VISUAL FEATURE LEARNING

A Dissertation Presented

by

JUSTUS H. PIATER

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2001 (revised June 14, 2001)

Department of Computer Science

VISUAL FEATURE LEARNING

A Dissertation Presented

by

JUSTUS H. PIATER

Approved as to style and content by:

Roderic A. Grupen, Chair

Edward M. Riseman, Member

Andrew H. Fagg, Member

Neil E. Berthier, Member

James F. Kurose, Department Chair
Department of Computer Science

For my daughter Zoé,
born 3 days and 18 hours after my dissertation defense,
her siblings, Bennett and Talia,
and my wife Petra.

I will love you all forever!

ACKNOWLEDGMENTS

Few substantial pieces of work are the achievement of a single person. This dissertation is no exception. Many people have contributed to its development. First of all, this endeavor would have been impossible without the continual support of my wife Petra. She walked many extra miles and backed me during times of highly concentrated effort. I love you! I also thank my three children, who – one after the other – emerged on the scene during my five years of graduate studies at the University of Massachusetts. They always wanted me to be home with them all day. Forgive me for having a job! Bennett in particular has unknowingly made his contributions to this research. The way he learned to recognize hand-drawn shapes during his early years was very inspiring.

I am indebted to my advisor Rod Grupen for many inspiring discussions, unshakable confidence, his vision for task-driven learning by interaction, and for simply being a great guy. Thank you for inviting me into your lab! The first ideas about visual learning grew out of my earlier work with Ed Riseman. He has been a constant source of encouragement, and his keen observations and suggestions have greatly improved the quality of this presentation. Great credit is due to Andy Fagg, who has reviewed the ideas and their presentation with unparalleled scrutiny, and suggested many insightful improvements. Of course, I am to blame for any remaining errors. Thanks to their valuable feedback, this dissertation turned into a picture book that includes 336 individual graphics files in 62 figures.

I am grateful to Paul Utgoff for sharing his insight during numerous discussions of machine learning issues that have shaped my thinking forever. He introduced me to the Kolmogorov-Smirnoff distance that serves several pivotal roles in this work. I deeply regret that we never got around to playing music together! Special thanks go to the psychologists Rachel Clifton and Neil Berthier who introduced me to the world of human perceptual development and learning. This had a strong impact on my research interests, and has greatly enhanced the interdisciplinary aspects of this dissertation. Their sharp insights have added great value to my studies.

On the more mundane side, I acknowledge the helpful support of Hugin A/S. Hey Danes, neighbors! They provided me with a low-cost Ph.D. license of their Bayesian network library for Solaris, and later with a Linux port. This library was used in the implementation of the system described in Chapters 4 and 5. I also want to use this opportunity to express my appreciation of the Free Software / Open Source community, whose efforts have produced some of the best software in existence. Without Linux and many GNU tools, my life would be considerably harder. I wonder how people ever lived without Emacs? Thanks to the excellent LaTeX 2$_\varepsilon$ class file by my fellow graduate student John Ridgway (that builds on code contributed by Rod Grupen, and others), this document looks about as nice as our Graduate School allows ☺.

Many friends and colleagues helped make my time at the UMass Computer Science department very pleasant. The Laboratory of Perceptual Robotics is a wonderful bunch of creative people. Thanks to Elizeth for reminding me of my own birthday so the lab would get their well-deserved cake. She and her co-veterans Jefferson and Manfred take most of the credit for making LPR a great place to hang out. Their successors in the lab now have to start over. And thanks to my wife for baking those cakes. I think they will be remembered longer than I. Thanks to my South-American and Italian friends for restoring my sanity by dragging me away from my desk and onto the soccer field. I apologize to my fellow grads from LPR, ANW, and EKSL, as my floods of long-running experiments left them with few clock cycles on our shared compute cluster. A big thank-you goes to my special friends and brothers in Christ, Craig the scientist

and Achim the musician, for being there for me in my joys and sorrows. How empty would life be without true friends!

Most of all, I want to thank God for calling our world into being. It is so wonderfully rich and complex that no one but You will ever fully understand it. Exploring the why's and how's of physics and life makes for a fulfilled professional life. The more I learn about Your creation, the more I am in awe. My life belongs to You.

O Lord, how manifold are Your works!
In wisdom You have made them all.

Psalm 104:24

ABSTRACT

VISUAL FEATURE LEARNING

FEBRUARY 2001 (REVISED JUNE 14, 2001)

JUSTUS H. PIATER

Dipl.-Inform., UNIVERSITY OF MAGDEBURG, GERMANY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Roderic A. Grupen

Humans learn robust and efficient strategies for visual tasks through interaction with their environment. In contrast, most current computer vision systems have no such learning capabilities. Motivated by insights from psychology and neurobiology, I combine machine learning and computer vision techniques to develop algorithms for visual learning in open-ended tasks. Learning is incremental and makes only weak assumptions about the task environment.

I begin by introducing an infinite feature space that contains combinations of local edge and texture signatures not unlike those represented in the human visual cortex. Such features can express distinctions over a wide range of specificity or generality. The learning objective is to select a *small* number of *highly useful* features from this space in a task-driven manner. Features are learned by general-to-specific random sampling. This is illustrated on two different tasks, for which I give very similar learning algorithms based on the same principles and the same feature space.

The first system incrementally learns to discriminate visual scenes. Whenever it fails to recognize a scene, new features are sought that improve discrimination. Highly distinctive features are incorporated into dynamically updated Bayesian network classifiers. Even after all recognition errors have been eliminated, the system can continue to learn better features, resembling mechanisms underlying human visual expertise. This tends to improve classification accuracy on independent test images, while reducing the number of features used for recognition.

In the second task, the visual system learns to anticipate useful hand configurations for a haptically-guided dextrous robotic grasping system, much like humans do when they preshape their hand during a reach. Visual features are learned that correlate reliably with the orientation of the hand. A finger configuration is recommended based on the expected grasp quality achieved by each configuration.

The results demonstrate how a largely uncommitted visual system can adapt and specialize to solve particular visual tasks. Such visual learning systems have great potential in application scenarios that are hard to model in advance, e.g. autonomous robots operating in natural environments. Moreover, this dissertation contributes to our understanding of human visual learning by providing a computational model of task-driven development of feature detectors.

TABLE OF CONTENTS

Page

CHAPTER

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Humans have a remarkable ability to act reasonably based on perceptual information about their environment. Our perceptual system functions with such speed and reliability that we are deluded into underestimating the complexity of everyday perceptual tasks. In particular, humans rely heavily on visual perception. We orient ourselves, recognize environments, objects, and people, and manipulate items based on vision without ever thinking about it. Given the importance of vision to humans, it is not surprising that vision has been the most-studied mode of machine perception since the early days of artificial intelligence. Nevertheless, despite fifty years of active research in artificial intelligence, robotics and computer vision, many real-world visuomotor tasks remain that are easily performed by humans but are still unsolved by machines. The robustness and versatility of biological sensorimotor interaction cannot yet be matched in robotic systems.

What is it that enables higher animals, first and foremost humans, to outperform machines so dramatically on real-world visuomotor tasks? I believe that the answer is grounded in the following two theses that form the basis of this dissertation:

- The human visual system is *adaptive.* During the first years of life, the visual capabilities of children increase dramatically. These capabilities are not limited by the design of the visual system alone, but are modified by learning. All throughout life, the human visual system continues to improve performance on both novel and well-practiced tasks.

  In contrast, most current machine vision systems do not learn in this way. They are designed to perform, and their performance is limited by the design. They do not usually improve over time or adapt to novel situations unforeseen by the designer.

- The human visual system is inextricably linked with human *activity.* Activity operates in synergy with vision and facilitates visual learning, and vision subserves activity. Human vision operates in a highly task-dependent way and is tuned to deliver exactly the information needed.

  In contrast, most research in computer vision has focused on task-independent visual functionality. In a typical scenario, a computer vision system produces a generic result or representation for use by subsequent processing stages.

Both of these points will be further discussed in Chapter 2. The remainder of this opening chapter serves to define the scope and organization of this dissertation.

1.1 Closed and Open Task Domains

The field of computer vision is commonly subdivided into low-level and high-level vision. Low-level vision is typically concerned with task-independent image analysis such as edge extraction or computation of disparity or optical flow. High-level vision considers application-level problems, e.g. object recognition or vision-guided grasping. Considerable progress has been made in both areas during the past decade. For example, machine recognition systems have achieved unprecedented levels of performance [73, 104, 71, 77, 78]. Increasingly impressive recognition results are reported on large databases of various objects. Automated character recognition systems are involved in sorting most of the U.S. mail. Optical biometric identification systems have reached commercial maturity.

While these successful systems are truly remarkable, most of them are designed for tasks that are limited in scope and well defined at design time. For instance, most object recognition systems operate on a fixed set of known objects. Many algorithms in fact require access to a complete set of training images during a dedicated training phase. Deployed OCR and biometric identification systems operate under highly controlled conditions where parameters such as size, location, and approximate appearance of the visual target are known. Similar arguments can be made for other computer vision problems such as face detection, terrain classification, or vision-guided navigation.

Task domains that share these characteristics I call *closed.* I argue that many practical vision problems are not closed. For instance, a human activity recognition system should be able to operate under many different lighting conditions and in a variety of contexts; indoors or outdoors, with any number of people in the scene. A visually navigated mobile robot should be able to learn distinctive landmarks by itself. If the robot is moved from one environment to another, one does not want to redesign the recognition algorithm – the same algorithm should be applicable in, and adaptive to, a variety of environments. An autonomous robot that traverses unknown terrain or collects specimens should be able to learn to predict the effect of its actions based on perceptual information in order to improve its actions with growing experience. Ultimately, it should be the interaction of an agent with its environment – as opposed to a supervisory training signal – that drives the formation of perceptual capabilities while performing a task [67, 114].

Table 1.1. Typical characteristics of closed vs. open task domains.

	Closed Tasks	*Open Tasks*
Task parameters:	all known at the outset	some to be discovered
	stationary	may be non-stationary
Training data:	fixed	dynamic, generative
	fully accessible	partially accessible via interaction
Learning:	batch	incremental
	off-line	on-line
Visual features:	may be fixed	must be learned

Thus, many realistic visual problems constitute *open* task domains (see Table 1.1). Closed and open tasks constitute two extremes along a continuum of task characteristics. Open tasks are characterized by parameters that are unknown at design time and that may even change over time. Therefore, the perceptual system of the artificial agent cannot be completely specified at the outset, but must be refined through learning during interaction with the environment. There is no fixed set of training data, complete or otherwise, that could be used to train the system off-line. Training information is available in small amounts at a time through interaction of the agent with its environment. Therefore, learning must be on-line and incremental.

1.2 Scope

Autonomous robots that perform nontrivial sensorimotor tasks in the real world must be able to learn in both sensory and motor domains. A well-established research community is addressing issues in motor learning, which has resulted in learning algorithms that allow an artificial agent to improve its actions based on sensory feedback. Little work is being conducted in sensory learning, here understood as the problem of improving an agent's perceptual skills with growing experience. The sophistication of perceptual capabilities must ultimately be measured in terms of their value to the agent in executing its task.

This dissertation addresses a subset of the problems outlined above. At a broad level, its goal is to make progress toward computational models for perceptual learning in open task domains. While most work in computer vision has focused on closed tasks, the following chapters present learning methods for open tasks. These methods are designed to be very general. They make few prior assumptions about the tasks, and can learn incrementally and on-line. I hope that this work will spark new research aimed at expanding the scope of machine perception and autonomous robots to increasingly open task domains.

A key unit of visual information employed by biological and machine vision systems is a *feature*. Loosely speaking, a visual feature is a representation of some aspect of local appearance, e.g. a corner formed by two intensity edges, a spatially localized texture signature, or color. Most current feature-based machine vision systems employ hand-crafted feature sets. In the context of open tasks, I argue that learning must take place even at the level of visual feature extraction. Any fixed, finite feature set would constrain the range of tasks that can be learned by the agent. This does not necessarily mean that the features cannot be computed by a fixed operator set. If they are, then these operators must be suitably parameterized to provide for the flexibility and adaptability demanded by a variety of open tasks.

The key technical contribution of this dissertation consists of methods for learning visual features in support of specific visual or visuomotor tasks. These features capture local appearance properties and are sampled from an infinite feature space. Most parameters of the system are derived on-line using probabilistic methods. The validity of the proposed methods are demonstrated using two very different example skills, one based on categorization (visual discrimination) and the other on regression (visual support of haptically-guided grasping). The sampling methods for feature generation and the associated model-fitting techniques can be adapted to other visual and non-visual perceptual tasks.

This work constitutes exploratory research in an area where computer vision and machine learning meet. This area has received relatively little attention by these subdisciplines of computer science that are relatively distinct even though both grew out of the artificial intelligence community. While the research questions are addressed primarily from the perspective of computer vision and machine learning, much of the motivation is drawn from observations in psychology and visual neuroscience. Parts of the model account for relevant aspects of the function or performance of the human visual system. The behavior of the model resembles critical aspects of phenomena in human learning.

1.3 Outline

The next chapter places this work into the context of research in psychology and artificial intelligence and discusses the motivation, goals and means against the backdrop of related research within and outside of computer science. Chapters 3–6 constitute the heart of this dissertation. These technical chapters share the same general structure: A review of related work is followed by a precise problem statement, a presentation of the contributed solution, experimental results where applicable, and a discussion. In these chapters, a section titled “Background” briefly introduces prerequisite concepts, terminology, and notation.

- Chapter 3 is self-contained and defines an infinite feature space that is defined to overcome the limitations of finite feature sets for open learning tasks. Features are constructed hierarchically by composing primitive features in specific ways. The learning algorithms discussed in subsequent chapters are based on this feature space.

- Chapter 4 describes a system for learning features in an open recognition task, building on the feature space defined in the preceding chapter. To reinforce the point of learning the features themselves, the algorithm is biased to find few but highly distinctive features. Probabilistic pattern classification is combined with information-theoretic feature selection.

- Chapter 5 extends the previous chapter by introducing the concept of learned visual expertise in a way that resembles human expert visual behavior. In both cases, a visual expert exhibits faster and more reliable recognition performance than a non-expert. It is conjectured that superior features underly expertise, and a method is introduced for continuing to learn improved features with growing experience by the learning system.

- Chapter 6 describes a system for learning features to support a haptically-guided robotic grasping process. Features are learned that enable reliable initializations of hand configurations before the onset of the grasp, resembling human reach-and-grasp behavior. This chapter again builds on the feature space introduced in Chapter 3, but is otherwise self-contained.

Finally, Chapter 7 concludes with a general discussion of the impact of this work and future directions.

CHAPTER 2

THIS WORK IN PERSPECTIVE

The questions addressed in this dissertation span a wide range of disciplines in- and outside of computer science. This chapter places these questions into the context of other research in perceptual skill learning in psychology and artificial intelligence. Motivated by this interdisciplinary background, I pose a general research challenge that is larger than the scope of this dissertation. Finally, I state some important personal preferences and biases – many of which are motivated by insights from psychology and neurobiology – that influenced many of the design choices presented in the subsequent technical chapters.

2.1 Human Visual Skill Learning

The human visual system is truly remarkable. It routinely solves a wide variety of visual tasks with such reliability and deceiving ease that belittles their actual difficulty. These spectacular capabilities appear to rest on at least two foundations. First, the human brain devotes *enormous computational resources* to vision: About half of our brain is more or less directly involved in processing visual information [54]. Second, essential visual skills are *learned* in a long process that extends throughout the first years of an individual's life. At the lowest level, the formation of receptive fields of neurons along the early visual pathway is likely influenced by retinal stimulation. Some visual functions do not develop at all without adequate perceptual stimulation within a *sensitive period* during maturation, e.g. stereo vision [12, 44]. Higher-order visual functions such as pattern discrimination capabilities are also subject to a developmental schedule [40]:

- Neonates can distinguish certain patterns, apparently based on statistical features such as spatial intensity variance or contour density [103].

- Infants begin to note simple coarse-level geometric relationships, but perform poorly in the presence of distracting cues. They do not consistently pay attention to contours and shapes [101].

- At the age of about two years, children begin to discover fine-grained details and higher-order geometric relationships. However, attention is still limited to "salient" features [123].

- Over much of childhood, humans learn to discover distinctive features even if they are overshadowed by more salient distractors.

There is growing evidence that even adults learn new features when faced with a novel recognition task [107]. In a typical experiment, subjects are presented with computer-generated renderings of unfamiliar objects that fall into categories based on specifically designed but unobvious features. Before and after learning the categorization, the subjects are asked to delineate what they perceive to be characteristic features of the shapes. Before learning, the subjects show little agreement in their perception of the features. However, after learning most subjects point out those features that by design characterize the categories [108, 123].

Schyns and Rodet demonstrated convincingly that humans learn features in task-driven and task-dependent ways [109]. Subjects were presented with three categories of “Martian cells”, two-dimensional grayscale patterns that loosely resemble biological cells containing intracellular particles. The first category was characterized by a feature X, the second by a feature Y, and the third by a feature XY, which was a composite of X and Y. Subjects were divided into two groups that differed in the order that they had to learn the categories. Subjects in one group first learned to discriminate categories X and Y and then learned category XY, whereas the other group learned XY and X first, then Y. After learning the categorization, the subjects were asked to categorize other “Martian cells” that exhibited controlled variations of the diagnostic features. Their category assignments revealed the features used for categorization: Subjects of the first group learned to categorize all objects based on two features (X and Y), whereas the subjects of the second group learned three features, not realizing that XY was a compound consisting of the other two. Evidently, feature generation was driven by the recognition task.

Feature learning does not necessarily stop after learning a concept. Tanaka and Taylor [120] found that bird experts were as fast to recognize objects at the subordinate level (“robin”) as they were at the basic level (“bird”). In contrast, non-experts are consistently faster on basic-level discriminations as compared to subordinate-level discriminations. Gauthier and Tarr [38] trained novices to become experts on unfamiliar objects and obtained similar results. These findings indicate that the way experts perform recognition is qualitatively different than novices. It has been suggested that experts have developed specialized features, facilitating rapid and reliable recognition in their domain of expertise [107].

Despite this accumulated evidence of visual feature learning in humans, little is known about the mechanisms of visual learning. At least, recent neurophysiological and psychological studies have shed some light on what the features represent [129]. The bulk of the evidence points to view-specific appearance-based representations in terms of local features. The view-dependence of human object recognition has been firmly established [122]. Recognition performance decreases as a function of viewpoint disparity from previously learned views. In the light of this evidence, Wallis and Bülthoff dismiss recognition theories based on geometric models – two- or three-dimensional – by declaring that “there remains little or no neurophysiological evidence for the explicit encoding of spatial relations or the representation of geon primitives” [129].

The strong viewpoint dependency of human visual representations is even more apparent in the context of spatial orientation [11]. Here, the representation employed by the brain is clearly based on a viewer-centered perceptual reference frame. Abstract spatial reasoning is a cognitive skill that requires extensive training, involving multiple perceptual modalities as well as physical activity [1, 2, 95, 97].

Few definitive statements can be made about the spatial extent of the features used by the visual brain. Nevertheless, it has been shown that even for the recognition of faces – often cited as a prime example of holistic representations – local features play a major role. Solso and McCarthy generated artificial face images by composing facial features taken from real face images. Subjects regarded these artificial faces as highly familiar if the constituent features were taken from known faces, even though the complete (artificial) faces had never been seen before [113].

2.2 Machine Perceptual Skill Learning

For the purpose of this work, machine learning is concerned with methods and algorithms that allow an artificial sensorimotor agent to improve or adapt its actions over time, based on perceptual feedback from the environment that is acted upon. This definition stresses that learning is task-driven and incremental. It excludes non-incremental mechanisms for discovering structure in data such as many conventional classification and regression algorithms, typically considered machine learning methods. Nevertheless, such algorithms may form important components of the type of learning methods of interest here.

For a sensorimotor agent, deriving the next action to be performed involves the following two steps:

1. Analyze the current set of sensory input to extract information – so-called *features* – suitable for action selection.

2. On the basis of these features, and possibly other state information, derive the next action to be taken.

This dichotomy is somewhat idealized, as many machine learning algorithms involve transformations of the feature set, e.g. feature selection or principal component analysis. Most work on machine learning has focused on the second step, with the goal of identifying improved methods for generalizing from experience given a set of (possibly transformed) features. In contrast, a mechanism for *perceptual learning* focuses on improving the extraction of features. The following paragraphs touch on a few representative examples of perceptual learning systems. Previous work related to specific methods and problems will be discussed in later chapters.

The goal of Draper et al.'s ADORE system is to learn reactive strategies for recognition of man-made objects in aerial images [29]. The task is formulated as a Markov decision process, a well-founded probabilistic framework in which a *policy* maps perceptual *states* to *actions*. In ADORE, an action is one of a set of image processing and computer vision operators. The output data produced by taking an action characterize the state, and the policy is built using reinforcement learning techniques [115]. ADORE learned reactive policies superior to any static sequence of operators assembled by the authors.

Steels and his collaborators investigate the problems of perceptual categorization and language acquisition by a group of communicating agents. Sensory information is available to an agent in the form of continuous-valued "streams" [114], possibly extracted from live video [27]. An agent can decide on a "topic" characterized by certain ranges of values in a subset of the sensory channels, and can invent "words" to designate this topic. The agents interact in the form of "language games" in which one of them chooses a known topic or invents a new one, and utters the corresponding word(s). A listening agent matches the utterance with its current sensory input. If the utterance does not match the listening agent's concept of the words, or if it contains unknown words, the agent refines its sensory characterization associated with the words. This is done by successively subdividing sensory ranges and choosing additional sensory channels if necessary. The sensory categorization thus formed is adequate for the current topic, but it does not necessarily match the concept of the speaker. However, over the course of many language games the agents form a shared vocabulary through which they can communicate about what they perceive.

A similarly motivated mechanism for discovering informative structure in sensory channels was used by Cohen et al. [25]. Here, concepts are formed that relate perceptions to the agent's interaction with its environment, in contrast to Steels' social communication.

In these two examples, perceptual distinctions are learned by carving up the perceptual space into meaningful regions. This can be done successfully as long as the information contained in an instantaneous perception is sufficient to make these distinctions. In many practical cases, however, world states that require different actions by the agent may not be distinguishable on the basis of an instantaneous perception. In this case, the recent history of percepts may allow the disambiguation of these perceptually aliased states. This is the basic idea underlying McCallum's Utile Distinction Memory and Utile Suffix Memory algorithms [68, 69]. In addition to resolving hidden state, his U-Tree algorithm [67] performs selective perception by testing which perceptual distinctions are meaningful.

Temporal state information is also the basis of Coelho's system for learning haptically-guided grasping strategies [22, 23]. Grasping experience is recorded as trajectories in a phase space. These trajectories are clustered and represented by parametric models that define discrete states of the grasping system during its interaction with a target object. A reinforcement learning procedure is used to select appropriate closed-loop grasp controllers at each of these states.

2.3 Objective

Draper's ADORE system [29] is a rare example of perceptual strategy learning. The other systems cited in the previous section perform perceptual learning by subdividing the feature space spatially [114, 27, 25, 67] and/or temporally [67, 22, 23], but they do not learn the features themselves.

It is often argued that multilayer neural networks learn features that are represented by the nodes of the hidden layer(s). Likewise, the basis vectors extracted by eigen-subspace decomposition and similar methods can be regarded as learned features. These methods essentially project the input space into another space, typically of lower dimension. However, they provide little control over the properties of the resulting features with respect to criteria external to the projection method. For example, in computer vision one may want to preserve locality of features or invariance to various imaging conditions. This motivates the use of other types of features that are not based on subspace decomposition, but exhibit the desired properties by design.

The central, technical contribution of this dissertation is a mechanism for learning such features. These are extracted directly from the raw image data, and express image properties that are very hard to discover using general techniques for dimensionality reduction. Features are chosen from a very large *feature space.* The specification of this space constrains the structure of the learned features. Thus, it is possible to bias or limit the learning system to features that have specific representational or invariance properties. For example, one may want features to encode corners under rotational invariance.

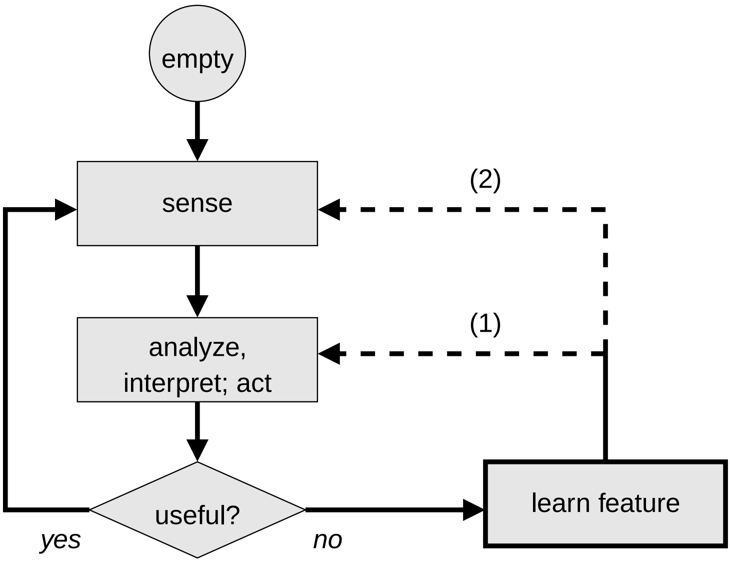


Figure 2.1. A general model of incremental feature learning.

The feature learning procedure is incremental and is suitable for interactive learning. Features are learned as demanded by a given task. Figure 2.1 depicts a general model of the interaction of the system with its environment. The system starts from a blank slate, and is constrained only by the specification of the feature space and the feature learning algorithm. It operates in a perception-action cycle that begins with the acquisition of an image. This image is analyzed in order to derive a decision or action in response to the current perception. It is assumed that the system can observe the quality or usefulness of this decision or action. If this response reveals the adequacy of the action, the agent proceeds with the next perception-action loop. Otherwise, one or more new features are generated with the goal of finding one that will result in superior future actions. The dashed lines designate two alternative ways of operation:

1. If the features can be evaluated immediately without changing the environment, then the agent can iterate feature learning and decision making until a suitable feature has been found. The recognition system described in Chapter 4 operates in this way. When the system misrecognizes an image, new candidate features are generated one at a time, and are evaluated immediately by rerunning the recognition procedure on the same image. This is indicated by the lower dashed line in Figure 2.1.

2. If a feature cannot be evaluated immediately, then the agent proceeds with the next perception-action loop. In this case, the generated features are evaluated over time. In the grasping application (Chapter 6), immediate feature evaluation would require regrasping of an object. To avoid this interference with the task context, after generating new candidate features the system instead proceeds with the next object, as indicated by the upper dashed line. Evaluation of the newly sampled candidate features is distributed over many future grasps.

These two applications differ significantly not only in their learning mechanisms, but also in their learning objectives. The recognition application is a supervised classification task. The utility of a feature is immediately assessed in terms of its contribution to the classification procedures. In contrast, the grasping application is primarily a regression task, where features are learned that predict a category-specific angular parameter. The utility of a feature depends on its contribution to the value of future behavior of the robot. Training is grounded in the robot's interaction with the grasped object, and does not involve an external supervisor. Nonetheless, both of these applications employ the same feature space, introduced in the following chapter, and their feature learning algorithms are based on the same basic principles. These differences and similarities are summarized in Table 2.1.

Table 2.1. Summary of important characteristics of the two applications discussed in this dissertation.

<table>
<tr><th>Recognition (Ch. 4)</th><th>Grasping (Ch. 6)</th></tr>
<tr><td>classification</td><td>regression</td></tr>
<tr><td>external supervisor</td><td>learning grounded in interaction</td></tr>
<tr><td>immediate feature evaluation</td><td>features evaluated over time</td></tr>
<tr><td colspan="2">same feature space</td></tr>
<tr><td colspan="2">same principles for feature learning</td></tr>
</table>

All learning algorithms presented in the following chapters are designed to operate incrementally. They are very uncommitted to any specific task, and make few assumptions about the task environment. For example, the recognition algorithm does not know how many known object categories – if any – are present in any given scene. This is in contrast to most existing recognition algorithms that always assign one (or no) class label to each scene. Also, neither application has any prior knowledge about the number or nature of the categories to be learned. In short, the algorithms do not assume a *closed world*, which has many implications on their design. A consistent set of learning algorithms for *open* domains constitute the second key contribution of this dissertation.

2.4 Motivation

In addition to the objectives stated in the preceding section, there are additional, somewhat less tangible biases and motivations that influenced the design of the representations and algorithms presented in the following chapters. These are introduced briefly here.

The primary goal of this dissertation is to contribute to our understanding of mechanisms of perceptual learning in humans that can be applied in machine vision systems. Many of the ideas manifested in this work are motivated by insights from perceptual psychology. At the outset, I believe that humans (children and adults) learn visual features as demanded by tasks they encounter. Above I presented evidence collected by Schyns and others in support of this belief. A subgoal of this work is to contribute to our understanding of human vision, by devising plausible computational models that describe certain aspects of human vision. I firmly believe that we can learn a great deal from biology for the purpose of advancing technology in the service of human interests. Specifically, I believe that task-driven, on-line, incremental visual learning is essential for building sophisticated applied vision systems that operate in general, uncontrolled environments.

With this long-term objective in mind, the mechanisms contributed are applicable in scenarios where no external supervisor is available. While the basic learning framework is supervised, the supervisory signal can be produced by the agent itself. In subsequent chapters, I will show how this can be done.

The construction of the algorithms and the prototype implementation involved a number of design choices, many of them concerning inessential details. Wherever reasonable, choices were made to emulate biological vision. Some of the resulting algorithms are very efficiently implemented on massively parallel neural hardware, but are quite expensive when run on a conventional serial computer.

CHAPTER 3

AN UNBOUNDED FEATURE SPACE

The necessity for a very large feature space has been motivated briefly above. This chapter introduces a particular feature space suitable for the objectives of this work.

3.1 Related Work

The idea of representing data in terms of features from an infinite feature set (i.e., a feature space) is quite old. A natural method is Principal Component Analysis (PCA) that returns the eigenvectors of the covariance matrix of the data. These can be regarded as a small set of "features" selected from an N-dimensional continuous space, where N is the dimensionality of the data. If the data resemble a low-dimensional zero-mean Gaussian cloud in this space, then they can be faithfully reconstructed using only a few features corresponding to the largest eigenvalues. Turk and Pentland [125] popularized these *eigen-features* for face recognition. Figure 3.1 shows the eight principal components of a set of faces. Interestingly, some of them correspond to intuitively meaningful facial features.

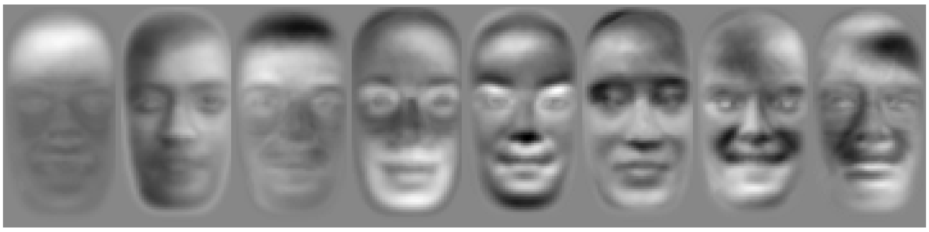


Figure 3.1. "Eigen-faces" – eigenvectors corresponding to the eight largest eigenvalues of a set of face images. The brightness of each pixel represents the magnitude of a coefficient. Zero-valued coefficients are rendered in intermediate gray; large positive and negative coefficients in white and black, respectively. (Reproduced with permission from Moghaddam and Pentland [72].)

Nayar, Murase and collaborators [74, 73, 75] developed this general approach for view-invariant object recognition, tracking and simple robotic control. It has been pointed out that PCA is not necessarily well suited for discrimination between object classes [117]. Several adaptations of the idea of PCA to generate discriminative (as opposed to descriptive) features have been suggested, such as the Fisher Linear Discriminant [31], the Fukunaga-Koontz transform [37], and the Most Discriminating Feature [117]. Talukder and Casasent [118, 119] suggest an adaptation of nonlinear PCA that simultaneously optimizes the descriptive and discriminative power of the extracted features. Some types of eigen-features can be learned and updated incrementally [130].

Hidden nodes in a neural network may be seen as computing features from an infinite set. In particular, neural networks can implement several projection pursuit methods as well as PCA in biologically plausible ways [47, 48, 91]. Projection pursuit iteratively seeks low-dimensional projections of high-dimensional data that maximize a given projection index. The projection index encodes some measure of "interestingness" of the data, typically based on the deviation from Gaussian normality [59, 36, 99]. The projections generated by a projection index based on second-order statistics correspond to the principal components of the data.

All of these methods, with the exception of certain types of neural networks, compute *global* features. In contrast, it is often desirable to produce *local* features that represent spatially localized image characteristics. Localized variants of eigen-features have been explored [26].

A variety of local features have been proposed in the context of recognition systems. In a typical approach, a feature is a vector of responses to a set of locally applied basis filter kernels. Koenderink [57] suggests that the neighborhood around an image point can be represented by a set of Gaussian derivatives of various orders, termed a *local jet*, derived from a local Taylor series expansion of the image intensity surface. Some related schemes are based on steerable bases of Gaussian-derivative filters [90, 104, 94, 43], which will be briefly discussed in Section 3.3.2. Others employ Gabor filters [128, 20], curved variants known as *banana wavelets* [58], or Haar wavelets [81] to represent local appearance.

A different way to generate an unbounded feature space is by defining a feature as a parameterized composition of several components. Cho and Dunn [18] define a corner feature by the distance and angle between two straight line segments. Their feature set is finite because these parameters are quantized.

Segen [110] was one of the first to consider an infinite combinatorial feature space in a computer vision context. His shape recognition system is based on structural compounds of local features. Geometric relations between local curvature descriptors are represented by a multilevel graph, and concept models of 2-D shapes are constructed by matching and merging multiple instances of such graphs. Califano and Mohan [16] combined triplets of local curvature descriptors to form a multidimensional index used for recognition of 2-D shapes. Their contributions include a quantitative probabilistic argument demonstrating that the discriminative power of features increases with their degrees of freedom.

Amit, Geman and their coworkers create a potentially infinite variety of features from a quantized space by means of combinatorics. Primitive local features are composed to produce increasingly complex arrangements. Primitive features are defined by co-occurrences of small numbers of binarized pixel values, and compounds are characterized by relaxed geometric relationships. Discriminative features are constructed and queried efficiently using a decision tree procedure. This type of approach has been applied to handwritten character recognition [8, 5] and face detection [7, 6] with remarkable success, and has also been extended to recognition of three-dimensional objects [50]. The approach employed in this dissertation borrows and extends key ideas from this work.

3.2 Objective

The representational power of any system that uses features to represent concepts is determined to a large extent by the available features. If the feature set is not sufficiently expressive for the task at hand, the performance of the system will be limited forever. If the feature set is too expressive, the system will be prone to a variety of problems, including overfitting, and to biases introduced through pairwise correlated features or distractor features that do not correlate well with important distinctions. Clearly, different features may be relevant for different tasks. Moreover, given a task, different feature sets may be suitable for different algorithms employed to solve the same task [52].

These considerations require that features either be hand-crafted for a particular setting (consisting of task, algorithm and data), or that they be learned by the system. This chapter defines a feature space for use with a feature learning system, subject to the following objectives:

1. As this work is concerned with visual tasks, features are computed on intensity images.

2. The feature space should not be geared toward any task, but it should contain features that are suitable for a variety of visual tasks at various levels of specificity and generality. In other words, there should be some features that are highly useful for any of a variety of tasks that the system may be asked to solve.

3. Since the characteristics of a feature learning system are best demonstrated on a system that learns few but powerful features, the feature space should contain features that are individually very useful for given tasks. This is in contrast to most existing feature-based vision systems that derive their power primarily from a large *number* of features that are not necessarily very powerful individually.

4. The feature space should be sufficiently complete to demonstrate success on example tasks using feature learning systems discussed in subsequent chapters.

5. Since image-plane orientation is important in some tasks and irrelevant in others, features should inherently allow the measurement of feature orientation in a way that can be exploited to achieve normalization for rotation.

6. Since in many visual tasks there is no prior scale information available, features should be robust to variations in scale to simplify computation.

7. To enhance robustness to clutter and occlusion and to facilitate the learning of categories characterized by unique object parts, features should be local in the image array.

8. For generality, features should be applicable in general classification and regression frameworks.

9. The features should be a plausible functional model of relevant aspects of the human visual system.[1]

All of the methods discussed in the preceding section satisfy some of these objectives, but none of them satisfies all. Eigen-features and their variations are well suited for a variety of tasks, but rotational invariance and image-plane locality are hard to achieve. Wavelets and Gabor filters and their variants are local, but are not rotationally invariant. Filters based on steerable Gaussian-derivative filters satisfy all of the above objectives. The steerability property, to be explained in Section 3.3.2, allows the synthesis of filters and their responses at arbitrary orientations from a small set of basis filters, and can be exploited to achieve rotational invariance at very little computational cost. However, the representational and distinctive power of individual, well-localized (Objective 7) Gaussian-derivative filters is not sufficient for many tasks (Objective 3). Amit and Geman [8, 5] used relatively weak individual features, sparse binary pixel templates, and showed how to increase their power by composing them in a combinatorial fashion.

The key idea to achieving all of the above objectives is to combine steerable Gaussian-derivative filters with an adaptation of Amit and Geman's idea, producing features that consist of spatial combinations of local appearance characteristics. Before the details are presented, the following section provides some helpful background.

3.3 Background

This section presents brief introductions to relevant concepts that have been well established in the literature. Some familiarity with these concepts is required to understand the contributions presented in subsequent sections.

[1] Since this contradicts Objective 5, rotational (non-)invariance is not considered a relevant aspect here.

3.3.1 Gaussian-Derivative Filters

The isotropic zero-mean Gaussian function with variance σ^2 of a two-dimensional parameter space at a point $\mathbf{x} = [x, y]^T \in R^2$ is defined as

$$G(\mathbf{x}, \sigma) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{\mathbf{x}^T\mathbf{x}}{2\sigma^2}}. \tag{3.1}$$

For the purposes of this work, the oriented derivative of a Gaussian of order d at orientation $\theta = 0$, i.e. in the x direction, is written as

$$G_d^0(\mathbf{x}, \sigma) = \frac{\partial^d}{\partial^d x} G(\mathbf{x}, \sigma). \tag{3.2}$$

For general orientations θ, G_d^θ is defined as an appropriately rotated version of $G_d^0(\mathbf{x}, \sigma)$, i.e.

$$G_d^\theta(\mathbf{x}, \sigma) = G_d^0(R_\theta \mathbf{x}, \sigma) \tag{3.3}$$

with a rotation matrix

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Gaussian-derivative filter kernels are discrete versions of Gaussian point spread functions computed in this way. To generate a filtered version $I_G(\mathbf{x})$ of an image $I(\mathbf{x})$, a discrete convolution with a kernel G is performed. Two-dimensional convolutions with Gaussians and their derivatives can be computed very efficiently because the kernels are separable, and highly accurate and efficient recursive approximations exist [127]. Unfortunately, this is not generally true of rotated derivatives (Equation 3.3).

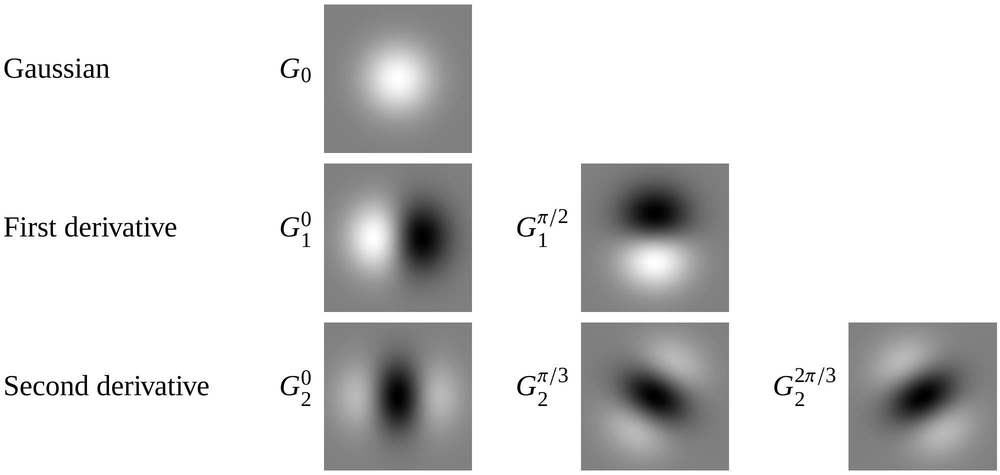


Figure 3.2. Visualization of a two-dimensional Gaussian function and some oriented derivatives (cf. Equation 3.3). Zero values are shown as intermediate gray, positive values are lighter, negative values darker.

Figure 3.2 illustrates some oriented Gaussian-derivative filters used in the context of this work. Gaussian filters act as low-pass filters, and their derivatives as bandpass filters. The center frequency increases (see Figure 3.2) and the bandwidth decreases as the order of the derivative increases. First-order derivative kernels respond to intensity gradients in I. To extract edge information from an image I, I is convolved with the two orthogonal basis filters G_1^0 and $G_1^{\pi/2}$. The gradient magnitudes and orientations in I are then easily computed:

$$|\nabla I| \quad = \quad \sqrt{I_{G_1^0}^2 + I_{G_1^{\pi/2}}^2} \tag{3.4}$$

$$\tan\theta_{\nabla I} \quad = \quad I_{G_1^{\pi/2}} \;/\; I_{G_1^0} \tag{3.5}$$

When computing $\theta_{\nabla I}$ using Equation 3.5, the full angular range of 0–2π can be recovered by taking into account the signs of the numerator and the denominator when computing the arctangent. For a thorough discussion of Gaussian-derivative filters see a recent book chapter [93].

3.3.2 Steerable Filters

A class of filters is *steerable* if a filter of a particular orientation can be synthesized as a linear combination of a set of *basis filters* [35]. For example, first-order Gaussian-derivative filters are steerable, since

$$G_1^\theta = G_1^0 \cos\theta + G_1^{\pi/2} \sin\theta \tag{3.6}$$

as is easily verified. Due to the linearity of the convolution operation, an image filtered at any orientation can be computed from the corresponding basis images:

$$I_{G_1^\theta} = I_{G_1^0} \cos\theta + I_{G_1^{\pi/2}} \sin\theta \tag{3.7}$$

which is much more efficiently computed for many different values of θ than by explicit convolution with the individual filters G_1^θ.

Gaussian-derivative filters of any order d are steerable using $d + 1$ basis filters $G_d^{\theta_{k,d}}$ that are equally spaced in orientation between 0 and π, i.e.,

$$\theta_{k,d} = \frac{k\pi}{d+1}, \qquad k = 0, \ldots, d.$$

Incidentally, Figure 3.2 shows the basis filters $G_d^{\theta_{k,d}}$ for the first two derivatives. To synthesize a filter at any given orientation θ, these basis filters are combined using the rule

$$G_d^\theta = \sum_{k=0}^{d} G_d^{\theta_{k,d}} c_{k,d}^\theta \tag{3.8}$$

where

$$\begin{array}{lcll} c_{k,1}^\theta & = & \cos(\theta - k\pi/2) & k = 0, 1 \\ c_{k,2}^\theta & = & \frac{1}{3}\big(1 + 2\cos(2(\theta - k\pi/3))\big) & k = 0, 1, 2 \\ c_{k,3}^\theta & = & \frac{1}{2}\big(\cos(\theta - k\pi/4) + \cos(3(\theta - k\pi/4))\big) & k = 0, 1, 2, 3 \end{array}$$

which contains Equation 3.6 as a particular case [35, 90]. Again, due to the linearity of convolution, the same operation can be performed on the filtered images, as opposed to the filters themselves. See Freeman and Adelson [35] for a thorough treatment of steerable filters.

Example 3.1 (Operations on Gaussian-filtered images) Figure 3.3 illustrates some manipulations on filtered images that are of interest in this context. The image shown in Figure 3.3a is convolved with the kernels G_1^0 and $G_1^{\pi/2}$, resulting in the vertical and horizontal edge images shown in Figures 3.3b and 3.3c. Edge energy is computed directly from these images using Equation 3.4 (Figure 3.3d), and edge orientation using Equation 3.5 (Figure 3.3e). In the orientation image, note that the angular domain is mapped to the linear gray scale. Consequently, both black and white correspond to near-zero angles. Figure 3.3f was generated by steering the edge images using Equation 3.7. This image is identical to the one that would be obtained by convolving the original image (Figure 3.3a) with the kernel $G_1^{\pi/4}$. ■

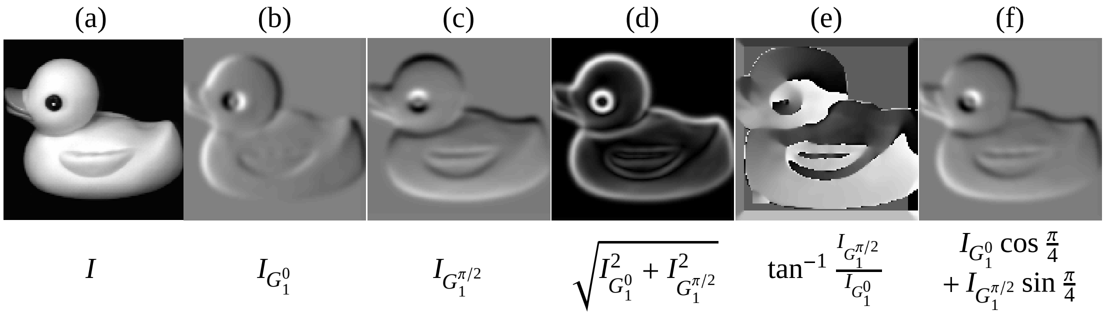

(a)
(b)
(c)
(d)
(e)
(f)
I
$I_{G_1^0}$
$I_{G_1^{\pi/2}}$
$\sqrt{I_{G_1^0}^2 + I_{G_1^{\pi/2}}^2}$
$\tan^{-1} \frac{I_{G_1^{\pi/2}}}{I_{G_1^0}}$
$I_{G_1^0} \cos \frac{\pi}{4} + I_{G_1^{\pi/2}} \sin \frac{\pi}{4}$

Figure 3.3. Example manipulations with first-derivative images. Image a shows the original image of size 128 × 128 pixels. In images b, c, and f, zero values are encoded as intermediate gray, and positive and negative values as lighter and darker shades, respectively. In image d, black represents zero. In image e, the angular range of 0–2π is mapped to the range from black to white. The derivatives were computed with $\sigma = 2$ pixels. See Example 3.1 for discussion.

3.3.3 Scale-Space Theory

A fundamental problem in computer vision is to decide on the right scale for image analysis. A typical question is: "How large are the features of interest relative to the pixel size?" In many applications, no prior scale information is available. Therefore, many computer vision systems perform their analysis over an exhaustive range of scales and integrate the results. Scale-space theory provides a sophisticated mathematical framework for multi-scale image analysis [62], and is consulted in this work to answer the above question. A (linear) scale-space representation, or simply *scale space* for short, of an image is the stack of images generated by increasingly blurring the original image. This process can be described by the diffusion equation [56]. In practice, a scale-space representation is computed by successive smoothing (Figure 3.4). The Gaussian kernel has been shown to be the unique smoothing operator for generating a linear scale space [34].

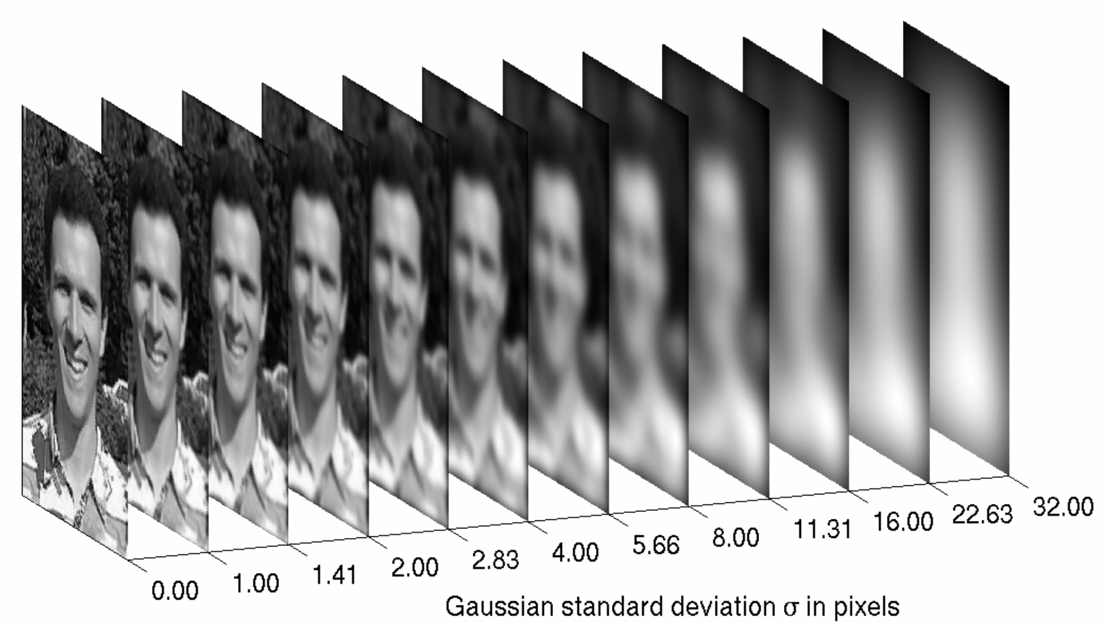


Figure 3.4. Visualization of a linear scale space generated by smoothing with Gaussian kernels of increasing standard deviation. The image size is 168 × 234 pixels.

For the purposes of this work, scale-space theory offers well-motivated ways to identify appropriate scales for image analysis at each location in an image, based on the following *scale-selection principle*:

> "In the absence of other evidence, assume that a scale level, at which some (possibly non-linear) combination of normalized derivatives assumes a local maximum over scales, can be treated as reflecting a characteristic length of a corresponding structure in the data." [64]

Lindeberg [64] provides some theoretical justification for this principle derived from the observation that if an image is rescaled by a constant factor, then the scale at which a combination of normalized derivatives assumes a maximum is multiplied by the same factor. The choice of a specific combination of normalized derivatives – in the following denoted a *scale function* $s(\sigma)$ – depends on the type of image structure of interest. Such structures and their associated scales can then be detected simultaneously in an image by finding local maxima of s in the scale space. Thus, there are typically multiple *intrinsic scales* associated with each image location. The normalization is done with respect to the operator scale σ, and ensures that the values of the scale function can be compared across scales.

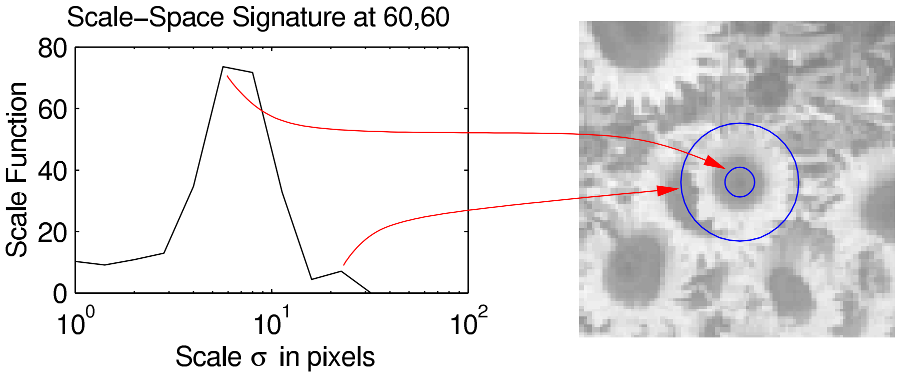


Figure 3.5. Example of a scale function $s(\sigma)$. The graph shows the scale function computed for 13 discrete values of σ at the center pixel of the image (121×121 pixels). Each local maximum defines an intrinsic scale at this point. The two intrinsic scales are illustrated by circles of corresponding radii.

Example 3.2 (Intrinsic Scale) Figure 3.5 illustrates the computation of the intrinsic scales at an image location. The graph plots the values of $s(\sigma)$ at this point, computed for discrete values of σ at half-octave spacing. The local maxima of this graph define the intrinsic scales, which are illustrated by circles of corresponding radii in the image. The stronger maximum ($s(5.7) = 73.6$) captures the size of the dark, solid region at the center of the sunflower, while the weaker maximum ($s(22.6) = 7.1$) roughly corresponds to the size of the entire flower. The maximum at the minimal value of $\sigma = 1$ is not considered a scale-space maximum because it is not enclosed between lower values of s.

To illustrate how $s(\sigma)$ can be used to detect image structures of interest at their intrinsic scale, Figure 3.6 shows the 200 strongest local maxima of the same scale function $s(\sigma)$ computed everywhere in the image. To generate this figure, the value of $s(\sigma)$ was computed at each pixel for each value of σ. The corresponding local maxima within the resulting scale space are illustrated by circles of the corresponding radius σ at the corresponding location $\mathbf{x}$. The figure clearly illustrates the correspondence of intrinsic scale with the size of local image structures, which is here dominated by sunflowers. The scale found for the large sunflower at the lower right is smaller than expected because no larger scales were considered at this point to avoid artifacts caused by the the filter kernel exceeding the image boundary. The scale function $s(\sigma)$ used for these examples is defined by Equation 3.9, discussed next. ■

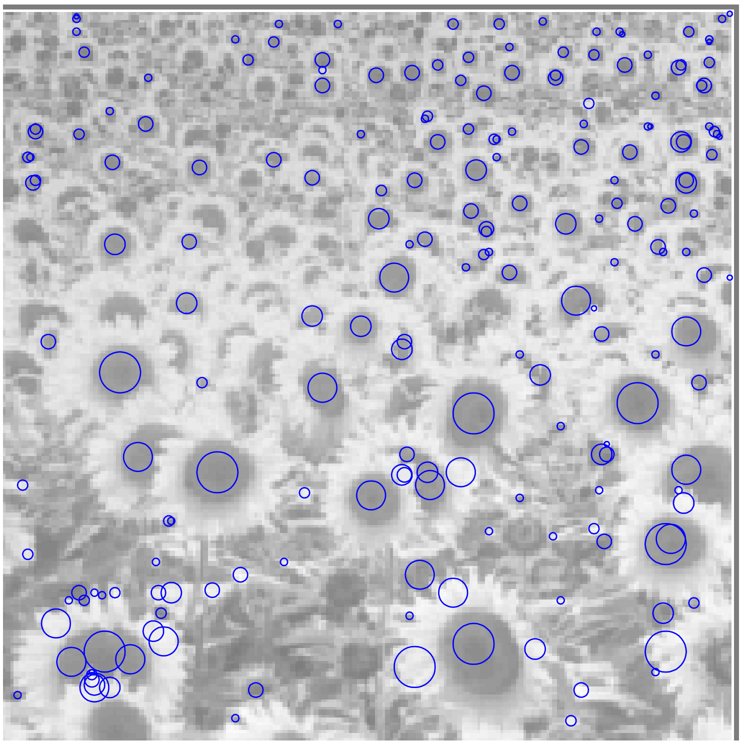


Figure 3.6. Illustration of the 200 strongest scale-space maxima of $s_{\text{blob}}(\sigma)$. The radius of each circle corresponds to the local scale σ. See Example 3.2.

The types of structures of interest in this dissertation are blobs, corners, and edges. *Blob* is a commonly used term referring to a roughly circular (or ellipsoidal) image region of roughly homogeneous intensity (or texture). A suitable isotropic scale function is defined in terms of the trace of the Hessian:

$$s_{\text{blob}}(\sigma) = \sigma^2 \left| I_{G_2^0} + I_{G_2^{\pi/2}} \right| \tag{3.9}$$

where the scale is given by σ which is the scale (standard deviation) of the filters G_2^0 and $G_2^{\pi/2}$. The normalization factor, σ^2, ensures that the scale-selection principle holds [64], and the absolute value is taken to remove sensitivity to the sign of the Hessian. This scale function is rotation invariant as are the other two mentioned below. Figure 3.6, described in Example 3.2, illustrates the image property described by a blob.

A commonly used measure for corner detection is the curvature of level curves in the intensity data combined with the gradient magnitude. Lindeberg [64] recommends the product of the level curve curvature and the gradient magnitude raised to the third power, which leads to the scale function

$$s_{\text{corner}}(\sigma) = \sigma^4 \left| I^2_{G_1^{\pi/2}} I_{G_2^0} - 2 I_{G_1^0} I_{G_1^{\pi/2}} I_{G_{x,y}} + I^2_{G_1^0} I_{G_2^{\pi/2}} \right| \tag{3.10}$$

where $G_{x,y} = \frac{\partial^2}{\partial x \partial y} G$. Again, the normalization factor σ^4 ensures that the scale-selection principle holds, and the absolute value removes sensitivity to the sign. Figure 3.7 shows how the scale-space maxima detected by s_{corner} cluster at corners, and also occur at high-contrast edges.

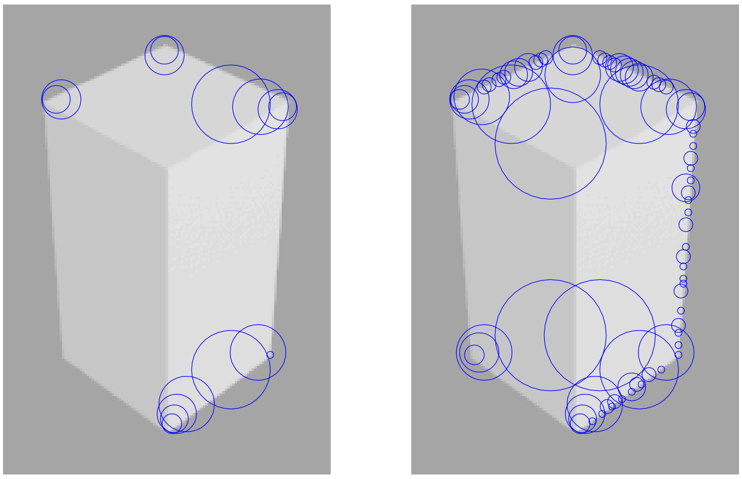


Figure 3.7. Illustration of scale-space maxima of $s_{\text{corner}}(\sigma)$. The left image shows the strongest 15 scale-space maxima, and the right image the strongest 80. The radius of each circle corresponds to the local scale σ.

For intensity edges, a useful scale function is based on the first spatial derivative [63]:

$$s_{\text{edge}}(\sigma) = \sqrt{\sigma\left(I^2_{G_1^0} + I^2_{G_1^{\pi/2}}\right)} \tag{3.11}$$

The normalization factor $\sqrt{\sigma}$ ensures the scale-selection principle as before. The normalization factors for blobs and corners as formulated above are chosen such that the recovered scales correspond to the size of the local image structure, and will thus typically be related to the size of objects depicted in the image. In contrast, the normalization factor for edges is chosen to reflect the sharpness of the edge. Sharp edges attain their scale-space maxima at smaller scales than blurry edges. For the purposes of this dissertation, the reason for this choice of scale normalization is that it ensures perfect localization of the recovered edges at the maxima in the gradient magnitude along the gradient direction.

As noted above, blob and corner features are identified as local maxima of the scale function in the scale space. To extract edges in scale space, gradient-parallel maxima are extracted from the scale space according to the notion of non-maximum suppression [17]. This results in thin and mostly contiguous edges. Since the details of scale-space edge extraction are unimportant in the context of this work, the interested reader is referred to the literature [63].

3.4 Primitive Features

Sections 3.1 and 3.2 discussed important properties and various ways to generate global and local features. Since eigen-features have a number of desirable properties, it is worth considering how their primary drawback – their global spatial extent – can be overcome. Colin de Verdière and Crowley investigated this question and developed a recognition system based on local descriptors derived by PCA [26]. Many of the most important eigen-features generated by their system resemble oriented bandpass filters. This is no accident: If natural images are decomposed into linearly independent (but not necessarily orthogonal) basis functions, using a sparseness or minimum-description-length objective that favors representations with many vanishing coefficients, then basis functions emerge that resemble oriented bandpass filters. Notably, these are strongly *localized* – even without an explicit constraint or bias toward locality [80, 91]. This result has been well established in the literature, and serves as a natural explanation for the shape of receptive fields in the mammalian early visual pathway. The receptive fields of visual neurons in the primary visual cortex have often been modeled by oriented derivatives of Gaussian functions [132, 57].

In this dissertation, these insights motivate the choice of Gaussian-derivative filters as a fundamental representation of local appearance. Since principled and biologically plausible feature extraction mechanisms tend to result in features resembling Gaussian derivatives, they are used here directly, avoiding the added effort of eigen-subspace decomposition. Moreover, Gaussians and their derivatives have a number of useful properties; those that are important for this work were introduced in Section 3.3.

The feature space considered in this work is defined recursively: A feature from this space is either a *primitive* feature, or a *compound* feature. Compound features consist of other compound features and/or primitive features, and are discussed in Section 3.5. A primitive feature is defined by a vector of responses of Gaussian-derivative filters all computed at the same image location $\mathbf{x}$. The following sections describe two types of primitive features.

3.4.1 Edgels

Edges are a fundamental concept in vision, deemed important by both machine and biological vision research. Edges describe aspects of shape in terms of intensity boundaries. Here, individual edge elements – so-called *edgels* – are considered in isolation, without attempting to join them into contours. An edgel is defined by the 2-vector

$$\mathbf{f}_{\mathrm{edgel}}(\mathbf{x}) = \begin{bmatrix} I_{G_1^0}(\mathbf{x}) \\ I_{G_1^{\pi/2}}(\mathbf{x}) \end{bmatrix}$$

where the filters G are computed at the *intrinsic* scale $\sigma_{\mathbf{x}}$ that maximizes $s_{\mathrm{edgel}}(\sigma)$ at the image location $\mathbf{x}$. An edgel encodes the orientation and magnitude of the local intensity gradient at the intrinsic scale. Note that the gradient orientation is always numerically stable at a scale-space maximum, because the numerator and denominator in Equation 3.5 cannot both be close to zero at a maximum of the scale function s_{edgel}.

3.4.2 Texels

Edgels are a weak descriptor of the local intensity surface in that they are defined by merely two parameters. It may often be desirable to have a more expressive local descriptor that uses more parameters and captures more structure than just the local intensity gradient. Intuitively, this structure corresponds to a local pattern or texture, which motivates the term *texel* (for texture element). A texel $\mathbf{f}_{\mathrm{texel}}$ is defined by a vector containing the steerable bases of the first n_d Gaussian derivatives at n_s scales. One of these scales is the intrinsic scale σ that maximizes $s_{\mathrm{blob}}(\sigma)$ or $s_{\mathrm{corner}}(\sigma)$ at an image location.

Besides the choice of the scale function, an edgel is simply a texel with $n_d = n_s = 1$. The motivation for using texels with several derivatives and scales is that the additional parameters represent more information about local image structure (or texture), increasing the specificity of the feature [90]. This increased specificity may or may not be desirable in a given task. Therefore, both edgels and texels are included in the feature space, and it is up to the feature learning algorithms (Chapters 4 and 6) to use any or both of these primitive feature types.

This work consistently uses texels composed of the first two derivatives at three scales, yielding a total of $(2 + 3) \times 3 = 15$ filter responses. The middle scale is the intrinsic scale at that image location, and the other two scales are spaced half an octave below and above. Derivatives of an order greater than about two or three rarely respond significantly in practice because of their narrow bandpass characteristics. Therefore, they contribute little to the specificity of a feature. The specificity can arbitrarily be increased by adding scales (within the limits imposed by the pixel and image sizes). However, adding larger scales reduces the locality of the feature. Also, good generalization properties are usually expected of a feature. This choice of parameters constitutes a compromise between expressive power and locality of the filter in space and scale, and has been found useful in other applications [93].

The orientation of a texel is defined in terms of the two first-derivative responses. At what scale should these responses be measured? This is a difficult question because the edge energy at the intrinsic scale as determined according to s_{blob} and s_{corner} may be very low, rendering the texel orientation unreliable. This was pointed out by Chomat et al. [19], who recommended that the orientation of blobs be defined using the scale that maximizes s_{edge} at this location. However, this scale may be very different than the texel scale, resulting in an orientation that corresponds to an image structure other than that described by the texel. In particular, this can be a problem in the presence of non-rigid objects or substantial clutter in the scene. Most of the imagery considered in this dissertation shows rigid objects and no clutter. Chomat et al.'s empirical results and my own informal experiments confirmed the clear superiority of their method, which is therefore applied throughout this work.

3.4.3 Salient Points

In principle, edgel and texel features can be computed at any location in an image. In practice, it is often useful to concentrate on *salient* image locations. For a given scale function $s(\sigma)$, any local maximum within the three-dimensional scale space defines a salient point and its intrinsic scale σ at which this maximum is attained (cf. Example 3.2 on page 17).

In order to identify scale-space maxima for edges, the scale space of s_{edge} is first projected onto the image plane by taking the maximum of s_{edge} over the scales σ at each pixel. The resulting image specifies the gradient magnitude, orientation, and intrinsic scale for each pixel. Then, gradient-parallel maxima are extracted on this 2-D plane in the conventional fashion, resulting in well-localized and mostly contiguous contour segments.

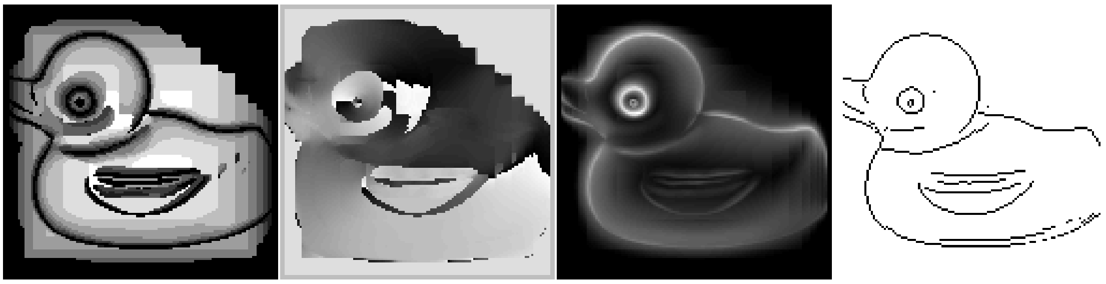

(a) intrinsic scales (b) gradient oriens. (c) grad. magnitudes (d) extracted edges

Figure 3.8. Extracting edges from the projected scale space. See Example 3.3 for explanation.

Example 3.3 (Extracting Edges from the Projected Scale Space) Figure 3.8a shows the intrinsic scale corresponding to each pixel location. At each pixel, the gray value encodes the scale at which s_{edge} is maximized at that image location. Bright shades correspond to large scales. High-contrast regions in the original image (cf. Figure 3.3a, page 16) clearly correspond to smaller scales. The blocky artifacts are boundary effects; filters are not applied near the image boundary where the filter kernel exceeds the image. Edges (Figure 3.8d) are extracted as gradient-parallel maxima from the image shown in Figure 3.8c, containing at each pixel the strongest edge energy across all scales, attained at the scale shown in Figure 3.8a, using the corresponding gradients shown in Figure 3.8b. ■

For texels, local maxima are identified in each of the two scale spaces generated by s_{blob} and s_{corner}. These local maxima are then projected onto the image plane. On those relatively rare occasions where two local maxima project to the same image location, the maximum corresponding to the larger scale is shifted slightly. The set of salient texel points is then given by the union of the projected maxima of both scale spaces. As a result, each salient image point has exactly one intrinsic scale (Figure 3.9).

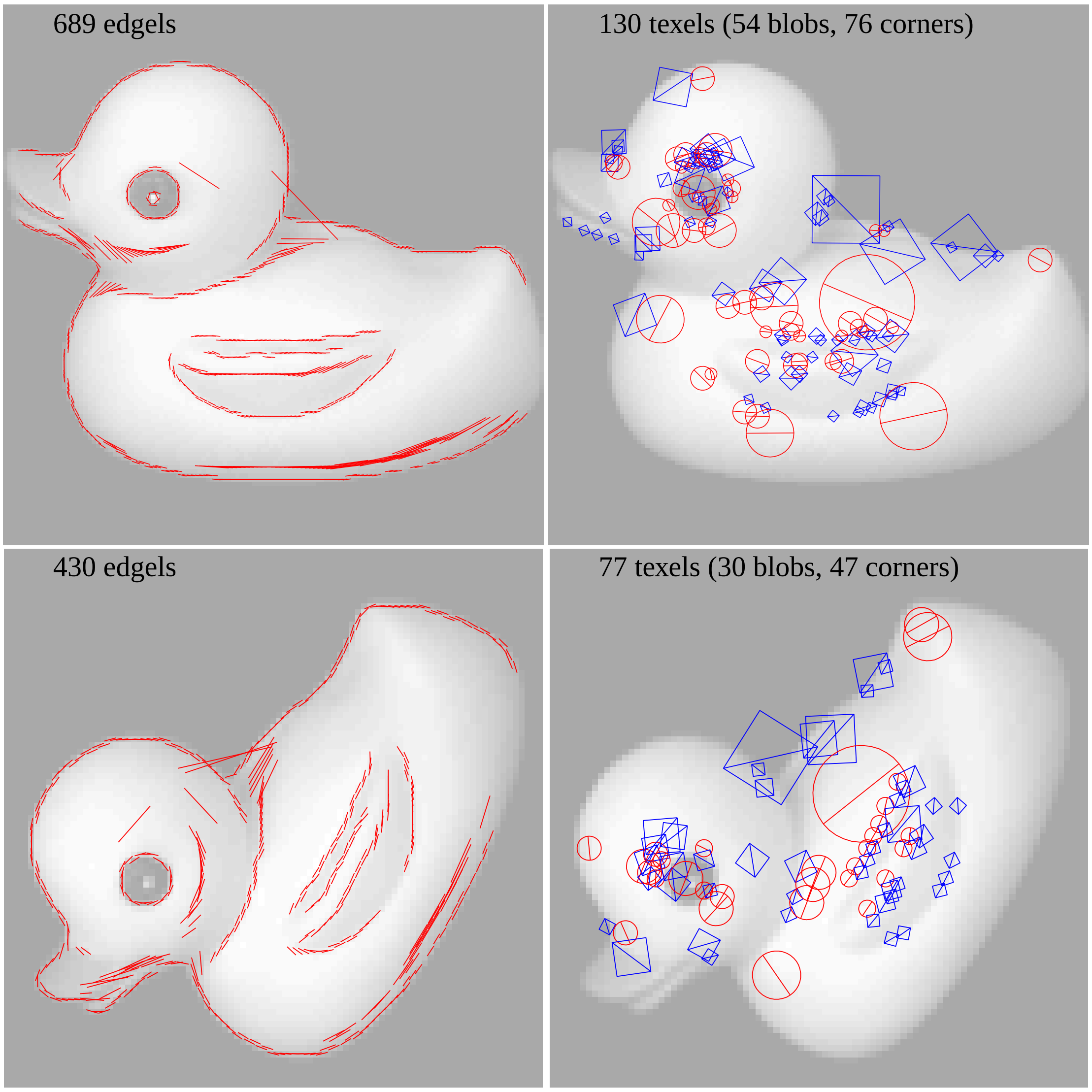


Figure 3.9. Salient edgels and texels extracted from an example image. On the left, each dark line segment corresponds to an edgel (i.e., a scale-space maximum of s_{edge}) located at the midpoint of the line, with scale σ identical to half the length of the line. On the right, square markers correspond to scale-space maxima of s_{corner}, and circular markers to scale-space maxima of s_{blob}. Collectively, these are the salient texels. The orientations of these are illustrated by the center lines of each marker. The radius of each marker is equal to the local scale σ. See Example 3.4 for discussion.

Scale spaces are computed at half-octave spacing, i.e., $\sigma_{i+1} = \sigma_i \sqrt{2}$. This choice provides sufficient resolution at moderate computational cost [93]. The minimum plausible scale is related to the pixel size, and the maximum is limited by the size of the image. This work uses scales in the range between one pixel and one quarter of the lesser image dimension.

Example 3.4 (Salient Edgels and Texels) Figure 3.9 illustrates the salient edgels and texels extracted from an example image. The radius of each edgel and texel marker shown in the figure is equal to the local scale σ. Circles and squares correspond to blobs and corners, respectively. The bottom row illustrates the degree of robustness of the salient points with respect to rotation and scale. These images were obtained by rotating the original duck image by 60 degrees and scaling it by a factor of 0.7 and subsampling, but are shown enlarged to match the original size to facilitate visual comparison. Naturally, some amount of detail is lost through the subsampling, which accounts for the reduced number of salient points.

Most edgels are located at high-contrast image locations and are extracted at a very small scale. In addition, some edgels were extracted at much larger scales, capturing large-scale image structures as opposed to pixel-level localized gradients. Likewise, texels cluster around high-contrast image regions. While edgel scales reflect the smoothness of the local edge, texel scales correspond to the size of the image structure. Note, for example, the large corner behind the neck, or the small blob at the tip of the tail. All of these are located at local maxima in the scale function; there was no threshold on the prominence of these local maxima. Many of the weaker maxima tend to occur in clusters along intensity edges, especially at smaller scales. As the figure illustrates, such texels are less robust to image transformations than texels at more pronounced locations, e.g. the blob centered at the eye, the large corner behind the neck, the small blob at the tip of the tail, or the blob at the chest underneath the chin.

Figure 3.10 illustrates the extent to which texels are stable across viewpoint changes. Again, those texels that correspond to pronounced locations tend to be more stable. Texels located at accidental maxima along intensity edges are reliably present, but their precise number and location varies across viewpoints. ■

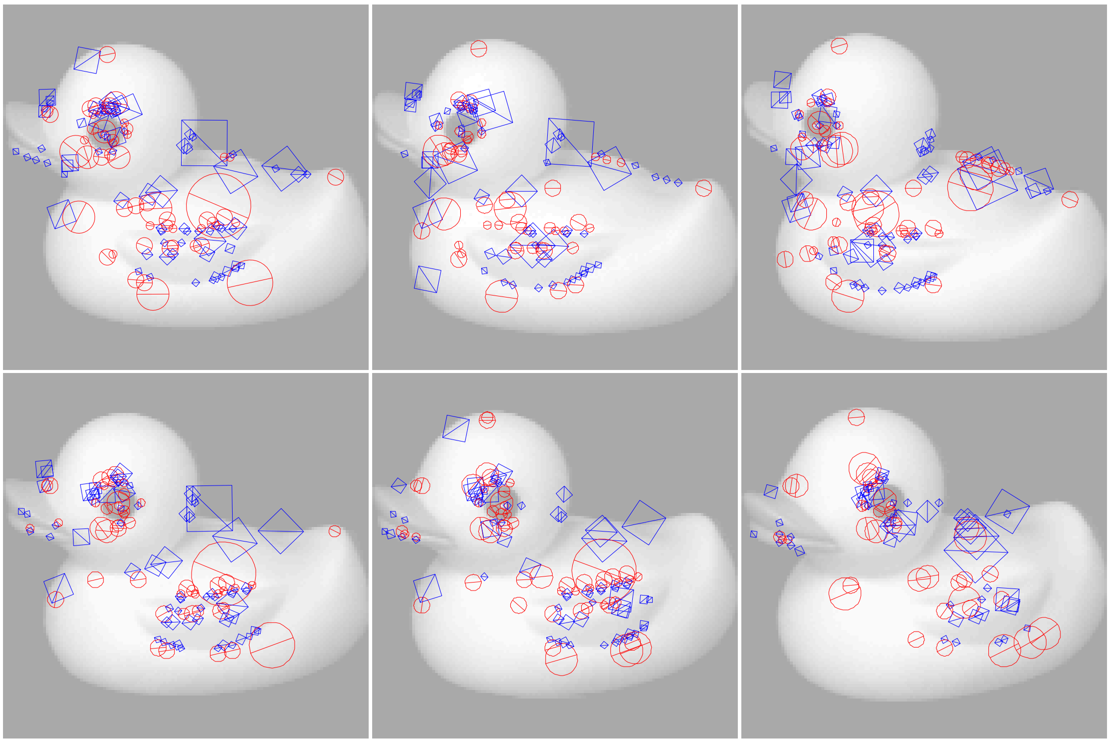


Figure 3.10. Stability of texels across viewpoints. Adjacent viewpoints differ by 10 degrees. See Example 3.4 for discussion.

3.4.4 Feature Responses and the Value of a Feature

As introduced in Sections 3.4.1 and 3.4.2, a specific instance $\mathbf{f}(\mathbf{p})$ of a primitive feature is defined by the numerical values comprising the feature response vector, typically generated by computing the applicable filter responses at a location $\mathbf{p}$ in some image. To find occurrences of $\mathbf{f}(\mathbf{p})$ in an image I, the response strength or *value* $f_{\mathbf{f}(\mathbf{p})}(\mathbf{x})$ of $\mathbf{f}(\mathbf{p})$ is measured at each point $\mathbf{x}$ in I. To measure the feature value at a location $\mathbf{x}$, the feature response vector $\mathbf{f}(\mathbf{x})$ is computed at $\mathbf{x}$, using the local intrinsic scale $\sigma_{\mathbf{x}}$. The feature value is then given by the normalized inner product of the observed response $\mathbf{f}(\mathbf{x})$ with the reference response $\mathbf{f}(\mathbf{p})$:

$$f_{\mathbf{f}(\mathbf{p})}(\mathbf{x}) = \max\left\{0, \frac{\mathbf{f}(\mathbf{p})^T\,\mathbf{f}(\mathbf{x})}{\|\mathbf{f}(\mathbf{p})\|\,\|\mathbf{f}(\mathbf{x})\|}\right\} \tag{3.12}$$

Negative values are considered as weak as orthogonal response vectors, and are therefore cut off at zero. Hence, $0 \leq f_{\mathbf{f}} \leq 1$. The normalized inner product is pleasing in that it returns the cosine of the angle between the vectors in question which, for edgels, is identical to the cosine of the angle between the edgels in the image plane: $f_{\mathbf{f}(\mathbf{p})}(\mathbf{x}) = \max\{0, \cos(\theta_{\mathbf{p}} - \theta_{\mathbf{x}})\}$. Moreover, the feature value is invariant to linear changes in image intensity.

If the feature value is to be computed regardless of the orientation of a feature in the image plane, the measured feature response vector $\mathbf{f}(\mathbf{x})$ is first steered to match the orientation of the pattern vector $\mathbf{f}(\mathbf{p})$ before Equation 3.12 is applied. This is done using a function $\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{p})$ that applies the steering equation (3.8, page 15) to all components of $\mathbf{f}(\mathbf{x})$, using $\theta = \theta_{\mathbf{p}} - \theta_{\mathbf{x}}$. Thus, Equation 3.12 becomes

$$\tilde{f}_{\mathbf{f}(\mathbf{p})}(\mathbf{x}) = \max\left\{0, \frac{\mathbf{f}(\mathbf{p})^T\,\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{p})}{\|\mathbf{f}(\mathbf{p})\|\,\left\|\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{p})\right\|}\right\} \tag{3.13}$$

for rotationally invariant feature value computation. Note that this rotationally-invariant matching implies that $f_{\mathbf{f}}$ is always equal to one if $\mathbf{f}$ is an edgel. Hence, individual edgels carry no information. Rotation-invariant edgels are only useful as part of a compound feature, as introduced in the following section.

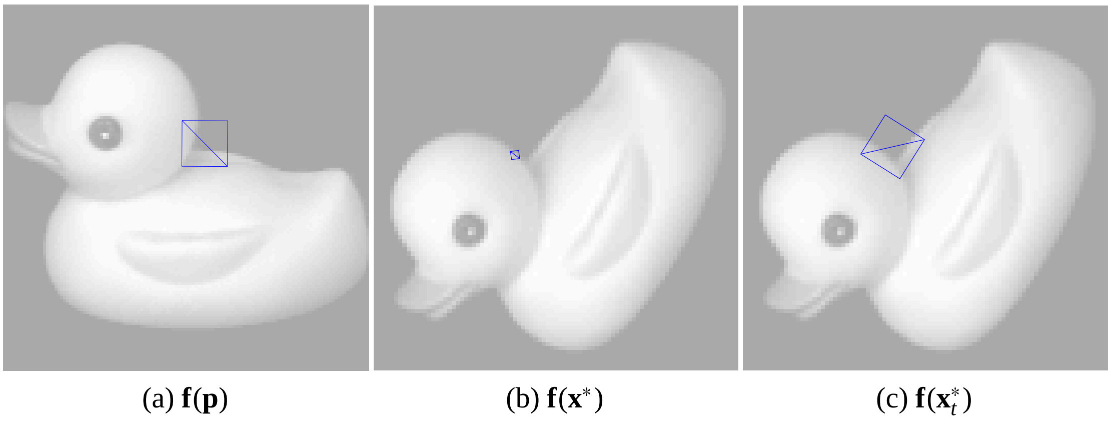

(a) $\mathbf{f}(\mathbf{p})$ (b) $\mathbf{f}(\mathbf{x}^*)$ (c) $\mathbf{f}(\mathbf{x}_t^*)$

Figure 3.11. Matching a texel (a) across rotation and scale, without (b) and with (c) rotational normalization. See Example 3.5 for explanation.

Example 3.5 (Localizing a Feature) Figure 3.11a shows one of the salient texels extracted from the familiar duck image (cf. Figure 3.9). Figures 3.11b and 3.11c show the same rotated and scaled version as used in Figure 3.9. Let $\mathbf{p} = [71, 49]^T$, which is the location of the texel shown in Figure 3.11a. Then, $\mathbf{f}(\mathbf{p})$ is the feature response vector defining this texel feature. We now want to find this feature in the rotated and scaled image. To do this without regard to feature orientation, the location $\mathbf{x}^*$ that maximizes $f_{\mathbf{f}(\mathbf{p})}(\mathbf{x})$ is identified using Equation 3.12. The salient point $\mathbf{x}^*$ that maximizes Equation 3.12 is that shown in Figure 3.11b. Here, $f_{\mathbf{f}(\mathbf{p})}(\mathbf{x}^*) = 0.956$. It is no accident that this texel has the same orientation as the pattern feature $\mathbf{f}(\mathbf{p})$ shown in Figure 3.11a, but the two locations do not correspond (even though they are spatially close to one another in this example). The true corresponding feature is shown in Figure 3.11c; let us call this location $\mathbf{x}_\mathrm{t}^*$. Disregarding the difference in orientation, $f_{\mathbf{f}(\mathbf{p})}(\mathbf{x}_\mathrm{t}^*) = 0.578 < f_{\mathbf{f}(\mathbf{p})}(\mathbf{x}^*)$. If at each location $\mathbf{x}$ the feature response vector $\mathbf{f}(\mathbf{x})$ is first steered to match the orientation of $\mathbf{f}(\mathbf{p})$ using Equation 3.13, then the correct location is found with $\tilde{f}_{\mathbf{f}(\mathbf{p})}(\mathbf{x}_\mathrm{t}^*) = 0.996$ (Figure 3.11c). In contrast, the value at $\mathbf{x}^*$ increases only marginally from $f_{\mathbf{f}(\mathbf{p})}(\mathbf{x}^*) = 0.956$ to $\tilde{f}_{\mathbf{f}(\mathbf{p})}(\mathbf{x}^*) = 0.960$, remaining below the value at the correct location $\mathbf{x}_\mathrm{t}^*$. ■

In all parts of this work, feature values are computed with rotational normalization using Equation 3.13. From now on, the tilde will be omitted for simplicity, and the notation $f_\mathbf{f}$ will be used with the understanding that feature response vectors are always steered to achieve rotational invariance.

A feature $\mathbf{f}(\mathbf{p})$ is present in an image I to the degree

$$f_{\mathbf{f}(\mathbf{p})}(I) = \max_{\mathbf{x} \in I} f_{\mathbf{f}(\mathbf{p})}(\mathbf{x}). \tag{3.14}$$

This computation is quite expensive, as it involves the measurement of the intrinsic scale and the computation of $\mathbf{f}(\mathbf{x})$ and $f_{\mathbf{f}(\mathbf{p})}(\mathbf{x})$ at each location in the image. A large proportion of this computation is going to be wasted because most image regions will not contain features of practical relevance. Therefore, the assumption is made that all features of practical relevance are located at salient points. To identify the salient points it is still necessary to evaluate $s(\sigma)$ over the entire range of scales σ everywhere in the image, which constitutes a highly parallelizable operation. Moreover, if the application provides a fixed set of training images, this computation can be performed off-line. In practice, this leads to a great reduction in computation time and space, as the maximum in Equation 3.14 is only taken over the salient points rather than the entire image.

3.5 Compound Features

A core objective of this work is to demonstrate that features can be learned that are highly meaningful individually. The limited descriptive power of individual primitive features is overcome by composing two or more primitive features into compound features. Since compound features contain more parameters and cover a larger image area than do primitive features, they provide a potentially more specific and robust description of relevant aspects of appearance. What it means for a feature to be relevant, and how relevant features are sought, depends on the given task. Chapters 4 and 6 present two illustrative examples. This section describes three complementary ways in which primitive features can be composed into compound features.

There is evidence that the mammalian visual cortex also composes simple and generic features hierarchically into more complex and specific features. The primary visual cortex contains retinotopic arrays of local and oriented receptive fields [45] which have been modeled as Gaussian-derivative functions [132, 57]. Many neurons found in higher-order visual cortices are tuned to more complex stimuli. Riesenhuber and Poggio [96] proposed a hierarchical model of recognition in the visual cortex, using spatial combinations of view-tuned units not unlike the compound features introduced below.

3.5.1 Geometric Composition

The structure of a geometric compound of $n_\mathbf{f} \geq 2$ primitive features $\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{n_\mathbf{f}-1}$ (Figure 3.12) is defined by the attitudes ϕ_i of each subfeature, and the angles $\Delta\theta_i$, distances d_i, and scale ratios $\Delta\sigma_i$ between the constituent subfeatures $\mathbf{f}_i$. The distances d_i are given relative to the intrinsic scale σ_{i-1} at point $i - 1$. Each primitive feature is either an edgel or a texel. A specific instance $\mathbf{f}_{\text{geom}}$ of a geometric feature with a given structure is defined by the concatenation of the vectors comprising the primitive subfeatures. Its orientation is denoted by θ, which is the orientation of the reference point.

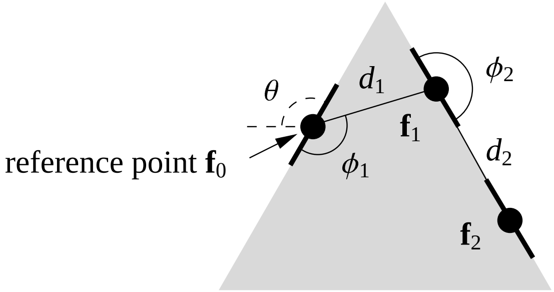


Figure 3.12. A geometric compound feature consisting of three primitive features.

The location of a feature is defined by the location of the reference point. To extract the filter response vector defining a geometric feature $\mathbf{f}(\mathbf{x})$ at location $\mathbf{x}$, the following procedure is used:

Algorithm 3.6 (Extraction of a Geometric-Compound Feature Response Vector)

1. The response vector $\mathbf{f}_0(\mathbf{x})$ is computed at $\mathbf{x}$. This defines the orientation $\theta_{\mathbf{x}}$ and intrinsic scale $\sigma_{\mathbf{x}}$ of $\mathbf{f}(\mathbf{x})$. Initially, let $i = 1$:
2. If $i = n_{\mathbf{f}}$, then stop.
3. The location $\mathbf{x}_i$, orientation θ_i, and scale s_i of the next constituent subfeature $\mathbf{f}_i$ are computed:

$$\begin{aligned}
\mathbf{x}_i &= \mathbf{x}_{i-1} + \sigma_{i-1} d_i \begin{bmatrix} \cos(\theta_{i-1} + \phi_i) \\ \sin(\theta_{i-1} + \phi_i) \end{bmatrix} \\
\theta_i &= \theta_{i-1} + \Delta\theta_i \\
\sigma_i &= \sigma_{i-1} \Delta\sigma_i
\end{aligned}$$

4. The feature response vector $\mathbf{f}_i(\mathbf{x}_i)$ is computed at scale σ_i, and is rotated to orientation θ_i using the steering equation 3.8 (page 15) with angular argument $\theta = \Delta\theta_i$. Increment i, and continue at step 2.

The filter response vector is given by the concatenation of the individual response vectors of the constituent primitive features $\mathbf{f}(\mathbf{x}_i)$, $i = 0, 1, \ldots, n_{\mathbf{f}} - 1$. ■

The value $f_{\mathbf{f}_{\text{geom}}}(I)$ of a geometric compound feature $\mathbf{f}_{\text{geom}}$ in an image I is computed in the same way as for a primitive feature (Equations 3.13 and 3.14), using the concatenated response vectors. This involves running Algorithm 3.6 at each of the n_{sal} applicable salient image locations. These are either all salient edgels or all salient texels, depending on the type of the first subfeature $\mathbf{f}_0$ of $\mathbf{f}_{\text{geom}}$. It is easily seen that the time required to compute $f_{\mathbf{f}_{\text{geom}}}$ is proportional to $n_{\text{sal}} n_{\mathbf{f}}$.

3.5.2 Boolean Composition

A Boolean compound feature consists of exactly two subfeatures, each of which can be a primitive or a compound feature. The relative locations of the subfeatures are unimportant. Two types of Boolean features have been defined:

- The value of a *conjunctive* feature is given by the product of the values of the two subfeatures.
- The value of a *disjunctive* feature is given by the maximum of the values of the two subfeatures.

For conjunctive features, the product is used instead of the minimum because it is natural to assign a higher value to a feature if the larger of the two component values is increased even further. For example, the value $f_{\mathbf{f}}$ of a conjunctive feature with subfeature values $f_{\mathbf{f}_1} = 0.6$ and $f_{\mathbf{f}_2} = 0.9$ should be larger than for a feature with subfeature values $f_{\mathbf{f}_1} = 0.6$ and $f_{\mathbf{f}_2} = 0.7$.

Feature values of Boolean compound features are measured recursively by first measuring the values of each of the two subfeatures individually, and then combining their values using the product or the maximum, respectively. Hence, the time required to compute the value $f_{\mathbf{f}_{\text{bool}}}(I)$ of a Boolean compound feature $\mathbf{f}_{\text{bool}}$ in an image I is proportional to $n_{\text{sal},0}t_0(I) + n_{\text{sal},1}t_1(I)$, where $n_{\text{sal,i}}$ denotes the number of salient image points applicable to subfeature $\mathbf{f}_i$, and $t_i(I)$ denotes the time required to compute the value $f_{\mathbf{f}_i}(I)$ of subfeature $\mathbf{f}_i$ in the image. Note that each subfeature of a Boolean compound feature can be any type of feature, including primitives, or geometric, conjunctive or disjunctive compounds.

3.6 Discussion

Section 3.2 formulated a list of objectives (page 12) to be met by the feature design. All of these are addressed by the compositional feature space:

1. Features are computed on intensity images.
2. The feature space was designed without a specific task in mind. Features can be composed to arbitrary structural complexity to increase their specificity.
3. Therefore, it seems justified to hope that powerful individual features are contained in this feature space. The extent to which this goal was achieved will be discussed in Chapters 4–6.
4. Edgels and texels are complementary and capture important aspects of image structure. Geometric and Boolean compositions are also complementary and have intuitive semantics. Subsequent chapters will illustrate possibilities and limitations of this feature space for practical tasks.
5. Feature orientations are easily computed and compensated for using the steerability property of the filters employed.
6. The features are computed at the intrinsic scale observed in the images. Therefore, they are useful for scale-invariant image analysis.
7. The features are local, since the filters employed have local support. The locality of geometric features can be controlled by placing constraints on the distances between primitives and the number of primitives allowed.
8. Features return a scalar value, which makes them applicable in general classification and regression frameworks. The orientation attribute of a feature can also be used for regression, as will be seen in Chapter 6.
9. The biological relevance of this feature space was briefly discussed in the introductions to Sections 3.4 and 3.5.

A critical property of this compositional feature space is that it is partially *ordered* by the structural *complexity* of a feature, according to the number of primitives contained in a compound. Also, the three compositional rules are ordered by their expressive power. A disjunctive compound can be regarded as more complex than a conjunctive compound in that it can express a strictly larger class of concepts. For example, a hierarchy of disjunctions of sufficiently complex geometric features can memorize almost any arbitrary set of images. Conjunctive features are more expressive than geometric features because the former can be composed out of the latter, but not vice versa.

This orderly structure will play an important role in the following chapters, where algorithms are described for learning features from this space. These algorithms create features in a simple-to-complex manner, in the sense that features containing few primitives are preferred over features with many primitives, and geometric composition is preferred over conjunctive composition, which in turn is preferred over disjunctive composition. The underlying assumption is that simpler features generally provide for better generalization, or in other words, that the "real" structure underlying a task tends to be expressible by simple features.

In summary, all design goals laid out in Section 3.2 are addressed by the feature space defined in this chapter. It is unique in its hierarchical and parametric organization that facilitates simple-to-complex search for highly distinctive, local features in an infinite space. Because of these key properties it is well suited for open tasks, as argued in Chapters 1 and 2. The following chapters will demonstrate how important visual skills can be learned on the basis of this feature space.

CHAPTER 4

LEARNING DISTINCTIVE FEATURES

This chapter introduces the first of two feature learning applications that build on the feature space introduced in Chapter 3. The task is to learn to distinguish visual contexts by discovering few but highly distinctive features. The scenario addressed in this chapter is substantially less constrained than in most conventional approaches to visual recognition. The learning system does not assume a closed world. Learning is incremental, and new target object classes can be added at any time. An image presented to the system may show any number of target objects, including none. No explicit assumptions are made about the presence or absence of occlusions or clutter, and no object segmentation is performed. Consequently, this chapter does not set out to improve on the recognition rates achieved by today's more specialized recognition systems. Rather, the goal is to demonstrate that a highly uncommitted system, starting from a nearly blank slate, can learn useful recognition skills.

4.1 Related Work

The general problem of visual recognition is too broad and underconstrained to be addressed comprehensively by today's artificial systems. Therefore, the following three subproblems have been identified in the literature that are more easily addressed separately by specialized solutions:

Specific Object Recognition. In a typical scenario, a test image depicts exactly one target object (with or without occlusion and/or clutter). The task is to recognize that object as one of several specific objects [46, 73, 104, 71], e.g. a particular toy or a particular landmark. The appearance or shape of the objects in the database are precisely known. Naturally, most work concentrates on rigid objects, with the notable exception of face recognition.

Generic Object Classification. The scenario is the same as in Specific Object Recognition, but here the task is to label the target object as belonging to one of several known object categories or classes [133, 111]. The individual objects within a class may vary in appearance. A class ("car", "chair") can be defined in various ways, e.g. by object prototypes, by abstract descriptions, or by models containing enough detail to discriminate between classes, but not enough to distinguish between objects within a class.

Object Detection. Test images contain various items and typically show natural (indoor or outdoor) scenes. The task is to localize regions of the image that depict instances of a target object or class of objects, e.g. faces or people [81, 7, 6]. Object detection is closely related to fundamental problems in image retrieval [92].

The requirements and difficulties differ between these subproblems. Therefore, many existing recognition systems are designed to solve only one of them. Moreover, some practical recognition problems further specialize these categories. For example, face recognition [125] is a distinct subcategory of specific object recognition that must cope with gross non-rigid transformations that are hard to model mathematically. Optical character recognition is a distinct subcategory of generic object recognition [8], etc. Other practical recognition problems are often addressed in related ways. For example, Weng et al. [131] treat indoor landmark recognition as a simplified object recognition problem without generalization across size or pose. Riseman et al. [100] represent outdoor landmarks by line models and recognize them by many-to-many model matching in two and three dimensions. Very few promising attempts have been made to develop systems that are applicable to more than one of these subproblems. Among these, most systems employ features that explicitly allow certain image variations [128, 5, 77, 78].

For visual recognition, two major classes of techniques can be identified:

Model-based methods employ geometric models of the target objects for feature extraction or alignment [46, 100].

Appearance-based or model-free methods extract features directly from example images without explicitly modeling shape properties of the target objects. Various types of features have been employed, e.g. principal components computed from the image space [73, 125], local image statistics [116, 104, 71], locally computed templates [128], or primitive local shape descriptors such as oriented edges or corners [104, 71].

Hybrid techniques have also been proposed in which appearance-based methods served as indexing mechanisms to reduce the number of candidate object models [55]. In fact, it appears that this was historically the primary motivation for appearance-based methods. Only later was their full potential discovered, and systems began to emerge that omitted the model-matching phase and relied on appearance-based methods alone.

Most appearance-based visual recognition systems express recognition as a pattern classification problem, and many employ standard pattern recognition algorithms [128, 71, 8, 81]. Also very common in computer vision, custom classifiers are designed for specific recognition paradigms. For example, Swain and Ballard [116] and Schiele and Crowley [104] proposed methods for comparing histograms. Turk and Pentland [125] and Murase and Nayar [73] find nearest neighbors in eigen-subspaces. Fisher and MacKirdy [32] use a perceptron to learn the weights for a weighted multivariate cross-correlation procedure.

A large variety of features have been proposed (see also Section 3.1). Those that relate to this work fall into two broad categories: eigen-subspace methods that operate in image space [125, 48, 73, 131, 119], and locally computed image properties [128, 116, 104, 71, 5, 32, 66]. Eigen-subspace methods are well suited for learning highly descriptive [125, 73, 119] or discriminative features [131, 119]. Locally computed features are typically defined a priori [116, 104, 71], or they are derived to be descriptive of specific object classes [128, 32, 6, 33, 66]. Few approaches exist that attempt to derive discriminative local features [81, 8, 5].

4.2 Objective

Object recognition has reached a remarkable maturity in recent years. The problem of finding and recognizing specific objects under controlled conditions can almost be considered solved, even in the presence of significant clutter and occlusion. Generic object recognition is considerably harder because the within-class variability in object appearance can be arbitrarily large. If the within-class variability is large in comparison to the between-class variability, it can be very difficult to find feature sets that cleanly separate the object classes in feature space. This problem has not yet received much systematic attention. Essentially all current object recognition and detection systems operate on the assumption that the within-class appearance variability is small compared to the between-class variability, i.e., objects within a class appear more similar (to a human observer) than objects from distinct classes. Consequently, researchers choose feature sets that reflect our intuitive notion of visual similarity.

In practical tasks, this assumption does not always hold. I believe that this constitutes one of the key difficulties in generic object recognition: In practice, meaningful object classes are often defined not by their appearance, but by some other similarity metric, e.g. functional properties. A classical example is that of chairs: A chair is something that affords sitting. In a given context, anything that has this functional property is a chair. However, chairs can look quite dissimilar. It is very hard to specify a generic appearance model of a chair for object recognition purposes.

In contrast, I argue that humans recognize chairs on the basis of their experience that objects that afford sitting tend to exhibit certain visual properties, e.g. a roughly horizontal surface of appropriate size and height. It is our *experience* that created the association between function and specific aspects of appearance. None of the work cited above – arguably with the exception of Papageorgiou and Poggio's wavelet-based trainable object detection system [81] – considers object classes that arise from an application context other than subjective similarity of appearance.

A corollary of the necessity of learning object models from experience is that learning must be incremental. Some of the recognition algorithms mentioned above can be altered to learn incrementally [130, 81], but none of these authors noted incremental learning as a desirable property of machine vision systems.

I perceive the distinction between specific and generic object recognition and detection as rather artificial. It is the necessity to define problems clearly and to design testable and comparable algorithms that has produced this taxonomy of approaches, rather than well-established principles of vision. The same basic mechanism should produce and refine features at various levels of distinction, as demanded by a task. Features should express just the level of specificity or generality required along a continuum between specific and generic recognition [122, 121]. Likewise, it is hard to think of visual search or object detection without doing some recognition of the detected objects at the same time. Ultimately, we need a uniform framework for visual learning that addresses these subtasks in a coherent fashion.

Building on the feature space introduced in Chapter 3, this chapter presents a framework for incremental learning of features for visual discrimination. The task scenario is not restricted to specific objects, landmarks, etc. A design goal is to be able to learn a wide variety of perceivable distinctions between visual contexts that may be important in a given task. A visual context may correspond to an individual object, a category of objects, or any other visually distinguishable entity or category. In agreement with pattern recognition terminology, these distinct entities or categories will be referred to as *classes.* The following properties distinguish this framework from most conventional recognition schemes (see also Table 1.1 on page 2):

- At the outset, the number and nature of the classes to be distinguished are unknown to the learner.
- Learning is performed incrementally on a stream of training images. No fixed set of training data is assumed. Only a small number of training images need to be accessed at any one time.
- There is no distinction between training and test phases. Learning takes place on-line, during execution of an interactive task.
- There can be any number of target objects in a scene, not just one. Prior to recognition, this number is unknown to the system.
- In principle, there is no distinction between specific recognition, generic recognition, and detection. The unified framework encompasses each of these types of problems.
- Prior to learning, the algorithms and representations are largely uncommitted to any particular task. The objective is to learn *distinctive* local features.

Definition 4.1 (Distinctive Feature) A feature is *distinctive* if it has both of the following properties:

- It is *characteristic* of a particular class or collection of classes. These classes are called the *characterized* classes. Objects of a characterized class are likely to exhibit this feature.
- It is *discriminative* with respect to another class or disjoint collection of classes. Objects of such a *discriminated* class are unlikely to exhibit this feature.

■

Taken together, these properties allow one to argue that if a given distinctive feature is found in a scene, an instance of the characterized class(es) is likely to be present in the scene. This reasoning forms the basis for the feature learning algorithm presented in Section 4.4. The goal is to show that useful recognition performance can be achieved by an uncommitted system that learns distinctive features. To reinforce this point, the system starts out with a blank slate, without any task-specific tuning, and learns a *small* number of features to solve a task.

It is worth pointing out that, strictly speaking, neither of these two characteristics is biologically correct: Firstly, the human visual system indeed shows very little hard-wired specialization and draws much of its power from its versatility and adaptability. However, it is also true that infant visual learning is guided by highly structured developmental processes. Secondly, we are capable of learning individual, distinctive high-level features, e.g. corresponding to object parts. On the other hand, the robustness of our visual capabilities is largely due to the redundancy of a large number of low-level features such as line segments, corners and other descriptors of local image structure represented by neurons along the visual pathway between the primary visual cortex and the inferotemporal cortex.

4.3 Background

The framework for incremental learning of discriminative features, introduced in the following section, builds on two important existing concepts – Bayesian networks and the Kolmogorov-Smirnoff distance – that are be briefly described here.

4.3.1 Classification Using Bayesian Networks

In the context of this work, *classification* refers to the process of assigning a class label to an instance described by a *feature vector*. The elements of a feature vector may be scalar or categorical values, whereas a class label is always categorical. An abstract device that performs classification is called a *classifier*. Here, a classifier is trained inductively, i.e., by generalization from training examples. There are many well-established frameworks for classification, e.g. decision trees, neural networks, nearest-neighbor methods, and support vector machines.

Bayesian networks constitute a rather general graphical framework for representing and reasoning under uncertainty. A Bayesian network encodes a joint probability distribution. Each node represents a random variable, here assumed to be discrete. The network structure specifies a set of conditional independence statements: The variable represented by a node is conditionally independent of its non-descendants in the graph, given the values of the variables represented by its parent nodes. In more intuitive terms, an arc indicates direct probabilistic influence of one variable on another. Each node X has an associated table specifying the conditional probabilities of the value of X given the values of its parents. In other words, the conditional probability table of X contains the values of $P(X = x \mid A = a, B = b, \ldots)$, where $A, B, \ldots$ are the parents of X in the graph, and $x, a, b, \ldots$ each denote a particular value – commonly called a *state* – of the corresponding random variable.

Bayesian networks permit the computation of many interesting pieces of information. Most importantly, one can infer the probabilistic value of some variables, given the values of some other variables. This involves the following two essential steps:

Instantiation of evidence: The values of some variables are set to their observed values. These values may themselves be (unconditional) probability distributions over the possible values of these variables.

Propagation of evidence: Using the conditional probability tables, the posterior probabilities of neighboring nodes are iteratively computed, generally until some nodes of interest – *target* variables – are reached. These nodes now contain the posterior probabilities of their values given all evidence entered into the net, also called *beliefs*: $BEL(X = x) = P(X = x \mid A = a, B = b, \ldots)$ where $A, B, \ldots$ include all nodes in the net that contain evidence [83]. If evidence is propagated exhaustively throughout the entire network, the net is said to be in *equilibrium.*

Mathematical methods exist that allow these two steps to be freely intermixed. In general, exact inference is NP-hard. In practice however, due to the sparseness of most practical networks, inference is performed very quickly for up to a few tens or hundreds of nodes.

Often it is useful to know the influence of one variable on another. The influence of variable X on variable Y is conveniently defined by the mutual information [112] between X and Y, that is, the potential of knowing X to reduce uncertainty regarding the value of Y:

$$I(Y, X) = H(Y) - H(Y \mid X) \tag{4.1}$$

where[1] the uncertainty in Y is given by the entropy function

$$H(Y) = -\sum_y P(y) \log P(y)$$

and

$$\begin{aligned} H(Y \mid X) &= \sum_x H(Y \mid x)\, P(x) \\ &= -\sum_x \sum_y P(y \mid x)\, P(x) \log P(y \mid x) \\ &= -\sum_x \sum_y P(y, x) \log P(y \mid x) \end{aligned}$$

is the expected residual uncertainty given that the value of X is known. Expanding the conditional probabilities and using belief notation, Equation 4.1 becomes

$$I(Y, X) = -\sum_x \sum_y BEL(y, x) \log \frac{BEL(y, x)}{BEL(y)\, BEL(x)}. \tag{4.2}$$

To compute $I(Y, X)$, note that $BEL(y, x) = BEL(x \mid y)\, BEL(x)$. One can compute $BEL(x \mid y)$ by temporarily setting Y to value y, propagating beliefs through the network to establish equilibrium, and using the resulting posterior probabilities at node X as the $BEL(x \mid y)$ [83, 98]. The unconditional beliefs $BEL(y)$ and $BEL(x)$ are available at the respective nodes prior to application of this procedure.

The theory of Bayesian networks has been best developed for discrete random variables. With some restrictions, inference is also possible with so-called conditional Gaussian distributions [60], and the theory is constantly being advanced. For more information on Bayesian networks, excellent texts are available [83, 51].

A *naive Bayesian classifier* is easily represented using a Bayesian network. It has the topology of a star, with a node representing the class random variable in the center, and arcs going out to each feature variable. Intuitively, each arc specifies that the presence of a class gives rise to the feature pointed to by the arc. A general Bayesian classifier models dependencies between features by inserting arcs between them, where applicable.

[1] The notation I, conventionally used to denote mutual information, is not to be confused with the use of I for an image intensity function introduced in Section 3.3.1. In the following, the context will always indicate the referent of I.

4.3.2 Kolmogorov-Smirnoff Distance

Let X be a continuous random variable, and let $p(x \mid a)$ and $p(x \mid b)$ describe the conditional probability densities of X under two conditions a and b. Suppose one is asked to partition the domain of X into two bins using a single cutpoint α such that most values of X that fall into one bin are generated under condition a, and most values in the other bin under condition b. A Bayes-optimal value of α is one that minimizes the probability of any given instance x ending up in the wrong bin.

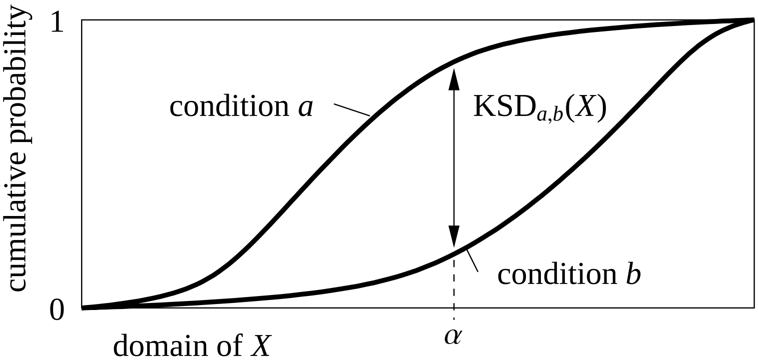


Figure 4.1. Definition of the Kolmogorov-Smirnoff distance.

Let $\mathrm{cdf}(x \mid a)$ and $\mathrm{cdf}(x \mid b)$ denote the cumulative conditional densities corresponding to $p(x \mid a)$ and $p(x \mid b)$, respectively. The *Kolmogorov-Smirnoff distance* or *KSD* for variable X under conditions a and b is defined as

$$\mathrm{KSD}_{a,b}(X) = \max_{\alpha} \left| \mathrm{cdf}(\alpha \mid a) - \mathrm{cdf}(\alpha \mid b) \right| \tag{4.3}$$

(see Figure 4.1). A value of α maximizing this expression constitutes a Bayes-optimal cutpoint in the above sense, if the prior probabilities of a given instance x having been generated under conditions a and b are equal.

In practice, the cumulative distributions are often not available in analytic form. Fortunately, it is easy to estimate these from sampled data. Given a particular cutpoint α, simply count the data samples that fall into each bin, and normalize the counts to convert them into cumulative probabilities [126]. To find an optimal α, this procedure is performed for all candidate cutpoints that include all midpoints between adjacent values in the sorted sequence of the available data.

4.4 Feature Learning

Figure 4.2 is intended to serve as a road map to the algorithms defined in the present chapter. It shows an instantiation of the generic model of incremental learning (Figure 2.1 on page 8) applied to learning distinctive features. This is the task scenario underlying this chapter: At the outset, the agent does not know anything about the types of distinctions and classes to be learned, and it has no features available. As it performs its task, it acquires one image at a time, and attempts to recognize its contents as far as they are relevant to the task. If the recognition succeeds, the agent simply continues to perform its task. If not, it attempts to learn a new feature that solves the recognition problem.

In the basic form introduced below, the feature learning algorithm requires two key properties of the task environment:

1. The agent must be able to compare its recognition result with the correct answer (i.e., the learning algorithm is *supervised*).

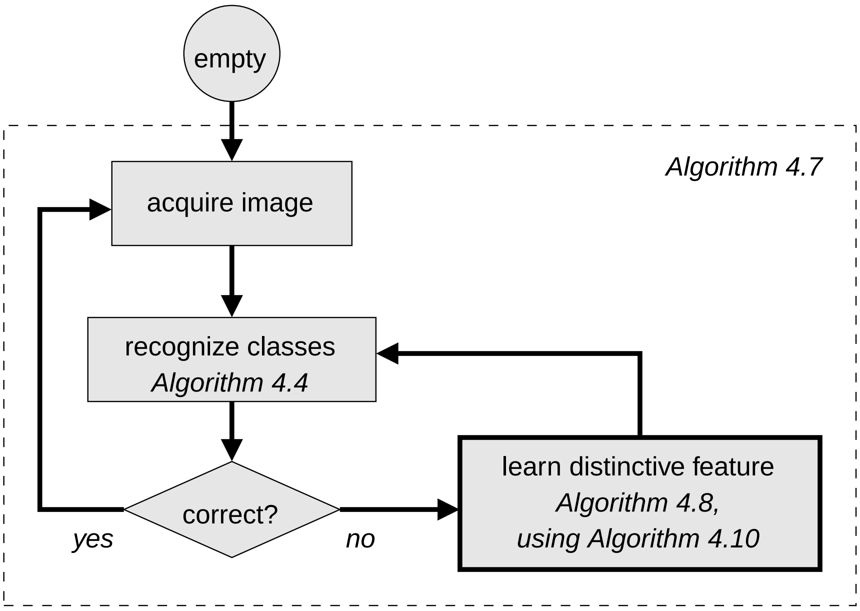


Figure 4.2. A model of incremental discrimination learning (cf. Figure 2.1, page 8).

2. The agent must be able to acquire random *example images* of scenes containing instances of specific classes. This permits statistical evaluation of candidate features.

The second requirement is not a critical limitation in many realistic interactive scenarios. For example, an infant can pick up a known object and view it from various viewpoints; or a child receives various examples of letters of the alphabet from a teacher. A robot may aim its camera at known locations, or may pick up a known object from a known location.

The subsequent sections introduce the key components of the visual learning algorithm in turn; namely, the classifier subsystem, the recognition algorithm, and the feature learning procedure.

4.4.1 Classifier

Conventionally, a Bayesian network classifier models the classes using a single discrete random variable that has one state for each class. This representation assumes that there is exactly one correct class label associated with each instance. Also, each feature – if instantiated – always contributes to the classification process, no matter what the current class is, because each feature node contains a full conditional probability table that *for each class* represents the probability of this feature being observed. This is contrary to the concept of distinctive features that specialize in specific distinctions between a characteristic class and a small subset of the other classes. In a naive Bayesian classifier, the number of entries contained in the probability table at each feature node is equal to the product of the number of classes times the number of states of the feature variable. If there are many classes, these tables can become quite large. All their conditional probability entries must be estimated. Even worse, all of them must be re-estimated whenever a new class is added to the system.

A natural way to avoid these problems is to represent each class by an individual Bayesian network classifier. An example is shown in Figure 4.3. The class node of each network has two states, representing presence or absence of an instance of this class. Here, each feature is characteristic (in the sense of Definition 4.1, page 31) of the class represented by its Bayesian network, and is discriminative with respect to one or more specific other classes. In Figure 4.3, all features are characteristic of class C, and the discriminated classes are shown inside the feature nodes.

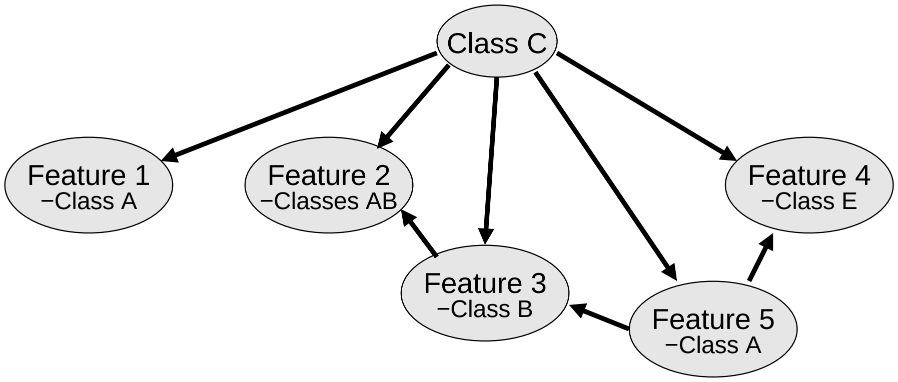


Figure 4.3. A hypothetical example Bayesian network.

This representation keeps the classes and their characteristic features separate. When a new class is learned, a corresponding network is instantiated, but the existing networks and their conditional probability tables remain unaffected. Moreover, this representation builds in the assumption that the probability of the presence of each class is independent from all other classes. Since each network separately decides whether an object of its class is present, multiple target classes within a scene are easily handled without interference. Each network effectively acts as an object *detection* system. In contrast, most current recognition systems act as *classification* systems and assign exactly one class label to each image.

Together with the concept of distinctive features, an important implication of this representation is that the absence of a feature in an image never increases the belief in the presence of a class. Inference in favor of a class is only drawn as a result of detected features. This behavior is a design choice reflecting the openness of the uncommitted learning system. It is noteworthy that this design is in contrast to conventional recognition systems that always assign exactly one of a given set of class labels to an image. Under this paradigm, called *forced-choice* recognition by psychologists, not detecting a characteristic feature will inevitably increase the belief in the presence of other classes. This effect may be desirable in a *closed world*, but can be detrimental in realistic open scenarios. Many conventional recognition systems adopt confidence thresholds that avoid the assignment of a class label in the absence of sufficient evidence for any class, which reduces but does not eliminate this problem.

Like class nodes, feature nodes also have two states, corresponding to presence or absence of the feature. As in a typical naive Bayesian network classifier, arcs are directed from the class node to the feature nodes. To model known dependencies between features, additional arcs connect features where appropriate. Such dependencies arise by the construction of compound features. Specifically, if a geometric compound feature (say, Feature 3 in Figure 4.3) has a component feature that is also present in the network (e.g., Feature 2 in Figure 4.3), then it is clear that the presence of Feature 3 implies a high probability of finding Feature 2 in the scene also. Feature 2 does not necessarily have to be present because the thresholds that define "presence" are determined independently for each feature node (see Definition 4.2 below). An analogous argument holds for the components of a conjunctive feature. In the case of a disjunctive feature, the direction of the argument – and that of the arcs – is reversed. Say, Feature 4 is a disjunctive feature, and Feature 5 is one of its components. Then, finding Feature 5 in a scene implies the likely presence of the disjunctive Feature 4 in the same scene. The reverse is not necessarily true because Feature 4 may be present by virtue of its other subfeature, with Feature 5 being completely absent. Therefore, the causal influence is modeled by an arc from the component feature to the compound.

Note that the same feature (defined by its structure and its response vector) may occur in more than one Bayesian network classifier. Feature values are measured using Equation 3.14

(page 25). Since these feature values are real numbers, they must be discretized by applying a threshold marking the minimum value required of a feature in order to be considered present. This threshold is determined individually for each feature node so as to maximize its distinctive power:

Definition 4.2 (Distinctive Power of a Feature) The *discriminative power* of a feature **f** is the Kolmogorov-Smirnoff distance (KSD) between the conditional distributions of feature values observed under (*a*) the discriminated classes versus (*b*) the characterized class (cf. Equation 4.3 and Figure 4.1). The *distinctive power* of a feature is identical to its discriminative power, if $\mathrm{cdf}(\alpha_{\mathbf{f}} \mid a) > \mathrm{cdf}(\alpha_{\mathbf{f}} \mid b)$ for the cutpoint $\alpha_{\mathbf{f}}$ associated with the KSD, i.e., the feature is in fact characteristic of its class. Otherwise, the distinctive power is defined to be zero. ■

As was noted in Section 4.3.2, this definition yields an optimal cutpoint in the Bayesian sense that if the presence of a class is to be inferred on the basis of this feature alone, the probability of error is minimized, given that the prior probability of the presence of this class equals 0.5. In general, however, the prior probability of a class will not be equal to 0.5. In this case, this choice of a cutpoint is not optimal in any known sense. Nevertheless, Utgoff and Clouse [126] make a strong argument in favor of using the KSD *without paying attention to the prior class probabilities* as a test selection metric for decision trees. It is based on the observation that even in cases where the Bayesian choice is always the same regardless of the test, the cutpoint given by the KSD separates two very different regions of certainty regarding the presence of the class.

Example 4.3 (KSD and Disparate Priors) Consider the class-conditional distributions shown in Figure 4.4. There are many more samples from the discriminated classes than from the characterized class. In other words, given an unclassified sample, the prior probability that it belongs to the discriminated classes is much higher than that it belongs to the characterized class. Where should the Bayes-optimal cutpoint be placed that allows one to guess what conditional distribution generated a given sample? The two distributions fully overlap! Because of this and the disparate priors, it is clear from Figure 4.4 that no matter where a cutpoint is placed, the expected error rate is minimized by always guessing that a sample came from the discriminated classes, irrespective of the sample's feature value. However, it is also obvious that a cutpoint can be found that is meaningful in the sense that almost all samples of the characterized class fall above it, and almost none below it. While a decision rule based on such a cutpoint is not Bayes-optimal, this cutpoint would tell us something about the posterior probability of a sample belonging to each of the two conditional distributions. Utgoff and Clause simply suggest that the simple definition of the KSD given in Equation 4.3 be applied, which – being based on cumulative distributions – ignores the prior probabilities. ■

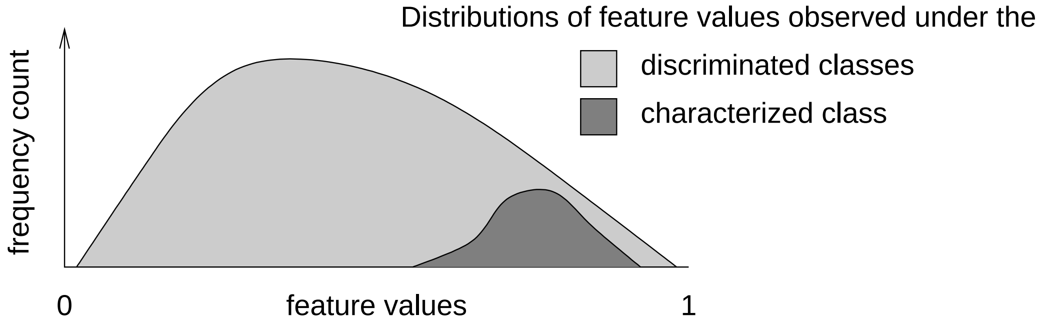


Figure 4.4. Highly uneven and overlapping feature value distributions.

The computation of a KSD requires a description of the two conditional probability distributions. Here, these distributions are approximated by the cumulative experience of the agent:

The agent maintains an *instance list* of experiences. Each image encountered adds to the instance list a new *instance vector* containing the feature values measured and the true class label of this training image. All probability estimates relating to the Bayesian network classifiers are estimated by counting entries in the instance list. This includes the cumulative distributions for estimating the KSDs to discretize feature variables, the conditional probability tables at the feature nodes, and the prior class probabilities at the class nodes.

4.4.2 Recognition

Recognition of a scene can be performed in the conventional way by first measuring the strength of each feature in the image, setting the feature nodes of the Bayesian networks to the corresponding values, and computing the posterior probability of the presence of each class. In this case, the absence of a feature is meaningful to the system. Alternatively, robustness to occlusion can be built into the system by setting only feature nodes corresponding to found features, and leaving the others unspecified. In this case, the posterior probability of these features being present (but occluded) can easily be computed. Incorporating positive evidence into the net will monotonically increase the posterior probability of a class. In particular, the posterior probability is always greater than or equal to the prior. Without effective countermeasures however, in practice this monotonic increase will often result in very high posterior probabilities for many classes. Since the posterior class probabilities are computed independently by separate networks for each class, evidence found for one class does not reduce the posterior belief in any other class. To avoid these problems, in this work both positive and negative evidence is incorporated into the network by setting the applicable feature nodes to the corresponding state, presence or absence.

Computing a feature value in an image is expensive (cf. Equation 3.14) in comparison to propagating evidence in a Bayesian network of moderate size. Moreover, the contribution of individual features to the classification decision is unclear if all of them are computed at once. Therefore, features are computed sequentially, and the classification decisions are made incrementally within each Bayesian network, according to the following algorithm.

Algorithm 4.4 (Sequential Computation of Class Beliefs)

1. Bring the Bayesian network to equilibrium by propagating evidence.
2. Find the maximally informative feature among all features of all classes. Let C and $F_{\mathbf{f}}$ denote two-state random variables representing the presence or absence of a class and of a feature $\mathbf{f}$, respectively. Then, the maximally informative feature $\mathbf{f}_{\text{max}}$ is the one that maximizes
$$I_{\text{max}} = \max_{\mathbf{f}} \max_{C} I(C, F_{\mathbf{f}})$$
where $I(C, F_{\mathbf{f}})$ is the mutual information defined in Equation 4.2 on page 33.
3. If $I_{\text{max}} = 0$, then stop.
4. Measure the value of this feature $\mathbf{f}_{\text{max}}$, $f_{\mathbf{f}_{\text{max}}}$, in the current image using Equation 3.14.
5. Instantiate all nodes corresponding to $\mathbf{f}_{\text{max}}$ in all networks to the appropriate values by comparing $f_{\mathbf{f}_{\text{max}}}$ against the respective thresholds $\alpha_{\mathbf{f}}$ (cf. Definition 4.2).
6. To keep track of the usage of each feature, label $\mathbf{f}_{\text{max}}$ with the running number of the current training image. Then, continue with Step 1.

■

Note that at each iteration of this algorithm, one feature is instantiated. For an instantiated feature $\mathbf{f}$, the uncertainty in its characterized class C is $H(C) = H(C \mid F_{\mathbf{f}})$, and thus $I(C, F_{\mathbf{f}}) = 0$ (Equation 4.1). Therefore, the algorithm stops at Step 3 after all features have been instantiated.

In practice, it often stops sooner because among the uninstantiated features there are none with nonzero mutual information. This results if the entropies in all of the class nodes have been reduced to zero, or if none of the remaining features are informative with respect to any class variable with nonzero entropy. The labeling in Step 6 allows the algorithm to detect obsolete features. If a feature ceases to be used, it can be discarded. This will further be discussed in Section 4.5.

There are various ways to add sophistication to this basic algorithm. Instead of consulting features exhaustively, a stopping criterion can be introduced that terminates Algorithm 4.4 once $I(C, F_{\mathbf{f}})$ drops below a threshold at Step 2. By definition, features with low mutual information have little potential to affect the posterior class distributions. The tradeoff between computational expense and confidence in the recognition result can be adjusted in terms of this threshold. Another computational tradeoff concerns the relative cost of measuring feature values. Large compound features consisting of many subfeatures are more costly to evaluate than smaller features, as are Boolean compounds relative to geometric compounds (see Section 3.5.1 on page 26, and Section 3.5.2). This cost can be considered at Step 2 of Algorithm 4.4, for example using the framework of decision theory [102, 98]. Such thresholds and costs depend on specific applications. For simplicity, these issues are avoided here.

Since each class is represented by its own Bayesian network, the possibility of multiple classes present in a scene is built into the system. The presence of each class is determined independently of all others. The possibility of arbitrary numbers of target classes within a scene makes the evaluation of recognition results somewhat more complicated than the conventional correct/wrong dichotomy. A recognition result is derived from the posterior class probabilities according to the following definition:

Definition 4.5 (Evaluation of Recognition Results) Those classes with a posterior probability greater than 0.5 are said to be *recognized.* A class is called *true* if it is present in the scene, and *false* otherwise. Each recognition falls into exactly one of the following categories:

correct: Exactly the true classes are recognized.

wrong: The correct number of classes are recognized, but these include at least one false class. At least one true class was missed.

ambiguous: All of the true classes are recognized, plus one or more false classes.

confused: None or some (but not all) of the true classes, and some false classes are recognized. The number of recognized classes is different from the number of target classes present in the scene. (Otherwise, it is considered a *wrong* recognition.)

ignorant: None or some (but not all) of the true classes, and none of the false classes are recognized.

■

The category of *wrong* recognitions constitutes a special case of misrecognitions that are otherwise in the *confused* category. Wrong recognitions are important in that they contain pairwise errors where one class is mistaken for one other class. In a classical forced-choice scenario, this is the only type of incorrect recognitions.

In conventional forced-choice classification, there are only correct and wrong categories. Since this system has the additional freedom to assign multiple class labels or no label at all, the number of correct recognitions may be lower than in a forced-choice situation. To facilitate comparison with forced-choice scenarios, at least the number of ambiguous recognitions should somehow be taken into account. These contain the right answer, which is valuable especially if the number of false positive labels in an ambiguous recognition is low. Similarly, an ignorant recognition should contribute to the same extent as a random result. This idea is formalized in the following definition.

Definition 4.6 (Accuracy) A *recognition result* is a set of classes recognized in an image I by Algorithm 4.4. This set includes n_t true positives and n_f false positives, where $n_t + n_f \leq n_c$, the total number of classes known to the system. Let n_T denote the actual number of target objects depicted in the scene. Then,

$$\mathrm{acc}(I) = \begin{cases} \dfrac{n_T}{n_c} & \text{if } n_t + n_f = 0 \\ \dfrac{n_t}{n_T + n_f} & \text{otherwise} \end{cases}$$

is the *accuracy* of this recognition result. The accuracy achieved on a set $\mathcal{I}$ of images is given by

$$\mathrm{acc}(\mathcal{I}) = \frac{1}{|\mathcal{I}|} \sum_{I \in \mathcal{I}} \mathrm{acc}(I).$$

■

This definition is a generalization of the notion of correct recognition that values ambiguous recognitions according to the number of false positives involved. An ignorant recognition is regarded as equivalent to a fully ambiguous recognition, i.e., a result that lists all n_c class labels. The accuracy is a value between zero and one, where the maximum corresponds to 100% correct recognition. If all recognitions are ignorant or fully ambiguous, then the accuracy is the inverse of the number of classes known to the system. If $n_T = 1$ everywhere, this is identical to the chance-level proportion of correct recognitions attained by uniformly random forced-choice classifiers. Thus, the accuracy can be directly compared to the percentage-correct result delivered by a conventional forced-choice classifier.

4.4.3 Feature Learning Algorithm

In the incremental learning scenario (Figure 4.2 on page 35), the agent receives training images one by one. Initially, the agent does not know about any classes or features. When it is presented with the first image, it simply remembers the correct answer. When it is shown the second image, it will guess the same answer. This will continue, and no features will be learned, until this answer turns out to be incorrect.

Algorithm 4.7 (Operation of the Learning System) This algorithm provides details of the basic model of incremental discrimination learning shown in Figure 4.2, page 35.

1. A new training image is acquired, denoted I.
2. Image I is run through the recognition algorithm 4.4 as described in Section 4.4.2. If recognition is correct according to Definition 4.5, then continue at Step 1.
3. Assume recognition failed because of inaccurate conditional probability estimates stored in the Bayesian networks. To remedy, all feature nodes are re-discretized to maximize their distinctive power (cf. Definition 4.2, page 37). Their conditional probabilities are estimated by counting co-occurring feature values and class labels in the instance list (cf. Section 4.4.1, page 38). In the same way, the prior probabilities associated with the class nodes are re-estimated.
4. Re-run the recognition algorithm 4.4 on the image I. If I is now recognized correctly, then continue at Step 1.
5. Conclude that at least one new feature must be added to at least one Bayesian network to recognize I correctly. Attempt to do this by invoking Algorithm 4.8 (described below). Then, continue at Step 1.

■

This algorithm re-estimates the conditional probability tables of the Bayesian networks only when an image is misrecognized (Step 3). Alternatively, after each recognition – successful or otherwise – the conditional probability tables of all nodes related to queried features could be updated, maintaining a best estimate of the actual probabilities at all times. A drawback of this approach is that the behavior of the system would fluctuate even in the absence of any mistakes. For example, two consecutive presentations of the same image could result in two different recognition results. Since it is easier to characterize the behavior of the system if its behavior does not depend on a recent history of correct recognitions, it is designed to learn only from mistakes.

At Step 5 in this algorithm, the objective is to derive a new feature to discriminate one or more true classes from one or more *mistaken* classes, i.e., those false classes that were erroneously recognized. Two cases must be distinguished. In the first case, a true class is not among the recognized classes (wrong, confused, or ignorant recognition). The agent needs to find a new feature of this true class in the current image – the *sample image* – that, if detected, will cause this class to be recognized. In the second case, some false classes were recognized in addition to the true classes (ambiguous recognition). Here, the agent needs to find a feature in an image of a false class that is not present in the currently misrecognized image, and that, if not detected, will prevent this false class from being recognized. Again, this image is called the sample image.

Algorithm 4.8 (Feature Learning)

1. For the purpose of learning this feature, the true and mistaken classes are (re)defined so that there is generally exactly one true and one mistaken class. The classes selected are the true class with minimum belief and the false class with maximum belief, respectively, representing the most drastic case. In either case, if there is no such unique class, belief values from earlier (non-final) recognition stages (Algorithm 4.4) are consulted. In particular, this covers the case of ignorant recognition. See below for further discussion and an example.
2. A set of *example images* is retrieved from the environment. This set contains n_{ex} random views of each of the true class and the mistaken class. The precise value of n_{ex} is unimportant, but involves a tradeoff that will be discussed below.
3. A new feature is generated, generally by random sampling from the sample image, using Algorithm 4.10. Algorithm 4.10 may be called only a limited number of times within a single execution of Algorithm 4.8. If this limit is exceeded, Algorithm 4.8 terminates unsuccessfully. The details will be discussed below in the context of Algorithm 4.10.
4. A new node representing this new feature is added to the Bayesian network that models the class of the sample image, along with an arc from the class node and any arcs required to or from any other feature nodes, to model any known interdependencies with pre-existing features (cf. Section 4.4.1).
5. The values of this feature, as well as any existing features represented by nodes linked to or from the new feature node in the Bayes network, are measured within each example image. The resulting instance vectors are added to the instance list.
6. The new feature node is discretized such that the cutpoint maximizes the distinctive power between the true and mistaken classes (see Definition 4.2 on page 37), as estimated by counting entries in the instance list. If the distinctive power is zero, the algorithm continues at Step 3.
7. The conditional probability tables of the affected nodes are (re)estimated by counting entries in the instance list.
8. The recognition procedure (Algorithm 4.4) is run on the sample image, using the added feature.

(a) If the sample image is of a true class that is erroneously not recognized, or if the sample image is of a false class that is erroneously recognized, then the new feature node is removed from the Bayes net. However, the definition of the new feature (consisting of its structural description and response vector) is retained until this algorithm terminates. Now, it proceeds with Step 3.

(b) If the image is of a true class and is now recognized successfully, or if it is of a false class and is successfully not recognized, then this algorithm terminates successfully.

■

Note that this feature search procedure requires that each new feature individually solves an existing recognition problem. This results in a strong tendency to learn few but informative features. Each feature is learned specifically to distinguish one class from one other class (see Step 1). Instead of using just one true and one mistaken class, one could use all misrecognized classes, or even all true and false classes. However, in general it will be much easier to find features that discriminate well between pairs of classes, than features that discriminate between two groups of several classes. In practice, this may in fact be all that is required to solve a given visual task. Moreover, using only a pair of classes ensures that the execution time of this algorithm is independent of the total number of classes.

Example 4.9 (Selecting True and Mistaken Classes) Table 4.1 shows some examples of how the characterized and discriminated classes are chosen in Step 1 of Algorithm 4.8. The first case shows a wrong recognition result. Since all classes with posterior beliefs greater than 0.5 are said to be recognized, the system erred on classes B and C. Here, it is clear that a new feature needs to be learned for class B that will cause class B to be recognized in the current image. This new feature should be discriminative with respect to class C, because the system mistook class B for class C. Note that the new feature will not affect the recognition result for class C (nor for any other class but the characterized class B). Likely, it will cause an ambiguous result like that shown in the second row. Now, a feature must be learned that is characteristic of class C, and that, if not found in an image, will prevent it from being recognized as class C. Class A is chosen as the discriminated class because it has the lowest posterior belief among all true classes, and is therefore presumably most likely confused with class C.

Table 4.1. Selecting true and mistaken classes using posterior class probabilities. Bold type indicates the characterized class of the new feature to be learned. The discriminated classes are shown in italics.

	True Classes		False Classes		
	A	B	C	D	E
wrong	0.7	**0.4**	*0.8*	0.3	0.2
ambiguous	*0.7*	0.9	**0.8**	0.3	0.2
ignorant	**0.3**	0.4	*0.4*	0.3	0.2
	0.4	**0.4**	*0.3*	*0.3*	0.2
ignorant	0.0	0.0	0.0	0.0	0.0

The first (wrong) example would constitute a confused recognition if, for example, class D had received a posterior belief of greater than 0.5. The third example is an ignorant recognition. Like in wrong and confused cases, the characteristic class is the true class with lowest posterior belief, and the discriminated class is the false class with highest posterior belief. In the final example, Algorithm 4.4 resulted in zero belief for all classes. In such a case where true and characteristic classes cannot be determined from the final recognition result, the system consults the intermediate recognition result generated by the penultimate iteration of Algorithm 4.4. This result is guaranteed to contain beliefs with nonzero entropy. In this example, there are two potential characteristic classes because both true classes have identical belief. In this case, one of them is chosen at random because Algorithm 4.4 requires a single sample image. There are also two potential discriminated classes, both of which are simply used. In practice, such ambiguous cases occur very rarely. ■

The number of example images per class, n_{ex} (Algorithm 4.8, Step 2), determines how accurately the conditional probabilities associated with the new feature are estimated at Step 7. A large value yields accurate estimates, but these are expensive to evaluate. The danger of using too small a value of n_{ex} is that many good features are discarded because of overly pessimistic probability estimates. Optimistic estimates result in the addition of features that later turn out to be of little value. Consequently, these will cease to be used as they are superseded by more discriminative features. Thus, using a small n_{ex} will result in more feature searches, each of which will take less time than when using a large n_{ex}. In a practical interactive application, the choice of this parameter should depend on the relative cost of acquiring images vs. generating experience in the real world. If experience is costly to obtain (e.g. because it involves sophisticated maneuvers with physical manipulators), but a variety of representative images are cheaply obtained at Step 2 of Algorithm 4.8, then n_{ex} should be chosen large. On the other hand, if each real-world experience is generated cheaply but example images are expensive to obtain, then a small value of n_{ex} should be used. The value does not need to be fixed, nor does it need to be the same for the true and the mistaken class. All experiments reported here use $n_{\text{ex}} = 5$. This value was chosen relatively small because it emphasizes the incremental nature of learning even if a small, fixed training set is used. At the other extreme, n_{ex} could be chosen so large that the example images always include the entire training set (so it exists), yielding immediately accurate estimates of the distinctive power of features without the need for incremental updating.

It remains to be explained how a feature is generated at Step 3 of Algorithm 4.8. There are five ways to generate a new feature that differ in the structural complexity of the resulting feature. In accord with *Occam's razor*, simple features are preferred over complex features. Also, they are more likely to be reusable and are computed more rapidly than complex features. The following algorithm implements a bias toward structurally simple features by considering features in increasing order of structural complexity.

Algorithm 4.10 (Feature Generation) These methods are applied in turn at Step 3 of Algorithm 4.8:

1. **(Reuse)** From a Bayesian network representing an unrelated class (other than the true or mistaken classes), select a random feature that is not in use for the true or mistaken classes, to see if it can be reused for the current recognition problem. This promotes the emergence of general features that are characteristic of more than one class.
2. **(New Primitive)** Sample a new feature directly from the sample image by either picking two points and turning them into a geometric compound of two edgels, or by picking one point and measuring a texel feature response vector. (Recall that features are measured under rotational invariance. Therefore, individual edgels do not carry any information.) This is the only step where new features are generated from scratch.
3. **(Geometric Expansion)** From among the features currently in use, or from among the failed candidate features tried earlier during this execution of Algorithm 4.8 (see Algorithm 4.8, Step 8a), choose a feature at random. Find a location in the sample image where this feature occurs most strongly (Equation 3.14), and expand it geometrically by adding a randomly chosen salient edgel or texel nearby. This expansion yields a new feature that is more specific than the original feature.
4. **(Conjunction)** Pick two random existing features and combine them into a conjunctive feature. Again, the result is a new feature that is more specific than either of the constituent features individually.

5. **(Disjunction)** Pick two random existing features and combine them into a disjunctive feature. Here, the result is a new feature that is less specific than the constituent features. ■

For a given run of Algorithm 4.8 (Feature Learning, page 41), each time Algorithm 4.8 reaches its Step 3, one method of Algorithm 4.10 is applied. The first n_{fgen} times Algorithm 4.8 reaches Step 3, Method 1 of Algorithm 4.10 is performed; the next n_{fgen} times, Method 2, and so on. This continues until either a suitable feature is found, or until the last method has been applied n_{fgen} times, at which point the feature search (Algorithm 4.8) terminates unsuccessfully. The parameter n_{fgen} in effect controls the bias toward simple features. A large value will expend more effort using each method, while a small value will give up sooner and move on to the next method. The current implementation uses $n_{\text{fgen}} = 10$, which is considered to be a relatively small value in relation to the roughly 100–1000 salient points present in typical images used here (cf. Chapter 3).

The specific sequence of methods defined in Algorithm 4.10 was chosen because it implements a bias toward few and structurally simple features. A new feature is only generated if the system fails to find a reusable feature. Before existing features are expanded, new primitive features are generated. Geometric expansions are preferred over conjunctive features because the former are local and rigid, while the latter can encompass any variety of individual features. Local features are more robust to partial occlusions and minor object distortions than are non-local features. Nevertheless, conjunctive features may be important because they encode the presence of several subfeatures without asserting a geometric relationship between them. This makes them more robust to object distortions than geometric features.

The last step involves the generation of disjunctive features. These are susceptible to overfitting because a single disjunctive feature can in principle encode an entire class as a disjunction of highly specific features, each characterizing an individual training image. Thus, they can help learn a training set without generalization. On the other hand, disjunctive features are necessary in general because objects within a class may differ strongly in their appearance. For example, a disjunctive feature can encode a statement such as “If I see a dial *or* a number pad, I am probably looking at a telephone.”

4.4.4 Impact of a New Feature

In Algorithm 4.8 (Feature Learning), Step 3, a new feature is generated in the hope that it will fix the recognition problem that triggered the execution of Algorithm 4.8. Clearly, a necessary condition for success is that the new feature is actually consulted by the recognition algorithm 4.4, when called at Step 8 of Algorithm 4.8. Algorithm 4.4 continues to consult features as long as there are features that can potentially reduce the uncertainty in any class nodes. Therefore, the new feature will be queried if the entropy in its own class node does not vanish as a result of querying other features of this Bayesian network. The only way Algorithm 4.4 can terminate without querying the new feature is if there is a feature **f** in this Bayesian network that drives the entropy in its class node to zero, and that has a greater mutual information with respect to the class node than the new feature, causing it to be queried before the new feature. Note that this recognition is incorrect, as it triggered the present execution of the feature learning algorithm 4.8 at Step 5 of Algorithm 4.7 (Operation of the Learning System). The newly generated feature is not going to be used on this image, irrespective of any discriminative power it may have.

How likely is this worst-case situation to occur? Consider a case where the system failed to recognize a true class. Figure 4.5 illustrates the class-conditional cumulative probabilities of a feature **f** that can drive the entropy in the class node to zero, concluding that its class is absent. The critical property of the graph is that if the class is present, the feature **f** is always present with $f_{\mathbf{f}} > \alpha$, allowing one to infer the absence of the class if $f_{\mathbf{f}} \leq \alpha$. (The figure also illustrates that

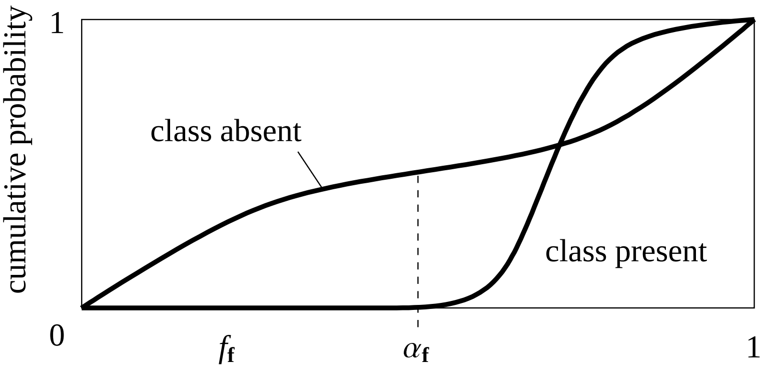


Figure 4.5. KSD of a feature **f** that, if not found in an image, causes its Bayesian network to infer that its class is absent with absolute certainty.

the KSD of such a feature can be quite low – roughly 0.5 in this case.) Can this feature prevent the successful generation of new features by keeping them from getting queried? The answer is no because it will quickly lose its power to reduce the entropy to zero. To see this, observe that each processed image generates an instance that is added to the instance list, recording the measured feature values together with the true class labels. In the situation described here, $f_\mathbf{f} \leq \alpha_\mathbf{f}$, but the class was present. This situation is incompatible with the critical property of Figure 4.5. Consequently, the next time the conditional probabilities of **f** are re-estimated based on the instance list, at least one of two changes must occur: The probability of $f_\mathbf{f} \leq \alpha_\mathbf{f}$ given the presence of the class becomes greater than zero, or the threshold $\alpha_\mathbf{f}$ is reduced. In the first case, the feature immediately loses its power to determine the absence of the class with absolute certainty. The second case clearly cannot reoccur indefinitely because $\alpha_\mathbf{f}$ is chosen to maximize the KSD, ensuring that $f_\mathbf{f} < \alpha_\mathbf{f}$ for the bulk of those cases where the class is absent. As specified in Algorithm 4.7 (Operation of the Learning System) on page 40, the cutpoints α and the conditional probability tables of all feature nodes are re-estimated after every incorrect recognition.

In summary, a newly sampled feature is not guaranteed to be used immediately. However, those cases where it is not used can be expected to be rare, since they require the presence of a feature **f** with the specific properties described above. Importantly, any such degenerate situation is guaranteed to resolve itself soon because the feature node parameters are re-estimated after every incorrect recognition. This self-repairing property is a key aspect of the design of the learning algorithm, and is critical to its successful operation.

4.5 Experiments

The algorithms developed in the preceding sections were developed for open task scenarios in which images are encountered while the agent performs a task. In effect, each newly acquired visual scene first serves as a test image that the agent attempts to recognize. If it succeeds, the agent moves on. If the result of recognition is incorrect, this scene becomes a training image. Additional example images are then acquired to enable the agent to generate an initial estimate of the distinctive power of newly generated features.

Testing this algorithm on such an interactive task is a nontrivial endeavor. Implementing a robot that can interact in meaningful ways with many objects is challenging in its own right and beyond the scope of this dissertation. The results cited in this section were generated on simulated tasks.

A second simplification concerns the number of objects shown in an image. The algorithms described above make no assumptions regarding the number of target or distractor objects shown present in any given image. For simplicity and to facilitate comparison of the results with conventional methods, all experiments were performed in the conventional way, with each image containing exactly one object. It is important to note however that the learning system did not take advantage of this in any way. The algorithms were applied in their full generality as introduced above.

A simulated task consists of a closed set of images. This set of images was divided into equally sized training and test sets, such that the number of images per class was identical to the extent possible. Each experiment was run a second time with the roles of training and test sets reversed. In the machine learning terminology, this procedure is called two-fold stratified cross validation. Training took place as shown in the overview given in Figure 4.2 on page 35, and as described in detail in Section 4.4.3. Images from the training set were presented to the system one by one, cycling through the training set in random order. A new permutation was used for each cycle. On receipt of an image, the system analyzed it using Algorithm 4.4. If the recognition result was not correct (in the sense of Definition 4.5 on page 39), then a new feature was sought according to Algorithm 4.8. At Step 2 of this algorithm, the example images were chosen at random from the training set (not from the cycling order). The system cycled through the entire training set until all recognition results were correct.

The empirical performance of the learning system are discussed in terms of the evaluation categories given in Definition 4.5, which apply to the general case of an arbitrary number of target classes. Since all experiments were performed using only a single target class, these categories are restated here for this special case, to simplify their interpretation.

correct: Exactly the true class is recognized.

wrong: Exactly one false class is recognized.

ambiguous: The true class is recognized, plus one or more false classes.

confused: More than one false classes are recognized, but not the true class.

ignorant: No class is recognized.

Experiments were conducted using three data sets with very different characteristics. These will be discussed in the following sections. The next chapter will introduce a simple extension to the basic learning algorithm that results in pronounced performance improvements.

4.5.1 The COIL Task

The COIL task consisted of 20 images of each of the first six objects of the 20-object Columbia Object Image Library [79]. One sample image of each class, representing the middle view, is shown in Figure 4.6. Neighboring views are spaced 5 degrees apart, at constant elevation. Neighboring images were assigned alternately to training and test sets. All images are of size 128×128 pixels.

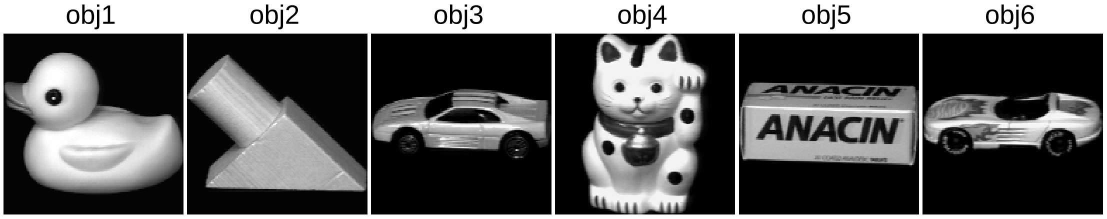


Figure 4.6. Objects used in the COIL task. Shown are middle views from the sorted sequence of viewpoints.

Table 4.2 summarizes important performance parameters for the COIL task (and also for the other two tasks, Plym and Mel, but these will be discussed later). The most striking observation is that the trained recognition system is very cautious in that it makes very few mistakes. The proportions of wrong and ambiguous recognitions are both very low. Unfortunately, the proportion of correct recognition is also fairly low at about 50%. In the remaining 33% ignorant cases, no class was recognized at all.

Table 4.2. Summary of empirical results on all three tasks. The first four entries give the name of the task, the number of classes, the fold index, and the accuracy according to Definition 4.6. The five entries under "Recognition Results" in each row sum to one, barring roundoff errors. Zero entries are left blank for clarity. The columns under the heading "# Features" give the total number of feature nodes in all Bayesian networks (BN), the total number of different features (dif.), the average number of features queried (qu'd) during a single recognition procedure (Algorithm 4.4), and the total cumulative number of features sampled (sp'd), respectively. The rightmost two columns list the number of cycles through the training set, and the total number of training image presentations.

				Recognition Results					# Features				Tr. Set	
Task	Cls.	Fold	acc.	cor.	wrg.	amb.	cnf.	ign.	BN	dif.	qu'd.	sp'd.	cyc.	imgs.
COIL	6	1	.58	.48	.08	.08		.35	46	35	22.1	471	6	360
COIL	6	2	.61	.52	.08	.10		.30	60	40	22.1	911	8	480
COIL	6	avg.	.60	.50	.08	.09		.33	53	38	22.1	691	7	420
Plym	8	1	.57	.50	.06	.05	.02	.38	84	59	22.0	977	8	448
Plym	8	2	.61	.52	.02	.09		.38	86	55	32.1	1019	8	512
Plym	8	avg.	.59	.51	.04	.07	.01	.38	85	57	27.1	998	8	480
Mel	6	1	.49	.36	.14	.11		.39	50	34	17.4	608	8	288
Mel	6	2	.47	.33	.08	.14		.44	47	29	19.6	477	6	216
Mel	6	avg.	.48	.35	.11	.13		.42	48	32	18.5	543	7	252

The next two columns give the number of feature nodes summed over all class-specific Bayesian network classifiers, and the number of features that are actually different. The difference between these two numbers corresponds to features that are used by more than one classifier (see Algorithm 4.10, Feature Generation, Step 1). Here, about one-third of all feature nodes share the same feature with at least one other node. How many of these features are actually queried during an average recognition procedure (Algorithm 4.4)? The next column shows that on average, only a little more than half of all available features are queried before the entropies in the class nodes are reduced to zero, or until none of the unqueried features has any potential to reduce a nonzero entropy. On average, the classifier system ended up with about 9 feature nodes per class (53 feature nodes / 6 classes).

The rightmost three columns in Table 4.2 give a flavor of the total number of features generated during learning, the number of iterations through the training set performed until it was perfectly learned, and the total number of training images seen by the learning system. These numbers become more meaningful in the context of the following chapter.

Table 4.3 provides deeper insight into the recognition performance, separately for each class. Interestingly, it shows that the performance characteristics differ widely between the two folds of the two-fold cross validation. For example, in Fold 1 almost all instances of obj3 were recognized correctly, while in Fold 2 most of them were not recognized at all. However, some trends can still be found: Performance was consistently high on obj2, and consistently low on obj6 and obj4. Moreover, instances of obj4 tended to be misclassified as obj6, but not vice versa. The variation between the two folds is mostly due to the randomness inherent in the feature learning algorithm. This topic will be further discussed in Section 4.6 and in Chapter 5.

Table 4.3. Results on the COIL task. The table shows, for each class, how many instances were recognized as which class for correct and wrong recognitions, and the number of ambiguous/confused/ignorant recognition outcomes. In each row, these entries sum to the number of test images of each class (here, 10). The rightmost column shows the number of feature nodes that are part of the Bayesian network representing each class. The bottom row of the bottom panel gives the average proportion of the test set that falls into each recognition category.

	Confusion Matrix correct/wrong									# Feats.
Class	obj1	obj2	obj3	obj4	obj5	obj6	amb.	cnf.	ign.	BN
Fold 1:										
obj1	6								4	8
obj2		8					1		1	4
obj3			8						2	9
obj4			1	2	1	2	2		2	11
obj5					3		2		5	5
obj6			1			2			7	9
Sum	6	8	10	2	4	4	5	0	21	46
Fold 2:										
obj1	5								5	10
obj2		10								7
obj3			2		1				7	16
obj4				4		2	2		2	9
obj5					7		2		1	8
obj6					2	3	2		3	10
Sum	5	10	2	4	10	5	6	0	18	60
Average of both folds (proportions):										
obj1	.55								.45	9
obj2		.90					.05		.05	6
obj3			.50		.05				.45	13
obj4			.05	.30	.05	.20	.20		.20	10
obj5					.50		.20		.30	7
obj6			.05		.10	.25	.10		.50	10
Avg.	.09	.15	.10	.05	.12	.08	.09	.00	.33	9

In the cases of ambiguous recognition, which classes were the false positives? The answer is given by Table 4.4. In both folds, obj4 and obj5 could often not be disambiguated from each other. These wrong and ambiguous recognitions do not seem intuitive, given that they appear very dissimilar to a casual observer (Figure 4.6). However, in Fold 1, obj4 shares one feature with each of obj5 and obj6. This is an indication that at some level, these classes do seem similar to the learning system. Even more strongly in Fold 2, obj4 shares one feature with obj5, two features with obj6, and one more feature with both obj5 and obj6. Examples of the features learned for each class are shown in Figure 4.7. Edgels and texels and their orientations and scales are represented as in Figure 3.9 on page 22. In addition, the subfeatures of a compound feature are linked with a line, that is solid for geometric and dotted for Boolean compounds.

Table 4.4. Results on the COIL task: Confusion matrices of ambiguous recognitions. The italic numbers on the diagonal indicate how many times the class given on the left was recognized ambiguously (cf. Table 4.3), and the upright numbers in each row specify the false-positive classes.

Class	obj1	obj2	obj3	obj4	obj5	obj6
Fold 1:						
obj1						
obj2		*1*	1			
obj3						
obj4				*2*	1	1
obj5			1	1	*2*	
obj6						
Fold 2:						
obj1						
obj2						
obj3						
obj4				*2*	2	1
obj5				1	*2*	1
obj6			1		1	*2*

To a casual human observer, the two cars (obj3 and obj6) look very similar. Why does the learning system not seem to have any difficulty with these two objects? The answer is that the similarity occurs at a high level, concerning the overall shape of the body, the location of the wheels, etc. However, the learning system does not have direct access to these high-level features. It only looks at a small number of strongly localized features, and at combinations of these. At the level of local features, the two cars in fact look rather different even to humans.

What are the appearance characteristics captured by individual features? Figures 4.8 and 4.9 give an intuition. Figure 4.8 shows all images of class obj1. Each image displays all features characteristic of this class that is present in this image (i.e., $f_{\mathbf{f}} > \alpha_{\mathbf{f}}$; cf. Algorithm 4.4, Step 5, page 38). Each feature is annotated with a unique identifier k (feature k was the kth feature learned by the system). The images show that some features are reliably located at corresponding parts of the duck (features 5, 8, 25). Other features change their location with the viewpoint (features 11, 13). Most features appear on most views, while feature 22 only occurs in a few images. Feature 9 is a disjunctive feature. In most views, its dominant subfeature is a large-scale geometric compound of two edgels. Only the first image shows its other subfeature, a texel.

Figure 4.9 illustrates the specificity of a representative subset of the features characteristic of class obj1. The two center columns indicate how strongly a feature responded everywhere in the image. For purposes of illustration, the feature response value was computed at all image points (not just at salient points). At each point, all scales corresponding to scale-space maxima

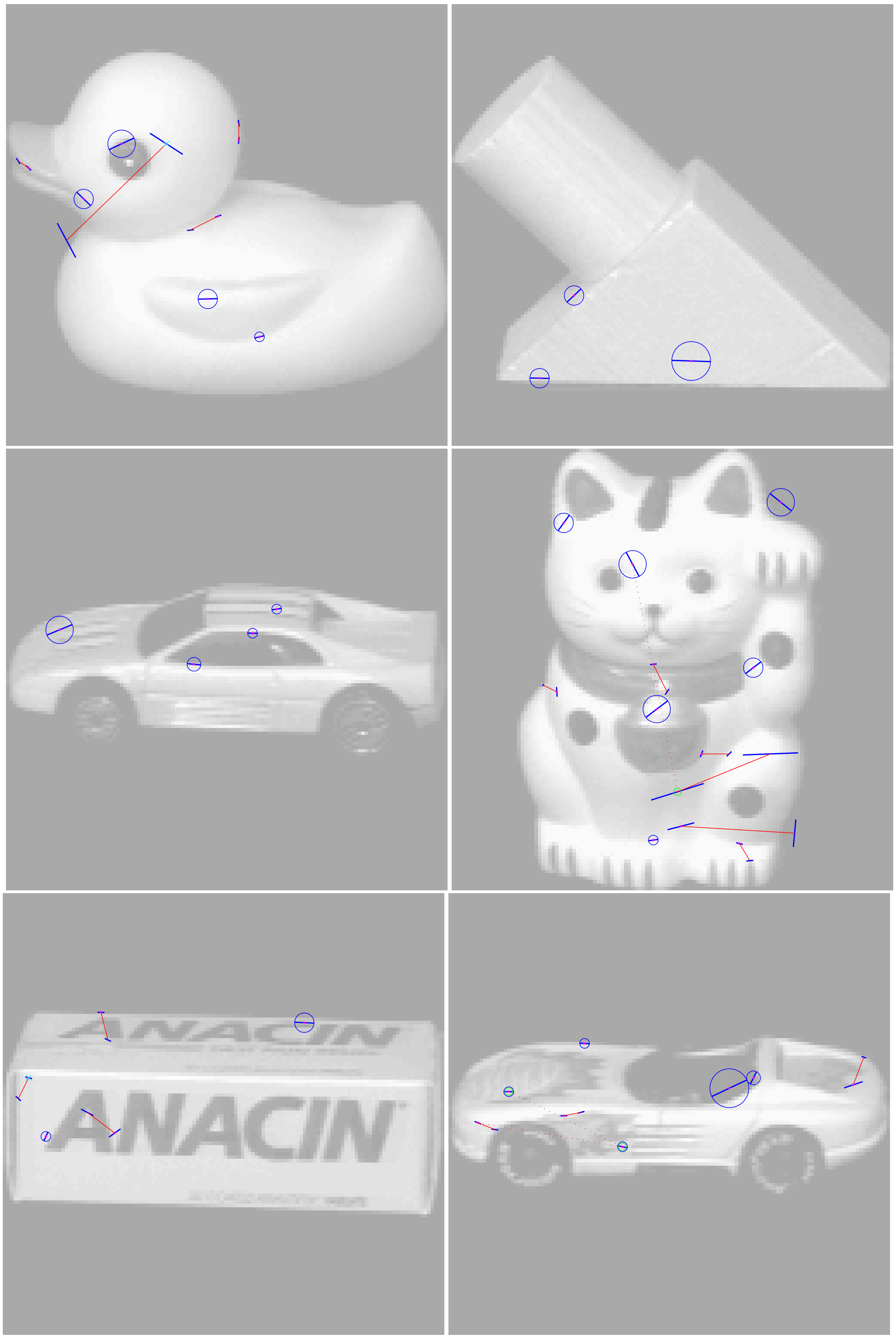


Figure 4.7. Examples of features learned for the COIL task. For each class, all features characterizing this class in Fold 1 are shown that are considered to be present in the image by the classifier.

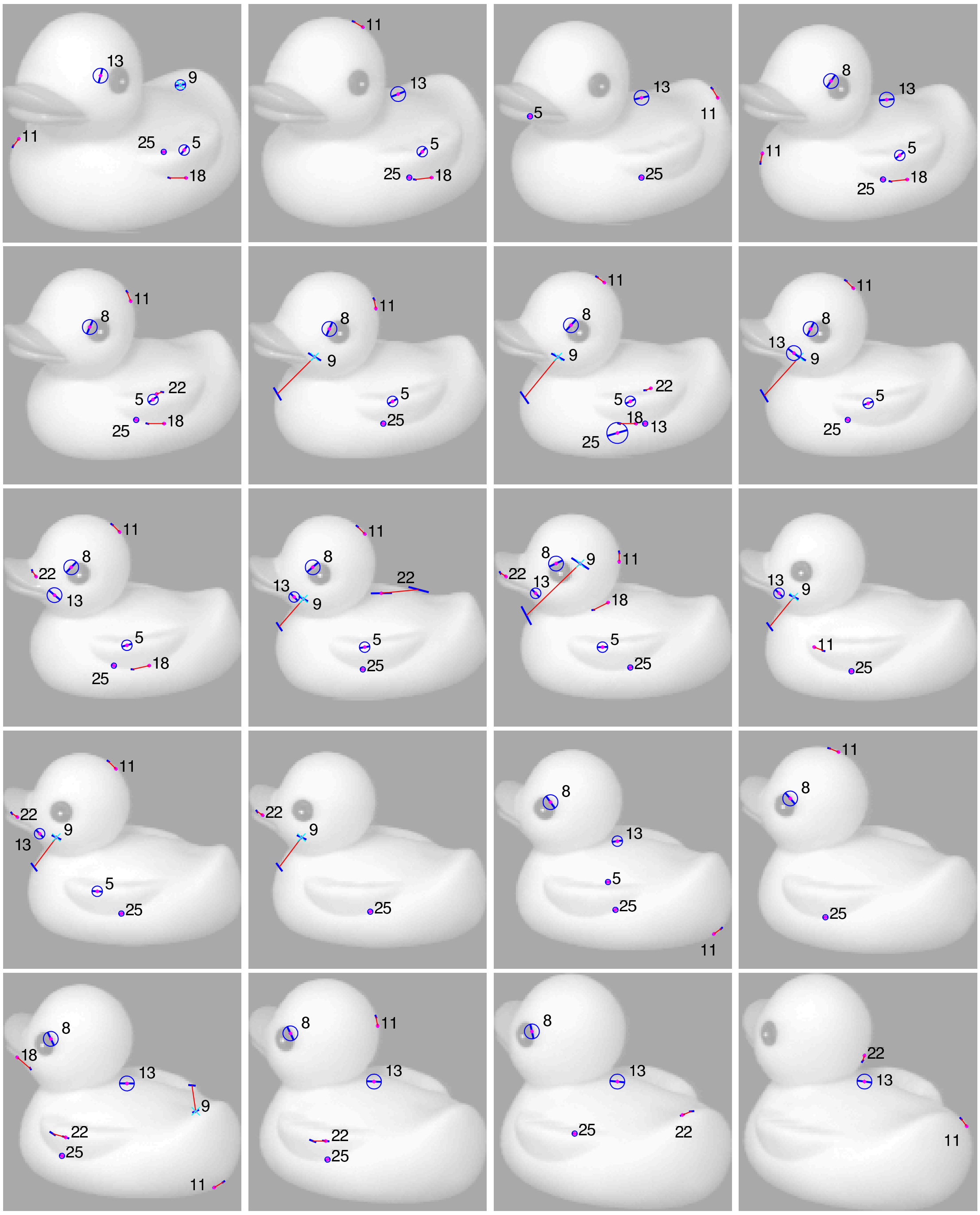


Figure 4.8. Features characteristic of obj1 located in all images of this class where they are present. These features were learned in Fold 1, and are here identified by unique ID numbers.

were considered. The gray value at a pixel corresponds to the maximum feature value $f_{\max}$ over these scales. The location of a feature is identified with the location of its reference point (cf. Figure 3.12, page 26). To enhance detail, the gray value is proportional to $f_{\max}^4$, as most feature responses are quite close to unity. A pixel was left white if the scale function did not assume any local maxima at this image location. The figure shows that some features are more specific than others. Feature 13 is the least specific feature characteristic of obj1. It responds strongly in many areas both around the periphery and in the interior of the duck, and also the cat. Feature 5 is the most specific feature of obj1, and responds primarily to certain areas interior to the duck. This feature is very similar to feature 25. Not accidentally, feature 13 assumed its strongest response at various places in Figure 4.8, while features 5 and 25 are stable with respect to their location. Feature 18 is an example of an edge feature. In fact, its response is much more strongly localized to edges than the texel features shown in Figure 4.9. Feature 9 responds only weakly everywhere, as its constituent features were sampled from other images. Its presence threshold $\alpha_{\mathbf{f}}$ is correspondingly lower (not shown in the figure).

4.5.2 The Plym Task

The second task used eight artificially rendered, geometric objects,[2] that are shown in Figure 4.10. There are 15 views per object, covering a range of ±10 degrees about the vertical axis relative to the middle position shown in Figure 4.10 in steps of two degrees, at constant elevation. Two views are rendered from elevations 10 degrees above and below the middle position, and two views at 10 degrees in-plane rotation. The object surfaces are free of texture. This makes the Plym task difficult to learn using the feature space used here, as the only discriminant information available to the learning system is given by local contour shading and angles between edges – and both are adversely affected by perspective distortions. The rendering viewpoints are closer together and cover a smaller section of the viewing sphere than the COIL task. Again, neighboring views were assigned alternately to training and test sets, yielding one training/test set with 8 and one with 7 images per class. The objects are rendered at similar size, and the images were cropped to leave a 16-pixel boundary around the object, resulting in image sizes of about 100×170 pixels.

Table 4.2 reveals that the performance of the learning system on the Plym task was very similar to the COIL task. There was about the same number of correct recognitions, but slightly fewer wrong and ambiguous recognitions, which is reflected by a higher proportion of ignorant results. One of the very few *confused* recognition results ever encountered occurred here: A cucon was recognized as a cucy and a cyl3. Incidentally, in this fold the cucon Bayesian network shared one feature each with each of the cucy and cyl3 networks, which indicates that these classes were considered similar by the learning system. All other numbers shown in Table 4.2 are roughly in proportion to the COIL task.

The confusion matrices (Table 4.5) again reveal striking consistencies and differences between the two folds. For example, cyl3 was perfectly recognized in Fold 1, but poorly in Fold 2. Object cycu was almost perfectly recognized in both folds. This object is characterized by a unique rectangular protrusion attached to a smooth surface. This turned out to be a powerful clue to the learning system: The compound feature (from Fold 2) sitting precisely at the attachment location in Figure 4.11 has a KSD of 1.0 – it is a perfect predictor of this object. In Fold 1, a coarse-scale texel feature was learned that responds to the unique concavity created by the protrusion. This feature (not shown) also has a KSD of 1.0.

The consistently most difficult objects to recognize were cyl6 and tub6. Most severely, none of these was recognized correctly in Fold 1. It is no coincidence that cyl6 is the most featureless object in the dataset, and it is identical to the tub6 except that the latter is hollow. This is a

[2] These objects were generated by Levente Tóth at the Centre For Intelligent Systems, University of Plymouth, UK, and are available under the URL http://www.cis.plym.ac.uk/cis/levi/UoP_CIS_3D_Archive/8obj_set.tar.

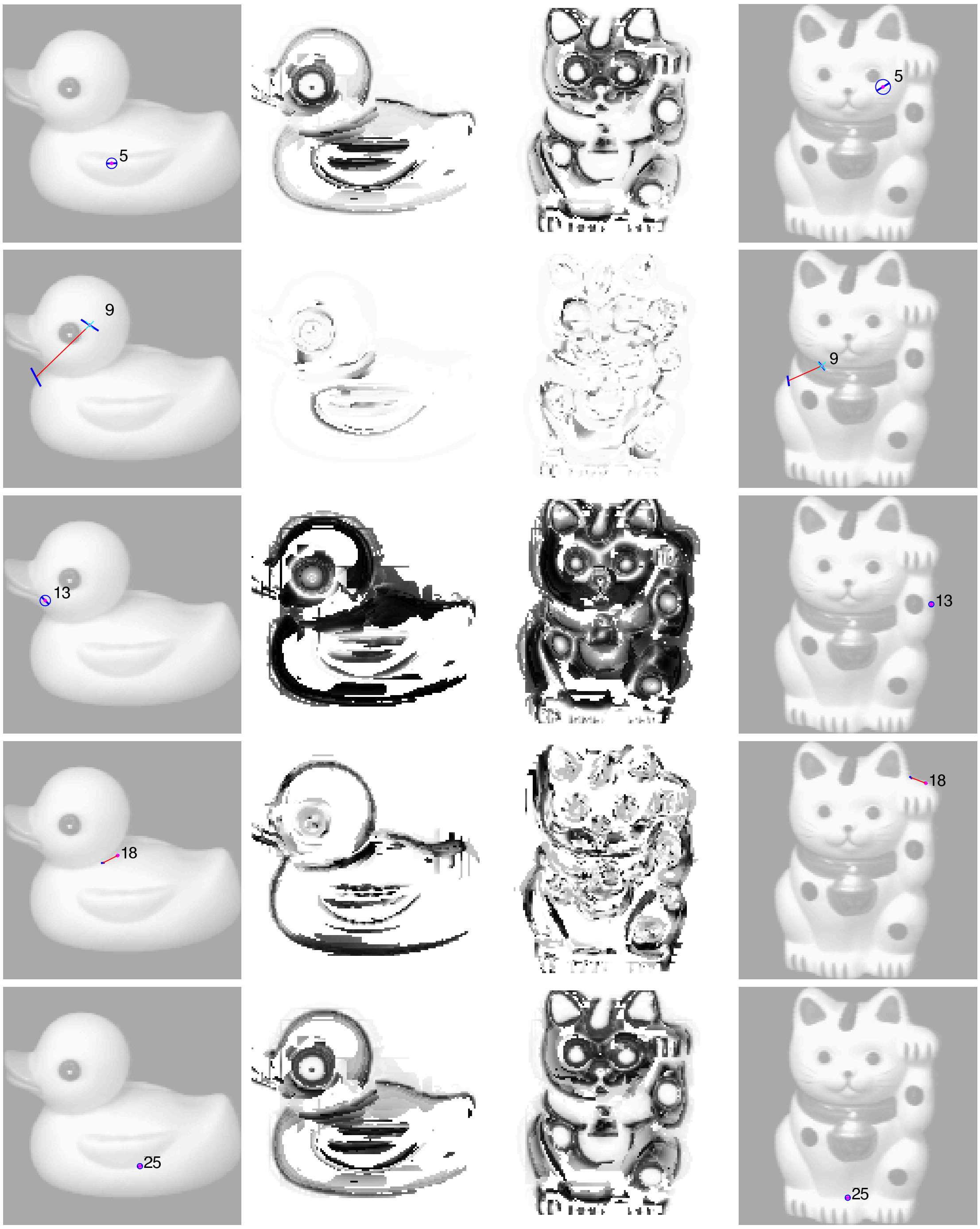


Figure 4.9. Spatial distribution of feature responses of selected features characteristic of obj1 (left columns), for comparison also shown for obj4 (right columns). In the center columns, the gray level encodes the response strength of the feature shown in the outer panels. Black represents the maximal response of $f = 1.0$; in white areas no response was computed. Features are labeled by their ID numbers.

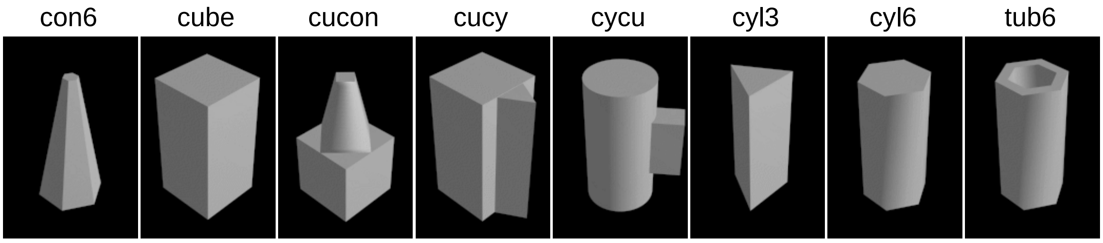


Figure 4.10. Objects used in the Plym task. Shown are middle views from the sorted sequence of viewpoints.

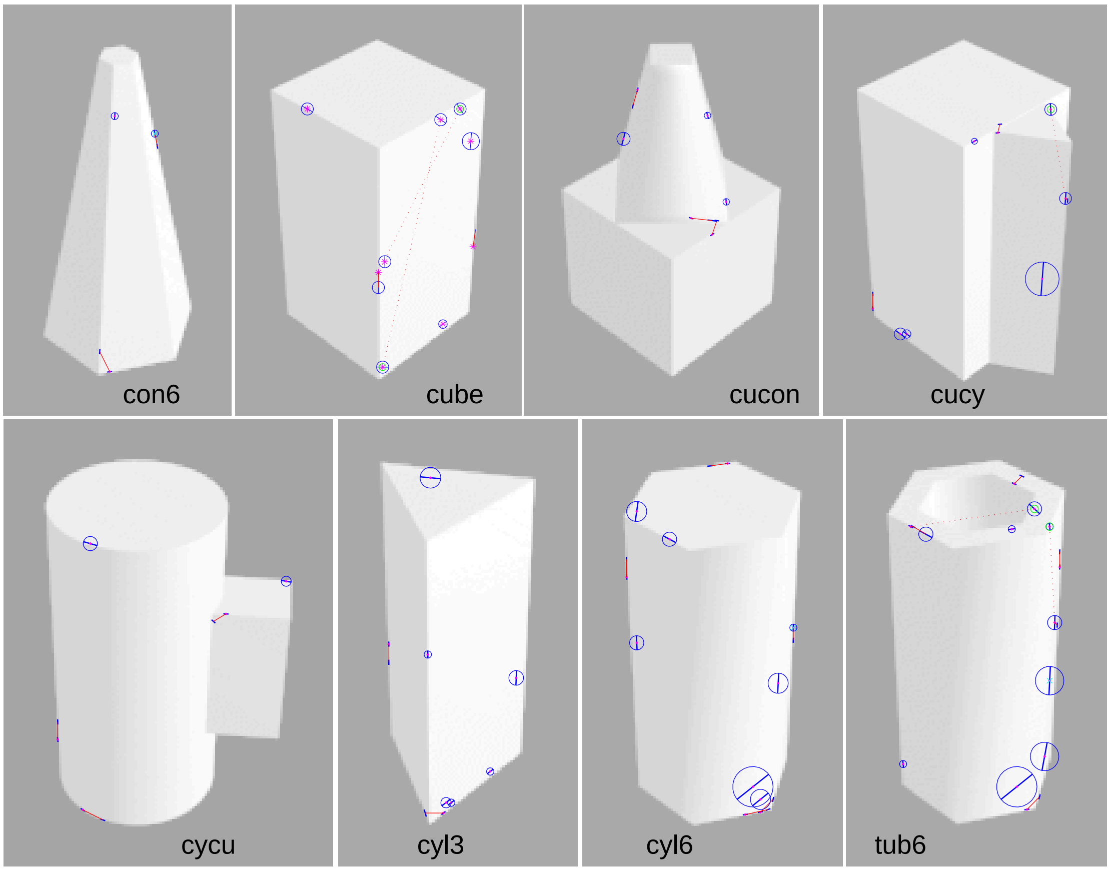


Figure 4.11. Examples of features learned for the Plym task. For each class, all features characterizing this class in Fold 2 are shown that are considered to be present in the image by the classifier.

Table 4.5. Results on the Plym task. The table shows, for each class, how many instances were recognized as which class for correct and wrong recognitions, and the number of ambiguous/confused/ignorant recognition outcomes. In each row, these entries sum to the number of test images of each class (here, 8 in Fold 1 and 7 in Fold 2). The rightmost column shows the number of feature nodes that are part of the Bayesian network representing each class. The bottom row of the bottom panel gives the average proportion of the test set that falls into each recognition category.

	Confusion Matrix correct/wrong											# Feats.
Class	con6	cube	cucon	cucy	cycu	cyl3	cyl6	tub6	amb.	cnf.	ign.	BN
Fold 1:												
con6	6										2	13
cube		3									5	16
cucon	1		5							1	1	5
cucy				3					3		2	10
cycu					7						1	8
cyl3						8						4
cyl6											8	15
tub6						2	1				5	13
Sum	7	3	5	3	7	10	1	0	3	1	24	84
Fold 2:												
con6	4										3	6
cube		5									2	12
cucon			1						5		1	6
cucy				3							4	7
cycu					7							6
cyl3						3					4	14
cyl6							3				4	21
tub6							1	3			3	14
Sum	4	5	1	3	7	3	4	3	5	0	21	86
Average of both folds (proportions):												
con6	.67										.33	10
cube		.53									.47	14
cucon	.07		.40						.33	.07	.13	6
cucy				.40					.20		.40	9
cycu					.93						.07	7
cyl3						.73					.27	9
cyl6							.20				.80	18
tub6						.13	.13	.20			.53	14
Avg.	.09	.07	.05	.05	.12	.11	.04	.03	.07	.01	.38	11

situation where the lack of a closed-world assumption (cf. Section 4.4.1) impairs recognition performance: The only way to decide that an object is a cyl6 as opposed to a tub6 is by asserting the *absence* of features corresponding to the cavity. This type of inference is not performed by the recognition system. Therefore, the learning system tries hard to identify features characteristic of cyl6, finding features that respond to the individual training images, but do not generalize. These features fail to respond to unseen test views, which results in ignorant recognitions. The similarity between these two objects is further illustrated by the fact that in both folds, their respective Bayesian networks share three features.

No pattern is apparent in the few ambiguous recognitions (Table 4.6). In each fold, only one class experienced ambiguous recognitions. Neither of these ambiguities is reflected in shared features.

Table 4.6. Results on the Plym task: Confusion matrices of ambiguous recognitions. The italic numbers on the diagonal indicate how many times the class given on the left was recognized ambiguously (cf. Table 4.5), and the upright numbers in each row specify the false-positive classes.

Fold 1:

Class	con6	cube	cucon	cucy	cycu	cyl3	cyl6	tub6
con6								
cube								
cucon								
cucy				*3*	2	1		
cycu								
cyl3								
cyl6								
tub6								

Fold 2:

Class	con6	cube	cucon	cucy	cycu	cyl3	cyl6	tub6
con6								
cube								
cucon		1	*5*			4		
cucy								
cycu								
cyl3								
cyl6								
tub6								

4.5.3 The Mel Task

The Mel task used all 12 images of each of six non-rigid objects from the image database used to train and test Mel's SEEMORE system [71]. All images are shown in Figure 4.12. To speed up processing, the images were subsampled to 320 × 240 from their original size of 640 × 480 pixels. The views represent random samples from the objects' configuration space, taken at two different scales that differed by a factor of two. Images were randomly assigned to training and test sets, each containing six images. This data set was intended to test the limits of the learning system. As the feature space derives most of its expressive power from geometric compound features, it is best suited for rigid objects. The Mel task, however, consists of non-rigid objects in widely varying configurations, limiting useful geometric compounds to very small scales.

Table 4.2 shows that the system did poorly on the Mel database, as reflected by all performance categories except for confused recognitions. Ignorant recognitions even exceeded correct

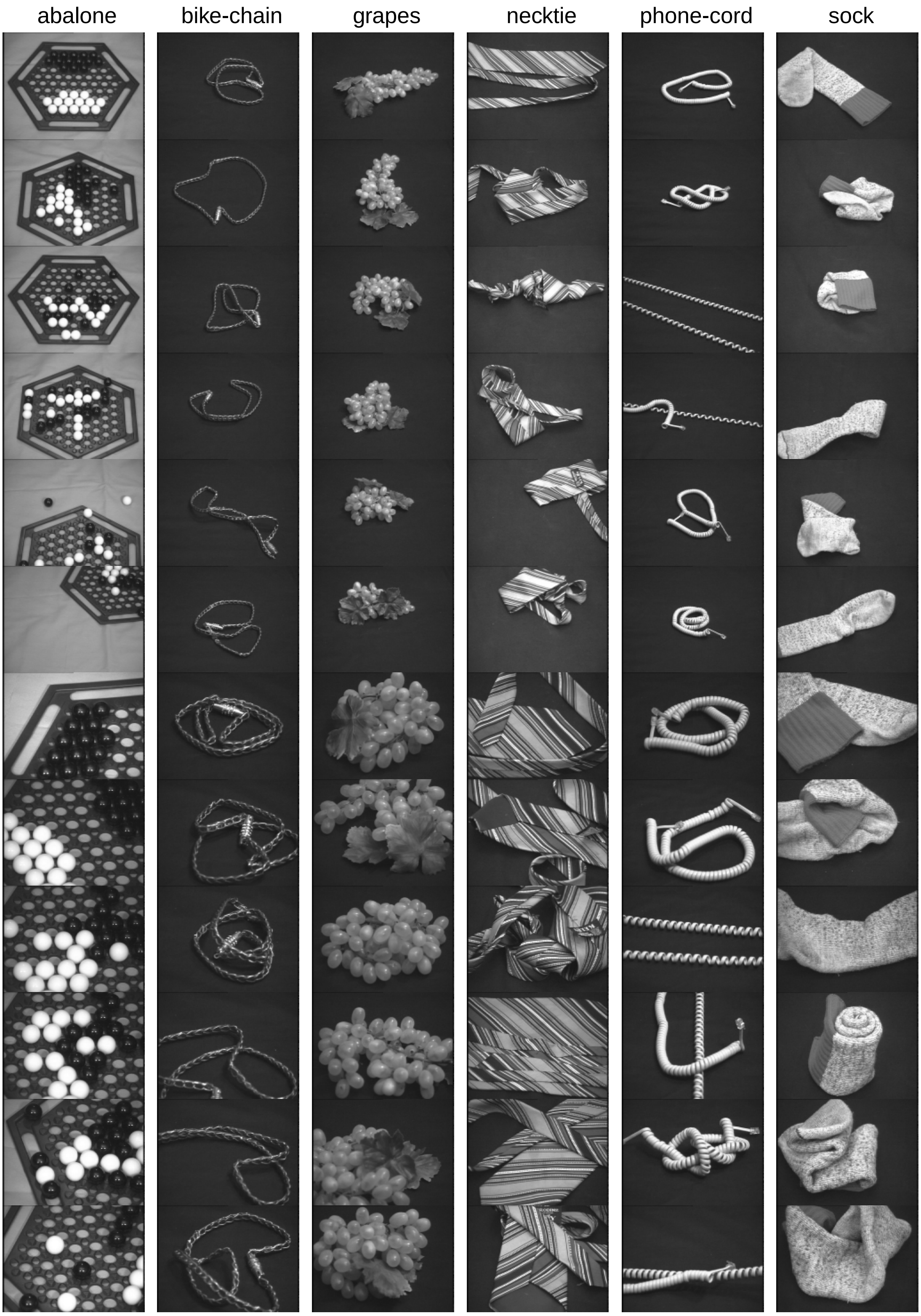


Figure 4.12. All images used in the Mel task.

recognitions. According to Table 4.7, the four weakest-performing classes were phone-cord, grapes, abalone, and sock. In particular, no phone cords were recognized correctly. One problem was that for the smaller-scale views, much of the crucial detail was represented at scales that approached the pixel size. Each of the other three classes reveals a limitation of the system as presented here. A sock is primarily characterized by the gray-level statistics of its structureless surface. However, the features described in Chapter 3 are designed to characterize gray-level structure, not statistics.

Table 4.7. Results on the Mel task. The table shows, for each class, how many instances were recognized as which class for correct and wrong recognitions, and the number of ambiguous/confused/ignorant recognition outcomes. In each row, these entries sum to the number of test images of each class (here, 6). The rightmost column shows the number of feature nodes that are part of the Bayesian network representing each class. The bottom row of the bottom panel gives the average proportion of the test set that falls into each recognition category.

	Confusion Matrix correct/wrong									# Feats.
Class	abalone	chain	grapes	necktie	cord	sock	amb.	cnf.	ign.	BN
Fold 1:										
abalone	2						2		2	10
bike-chain		4							2	8
grapes			2		1				3	7
necktie		1		3					2	7
phone-cord	1						1		4	9
sock				2		2	1		1	9
Sum	3	5	2	5	1	2	4	0	14	50
Fold 2:										
abalone	2						1		3	6
bike-chain		4							2	6
grapes	1		1			1			3	7
necktie				3			3			8
phone-cord		1							5	12
sock						2	1		3	8
Sum	3	5	1	3	0	3	5	0	16	47
Average of both folds (proportions):										
abalone	.33						.25		.42	8
bike-chain		.67							.33	7
grapes	.08		.25		.08	.08			.50	7
necktie		.08		.50			.25		.17	8
phone-cord	.08	.08					.08		.75	11
sock				.17		.33	.17		.33	9
Avg.	.08	.14	.04	.11	.01	.07	.13	.00	.42	8

The abalone board contains strong gray-level structure, but this structure varies widely between images (Figure 4.12). The most meaningful texel features occur about at the scale of the abalone balls. These do not have a meaningful orientation, as they will mostly be measured at scales that are affected by changes in the board configuration. This prevents the construction of powerful geometric compound features, as these rely on predictable relative orientations between features.

For practical, reasons, the feature sampling procedure relies on salient points and their intrinsic scales as extracted by scale-space methods. The grapes class exposes the critical impact of this procedure on recognition performance. Due to the low contrast and the high degree of self-occlusion and clutter present in the grapes images, the extracted texels and their scales rarely corresponded to individual grapes. Moreover, similarly to the abalone board, the orientations of individual texels is rarely meaningful for larger-scale texels in the grape images.

Table 4.8 of ambiguous recognitions does not reveal any conclusive insights. Figure 4.13 shows examples of features learned for the Mel task. Curiously, two background features were found characteristic of the abalone class. From the viewpoint of task-driven learning, this is not considered bad. Any feature that is helpful on a task should be considered. If the background is predictive of a target class, then background features may contribute to performance.

Table 4.8. Results on the Mel task: Confusion matrices of ambiguous recognitions. The italic numbers on the diagonal indicate how many times the class given on the left was recognized ambiguously (cf. Table 4.7), and the upright numbers in each row specify the false-positive classes.

Class	abalone	chain	grapes	necktie	cord	sock
Fold 1:						
abalone	*2*				2	1
bike-chain						
grapes						
necktie						
phone-cord	1				*1*	
sock	1					*1*
Fold 2:						
abalone	*1*				1	
bike-chain						
grapes						
necktie		1	1	*3*	1	
phone-cord						
sock				1	1	*1*

4.6 Discussion

This chapter presented an incremental and task-driven method for learning distinctive features from the feature space introduced in the previous chapter. Although the experiments described above used conventional data sets used in object recognition research, the method developed here is far more general than most existing recognition systems. In contrast to most methods based on eigen-subspace decomposition, this approach does not rely on global appearance or background segmentation (but see Colin de Verdière and Crowley [26] for an example of locally applied eigen-features, and Black and Jepson [14] and Leonardis and Bischof [61] for robust versions of global eigen-features). In contrast to most approaches based on local features, this system does not require a hand-built feature set, but learns distinctive features from a feature space. No prior knowledge about the total number of target classes is used, and no assumptions are made regarding the number of target classes present in any given image. The learner learns distinctions as required, adding classes as they arise.

The feature learning algorithm relies on two key concepts for incremental learning of conditional probabilities: the example images, and the instance list. The concept of example images is analogous to the human capacity to keep a small number of example views in short-term memory. A human acquires these views simply by looking at an object for an extended period

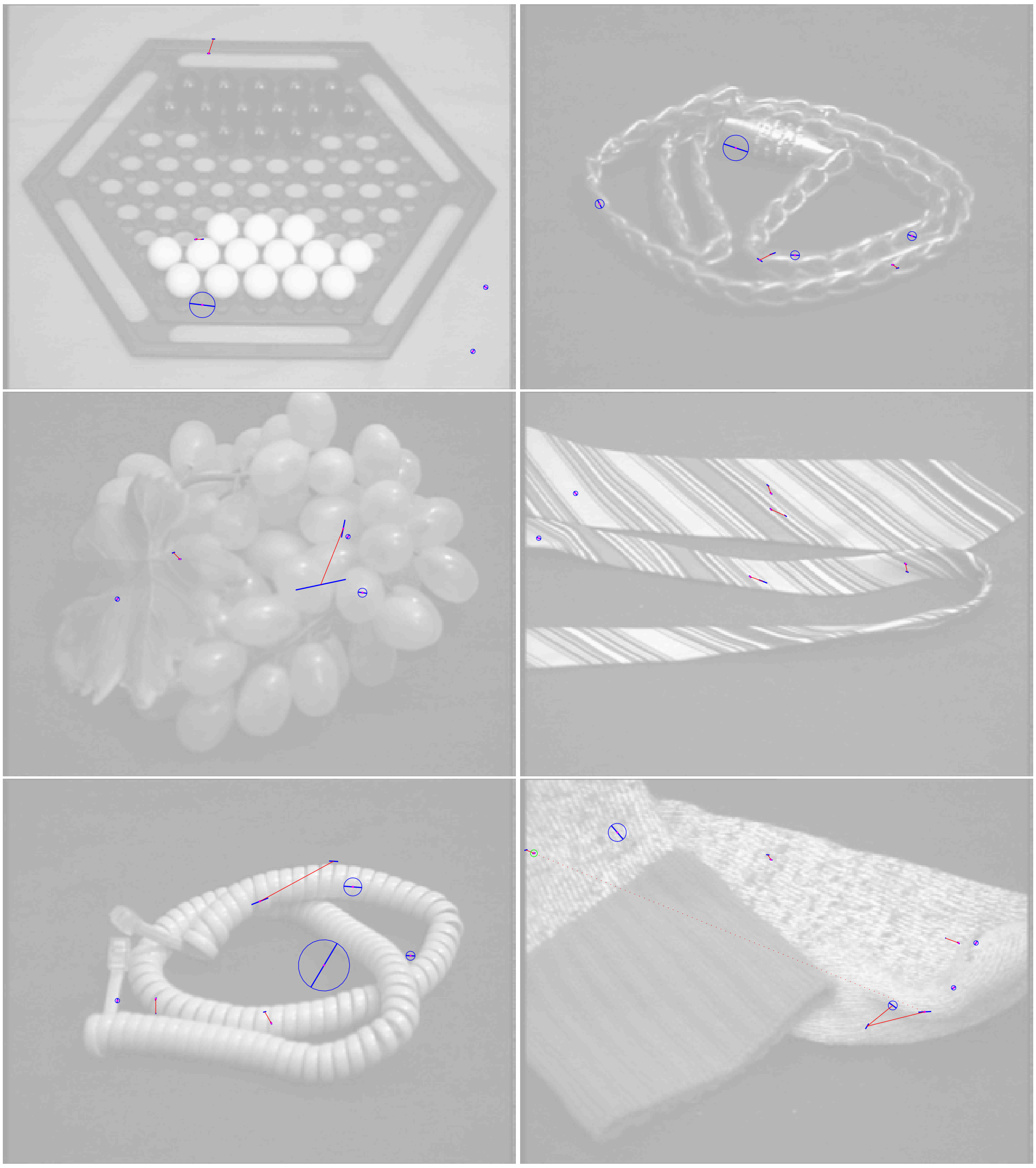


Figure 4.13. Examples of features learned for the Mel task. For each class, all features characterizing this class in Fold 1 are shown that are considered to be present in the image by the classifier.

of time while changing the viewpoint. Importantly, these acquired views are not distinct, but form a coherent motion sequence, which allows the observer to track features across viewpoints. This can greatly facilitate the discovery of stable and distinctive features [129]. In the system discussed in the present chapter, no such assumptions are made about the example images. Future work should investigate how to take advantage of passively provided or actively acquired motion sequences.

Notably, example images can be avoided altogether. In this case, evaluation of newly sampled features is deferred until a sufficient number of training views of the object classes in question has been acquired over time. However, this deferred evaluation precludes the goal-directed, simple-to-complex feature search implemented by Algorithms 4.8 and 4.10. Instead, a number of features must be generated and kept around until they can be evaluated. To illustrate how this can be implemented, the second application discussed in Chapter 6 operates without example images.

The second key component, the instance list, is not easily dispensable because it is used to estimate class-conditional feature probabilities. The dynamic recomputation of the thresholds $\alpha_{\mathbf{f}}$ (cf. Definition 4.2 (Distinctive Power of a Feature) on page 37, and Algorithm 4.8 (Feature Learning), Step 7 on page 41) relies on the ability to count the individual instances. However, this dynamic recomputation is not an essential part of the overall algorithm. If one is willing to commit to an early estimate of the thresholds $\alpha_{\mathbf{f}}$, then the conditional probability distributions can be maintained in the form of parametric approximations that can be updated incrementally as new training examples become available. In this case, the instance list can be completely avoided. If an instance list is used, it may be desirable to truncate it at a given length. This has two important consequences: First, the memory requirements of the feature learning mechanism are essentially constant for a fixed number of classes. Second, the learner would continually adapt to a nonstationary environment, learning new features as required, and forgetting old features as they become obsolete.

The recognition system is designed to be very general. Three important design choices that were motivated by this goal are the use of local features that are more robust to the presence of clutter and occlusions than global features, the use of scale-space theory to infer the appropriate local scales for image analysis, and the use of separate classifiers for each class, each of which decides whether an object of its class is present in the scene, independently of the other classes. These characteristics result in a very general system that makes few assumptions about the viewed scenes. On the other hand, if such prior information is available in the context of a given task, any general system is clearly put at a disadvantage with respect to more specialized systems that explicitly exploit this information. For example, if the scale of the target objects is known, scale-space theory (or any other method for multiscale processing) introduces unnecessary ambiguity. If it is known that there is exactly one target object present in each scene, it would be preferable to have the classes compete in the sense that evidence found in favor of one class reduces the belief in the presence of other classes. This is implemented by the classical model of a Bayesian classifier, where a single class variable has one state for each class. If objects are rigid and presented in isolation on controlled backgrounds, as was the case in the COIL and Plym tasks, then superior results are typically achieved using global features.

To avoid unnecessary complication, all experiments were performed using controlled imagery with one well-contrasted object present in each scene. Moreover, with the exception of the Mel task, objects were rigid and subject to only minor changes in scale. As is to be expected, somewhat superior results were achieved by earlier versions of the feature learning system that took advantage of known task constraints. Instead of the multiple Bayesian network classifiers used in this chapter, a single Bayesian network classifier [85] and a decision tree [84, 86] assigned exactly one class label to each image. Other systems processed imagery at a single scale [84] or at multiple scales, spaced by a factor of two [86, 88, 87, 89].

The principal reason for the relatively low recognition accuracies achieved in this chapter is that the learning algorithm accepts any feature that solves a given recognition problem (Algorithm 4.8 (Feature Learning), Step 8b, page 42). However, such features may have very little distinctive power. Therefore, the generalization properties of the resulting classifiers are subject to the randomness of the feature sampling procedure. The following Chapter 5 will present a method for learning improved features, which leads to improved recognition results while preserving the openness of the system. Therefore, a more detailed discussion of the performance characteristics of the feature learning system is deferred to Chapter 5.

Another important limitation of the system as described in this chapter is related to the computation of the salient points. Due to limited computational resources, computation of feature values is restricted to salient points. The fewer salient points are extracted, the more the learning success will depend on the variety and stability of the salient points, which is the flip side of the computational savings. Moreover, in the experiments reported here, the scale space was relatively coarsely sampled at half-octave spacing. If more intermediate scales are used, more local scale-space maxima and thus more salient points will generally be found, and intrinsic scales will be measured more accurately, again at increased computational expense. Section 5.4.5 will examine the computational cost in more detail.

The goal of this work is not to achieve high recognition rates by exploiting known prior constraints, but to begin with a highly uncommitted system that learns useful visual skills with experience. To fully evaluate the the general system, large-scale experiments would be required to test it under various degrees of clutter and occlusion at various scales, with various numbers of target objects present in a scene. There is no known suitable image data set currently available. Hence, a full-scale performance evaluation would constitute a substantial endeavor, which is beyond the scope of this dissertation.

CHAPTER 5

EXPERT LEARNING

Chapter 4 presented an algorithm for learning distinctive features from the feature space described in Chapter 3. This brief chapter introduces a simple extension to the learning algorithm that produces markedly improved features. A resulting significant improvement in recognition performance is reflected by all evaluation criteria.

5.1 Related Work

Most current machine vision systems are constructed or trained off-line or in a dedicated training phase. In contrast, human visual performance improves with practice. For example, recognition accuracy and speed both increase with growing experience. It is not clear what the biological mechanisms are for this type of visual learning. Nevertheless, there is substantial evidence that at least part of the performance improvement can be attributed to better features. Tanaka and Taylor [120] found that bird experts were as fast to recognize objects at a subordinate level (“robin”) as they were at the basic level (“bird”). In contrast, non-experts are consistently faster at basic-level discriminations. This effect is so pronounced that it has been proposed as a definition of expertise in visual recognition [120, 38]. It seems that experts, colloquially speaking, know what to look for, suggesting they have developed highly discriminative features.

Gauthier and Tarr [38] investigated the phenomenon of recognition expertise using artificial, unfamiliar stimuli, the so-called *Greebles.* Greebles are organized into genders and families, that are characterized by the shapes of an individual’s body parts. Subjects were trained to recognize Greebles at three levels – gender, family, and individual – until they reached the expert criterion, i.e., they were as fast to recognize Greebles at the individual level as at the family or gender level. Training required between 2700 and 5400 trials, spread across a total of 7 to 10 one-hour sessions. Experts thus trained were slower to recognize Greebles with slightly transformed body parts, as compared to the Greebles they saw during training. All expert subjects reported noticing these transformations. In contrast, for non-experts there was no such speed or accuracy difference, and none of them noticed the transformations. Again, this result strongly suggests that experts had developed highly specific features. In fact, Gauthier and Tarr write:

> “In our experiment, expertise training may have led to the assembly of complex feature-detectors, extracted from the statistical properties of the Greeble set that proved useful for performing the training discriminations.” [38]

The feature space introduced in Chapter 3 fits this description. The present chapter provides a computational model that is consistent with the empirical results and proposed mechanisms reported by these authors. Regarding a possible neural substrate, Gauthier and Tarr speculate that the complex feature detector neurons found in the inferotemporal cortex are not fixed but can be modified in response to experience, citing Logothetis and Pauls [65] who found that these neurons can become highly selective for previously novel stimuli.

I am not aware of any work in machine vision that is related to the development of visual expertise and the formation of improved features. Feature extraction methods based on discriminative eigen-subspaces generate highly distinctive features, but these methods are usually applied off-line, and the extracted features are typically non-local [31, 37, 117].

5.2 Objective

The basic feature learning procedure introduced in the preceding chapter is driven by a single criterion, that is, to learn the training images. In the absence of any misrecognitions, no learning occurs. In analogy to human visual learning, the resulting visual system remains a life-long non-expert. The objective in this chapter is to extend the basic learning procedure such that learning can continue even in the absence of misrecognitions. As motivated by the phenomenon of human expertise, learning then focuses on improving the features.

How can the learner improve its features? What is the criterion that measures the quality of a feature? In the previous chapter, no attention was paid to the quality of a feature. The only requirement of a candidate feature for inclusion in a classifier was that it solved the recognition problem that triggered the present feature search (Algorithm 4.8 (Feature Learning), Step 8b, page 42). The algorithm was *biased* toward finding few and powerful features (by adding a maximum of one feature for each failed recognition), and toward structurally simple features (by virtue of the simple-to-complex ordering of steps in the feature-generation algorithm 4.10 on page 43). However, there was no *explicit* pressure on the number or quality of features. The following section defines an obvious measure for the quality of a feature, and a simple mechanism for learning improved features. A desired side effect is the reduction of the number of features used during recognition, which resembles and might explain the fast recognition performed by human experts.

5.3 Expert Learning Algorithm

The goal of the learning system is to learn *distinctive* features (Definition 4.1, page 31). To discretize the continuous feature values, a cutpoint is chosen that maximizes the distinctive power of a feature (Definition 4.2, page 37). Hence, a natural way to define the quality of a feature for the purposes of expert learning is to equate it with its distinctive power between the characterized class and one or more discriminated classes.

The basic idea of the expert learning algorithm is to reduce uncertainty in the recognition result by learning features that discriminate well between classes that are difficult to distinguish by the recognition system. This difficulty is measured in terms of the KSD achieved by any feature available to discriminate between these classes. If for a given pair of classes this KSD falls below a global threshold t_{KSD}, then a new feature is sought that discriminates between these classes with a KSD of at least t_{KSD}. A high-level view of the expert learning idea is given by the following algorithm.

Algorithm 5.1 (Expert Learning) On receipt of an image, the following steps are performed:

1. The image is recognized using Algorithm 4.4 (page 38). If the result is correct (in the sense of Definition 4.5), then continue at Step 2. Otherwise, the feature learning algorithm (4.8, page 41) is run, with one slight but important modification: At Step 3, Algorithm 4.10 is called repeatedly until it returns a candidate feature with a KSD greater than t_{KSD}. The completion of Algorithm 4.8 also concludes this Algorithm 5.1.
2. If there is no class with nonzero residual entropy after recognition, the algorithm stops. No expert learning is performed in this case – the learner is considered a fully trained expert on this image.
3. Identify a pair (c, d) of classes for which an improved distinctive feature is to be learned. The details are given in Algorithm 5.2 below.
4. A new feature is learned that is characteristic of class c and discriminative with respect to class d, using Algorithm 4.8, beginning at Step 2, subject to the same minimum-KSD requirement as described at Step 1 above.

■

A crucial part of this straightforward algorithm is Step 3. There are many ways to define a meaningful pair of classes that warrant learning an improved distinction. The method described in Algorithm 5.2 below identifies a pair of classes that constitutes a “near miss” in the recognition. Hence, one of the classes is a true class (i.e., a target class present in the current image), and the other is a false class. The specific pair of classes is chosen that are least well discriminated by any feature consulted during the recognition procedure (Step 1 in Algorithm 5.1). Therefore, this is the pair of classes that would benefit the most from a better feature. Crucially, the characterized class of the new feature must have had nonzero residual entropy after recognition. This guarantees that the new feature is going to be queried next time the same training image is seen. Thus, the new feature actually improves the situation, and endless loops are avoided.

Algorithm 5.2 (Choosing a Pair of Classes for an Expert Feature) This algorithm is run at Step 3 of Algorithm 5.1, and determines a characteristic class c and a discriminated class d that are least well discriminated by existing features:

1. For all classes c_i with nonzero residual entropy:
   (a) If c_i is a true class, then define D as the set of all false classes; otherwise, let D be the set of all true classes.
   (b) Determine the class $d_i^{\min} \in D$ that is least well discriminated by any feature characteristic of class c_i, in terms of the $\mathrm{KSD}_{c_i,d_i^{\min}}$. This is accomplished by computing for all classes $d_j \in D$:
      - Over all features $\mathbf{f}$ that are characteristic of class c_i, and that have been consulted during the recognition procedure, compute the maximum $\mathrm{KSD}_{c_i,d_j}(\mathbf{f})$. Each $\mathrm{KSD}_{c_i,d_j}(\mathbf{f})$ is computed by counting applicable instances in the instance list.

      Then, the least-well discriminated class $d_i^{\min}$ is given by the feature with the weakest value of the $\mathrm{KSD}_{c_i,d_j}(\mathbf{f})$:

      $$\mathrm{KSD}_{c_i,d_i^{\min}} = \min_{d_j \in D} \max_{\mathbf{f}} \mathrm{KSD}_{c_i,d_j}(\mathbf{f})$$

2. The characterized and discriminated classes are given by the pair with the weakest associated KSD. Formally, $c = c_i$ and $d = d_i$, where

   $$i = \operatorname*{argmin}_j \mathrm{KSD}_{c_j,d_j^{\min}}. \tag{5.1}$$

■

The feature learning procedure defined by Algorithms 5.1 and 5.2 attempts to learn improved features as long as there are classes with nonzero residual entropy, and as there are classes that are discriminated at a KSD less than t_{KSD} by any currently available feature. The threshold t_{KSD} can be increased over time. There are other possible ways to learn improved features. For example, one can attempt to learn highly distinctive features between all pairs of classes exhaustively. However, this is impractical for even moderate numbers of classes, and it would likely generate a large number of features that are never going to be used in practice. A nice property of the above algorithms is that expert feature learning – like the basic feature learning – is driven by actual experience. Successfully learned features are guaranteed to be used should the same situation arise again. Thus, each new feature has a well-defined utility to the recognition system. When there is no measurable need for better features, the expert learning algorithm stops.

5.4 Experiments

Experiments were performed using the same simulated, incremental paradigm as in Section 4.5, and using the same data sets. Here, training occurred in *stages*. The purpose of these stages was to progressively increase the threshold t_{KSD}, below which a feature was considered weak and triggered expert learning.

Algorithm 5.3 (Training Stages)

1. Initially, set the stage index to $s = 0$, and let $t_{\text{KSD}}(s) = 0$.
2. Train the system by cycling through the training set, as described in Section 4.5. Cycle until either the training set has been learned completely (i.e., all recognitions are correct), or give up after 20 iterations. During feature learning, apply the $t_{\text{KSD}}(s)$ threshold as described in the expert learning algorithm 5.1, Step 1.

   At the end of each cycle, discard all features that have not been consulted at all during this past cycle.

   During recognition, monitor the minimum KSD encountered by Algorithm 5.2 during the final iteration through the training set. This value is given by

   $$\text{KSD}_{\min}(s) = \min_{\text{recognitions}} \min_i \text{KSD}_{c_i, d_i^{\min}}$$

   where the first minimum is taken over all executions of Algorithm 5.1 during the final iteration through the training set (cf. Equation 5.1).
3. For the next stage $s + 1$, set

   $$t_{\text{KSD}}(s+1) = 1 - \left(1 - \frac{\min\{\text{KSD}_{\min}(s),\ t_{\text{KSD}}(s)\}}{2}\right)$$

   This rule attempts to ramp up the target KSD t_{KSD} exponentially, asymptoting at unity, but only if the target KSD was reached during the stage just completed, i.e. if $\text{KSD}_{\min}(s) \geq t_{\text{KSD}}(s)$. If $t_{\text{KSD}}(s+1) > 0.98$, then it was set to $t_{\text{KSD}}(s+1) = 1$.
4. Increment s, and continue at Step 2.

■

According to this algorithm, the first stage ($s = 0$) is precisely the basic feature learning procedure used in Section 4.5. In fact, the results reported there are those obtained after the first stage of a comprehensive training procedure, the results of which are presented below.

5.4.1 The COIL Task

The first block of results listed in Table 5.1 corresponds to the COIL results in Table 4.2 on page 47, but were attained after expert learning. (The '-inc' tasks in Table 5.1 will be discussed in Section 5.4.4.) Most of the reported parameters are visualized for all training stages in Figure 5.1. In these graphs, training stage 0 corresponds to the data in Table 4.2, and the last training stage corresponds to Table 5.1.

The recognition results in Table 5.1, corresponding to the top panels in Figure 5.1, indicate a dramatic performance increase. With cumulative expert training, the proportion of correct recognitions rises by about 50% compared to the initial training stage, ignorant recognitions are reduced by about 70%, and wrong recognitions are almost eliminated. The number of ambiguous recognitions also increases somewhat. This is not a desired effect, but ambiguity is the best recognition outcome short of correctness. The accuracy reaches more than 80%, which begins to be comparable to object recognition results reported for more specialized machine vision systems.

Another dramatic development can be observed in the evolving feature set. The middle panels in Figure 5.1 reveal a typical behavior. Throughout expert learning, the number of features queried during a recognition procedure decreases sharply. During Stage 1, the number of

Table 5.1. Summary of empirical results after expert learning on all tasks. The first four entries give the name of the task, the number of classes, the fold index, and the accuracy according to Definition 4.6. The five entries under “Recognition Results” in each row sum to one, barring roundoff errors. Zero entries are left blank for clarity. The columns under the heading “# Features” give the total number of feature nodes in all Bayesian networks (BN), the total number of different features (dif.), the average number of features queried (qu’d) during a single recognition procedure (Algorithm 4.4), and the total cumulative number of features sampled (smp’d), respectively. The rightmost two columns list the number of cycles through the training set, and the total number of training image presentations.

				Recognition Results					# Features				Tr. Set	
Task	Cls.	Fld.	acc.	cor.	wrg.	amb.	cnf.	ign.	BN	dif.	qu’d.	smp’d.	cyc.	imgs.
COIL	6	1	.86	.78	.02	.13		.07	10	8	6.4	19448	60	3600
COIL	6	2	.78	.70	.05	.12		.13	23	17	11.1	19203	68	4080
COIL	6	av.	.82	.74	.03	.13		.10	17	13	8.8	19326	64	3840
COIL-inc	6	1	.87	.80		.12		.08	7	6	5.5	13867		6850
COIL-inc	6	2	.88	.82	.02	.10		.07	11	11	6.5	17491		11000
COIL-inc	6	av.	.87	.81	.01	.11		.08	9	9	6.0	15679		8925
COIL-inc	10	1	.82	.77	.01	.08		.14	34	21	12.4	69450		43520
COIL-inc	10	2	.80	.73		.10	.01	.16	28	16	13.0	87996		57060
COIL-inc	10	av.	.81	.75	.01	.09	.01	.15	31	19	12.7	78723		50290
Plym	8	1	.70	.58	.06	.22		.14	23	17	8.9	33711	155	8680
Plym	8	2	.80	.75	.04	.09		.13	11	10	8.5	25513	122	7808
Plym	8	av.	.75	.67	.05	.15		.13	17	14	8.7	29612	139	8244
Plym-inc	8	1	.77	.73	.02	.08		.17	21	11	8.2	71183		36274
Plym-inc	8	2	.83	.79	.02	.05		.14	16	10	7.5	28078		15640
Plym-inc	8	av.	.80	.76	.02	.07		.16	19	11	7.9	49631		25957
Mel	6	1	.61	.47	.03	.19	.03	.28	43	35	15.0	3766	38	1368
Mel	6	2	.55	.42	.14	.22	.03	.19	26	25	8.8	13461	64	2304
Mel	6	av.	.58	.44	.08	.21	.03	.24	35	30	11.9	8614	51	1836
Mel-inc	6	1	.64	.44	.06	.39		.11	25	22	10.4	8313		3774
Mel-inc	6	2	.56	.42	.03	.17		.39	21	19	7.5	15181		3630
Mel-inc	6	av.	.60	.43	.04	.28		.25	23	21	9.0	11747		3702

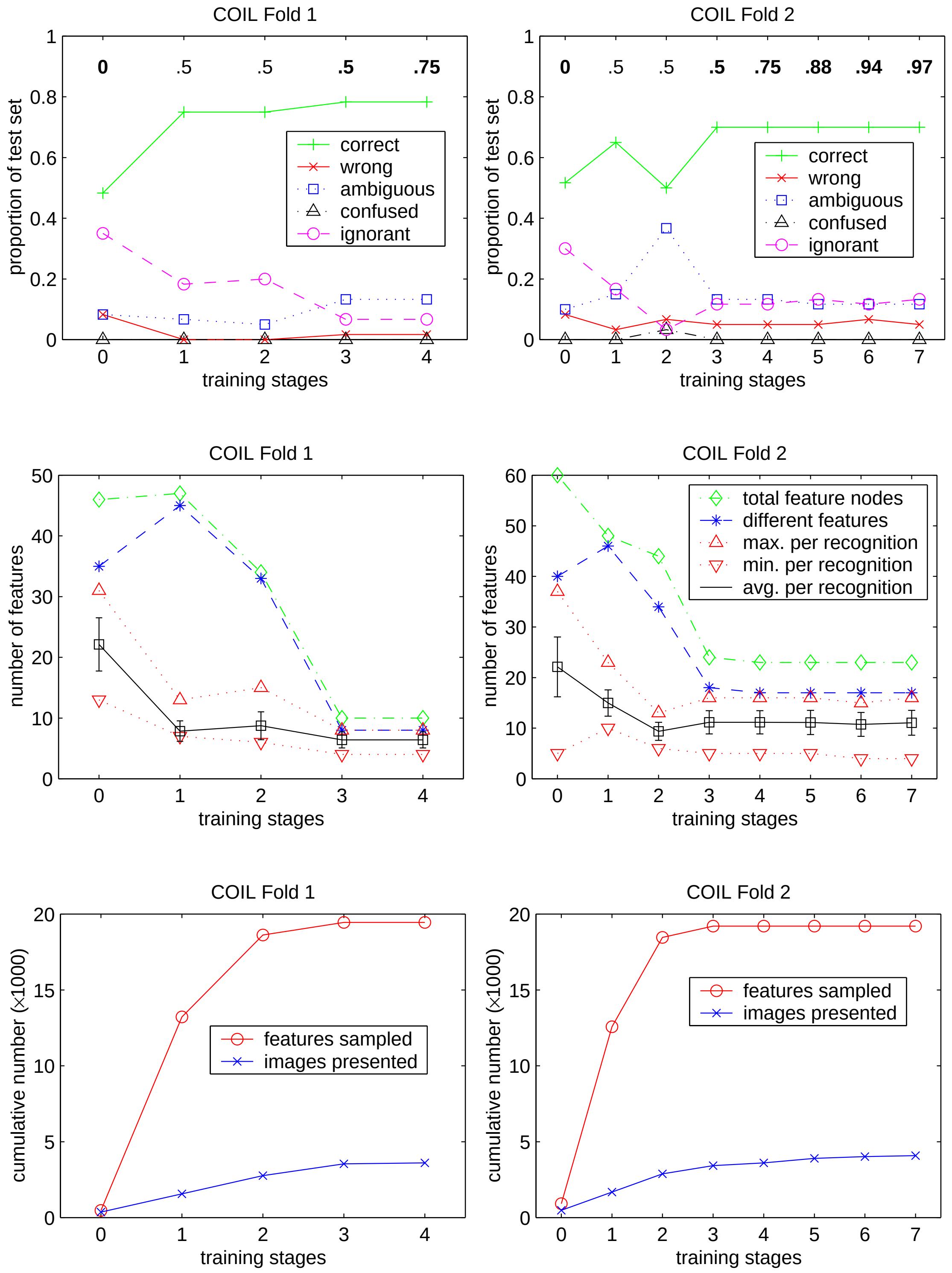


Figure 5.1. Expert Learning results on the COIL task. The numbers shown inside the top graphs give the value of t_{KSD} at the respective training stage. Where this number appears in bold face, the training set was learned with perfect accuracy, and the t_{KSD} goal was achieved. Otherwise, either the training set was not learned perfectly, or the t_{KSD} goal was not attained. The error bars in the middle plots have a width of one standard deviation.

different features present in all Bayesian networks increases, eliminating most of the re-used features. This constitutes a qualitative change in the feature set, during which general features are replaced by more specialized, distinctive features. The total number of features is hardly affected (Fold 1) or is even reduced (Fold 2). During further expert learning stages, a relatively small number of highly distinctive features are learned, rendering a large proportion of the existing features obsolete. Thus, the number of features stored by the system is reduced from 53 (average over both folds) to 17. Due to the high distinctive power of the remaining features, few are queried during a recognition procedure. Fold 1 constitutes a quite drastic example: On average, little more than six features are queried to recognize an image (Table 5.1). This is just one feature per class!

Figure 5.2 shows some of the expert features. Compared to the novice features shown in Figure 4.7 on page 50, these are much more meaningful intuitively. One car is characterized by the texture on the roof, the other by a wheel feature. Both the duck and the cat are characterized by an eye/forehead feature. In fact, this feature is absolutely identical; it is one of the two surviving shared features in Fold 1. Curiously, this is the only feature characteristic of a cat (Table 5.2). How can it alone suffice to characterize a cat if it is also a distinctive duck feature? The answer is that the threshold $\alpha_{\mathbf{f}}$ for this feature to be considered present is much higher in the cat's Bayesian network than in the duck's (cf. Definition 4.2, Distinctive Power of a Feature, page 37). Evidently, on all duck images contained in the training set, the response value $f_{\mathbf{f}}$ of this feature remained below the corresponding $\alpha_{\mathbf{f}}$ threshold of the cat's network, while on all cat images, it exceeded this threshold. Otherwise, this feature alone would not have been sufficient to characterize a cat.

The confusion matrices of ambiguous recognitions is shown in Table 5.3. Some of these ambiguities are not surprising. For instance, in Fold 1 a cat (obj4) was also labeled as a duck (obj1). In both folds, the two cars (obj3 and obj6) tended to be confused. Car/car and duck/cat confusions also accounted for two of the four wrong recognitions shown in Table 5.2.

Finding highly distinctive features is not easy. The training set was first learned after a mere seven iterations (Table 5.1), during which about 700 candidate features were generated. After complete expert training, almost 20,000 features had been tried in more than 60 iterations. As shown in the bottom panel in Figure 5.1, the bulk of this effort is spent at the first stage of expert learning. The reason is that initially, most of the recognitions end with a high degree of uncertainty, and most pairwise uncertainties discovered by Algorithm 5.2 have $\mathrm{KSD}_{c_i,d_i^{\min}} = 0$.

The superior power of expert features over novice features not only results in fewer features queried per recognition, but is also manifested in increased stability across viewpoints and increased specificity. Figure 5.3 shows all features characteristic of obj1. Remarkably, all three features are present in all views except for the very last, where feature 665 is missing. All features responded reliably to corresponding object parts across most of the viewpoint range. Figure 5.4 demonstrates the superior specificity of these expert features compared to the novice features (see Figure 4.9 on page 53 for comparison). The least specific expert feature 557 responds more specifically to distinct parts of the duck than the most specific novice feature 5. Even more significantly, it response on the non-duck image is weaker in most places. Feature 566 acts as an eye detector – it responds to a small dark spot with a neighboring larger light spot. As mentioned earlier, this feature is also characteristic of the cat (obj4). Feature 665 detects the wing in a specific spatial relation to an outer edge of the duck object. It responds selectively to a few locations on the duck image, but almost nowhere on the cat image.

It is quite remarkable that the learning system was able to construct such powerful features: They are specific to an object, and at the same time are general in that they respond well over a wide range of viewpoints. Interestingly, they correspond to intuitively meaningful object parts. It would not have been easy to construct such specific and yet viewpoint-invariant features by hand.

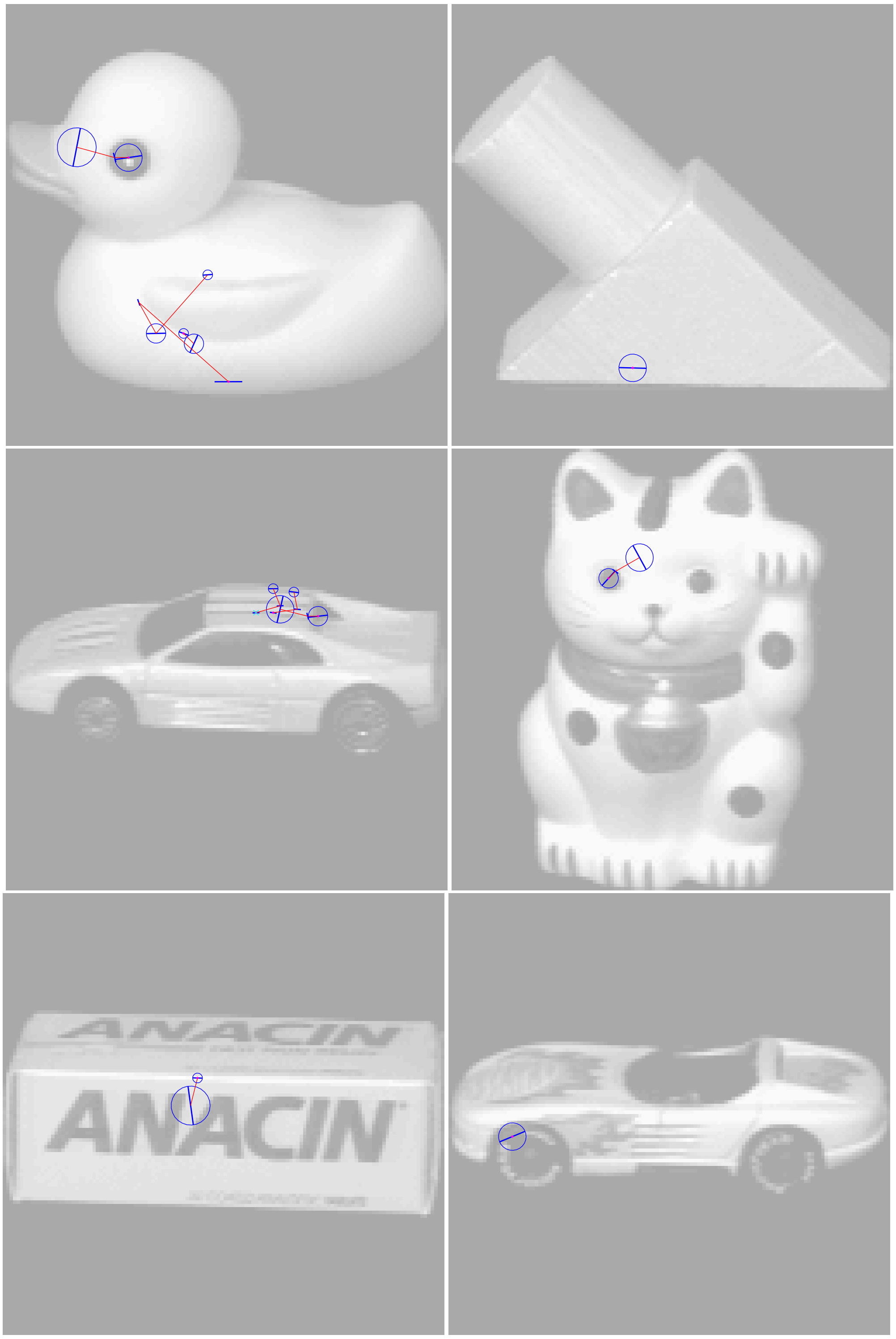


Figure 5.2. Examples of expert features learned for the COIL task, Fold 1.

Table 5.2. Results on the COIL task after Expert Learning. The table shows, for each class, how many instances were recognized as which class for correct and wrong recognitions, and the number of ambiguous/confused/ignorant recognition outcomes. In each row, these entries sum to the number of test images of each class (here, 10). The rightmost column shows the number of feature nodes that are part of the Bayesian network representing each class. The bottom row of the bottom panel gives the average proportion of the test set that falls into each recognition category.

	Confusion Matrix correct/wrong									# Feats.
Class	obj1	obj2	obj3	obj4	obj5	obj6	amb.	cnf.	ign.	BN

Fold 1:

obj1	10									3
obj2		8					2			1
obj3			8			1	1			3
obj4				8			2			1
obj5					6		2		2	1
obj6						7	1		2	1
Sum	10	8	8	8	6	8	8	0	4	10

Fold 2:

obj1	9								1	8
obj2		8					2			2
obj3			5				2		3	4
obj4	1			4			1		4	1
obj5				2	8					1
obj6						8	2			7
Sum	10	8	5	6	8	8	7	0	8	23

Average of both folds (proportions):

obj1	.95								.05	6
obj2		.80					.20			2
obj3			.65			.05	.15		.15	4
obj4	.05			.60			.15		.20	1
obj5				.10	.70		.10		.10	1
obj6						.75	.15		.10	4
Avg.	.17	.13	.11	.12	.12	.13	.13	.00	.10	3

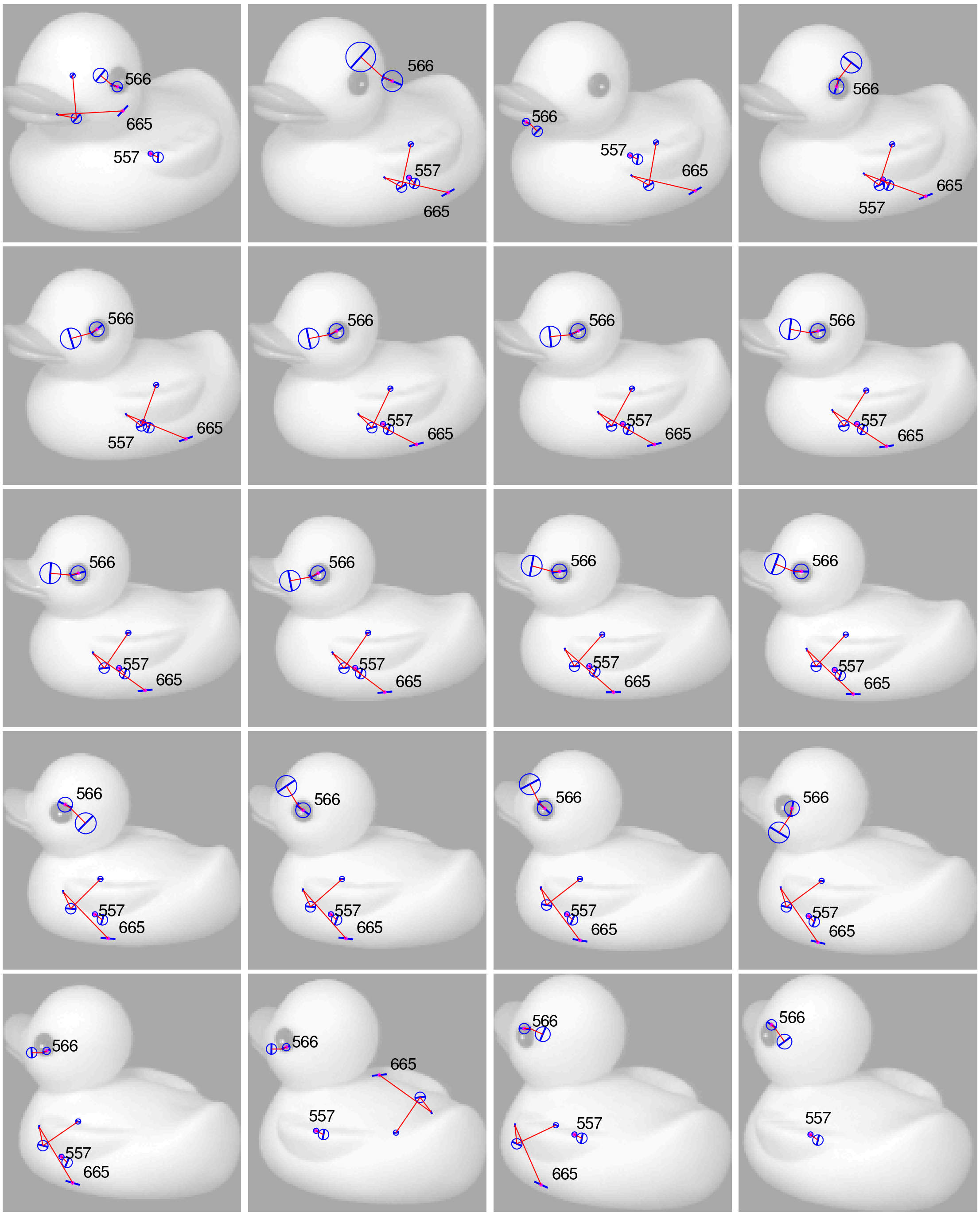


Figure 5.3. Expert Features characteristic of obj1 located in all images of this class where they are present. See Figure 4.8 on page 51 for comparison.

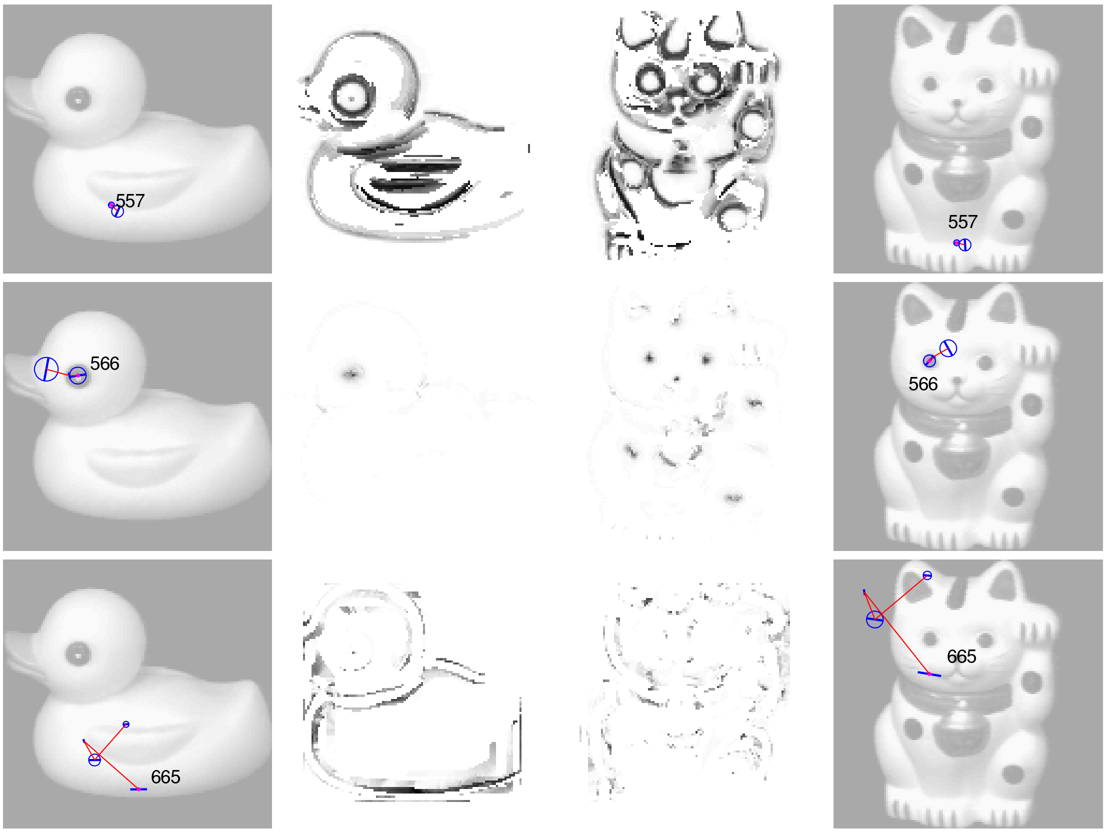


Figure 5.4. Spatial distribution of expert feature responses of all features characteristic of obj1 learned in Fold 1 (left columns), for comparison also shown for obj4 (right columns). In the center columns, the gray level encodes the response strength of the feature shown in the outer panels. Black represents the maximal response of $f = 1.0$; in white areas no response was computed. Features are labeled by their ID numbers. See Figure 4.9 on page 53 for comparison.

Table 5.3. Results on the COIL task after Expert Learning: Confusion matrices of ambiguous recognitions. The italic numbers on the diagonal indicate how many times the class given on the left was recognized ambiguously (cf. Table 5.2), and the upright numbers in each row specify the false-positive classes.

Class	obj1	obj2	obj3	obj4	obj5	obj6
Fold 1:						
obj1						
obj2		*2*				2
obj3			*1*			1
obj4	1		1	*2*		
obj5				2	*2*	
obj6			1			*1*
Fold 2:						
obj1						
obj2	1	*2*			1	
obj3	1	1	*2*			1
obj4				*1*		1
obj5						
obj6			1	1		*2*

The feature learning algorithm employs a random feature generation procedure (Algorithm 4.10, page 43). How sensitive is the performance of the trained system to this randomness? To give an intuition, the system was trained on Fold 1 a total of ten times, seeding the random-number generator with unique values. Figure 5.5 shows the average performance of the trained novices and experts. According to a paired-sample one-tailed *t* test, the superior accuracy of experts vs. novices is highly significant ($p < 0.0001$). There is considerable variation in the accuracies, while the number of features queried stabilizes reliably during expert learning. The reason for this is that learning ends as soon as all recognitions leave all classifiers with zero residual entropy. By this time, all objects are characterized by a small number of features that are highly distinctive on the training set – however, no definitive statement can be made about their generalization properties to unseen test images. In realistic, open learning scenarios, this is not going to be a problem: If an image is misrecognized or is recognized with less than perfect certainty, it immediately becomes a training image, and learning continues. Thus, in practice the final performance is expected to depend much less on the randomness intrinsic to the learner, but is determined by the expressiveness of the feature space.

5.4.2 The Plym Task

All of the general comments made in the previous section about the COIL task apply also to the Plym task (Figure 5.6). As is also seen in Table 5.1 in comparison to Table 4.2, all performance parameters improved substantially as a result of expert learning, with the exception of an increased number of ambiguous recognitions. In Fold 1, progress of expert learning was markedly non-monotonic. For many stages, it failed to achieve the goal of $t_{\mathrm{KSD}} = 0.5$. At the end, however, it learned the training set with a remarkable $t_{\mathrm{KSD}} = 0.94$.

Similarly to the basic learning stage, cyl6 and cucon were problematic classes (Tables 5.4 and 5.5). Examples of the features learned are shown in Figure 5.7. Similarly to the COIL task, these few but highly distinctive features make much more intuitive sense than the features learned during the first stage (see Figure 4.11 on page 54). Notice the con6 feature characteriz-

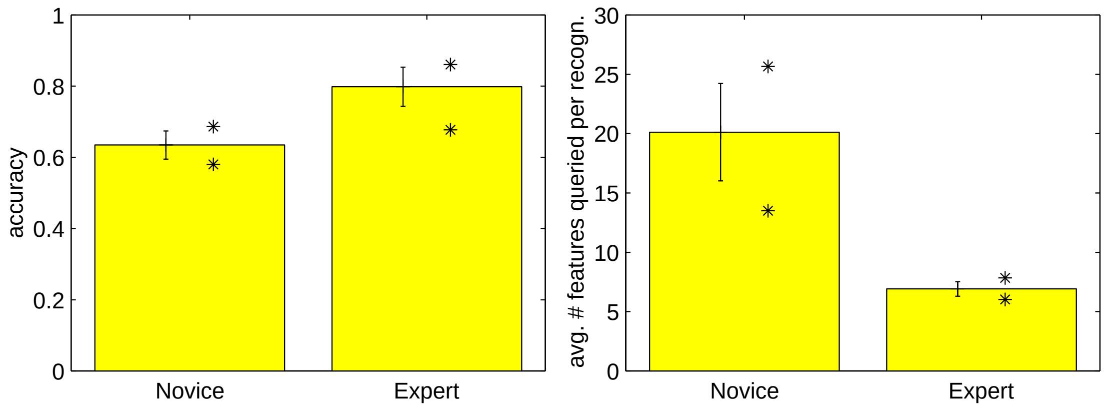


Figure 5.5. Variation due to randomness in the learning procedure. The bars represent the mean performance achieved on 10 independent learning trials of COIL-Fold 1. The error bars extend one standard deviation above and below the mean. Extrema are indicated by asterisks.

ing the converging edges, the cucon, cucy, cycu, and cyl3 features located at characteristic folds or corners, and the tub6 feature that responds to the top rim.

What is the critical limiting factor of the performance on the Plym task, the feature space or the learning procedure? There is no absolute answer to this question. However, if the system can learn to discriminate well between two particularly difficult classes if more training data are used, then this would indicate that the limited performance is primarily an effect of the learning procedure. A leave-one-out cross validation procedure was run on classes cube and cucon. In Fold 1 of the full-scale eight-object task, these were two of the most difficult classes to recognize (Table 5.4), and in two cases a cube appeared as a false positive together with a cucon (Table 5.5). In Fold 2, cucon was one of the weakest classes.

Table 5.6 shows that this distinction was learned quite well, with an accuracy of 0.93. This suggests that the feature space is capable of expressing such distinctions. The critical limiting factor is probably the fixed training set: Once features have been learned that always achieve perfect recognition with zero residual entropy, expert learning ends (cf. Section 5.3). At this stage, the learned feature set will generally still be imperfect in that they do not fully disambiguate all pairs of classes with a KSD of 1.0. It is features with a weak KSD and the potential to eliminate all entropy that cause the system to fail on unseen test images. In a practical, open task, this is not a problem because any imperfectly recognized test image automatically becomes a training image, and learning continues.

5.4.3 The Mel Task

The Mel task reflects the general phenomena observed in the COIL and Plym tasks, but to a lesser extent (Figure 5.8 and Table 5.7). Contrary to this development is the slight increase of wrong recognitions in Fold 2, and the emergence of one confused recognition in both folds (in Fold 1, an abalone board was mistaken for a phone-cord and a bike-chain; in Fold 2, a sock was mislabeled as grapes and bike-chain). In Fold 2, no instances of phone-cord were recognized correctly, as was the case for both folds after the initial learning stage (see Table 4.7 on page 58). In contrast, Fold 1 achieved some success on this class, but only at the expense of 20 features dedicated to it. This is the largest number of features observed for any object in all experiments after expert learning. In fact, for both folds the number of phone-cord features increased significantly during expert learning, while it decreased for almost all other classes in all experiments.

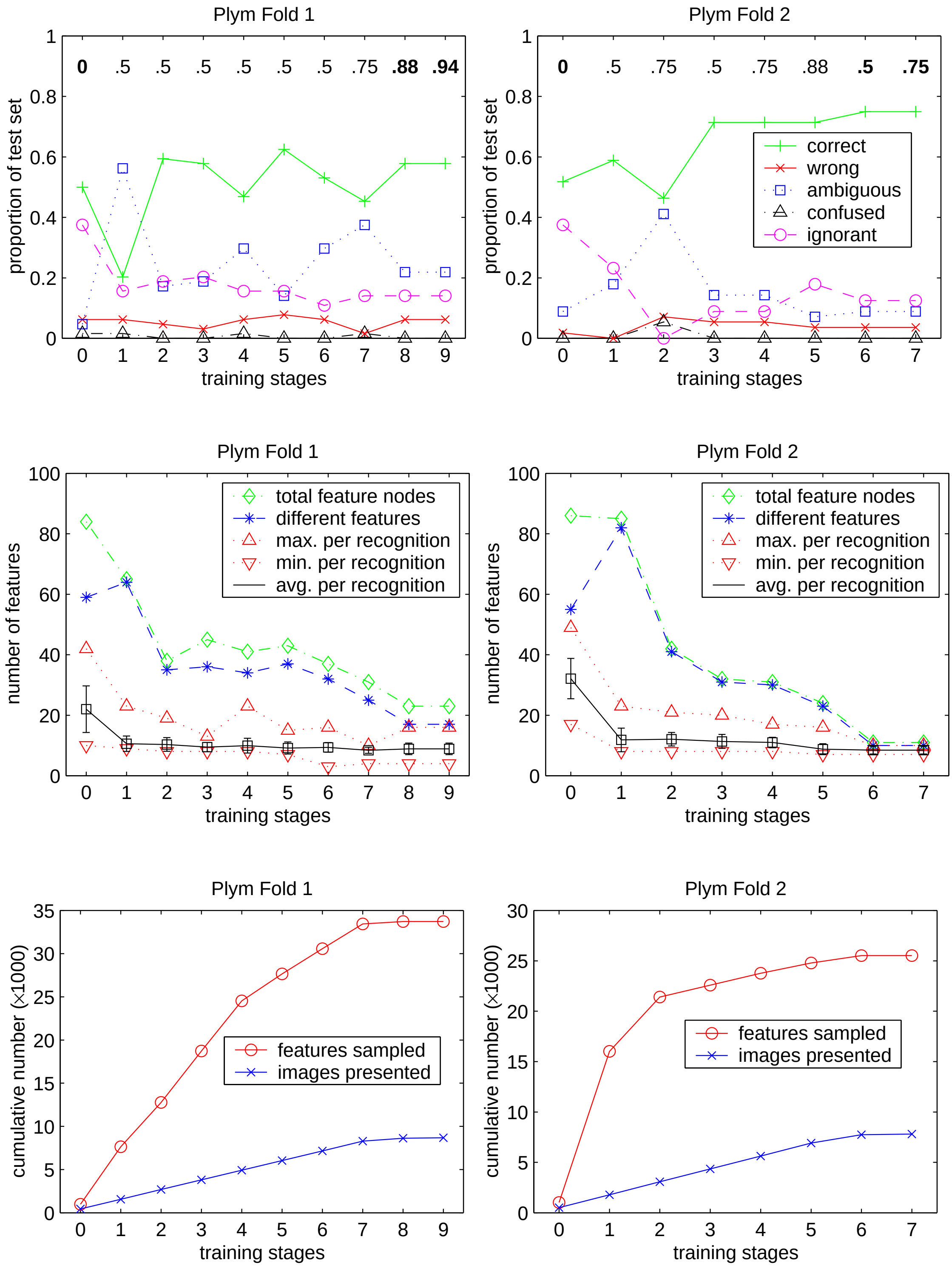


Figure 5.6. Expert Learning results on the Plym task. The numbers shown inside the top graphs give the value of t_{KSD} at the respective training stage. Where this number appears in bold face, the training set was learned with perfect accuracy, and the t_{KSD} goal was achieved. Otherwise, either the training set was not learned perfectly, or the t_{KSD} goal was not attained. The error bars in the middle plots have a width of one standard deviation.

Table 5.4. Results on the Plym task after Expert Learning. The table shows, for each class, how many instances were recognized as which class for correct and wrong recognitions, and the number of ambiguous/confused/ignorant recognition outcomes. In each row, these entries sum to the number of test images of each class (here, 8 in Fold 1 and 7 in Fold 2). The rightmost column shows the number of feature nodes that are part of the Bayesian network representing each class. The bottom row of the bottom panel gives the average proportion of the test set that falls into each recognition category.

	Confusion Matrix correct/wrong											# Feats.
Class	con6	cube	cucon	cucy	cycu	cyl3	cyl6	tub6	amb.	cnf.	ign.	BN
Fold 1:												
con6	6								2			1
cube		2							2		4	7
cucon			3						4		1	1
cucy				4				1	2		1	1
cycu					6				1		1	1
cyl3						8						3
cyl6	1				2		3				2	8
tub6								5	3			1
Sum	7	2	3	4	8	8	3	6	14	0	9	23
Fold 2:												
con6	7											1
cube		5				1					1	4
cucon			4						2		1	1
cucy				6							1	1
cycu					5				2			1
cyl3						7						1
cyl6							4	1			2	1
tub6								4	1		2	1
Sum	7	5	4	6	5	8	4	5	5	0	7	11
Average of both folds (proportions):												
con6	.87								.13			1
cube		.47				.07			.13		.33	6
cucon			.47						.40		.13	1
cucy				.67				.07	.13		.13	1
cycu					.73				.20		.07	1
cyl3						1.00						2
cyl6	.07				.13		.47	.07			.27	5
tub6								.60	.27		.13	1
Avg.	.12	.06	.06	.08	.11	.13	.06	.09	.16	0	.13	2

Table 5.5. Results on the Plym task after Expert Learning: Confusion matrices of ambiguous recognitions. The italic numbers on the diagonal indicate how many times the class given on the left was recognized ambiguously (cf. Table 5.4), and the upright numbers in each row specify the false-positive classes.

Class	con6	cube	cucon	cucy	cycu	cyl3	cyl6	tub6
Fold 1:								
con6	*2*		1		1			
cube	2	*2*						
cucon		2	*4*	1		1		
cucy		1		*2*		1		
cycu		1			*1*			
cyl3								
cyl6								
tub6	1			1	1	2		*3*
Fold 2:								
con6								
cube								
cucon			*2*		2	1		
cucy								
cycu		1	1	1	*2*			
cyl3								
cyl6								
tub6							1	*1*

Table 5.6. Results on a two-class Plym subtask after 15-fold Expert Learning.

	correct/wrong					
Class	cube	cucon	amb.	cnf.	ign.	Sum
cube	13	1			1	15
cucon		14	1			15
Sum	13	15	1	0	1	30

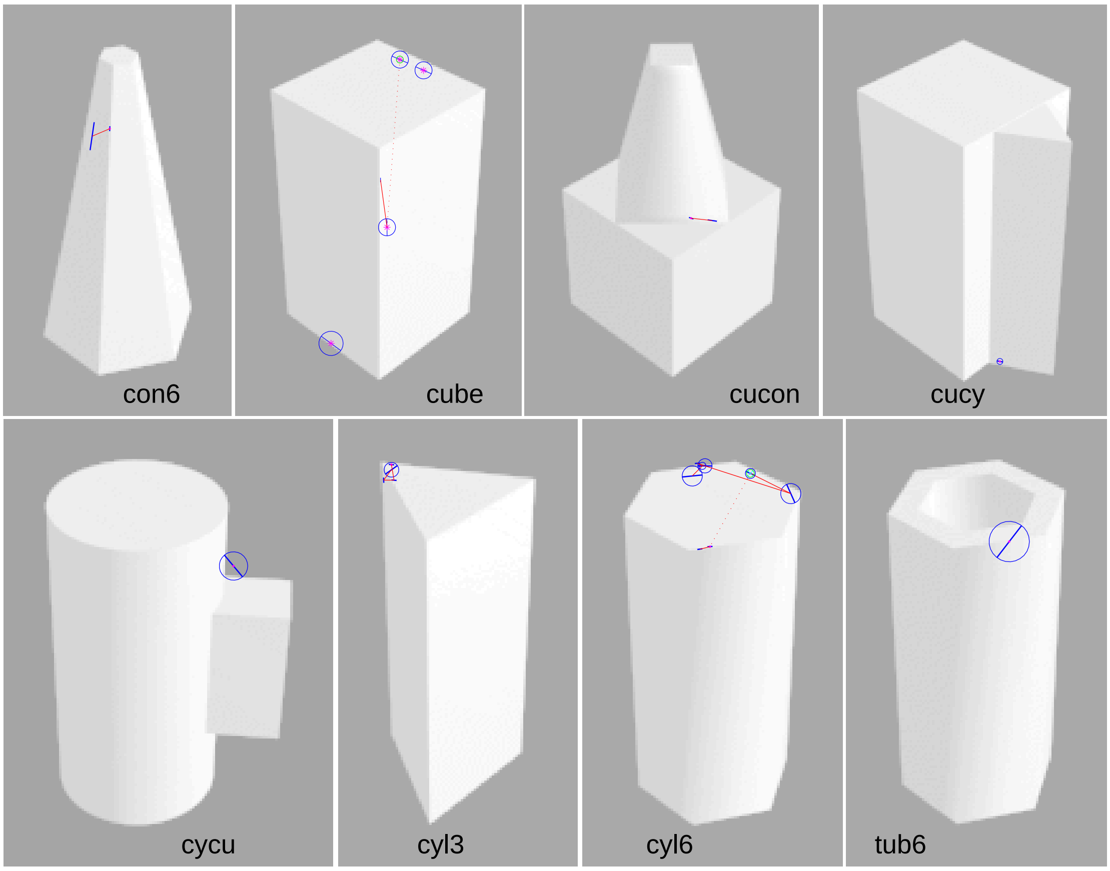


Figure 5.7. Examples of expert features learned for the Plym task, Fold 1.

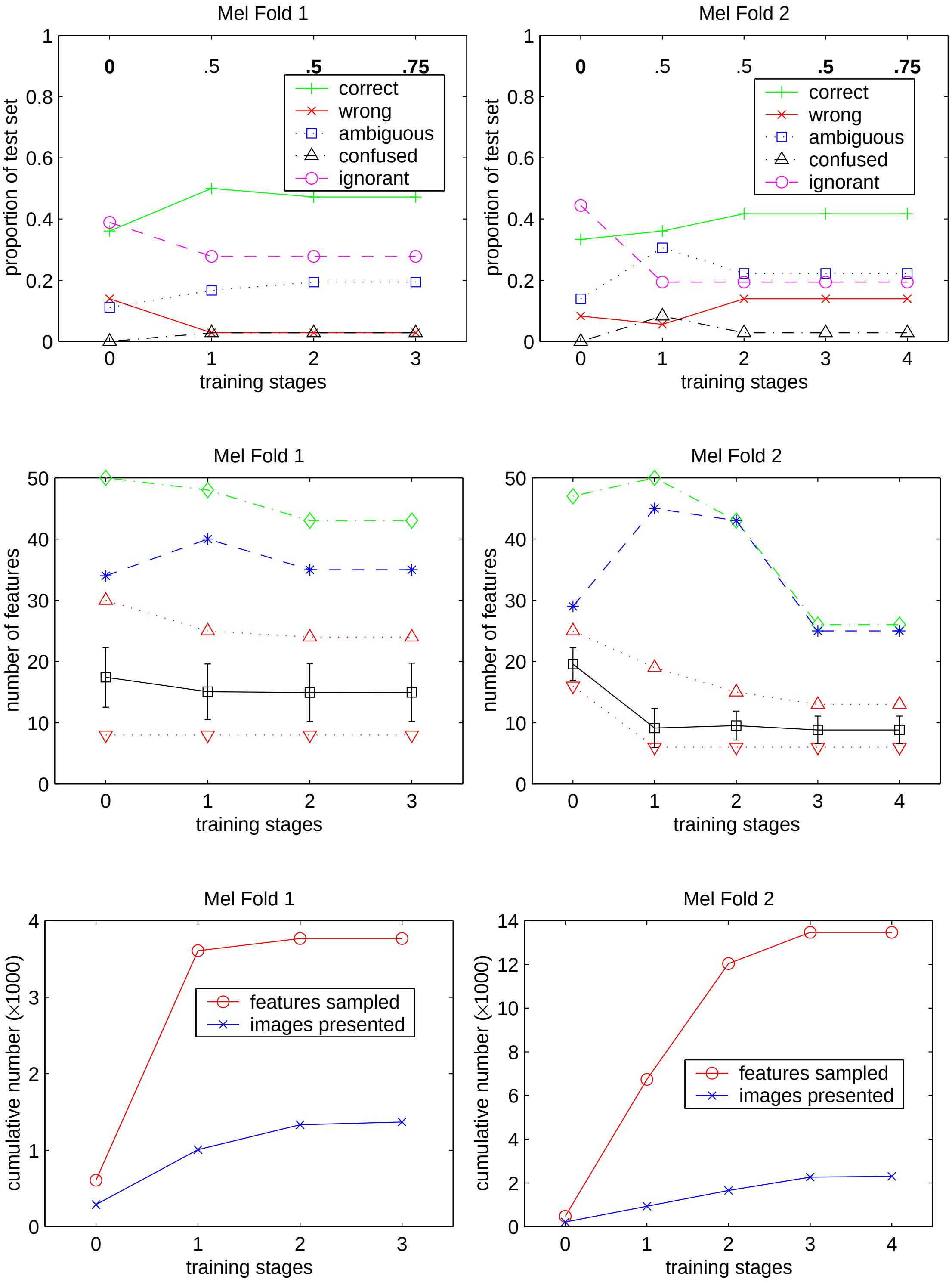


Figure 5.8. Expert Learning results on the Mel task. The numbers shown inside the top graphs give the value of t_{KSD} at the respective training stage. Where this number appears in bold face, the training set was learned with perfect accuracy, and the t_{KSD} goal was achieved. Otherwise, either the training set was not learned perfectly, or the t_{KSD} goal was not attained. The error bars in the middle plots have a width of one standard deviation.

Table 5.7. Results on the Mel task after Expert Learning. The table shows, for each class, how many instances were recognized as which class for correct and wrong recognitions, and the number of ambiguous/confused/ignorant recognition outcomes. In each row, these entries sum to the number of test images of each class (here, 6). The rightmost column shows the number of feature nodes that are part of the Bayesian network representing each class. The bottom row of the bottom panel gives the average proportion of the test set that falls into each recognition category.

	Confusion Matrix correct/wrong									# Feats.
Class	abalone	chain	grapes	necktie	cord	sock	amb.	cnf.	ign.	BN
Fold 1:										
abalone	2						2	1	1	5
bike-chain		3					3			7
grapes			1			1			4	7
necktie				2			2		2	1
phone-cord					4				2	20
sock						5			1	3
Sum	2	3	1	2	4	6	7	1	10	43
Fold 2:										
abalone	3					1	2			1
bike-chain		1		1			2		2	1
grapes			4				2			8
necktie				4			2			1
phone-cord			3						3	14
sock						3		1	2	1
Sum	3	1	7	5	0	4	8	1	7	26
Average of both folds (proportions):										
abalone	.42					.08	.33	.08	.08	3
bike-chain		.33		.08			.42		.17	4
grapes			.42			.08	.17		.33	8
necktie				.50			.33		.17	1
phone-cord			.25		.33				.42	17
sock						.67		.08	.25	2
Avg.	.07	.06	.11	.10	.06	.14	.21	.03	.24	6

In contrast to the other two tasks, performance on the Mel task is characterized by widely varying results and a lack of discernible patterns, even after expert learning. This is also reflected by the many ambiguous results and the lack of agreement between the folds (see Table 5.8). It seems surprising that even though features were learned that disambiguated all classes with nonzero entropy with a KSD of 0.75 (see Figure 5.8, and Figure 5.9 for examples of the features), the test-set results were still relatively poor. This suggests that the training set was too small in relation to the enormous within-class variety of appearances, enabling the feature learning procedure to discover highly discriminative features that generalized poorly nevertheless. Also, recall that expert learning acts only in the case of nonzero residual entropy in at least one class node. In the Mel task, expert learning was apparently hampered by the abundant occurrence of overly specific features that responded to very few images. Such features, if not found in an image, are likely to drive the belief in their class to zero, as discussed in Section 4.4.4 on page 44, disabling expert learning.

Table 5.8. Results on the Mel task after Expert Learning: Confusion matrices of ambiguous recognitions. The italic numbers on the diagonal indicate how many times the class given on the left was recognized ambiguously (cf. Table 5.7), and the upright numbers in each row specify the false-positive classes.

Class	abalone	chain	grapes	necktie	cord	sock
Fold 1:						
abalone	*2*				2	
bike-chain		*3*		1	1	1
grapes						
necktie		1		*2*	1	1
phone-cord						
sock						
Fold 2:						
abalone	*2*		1			2
bike-chain		*2*	1			1
grapes			*2*			2
necktie		1	2	*2*		
phone-cord						
sock						

In order to find out to which extent the Mel task is learnable by the system, a leave-one-out cross validation procedure was run on a two-class subtask, using two of the most problematic classes, grapes and phone-cord. In Fold 1 of the full six-class task, grapes were the poorest class to be recognized (Table 5.7); in Fold 2, phone-cord was the poorest class, and half of the test images were mislabeled as grapes. A confusion matrix summarizing the results of the twelve folds on the two-class subtask is shown in Table 5.9. The results show that the feature space has some power to express this class distinction, but they are still not outstanding. The accuracy is 0.83, with 0.5 being chance performance. Clearly, the structure of the Mel task is not easily captured by this learning algorithm. This topic will receive further attention in Section 7.3.

5.4.4 Incremental Tasks

In all experiments discussed so far, training was incremental in that images were presented to the learner sequentially. Training images were chosen from all classes in random order. In many practical situations, a more realistic scenario will require the learner to acquire concepts

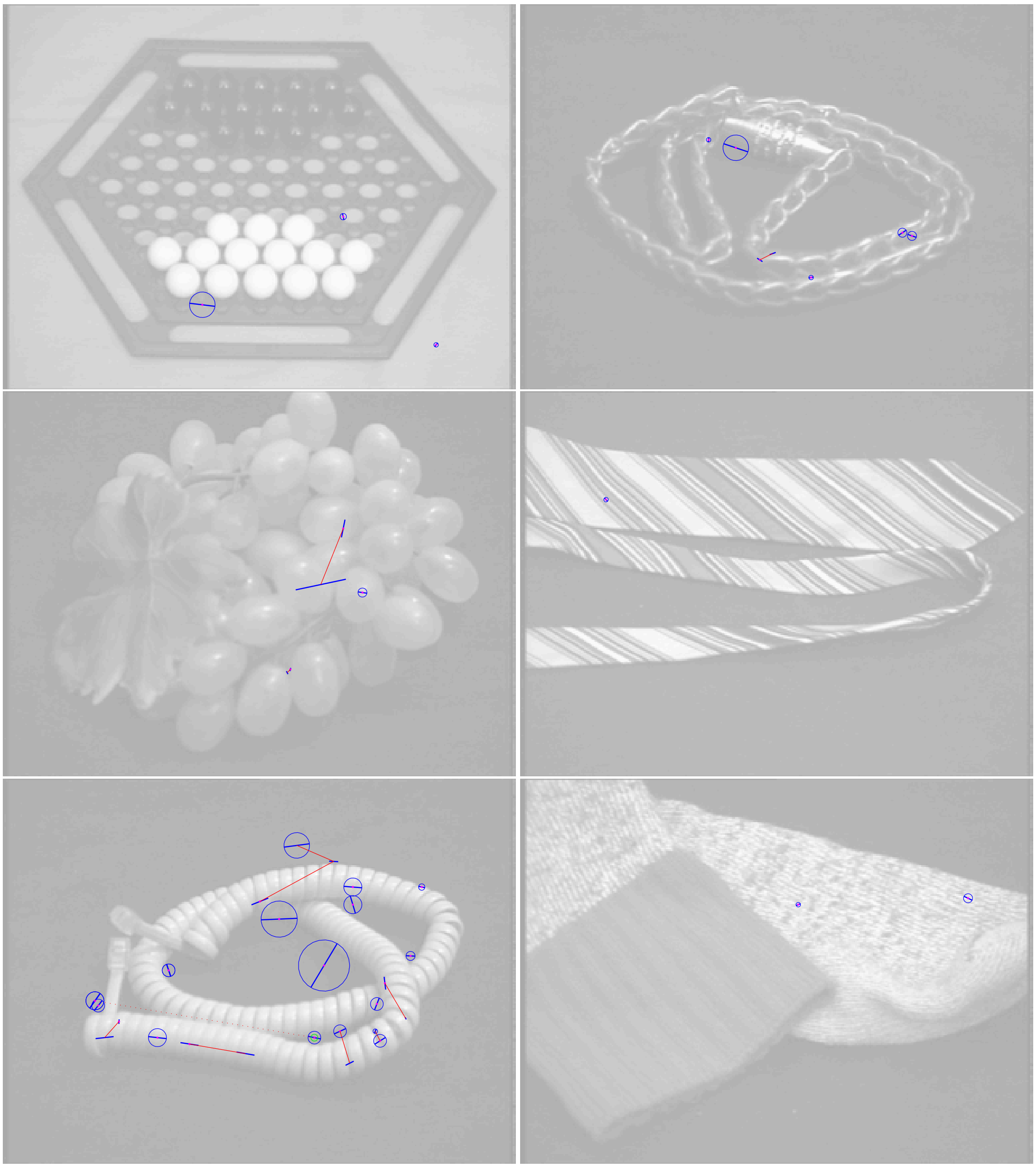


Figure 5.9. Examples of expert features learned for the Mel task, Fold 1.

Table 5.9. Results on a two-class Mel subtask after 12-fold Expert Learning.

	correct/wrong					
Class	grapes	cord	amb.	cnf.	ign.	Sum
grapes	9		2		1	12
phone-cord	1	8	1		2	12
Sum	10	8	3	0	3	24

sequentially. In particular, this is how humans are taught. It helps us to master skills if we learn only a limited number of them in parallel. In this section, tasks are presented to the learner incrementally, as described in the following algorithm.

Algorithm 5.4 (Incremental Task) To train the learning system on an incremental task, begin with a blank-slate learner, and then perform the following steps.

1. Create a training set consisting of all training images of the first two classes.
2. Train the system using Algorithm 5.3, with one minor modification: At Step 4, stop after a maximum of 10 stages of expert learning.
3. If there are untaught classes, then add all training images of the next untaught class to the training set, and continue with Step 2. Otherwise, stop.

Importantly, the experience of the learner is not discarded between executions of Step 2. ■

The results achieved on the three tasks using Algorithm 5.4 are included in Table 5.1, with '-inc' appended to the task name. Interestingly, in all cases the achieved accuracy was slightly better than in the corresponding non-incremental tasks. Moreover, the number of features sampled – which is closely related to the overall running time of the system – is slightly reduced for the COIL task. The number of cycles through the training set is not given in Table 5.1. It has little meaning for incremental tasks, as the size of the training set increases over time. The number of training image presentations is drastically increased. The reason for this is that the growing number of already-learned images is included with every cycle, while learning concentrates primarily on images of the new class. This effect can be avoided by more sophisticated incremental learning schemes. For example, the simulated environment could choose the class of a training image according to the empirical misrecognition rate attained by the learner. In any case, the choice of a learning strategy depends on the interaction dynamics between the learner and its environment. An environment or a task may or may not allow the learner to influence the distribution of training images.

The COIL-inc task was continued up to a total of 10 classes, which are illustrated in Figure 5.10. Figures 5.11 and 5.12 show performance parameters on a test set as they evolve during incremental expert training. The test set contains the same classes as the training set. Usually, the introduction of a new class causes a temporary drop in performance that is reflected by most performance parameters: The proportion of correct recognitions drops, while most of the other parameters increase. Over the next few stages of expert training, the system recovers. Importantly, the transient drops in performance co-occur with an imperfectly learned training set, as indicated by the non-bold KSD indicators in Figures 5.11 and 5.12. In other words, good performance on the training images generally predicts good performance on unseen test images. This is good news, given the often non-monotonic performance improvement during expert learning. Nevertheless, a slight drop in accuracy is noticeable, as the number of classes increases.

How difficult are individual classes to learn? There is no way to answer this question for the non-incremental version of expert training. However, incremental training lends insight into a related question: How much effort does it take to learn a new class, after having learned some other classes? Figure 5.13 reveals that some classes are learned after sampling only a small number of features, while others require a lot of sampling effort. Both folds learned the first four objects quite effortlessly, while obj7 and obj8 were more difficult. Interestingly, both folds spent most effort disambiguating obj7 from obj2 – these are the two wooden blocks in the data set. In addition, Fold 1 often confused obj7 with obj6, possibly due to the striped texture shared by both objects. In the case of obj8, there was no such singular source of confusion. Nevertheless, the two classes that accounted by far for most confusions were obj1 and obj6. Almost all confusions on the training set that occurred at any time during learning involved the

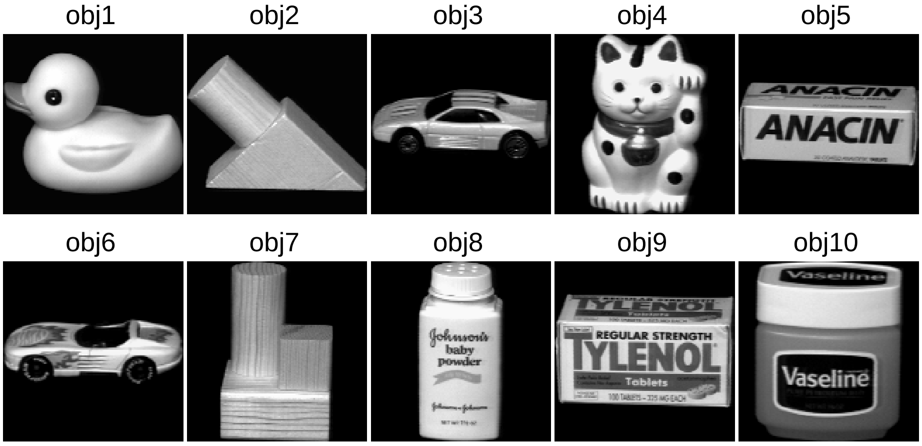


Figure 5.10. Objects used in the 10-object COIL task.

most recently introduced class, especially during the early stages of expert learning. Beyond these qualitative remarks, it is difficult to draw strong conclusions from only two folds of cross validation, and from a single presentation order of new classes.

The number of expert learning stages taken until a class was learned is clearly correlated with the number of features sampled. However, it is not as reliable an indicator of class difficulty, because – in contrast to the number of features sampled – it is largely unrelated to the proportion of difficult training images of a class. A single difficult image can stretch expert learning over many stages, but far fewer features are sampled than in the case of many difficult images. For example, in Fold 1, for obj6 11,470 features were sampled during 4 stages, whereas for obj10 only 5,760 features were sampled during 9 stages.

The results for the incremental Plym task (Figures 5.14, 5.15 and 5.16) show the same characteristics as the COIL-inc task. Here, both folds agree that tub6, the eighth class, is the most difficult class (Figure 5.16). Not surprisingly, confusions with cyl6 accounted for the overwhelming majority of the feature learning efforts. Notably, Fold 1 had trouble with a few individual images while learning the sixth and seventh classes (cyl3 and cyl6). At almost all of these stages, Algorithm 5.3 at Step 2 cycled through the training set the maximum of 20 times, while sampling very few features.

The Mel-inc results again show similar behavior, but to a lesser extent (Figures 5.17 and 5.18). These graphs reflect the same variability as discussed for the other Mel experiments above. In contrast to the COIL and Plym tasks, it appears that the number of features sampled grows purely as a function of the number of classes. This is another indicator of the intrinsic difficulty of the Mel data set, which makes it hard for the learning procedure to find powerful and well-generalizing features. Interestingly, while learning the fourth class (necktie) in Fold 2, only 69 features were sampled, all of them during Stage 0. The following four stages of expert learning only re-estimated the conditional probabilities in the Bayesian networks, but never required a new feature.

In both folds, the system failed to recover after the introduction of the sixth class (sock). This is somewhat surprising, given that this class performed better than any other class after non-incremental training (Table 5.7), and is yet another indicator that expert learning was severely hampered on the Mel task. Expert learning essentially failed here because a large number of overly specialized features caused most recognitions to result in zero entropy in all class nodes. Such an effect is only likely to occur on static, small, and heterogeneous training sets such as that used in the Mel task.

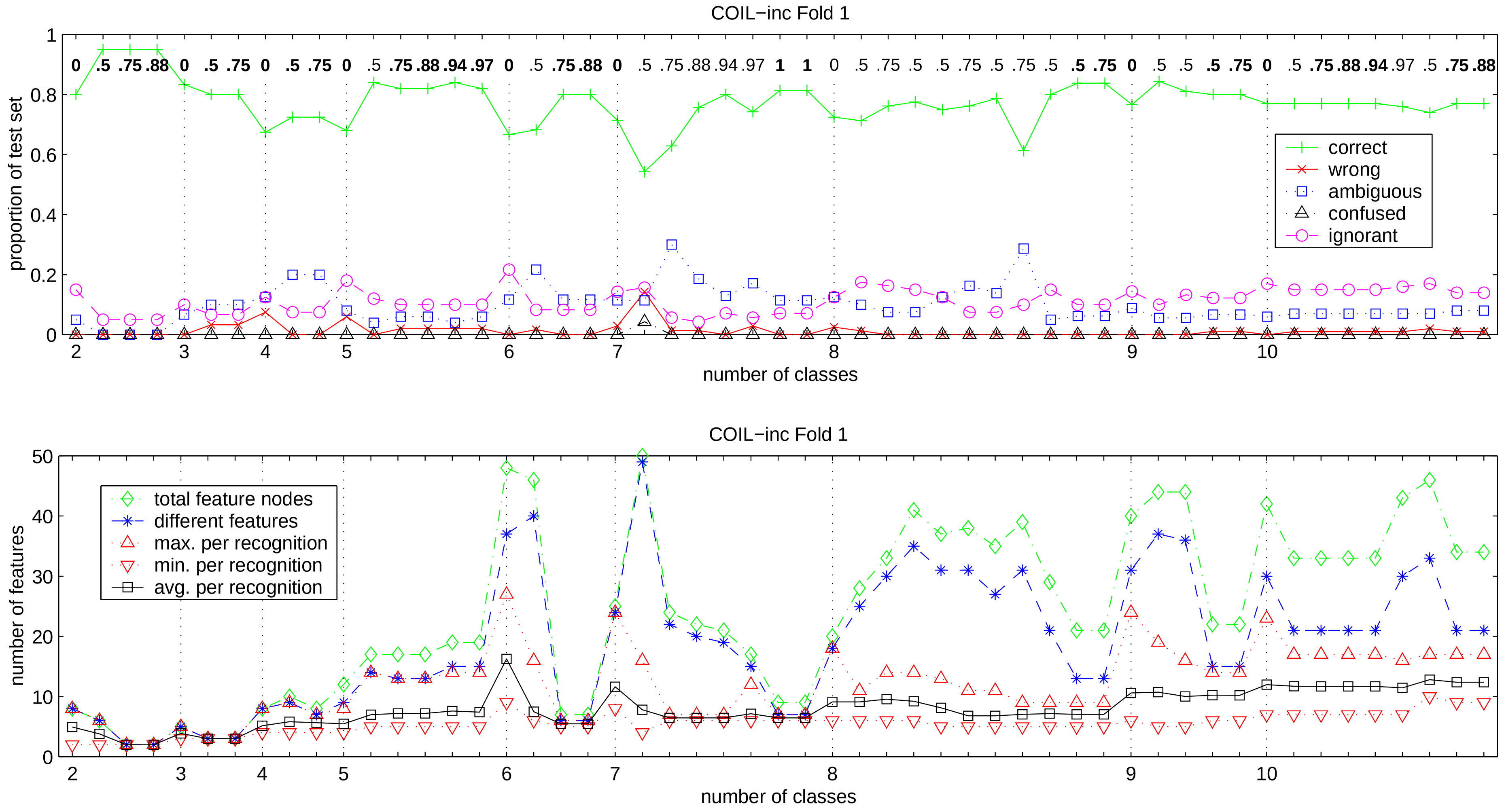


Figure 5.11. Incremental Expert Learning results on the COIL-inc task, Fold 1. The vertical dotted lines indicate the addition of a new class to the training set. Below the graphs, these are annotated with the total number of classes represented in the training set. Data points between these lines correspond to expert learning stages (cf. Figure 5.1, page 68), which are not numbered here. The numbers inside the top graph indicate the values of t_{KSD} as in Figure 5.1.

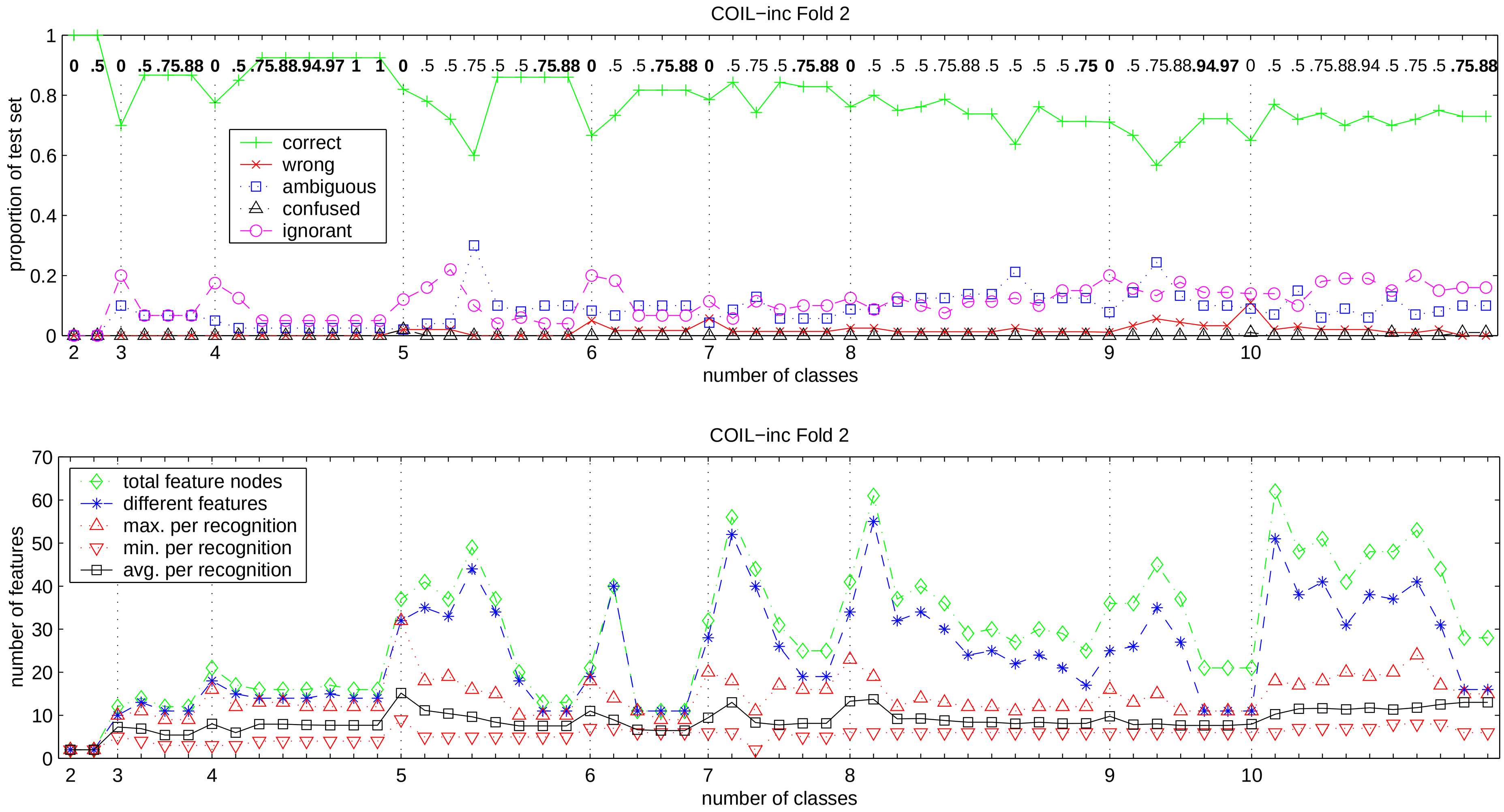


Figure 5.12. Incremental Expert Learning results on the COIL-inc task, Fold 2. The vertical dotted lines indicate the addition of a new class to the training set. Below the graphs, these are annotated with the total number of classes represented in the training set. Data points between these lines correspond to expert learning stages that are not numbered here. The numbers inside the top graph indicate the values of t_{KSD}.

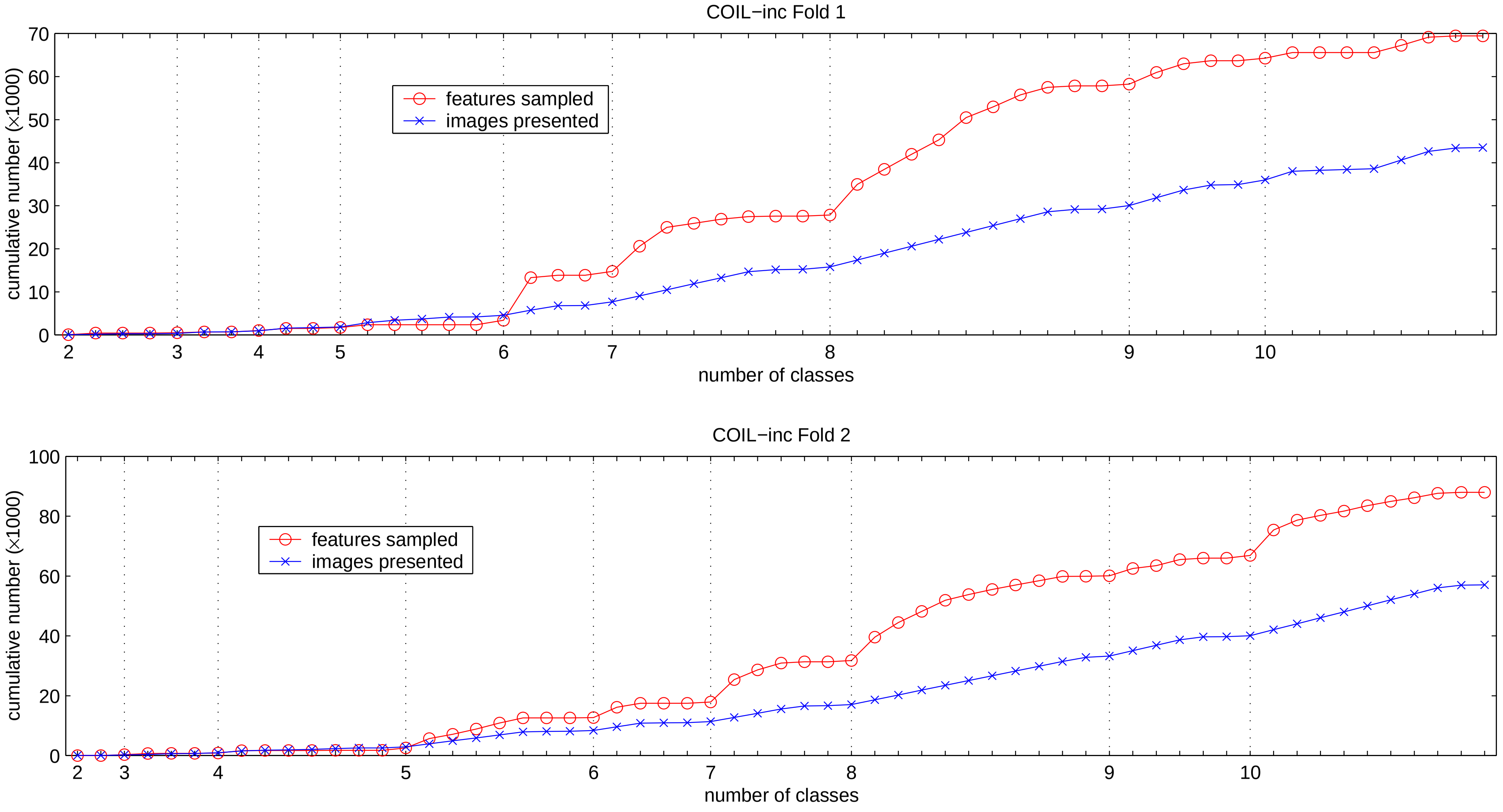


Figure 5.13. Numbers of features sampled during the COIL-inc task.

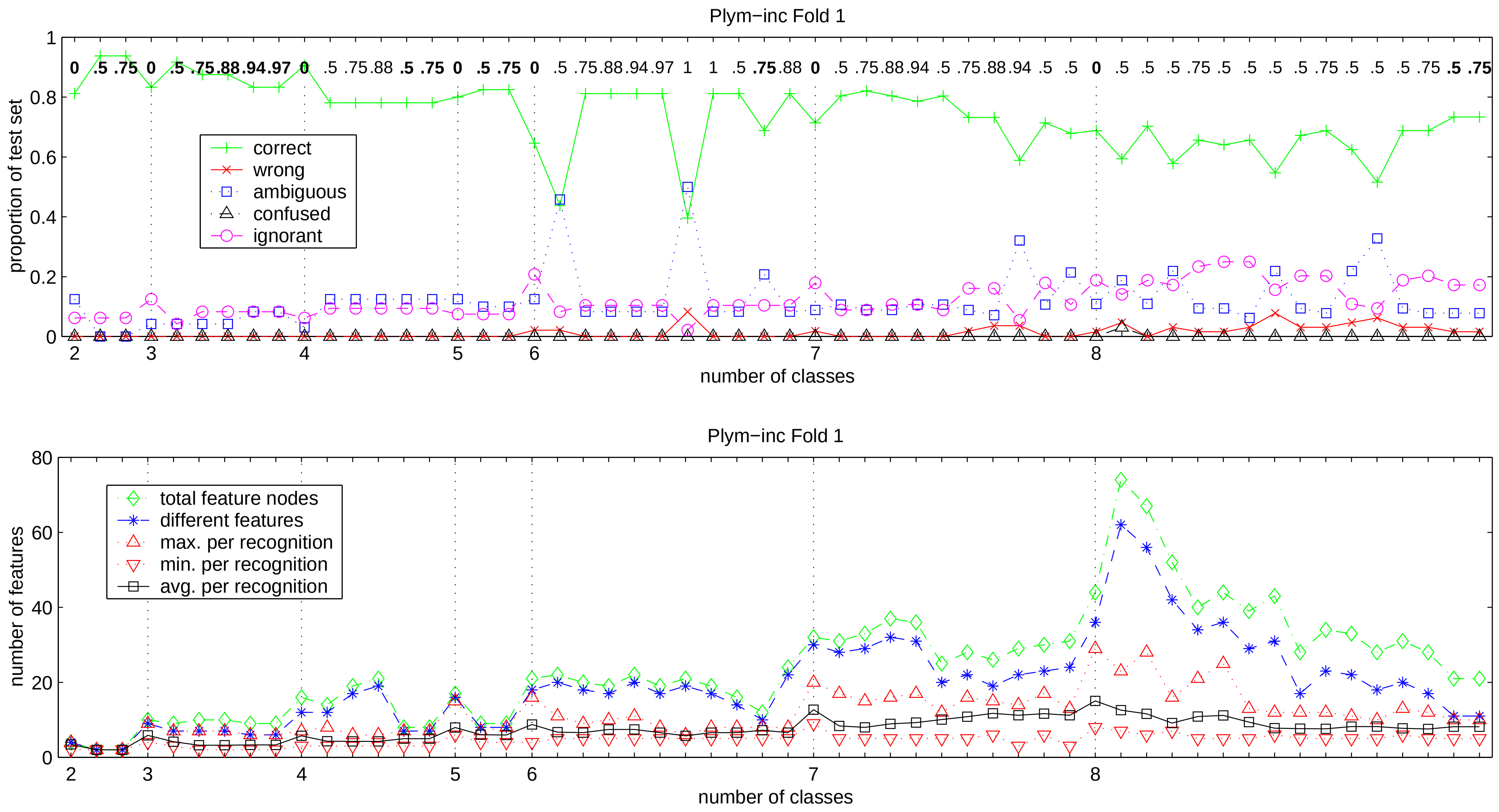


Figure 5.14. Incremental Expert Learning results on the Plym-inc task, Fold 1. The vertical dotted lines indicate the addition of a new class to the training set. Below the graphs, these are annotated with the total number of classes represented in the training set. Data points between these lines correspond to expert learning stages that are not numbered here. The numbers inside the top graph indicate the values of t_{KSD}.

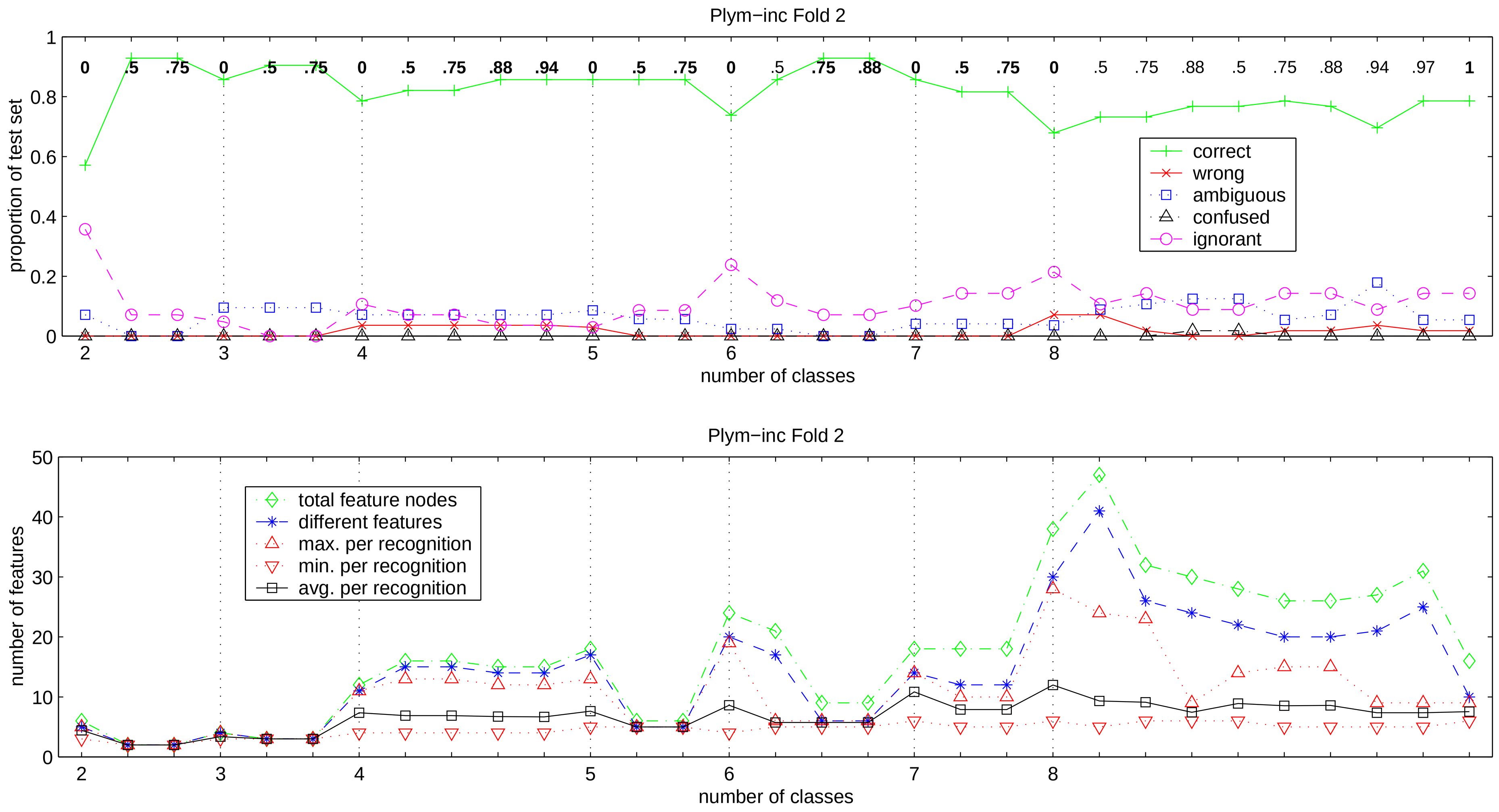


Figure 5.15. Incremental Expert Learning results on the Plym-inc task, Fold 2. The vertical dotted lines indicate the addition of a new class to the training set. Below the graphs, these are annotated with the total number of classes represented in the training set. Data points between these lines correspond to expert learning stages that are not numbered here. The numbers inside the top graph indicate the values of t_{KSD}.

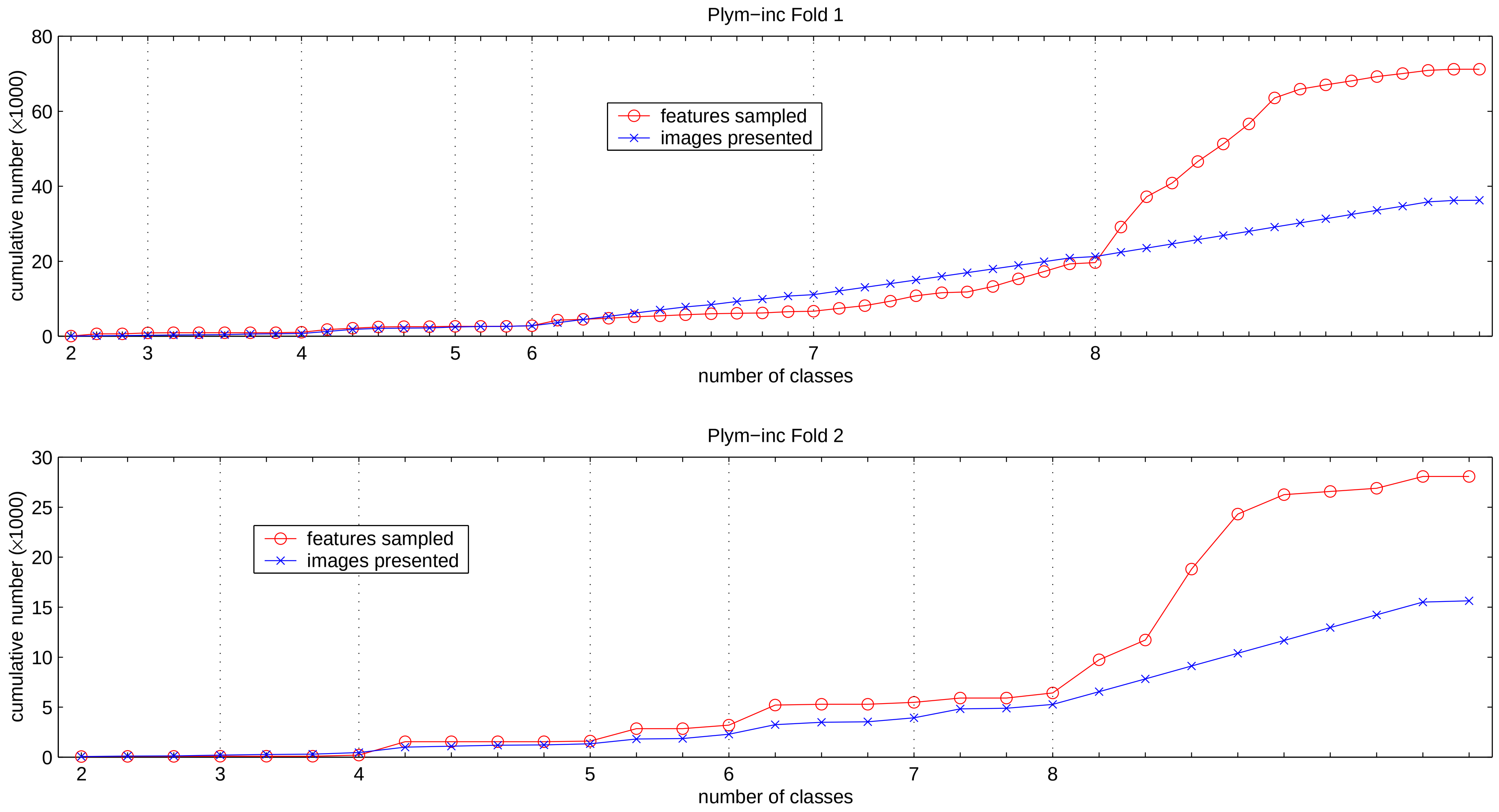


Figure 5.16. Numbers of features sampled during the Plym-inc task.

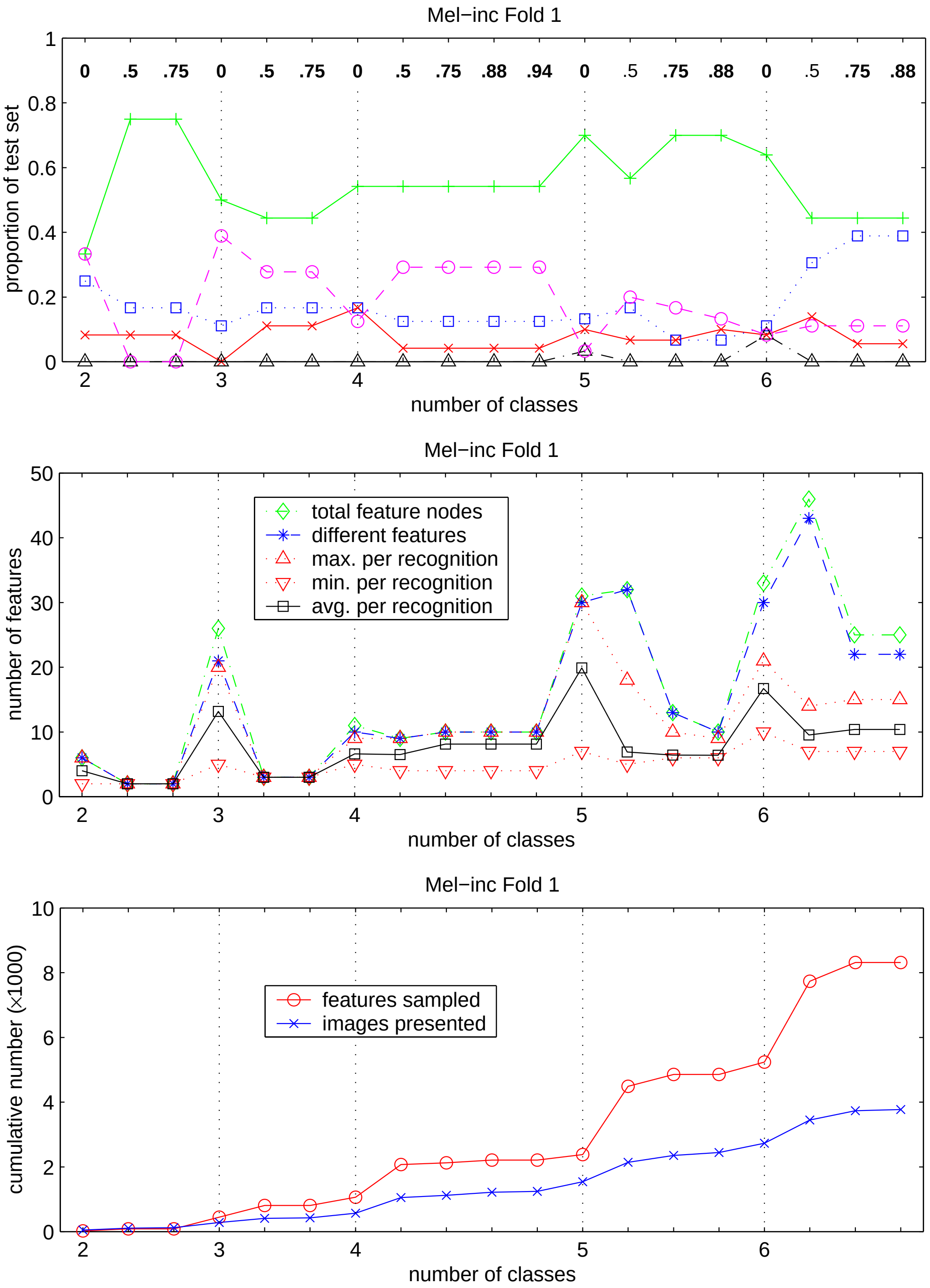


Figure 5.17. Incremental Expert Learning results on the Mel-inc task, Fold 1. The vertical dotted lines indicate the addition of a new class to the training set. Below the graphs, these are annotated with the total number of classes represented in the training set. Data points between these lines correspond to expert learning stages that are not numbered here. The numbers inside the top graph indicate the values of t_{KSD}.

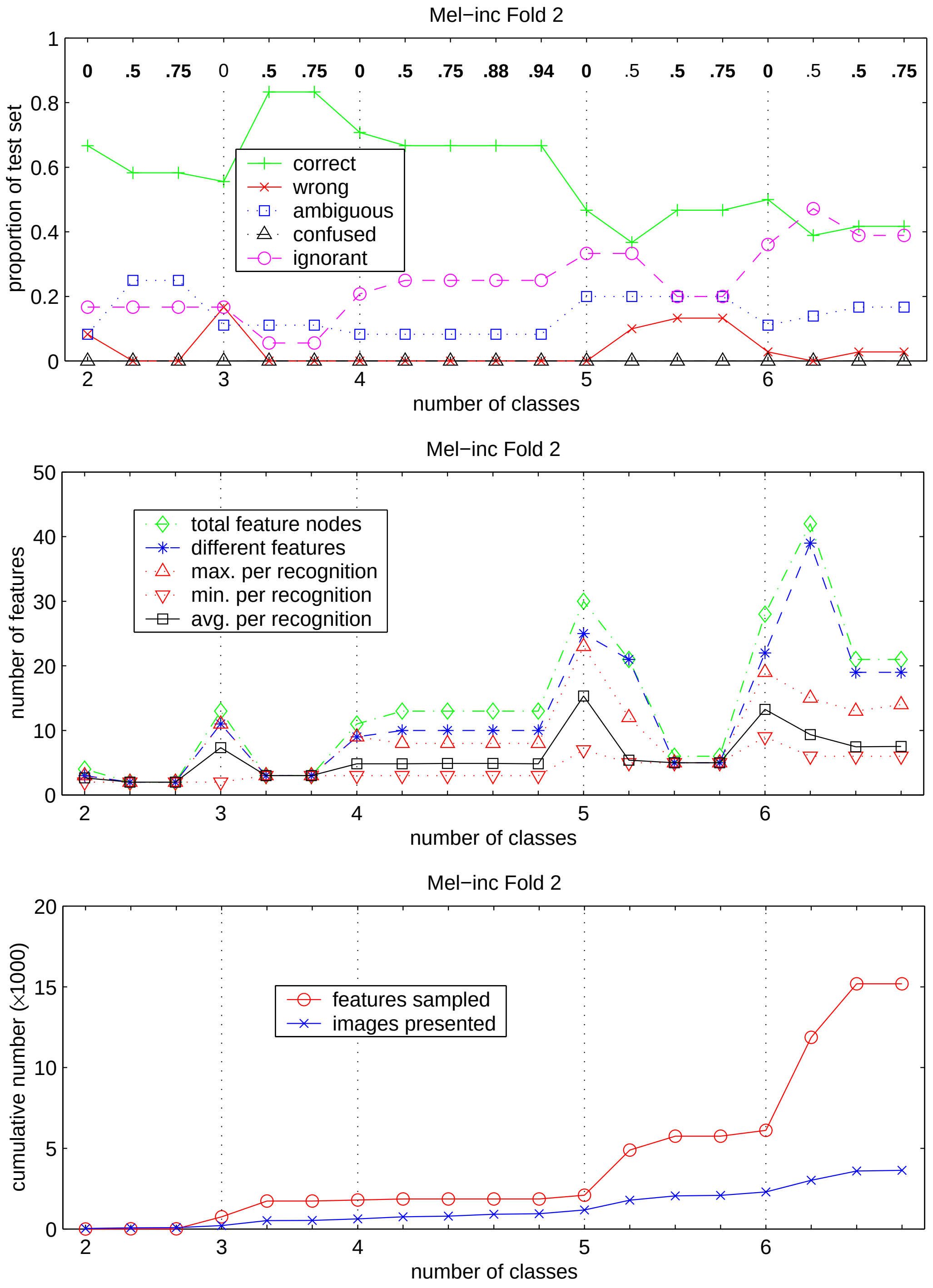


Figure 5.18. Incremental Expert Learning results on the Mel-inc task, Fold 2. The vertical dotted lines indicate the addition of a new class to the training set. Below the graphs, these are annotated with the total number of classes represented in the training set. Data points between these lines correspond to expert learning stages that are not numbered here. The numbers inside the top graph indicate the values of t_{KSD}.

5.4.5 Computational Demands

The current method attempts to find features by random sampling in image space, which is guided by a set of simple-to-complex heuristics defined by the feature generation algorithm 4.10 (page 43). Highly distinctive features are rare and difficult to find, as illustrated by the graphs in this chapter. This opens up the question of computational demands. The current MATLAB$^{\text{TM}}$ implementation of the learning system takes many days of computation on a 700 MHz Pentium III processor to learn a typical task. The bulk of this time is spent measuring the value of features in images (Equation 3.14 on page 25). This is the primary reason why the extraction of salient points is so important (Section 3.4.3, page 21). For every feature **f** sampled, its value $f_{\mathbf{f}}(I)$ must be computed on $2n_{\text{ex}}$ example images (Algorithm 4.8, Feature Learning, page 41, Steps 2 and 5).

Given the number of pixels in an image (16,384 for the COIL images, 76,800 for the Mel images) and the infinite combinatorics opened up by the feature space, sampling 10,000 or 100,000 features to solve a task involving 100 training images does not seem outrageous. What would it take to run the feature learning algorithms on massively parallel neural hardware? Computing feature values is a pixel-level parallel operation that can plausibly be performed by the human visual cortex. Such a parallel implementation could measure a feature value in an image in constant time, independent of the number of salient points in an image. Assuming an average of 1,000 salient points in an image, and assuming that 99% of the compute time is spent computing feature values, this would reduce the time to solve a reasonably large problem from, say, 10 days to less than 2 hours 40 minutes – about two orders of magnitude.

The above argument assumes that the time spent to compute a feature value is approximately constant. This is not strictly true, as it depends on the length of a response vector, which is 15 filter responses for texels and 2 for edgels. It also depends on the local scale of a feature and its subfeatures, which determines the size of the filter kernels used in Algorithm 3.6 (Extraction of a Geometric-Compound Feature Response Vector, page 26) at Steps 1 and 4. This size varies from about 25 coefficients up to about 1/16 the size of the image. The early visual cortex extracts such response vectors in parallel, for all image locations, over a range of scales, reducing this computation to constant time. If these factors are taken into account, the computation time can be reduced by at least two more orders of magnitude.

In summary, the computational demands of the expert learning procedure limit the size of the problems that can be addressed using today's conventional computing hardware. However, a biologically plausible implementation that exploits the high degree of inherent parallelism should reduce the computational demands by about four orders of magnitude. In principle, this would permit much larger-scale applications.

With respect to scalability, an important consideration is the number of distinctions that has to be learned by the system. In general, this number is quadratic in the number of classes: Each class is to be discriminated from each of the other classes. Figure 5.19 plots the cumulative number of features sampled in the incremental tasks versus the number of classes. Clearly, the growth is superlinear. However, there are not sufficient data to allow a conclusive statement regarding the complexity of the learning algorithm with respect to the number of classes. More experiments using more classes are required. In any case, this superlinear growth is likely to limit the maximal problem size that can be addressed by this system. However, this need not constitute a practical limitation. Humans clearly do not learn expert-level distinctions between all categories that have any meaning to them in any context. Rather, at any given time, the behavioral context constrains the relevant categories to a relatively small set. Expert learning is only required where distinctions need to be refined that are relevant within a given context.

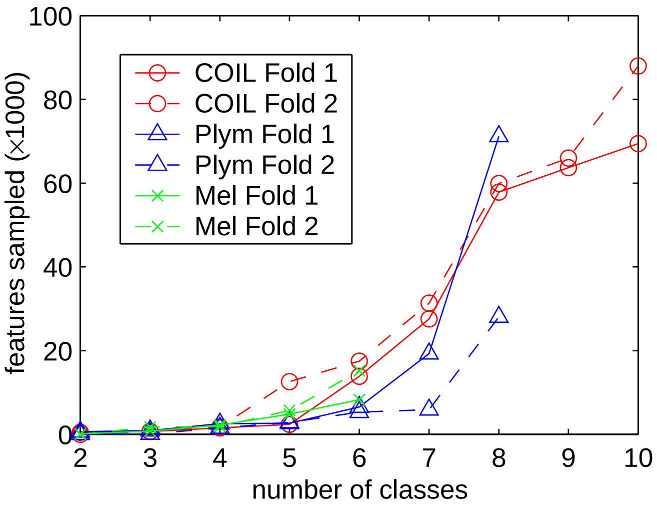


Figure 5.19. Number of features sampled for various tasks, plotted versus the number of classes.

5.5 Discussion

Motivated by the human phenomenon of visual expertise, this chapter introduced a simple algorithm for learning improved features. The idea is to seek features with increased discriminative power between pairs of classes. Thus, expert learning has exactly the opposite effect of overtraining. Because explicit requirements of discriminative power are enforced, learning can – in principle – continue forever, without any danger of overfitting. Resembling human expert behavior, this has the desired effects of increased recognition accuracy and efficiency. I am not aware of any existing related work in machine learning on incremental feature improvement. The vast majority of comparable machine learning methods either use a fixed feature set, perform feature selection, or compute new features on the basis of a fixed feature set (e.g. PCA). The expert learning idea presented here critically depends on the ability of the algorithm to search a virtually unlimited feature space.

The most severe limitation of the expert learning algorithm, as presented here, is that it depends on class nodes with nonzero entropy at the end of a recognition procedure. Consequently, features that have the ability to infer classification results with absolute certainty (cf. Section 4.4.4) can effectively disable expert learning. This is especially a problem if these features are poor, i.e. they have little discriminative power, because they prevent their own replacement by better features. This was the case in the Mel data set. However, such features are only likely to occur in small, static, and heterogeneous data sets such as the Mel data. They emerge because the exact same images are presented to the learner over and over, until they are recognized correctly. On an interactive task in a physical environment, no training image is ever going to be seen again, and training images from the a class are drawn from a continuum of appearances. This prevents the memorization of individual training images.

Other ways to trigger expert learning could possibly avoid this problem even for static feature sets. A brute-force method would attempt to learn highly distinctive features between all pairs of classes, regardless of any empirical uncertainties. However, this requires a number of discriminations that is quadratic in the number of classes. Moreover, many of these will usually be unnecessary because many pairs of classes are never confused anyway. Expert learning – like the basic learning studied in Chapter 4 – ought to be driven by the task: Where uncertainties are likely to occur, improved features should be learned. This principle is implemented by the expert learning algorithm 5.1.

CHAPTER 6

LEARNING FEATURES FOR GRASP PRE-SHAPING

The preceding chapters applied feature learning in a supervised object recognition scenario. Many of the visual skills that humans acquire from experience affect their behavior at a very fine granularity. For example, between about five to nine months of age, infants learn to pre-orient their hands to match the orientation of a target object during reaching [70]. In this chapter, visual features are learned that enable similar prospective behavior in a haptically-guided robotic grasping system.

6.1 Related Work

There is extensive literature on visual servoing; a thorough review is beyond the scope of this dissertation. Most of this work falls into one of two main categories according to the coordinate frame in which error signals are computed, namely, three-dimensional workspace coordinates or two-dimensional image coordinates. Three-dimensional methods typically rely on a reasonably complete and accurate visual reconstruction of the workspace. More related to this chapter are two-dimensional approaches where manipulator control signals are directly derived from image-space errors [105, 49]. Here, local features of the manipulator and of workpieces are identified, and their actual positions are compared to reference positions. Image-space errors are then transformed into errors in the degrees of freedom of the manipulator. Instead of local appearance, another line of research builds on global appearance and computes error signals in eigen-subspaces [74].

6.2 Objective

To date, most work in sensor-guided robotic manipulation has concentrated on exclusively vision-based systems. There has been much less work on haptically-guided grasping [21, 23], or on combining haptic and visual feedback [3, 42]. This seems somewhat surprising, as these two sensory modalities ideally complement each other, especially when a dextrous manipulator is used. The visual system is most beneficial during the reach phase when no haptic feedback is available. Relatively coarse differential error signals in image space, using appearance features of the wrist or the arm, are sufficient to achieve a highly efficient vision-guided reach. When the end effector draws close to the target object, visual information can further contribute to the extent that fingertips and object details are accurately identified. However, this is difficult to achieve because these features will often be occluded by the hand itself. Moreover, even if detailed information about finger configurations can be obtained by the vision system, it will be difficult to exploit in the context of a dextrous manipulator due to the many degrees of freedom involved, many of which are redundant. This stage of the grasp is ideally suited for haptic feedback. Tactile fingertip sensors are highly sensitive to relevant surface characteristics such as the orientation of the local surface tangent. Coelho [21, 23] has demonstrated that this information can be exploited by local grasp controllers that converge toward stable grasp configurations, thereby obviating the need to solve a high-dimensional (and often intractable) planning problem in global joint space.

McCarty et al. [70] demonstrated that infants learn to pre-orient their hand during a reach to match the orientation of the target object. Notably, this prospective behavior does not depend on the visibility of the hand or of the object during the reach. Evidently, humans use an initial visual cue to initiate a reach, that is then guided by proprioception. Once the target object is contacted, the grasp is completed using tactile feedback.

Coelho's closed-loop grasping system is "blind" in the sense that no visual information is exploited during the grasp process. The objective in this chapter is to add a visual component to this system that extracts minimal visual information to pre-orient the hand appropriately in relation to the target object. Specifically, it recommends two critical parameters to the haptic system:

- the orientation (azimuthal angle) of the hand, and
- the number of fingers to use (two or three).

These recommendations are purely based on features of appearance that correlate robustly with these two parameters, using a feature learning system similar to that discussed in the preceding chapters. The learning procedure is driven entirely by the haptic grasping system, without an external supervisor.

6.3 Background

The grasping task discussed in this chapter is based on a haptically-guided grasping system developed by my colleague Jefferson Coelho, which is introduced in the next section. Then, Section 6.3.2 briefly introduces and illustrates probability distributions on angular domains. These play an important role in the algorithms presented in this chapter.

6.3.1 Haptically-Guided Grasping

Coelho [21] describes a framework for closed-loop grasping using a dextrous robotic hand (Figure 6.1) equipped with force/torque touch sensors at the fingertips. Reference and feedback error signals are expressed in terms of the object-frame *wrench residual* ρ, a six-element vector summarizing the residual forces and torques acting on the object. The current object wrench is computed using the instantaneous grip Jacobian G, which in turn is locally estimated using sensor feedback in the form of contact positions and normals. The grasp controller π receives an error signal ϵ, which for a zero reference wrench is simply the squared wrench residual $\epsilon = \rho^T\rho$, and computes incremental displacements for a subset of the fingertip contacts so as to reduce the error signal. As a result, the fingers probe the surface of the object until their positions converge to a local minimum in the squared-wrench-residual surface.

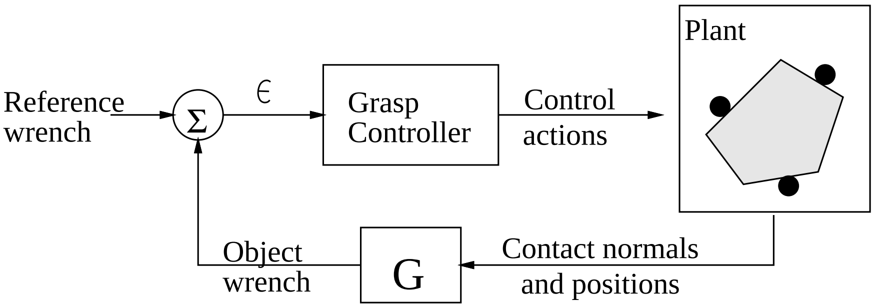


Figure 6.1. Grasp synthesis as closed-loop control. (Reproduced with permission from Coelho and Grupen [21].)

The three-fingered Stanford/JPL robotic hand (Figure 6.2) employed in this work can perform grasps using any two- or three-fingered grasp configuration. Each of these finger combinations c gives rise to an individual grasp controller π_c, yielding a total of four controllers. For the purpose of this chapter, it is sufficient to distinguish between two classes of grasp controllers, namely, two- and three-fingered grasps.

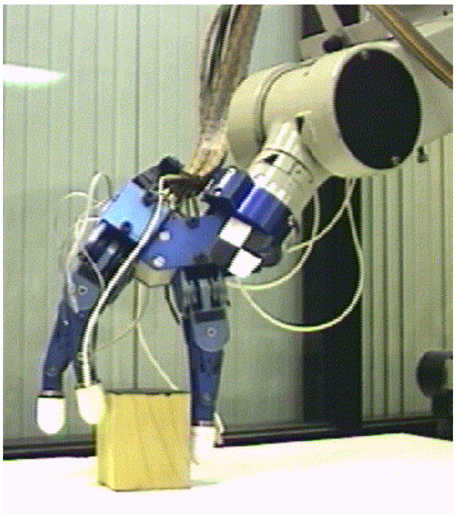


Figure 6.2. The Stanford/JPL dextrous hand performing haptically-guided closed-loop grasp synthesis.

As a grasp controller π_c executes, the control error ϵ and its time derivative $\dot{\epsilon}$ trace out phase portraits that are characteristic of object shapes (Figure 6.3). In this way, the grasping system forms haptic categories. Qualitatively different regions in a phase portrait constitute haptic *contexts* and can be represented as states of a finite state machine (Figure 6.4). Coelho [22, 23] describes a reinforcement learning framework for learning grasp control policies that select a controller π_c based on the context currently observed. The learned policies achieve grasps of consistently higher quality than any fixed *native* controller π_c by escaping local minima in the wrench residual surface.

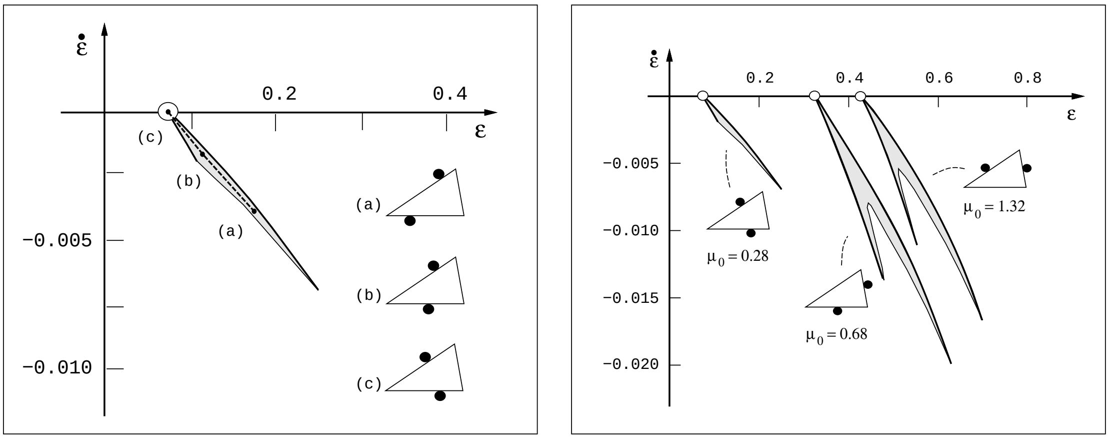


Figure 6.3. Object wrench phase portraits traced during grasp syntheses. The left panel depicts the evolution of a two fingered grasp trial from configuration (a) to (b) and to (c), the convergent configuration. The complete, two-fingered phase portrait for the irregular triangle is shown on the right. (Reproduced with permission from Coelho et al. [24])

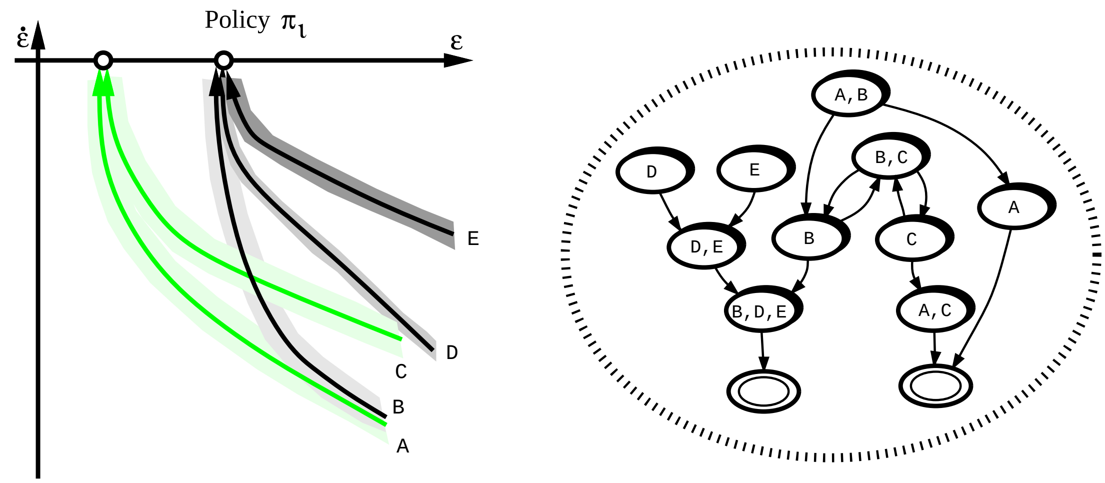


Figure 6.4. A hypothetical phase portrait of a native controller π_c (left) and all possible context transitions (right). (Reproduced with permission from Coelho et al. [24])

The quality of a grasp is characterized by the minimum friction coefficient μ_0 required for a null space to exist in the grip Jacobian G, with rank ≥ 1. For an ideal grasp, $\mu_0 = 0$, meaning that the grasp configuration is suitable for fixing the object in place using frictionless point contacts. Depending on the object shape and the number of fingers available, this ideal may not be achievable. The right panel in Figure 6.3 is annotated with the values of μ_0 associated with the three converged configurations.

6.3.2 The von Mises Distribution

Predicting hand orientations in relation to objects involves the estimation of relative angular information. Most conventional continuous probability distributions have a linear domain, for example, the normal distribution. In the case where a random variable represents an angle, one probability distribution that is often used in lieu of a normal distribution on a circular domain is the *von Mises* distribution [30]. It has the probability density function

$$p_{\mathrm{vM}}(\theta \mid \mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} \mathrm{e}^{\kappa \cos(\theta - \mu)} \tag{6.1}$$

where $0 \leq \mu < 2\pi$, $0 \leq \kappa < \infty$, and $I_0(\kappa)$ is the modified Bessel function of order zero. The mean direction of the distribution is given by μ, and κ is a concentration parameter with $\kappa = 0$ corresponding to a circular uniform distribution, and $\kappa \to \infty$ to a point distribution. Figure 6.5 illustrates four von Mises distributions with different parameters. The probability density at an angle corresponds to the radial distance from the unit circle to the density curve.

The parameter μ of the von Mises distribution is not to be confused with the friction coefficient μ_0 introduced in the previous section. I use this notation in agreement with established conventions. In this chapter, the use of μ is consistent: μ_0 (occasionally followed by one more index) always refers to the friction coefficient, and μ (often with an index other than zero) always denotes the mean direction of a von Mises distribution.

6.4 Feature Learning

In Coelho's haptically-guided grasping system, no visual cues guide the placement of the fingers on the object. Here, the goal is to learn visual features that predict successful hand orientations,

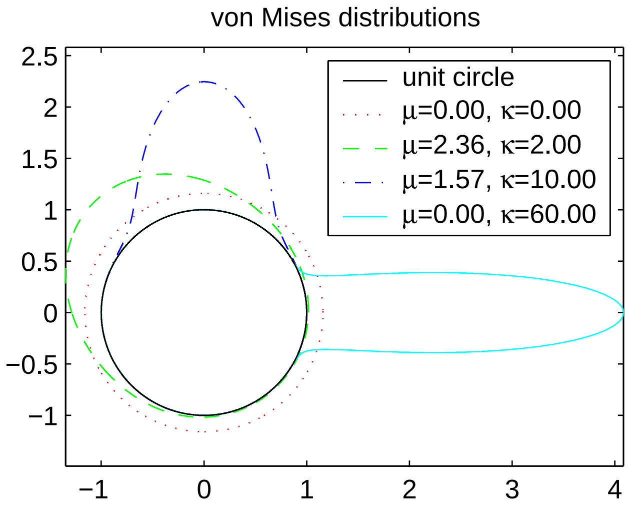


Figure 6.5. Illustration of four von Mises probability densities on a circular domain.

given prior haptic experience. These can be used to choose a two- or a three-fingered grasp controller, and to pre-shape the hand in order to place contacts reliably near high-quality grasp configurations for a task. In terms of Figure 6.4, this corresponds to initializing the haptic system in a context near the minimum achievable wrench residual. From here, a single native grasp controller can lead reliably to the preferred solution. In effect, pre-grasp visual cues are learned to subsume haptically-guided grasp policies.

The general scenario for the grasping problem is shown in Figure 6.6. The visual system attempts to identify features of the object that correlate with successful grasp parameters, including the relative hand orientation and the number of fingers used. An overhead view of the target object is obtained before the onset of the reach, and a prediction of useful grasp parameters is derived. Subsequently, the grasp is executed using the predicted grasp parameters. Upon convergence of the haptic controller to a final grasp configuration, grasp parameters are recorded and used for learning. The orientation of the hand during a given grasp configuration, θ_H, is defined as illustrated in Figure 6.7.

The robot may encounter a variety of objects that differ in their shapes. Each type of object may require a dedicated feature to recommend a hand orientation. Object identities are not known to the system; the need for dedicated features must be discovered by grasping experience. Visual features that respond selectively to haptic categories permit a recommendation to be made regarding the number of fingers to use for a grasp (Figure 6.8).

Features of the type defined in Chapter 3 are sampled from images taken by an overhead camera. Assuming that these features respond to the object itself, their image-plane orientation $\theta_{\mathbf{f}}$ (Equation 3.5, page 15) should be related to the azimuthal orientation θ_H of the robotic hand by a constant additive offset θ. A given feature, measured during many grasping experiences, generates data points that lie on straight lines on the toroidal surface spanned by the hand and feature orientations (Figure 6.9). Here, $\theta = \theta_H - \theta_{\mathbf{f}}$. There may be more than one straight line because a given visual feature may respond to more than one specific object location (e.g., due to object symmetries), or to several distinct objects that differ in shape. Given θ, one can then infer appropriate hand orientations as a function of observed feature orientations:

$$\theta_h = \theta_{\mathbf{f}} + \theta \tag{6.2}$$

The remaining problem is to find the offsets θ. Assuming these can be modeled as random variables with unimodal and symmetric probability distributions $p_k(\theta)$, with the k corresponding to the clusters of points in Figure 6.9, then this is an instance of a K-Means problem in one-

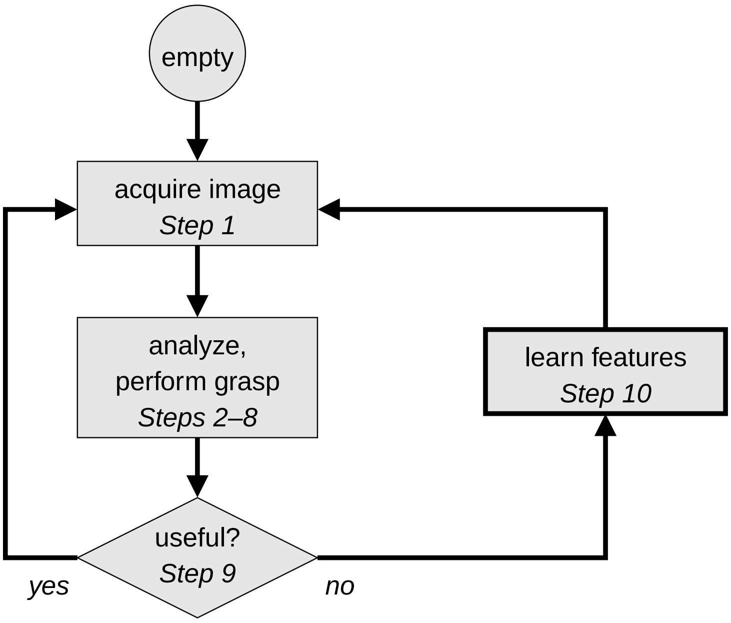


Figure 6.6. Scenario for learning features to recommend grasp parameters (cf. Figure 2.1, page 8). The step indices correspond to Algorithm 6.2, to be discussed later in this chapter.

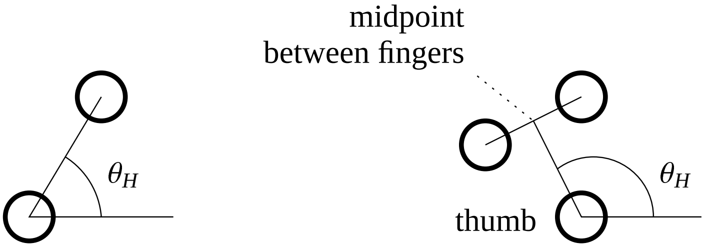


Figure 6.7. Definition of the hand orientation (azimuthal angle) for two- and three-fingered grasps.

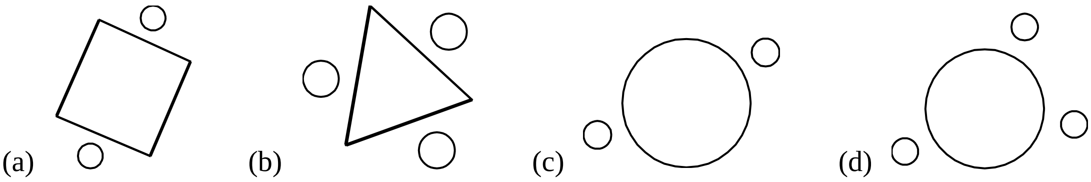


Figure 6.8. By Coelho's formulation, some objects are better grasped with two fingers (a), some with three (b), and for some this choice is unimportant (c, d).

dimensional circular (angular) space, with K unknown, and can be represented as a mixture distribution

$$p_{\text{mix}}(\theta) = \sum_{k=1}^{K} p_k(\theta)\, P(k) \tag{6.3}$$

with mixture proportions $0 < P(k) < 1$, $\sum_k P(k) = 1$. The following section shows how to find the parameters of this mixture distribution, which collectively constitute the *feature model* of **f** for a particular grasp controller. Subsequent sections describe how to select data points for fitting object-specific feature models, and how to use features to predict the quality of a grasp. Section 6.4.4 puts these pieces together and presents the full Algorithm 6.2 (cf. Figure 6.6) for learning multiple features with multiple grasp controllers, and addresses the interaction with the haptic grasping system.

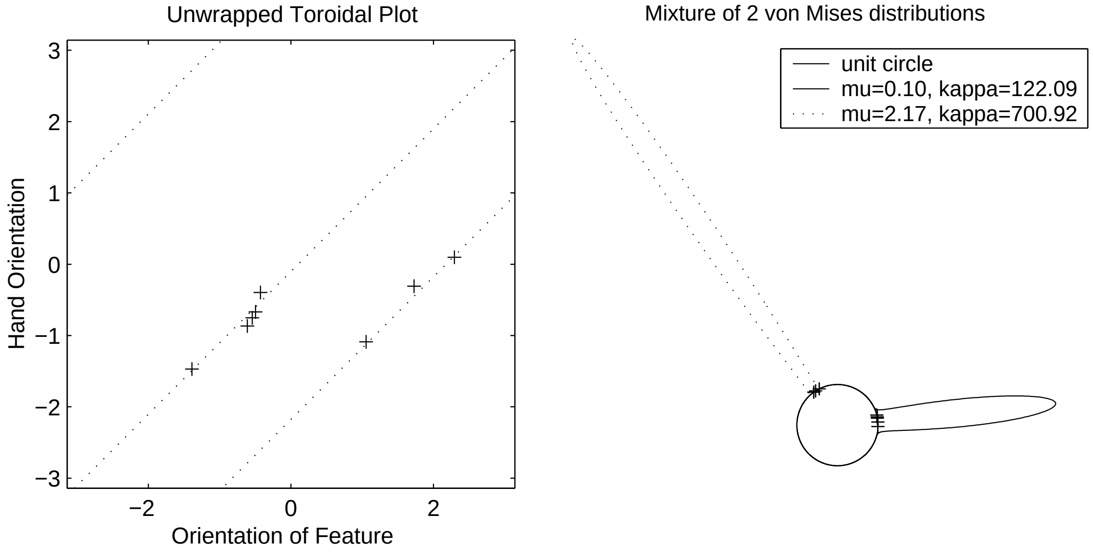


Figure 6.9. Left: Data points induced by a given feature on various images of an object form straight lines on a torus (two in this case); right: A mixture of two von Mises distributions was fitted to these data.

6.4.1 Fitting a Parametric Orientation Model

While the system learns and performs, all features are evaluated on all images. The response magnitudes $f_{\mathbf{f}}$ of all features **f**, their orientations $\theta_{\mathbf{f}}$, the actual hand orientations θ_H (the training signal), and the prediction errors $\Delta\theta_{\mathbf{f}} = |\theta_H - \theta_h|$ produced by each feature are stored in an *instance list*. To compute the mixture model for a feature, this feature's data points (such as those shown in Figure 6.9) are taken from this instance list.

Assume the θ are drawn independently from a mixture of von Mises distributions. The mixture distribution 6.3 (see Figure 6.9) then becomes

$$p_{\text{mix}}(\theta \mid \mathbf{a}) = \sum_{k=1}^{K} p_{\text{vM}}(\theta \mid \mu_k, \kappa_k)\, P(k) \tag{6.4}$$

where **a** is shorthand for the collection of parameters μ_k, κ_k and $P(k)$, $1 \le k \le K$.

For all plausible numbers of clusters K, a $(3K - 1)$-dimensional non-linear optimization problem is to be solved to find the μ_k, κ_k and $P(k)$. The appropriate number K of mixture components is then chosen as the one that maximizes the Integrated Completed Likelihood criterion [13], an adaptation to clustering problems of the more well-known Bayesian Information Criterion [106]. In practice, the method of choosing the right number of components is not critical, because as better features are learned, the clusters of data points will become increasingly well separated.

For a particular value of K, a maximum-a-posteriori (MAP) estimate of a mixture model is a parameterization $\mathbf{a}$ that maximizes the posterior probability given by Bayes' rule:

$$p(\mathbf{a} \mid \Theta) = \frac{L(\Theta \mid \mathbf{a})\, p(\mathbf{a})}{\sum_m L(\Theta \mid \mathbf{a}_m)\, p(\mathbf{a}_m)}$$

where $L(\Theta \mid \mathbf{a})$ is the likelihood of the observed data $\Theta = \{\theta_1, \ldots, \theta_N\}$. We now assume a uniform prior probability density over all possible model parameterizations $\mathbf{a}_m$. In this case, the MAP estimate is identical to the maximum-likelihood estimate that maximizes the log-likelihood of the data:

$$\log L(\Theta \mid \mathbf{a}) = \sum_{i=1}^{N} \log p_{\mathrm{mix}}(\theta_i \mid \mathbf{a}) \tag{6.5}$$

An elegant way to fit mixture models such as this is via the Expectation-Maximization (EM) algorithm. The EM algorithm is most practical if the model parameters μ_k, κ_k and $P(k)$ can be computed in closed form at the Expectation step. Unfortunately, this is not possible here because the Bessel functions cannot be inverted algebraically (see the Appendix). Facing this added difficulty, the mixture model (6.4) is more easily fitted directly, subject to the applicable constraints on the κ_k and $P(k)$, via gradient descent using the partial derivatives of the objective function (6.5):

$$\begin{aligned}
\frac{\partial}{\partial \mu_k} \log L(\Theta \mid \mathbf{a}) &= \sum_{i=1}^{N} \frac{P(k)}{p_{\mathrm{mix}}(\theta_i \mid \mathbf{a})} \frac{\partial}{\partial \mu_k} p_{\mathrm{vM}}(\theta_i \mid \mu_k, \kappa_k) \\
\frac{\partial}{\partial \kappa_k} \log L(\Theta \mid \mathbf{a}) &= \sum_{i=1}^{N} \frac{P(k)}{p_{\mathrm{mix}}(\theta_i \mid \mathbf{a})} \frac{\partial}{\partial \kappa_k} p_{\mathrm{vM}}(\theta_i \mid \mu_k, \kappa_k)
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial}{\partial \mu} p_{\mathrm{vM}}(\theta \mid \mu, \kappa) &= \kappa \sin(\theta - \mu)\, p_{\mathrm{vM}}(\theta \mid \mu, \kappa), & (6.6) \\
\frac{\partial}{\partial \kappa} p_{\mathrm{vM}}(\theta \mid \mu, \kappa) &= -\frac{I_{-1}(\kappa) + I_1(\kappa) - 2I_0(\kappa)\cos(\theta - \mu)}{2I_0(\kappa)}\, p_{\mathrm{vM}}(\theta \mid \mu, \kappa). & (6.7)
\end{aligned}$$

The $K-1$ partial derivatives with respect to $P(k)$ must take into account the constraint $\sum_{k=1}^{K} P(k) = 1$:

$$\begin{aligned}
&\frac{\partial}{\partial P(k)} \log L(\Theta \mid \mathbf{a}) \\
= \;& \frac{\partial}{\partial P(k)} \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K-1} p_{\mathrm{vM}}(\theta_i \mid \mu_k, \kappa_k) P(k) + p_{\mathrm{vM}}(\theta_i \mid \mu_K, \kappa_K) \left(1 - \sum_{k=1}^{K-1} P(k) \right) \right) \\
= \;& \sum_{i=1}^{N} \frac{1}{p_{\mathrm{mix}}(\theta_i \mid \mathbf{a})} \Big(p_{\mathrm{vM}}(\theta_i \mid \mu_k, \kappa_k) - p_{\mathrm{vM}}(\theta_i \mid \mu_K, \kappa_K) \Big), \qquad k = 1, \ldots, K-1.
\end{aligned}$$

Each feature is annotated with its model parameters, which include the μ_k, κ_k, $P(k)$, and the total number $N_{\mathbf{f}}$ of valid data points modeled by this mixture.

6.4.2 Selecting Object-Specific Data Points

If different types of objects are encountered, dedicated features may have to be learned. Without a supervisor providing object identities, the data collections (Figure 6.9) will be an indiscernible mix of feature responses corresponding to various objects, and any reliable patterns will be obscured. The key to learning dedicated features is to ignore data points corresponding to weak feature responses. This permits features to emerge that respond strongly only to specific, highly characteristic object parts, but that respond weakly in any image that does not contain such an object. These weak responses will be ignored, and reliable models of θ can be fitted to the strong responses.

Deciding whether a given point is “strong” in this sense involves a threshold. Such thresholds $\alpha_{\mathbf{f}}$, specific to each feature, can be determined optimally in the Bayesian sense that the number of poor recommendations made by the resulting model is minimized.[1] To do this for a given feature **f**, the history of feature responses $f_{\mathbf{f}}$ and the associated prediction errors $\Delta\theta_{\mathbf{f}}$ are analyzed in order to find a threshold $\alpha_{\mathbf{f}}$ such that most predictions for cases with $f_{\mathbf{f}} > \alpha_{\mathbf{f}}$ are correct. To formalize this intuitive notion, a global threshold $t_{\Delta\theta}$ is introduced, meaning that a prediction with $\Delta\theta_{\mathbf{f}} < t_{\Delta\theta}$ is correct, and false otherwise. The optimal threshold $\alpha_{\mathbf{f}}$ is then defined as a value that maximizes the Kolmogorov-Smirnoff distance $\mathrm{KSD}_{\mathbf{f}}$ between the two conditional distributions of $f_{\mathbf{f}}$ under the two conditions that the associated predictions are correct and false, respectively (Figure 6.10). The feature model of **f** is then fitted to all data points θ with $f_{\mathbf{f}} > \alpha_{\mathbf{f}}$. The threshold $t_{\Delta\theta}$ is a global run-time parameter that can gradually be reduced over time to encourage the learning of improved features.

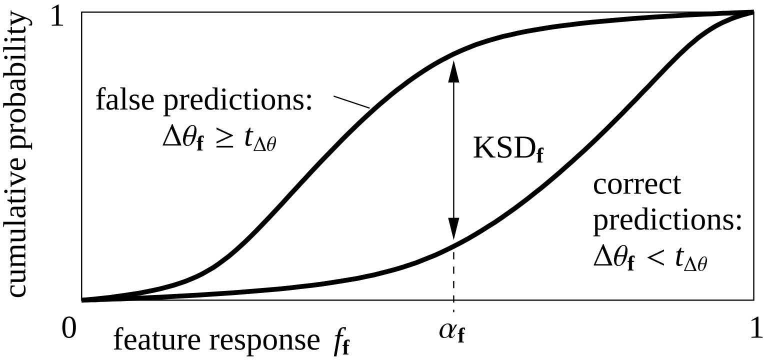


Figure 6.10. Kolmogorov-Smirnoff distance $\mathrm{KSD}_{\mathbf{f}}$ between the conditional distributions of feature response magnitudes given correct and false predictions.

6.4.3 Predicting the Quality of a Grasp

The quality of a grasp is defined by the minimum friction coefficient μ_0 required to execute the grasp using a given finger configuration (cf. Section 6.3.1). In this model, the lower (closer to zero) a value of μ_0, the better the grasp. It is not possible to separate good from poor grasps based on a generic threshold on μ_0 alone because, for any given object, the best achievable grasp depends on the object properties, the number of fingers, and – in general – on the end-to-end task. For example, the best achievable grasp of a triangular prism using two fingers is far worse than if three fingers are used. Under Coelho’s formulation, cubes are best grasped with two fingers to take advantage of symmetric contact forces across parallel opposing surfaces (Figure 6.8).

It is possible to estimate the expected value of μ_0 associated with a given feature **f**. For this purpose, the actually experienced value of μ_0 is stored in the instance list along with each

[1] Bayesian optimality holds only under the assumption that the prior probabilities of the two conditions are equal. See Sections 4.3.2 and 4.4.1 for a discussion.

executed grasp. These μ_0 values are regarded as samples of a continuous random variable M_0 with probability density function $p(\mu_0)$ and expected value

$$E[M_0] = \int_0^\infty \mu_0 \, p(\mu_0) \, \mathrm{d}\mu_0.$$

Observe that each measured value of μ_0 is associated with the measured prediction error $\Delta\theta_{\mathbf{f}} = |\theta_H - \theta_h|$, generated by the same grasp. If $\mathbf{f}$ is indeed highly predictive of the grasp outcome, then there should be an approximate proportional relationship between the μ_0 and the $\Delta\theta_{\mathbf{f}}$: Good grasps with small μ_0 should tend to exhibit a small angular error $\Delta\theta_{\mathbf{f}}$. Thus, the probability density function $p(\mu_0)$ should be well approximated by the distribution $p_{\mathrm{vM}}(\Delta\theta_{\mathbf{f}})$ of the angular offsets observed between the feature and hand orientations. Therefore, a sample estimate of $E[M_0(\mathbf{f})]$ for feature $\mathbf{f}$ can be approximated as

$$\hat{E}[M_0(\mathbf{f})] = \frac{\sum_i \mu_{0,i} \, p_{\mathrm{vM}}(\Delta\theta_{\mathbf{f},i})}{\sum_i p_{\mathrm{vM}}(\Delta\theta_{\mathbf{f},i})} \tag{6.8}$$

where the corresponding pairs of $\mu_{0,i}$ and $\Delta\theta_{\mathbf{f},i}$ are taken from the instance list, using only instances corresponding to occurrences of $\mathbf{f}$ with $f_{\mathbf{f},i} > \alpha_{\mathbf{f}}$.

6.4.4 Feature Learning Algorithm

Using the concepts introduced in the preceding sections, the procedure for fitting a feature model is summarized in the following algorithm.

Algorithm 6.1 (Fitting a Feature Model) For a given feature $\mathbf{f}$, the following steps are taken to compute the associated orientation model, consisting of the μ_k, κ_k, $\mathrm{KSD}_{\mathbf{f}}$, and $\hat{E}[M_0(\mathbf{f})]$:

1. Using the current value of $t_{\Delta\theta}$, identify those occurrences of $\mathbf{f}$ in the instance list that correspond to correct and false predictions. Using the response magnitudes f of these occurrences, compute $\mathrm{KSD}_{\mathbf{f}}$ and the associated threshold $\alpha_{\mathbf{f}}$ (Section 6.4.2).
2. Using the $N_{\mathbf{f}}$ of these instances with $f > \alpha_{\mathbf{f}}$, fit a parametric orientation model (Equation 6.4), obtaining $K_{\mathbf{f}}$, and values for μ_k, κ_k, and $P(k)$, $k = 1, \ldots, K_{\mathbf{f}}$ (Section 6.4.1).
3. Using this orientation model and the values of μ_0 and $\Delta\theta_{\mathbf{f}}$ corresponding to the measurements of $\mathbf{f}$ in the instance list, compute $\hat{E}[M_0(\mathbf{f})]$ (Equation 6.8, Section 6.4.3).

■

The system operates by recommending hand orientations and observing the outcomes of grasping procedures. New features are added to the repertoire in parallel to the execution of grasping tasks, and are evaluated over the course of many future grasps. This is in contrast to the distinction-learning application (Chapter 4), where the discriminative power of new features was immediately estimated using example images. Here, no example images are necessary. The visual system does not have any influence on the types or identities of the grasped objects. Two separate feature sets $\mathcal{F}_2$ and $\mathcal{F}_3$ and two separate instance lists are maintained for two- and three-fingered grasps. Out of these two sets, the feature with the lowest expected friction coefficient μ_0 provides a recommendation of grasp parameters to the haptic system. The operation of the learning system is detailed in the following algorithm.

Algorithm 6.2 (Feature Learning and Application of Feature Models) Initially, the sets $\mathcal{F}_2$ and $\mathcal{F}_3$ of features known to the system are empty.

1. A new grasp task is presented to the grasping system, and an overhead image of the grasped object is acquired.
2. The following steps are performed for each feature set $\mathcal{F}_c$:

(a) The responses $f_{\mathbf{f}}$ of all features $\mathbf{f} \in \mathcal{F}_c$ are measured in the current image.

(b) Identify the above-threshold feature with the highest predictive power:

$$\mathbf{f}_c^* = \operatorname*{argmax}_{\mathbf{f} \in \{\mathbf{f} \in \mathcal{F}_c | f_{\mathbf{f}} > \alpha_{\mathbf{f}}\}} \mathrm{KSD}_{\mathbf{f}}$$

Now, choose the number c of grasp contacts that will result in the better grasp:

$$c = \operatorname*{argmin}_{c \in \{2,3\}} \hat{E}[M_0(\mathbf{f}_c^*)]$$

3. From the K components in the angular mixture model associated with $\mathbf{f}_c^*$, identify the most concentrated component distribution of angular errors:

$$k^* = \operatorname*{argmax}_{k \in \left\{1 \leq k \leq K_{\mathbf{f}_c^*} \middle| N_{\mathbf{f}_c^*} P(k) \geq 3\right\}} \kappa_k$$

   To avoid singular modes, the maximum is only taken over those component distributions that model at least three data points.

4. Using the orientation of $\mathbf{f}_c^*$ measured in the image and the angular offset associated with k^*, form a recommended hand orientation (cf. Equation 6.2):

$$\theta_h = \theta_{\mathbf{f}_c^*} + \mu_{k^*}$$

5. The robot performs a c-contact grasp with initial azimuthal angle of θ_h. Upon convergence of the haptic system, a new instance vector is added to the instance list, containing the response magnitudes, orientations, and prediction errors $\Delta\theta_{\mathbf{f}} = |\theta_H - \theta_h|$ of all features $\mathbf{f} \in \mathcal{F}_c$, as well as the actual final hand orientation θ_H, and the achieved friction coefficient μ_0.
6. If $\Delta\theta_{\mathbf{f}_c^*} < t_{\Delta\theta}$, i.e. the recommended grasp parameters were accurate and required little adjustment by the haptic system, then continue at Step 1.
7. Using Algorithm 6.1, the $\mathrm{KSD}_{\mathbf{f}}$ are recomputed for all $\mathbf{f} \in \mathcal{F}_c$, and all associated mixture models are re-estimated based on the cases recorded in the instance list of previous experiences.
8. A new prediction θ_h is formed based on the new models, following the same procedure as above, but leaving c fixed (Steps 2a, 2b, 3, 4).
9. If now $|\theta_H - \theta_h| < t_{\Delta\theta}$, continue at Step 1.
10. Add two new features to $\mathcal{F}_c$: One is created by sampling a random texel or a random pair of edgels from salient points in the current image (cf. Algorithm 4.10 (Feature Generation, page 43), Step 2). The other feature is generated by randomly expanding an existing feature geometrically by adding a new point (cf. Algorithm 4.10, Step 3). Then, continue at Step 1.

■

Many of the features randomly sampled at Step 10 will perform poorly, e.g. because they respond to parts of the scene unrelated to the object to be grasped. Such features will develop a poor KSD, as there is no systematic association between their response strengths and their prediction accuracies. Due to their low KSD, such features will cease to be used at all (Step 2b). On the other hand, if a feature performs well, its KSD will be increased, and it will more likely be employed. Moreover, since only features are consulted that are asserted to be present in the image (i.e., $f_{\mathbf{f}} > \alpha_{\mathbf{f}}$), features can be learned that selectively respond to different object shapes requiring different offsets θ. Unused features should be discarded periodically. The details of when this is done are unimportant, but if all features are kept around, then those steps in Algorithm 6.2 that involve iterating over all features will incur unnecessary computational cost.

This algorithm uses only a single feature to derive a hand orientation. Similarly to the preceding two chapters, the reason for this choice is to reinforce the point that a highly uncommitted adaptive visual system can learn powerful features. The robustness of the recommended angles can probably be increased by pooling the recommendations of several features. At Step 2, the feature of choice is determined according to its $\mathrm{KSD}_{\mathbf{f}}$. Then, at Step 3, one of the modes k of this feature's model is chosen to form the actual prediction. At first glance, it seems that this choice of a model is unstable because the actual reliability of the individual modes of a given feature model may differ widely. However, this variation does not cause instability because the KSD is derived according to the actual behavior of the feature in practice, as recorded in the instance list. Thus, a high KSD can only result if the practically used modes k perform consistently well.

In contrast to the feature learning algorithm for distinctive features (Algorithm 4.8 on page 41), Algorithm 6.2 does not evaluate features immediately, as this would require repeated grasps of the same object in lieu of the example images used by Algorithm 4.8. This algorithm instead chooses to spread the evaluation of newly sampled features across multiple future grasps. This prevents the use of an explicit simple-to-complex search procedure such as Algorithm 4.10. However, the expansion of features at Step 10 above has the same effect, spread over multiple grasps. Boolean compound features are not used here because their orientations are less meaningful than those of geometric compounds.

Incidentally, this search for good predictive models constitutes an Expectation-Maximization (EM) algorithm. An EM algorithm alternates between estimating a set of unknown data values based on a current estimate of a model (Expectation step), and estimating the model based on the full data (Maximization step). Here, the parametric model to be optimized is the collection of all feature-specific models that are used by the angular recommendation process. The unknown data values specify which recorded data points should participate in the feature-specific models. At the Expectation step, these hidden parameters are estimated by computing the $\mathrm{KSD}_{\mathbf{f}}$ such that the probability of making the right choice for each data point is maximized, given the current model. At the Maximization step, the likelihood of the model given the participating data points is maximized by optimizing the model parameters according to Equation 6.5. As the system operates, these two steps alternate.

This instance of the EM algorithm is unusual in that the Expectation step does not utilize all available current data. Instead, the instance list of past experiences is consulted for previous prediction results, which were generated by models derived from all data available *at that time*. Taking the correct expectation using the most recent model would involve revisiting all previously-seen images at each Expectation step, which is clearly impractical. Nevertheless, the convergence properties of the EM algorithm are unaffected. As data accumulate, the accuracy of recent expectations can only increase, and the influence of possibly inaccurate data from early history diminishes.

6.5 Experiments

The learning algorithm just presented is designed to operate on-line. However, a single grasp performed by the real robot using Coelho's system (Section 6.3.1) takes on the order of ten minutes, while the learning algorithm requires hundreds of grasps to converge. Therefore, a thorough on-line evaluation was impractical. Instead, experiments were conducted in simulation in a way that closely resembles the actual integrated grasping system. These experiments utilized a detailed kinematic and dynamic simulator of the robotic hand/arm system, developed by Jefferson Coelho, that cycles through the following steps:

1. Generate a random polyhedral object geometry by drawing object parameters from parametric probability distributions derived from actual experiments using the real robot. Random parameters include the dimensions and angles defining the object.

2. Perform a simulated grasp of this object. The haptic control system is the same as that used for the real robot. The simulator interprets the motor commands and generates sensory feedback in the form of joint angles and (relatively noisy) force/torque readings of the fingertip sensors.

The three types of object geometries used included quadrangular (roughly square) and triangular (roughly equilateral) prisms, and cylinders. Typical object geometries and converged grasp configurations are illustrated in Figure 6.11. The ray originating from the object center indicates the wrist orientation in accordance with Figure 6.7. The categorization into “good” and “poor” grasps is for illustration only. As mentioned earlier, there is no way to grasp a triangle well using two fingers, and all grasps of the cylinder were of consistently high quality. Visual input for the visual learning system was generated by producing photo-realistically rendered and noise-degraded views on the basis of the object specifications generated at Step 1 above (Figure 6.12). All results reported below were computed in two-fold cross-validation. For simplicity and speed, no texels were used. However, other unpublished experiments indicate that the presence or absence of texels does not have a noticeable impact on this task.

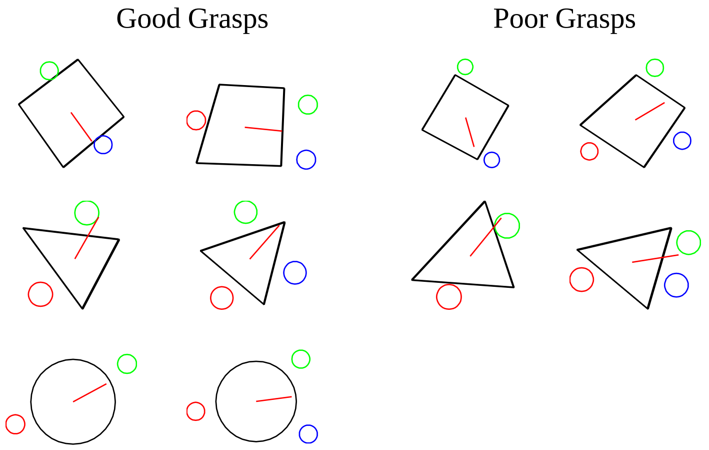


Figure 6.11. Representative examples of synthesized objects and converged simulated grasp configurations using Coelho’s haptically-guided grasping system.

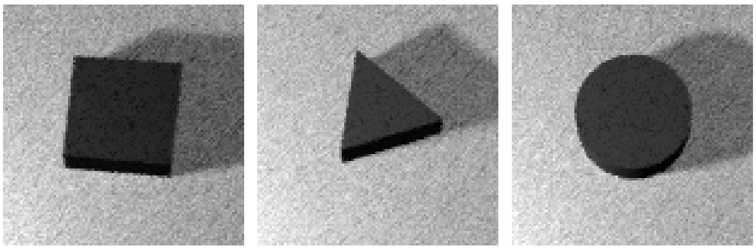
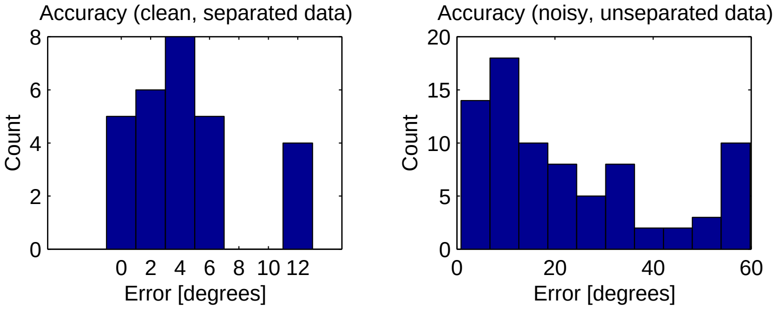


Figure 6.12. Example views of objects used in the grasping simulations.

6.5.1 Training Visual Models

To generate training data, Coelho’s grasp simulator performed several hundred simulated grasps on instances of the three object types, using two- and three-fingered grasp controllers, and recorded the final grasp configurations on convergence, along with the grasp quality index μ_0. These data were used by Algorithm 6.2 to train visual feature models. Lacking the ability to perform an actual grasp at Step 5, the recommended grasps were simulated by comparing the recommended hand orientation θ_h with the previously executed hand orientation θ_H associated with the training image, modulo the known rotational symmetry properties of the object. Since cylinders have infinite-fold rotational symmetry, any hand orientation works equally well. Consequently, no features were ever learned for cylinders.



(a) (b)

Figure 6.13. Quantitative results of hand orientation prediction. The rightmost bin of the right histogram includes all cases where the prediction error exceeded 60 degrees.

Figure 6.13 demonstrates that the system succeeds in learning features that are indicative of hand orientations. The data in Figure 6.13a were produced by feature models that were trained using 2-fold cross validation on the 20 best two-fingered grasps of cubes, and the 20 best three-fingered grasps of triangular prisms available in the simulated training set. As always in this chapter, the quality of a grasp was assessed in terms of the friction coefficient μ_0. These data can be expected to contain only little noise in the training signal, i.e., the actual hand orientation θ_H on grasp convergence. Performance on an independent test set is almost always excellent, with prediction error magnitudes on the order of the variation in the training signal.

The data in Figure 6.13b were produced by feature models that were trained on a representative sample of 80 grasps, including 20 grasps each of cubes and triangular prisms using two- and three-fingered grasp controllers. On such noisy training data that contains hand orientations that produced a poor grasp, the system expends a lot of effort trying to learn these outliers. However, performance degrades gracefully because features are selected by Kolmogorov-Smirnoff distance, which prefers generic features that work well for the majority of useful training examples. On a noisy test set, most poor recommendations occur on outliers. Notably, two-fingered grasps of the triangular object are inherently unstable and unpredictable. Here, prediction errors produced by the trained system depend on the error threshold that divides “good” from “poor” predictions during training. Choosing a low threshold generally produces more accurate predictions on a test set, as long as this threshold is larger than the variation contained in the majority of the training data.

6.5.2 Using Visual Models to Cue Haptic Grasping

Several experiments were performed to evaluate the effect of visual priming on the haptic grasping procedure. The two primary evaluation criteria were the number of haptic probes executed before the grasp controller converged, and the quality of the achieved grasp on convergence in terms of the friction coefficient μ_0. All experiments employ the native grasp controllers described in Section 6.3.1.

In the first experiment, a two-fingered native grasp controller was cued using features trained on the best 20 two-fingered grasps of cubes available in the training set. An analogous experi-

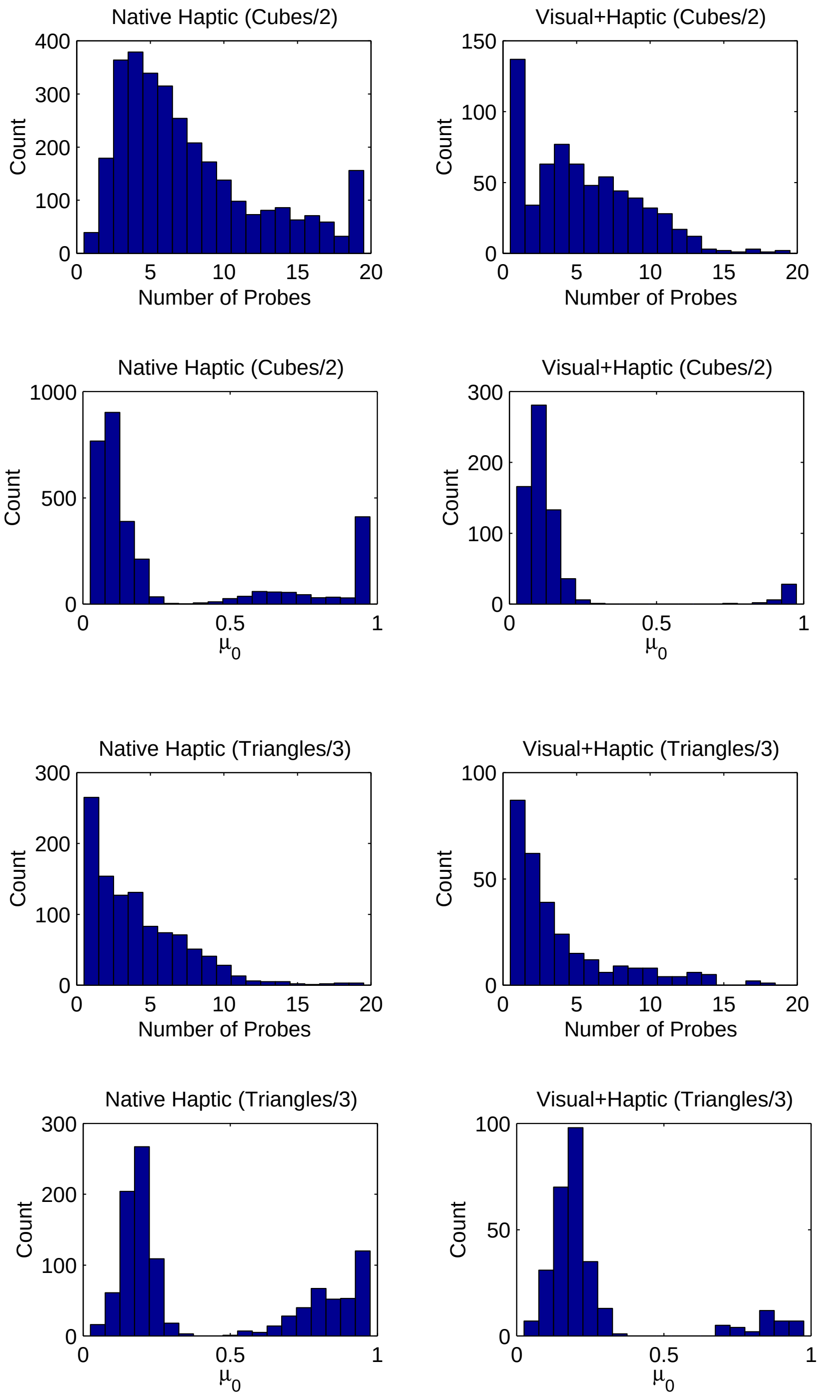


Figure 6.14. Results on two-fingered grasps of cubes (top half), and three-fingered grasps of triangular prisms (bottom half). The left column shows purely haptic grasps; in the right column, grasps were cued using learned visual feature models. The rightmost bin in each histogram includes all instances with a number of probes ≥ 20, or $\mu_0 \geq 1$, respectively.

ment was performed using three-fingered grasps of triangular prisms. The feature models were the same as those that generated the results shown in Figure 6.13a. The results are shown in Figure 6.14. For cubes, the number of lengthy grasps that required more than about 13 haptic probes was drastically reduced. Such grasps typically do not converge to a stable configuration. Likewise, the number of extremely fast grasps that required only a single probe increased substantially. For both cubes and triangular prisms, the expected number of probes was not significantly affected. However, in both cases the number of poor grasps (in terms of μ_0) was dramatically reduced. Figure 6.15 shows examples of the features learned during the training procedure. Interestingly, one feature captures both the cube and the boundary of its shadow. Apparently there were enough fitting examples in the data that such a feature was advantageous.

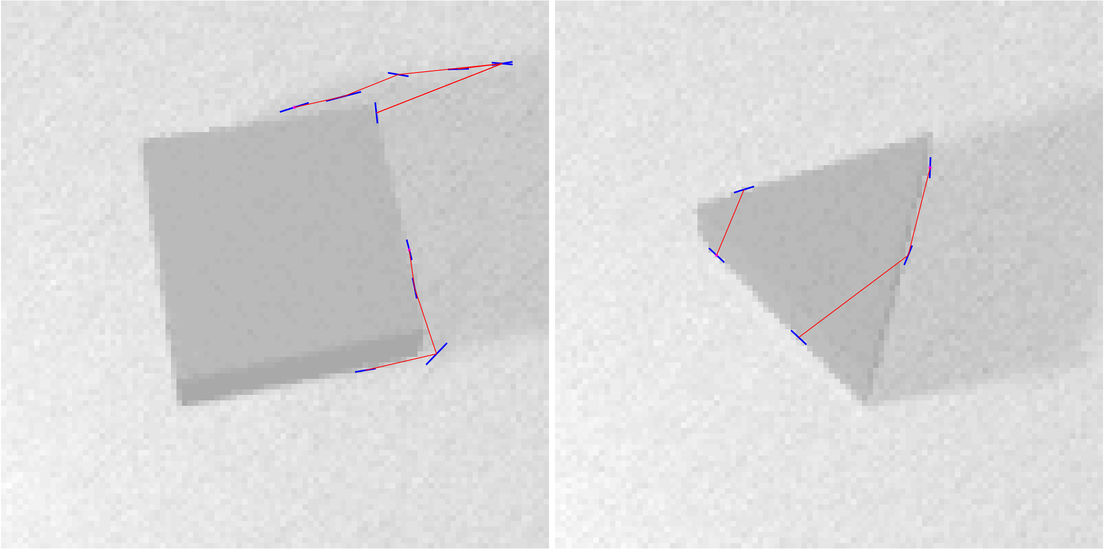


Figure 6.15. Examples of features obtained by training two-fingered models on cubes and three-fingered models on triangular prisms, using clean data separated by object type (cf. Figure 6.14).

In a second experiment, a set of visual feature models was trained using a training set of 80 grasps, including 20 grasps each of cubes and triangular prisms using two- and three-fingered grasp controllers. This training set differed from the one used in Figure 6.13b: There, the training set was representative of the population of grasp experiences, including outliers. Here, the best 20 grasps in each of the four categories was used for training. Both the two- and three-fingered visual feature models were exposed both to cubes and to triangular prisms, and the learning procedure learned features that were specific to cubes or triangular prisms exclusively. Thus, these features gathered meaningful statistics of the experienced friction coefficients, which could then be used to recommend a two- or a three-fingered grasp (see Section 6.4.3, and Algorithm 6.2, Step 2). This experiment was run in accordance with Algorithm 6.2, Steps 1–5. Figure 6.16 shows examples of the features learned. Note that in the case of the triangular prism, a feature spuriously responded to the shadow, which is not correlated with the orientation of the object.

The results are shown in Figure 6.17. The two columns on the left display the results achieved by the purely haptic system, for two- and three-contact native controllers. In both cases, grasped objects included both cubes and triangular prisms. This explains the bimodal distributions of μ_0 shown in the bottom row: The modes centered near $\mu_0 = 0.2$ mostly correspond to two-fingered grasps of cubes and three-fingered grasps of triangular prisms, while the modes centered near $\mu_0 = 0.6$ tend to correspond to three-fingered grasps of cubes and two-fingered grasps of triangular prisms. This illustrates that neither two- nor three-fingered

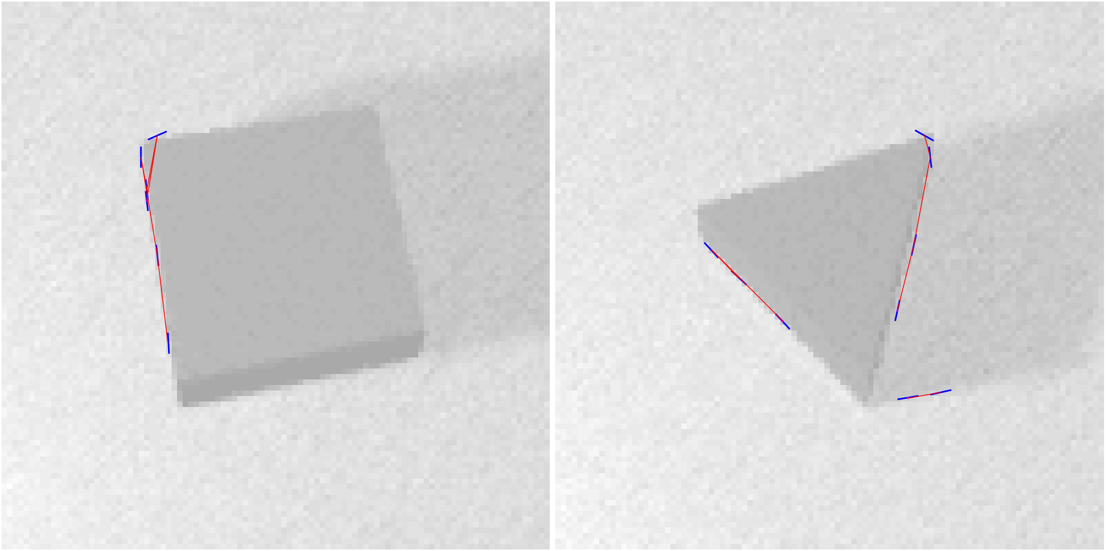


Figure 6.16. Examples of features obtained by training two- and three-fingered models on cubes and triangular prisms, using a single dataset containing relatively noise-free grasps of cubes and triangular prisms (cf. Figure 6.17).

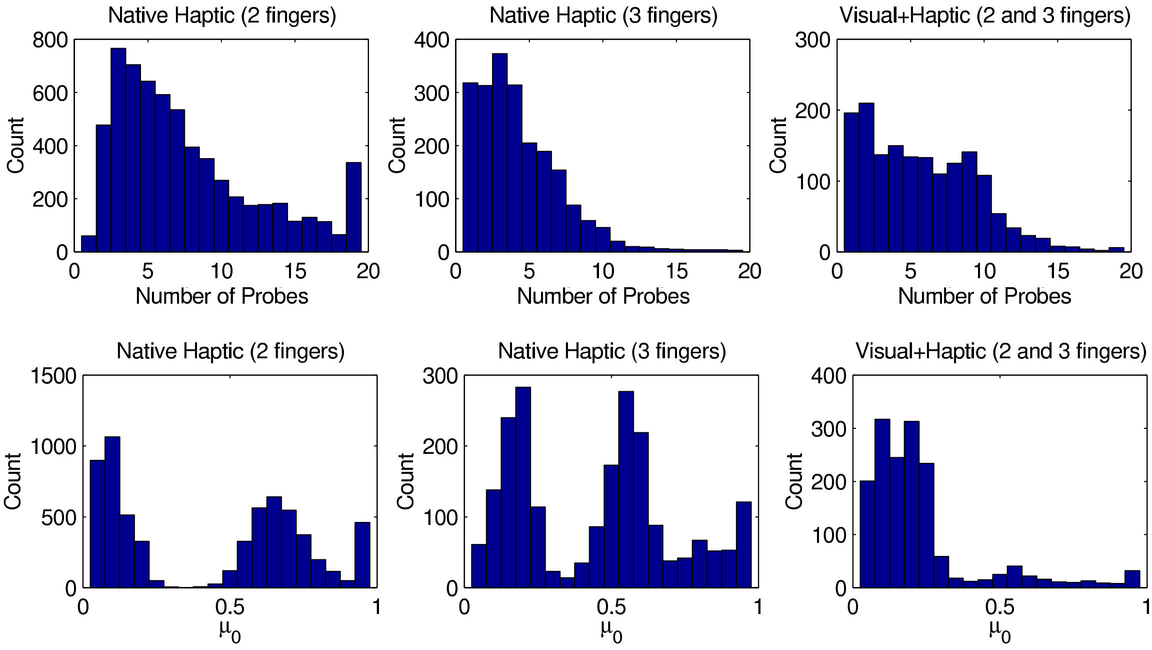


Figure 6.17. Grasp results on cubes and triangular prisms. The number of grasp contacts was chosen using the learned visual feature models.

native controllers alone are sufficient to execute high-quality grasps reliably for both cubes and triangular prisms.

The rightmost column shows the results achieved if the learned visual models determine which native policy to use, and how to orient the hand at the outset of a grasp. Almost all grasps result in a very good μ_0; the second mode has almost completely disappeared. Moreover, very long trials (more than about 20 probes) are practically eliminated (cf. the two-fingered native controller on the left). However, the expected number of probes increases slightly. The reasons for this undesired effect are unclear. One possible explanation is that the angular recommendations made by the learned feature models were relatively poor, and possibly skewed by a systematic bias, due to the presence of many unreliable grasps in the training set, predominantly two-fingered grasps of triangular prisms. Even if this is true, this slight drawback is more than outweighed by the enormous gain in the quality of the resulting grasps, as evidenced by the low values of μ_0.

6.6 Discussion

This chapter described a system for learning to recommend hand orientations and finger configurations to a haptically-guided grasping system. Localized appearance-based features of the visual scene are learned that correlate reliably with observed hand orientations. If the training data are not overly noisy, specialized features emerge that respond to distinct object classes. These can then be used to recommend the number of grasp contacts to be used, based on the expected quality of the grasp. In this way, visual guidance takes place without prior knowledge, explicit segmentation or geometric reconstruction of the scene. The interaction between haptic and visual system is a plausible model of human grasping behavior. Learning is on-line and incremental; there is no distinction between learning and execution phases. The results demonstrate the feasibility of the concept.

The main limitation of the system, as used in the experiments above, is that its performance was adversely affected by noise in the training data. It was unable to identify and ignore noisy training examples. However, it is important to note that this effect is inherent in the off-line nature of the training process. While the algorithm is designed to operate on-line, the experiments above were generated off-line using a fixed set of training data. In effect, the visual system learned solely by observing the haptic system in action. Without additional information, there was no way for it to distinguish good from poor grasp outcomes. Off-line training severs an important feedback loop between the visual and the haptic system. During on-line training, as described in Algorithm 6.2, the haptic system always uses the cues given by the visual system. If these cues are accurate, then the quality of the resulting grasps is of consistently high quality (Figures 6.14 and 6.17). In effect, *once highly predictive features emerge, the resulting training data will contain little noise!* Due to the KSD feature selection metric, good features will be used more often than poor features. Thus, their increased use will change the distribution of the training data, with the effect that the noise is increasingly reduced. This is a strong reason to expect that an on-line training procedure will bootstrap itself to yield reliable features by producing pure training data, which are in turn produced by reliable features. A merely observant learner is permanently confronted with the same stationary distribution of noisy training data, and wastes a lot of effort trying to learn the noise, contaminating its feature set with spurious features. In contrast, an interactive learner can actively eliminate the noise from the training data! The experimental verification of this argument must be left to future work.

The sensorimotor grasping system described in this chapter learns about objects, and how they are best grasped. This way of learning about activity afforded by the environment is reminiscent of the Gibsonian ecological approach to perception [41, 39]. The notion of *affordances* is a central concept in this theory. An affordance refers to the fit between an agent's capabilities and the environmental properties that make possible given actions. According to the

Gibsons, an agent learns about affordances by interacting with the environment. It discovers effects of actions and perceptual characteristics indicative of affordances. The ecological approach emphasizes that perception is *direct*; behaviorally useful information is extracted more or less directly from the egocentric sensory stream, without intermediate comprehensive representations such as geometric models. The Gibsonian theory is appealing to many roboticists because of its direct link between perception and action. Unfortunately, the underlying computational mechanisms are largely unclear. This is addressed by the methods described in this chapter. The haptic sensorimotor system learns about the types of grasps afforded by different objects, that is, the fit between hand and object. At the same time, the visual system learns to associate these affordances with visual appearance. Features of appearance are extracted directly from the sensory stream, which is compatible with the Gibsonian notion of direct perception. It will be interesting to explore such activity-driven learning mechanisms in more elaborate task scenarios.

CHAPTER 7

CONCLUSIONS

The first two chapters of this dissertation outlined a scenario in which autonomous robots refine their perceptual skills with growing experience, as demanded by their interaction with the environment. The following four technical chapters introduced a general framework for learning useful visual features, and applied it to two very different task domains. This final chapter collects the insights gained in the technical chapters, and reassembles them into parts of the big picture drawn at the outset. It discusses the contributions of this dissertation, and highlights some important directions for future research.

7.1 Summary of Contributions

The desire to construct autonomous robots that perform nontrivial tasks in the real world is one of the driving forces of research in artificial intelligence. It has long been argued that the world is too complex to allow the design of canned, robust action policies. Therefore, learning capabilities are generally agreed to be a crucial ingredient of sophisticated autonomous robots. However, the computer vision research community has not as widely embraced this viewpoint. In Section 1.2 I argued that visual learning is indispensable for agents that need to extract task-relevant information from uncontrolled environments. This dynamic and purposive paradigm of vision is commonly called *active vision* [4, 15]. The crucial role of visual learning in active vision has been pointed out in its early days [10, 9]. Since then however, relatively little emphasis has been placed on the learning aspects of task-driven vision, although some progress is being made [76]. A central aim of this dissertation is the advancement of visual learning as an essential ingredient of task-driven vision systems. The following items summarize the key contributions of this dissertation:

- The field of computer vision has a strong tradition of viewing vision problems in isolation. Under this paradigm, the goal of vision is to generate scene descriptions for use by higher-level reasoning processes. I argue that this approach is inappropriate in many visual tasks that arise in the context of autonomous systems, and instead advocate adaptive visual systems that learn to extract just the information needed for a task.

- I proposed an iterative, on-line paradigm for task-driven perceptual learning that cycles through sensation, action, and evaluation steps (Figures 2.1 on page 8, 4.2 on page 35, and 6.6 on page 102). Learning takes place only when needed, and is highly focused on solving problems encountered on line. This view of incremental visual learning departs from established traditions in computer vision.

- Extending Amit and Geman's work on combinatorial features [8, 5] and combining it with highly expressive and biologically plausible feature primitives, I introduced an infinite combinatorial and parametric feature space that contains useful features for a variety of tasks. Powerful features are sought in a simple-to-complex fashion by random sampling directly in training images.

- I demonstrated that powerful features can be learned in a task-driven manner by a general system that starts out with very little prior commitment to any particular task scenario. To reinforce this message, my algorithms are designed to learn only a small number of features that are highly distinctive individually.

- In the context of visual recognition, I generalized the conventional true/false dichotomy of classification by a multi-class framework that allows each class to be treated separately. This paradigm unifies the traditional paradigms of object recognition and detection. I also motivated an implementation on the basis of multiple Bayesian networks. This representation abandons the closed-world assumption made by conventional recognition systems.

- The efficacy of these algorithms and representations was evaluated on object discrimination tasks that simulated an on-line learning scenario.

- Motivated by the phenomenon of human expertise in visual recognition, I proposed a method for learning improved features by introducing explicit requirements of discriminative power. As a result, test-set performance significantly improves according to all performance metrics.

- On the basis of the feature space, I developed a method for learning features to recommend configuration parameters for a haptically-guided grasping system that resembles the way humans use visual information to pre-shape their hands while reaching for an object. The method results in improved grasps, without any explicit extraction or representation of object geometry.

Two very different applications demonstrated the potential of the general approach. The object discrimination task is essentially a *classification* problem, with the goal of finding features that are highly predictive of particular classes. A strongly focused feature search procedure explicitly applies simple-to-complex search heuristics. Candidate features are evaluated immediately, which requires a cooperative environment that can provide example images. The grasping task mainly constitutes a *regression* problem. Here, features are sought whose image-plane orientations robustly correlate with hand orientations. Feature evaluation is distributed over many successive grasping trials, and does not require example images. The simple-to-complex feature construction strategy is more implicit in this algorithm.

All experiments were performed in simulation. On-line learning tasks in real environments will benefit from an efficient implementation, especially if it capitalizes on the high degree of parallelism inherent in the feature search algorithms. Importantly, the algorithms presented here can in principle be applied in on-line, interactive tasks. This constitutes an advancement in the technology for building autonomous robots that perform on-line perceptual learning. The methodology begins with an uncommitted learning system that makes far weaker assumptions about the world and the task than most existing vision systems. In particular, no explicit assumptions are made regarding object segmentation, contour extraction, presence of clutter or occlusions, the number of target objects or classes present in a scene, or the number of classes to be learned. Nevertheless, target concepts are learned with remarkable accuracy, as reported in Chapters 5 and 6.

In addition to advancing technology, this dissertation also contributes to our understanding of human perceptual learning in that it provides a computational model of feature development. Sections 2.1 and 5.1 presented strong evidence that humans learn features not unlike those contained in the feature space introduced in Chapter 3. For the first time, a detailed computational model of feature learning was presented that explains many of the phenomena observed by psychologists [109, 107, 120, 38, 41, 39]. Key principles of the model are biologically plausible, including the primitive features and the principle of composition. It is not unlikely that humans perform a simple-to-complex feature search, producing empirically useful features that are discarded or refined as required by additional experience. Clearly, biological vision systems employ more sophisticated features not contained in the constrained space used here, as will be discussed in the following section.

7.2 Future Directions

The algorithms developed in this dissertation are far more general than the restricted scenarios used in the experiments. Much more systematic experimentation is required to illuminate their properties. The recognition system needs to be evaluated under the presence of multiple target objects, occlusions, and non-target clutter objects. Individual target classes should include different objects, including highly heterogeneous sets of objects that differ widely in appearance. Incremental introduction of target classes (in the sense of Section 5.4.4) requires further study, e.g. by examining the effect of presentation order on the learning process. Most importantly, both the recognition and the grasping system need to be integrated in real-world interactive tasks, breaking away from closed training sets. It was pointed out that both the expert learning and the grasping applications are expected to undergo a noticeable performance boost if training data are generated dynamically. Possible testbeds include humanoid or autonomous mobile robots, smart rooms, etc.

This dissertation explored two different paradigms of the general feature learning method. One was concerned with supervised classification, the other with regression. A third practical paradigm that should be explored involves an agent that learns multi-step reactive policies in a reinforcement learning framework. The agent takes policy *actions* based on the currently observed *state*. The only feedback available to the agent consists of a scalar *reward* that may be delayed over several policy actions, or may be given only at sparse intervals. In this framework, the state can be represented in terms of learned features that capture task-relevant information about the environment. For example, the agent can begin with an impoverished perceptual state space such that all world states are aliased. Then, the feature learning subsystem seeks perceptual features that resolve hidden state, similar in spirit to McCallum's U-Tree algorithm [67].

The following sections discuss important limitations and possible enhancements of the ideas and methods developed in this dissertation, and outline how these can be addressed by future research.

7.2.1 Primitive Features and their Composition

The two specific primitive feature types that form the basis of the feature space introduced in Chapter 3 were chosen because they constitute a small set and express complementary aspects of appearance. For general tasks, other types of primitive features should be introduced. Two of the most important features missing in the current system are color and statistical texture features. Gaussian-derivative features of color-opponent image signals have been explored for appearance-based object recognition [43]. Statistical features analyze the local intensity distribution without regard to their spatial organization, and thus complement the highly structured and oriented texels. For example, such features would probably prove valuable in the (gray-scale) Mel task.

An important characteristic of the feature space introduced in Chapter 3 is that it can be augmented almost arbitrarily by adding other types of primitive features and rules of composition. These additions are not limited to visual operators, but can include information from other perceptual modalities, or generally, any type of assertion about the world. Different modalities can even be combined within the same compound feature through appropriate compositional rules. For example, combining visual and haptic information using this framework would open up new perspectives in cross-modally-guided grasping.

The temporal and three-dimensional nature of the world gives rise to more features, many of which fit into this framework. For example, primitive or compound features can be computed on optical flow or disparity fields. These features are easily accessible to interactive agents, and are likely important in many everyday scenarios.

The current parametric representation of compound features is not very plausible biologically. For representations compatible with current knowledge about the organization of the visual brain, Riesenhuber and Poggio's model [96] can serve as a starting point.

7.2.2 Higher-Level Features

In its present form, the feature space contains primitive and compound features. This space excludes many types of features that are very informative to humans, but are not easily expressible within this framework. Examples include Gestalt features such as parallelism, symmetry (axial and radial), and continuity of contours. Another example is cardinality. To illustrate, a bicycle wheel has many spokes, and a triangle has three corners. These abstract concepts are not directly expressible by the current feature set, but appear to be very powerful cues to humans for scene interpretation. Some higher-level features may in principle be learnable by composing them from lower-level features.

7.2.3 Efficient Feature Search

The current feature learning algorithm engages a blind and random generate-and-test search procedure in static images. The only guidance available is given in the form of salient points, which constrain the search space to points that are deemed to be most important, independently of the task. In reality, there are much more powerful ways to draw attention to important features. For example, I postulate that the ability to track feature points of a moving object can give powerful cues about what appearance-based features are stable across a wide range of viewpoints. Also, such tracking allows the construction of (at least partially) view-invariant features of local three-dimensional object structure.

There may also be other, more direct ways to extract distinctive features, in the spirit of discriminative eigen-subspaces. I doubt that this is always possible, as even humans do not – at least not always – extract distinctive features directly in this way. On sufficiently difficult discrimination tasks, we struggle to identify distinctive features unless they are pointed out to us. There is strong psychological evidence suggesting that the subjective discriminability and organization of objects is altered by experience [107].

However, there is certainly much room for improvement on the current, blind generate-and-test method. One interesting possibility is to design a meta-learning process that notices patterns in the ways distinctive features are found, and thus learns how to look for good features efficiently.

7.2.4 Redundancy

Both mammalian and state-of-the-art feature-based computer vision systems owe their performance largely to their use of a large number of features [71, 104, 96, 77, 78, 8, 6, 82, 66]. The resulting redundancy or overcompleteness creates robustness to various kinds of image variation and degradation. The present work on feature learning purposely did not use this type of redundancy because without it, the distinctive power of individual features – and thus the success in learning – stand out much more. However, for practical applications such redundancy is required to increase robustness. Future work should investigate ways of learning large numbers of partially redundant features that are still powerful individually.

Recall from Chapter 5 that the trained recognition system rarely produces wrong results. Almost all incorrect recognitions are either ignorant or ambiguous, which are caused by a failure to find all relevant distinctive features in an image. This is an artifact of the extremely small number of features learned by the system. The availability of redundant features would greatly enhance the likelihood of finding distinctive features, which in turn would strongly reduce the number of ambiguous and ignorant recognitions. This makes me very confident that an extension of the current learning system with many redundant (but powerful) features would achieve results competitive with the best of today's, much more specialized recognition systems.

7.2.5 Integrating Visual Skills

Along with related work, this dissertation demonstrated some of the great potential of appearance-based vision. However, there are tasks that are not solved on the basis of appearance alone. There is a role for other techniques such as shape-from-X, geometric model matching, or a functional interpretation of items seen. Currently, such different paradigms are usually applied in isolation. We do not know how to integrate them into a unified whole, as the human visual system appears to do. Ideally, an autonomous robot should have at its disposal a repertoire of essential visual tools, and should learn how to use these jointly or individually.

7.3 Uncommitted Learning in Open Environments

The above discussion pointed out some features and rules for composition that should be added to extend the range of concepts expressible by the learning system. However, there are two important caveats: First, the combinatorics of the feature search grow badly with the number of primitives and rules (see Algorithm 4.10 on page 43). The second caveat is an instance of a critical problem shared by all inductive learning systems[1]: Even if unlimited computational resources were available to cope with arbitrary numbers of primitives and rules, having too many of these will necessarily hurt the learning process. The more degrees of freedom are available to an inductive learning system, the more ways exist to find patterns in the data. If the number of degrees of freedom is large in relation to the number of variables in the data, many spurious patterns will be found. There is no way to know which of these are valid in the sense that they will generalize to unseen test instances. Therefore, all inductive learners are necessarily limited in their expressive power. The guiding principles implementing this limitation are collectively called the *inductive bias* of the learner. In practice, the designer of a learning procedure must tailor the inductive bias to the presumed structure of the task.

A novelty of this work is the introduction of an infinite feature space based on commonly used visual feature primitives, expressive enough to learn training sets of almost any classification task – if necessary, by memorizing the training images. Without any guiding principles, this learning system would be bias free, and therefore unable to generalize. The bias is given by a simple-to-complex feature sampling procedure that makes the learner prefer *simple* features. The underlying assumption is that simple features generally provide for the best generalization, or in other words, that the "real" structure underlying a task tends to be expressible by simple features from the feature space (cf. Section 3.6, page 28).

This assumption is not always true, as exemplified by the Mel task in Chapters 4 and 5. The features found by the learner worked well on the training set, but generalized poorly nevertheless. It appears that the features represented accidental structure in the training set, which does not generalize to test images. In some cases, spurious structure represented by a feature may be supported by a large number of training images. In other cases, it may be supported by only very few images, resulting in overfitted features. Data sets that violate the structural assumptions reflected in the inductive bias are especially prone to this problem. This is apparently true

[1] Informally, an *inductive* learning system learns a function from training examples, and then generalizes to unseen test instances on the basis of this function.

of the Mel task, where simple combinations of edgels and texels did not tend to capture the real structure in the data.

How do humans solve the bias problem? Adult humans seem to be able to learn from very few examples, and are often able to pick out distinctive features with apparent ease. I suspect that the answer is related to our – learned – ability to control our learning bias. Guided by a wealth of background knowledge and functional understanding of the world, we are able to constrain our feature space adaptively and efficiently, so that relevant structure in the data is easily found. In other words, humans are not simple inductive learners, but our learning is supplemented by insights into functional and causal structure of the world. Likewise for artificial systems, the bias problem implies that straightforward inductive mechanisms are insufficient to build effective uncommitted learners; they must be supplemented by other mechanisms [124].

Notably, it seems plausible that this knowledge is itself acquired by inductive mechanisms. One can envision a hierarchy of inductive learners. Many learners learn specialized tasks on the basis of low-level features such as sensory signals. Their output signals constitute highly structured input signals to other learners, enabling these to learn functions that would be impossible to learn directly using the low-level features, because of their excessive degrees of freedom. These learning components can be trained in order, individually and largely independently, generally from the bottom up. This order can be given by a teacher who knows effective learning schedules. Where does the teacher come from? Imagine a world populated by a large number of such multilayer learning agents, all of which apply random learning schedules in an attempt to discover useful structure in the world. Because of their large number, some of them will eventually be successful. These can then become teachers of the others, sharing their empirically successful learning strategies.

This dissertation made progress toward building uncommitted perceptual systems that can learn to become expert at specific tasks. The next quantum leap toward understanding human learning on the one hand, and toward constructing autonomous artificial learning systems for open environments on the other hand, requires the defeat of the bias problem. This poses a great challenge to computer vision and machine learning research.

APPENDIX

EXPECTATION-MAXIMIZATION FOR VON MISES MIXTURES

The Expectation-Maximization (EM) algorithm [28] is an elegant class of methods for estimating the parameters **a** of a probabilistic model where only some of the underlying variables are known. The EM algorithm begins with any suitable initial estimate of the parameters **a**, and then alternates the following two steps until convergence:

Expectation: Given the current estimate of the model parameters **a** and the observed data, compute the expected values of the unobserved portion of the data.

Maximization: Given the observed data and the expected values of the unobserved data, compute a new set of model parameters **a** that maximizes some objective function.

If the objective function is continuous, this procedure will converge to a local maximum.

A typical application of EM is the estimation of the parameters of a mixture model

$$p_{\text{mix}}(\theta \mid \mathbf{a}) = \sum_{k=1}^{K} p(\theta \mid \mathbf{a}_k)\, P(k) \tag{A.1}$$

to fit an observed set Θ of data points θ_i, $i = 1, \ldots, N$. The mixing proportions $P(k)$ and the components k_i that generated each data point θ_i are unknown. The objective is to find the parameter vector $\mathbf{a}_k$ describing each component density $p(\theta \mid \mathbf{a}_k)$.

The Expectation step computes the probabilities $P(k \mid \theta_i)$ that each data point θ_i was generated by component k, given the current parameter estimates $\mathbf{a}_k$ and $P(k)$, using Bayes' Rule:

$$P(k \mid \theta_i) = \frac{p(\theta_i \mid \mathbf{a}_k)\, P(k)}{\sum_{j=1}^{K} p(\theta_i \mid \mathbf{a}_j)\, P(j)} = \frac{p(\theta_i \mid \mathbf{a}_k)\, P(k)}{p_{\text{mix}}(\theta \mid \mathbf{a})} \tag{A.2}$$

At the Maximization step, a new set of parameters $\mathbf{a}_k$, $k = 1, \ldots, K$, is computed to maximize the log-likelihood of the observed data:

$$\log L(\Theta \mid \mathbf{a}) = \sum_{i=1}^{N} \log p_{\text{mix}}(\theta \mid \mathbf{a}) \tag{A.3}$$

At the maximum, the partial derivatives with respect to all parameters vanish:

$$\begin{aligned} 0 \;=\; \frac{\partial}{\partial a_k} \log L(\Theta \mid \mathbf{a}) \;&=\; \sum_{i=1}^{N} \frac{P(k)}{p_{\text{mix}}(\theta \mid \mathbf{a})} \frac{\partial}{\partial a_k} p(\theta_i \mid \mathbf{a}_k) \\ &=\; \sum_{i=1}^{N} \frac{P(k \mid \theta_i)}{p(\theta_i \mid \mathbf{a}_k)} \frac{\partial}{\partial a_k} p(\theta_i \mid \mathbf{a}_k) \end{aligned} \tag{A.4}$$

where the second line (A.4) follows from substituting Equation A.2. The Maximization is then computed by solving this system (A.4) for all a_k. Moreover, the estimates of the component priors are updated by averaging the data-conditional component probabilities computed at the Expectation step:

$$P(k) = \frac{1}{N}\sum_{i=1}^{N} P(k \mid \theta_i) \tag{A.5}$$

For the particular case of a mixture of von Mises distributions, Equation A.4 is instantiated for the parameters μ and κ for each mixture component k. For μ_k, Equation A.4 becomes (cf. Equation 6.6, page 104)

$$\frac{\partial}{\partial\mu_k}\log L(\Theta \mid \mathbf{a}) \;=\; \kappa\sum_{i=1}^{N} P(k \mid \theta_i)\sin(\theta_i - \mu_k) \;=\; 0 \tag{A.6}$$

$$\hat{\mu}_k \;=\; \pm\sin^{-1}\frac{\sum_{i=1}^{N} P(k \mid \theta_i)\sin\theta_i}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} P(k \mid \theta_i)\,P(k \mid \theta_j)\cos(\theta_i - \theta_j)}} \tag{A.7}$$

Only one of the two solutions given by Equation A.7 is actually a root of Equation A.6. Which one this is can be determined by backsubstitution into Equation A.6. The root $\hat{\mu}_k$ thus determined either minimizes or maximizes the log-likelihood (A.3), as can be verified by consulting the second derivative:

$$\frac{\partial^2}{\partial^2\mu_k}\log L(\Theta \mid \mathbf{a}) \;=\; -\kappa\sum_{i=1}^{N} P(k \mid \theta_i)\cos(\theta_i - \mu_k) \tag{A.8}$$

If the second derivative (A.8) at $\hat{\mu}_k$ is less than zero, then $\mu_k = \hat{\mu}_k$ maximizes the log-likelihood (A.3); otherwise, $\hat{\mu}_k$ minimizes the log-likelihood, and its mirror image on the circle, $\mu_k = \hat{\mu}_k + \pi$, is to be used instead.

In the case of κ, the partial derivatives (cf. Equation 6.7)

$$\frac{\partial}{\partial\kappa_k}\log L(\Theta \mid \mathbf{a}) \;=\; -\sum_{i=1}^{N} P(k \mid \theta_i)\,\frac{I_{-1}(\kappa_k) + I_1(\kappa_k) - 2I_0(\kappa_k)\cos(\theta - \mu_k)}{2I_0(\kappa_k)} \;=\; 0$$

cannot be solved for κ_k in closed form because the modified Bessel functions are not invertible algebraically. The κ_k have to be approximated via numerical optimization, e.g. gradient descent. In other words, each iteration of the EM algorithm involves k one-dimensional numerical optimizations. In the face of this complexity, it seems simpler to perform a single $(3K - 1)$-dimensional numerical optimization to find the parameters μ_k, κ_k, and $P(k)$ of the mixture model (A.1) directly, instead of using the iterative EM algorithm.

BIBLIOGRAPHY

[1] Acredolo, L. P. Behavioral approaches to spatial orientation in infancy. In *The Development and Neural Bases of Higher Cognitive Functions*, A. Diamond, Ed., vol. 608. Annals of the New York Academy of Sciences, 1990, pp. 596–607.

[2] Acredolo, L. P., Adams, A., and Goodwyn, S. W. The role of self-produced movement and visual tracking in infant spatial orientation. *Journal of Experimental Child Psychology 38* (1984), 312–327.

[3] Allen, P. K. Surface descriptions from vision and touch. In *Proceedings of the 1984 Conference on Robotics* (Atlanta, GA, Mar. 1984), IEEE, pp. 394–397.

[4] Aloimonos, Y., Weiss, I., and Bandopadhay, A. Active vision. *International Journal of Computer Vision 1* (1988), 333–356.

[5] Amit, Y., and Geman, D. Shape quantization and recognition with randomized trees. *Neural Computation 9* (1997), 1545–1588.

[6] Amit, Y., and Geman, D. A computational model for visual selection. *Neural Computation 11*, 7 (Oct. 1999), 1691–1715.

[7] Amit, Y., Geman, D., and Jedynak, B. Efficient focusing and face detection. In *Face Recognition: From Theory to Applications*, H. Wechsler, P. Phillips, V. Bruce, F. F. Soulie, and T. Huang, Eds., NATO ASI Series F. Springer, Berlin, 1998.

[8] Amit, Y., Geman, D., and Wilder, K. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 11 (1997), 1300–1305.

[9] Bajcsy, R., and Campos, M. Active and exploratory perception. *Computer Vision, Graphics, and Image Processing: Image Understanding 56*, 1 (July 1992), 31–40.

[10] Ballard, D. H. Animate vision. *Artificial Intelligence 48* (1991), 57–86.

[11] Ballard, D. H., Hayhoe, M. M., Pook, P. K., and Rao, R. P. N. Deictic codes for the embodiment of cognition. *Behavioral and Brain Sciences 20*, 4 (Dec. 1995), 723–767.

[12] Banks, M. S., Aslin, R. N., and Letson, R. D. Sensitive period for the development of human binocular vision. *Science 190* (November 1975), 675–677.

[13] Biernacki, C., Celeux, G., and Govaert, G. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 7 (July 2000), 719–725.

[14] Black, M. J., and Jepson, A. D. EigenTracking: Robust matching and tracking of articulated objects using a view-based representation. In *European Conference on Computer Vision* (Cambridge, UK, 1996), pp. 329–342.

[15] Blake, A., and Yuille, A., Eds. *Active Vision.* MIT Press, Cambridge, Massachusetts, 1992.

[16] Califano, A., and Mohan, R. Multidimensional indexing for recognizing visual shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 4 (April 1994), 373–392.

[17] Canny, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 6 (1986), 679–698.

[18] Cho, K., and Dunn, S. M. Learning shape classes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 9 (1994), 882–888.

[19] Chomat, O., Colin de Verdière, V., Hall, D., and Crowley, J. L. Local scale selection for Gaussian based description techniques. In *European Conference on Computer Vision* (2000).

[20] Chomat, O., and Crowley, J. Probabilistic recognition of activity using local appearance. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '99)* (Ft. Collins, CO, June 1999), vol. 2, IEEE Computer Society, pp. 104–109.

[21] Coelho, Jr., J. A., and Grupen, R. A. A control basis for learning multifingered grasps. *Journal of Robotic Systems 14,* 7 (1997), 545–557.

[22] Coelho, Jr., J. A., and Grupen, R. A. Dynamic control models as state abstractions. In *NIPS'98 Workshop on Abstraction and Hierarchy in Reinforcement Learning* (Breckenridge, CO, 1998).

[23] Coelho, Jr., J. A., and Grupen, R. A. Learning in non-stationary conditions: A control theoretic approach. In *Proceedings of the Seventeenth International Conference on Machine Learning* (2000), Morgan Kaufmann.

[24] Coelho, Jr., J. A., Piater, J. H., and Grupen, R. A. Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. In *First IEEE-RAS International Conference on Humanoid Robots* (2000).

[25] Cohen, P. R., Atkin, M. S., Oates, T., and Beal, C. R. Neo: Learning conceptual knowledge by sensorimotor interaction with an environment. In *Proceedings of the First International Conference on Autonomous Agents* (1997), pp. 170–177.

[26] Colin de Verdière, V., and Crowley, J. L. Visual recognition using local appearance. In *Fifth European Conference on Computer Vision* (June 1998), vol. 1 of *Lecture Notes in Computer Science,* Springer, pp. 640–654.

[27] de Jong, E. D., and Steels, L. Generation and selection of sensory channels. In *Evolutionary Image Analysis, Signal Processing and Telecommunications First European Workshops, EvoIASP'99 and EuroEcTel'99 Joint Proceedings* (Berlin, May 1999), LNCS, Springer-Verlag, pp. 90–100.

[28] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society (Series B) 39,* 1 (1977), 1–38.

[29] Draper, B. A., Bins, J., and Baek, K. ADORE: Adaptive object recognition. *Videre 1,* 4 (2000), 86–99.

[30] Fisher, N. I. *Statistical Analysis of Circular Data.* Cambridge University Press, 1993.

[31] Fisher, R. A. *Contributions to Mathematical Statistics.* John Wiley, New York, 1950.

[32] Fisher, R. B., and MacKirdy, A. Integrating iconic and structured matching. In *European Conference on Computer Vision* (1998).

[33] Fleuret, F., and Geman, D. Graded learning for object detection. Tech. rep., Department of Mathematics and Statistics, University of Massachusetts, Amherst, Feb. 1999.

[34] Florack, L. M. J. *The Syntactic Structure of Scalar Images*. PhD thesis, University of Utrecht, 1993.

[35] Freeman, W. T., and Adelson, E. H. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 9 (1991), 891–906.

[36] Friedman, J. H., and Tukey, J. W. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers 23* (1974), 881–890.

[37] Fukunaga, K., and Koontz, W. L. G. Applications of the Karhunen-Loeve expansion to feature selection and ordering. *IEEE Transactions on Computers C-19* (1970), 917–923.

[38] Gauthier, I., and Tarr, M. J. Becoming a "Greeble" expert: Exploring mechanisms for face recognition. *Vision Research 37*, 12 (1997), 1673–1682.

[39] Gibson, E. J., and Pick, A. *An Ecological Approach to Perceptual Learning and Development.* Oxford University Press, in press.

[40] Gibson, E. J., and Spelke, E. S. The development of perception. In *Handbook of Child Psychology Vol. III: Cognitive Development,* J. H. Flavell and E. M. Markman, Eds., 4th ed. Wiley, 1983, ch. 1, pp. 2–76.

[41] Gibson, J. J. *The Ecological Approach to Visual Perception.* Houghton Mifflin, Boston, 1979.

[42] Grimson, W. E. L., and Lozano-Pérez, T. Model-based recognition and localization from tactile data. *Journal of Robotics Research 3*, 3 (1986), 3–35.

[43] Hall, D., Colin de Verdière, V., and Crowley, J. L. Object recognition using coloured receptive fields. In *European Conference on Computer Vision* (2000).

[44] Held, R. Binocular vision – behavioral and neuronal development. In *Neonate Cognition: Beyond the Blooming Buzzing Confusion,* J. Mehler and R. Fox, Eds. Lawrence Erlbaum Associates, 1985, pp. 37–44. Reprinted in [53], pp. 152–158.

[45] Hubel, D. H., and Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology (London) 160* (1962), 106–154.

[46] Huttenlocher, D. P., and Ullman, S. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision 5*, 2 (1990), 195–212.

[47] Intrator, N. Feature extraction using an unsupervised network. *Neural Computation 4* (1992), 98–107.

[48] Intrator, N., and Gold, J. I. Three-dimensional object recognition using an unsupervised BCM network: The usefulness of distinguishing features. *Neural Computation 5* (1993), 61–74.

[49] Jägersand, M., and Nelson, R. Visual space task specification, planning and control. In *Proc. IEEE Symposium on Computer Vision* (1995), pp. 521–526.

[50] Jedynak, B., and Fleuret, F. Reconnaissance d'objets 3D à l'aide d'arbres de classification. In *Image'Com '96* (Bordeaux, France, 20–22 May 1996).

[51] Jensen, F. V. *An introduction to Bayesian networks.* Springer, New York, May 1996.

[52] John, G. H., Kohavi, R., and Pfleger, K. Irrelevant features and the subset selction problem. In *Proceedings of the Eleventh International Conference on Machine Learning* (San Francisco, 1994), W. W. Cohen and H. Hirsh, Eds., Morgan Kaufmann.

[53] Johnson, M. H., Ed. *Brain Development and Cognition.* Bracewell, New York, 1993.

[54] Kandel, E. R., Schwartz, J. H., and Jessell, T. M., Eds. *Essentials of Neural Science and Behavior.* Appleton & Lange, Stamford, Connecticut, 1995.

[55] Knoll, T. F., and Jain, R. C. Recognizing partially visible objects using feature indexed hypotheses. *IEEE Journal of Robotics and Automation RA-2*, 1 (March 1986), 3–13.

[56] Koenderink, J. J. The structure of images. *Biological Cybernetics 50* (1984), 363–370.

[57] Koenderink, J. J., and van Doorn, A. J. Representation of local geometry in the visual system. *Biological Cybernetics 55* (1987), 367–375.

[58] Krüger, N., and Lüdtke, N. ORASSYLL: Object recognition with autonomously learned and sparse symbolic representations based on local line detectors. In *British Machine Vision Conference* (1998).

[59] Kruskal, J. B. Linear transformation of multivariate data to reveal clustering. In *Multidimensional Scaling: Theory and Application in the Behavioural Sciences*, R. N. Shephard, A. K. Romney, and S. K. Nerlove, Eds. Seminar Press, New York, 1972, pp. 179–191.

[60] Lauritzen, S. L. Propagation of probabilities, means, and variances in mixed graphical models. *Journal of the American Statistical Association (Theory and Methods) 87*, 420 (1992), 1098–1108.

[61] Leonardis, A., and Bischof, H. Robust recognition using eigenimages. *Computer Vision and Image Understanding 78*, 1 (Apr. 2000), 99–118.

[62] Lindeberg, T. *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, Dordrecht, Netherlands, 1994.

[63] Lindeberg, T. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision 30*, 2 (1998), 117–154.

[64] Lindeberg, T. Feature detection with automatic scale selection. *International Journal of Computer Vision 30*, 2 (1998), 77–116.

[65] Logothetis, N. K., and Pauls, J. Psychophysical and physiological evidence for viewer-centered object representation in the primate. *Cerebral Cortex 3* (1995), 270–288.

[66] Lowe, D. G. Towards a computational model for object recognition in IT cortex. In *Proceedings of the IEEE International Workshop on Biologically Motivated Computer Vision* (May 2000), S.-W. Lee, H. H. Bülthoff, and T. Poggio, Eds., vol. 1811 of *LNCS*, Springer-Verlag, pp. 20–31.

[67] McCallum, A. K. *Reinforcement Learning with Selective Perception and Hidden State.* PhD thesis, Department of Computer Science, University of Rochester, Rochester, NY, 1995. Revised 1996.

[68] McCallum, R. A. Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning* (1993), Morgan Kaufmann.

[69] McCallum, R. A. Utile suffix memory for reinforcement learning with hidden state. Tech. Rep. 549, Dept. of Computer Science, University of Rochester, Rochester, NY, Dec. 1994.

[70] McCarty, M. E., Clifton, R. K., Ashmead, D. H., Lee, P., and Goubet, N. How infants use vision for grasping objects. *Child Development* (in press).

[71] Mel, B. W. SEEMORE: Combining color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural Computation 9* (1997), 777–804.

[72] Moghaddam, B., and Pentland, A. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 7 (July 1997), 696–710.

[73] Murase, H., and Nayar, S. K. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision 14* (1995), 5–24.

[74] Nayar, S. K., Murase, H., and Nene, S. A. Learning, positioning, and tracking visual appearance. In *Proceedings of the International Conference on Robotics and Automation* (San Diego, CA, May 1994), IEEE.

[75] Nayar, S. K., Nene, S. A., and Murase, H. Subspace methods for robot vision. *IEEE Transactions on Robotics and Automation 12*, 5 (Oct. 1996), 750–758.

[76] Nayar, S. K., and Poggio, T., Eds. *Early Visual Learning.* Oxford University Press, Feb. 1996.

[77] Nelson, R. C., and Selinger, A. A cubist approach to object recognition. In *Proceedings of the International Conference on Computer Vision* (1998).

[78] Nelson, R. C., and Selinger, A. Large-scale tests of a keyed, appearance-based 3-D object recognition system. *Vision Research 38*, 15–16 (1998).

[79] Nene, S. A., Nayar, S. K., and Murase, H. Columbia object image library (COIL-20). Tech. Rep. CUCS-005-96, Columbia University, New York, Feb. 1996.

[80] Olshausen, B. A., and Field, D. J. Emergence of simple-cell receptive-field properties by learning a sparse code for natural images. *Nature 381* (1996), 607–609.

[81] Papageorgiou, C. P., Oren, M., and Poggio, T. A general framework for object detection. In *Proceedings of the International Conference on Computer Vision* (Jan. 1998).

[82] Papageorgiou, C. P., and Poggio, T. A pattern classification approach to dynamical object recognition. In *Proceedings of the International Conference on Computer Vision* (Corfu, Greece, Sept. 1999).

[83] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Francisco, 1988.

[84] Piater, J. H., and Grupen, R. A. A framework for learning visual discrimination. In *Proceedings of the Twelfth International FLAIRS Conference* (1999), Florida Artificial Intelligence Research Society, AAAI Press, pp. 84–88.

[85] Piater, J. H., and Grupen, R. A. Learning visual recognition with Bayesian networks. Computer Science Technical Report 99-43, University of Massachusetts, Amherst, MA, Mar. 1999.

[86] Piater, J. H., and Grupen, R. A. Toward learning visual discrimination strategies. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (June 1999), vol. 1, IEEE Computer Society, pp. 410–415.

[87] Piater, J. H., and Grupen, R. A. Constructive feature learning and the development of visual expertise. In *Proceedings of the Seventeenth International Conference on Machine Learning* (2000), Morgan Kaufmann, pp. 751–758.

[88] Piater, J. H., and Grupen, R. A. Distinctive features should be learned. In *Proceedings of the IEEE International Workshop on Biologically Motivated Computer Vision* (May 2000), S.-W. Lee, H. H. Bülthoff, and T. Poggio, Eds., vol. 1811 of *LNCS*, Springer-Verlag, pp. 52–61.

[89] Piater, J. H., and Grupen, R. A. Feature learning for recognition with Bayesian networks. In *Proceedings of the International Conference on Pattern Recognition* (Sept. 2000).

[90] Rao, R. P. N., and Ballard, D. H. An active vision architecture based on iconic representations. *Artificial Intelligence 78* (1995), 461–505.

[91] Rao, R. P. N., and Ballard, D. H. Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation 9*, 4 (May 1997), 721–763.

[92] Ravela, S., and Manmatha, R. Retrieving images by appearance. In *Proceedings of the International Conference on Computer Vision* (1998), pp. 608–613.

[93] Ravela, S., and Manmatha, R. Gaussian filtered representations of images. In *The Encyclopedia on Electrical & Electronics Engineering*, J. G. Webster, Ed. John Wiley & Sons, 1999, pp. 289–307.

[94] Ravela, S., Manmatha, R., and Riseman, E. M. Image retrieval using scale space matching. In *Fourth European Conference on Computer Vision* (Cambridge, U.K., April 1996), Springer, pp. 273–282.

[95] Rider, E. A., and Rieser, J. J. Pointing at objects in other rooms: Young children's sensitivity to perspective after walking with and without vision. *Child Development 59* (1988), 480–494.

[96] Riesenhuber, M., and Poggio, T. Hierarchical models of object recognition in cortex. *Nature Neuroscience 2* (1999), 1019–1025.

[97] Rieser, J. J., Garing, A. E., and Young, M. F. Imagery, action, and young children's spatial orientation: It's not being there that counts, it's what one has in mind. *Child Development 65* (1994), 1262–1278.

[98] Rimey, R. D., and Brown, C. M. Task-oriented vision with multiple Bayes nets. In *Active Vision*, A. Blake and A. Yuille, Eds. MIT Press, Cambridge, Massachusetts, 1992, ch. 14, pp. 217–236.

[99] Ripley, B. D. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[100] Riseman, E. M., Hanson, A. R., Beveridge, J. R., Kumar, R., and Sawhney, H. Landmark-based navigation and the acquisition of environmental models. In *Visual Navigation: From Biological Systems to Unmanned Ground Vehicles*, Y. Aloimonos, Ed. Lawrence Erlbaum Associates, 1997, ch. 11, pp. 317–374.

[101] Ruff, H. A. Infant recognition of the invariant form of objects. *Child Development 49* (1978), 293–306.

[102] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Upper Saddle River, NJ, 1995.

[103] Salapatek, P., and Banks, M. S. Infant sensory assessment: Vision. In *Communicative and Cognitive Abilities – Early Behavioral Assessment*, F. D. Minifie and L. L. Lloyd, Eds. University Park Press, Baltimore, 1978.

[104] Schiele, B., and Crowley, J. L. Object recognition using multidimensional receptive field histograms. In *Fourth European Conference on Computer Vision* (Cambridge, UK, Apr. 1996).

[105] Schnackertz, T. J., and Grupen, R. A. A control basis for visual servoing tasks. In *Proceedings of the International Conference on Robotics and Automation* (1995), IEEE.

[106] Schwarz, G. Estimating the dimensions of a model. *Annals of Statistics 6* (1978), 461–464.

[107] Schyns, P. G., Goldstone, R. L., and Thibaut, J.-P. The development of features in object concepts. *Behavioral and Brain Sciences 21*, 1 (1998), 1–54.

[108] Schyns, P. G., and Murphy, G. L. The ontogeny of part representation in object concepts. In *The Psychology of Learning and Motivation*, Medin, Ed., vol. 31. Academic Press, San Diego, CA, 1994, pp. 305–354.

[109] Schyns, P. G., and Rodet, L. Categorization creates functional features. *Journal of Experimental Psychology: Learning, Memory, and Cognition 23*, 3 (1997), 681–696.

[110] Segen, J. Learning graph models of shape. In *Proceedings of the 5th International Conference on Machine Learning* (Ann Arbor, June 12–14 1988), J. Laird, Ed., Morgan Kaufmann, pp. 29–35.

[111] Shams, S. Multiple elastic modules for visual pattern recognition. *Neural Networks 8*, 9 (1995), 1439–1456.

[112] Shannon, C. E., and Weaver, W. *The mathematical theory of communication.* University of Illinois Press, Urbana, 1949.

[113] Solso, R. L., and McCarthy, J. E. Prototype formation of faces: A case of pseudo-memory. *British Journal of Psychology 72* (1981), 499–503.

[114] Steels, L. Constructing and sharing perceptual distinctions. In *Proceedings of the European Conference on Machine Learning* (Berlin, 1997), M. van Someren and G. Widmer, Eds., Springer-Verlag.

[115] Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, Massachusetts, 1998.

[116] Swain, M. J., and Ballard, D. H. Color indexing. *International Journal of Computer Vision 7*, 1 (1991), 11–32.

[117] Swets, D. L., and Weng, J. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 8 (Aug. 1996), 831–836.

[118] Talukder, A., and Casasent, D. Classification and pose estimation of objects using nonlinear features. In *Applications and Science of Computational Intelligence* (1998), vol. 3390, SPIE.

[119] Talukder, A., and Casasent, D. A general methodology for simultaneous representation and discrimination of multiple object classes. *Optical Engineering* (Mar. 1998). Special issue on Advances in Recognition Techniques.

[120] Tanaka, J. W., and Taylor, M. Object categories and expertise: Is the basic level in the eye of the beholder? *Cognitive Psychology 23* (1991), 457–482.

[121] Tarr, M. J. Visual pattern recognition. In *Encyclopedia of Psychology*, A. Kazdin, Ed. American Psychological Association, Washington, DC, in press.

[122] Tarr, M. J., and Bülthoff, H. H. Is human object recognition better described by geon-structural-descriptions or by multiple-views? *Journal of Experimental Psychology: Human Perception and Performance 21*, 6 (1995), 1494–1505.

[123] Thibaut, J.-P. The development of features in children and adults: The case of visual stimuli. In *Proc. 17th Annual Meeting of the Cognitive Science Society* (1995), Lawrence Erlbaum, pp. 194–199.

[124] Thrun, S., and Mitchell, T. M. Lifelong robot learning. *Robotics and Autonomous Systems 15* (1995), 25–46.

[125] Turk, M., and Pentland, A. Eigenfaces for recognition. *Journal of Cognitive Neuroscience 3*, 1 (1991), 71–86.

[126] Utgoff, P. E., and Clouse, J. A. A Kolmogorov-Smirnoff metric for decision tree induction. Computer Science Technical Report 96-3, University of Massachusetts, Amherst, 1996.

[127] van Vliet, L. J., Young, I. T., and Verbeek, P. W. Recursive Gaussian derivative filters. In *Proc. 14th International Conference on Pattern Recognition (ICPR'98)* (Brisbane, Australia, Aug. 1998), vol. 1, IEEE Computer Society Press, pp. 509–514.

[128] Viola, P. A. Complex feature recognition: A Bayesian approach for learning to recognize objects. AI Memo 1591, Massachusetts Institute of Technology, November 1996.

[129] Wallis, G., and Bülthoff, H. Learning to recognize objects. *Trends in Cognitive Science 3*, 1 (1999), 22–31.

[130] Weng, J., and Chen, S. Incremental learning for vision-based navigation. In *Proceedings of the International Conference on Pattern Recognition* (Vienna, Austria, 1996), pp. 45–49.

[131] Weng, J., and Chen, S. Vision-guided navigation using SHOSLIF. *Neural Networks 11* (1998), 1511–1529.

[132] Young, R. A. The Gaussian derivative model for spatial vision: I. Retinal mechanisms. *Spatial Vision 2*, 4 (1987), 273–293.

[133] Zerroug, M., and Medioni, G. The challenge of generic object recognition. In *Object Representation in Computer Vision: Proc. NSF-ARPA Workshop* (1994), pp. 217–232.