royal Society Workshop on Knowledge-based Vision in Man and Machine*, February 1997.

[22] A. Bobick and J. Davis, "Real-time Recognition of Activity using Temporal Templates," *3rd IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, pp.39-42, 1996.

[23] A. Bobick and A. Wilson, "A State-Based Technique for the Summarization and Recognition of Gesture," *Proceedings of the International Conference on Computer Vision,* Cambridge, Massachusetts, 1995.

[24] A. Bobick and Y. Ivanov, "Action Recognition using Probabilistic Parsing," *Proceedings of IEEE Computer Vision and Pattern Recognition*, Santa Barbara, CA, 1998.

[25] A.F. Bobick and, C.S. Pinhanez, "Controlling view-based algorithms using approximate world models and action," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 955-61, San Juan, Puerto Rico, June 17-19, 1997.

[26] A. Bobick and J. Davis, "An Appearance-based Representation of Aciton," *International Conference on Pattern Recognition*, February 1996.

[27] T.L. Booth and R.A. Thompson, "Applying probability measures to abstract languages," *IEEE Transactions on Computers*, Vol. 22, pp. 442-450, 1973.

[28] D. Braggins, "News from AIA's vision business conference," *Advanced Imaging*, p. 31, March 1999.

[29] D. Braggins, "Military Images," *Advanced Imaging*, p. 31, August 1999.

[30] M. Brand, "Understanding Manipulation in Video," *Proceedings of IEEE Automatic Face and Gesture Recognition*, pp. 94-99, Killington, Vermont, October 14-16, 1996.

[31] M. Brand and N. Oliver, "Coupled Hidden Markov Models for Complex Action Recognition," *Proceedings of IEEE Computer Vision and Pattern Recognition*, 1997.

[32] M. Brand and I. Essa, "Casusal Analysis for Visual Gesture Understanding," *AAAI Symposium on Computational Models for Integrating Language and Vision*, 1995.

[33] C. Bregler, "Learning and Recognizing Human Dynamics in Video Sequences," *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 568-574, San Juan, Puerto Rico, 1997.

[34] F. Brill, T. Olson and C. Tserng, "Event Recognition and Reliability Improvements for Autonomous Video Surveillance Systems," *Proceedings of the 1998 Image Understanding Workshop*, Monterey, CA, Vol. 1, pp. 267-283, November 20-23, 1998.

[35] H. Buxton and S. Gong, "Advanced Visual Surveillance using Bayesian Networks," *International Conference on Computer Vision,* Cambridge, MA, June 1995.

[36] L. Campbell and A. Bobick, "Recognition of Human Body Motion using Phase Space Constraints," *Proceedings of the International Conference on Computer Vision*, Cambridge, Massachusetts, 1995.

[37] L. Campbell, D.A. Becker, A. Azarbayejani, A. Bobick, and A. Pentland, "Invariant Features for 3-D Gesture Recognition," *Second International Conference on Automatic Face and Gesture Recognition*, pp. 157-162, Killington, VT, October 14-16, 1996.

[38] L. Campbell and A.F. Bobick, "Using Phase Space Constraints to Represent Human Body Motion," *International Workshop on Automatic Face and Gesture Recognition,* Zurich, pp. 338-343, 1995.

[39] J. Cassel, "A Framework for Gesture Generation and Interpretation," *Computer Vision in Human-Machine Interaction*, R. Cipolla and A. Pentland, eds., Cambridge University Press, 1996.

[40] C. Cédras and M. Shah, "Motion-Based Recognition: A Survey," *Image and Vision Computing*, Vol. 13, No. 2, pp. 129-155, 1995.

[41] C. Cohen, L. Conway, D. Koditschek, "Dynamical System Representation, Generation, and Recognition of Basic Oscillatory Motion Gestures," *Second International Conference on Automatic Face and Gesture Recognition*, Killington, VT, pp. 60-65, October 14-16, 1996.

[42] Y. Cui and J. Weng, "Learning-Based Hand Sign Recognition," *International Workshop on Automatic Face and Gesture Recognition*, Zurich, pp. 201-206, 1995.

[43] T. Darrell and A. Pentland, "Active Gesture Recognition using Partially Observable Markov Decision Processes," *13th IEEE International Conference on Pattern Recognition,* Vienna, Austria, vol. 3, pp. 984-988, August 25-29, 1996.

[44] J.W. Davis and A.F. Bobick, "The Representation and Recognition of Action using Temporal Templates," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 928-934, San Juan, Puerto Rico, June 17-19, 1997.

[45] L. Davis, S. Fejes, D. Harwood, Y. Yacoob, I. Hariatoglu, and M. Black, "Visual Surveillance of Human Activity," *3rd Asian Conference on Computer Vision*, Hong Kong, China, January 1998.

[46] Z. Duric, J.A. Fayman, E. Rivlin, "Function from motion," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 18, No. 6, pp. 579-91, June 1996.

[47] I. Essa, C. Atkeson, G. Abowd, "Captured Experiences Grant Proposal from the National Science Foundation," Center for Graphics, Visualization, and Usability, College of Computing, Georgia Institute of Technology, 1999.

[48] I. Essa and A. Pentland, "Coding, Analysis, Interpretation and Recognition of Facial Expressions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp.757-63, July, 1997.

[49] I. Essa, "Ubiquitous sensing for smart and aware environments," *DARPA/NIST/NSF Workshop on Smart Environments*, Atlanta, GA, July 1999.

[50] I. Essa, "Computers Seeing People," *AI Magazine*, Vol. 20, No. 1, pp. 69-82, Summer 1999.

[51] D.W. Etherington and Reiter, "On inheritance hierarchies with exceptions," *Proceedings AAAI-83: Third National Conference on Artificial Intelligence*, Washington, D.C., August 22-26, 1983.

[52] B. Flinchbaugh, "Networked Cameras," *Proceedings of the 1998 Image Understanding Workshop*, Monterey, CA, Vol. 1, pp. 81-83, November 20-23, 1998.

[53] W.T. Freeman and M. Roth, "Orientation Histograms for Hand Gesture Recognition," *International Workshop on Automatic Face and Gesture Recognition,* Zurich, pp. 296-301, 1995.

[54] N. Friedman and M. Goldszmidt, "Building Classifiers using Bayesian networks," *Proceedings: 13th National Conference on Artificial Intelligence,* Portland, Oregon, pp. 1277-1284, 1996.

[55] W. Gates, Statement taken from Bill Gates, Chairman & CEO of Microsoft Corporation, during the Opening Session, *Microsoft's Professional Developer's Conference,* Denver, October 1998.

[56] D.M. Gavrila and L.S. Davis, "Tracking of Humans in Action: A 3D Model-Based Approach," *Proceedings of the IEEE Computer Vision and Pattern Recognition*, San Francisco, California, 1996.

[57] D.M. Gavrila and L.S. Davis, "Towards 3-D Model-Based Tracking and Recognition of Human Movement: A Multi-View Approach," *International Workshop on Automatic Face and Gesture Recognition,* Zurich, pp. 272-277, 1995.

[58] D.M. Gavila and V. Philomin, "Real-time object detection for *smart* vehicle," *Proceedings of the 7th IEEE Internation Conference on Computer Vision*, pp. 87-93, Vol. 1, September 20-26, 1999.

[59] D.M. Gavila, "The visual analysis of human movement: A survey," *Computer Vision and Image Understanding*, Vol. 73, No. 1, pp. 82-98, January 1999.

[60] D. Gerding, "Action!," *PC/Computing,* Vol. 11, p. 195, March 1998.

[61] Z. Ghahramani, M.I. Jordan, and P. Smyth, "Factorial hidden Markov models," *Machine Learning*, Vol. 29, No. 2-3, Nov.-Dec. 1997.

[62] S. Gutta, H. Huang, F. Iman, and H. Wechsler, "Face and Hand Gesture Recognition using Hybrid Classifiers," *Second International Conference on Automate Face and Gesture Recognition*, Killington, VT, pp. 164-169, October 14-16, 1996.

[63] R. Haralick and L. Shapiro, *Computer and Robot Vision, Volume I,* Addison-Wesley Publishing Co., New York, 1993.

[64] H. Hienz, K. Grobel, and G. Offner, "Real-Time Hand-Arm Motion Analysis using a Single Video Camera," *Second International Conference on Automate Face and Gesture Recognition,* Killington, VT, pp. 323-327, October 14-16, 1996.

[65] J. Hornegger, H. Niemann, and R. Risack, "Appearance-based object recognition using optimal feature transforms," *Pattern Recognition*, Vol. 33, No.2, pp. 209-24, February 2000.

[66] J. Hu, M. Brown, and W. Turin, "HMM based on-line handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 10, pp. 1039-1045, October 1996.

[67] X. D. Huang, Y. Ariki, and M. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh University Press: Edinburgh, 1990.

[68] INSPEC database, *Institution of Electrical Engineers,* Herts, England, January 1983 - September 1998.

[69] S. Intille, J. Davis, and A. Bobick, "Real-Time Closed-World Tracking," *Proceedings of IEEE Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997.

[70] S.S. Intille and A.F. Bobick, "Closed-world tracking," *Proceedings of IEEE International Conference on Computer Vision*, Cambridge, MA, pp. 672-8, 1995.

[71] M.A. Isard, "Contour tracking by stochastic propagation of conditional density", *Proc. European Conference on Computer Vision,* pp. 343-356, Cambridge, UK, April 1996.

[72] M.A. Isard and A. Blake, "CONDENSATION - conditional density propagation for visual tracking", *International Journal on Computer Vision,* Vol. 29, No. 1, pp. 5-28, 1998.

[73] D. Israel, J. Perry, and S. Tutiya, "Actions and Movements," *Proceedings of IJCAI '91*, Sydney, Australia, pp. 1060-1065, 1991.

[74] Y.A. Ivanov, "Application of stochastic grammars to understanding action," *Masters Thesis*, Massachusetts Institute of Technology, February 1998.

[75] Y.A. Ivanov and A.F. Bobick, "Recognition of Multi-agent Interactions in Video Surveillance," *IEEE Proceedings of the International Conference on Computer Vision*, Kerkyra, Greeece, Vol. 1, pp. 169-176, September 20-27, 1999.

[76] R. Jackendoff, *Semantics and Cognition*, The M.I.T. Press, Cambridge, MA, 1990.

[77] A. Jain and C. Dorai, "Practicing Vision: Integration, Evaluation and Applications," *IEEE Transactions on Pattern Recognition*, Vol. 30, No. 2, pp. 183-196, 1997.

[78] K.H. Jo, Y. Kuno, and Y. Shirai, "Manipulative Hand Gesture Recognition Using Task Knowledge for Human Computer Interaction," *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, pp. 468-473, 1998.

[79] F. Jelinek, "Continuous speech recognition by statistical methods," *Proceedings of the IEEE*, Vol. 64, pp. 532-536, April 1976.

[80] D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Foster, G. Tajchman, and N. Morgan, "Using a stochastic context-free grammar as a language model for speech recognition," In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, pp. 189-192, Detroit, MI, May 1995.

[81] C. Kidd, G. Abowd, C. Atkeson, I. Essa, B. MacIntyre, E. Mynatt, and T. Starner, "The Aware Home: A living laboratory for ubiquitous computing research," *2nd International Workshop on Cooperative Building 1999*, Eds. Streitz, J. Siegel, V. Hartkopf, S. Konomi, Pittsburg, PA, Springer: Heidelberg, 1999.

[82] R. Kjeldsen and J. Kender, "Toward the Use of Gesture in Traditional User Interfaces," *Second International Conference on Automate Face and Gesture Recognition*, Killington, VT, pp. 151-156, October 14-16, 1996.

[83] R. Kjeldsen and J. Kender, "Toward the Use of Gesture in Traditional User Interfaces," *Second International Conference on Automate Face and Gesture Recognition*, Killington, VT, pp. 151-156, October 14-16, 1996.

[84] P. Langley, "A model of early syntactic development," *20th Annual Meeting of the Association for Computational Linguistics*, Toronto, Ont., Canada, pp. 145-51, June 16-18, 1982.

[85] J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice Hall, 1990.

[86] E. Littmann, A. Drees, and H. Ritter, "Visual Gesture Recognition by a Modular Neural System," *International Conference on Artificial Neural Networks*, Bochum, Germany, pp. 317-322, July 16-19, 1996.

[87] R. Lunheim and G. Sindre, "Privacy and Computing: A Cultural Perspective," *Security and Control of Information Technology in Society (IFIP TC9/WG9.6 Working Conference)*, Vol. A-43, pp. 25-40, St. Petersburg, Russia, Aug. 12-17, 1993.

[88] C. Maggioni, "GestureComputer - New Ways of Operating a Computer," *International Workshop on Automatic Face and Gesture Recognition*, Zurich, pp. 166-171, 1995.

[89] R. Mann and A. Jepson, "Towards the Computational Perception of Action," *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 794-799, Santa Barbara, CA, 1998.

[90] S. Mann, "*Smart clothing*: wearable multimedia computing and *personal imaging* to restore the technological balance between people and their environments," *ACM Proceedings of 4th Multimedia Conference*, pp. 163-74, Boston, MA, Nov. 18-22, 1996.

[91] D. Marr, *Vision - A Computational Investigation into the Human Representation and Processing of Visual Information*, W.H. Freeman and Co.: San Francisco, 1982.

[92] G.T. Marx,"Ethics for the new surveillance," *Information Society*, pp. 171-85, Vol. 14, No. 3, July-Sept, 1998.

[93] D. McNeil, *Hand and Mind: What Gestures Reveal about Thought,* University of Chicago Press: Chicago, 1992.

[94] C.D. Mitchell, R.A. Helzerman, L.H. Jamieson, and M.P. Harper,"Parallel implementation of a hidden Markov model with duration modeling for speech recognition," *5th IEEE Symposium on Parallel and Distributed Processing*, pp. 298-306, Dallas, TX, Dec. 1-4, 1993.

[95] D. Moore, I. Essa, and M. Hayes, "ObjectSpaces: Context Management for Activity Recognition," *Proceedings of the 2nd Annual Audio-Visual Biometric Person Authentication Conference*, Washington, D. C., March, 1999.

[96] D. Moore, I. Essa, and M. Hayes, "Exploiting Human Actions and Object Context for Recognition Tasks ," *Proceedings of the 7th IEEE International Conference on Computer Vision*, Vol. 1, pp. 80-86, Corfu, Greece, September 20-26, 1999.

[97] G. Moore, "1968 theory: CPU performance doubles every 18 months," *Intel Corporation.*

[98] E.A. Mohammed, "An examination of surveillance technology and their implications for privacy and related issues. The philosophical legal perspective," *Journal of Information Law & Technology*, No.2, June 30, 1999.

[99] M.C. Mozer, "The neural network house: An environment that adapts to its inhabitants," In M. Coen (Ed.), *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, pp. 110-114, Menlo, Park, CA, 1998.

[100] M.C. Mozer, "An intelligent environment must be adaptive," *IEEE Intelligent Systems and their Applications*, Vol. 14, No. 2, pp. 11-13, 1999.

[101] E. Murbridge, *Animal Locomotion*, 1887.

[102] S. Nagaya, S. Seki, and R. Oka, "A Theoretical Consideration of Pattern Space Trajectory for Gesture Spotting Recognition," *Second International Conference on Automate Face and Gesture Recognition,* Killington, VT, pp. 72-77, October 14-16, 1996.

[103] H. Nagel, "From Image Sequences towards Conceptual Descriptions," *Image and Vision Computing*, Vol. 6, No. 2, pp. 59-74, 1988.

[104] V. Nalwa, *A Guided Tour of Computer Vision*, Addison-Wesley Publishing, New York, 1993.

[105] NCR Corp., "Customer Activity Analysis System," NCR Human Interface Technology Center, Atlanta, Georgia, John Ming (Manager), 1997.

[106] NCR Corp., "SCOT: Self-Check Out Terminal," NCR Human Interface Technology Center, Atlanta, Georgia, John Ming (Manager), 1997.

[107] A.V. Nefian and M.H. Hayes, "Embedded HMM-based approach for face detection and recognition, " *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 6, pp. 3553-3556, Phoenix, AZ, March 15-19, 1999.

[108] B. Neumann, "Natural Language Description of Time Varying Scenes," *Semantic Structures*, Lawrence Erlbaum Associates, 1989.

[109] T. Nishimura and R. Oka, "Spotting Recognition of Human Gestures from Time-Varying Images," *Second International Conference on Automate Face and Gesture Recognition,* Killington, VT, pp. 318-322, October 14-16, 1996.

[110] N. Oliver, B. Rosario, and A. Pentland, "Statistical Modeling of Human Interactions," *Proceedings from the Computer Vision and Pattern Recognition*, 1998.

[111] T. Olson, F. Brill,"Moving Object Detection and Event Recognition Algorithms for Smart Cameras," *Proceedings of the 1997 Image Understanding Workshop*, New Orleans, LA, Vol. 1, pp. 159-175, May 11-14, 1997.

[112] V.I. Pavlovic, R. Sharma, and T.S. Huang, "Visual interpretation of hand gestures for human-computer interaction: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No.7, pp.677-95, July 1997.

[113] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* Morgan Kaufmann Publishers, Inc.: San Mateo, California, 1988.

[114] S.D. Personick, "The information age: making the vision real," *First IEEE Enterprise Networking Mini-Conference (ENM-97) in conjunction with the ICC-97*, pp. 16-30, Montreal, Que., Canada, June 11-12, 1997.

[115] C. Pinhanez and A. Bobick, "Human Action Detection Using PNF Propagation of Temporal Constraints," *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 898-904, Santa Barbara, CA, 1998.

[116] D. Poole, A. Mackworth, and R. Goebel, *Computational Intelligence: A Logic Approach*, Oxford University Press: New York, 1998.

[117] F.K.H. Quek and M. Zhao, "Inductive Learning in Hand Pose Recognition," *Second International Conference on Automate Face and Gesture Recognition*, Killington, VT, pp. 78-83, October 14-16, 1996.

[118] L.R. Rabiner, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine*, pp. 4-16, January 1986.

[119] U. Ramachandran, R.S. Nikhil, N. Harel,J.M. Rehg, and K. Knobe, "Space-Time Memory: a parallel programming abstraction for interactive multimedia applications," *Seventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (SIGPLAN Notices), Atlanta, GA, Vol.34, No.8, pp.183-92, May 4-6, 1999.

[120] R. Raman, "Application of hidden Markov modeling to the characterization of transcription factor binding sites," *IEEE Proceedings of the Hawaii International Conference on System Sciences*, Part 5, pp. 275-283, Wailea, HI, Jan. 4-7, 1994.

[121] J.M. Rehg, U. Ramachandran, R. Halstead, C. Joerg, L. Kontothanassis, R. Nikhil, and S.B. Kang, "Space-Time Memory: A Parallel Programming Abstraction for Dynamic Vision Applications," *Techical Report Series*, Cambridge Research Laboratory (CRL/COMPAQ), April 1997.

[122] L.G. Roberts, "Machine Perception of Three-Dimensional Solids," *Optical and Electro-Optical Information Processing*, MIT Press, Cambridge, Massachusetts, pp. 159-197, 1965.

[123] D. Salber, A. Dey, and G. Abowd, "The Context Toolkit: Aiding the Development of Context-Enabled Applications," *GVU Techical Report GIT-GVU-98-33*, Georgia Institute of Technology, 1998.

[124] S. Schafer, "EasyLiving," Prototype Software Architecture, Microsoft Research, 1998.

[125] R.C. Schank, "Conceptual Dependency Theory," in *Conceptual Information Processing*, North-Holland, pp.22-82, 1975.

[126] J. Schlenzig, E. Hunter, and R. Jain, "Vision Based Hand Gesture Interpretation using Recursive Estimation," *IEEE 28th Asilomar Conference on Signals, Systems, and Computers,* vol. 2, pp. 1267-1271, Oct. 30-Nov. 2, 1994.

[127] R. Sharma, V.I. Pavlovic, T.S. Huang, "Toward Multimodal Human-Computer Interface," *Proceedings of the IEEE*, pp. 853-69, Vol. 86, No. 5, May 1998.

[128] J. Sodhi and P. Sodhi, *Object-oriented Methods for Software Development*, McGraw-Hill, New York, 1996.

[129] SPIE Web Site, International Society of Optical Engineers, URL: *http://www.spie.org/web/oer/october/oct97/industryfocus.html#4.*

[130] T. Starner and A. Pentland, "Visual Recognition of American Sign Language using Hidden Markov Models," *Proceedings of the International Workshop on Automatic Face and Gesture Recognition,* Zurich, Switzerland, 1995.

[131] T. Starner, J. Weaver, and A. Pentland, "Real-time American sign language recognition using desk and wearable computer based video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 12, pp. 1371-5, December 1998.

[132] A. Stolcke, "Bayesian Learning of Probabilistic Language Models," *Ph.D. Dissertation*, University of California at Berkeley, 1994.

[133] A. Stolcke, "An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities," *Computational Linguistics*, Vol. 21, No. 2, pp. 165 - 201,June 1995.

[134] A. Stolcke and J. Segal, "Precise n-gram probabilities from stochastic context-free grammars," *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM, pp. 74 - 79, June 26-30, 1994.

[135] T.M. Strat and M.A. Fischler, "Context-based Vision: Recognizing Objects using Information from both 2D and 3D Imagery," *IEEE Transcations on Pattern Analysis and Machine Intelligence,* pp. 1050-1065, Vol. 13, No. 10, 1991.

[136] T.A. Sudkamp, *Languages and Machines: An Introduction to the Theory of Computer Science, Second Edition*, Addison-Wesley: Reading, MA, 1997.

[137] R.G. Taylor, *Models of Computation and Formal Languages*, Oxford University Press, New York 1998, p. 504.

[138] A.M. Tekalp, *Digital Video Processing*, Prentice Hall, 1995.

[139] J.T. Tou and R.C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley: Reading, MA, 1974.

[140] S. Ullman, *High-level Vision: Object Recognition and Visual Cognition*, MIT Press, 1996.

[141] K. Umeda, I. Furusawa, S. Tanaka, "Recognition of hand gestures using range images," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* (Innovations in Theory, Practice and Applications), Vol. 3, pp. 1727-32, Victoria, BC, Canada, Oct. 13-17, 1998.

[142] P. Vamplew and A. Adams, "Recognition and Anticipation of Hand Motions using a Recurrent Neural Network," *IEEE International Conference on Neural Networks*, Perth, WA, Australia, Vol. 6, pp. 2904-2907, Nov. 27-Dec. 1, 1995.

[143] L. Vespremi, "First real-time color capture board: Truevision NuVista expensive but good," *MacWEEK,* Vol. 3, p. 40, August 15, 1989.

[144] M. Weiser, Quote by Mark Weiser, Xerox PARC Senior Researcher, deceased 1999.

[145] M. Weiser, "The Computer for the 21st Century," *Scientific American*, September 1991.

[146] A.D. Wilson and A.F. Bobick, "Recognition and Interpretation of Parametric Gesture," *Proceedings of IEEE 6th International Conference on Computer Vision*, Bombay, India, pp.329-36, Jan. 4-7, 1998.

[147] Y. Yacoob and M. Black, "Parametertized Modeling and Recognition of Activities," *International Conf. on Computer Vision*, Mumbai-Bombay, India, January 1998.

[148] J. Yamato, J. Ohya, and K. Ishii, "Recognizing Human Action in Time-Sequential Images using Hidden Markov Models," *IEEE Conference on Computer Vision and Pattern Recognition*, Champaign, IL, pp. 379-385, June 15-18, 1992.

[149] M.H. Yang and N. Ahuja, "Recognizing hand gesture using motion trajectories, " *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Fort Collins, CO, Vol. 1, pp. 466-72, June 23-25, 1999.

[150] J. Yi and D. Chelberg, "Model-based 3D object recognition using Bayesian indexing," *Computer Vision and Image Understanding*, Vol. 69, No. 1, pp.87-105, January 1998.

Vita

Darnell Janssen Moore attended Tyner High School in Chattanooga, Tennessee, before leaving for Phillips Academy (est. 1778) in Andover, Massachusetts, in his 11th grade year. While at Andover, Darnell involved himself in various organizations including the Afro-Latino-American Society (social coordinator), WPAA student radio (disc jockey & producer for a weekly 2 hour show), and as a tutor for the nearby Lawrence Housing Authority. He was also the recipient of the J. McIntyre scholarship and participated in the Leadership, Education, and Development (LEAD) program held at the Wharton School of Business, University of Pennsylvania, during the summer of 1987.

After graduating in 1988, he enrolled at Northwestern University in Evanston, Illinois, majoring in electrical engineering. While at NU, Darnell served in various organizations, including as president and treasurer for the civic fraternity Alpha Phi Alpha; as a national delegate for the local chapter of the National Society of Black Engineers; as chairperson of the Martin Luther King, Jr. Commemoration Service ('90 & '91); as a WNUR FM disc jockey & producer for a weekly 3 hour show; and as an executive Mayfest Committee member (1991). He also received several honors and awards, including the AT&T Service Award, GM/NSBE's Most Improved (Region IV) Award, and the Frank Hough Memorial Scholarship. Darnell also discovered a passion for various areas within the electrical engineering discipline and helped put together the Signal Processing Laboratory while working with Dr. Janet Rutledge. Summer internships at Ford and 3M compelled him to continue his education after obtaining his Bachelor of Science degree in June of 1992.

Funded by the GEM Master's fellowship, Darnell enrolled in graduate school at the Georgia Institute of Technology in Atlanta, Georgia. He concentrated in areas of digital signal processing, telecommunications, and control systems. After completing an intern

with his mentor, Dr. Fred Bacon at 3M Fiber Optics Laboratory in St. Paul, Minnesota, Darnell completed his Master of Science degree in December of 1993.

After graduation, Darnell accepted a position with Hayes Microcomputer Products, Inc. in Norcross, Georgia, as a Test Engineer. He elected to return for a doctorate in electrical engineering at Georgia Tech in the fall of 1994. Darnell participated in the Black Graduate Student Association as treasurer and sat on the executive committee of the 8th Annual Graduate Student Symposium. During his tenure in the Ph.D. program, Darnell has investigated several topics in signal processing and computer vision, particularly in the activity recognition area. He works with various research groups including the Center for Signal and Image Processing, the Computational Perception Laboratory, and Future Computing Environments. Darnell also garnered substantial industrial experience while completing his terminal degree, working with NCR's Human Interface Technology Center in Atlanta and the Xerox Palo Alto Research Center in Palo Alto, California. Upon graduation, Darnell will join Texas Instruments' DSP Solutions Groups in Dallas, Texas.