UNIVERSITY OF CALIFORNIA, SAN DIEGO

Learning Real-Time Object Detectors: Probabilistic Generative Approaches

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Cognitive Science

by

Ian Robert Fasel

Committee in charge:

Jochen Triesch, Chair
Virginia de Sa
Javier Movellan
Martin Sereno
Kenneth Zeger

2006

To my mother Carolyn,
my father Frederick,
my brother Aaron,
and my dear cats Shivalinga and Sallah.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This research would have been impossible without the dedicated support of Dr. Javier Movellan. Few graduate students could hope for an advisor more committed to his students, to science, and to making research both productive and extremely fun at the same time. His willingness to take risks, to challenge accepted beliefs, and to attempt the impossible stands as an example of what a good scientist should be.

The Machine Perception Laboratory has been an incredibly stimulating and productive environment; it has been a pleasure to work with Marni Bartlett, Gwen Littlewort, Claudia Lainsceck, John Hershey, Tim Marks, Jonathan Nelson, Joel Chenu, Jake Whitehill, Nick Butko, Luis Palacios, Paul Ruvolo, Cynthia Taylor, and many others.

Hiroshi Ishiguro has been a continuous source of ideas and questions; I am grateful to him for inviting me to work with him at ATR, Kyoto. Thanks also to the Sony Intelligence Dynamics Laboratory, in its various incarnations, particularly Masahiro Fujita, Fumihide Tanaka, Kenta Kawamoto, Kohtaro Sabe, and Ken'ichi Hidai, who are among the few, I believe, who truly understand – and have any hope of solving – many of the difficult problems faced by anyone (or any thing) that wants to operate a physical body in the real-world.

Professors Jochen Triesch and Virginia de Sa have always been available for any advice on any topic, and I look to them as examples as I begin my own professional career. Marty Sereno has been a source of seemingly endless knowledge of how the brain works and has always been available to answer whatever strange questions I can think to ask. Ken Zeger's difficult questions and high standards have been in the back of my mind throughout the work on this thesis.

Many of the broader questions about mind and brain which motivate my research began to take form while under the advisement of Ed Hutchins. He and Jim Hollan always pushed me to ask big questions and pursue my interests wherever they lay. Fellow graduate students in the Department of Cognitive Science, especially Sam Horodezsky, Hsin-Hao Yu, Rob Liebscher, Ayse Saygin, David Groppe, Laura Kemmer, David Molfese, and Brice Moritano, were the source of many wonderful discussions and arguments.

In addition to those mentioned above, I have deeply appreciated the support and friendship of Chika Ogawa, Flavia Fillimon, Chris Cordova, Chrissy Camblin, Dan

Eaton, Nick Fay, and Matt and Cassie Hawk. Finally, I am grateful to my parents and my brother for their unconditional love, constant support, and encouragement.

Ian Fasel was supported by a National Science Foundation Graduate Research Fellowship (NSFGRF), NSF grants NSF IIS-0220141, NSF IIS-0223052, NSF IIS-0329287 and a UC Discovery grant from Sony Corporation, UC DIMI-01-1030.

The text of Chapter 2, in part, is a reprint of the material as it appears in *Computer Vision and Image Understanding*, Ian Fasel, Bret Fortenberry, Javier Movellan, 2005, vol.1. Ian Fasel was the primary author and the second co-author listed in this publication directed and supervised the research which forms the basis for this chapter.

VITA

1999	B.S. Electrical Engineering, *Honors*, University of Texas at Austin
1999	B.A. Plan II Honors Liberal Arts, *Highest honors*, University of Texas at Austin, *model thesis*
2000–2003	National Science Foundation Graduate Research Fellowship
2002, 2004	*Research Intern* Applied Telecommunications Research (ATR) International, Intelligent Robotics and Communications Lab, Kyoto, Japan
2003	M.S., University of California San Diego
2006	Ph.D., University of California San Diego

PUBLICATIONS

Gwen Littlewort, Marianne S. Bartlett, Ian R. Fasel, Josh Susskind, and Javier R. Movellan. An automatic system for measuring facial expression in video. *Image and Vision Computing.* (in press)

Marianne S. Bartlett, Gwen Littlewort, Claudia Lainscsek, Ian R. Fasel, Mark G. Frank, and Javier R. Movellan. Fully automatic facial action recognition in spontaneous behavior. In *7th International Conference on Automatic Face and Gesture Recognition*, p. 223-228. 2006.

Marianne S. Bartlett, Gwen Littlewort, Mark G. Frank, Claudia Lainscsek, Ian R. Fasel, and Javier R. Movellan. Recognizing facial expression: Machine learning and application to spontaneous behavior. In *IEEE International Conference on Computer Vision and Pattern Recognition*, San Diego, CA, 2005.

Ian R. Fasel, Bret Fortenberry, and Javier R. Movellan. A generative framework for real-time object detection and classification. *Computer Vision and Image Understanding*, 98, 2005.

Ian R. Fasel and Javier R. Movellan. Weakly-supervised robust, real-time object detection and localization. In *NIPS Workshop on Automatic Discovery of Object Categories*, Whistler, BC, December 2005.

Marianne S. Bartlett, Gwen Littlewort, Claudia Lainscsek, Ian R. Fasel, and J.R. Movellan. Machine learning methods for fully automatic recognition of facial expressions and facial actions. In *IEEE International Conference on Systems, Man & Cybernetics*, The Hague, Netherlands, October 2004.

Takayuki Kanda, Nicolas Miralles, Masayuki Shiomi, Takahiro Miyashita, Ian R. Fasel, Javier R. Movellan, and Hiroshi Ishiguro. Face-to-face interactive humanoid robot. In *IEEE 2004 International Conference on Robotics and Automation*, 2004.

Gwen Littlewort, Marianne S. Bartlett, Ian R. Fasel, J. Susskind, and J.R. Movellan. Dynamics of facial expression extracted automatically from video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

Gwen Littlewort, Marianne S. Bartlett, Joel Chenu, Ian R. Fasel, T. Kanda, H. Ishiguro, and J.R. Movellan. Towards social robots: Automatic evaluation of human-robot interaction by face detection and expression classification. In *S. Thrun, L. Saul, and B. Schölkopf, editors, Advances in neural information processing systems*, volume 16, pages 1563–1570. MIT Press, Cambridge, MA, 2004.

Gwen Littlewort-Ford, Marianne S. Bartlett, Ian R. Fasel, Josh Susskind, and Javier R. Movellan. Dynamics of facial expression extracted automatically from video. In *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Face Processing in Video*, Orlando, FL, 2004.

Marianne S. Bartlett, Gwen Littlewort, Ian R. Fasel, and Javier R. Movellan. Real time face detection and facial expression recognition: Development and applications to human computer interaction. In *IEEE International Conference on Computer Vision and Pattern Recognition*, Madison, WI, 2003.

Javier R. Movellan, Ian R. Fasel, and Hiroshi Ishiguro. ATRV-I: a dataset for development and evaluation of pose estimation and expression recognition systems. In *Proceedings of the X Joint Symposium Neural Computation*, Irvine, CA, 2003.

Ian R. Fasel, Gedeon O. Deak, Jochen Triesch, and Javier R. Movellan. Combining embodied models and empirical research for understanding the development of shared attention. In *Proceedings of the International Conference on Development and Learning (ICDL02)*. MIT, 2002.

Ian R. Fasel and Javier R. Movellan. Comparison of neurally inspired face detection algorithms. In *Proceedings of the international conference on artificial neural networks (ICANN 2002)*. UAM, 2002.

Ian R. Fasel and Javier R. Movellan. Object detection as a Markov decision process. In *Proceedings of the 9th Symposium on Neural Computation*. California Institute of Technology, May 2002.

Ian R. Fasel, Marianne S. Bartlett, and Javier R. Movellan. A comparison of Gabor filter methods for automatic detection of facial landmarks. In *Proceedings of the 5th International Conference on Automatic Face and Gesture Recognition*. Washington DC, 2002.

Ian R. Fasel Marianne S. Bartlett Gwen Littlewort-Ford and Javier R. Movellan. Real time fully automatic coding of facial expressions from video. In *Proceedings of the 9th Symposium on Neural Computation*. California Institute of Technology, May 2002.

Gedeon O. Deak, Ian R. Fasel, and Javier R. Movellan. The emergence of shared attention: using robots to test developmental theories. In *Balkenius, editor, Proceedings of the first international workshop on epigenetic robotics: modeling cognitive development*, pages 95–104. Lund University Cognitive Studies, Lund, Sweden, 2001.

Ian R. Fasel and Javier R. Movellan. Meta analysis of neurally inspired face detection algorithms. In *Proceedings of the 8th Symposium on Neural Computation*. The Salk Institute for Biological Studies, May 2001.

Ian R. Fasel, Kurt D. Bollacker, and Joydeep Ghosh. A neural network based classification and real-time biofeedback system for clarinet tone-quality improvement. In *Proceedings of the 12th International Joint Conference for Neural Networks*, 1999.

ABSTRACT OF THE DISSERTATION

Learning Real-Time Object Detectors:
Probabilistic Generative Approaches

by

Ian Robert Fasel
Doctor of Philosophy in Cognitive Science

University of California San Diego, 2006

Jochen Triesch, Chair

This dissertation is a computational investigation of the task of locating and recognizing objects in unconstrained images in real-time, and learning to do so with minimal supervision. We take a probabilistic generative modeling approach, which involves formulating analytical models of several real-world vision problems, studying how optimal inference would proceed under such models, developing techniques for learning parameters under these models, and evaluating the performance of the optimal inference algorithms in realistic data.

We begin by developing a novel generative model of images under which an image is a collection of sets of pixels which are generated by different object categories. This provides a novel definition of "object" as a set of pixels that are co-dependent, but conditionally independent of the other sets of pixels in the image. We then develop an algorithm for optimal inference (i.e., detection of objects) and maximum likelihood learning when the segmentation of training images is known. We point out a computational tradeoff between robustness of object detection and precision of localization, and propose context dependent detectors as a way to solve the problem. These techniques are used to develop a state-of-the-art, real-time head, eye, and blink detector. We predict that similar context-dependent detectors may be found in the brain.

We develop an algorithm for optimal inference and maximum likelihood learning when the segmentation of training images is unknown. We test this on image datasets labeled with the identity but not the location of objects, and achieve state-of-the-art performance in discovery of object categories. We then test the algorithm in a fully unsupervised context, in which a real-time person detector is learned from just a few minutes of visual information self-labeled through multi-modal contingency detection. This suggests that early face (and other) preferences in humans infants may be evidence for rapid statistical learning rather than innate biases. We develop software for learning robust, real-time object detectors from both labeled and unlabeled examples, including a real-time head, eye, and blink detector available to the public.

1

Introduction

Trying to understand perception by studying only neurons is like trying to understand bird flight by studying only feathers: it just cannot be done. In order to understand bird flight, we have to understand aerodynamics, only then the structure of the feathers and the different shapes of birds' wings make sense.

–David Marr, **Vision**, 1982

Sensory input is continuous. Light arrives on our retina in an unbroken stream. Yet much of our understanding of the world relies on the perception that the world is divided into *objects*. The concept of object is such an integral part of so many perceptual skills that it is easy to take it for granted. However, designing a machine that can segregate pixels from a camera into different objects is a nontrivial task. Indeed, the general problem of segregating and recognizing objects with a computer program in unconstrained, real-world environments is completely unsolved.

This dissertation is a computational investigation of the task of locating and recognizing objects in unconstrained images in real-time. To better understand this task, we develop systems that learn to detect the presence or absence of an object from example images that are labeled only as containing or not containing an object of interest. Learning about objects from such non-specific labels is more challenging than standard, supervised learning approaches to object detection. In supervised approaches, humans must first develop a training set of images and indicate by hand precisely which pixels in the images contain the object of interest. This labeling process, which must be performed on thousands, if not tens of thousands of images, is labor intensive; as such, until now reliable object detectors have only been developed for a few carefully chosen object categories such as faces, pedestrians, and cars [Sung and Poggio, 1998;

Rowley et al., 1998; Schneiderman, 2000; Jones and Viola, 2003; Dalai and Triggs, 2005]. Being able to learn robust, real-time object detectors without the need for such specific hand labels would represent a significant advance in machine perception, allowing the development of a wide variety of applications.

A computational study of both how to detect objects and how we might learn to do so is also an important part of neuroscience. As David Marr emphasizes, a critical part of understanding how the brain works is to carefully study the problems it solves. Attempting to build machines that solve specific, real-world problems can lead to a deeper understanding of the computational requirements faced by the brain, and can suggest ideas about how the brain may solve them [Edelman and Vaina, 2001].

In this dissertation our primary analytical tool is probability theory, in particular probabilistic generative models. Generative models are a mathematical description of how hidden causes (random variable H) generate observed data (random variable X). Provided a generation model, we can then ask the question: "according to my model of how observable patterns are generated, what hidden causes could have produced the data I observed?"

A probabilistic model provides an explicit probability distribution $p(h)$, called the prior distribution, over the possible values of the hidden variable. In addition, it provides a model of how data are generated when their causes are known, $p(x|h)$. Together, these provide a model of the joint distribution of the hidden and observed variables: $p(x, h) = p(x|h)p(h)$. We can then use Bayes' rule to produce the inverse model, the posterior distribution $p(h|x)$, which represents the probabilities that each of the various causes h could have produced the observed data x:

$$p(h|x) = \frac{p(x|h)p(h)}{p(x)}. \tag{1.1}$$

It is useful to divide H into two components: a component λ, called the *model parameters*, that underlies *all* of our observations, and a component Y, which may be different between each observation. In the Bayesian approach, given the set of examples (or *training data*) $\mathbf{x}$ and a new independent sample x, inference about y involves marginalizing over all possible parameters, i.e.,

$$p(y|x, \mathbf{x}) = \sum_{\lambda} p(y, \lambda | x, \mathbf{x}) \tag{1.2}$$

$$= \sum_{\lambda} p(\lambda | x, \mathbf{x}) p(y | x, \mathbf{x}, \lambda). \tag{1.3}$$

Instead of computing this sum over all possible values of λ, a common approach, and the one taken in this dissertation, is to find the most probable λ given the training data, i.e.,

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}}\, p(\lambda|\mathbf{x}). \tag{1.4}$$

This "best estimate" $\hat{\lambda}$ is then taken to summarize all the relevant information about the training data necessary to infer the hidden causes for new data, so we do not need to consider $\mathbf{x}$ once we have $\hat{\lambda}$, i.e.,

$$p(y|x, \mathbf{x}) \approx p(y|x, \hat{\lambda}). \tag{1.5}$$

There are therefore two separate inference processes. First, given the example data, the *learning* process involves finding the value $\hat{\lambda}$ that best explains the training data. Second, given a *new* example and $\hat{\lambda}$, the *inference* process involves using $p(y|x, \hat{\lambda})$ to answer: "what is the probability that each of the various causes y caused this new example x?"

The generative modeling approach can be a fruitful way to investigate real-world problems. It forces us to state our underlying assumptions explicitly, and enables formal consideration of these two inference processes and how these inference processes might change if our assumptions are changed. The methodological stance referred to as *probabilistic functionalism* [Movellan and Nelson, 2001] argues that using this approach to solve real problems faced by organisms, in unconstrained environments and realistic (biologically relevant) time-scales, is an essential tool to learn about the problems faced by the brain.

In Chapter 2 we apply the approach of probabilistic functionalism to understanding the specific tasks of face, eye and blink detection. While we focus on this specific task, our goal is to learn about the more general problem of detecting objects and their parts. We begin by formulating a probabilistic generative model of scenes as a set of patches of pixels, each of which is generated either by a foreground or by a background appearance distribution. The pixels within each patch are co-dependent, but are independent of the other pixels in the image. The optimal inference algorithm for detecting objects requires we compute probability ratios for every candidate set of pixels in the scene – namely, the conditional probability of the set of pixels given that their underlying cause was object, divided by the the conditional probability of the set of pixels given that their underlying cause was not the object.

We then propose a hierarchical scheme in which objects are located by applying a chain of context dependent likelihood ratio estimates. Learning involves choosing parameters for these likelihood ratio estimates which maximize the probability of a set of example images which have been labeled by a human for head and eye, and eyelid-openness. Optimal inference (i.e., detection) involves finding the maximum probability hypothesis given a new image and fixed model parameters. We illustrate the approach by building a state-of-the art, real-time head, eye, and blink detector.

The approach used in Chapter 2 requires that training images come with labels in which the position of objects was fully specified. In Chapter 3, we turn to the problem of learning these parameters when such labels are not available. Instead, all that is required are a set of images labeled as containing or not containing the object of interest. We take a Bayesian approach to learning the likelihood ratios needed for optimal inference. Surprisingly, we show that it is possible to avoid a potentially costly marginalization over all possible image segmentations needed to compute the true likelihood of the data given parameters. We test this system on a variety of image datasets and find that our approach yields state-of-the-art performance in classification accuracy. Moreover, at runtime our system is able to perform detection several orders of magnitude faster than previous systems while still providing precise localization information.

Chapter 3 required that a human provide information about whether or not an object was present. In Chapter 4, we illustrate how weak cues from one sensory domain can be used to bootstrap learning in the visual domain, to learn robust object detectors in a fully unsupervised manner. We created a system that uses auditory contingency to self-generate labels of the presence or absence of people in a continuous stream of visual data. We find that using only a few hundred images, representing just six minutes of visual experience, the system is capable of learning a robust, real-time person detector. The results of this study have important implications for our understanding of human learning and development. First, it supports the hypothesis that auditory contingency is a useful multi-modal cue that could be used by the infant brain to learn about people. Second, we show that although the system only had experience with images of humans in a real, three-dimensional environment, it developed preferences to iconic sketches of faces similar to the preferences found in forty minute old infants [Johnson et al., 1991]. This represents the first computationally plausible alternative to the hypothesis that infants are born with an innate face preference, suggesting instead that early face (and other) preferences may be evidence for rapid statistical learning rather than innate knowledge about the appearance of faces.

Contributions of the dissertation

1. We developed a novel generative model under which an image is as a collection of *sets* of pixels which are generated by different object categories. This provides a novel definition of "object" as a set of pixels that are co-dependent, but conditionally independent of the other sets of pixels in the image. This notion can be generalized to modalities other than vision.

2. We developed an algorithm for optimal inference (i.e., detection of objects) and maximum likelihood learning when the segmentation of training images is known.

3. We pointed out a computational tradeoff between robustness of object detection and precision of localization, and propose context dependent detectors as a way to solve the problem. We predict that such context-dependent detectors may be found in the brain.

4. We developed an algorithm for optimal inference and maximum likelihood learning when the segmentation of training images is unknown.

5. We showed that with this algorithm it is possible to learn to localize human beings in real-time from a very small number of training examples and multi-modal contingency detection. This suggests that early face (and other) preferences in humans infants may be evidence for rapid statistical learning rather than innate biases.

6. We developed software for learning robust, real-time object detectors from both labeled and unlabeled examples, including a real-time face, eyes, and blink detector. The software is being made available for free – interested parties should contact the author, however portions of the code are already available (under a BSD license) from `http://mplab.ucsd.edu` under the title "Machine Perception Toolbox".

2

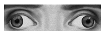
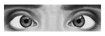
A Generative Framework for Real Time Object Detection

The study of face perception has been revitalized thanks to recent progress in cognitive neuroscience. The advent of modern neuro-imaging is revolutionizing the study of the mind and presenting a picture of the human brain far different from a general purpose computing machine. Single neuron recording and imaging studies are showing specific neural systems that play a crucial role in the perception of faces, facial features, and facial expressions. These include the fusiform face area, superior temporal sulcus, orbital frontal cortex, frontal operculum, right somatosensory cortex, and the amygdala [Kawashima et al., 1999; George et al., 2001].

Face perception has been a traditional area of research in developmental psychology, a discipline that studies how the human mind develops from infancy to adulthood. Face processing in general and eye detection in particular is deemed so important in this field that some of its most influential researchers have postulated the need for innate eye detection and gaze processing modules. These ideas are still controversial but recent experiments have shown that from birth human infants are exceptionally sensitive to the eye and to mutual gaze engagement [Farroni et al., 2002; Johnson, 2001]. These systems may help tune the newborn infant towards interaction with their caregivers [Baron-Cohen, 1995].

In recent years there has been an emerging community of machine perception scientists focused on automatic detection of faces and facial behavior. The special importance of the eyes is becoming quite clear within this community. There are at least

Table 2.1: FACS codes involving eyes, from `http://www.cs.cmu.edu/face/facs.htm`

Code	Descriptor	Muscles Involved	Example
AU5	Upper Lid Raiser	Levator Palpebrae Superioris	
AU6	Cheek Raiser	Orbicularis Oculi, Pars Orbitalis	
AU7	Lid Tightener	Orbicularis Oculi, Pars Palebralis	
AU41	Lid Droop	Relaxation of Levator Palpebrae Superioris	
AU42	Slit	Orbicularis Oculi	
AU43	Eyes Closed	Relaxation of Levator Palpebrae Superioris; Orbicularis Oculi, pars Palpebralis	
AU44	Squint	Orbicularis Oculi, pars Palpebralis	
AU45	Blink	Relaxation of Levator Palpebrae Superioris; Orbicularis Oculi, pars Palpebralis	
AU46	Wink	Relaxation of Levator Palpebrae Superioris; Orbicularis Oculi, pars Palpebralis	
AU61	Eyes Turn Left	Lateral and Medial Rectus	
AU62	Eyes Turn right	Lateral and Medial Rectus	
AU63	Eyes Up	Superior Rectus	
AU64	Eyes Down	Inferious Rectus	
AU65	Walleye	Lateral Rectus	
AU66	Crosseye	Medial Rectus	

two reasons for this: (1) Proper registration. In a recent evaluation of state of the art face recognition system it was proposed that a large proportion of the failures of these system was due to poor alignment and registration of facial features, particularly in outdoors conditions. Good eye detection in realistic environments may thus have a tremendous impact on the accuracy of face perception technologies [Phillips, 2003]. (2) Information value. Eyes and eye movements are a particularly important source of information in human interaction. Indeed, the Facial Action Coding System of Ekman and Friesen [1978], arguably the most comprehensive standard for coding facial behavior, devotes 15 categories to describe eye behavior (see Table 2). Only the mouth surpasses the eyes in the number of categories assigned to it. This reflects the fact that eye behavior is extremely rich and particularly informative about the state of human beings.

Current work on eye detection divides into approaches based on visible spectrum cameras and approaches based on near-infra-red (NIR) cameras. In indoor and relatively controlled conditions the spectral properties of the pupil under NIR illumination provide a very clean signal that can be processed very fast and accurately [Haro et al., 2000; Ji and Yang, 2001, 2002]. While NIR based methods are practical and worth pursuing, it is also important to pursue visual spectrum methods for the following reasons: (1) NIR based methods tend to produce a large number of false positives when used in relatively uncontrolled illumination conditions; (2) NIR based methods do little to further our understanding about the perceptual problem the brain solves when processing faces in natural conditions.

Of all the eye related behaviors, perhaps the most important is blinks, action unit 45 in the Facial Actions Coding System. This is due to its relevance in several fields, including neurology, physiology, and psychology. For example, blink rate is known to vary with physiological and emotional arousal, cognitive effort, anxiety, fatigue, and deceit [Holland and Tarlow, 1972; Ekman, 1985; Karson, 1988; Van-Orden et al., 2000; Ji and Yang, 2001]. Ji and Yang [2002] presents a state of the art method to detect blinks in real time using NIR imaging. Approaches based on visual spectrum images also exist. Bartlett, Braathen, Littlewort, Smith, and Movellan [2003] present an approach to detect blinks in indoors environment using Support Vector Machines. Cohn, Xiao, Moriyama, Ambada, and Kanade [in press] describe an approach that uses hand-coded eye-blink detectors. They report results comparable to those of Bartlett et al. [2003] on the same testing data set. Both systems handled out-of-plane rotations of the head by fitting a 3D deformable model of the head and then re-rendering the image into a frontal view.

2.1 A Generative Model for Images

In this section we frame the problem of finding faces and facial features as a Bayesian inference problem: We formulate a model of how images are generated and then derive an algorithm for making optimal inferences under this model. One advantage of generative models is that probability estimates of the categories of interest are computed explicitly, facilitating integration with other potential sources of information not necessarily considered at design time. In addition generative models force us to make our assumptions explicit, facilitating progress towards more effective algorithms.

Unless otherwise stated, capital letters will represent random variables and small letters specific values taken by those variables. When possible we use informal shorthand notation and identify probability functions by their arguments. For example, $p(y)$ is shorthand for the probability (or probability density) that the random matrix Y takes the specific value y.

We model the image as a collage of rectangular patches of arbitrary size and loca-

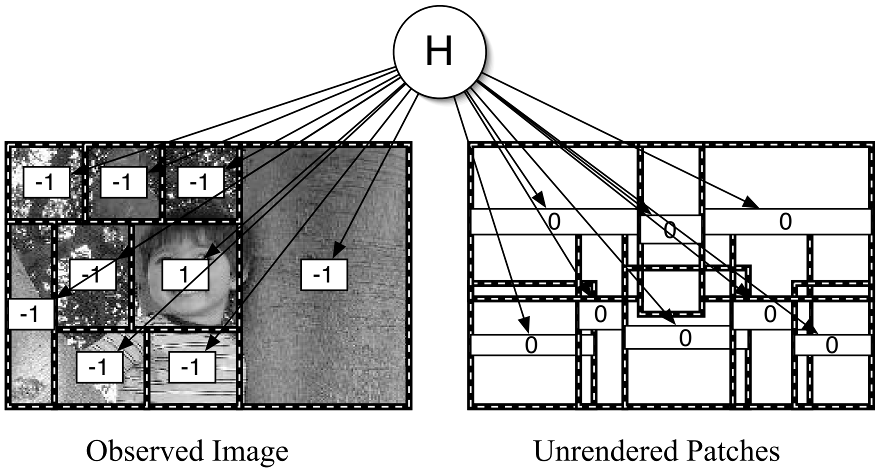


Figure 2.1: The hidden variable H determines which image patches will render the background (-1) which patches will render the object of interest (1) and which patches will not be rendered (0). The set of rendered patches determine the observed image.

tion, some patches rendering the object of interest, the others rendering the background. Given an image our goal is to discover the patches that rendered the object. Let Y be a random matrix representing an image with a fixed number of pixels. Let y be a specific sample from Y. Let $\mathcal{A} = (a_1, a_2, \ldots, a_n)$ be an enumeration of all possible rectangular image patches, e.g. a_i determines the position and geometry of a rectangle in the image plane. Let y_{a_i} be a matrix whose elements are the values of y for the pixels in the rectangle a_i. Let $H = (H_1, \ldots, H_n)$ be a random vector that assigns each of the n patches to one of three categories: H_i takes the value 1 when the patch a_i renders the object of interest, it takes value -1 when it renders the background, and value 0 when it is not rendered (see Figures 2.1 and 2.2). We refer to a value h as a *segmentation* of an image.

The image generation process proceeds as follows (see Figure 2.1). First a segmentation h is chosen with probability $p(h)$. Then for each patch a_i if $H_i = 1$ then an image of size a_i is chosen from the object distribution $q(\cdot \mid a_i, 1)$ independently of all the other patches. If $H_i = -1$ then a background image y_{a_i} is chosen from the background distribution $q(\cdot \mid a_i, -1)$. If $H_i = 0$ then a_i is not rendered. The observed image y is the collection of the rendered patches.

The model is specified by the prior probabilities $p(h)$ and by the object and background rendering distributions q. The prior is specified by the marginal probabilities $\{P(H_i = 1) \ : \ i = 1, \ldots, n\}$, with the constraint that values of h that do not partition the image plane have zero probability (i.e., each pixel must be rendered by exactly one object or background element), and by one of the two following constraints: (I) For cases in which we know there is one and only one object of interest in the image plane, only values of h with a single 1 are allowed. (II) For cases in which there may be an arbitrary number of objects of interest we assume the location of a rendered object does not inform us about the location of other objects, except for the fact that each pixel can only be rendered by a single object or background element. More formally, for $i = 1, \dots, n$, the random variables $\{H_j \ : \ j \neq i\}$ are independent of H_i when conditioning on the event $\{H_i \neq 0\}$. For a given image y our goal is to detect patches rendered by the object. There are two cases of interest: (I) We know there is one and only one patch rendered by the object ; (II) There is an unknown and arbitrary number of patches rendered by the object model.

2.1.1 Case I: Single Object

Suppose we know there is one and only one patch in the image plane that rendered the object of interest. Then our goal is to find the most probable patch $\hat{k} \in \{1, \dots, n\}$ given the image y, i.e,

$$\hat{k} = \operatorname*{argmax}_i P(H_i = 1 \mid y) \tag{2.1}$$

Using the law of total probability we have that

$$P(H_i = 1 \mid y) = \frac{\sum_h P(H_i = 1) p(h \mid H_i = 1) p(y \mid h, H_i = 1)}{p(y)} \tag{2.2}$$

Note that $p(h \mid H_i = 1)$ is zero if the segmentation h does not contain the patch a_i and one otherwise. Moreover, for any h that includes a_i we have that

$$p(y \mid h, H_i = 1) = \frac{q(y_{a_i}; a_i, 1)}{q(y_{a_i}; a_i, -1)} Z(h, y) \tag{2.3}$$

where

$$Z(h, y) = \prod_{i : h_i \neq 0} q(y_{a_i}; a_i, -1) \tag{2.4}$$

The term $Z(h, y)$ describes how well the image y can be explained by the segmentation h with all the patches rendering background, no objects. Thus

$$\begin{aligned} P(H_i = 1 \mid y) &= P(H_i = 1) \frac{q(y_{a_i}; a_i, 1)}{q(y_{a_i}; a_i, -1)} \frac{\sum_h p(h \mid H_i = 1) Z(h, y)}{p(y)} \\ &= P(H_i = 1) \frac{q(y_{a_i}; a_i, 1)}{q(y_{a_i}; a_i, -1)} \frac{E(Z(H, y) \mid H_i = 1)}{p(y)} \end{aligned} \tag{2.5}$$

The term $E(Z(H, y) \mid H_i = 1)$ represents how well the image y can be explained as a mosaic of background patches, provided one of those patches is a_i. If the background distribution model $q(\cdot \mid a_k, -1)$ includes wrongly shifted and scaled versions of the object of interest then $E(Z(H, y) \mid H_i = 1)$ should be small for the patch that actually rendered the object, and large otherwise. This is due to the fact that the patch that includes the object will be hard to explain by the background model (see Figure 2.2). More formally if $E(Z(H, y) \mid H_{\hat{k}} = 1) \leq E(Z(H, y) \mid H_i = 1)$ for $i = 1, \ldots, n$ then

$$\begin{aligned}
\hat{k} &= \operatorname*{argmax}_i P(H_i = 1 \mid y) \\
&= \operatorname*{argmax}_i P(H_i = 1) \frac{q(y_{a_i}; a_i, 1)}{q(y_{a_i}; a_i, -1)} \\
&= \operatorname*{argmax}_i \; \log P(H_i = 1) + \log \frac{q(y_{a_i}; a_i, 1)}{q(y_{a_i}; a_i, -1)} \qquad (2.6)
\end{aligned}$$

The optimal inference algorithm prescribes scoring all possible patches in terms of a function that includes the prior probability that the patch is generated by an object and a likelihood ratio term. The patch that maximizes this score is then chosen.

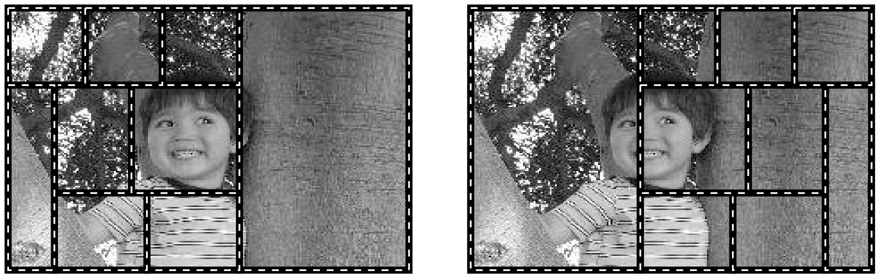


Figure 2.2: The segmentation on the left contains the patch that generated the object of interest (i.e. the face). It will be hard for this segmentation to explain the image as a collection of background patches. The segmentation on the right does not contain the object patch. Since the background model includes wrongly shifted versions of faces it will be easy to explain the image as a collection of background patches.

2.1.2 Case II: Multiple Objects

This case applies, for example, in face detection problems for which we do not know *a priori* how many faces may appear in the image plane. To formalize the problem we define a function Φ measuring the degree of match between any two arbitrary segmentations h and h'

$$\Phi(h,h') = \sum_{i=1}^{n} \rho(h_i, h'_i) \tag{2.7}$$

$$\rho(h_i, h'_i) = (\delta(h_i, 1) + \delta(h_i, -1))\, \delta(h_i, h'_i) \tag{2.8}$$

where δ is the Kroenecker delta function. ρ counts the number of patches for which both h and h' assign the same "object" or "background" label and ignores all the patches that are not rendered by h. Our goal is to find a partition $\hat{h}$ that optimizes the expected match

$$\hat{h} = \operatorname*{argmax}_{h'} E(\Phi(H,h') \mid y) = \operatorname*{argmax}_{h'} \sum_{h} p(h \mid y)\Phi(h,h') \tag{2.9}$$

The optimal assignment follows

$$\hat{h}_i = \begin{cases} 1 & \text{if } p(H_i = 1 \mid y) > p(H_i = -1 \mid y) \\ -1 & \text{else} \end{cases} \tag{2.10}$$

Thus, to find the optimal assignment we need to scan all possible image patches $a_1, \ldots, a_n$, compute the log posterior probability ratio

$$\log \frac{P(H_i = 1 \mid y)}{P(H_i = -1 \mid y)} \tag{2.11}$$

and assign "object" labels to the patches for which this ratio is larger than 0.

Using the law of total probability we have that

$$P(H_i = 1 \mid y) = \sum_{h} P(H_i = 1) p(h \mid H_i = 1) p(y \mid h, H_i = 1) \tag{2.12}$$

where $p(h \mid H_i = 1)$ is zero if the segmentation h does not contain the patch a_i, and

$$p(y \mid h, H_i = 1) = q(y_{a_i}; a_i, 1) \prod_{\{j : h_j = -1\}} q(y_{a_j}; a_j, -1) \tag{2.13}$$

Thus for $k = -1, 1$ we have that

$$P(H_i = k \mid y) = P(H_i = k) q(y_{a_i}; a_i, k) \sum_{h} p(h \mid H_i = k) \prod_{\{j : h_j = -1\}} q(y_{a_j}; a_j, -1) \tag{2.14}$$

and due to the fact that $\{H_j \;:\; j \neq i\}$ are independent of H_i given $\{H_i \neq 0\}$ it follows that

$$\log \frac{P(H_i = 1 \mid y)}{P(H_i = -1 \mid y)} = \log \frac{P(H_i = 1)}{P(H_i = -1)} + \log \frac{q(y_{a_i}; a_i, 1)}{q(y_{a_i}; a_i, -1)} \tag{2.15}$$

In order to make optimal inferences all we need is a model for the prior probability of object locations and a model for the log-likelihood ratios of image patches of arbitrary geometry. In Section 2.2 we will see how these models can be learned using boosting methods.

2.2 Learning Likelihood Ratios using GentleBoost

The inference algorithm presented above requires a likelihood ratio model. Given an arbitrary image patch y we need an estimate for the ratio between the probability of such a patch being generated by the object class *vs.* the background class. In this paper we learn these likelihood ratios using GentleBoost, a boosting algorithm developed by Friedman et al. [1998]. Boosting [Freund and Schapire, 1996b, 1999] refers to a family of machine learning algorithms for learning classifiers by sequential accumulation of experts that focus on the mistakes made by previous experts. Friedman et al. [1998] showed that boosting methods can be reinterpreted from the point of view of sequential statistical estimation, an interpretation that makes it possible to use it in the generative framework proposed here.

The goal is to learn a model for the log-likelihood ratio of arbitrary image patches. During training we are given a set of examples $\{\ (y_i, z_i) : i = i, \ldots, m\}$, where y_i is an image patch, and $z_i \in \{-1, +1\}$ its category label, i.e., object or background. The model used in GentleBoost is of the following form:

$$p(y) = \frac{1}{1 + e^{-2\sum_j f_j(y)}} \tag{2.16}$$

where $p(y)$ is the probability that image patch y belongs to one of the two categories of interest, and $f_i(y)$ is the opinion of the i^{th} expert, as defined in Figure 2.3. GentleBoost can be seen as an application of the Newton-Raphson optimization algorithm to the problem of minimizing the following chi-square error [Friedman et al., 1998]:

$$\rho = \sum_i \frac{t_i - p(y_i)}{\sqrt{p(y_i)(1 - p(y_i))}} \tag{2.17}$$

where $t_i = 0.5(z_i + 1) \in \{0, 1\}$ is the category label for the i^{th} training input y_i. Since $p(y_i)$ is the probability of a Bernoulli random variable with mean $p(y_i)$ and standard deviation $\sqrt{p(y_i)(1 - p(y_i))}$, then ρ can be seen as a the number of standard deviations between the observed label and the average label value. As the number of examples in the training set increases, minimizing the chi-square error becomes identical to maximizing the likelihood. However when the number of samples is small, chi-square estimators can be more efficient than maximum likelihood estimators.

2.2.1 Selecting Wavelets and Tuning Curves

GentleBoost chooses a set of experts $f_1, f_2, \ldots$ in a sequential manner. Each Newton-Raphson step results in the selection of the expert that maximally reduces the current chi-square error given the already selected set of experts. In practice this can be done in a variety of ways. We use the following approach:

We start with a large pool of wavelets $\{w_1, \ldots, w_n\}$, about 170,000 in our case (see Section 2.4), and define an expert as the combination of a wavelet w and a tuning curve h to be defined below. By iteration t of the Newton-Raphson method, we have already selected $t-1$ experts. At this point we go over each wavelet w in our pool and for each wavelet we estimate the tuning function $h : \mathbb{R} \to [-1, 1]$ that minimizes ρ given the outputs of the wavelet w and the information provided by the $t-1$ experts already selected. This function can be shown to have the following form

$$h(w(y)) = E^{P_t}[Z \mid w(y)] \tag{2.18}$$

where $Z \in \{-1, 1\}$ is the category label, and the expectation is taken with respect to the distribution induced by the weights assigned by GentleBoost to the different training data (see Figure 2.3). We estimate the function h using the Nadaraya-Watson kernel regression method for density estimation [Silverman, 1986]. The training examples used in this regression method are the set of triplets $\{(w(y_i), z_i, P_t(i)) : i = 1, \ldots, m\}$, where $w(y_i)$ is the regressor variable, z_i the label we wish to predict, and $P_t(i)$, the weight of example $y_i, z_i)$.

We call the function h the *tuning curve* for the wavelet w. After we find the optimal tuning curves for all the wavelets in the original pool, we choose the wavelet $\hat{w}$ and corresponding tuning-curve $\hat{h}$ that minimize ρ. This pair defines the expert selected for iteration t, i.e.,

$$f_t(y) = \hat{h}(\hat{w}(y)) \tag{2.19}$$

The process is iterated, each time adding a new wavelet and tuning curve, until ρ no longer decreases. This procedure is illustrated in Figures 2.5 and 2.3.

By the end of training we have a model for the posterior probability of the object class given arbitrary image patches y

$$p(y) = \frac{1}{1 + e^{-2\sum_{\tau=1}^{t} f_\tau(w_\tau(y))}} \tag{2.20}$$

```
• Let $\{\ (y_i, z_i) : i = i, \ldots, m\}$, be a set of training examples, where $y_i$ is the an image patch, and $z_i \in \{-1, +1\}$ its category label.
• Let $P_t(i)$ represent the weight assigned to the $i^{th}$ example at the beginning of iteration $t$ of the GentleBoost algorithm.
• Let the initial distribution be as follows: $P_0(i) = 1/m$, for $i = 1, \ldots, m$, i.e., each training example is weighted equally.
• For time $t = 1, \ldots$
    – For wavelet $w = 1, \ldots, n$
        ∗ Use kernel-regression to find the tuning curve $h$ that best fits the set of triplets $\{(w(y_i), z_i, P_t(i)) : i = 1, \ldots, m\}$.
    – Choose $(\hat{w}, \hat{f})$ the wavelet and tuning curve that minimize the error function $\rho$. They define the expert selected at iteration $t$
          $f_t(y) = \hat{h}(\hat{w}(y))$
    – Update the distribution over training elements
          $P_{t+1}(i) = P_t(i) \frac{e^{-f_t(y_i) z_i}}{Z_t}$
      where $Z_t$ is a normalization factor
          $Z_t = \sum_i P_t(i) e^{-f_t(y_i) z_i}$
    – Update the posterior probability model
          $p(y) = \frac{1}{1 + e^{-2 \sum_{\tau=1}^{t} f_\tau(y)}}$
```

Figure 2.3: The GentleBoost approach used in this paper

This posterior probability estimate reflects the particular proportion π of examples of each class used during training. The inference algorithm in (2.22) requires log-likelihood ratios, not log-posteriors. These can be easily derived from (2.20) using Bayes rule

$$\begin{aligned}\log \frac{q(y_{a_i}; a_i, 1)}{q(y_{a_i}; a_i, -1)} &= \log\left(\frac{1-\pi}{\pi}\right) + \log\left(\frac{p(H_k = 1 \mid y_{a_i})}{p(H_k = -1 \mid y_{a_i})}\right) \\ &= \log\left(\frac{1-\pi}{\pi}\right) + 2f(x)\end{aligned} \tag{2.21}$$

Combining and (2.6) and (2.20) we get

$$\hat{k} = \max_i p(H_i = 1 \mid y) = \max_i \log p(H_i = 1) + 2f(y_{a_i}) \tag{2.22}$$

2.3 Situation Based Inference

One common approach to eye detection is based on the operation of a set of independent feature detectors [Huang and Wechsler, 1999; Fasel et al., 2000]. The output of these detectors (e.g., a detector for the left eye, a detector for the right eye, a detector for the tip of the nose, etc.) is integrated by looking for configurations that match the distribution of inter-feature distances typical of the human face [Wiskott et al., 1997; Leung et al., 1995; Kothari and Mitchell, 1996]. Unfortunately this method scales exponentially with the number of false alarms of each feature detector. Suppose our goal is to find the center of an eye with 1 pixel accuracy. This requires for background models to include examples of eyes shifted by 1 pixel from the center position. In practice, a detector efficient at distinguishing eyes slightly shifted from center is also likely to produce a large number of false positives when scanning general backgrounds that do not include faces, creating an insurmountable problem for methods that rely on feature detection.

The approach we propose here is based on the idea of a bank of situational or context dependent experts operating at different levels of specificity. For example, since the eyes occur in the context of faces, it may be easier to detect eyes using a very large context that include the entire face and then formulate feature detectors specifically designed to work well under such context. While we may think of these as face detectors, we can also think of them as eye detectors that happen to have very large receptive fields. This form of eye detection works under very general context conditions, avoiding the proliferation of false alarms, but may provide poor information about the precise location of the eyes. These eye detectors are therefore complemented by context-specific eye detectors that provide very precise information about the position of the eyes provided the context is known.

More formally, let y represent an observed image, S represent a contextual situation (e.g., the location and scale of a face in the image plane), and O represent the location of the left eye of that face in the image. Using the law of total probability we have that

$$p(o \mid y) = \int p(s \mid y) p(o \mid s, y) \, ds \tag{2.23}$$

Here $p(s \mid y)$ works as a situation detector. Its role is to find regions in the image plane that are likely to contain eyes due to the fact that they contain faces. The $p(o \mid s, y)$ term is a situation specific eye detector. For example it may work when the location and scale of the face in the image plane is known. In this example $p(s \mid y)$ partitions the image pixels into those belonging to the face, y_f, and those belonging to the background, y_b. Once the position and scale of the face are known, the background provides no additional information about the position of the eye, i.e.,

$$p(o \mid y_f, y_b, s) = p(o \mid y_f, s) \tag{2.24}$$

The situational approach proposed here can be iterated, where one first detects a general context, followed by detection of a context within a context, each time achieving higher levels of precision and specificity allowed by the fact that the context becomes smaller and smaller on each iteration.

2.4 Real-time system architecture

In the next sections we describe and evaluate an algorithm that performs optimal inference under the assumptions of the generative model described above. The current system utilizes two types of eye detectors. The first type, which can be thought of as a face detector, starts with complete uncertainty about the possible location of eyes in the image plane. Its role is to narrow down the uncertainty about the location of the eyes while operating in a very wide variety of illumination and background conditions. The second type of detector operates on the output of the first detector. As such it can assume a restricted context and achieve high location accuracy. Once the most likely eye locations are chosen, the image patch surrounding the eyes is passed to a blink detection system for further analysis. The flowchart for this procedure is shown in Figure 2.4.

While the system described here operates on video images in real time, it currently treats each frame as independent of the previous frames, making it equally useful for static images as for video. Treating each video frame independently allows the system to simultaneously code eye location and behavior for multiple faces that may come in and out of the scene at random times.

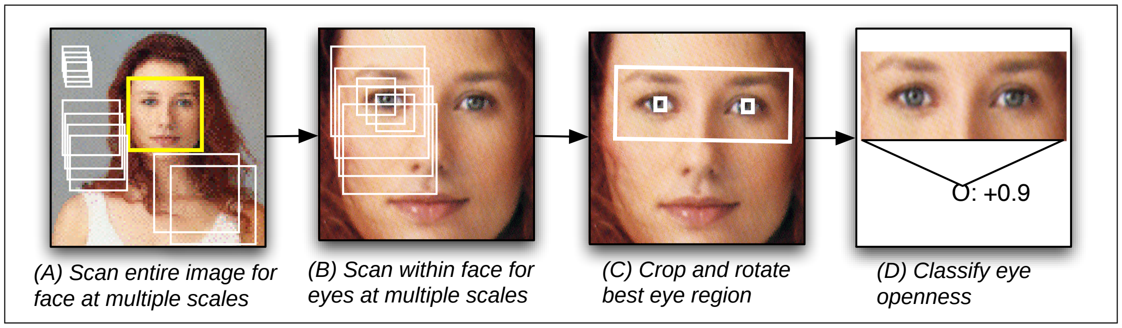

O: +0.9
(A) Scan entire image for face at multiple scales
(B) Scan within face for eyes at multiple scales
(C) Crop and rotate best eye region
(D) Classify eye openness

Figure 2.4: Flowchart for face, eye, and blink detection

2.4.1 Stage I: Eye detection in general background conditions

As described above the first component of the inference process locates regions of the image plane that contain faces, and thus eyes. This module operates under very general background and illumination conditions and greatly narrows down the plausible locations of eyes on the image plane. It makes no prior assumptions about the location of the face.

The general procedure for the image search is similar to the multiscale search of Rowley et al. [1998], who trained a single binary classifier to classify face *vs.* . nonface for patches of fixed size (20×20 pixels), then used that classifier to classify all possible patches in the image. Faces larger than the original size were found by repeating the search in copies of the image scaled to smaller sizes (thus, a 20×20 pixel face in a $1/4$ size copy of the image corresponds to an 80×80 pixel face in the corresponding location in the original).

We use a very similar scheme, however rather than a binary classifier, we developed a likelihood-ratio model using a data set of Web images provided by Compaq Research Laboratories. This data set contains 5000 images containing frontal upright faces taken under a variety of illumination conditions, facial expressions, facial hair, eyeglasses, hats, etc., of widely varying image quality. Faces were cropped and scaled to 24×24 pixels square. The negative examples were sampled from a data set of 8000 images collected from the Web and known not to contain faces. Similarly, these images contained a wide variety of natural indoor and outdoor scenes, text, illustrations, posed images of objects, etc., with varying image quality. The advantage of this Web data set is that it includes far more variability than most other closed databases.

Due to the multi-scale search, about 1 billion total patches are possible in these 8000 images. For the initial negative examples for training, 10,000 square patches, of arbitrary size and at arbitrary locations in the images, were sampled from this data set. Patches were then scaled down to 24×24 pixels. The set of negative samples changes during training thanks to the bootstrap round (described below), so ultimately all 1 billion possible patches were used at some time during training.

The likelihood-ratio model was trained using the GentleBoost method described in Section 2.2. GentleBoost sequentially chooses wavelets from a large pool and combines them to minimize a chi-square error function. The pool of wavelets we choose from was based on Viola and Jones [2001] and consists of Haar-like wavelets. The main reason for their use is that their output can be computed quickly by taking the sum of pixels in two, three, or four equal-sized, adjacent rectangles and taking differences of these sums. To this original set we add center-surround type wavelets and mirror image wavelets that are sensitive to patches symmetric about vertical axis (see Figure 2.7).

The GentleBoost approach described in Section 2.2.1 requires computing tuning curves on each of the wavelet candidates. It is very computationally expensive to perform an exhaustive search over all these wavelets– in a 24×24 pixel window, there are over 170,000 possible wavelets of this type. To speed up training, we break the wavelet selection step into two stages (see Figure 2.5). First, at each round of boosting, we take a random sample of 5% of the possible wavelets. For each wavelet we find the tuning curve that minimizes the loss function ρ if that particular wavelet were added to the pool of already chosen wavelets. In step two, we refine the selection by finding the best performing single-wavelet classifier from a new set of wavelets generated by shifting and scaling the best wavelet by two pixels in each direction, as well as composite wavelets made by reflecting each shifted and scaled wavelet horizontally about the center and superimposing it on the original. Using the chosen classifier as the weak learner for this round of boosting, the weights over the examples are then adjusted using to the GentleBoost rule. This wavelet selection process is then repeated with the new weights, and the boosting procedure continues until the performance of the system on a validation set no longer decreases.

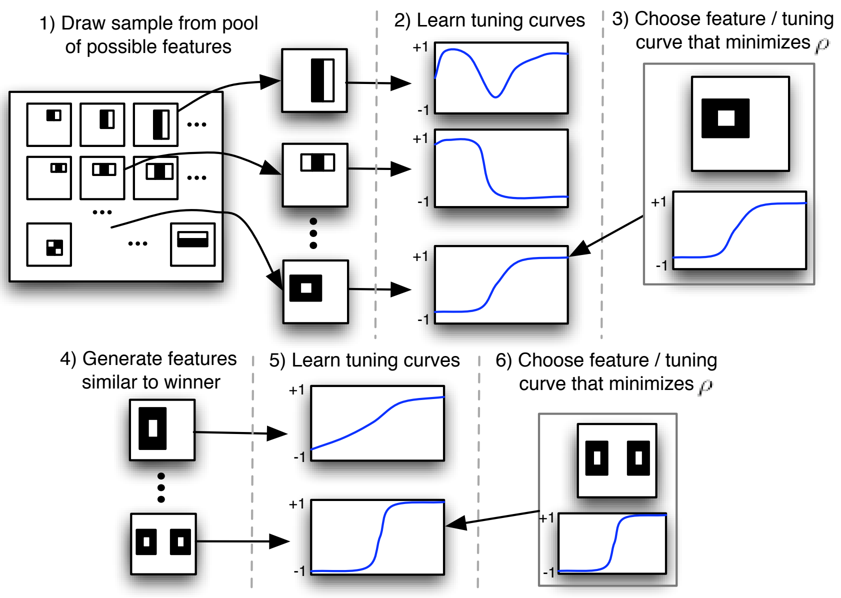

1) Draw sample from pool of possible features
2) Learn tuning curves
3) Choose feature / tuning curve that minimizes ρ
4) Generate features similar to winner
5) Learn tuning curves
6) Choose feature / tuning curve that minimizes ρ
+1
-1

Figure 2.5: Flowchart for one iteration of the feature selection procedure

The inference algorithm calls for likelihood ratio models at multiple scales. Likelihood ratios for larger image patches are obtained by linearly scaling the patches down to 24×24 pixels and then applying the likelihood ratio model trained on that particular scale. Thanks to the choice of Haar-like wavelets for the higher level image representation, this interpolation step can accomplished in constant time regardless of scale (see Viola and Jones [2001]; Jones and Viola [2003] for a more detailed explanation).

Following Viola and Jones [2001], rather than training a "monolithic" classifier which evaluates all its wavelets before it makes a decision, we divided the classifier into a sequence of smaller classifiers which can make an early decision to abort further processing on a patch if its likelihood-ratio falls below a minimum threshold. We can think of this as a situational cascade where each level of the cascade is trained only on patches that were not rejected by previous levels. After each element of the cascaded is trained, a boot-strap round (*à la* Sung and Poggio [1998]) is performed, in which the full system up to that point is scanned across a database of non-face images, and false alarms are collected and used as the non-faces for training the subsequent strong classifier in the sequence. Training the current face-detector took about ten days on a 1.1GHz Athlon-based PC. Figure 2.12 shows the first two wavelet chosen by the system along with the tuning curves for those wavelets.

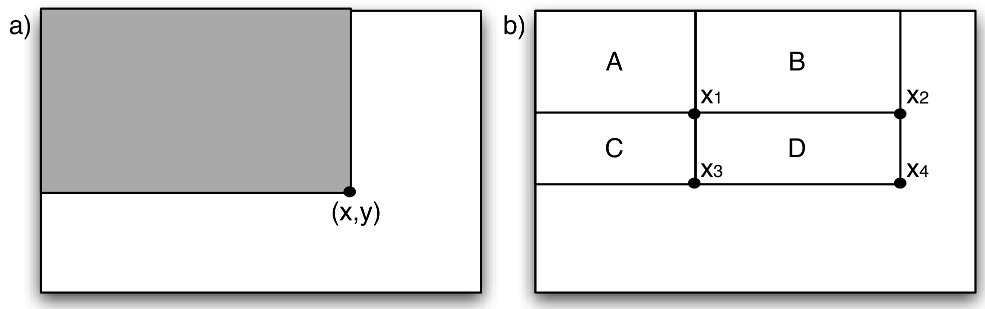


Figure 2.6: The Integral Image: (a) The value of the pixel at (x, y) is the sum of all the pixels above and to the left. (b) The sum of the pixels within rectangle D in the original image can be computed from points in the integral image by $x_4 - x_2 - x_3 + x_1$.

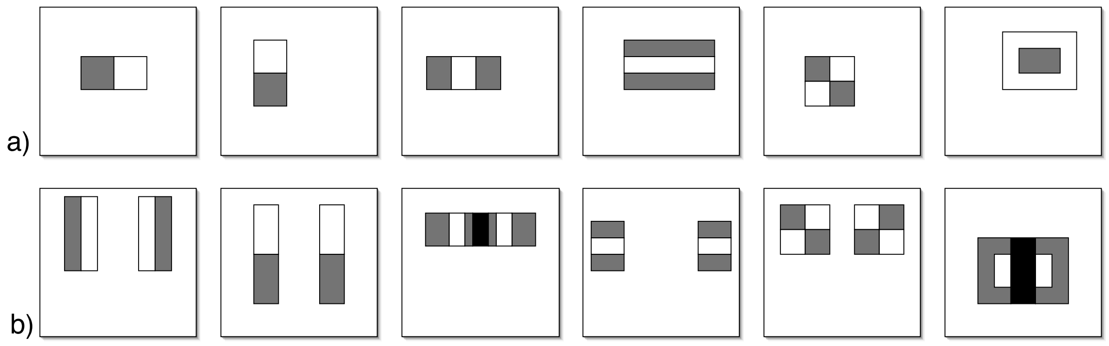


Figure 2.7: Each wavelet is computed by taking the difference of the sums of the pixels in the white boxes and grey boxes. (a) Wavelet types include those in Viola and Jones [2001], plus a center-surround type wavelet. (b) In the refinement step, the same wavelet types superimposed on their reflection about the Y axis are also possible.

At recognition time the inference algorithm calls for scanning the entire image plane and looking for square patches of arbitrary scale and location with large likelihood-ratios. In practice we start scanning patches of size 24×24, the minimum scale of interest and shift one pixel at a time until all possible patches of this size are scanned. Each larger scale is chosen to be 1.2 times the previous scale, and the corresponding offsets are scaled by the same proportion, for an additional $(n - 24s) \times (m - 24s)/s^2$ patches per scale. For a 640×480 pixel image, this produces over $400,000$ total patches.

Because the early layers in the cascade need very few wavelets to achieve good performance (the first stage can reject 60% of the non-faces using only 2 wavelets, using only 20 simple operations), the average number of wavelets that need to be evaluated for each window is very small, making the overall system very fast while still maintaining high accuracy. The current system is capable of achieving 30fps on images of 320×240 on a 3 GHz Intel Pentium 4-based desktop PC, with a minimum face size of about 24×24 pixels. Performance on the CMU-MIT data set (a standard, public data set for benchmarking frontal face detection systems) is comparable to other state-of-the art systems. For the experiments in this paper, the parameters for the face detector were chosen to yield a 91% hit rate with 10 false alarms when tested against all the images in the CMU-MIT data set. While CMU-MIT contains wide variability in images due to illumination, occlusions, shadows, and differences in image quality, the performance in controlled environments, such as in the BioID data set (used later in this study), containing faces that are frontal, focused and well lit, with simple background, is often close to 100% hit rate or frontal faces with few, if any, false alarms. While performance falls off as the face deviates from frontal (see Section 2.5.2), there are a wide variety of applications, in particular those in which the subject is watching a screen or driving on a road for example, for which frontal-view accuracy is sufficient. We discuss the ways to overcome this limitation in Section 2.6. Source code for this stage is available at `http://mplab.ucsd.edu`.

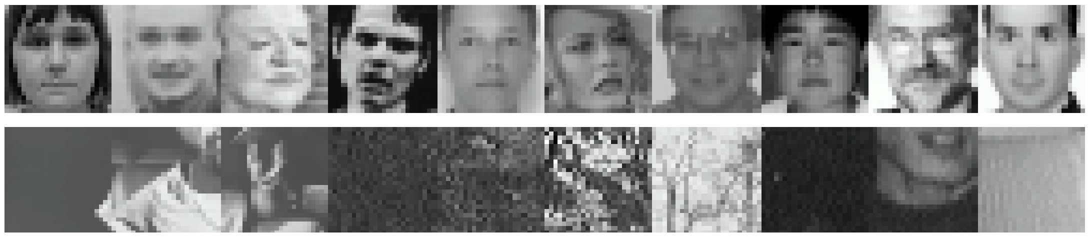


Figure 2.8: Examples of faces and nonfaces used in training the face detector

2.4.2 Stage II: Eye Detection in the Context of Faces

The first stage in the eye detection system specialized in finding general regions of the image plane that are highly likely to contain eyes. The output of the system is very resistant to false alarms but does not specify well the precise location of the eyes. The second stage specializes in achieving high accuracy provided it operates on the regions selected by the previous stage. This stage uses the same searching techniques as the previous stage: all patches at multiple scales, within a sub-region of the face restricted

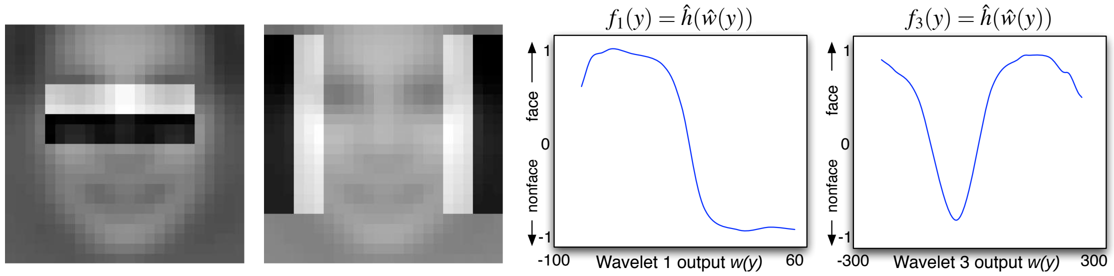


Figure 2.9: The first two wavelets (left) and their respective tuning curves (right) for face detection. Each wavelet is shown over the average face. The tuning curves show the evidence for face (high) *vs.* . non-face (low), as a function of the output of the wavelet, shown increasing from left to right. The first tuning curve shows that a dark horizontal region over a bright horizontal region in the center of the window is evidence for an eye, and for non-eye otherwise. The second tuning curve is bimodal, with high contrast at the sides of the window evidence for a face, and low contrast evidence for nonface.

in both location and scale, are submitted to a function (trained via GentleBoost) which returns the eye versus non-eye log-likelihood ratio. This log-likelihood ratio is then combined with the prior for probability of eye given location and size with respect to the face detection window to produce a final log posterior ratio of eye versus non-eye.

The data used for training was from the CMU-MIT face database and the Compaq face database used for training the face detection system. These images varied widely in image quality, lighting condition, background, facial expression, head size and orientation, head size (with respect to the image), and image quality, and contain faces with eyes closed as well as open. Positive examples were selected by cropping patches from each image such that they contain eyes at a canonical scale and location with respect to their face (described below), then scaling the patch to 24×24 pixels. Non-eye examples were taken from the same images at multiple non-eye locations and scales within the faces, with constraints described below. This resulted in 4826 positive eye examples and 10000 non-eye examples.

There are many possible ways to crop and center the eye patches for training. We present experimental results of several different choices of cropping and centering. We can parameterize the choice by introducing variables d: the distance between the eyes, r: the ratio of the distance between the center of the eye and the left and upper edges of the face cropping window, t: an offset parameter, and q: a scale parameter. Positive training samples were then prepared by cropping example images such that $r = q(d+td)$

and scaling them to 24×24 pixels. In other words, the size of the window was chosen to be proportional to the distance between the eyes, and could be off center by some fixed amount. Thus, a small q results in a small receptive field with high resolution and a large q results in a large receptive field with relatively low resolution, while t shifts the location of the eye with respect to the center of the patch.

From the situational inference approach, one might expect that pixels which are generated by background contain relatively little additional information about eye location once the location of the face is known, thus we should choose a t and q that maximizes the number of pixels in the positive example patches that are generated by face – i.e., about the size of the face and centered on the center of the face (i.e., the eye is off-center slightly), so that very few background pixels enter into the window. However, given a fixed input size of 24×24, it is possible that smaller values of q, such as one that just covers the eye (resulting in higher resolution examples with less surrounding context) allow us to maximally benefit from the information in pixels generated by the eye only. We present results on varying these parameters experimentally to find the best choice of offset parameter t and scale parameter q in section 2.5.

The situational inference approach also allows us to constrain how we choose non-eye examples: We model our prior belief about the eye location π as a normal distribution, with parameters for the mean and standard deviation of the true eye position and scale with respect to the window chosen by the face detector, as measured against the training set. In figure 2.10, we show the locations of eyes with respect to the size of the face detection window for some example data. Down on the vertical axis shows increasing ratio of the size of the face detection window to the distance between the eyes. When the face detector selects a small window relative to the true face size, resulting in a small detection width to eye distance ratio, the eyes tend to be far apart with respect to the detection window. When the face detector selects a large window compared to the distance between the eyes, the eyes tend to be located closer together, near the center of the detection window.

Using these statistics about the true eye positions with respect to the estimated face location, we can restrict the set of patches for searching – and thus for training – to a maximum Mahalanobis distance M from the mean location and scale of each eye. Choosing $M = 16.27$ gives a 99.9% confidence interval for one of the patches containing the eye (see Appendix 2.A).

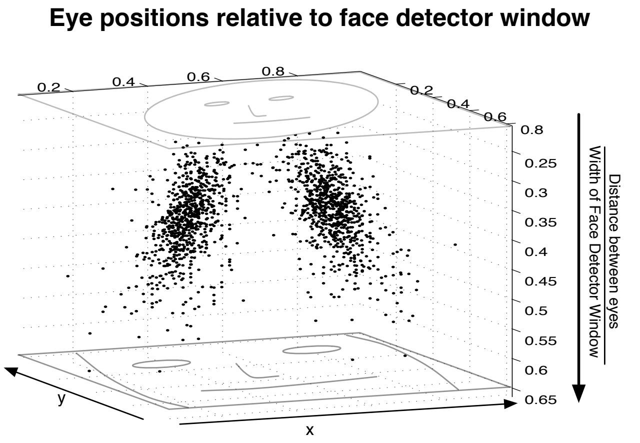


Figure 2.10: The face detection window can vary from closely cropping the face (negative z-axis) to loosely cropping the face (positive z-axis). The points show typical eye locations relative to the face detection window over a sample database of face images. We model this variability with a three-dimensional Gaussian, where the x- and y-axes are space, and the z-axis is scale, i.e., ratio of distance between eyes to size of the face-detector window. We use this to model the prior probability of a location containing an eye given the face detection window.

Using these criteria, for each example face, we created two positive training examples (one for each eye), and six negative training examples, where the negative examples were selected randomly from the set of patches satisfying the maximum distance from the mean eye patch size and location criterion. To make best use of our data, we flipped the positive and negative examples from the right eye about the vertical axis and combined them with the left eye examples to train a single left eye detector. Then this left eye detector was flipped about the vertical axis to get a right eye detector. Examples of eyes and non-eyes used in training is shown in Figure 2.11.

Once we have collected a set of positive and negative examples, we train this stage of the situational inference cascade with GentleBoost as described above. We found that it is possible to achieve excellent performance with only 50-100 wavelets without overfitting, as tested on a validation set. Since this already allows the system to operate in real-time with high accuracy, we did not use the attentional-cascade and boot-strap techniques needed for training the context-free face detector. Figure 2.9 shows example wavelets and their corresponding tuning curves for the best eye-detector.

While Stage I of our system (face detection) makes no assumptions about the number of faces in the image plane, the second Stage (precise location of the eyes)

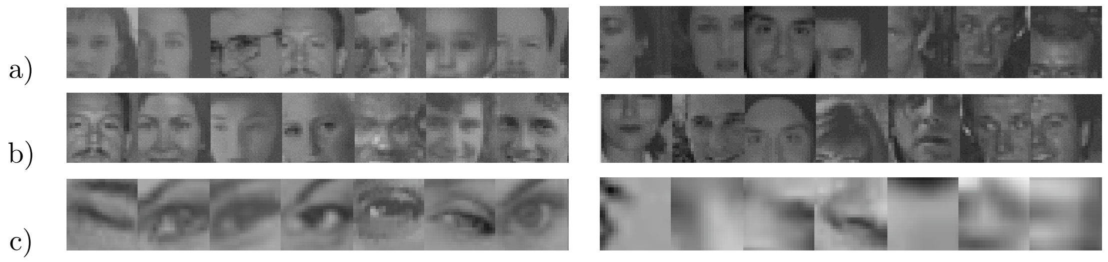


Figure 2.11: Examples of positive (left) and negative (right) example patches used for training three different eye detectors. Each patch is 24×24 pixels. (a) For this detector, positive examples were chosen centered on the eye ($t = 0$), with scaling factor $q = 1$. (b) This detector uses the same scaling factor in (a), but with offset parameter t chosen such that the eye is off center to maximize pixels generated by face. (c) With a smaller value of $q = .22$, the eye fills the window.

assumes that there is one patch rendering the left eye and one patch rendering the right eye. If the goal is to maximize the probability of choosing the correct rendering patch optimal inference requires choosing the patch that maximizes the log posterior ratio (2.21). However if the goal is to minimize the expected squared distance from the eye, optimal inference asks for computing the mean of the posterior distribution. Both approaches can be seen as examples of a more general algorithm that chooses the N patches with highest log posterior ratios and producing a weighted average of the opinions of those patches about the location of the feature of interest. In Section 2.5 we present accuracy results using different values of N.

2.4.3 Stage III: Blink Detection

Like face detection and eye detection, blink detection is done with a boosted classifier. In this case, the task is a binary classification task over a single patch per image, thus there is no need to perform a search across multiple patches. Instead, we use estimates of the eye locations to create a 44×22 pixel patch containing the eyes, doing scaling and rotation with simple linear interpolation. Training data was collected from 120 eye-open images and 120 eye-closed images collected from the Web by using the eye detector to label the eye locations, then cropping and rotating the region around the eyes to an upright frontal view. The data set is available at `http://mplab.ucsd.edu`. Figure 2.13 shows examples of the training data collected this way. GentleBoost is then used to select wavelets and tuning curves for this discrimination task. Figure 2.14 shows example wavelets and their corresponding tuning curves for the best blink detector.

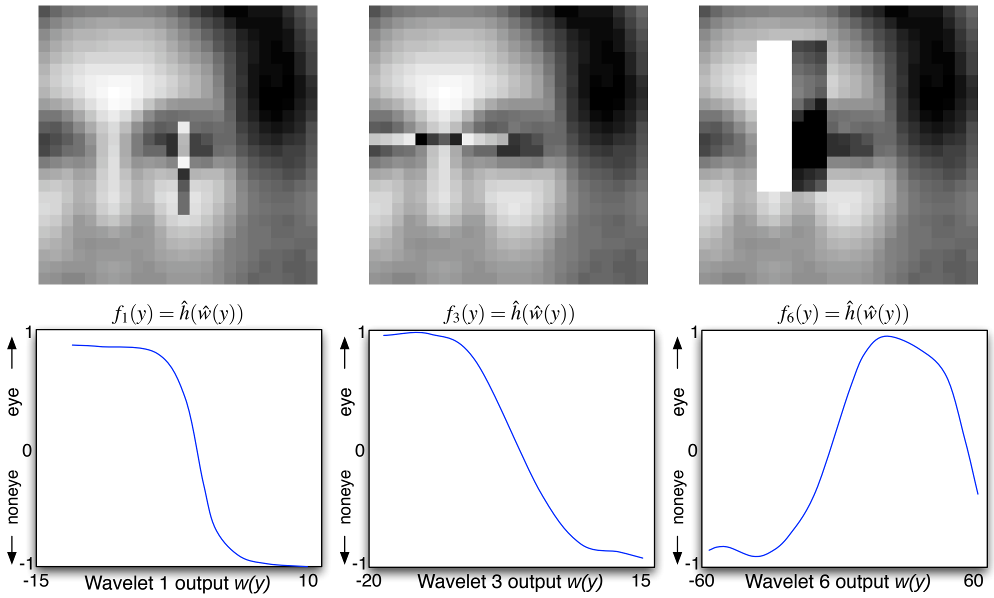


Figure 2.12: The first, third, and sixth wavelets (top) and their respective tuning curves (bottom) for the left eye detector centered on the eye with scale factor $q = 1$. Each wavelet is shown over the average positive (eye) example. The tuning curves show the evidence for eye (high) *vs.* . non-eye (low) as the wavelet output increases (shown increasing from left to right). The first tuning curve shows that a dark vertical region over a bright vertical region in the center of the window is evidence for an eye, and for non-eye otherwise. The middle tuning curve looks for a horizontal band that goes dark-light-dark towards the left of the window as evidence for an eye, which appears to be testing for the bridge of the nose. The rightmost wavelet also can be interpreted as a bridge of the nose detector, however it also indicates that *too much* difference between the left and right parts of the wavelet are evidence *against* eye.

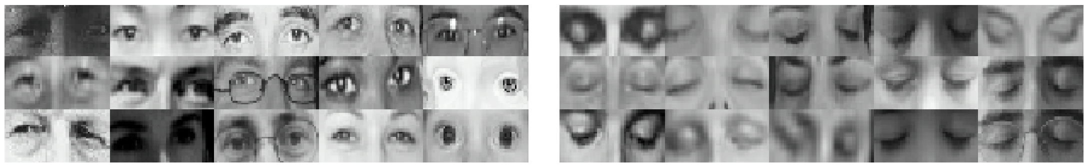


Figure 2.13: Example open eyes (left) and closed eyes (right) used to train the blink detector. About 120 images of each type were taken from the web to include a wide variety of lighting conditions, facial types, glasses, and image quality. The eye detection system was used to automatically crop, scale and rotate the image patches to an upright frontal view.

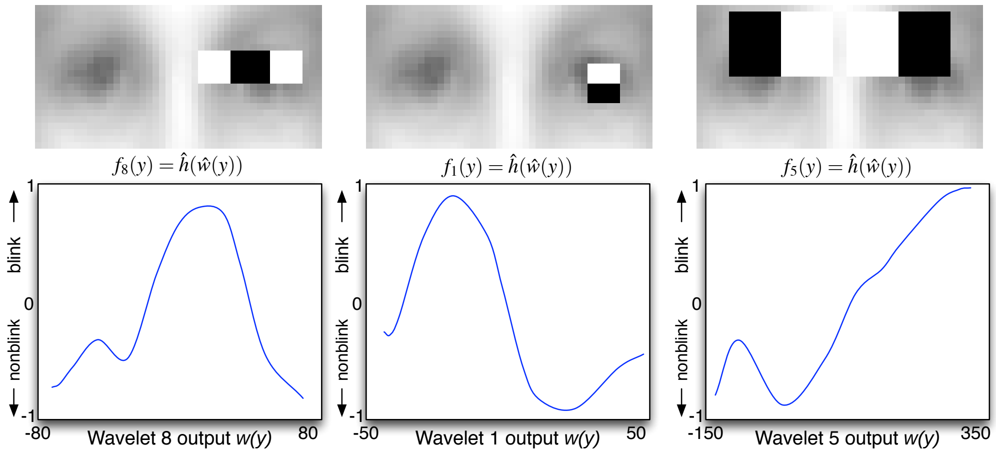


Figure 2.14: Features superimposed on the average open eye image (top) and their respective tuning curves (bottom) for the blink detector.

2.5 Experimental Results

2.5.1 Testing Data Sets

We tested the performance of the eye detector on three different types of data sets. The first data set was the BioID data set [Frischholz and Dieckmann, 2000; Jesorsky et al., 2001], a freely available collection of face images with eyes labeled. This data set contains 1521 images with good lighting conditions and frontal faces, and most subjects had their eyes open. This was to make it easier to compare our results with other eye-detection systems. The second data set was more challenging, consisting of 400 images collected from the Web and digital cameras. We are making this data set available at `http://mplab.ucsd.edu`. These images varied widely in image quality, lighting condition, background, facial expression, head orientation, head size (with respect to the image), and image quality, and contained 200 eyes-open and 200 eyes-closed examples. Measuring performance on this data set allows us to compare how different parameter choices affect the quality of the system in unconstrained situations. We believe that if one can achieve good performance in this highly unconstrained data set, then one can expect very good performance in better controlled situations. None of the images in this testing data set were used during training.

The third data set consisted of ten different heads in 153 different poses each, artificially generated from the USF Human ID 3-D database [Blanz and Vetter, 1999]. Each head in the database, obtained using a laser scanner, contains structure (3D coordinates) and texture (24-bit RGB color) information for each point on the surface, suitable for rendering a high-quality still of the face at any position. Each of the ten randomly chosen heads we used for our experiments was positioned from -40 to 40 degrees in elevation and 0 to 40 degrees in azimuth, in increments of 5 degrees, then rendered. This data set was used to provide an estimate of the performance of the face detection and eye detection components of the system as the pose was varied.

2.5.2 Eye Detection Experiments

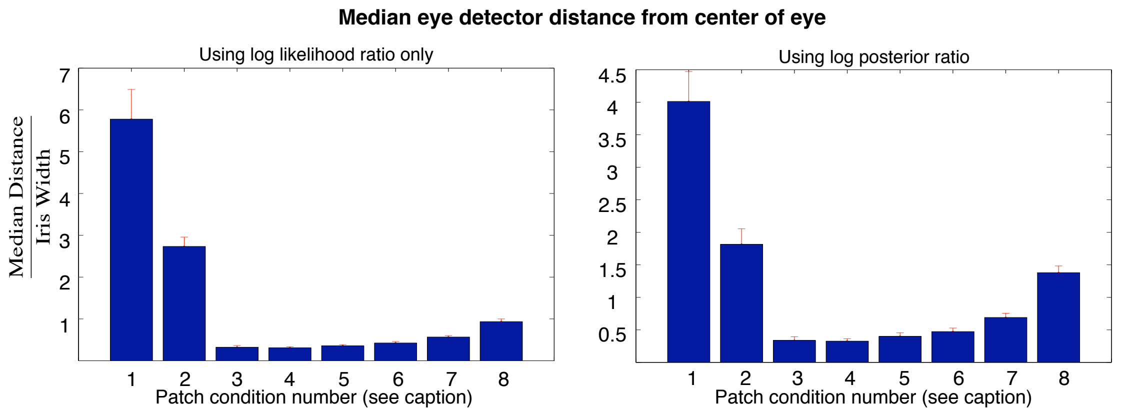


Figure 2.15: Median distance from center of labeled eye positions on the Web data-set as the scale parameter q and offset parameter t are varied. The graphs show the result using only the log likelihood ratio (left) and the log posterior ratio, which combines the prior and likelihood (right). The conditions, described in Section 6.2, are (1) $q = .11$, eye centered, (2) $q = .22$, eye-centered, (3) $q = .5$, eye-centered, (4) $q = 1$, eye-centered, (5) $q = 1$, face-centered, (6) $q = 1.5$, eye-centered, (7) $q = 1.5$, face-centered, (8) $q = 2.5$, eye-centered.

We tested the effect of the size and location of the receptive field used for eye detection. The receptive field size was expressed as the ratio q of the distance between the eyes. Location was expressed as "face-centered" or "eye-centered". Varying patch size from small enough to cover just the iris ($q = .11$) to large enough to cover an area four times the size of the head ($q = 2.5$) results in a U-shaped curve, with the best performance coming from the patch with size $q = 1$, which covers about 80% of the face. The best centering condition was eye-centered. The median accuracy of the best eye-detector under these conditions is 1/5 of an iris on the BioID data set and 1/3 of an iris on the difficult data set from the Web. Tables 2.2 and 2.3 show the results for each patch condition using different decision methods. These include choosing the maximum likelihood patch, taking the weighted average of the 10 most likely patches, taking the maximum posterior patch, and taking the weighted average of the 10 patches with the largest posterior. The fourth technique yielded the best results. Figure 2.17 shows examples of this system at work.

The fact that the detector trained to consider pixels covering much of the head performs much better than the detector trained to focus on the eye-area only suggests that the detailed structure of the appearance of the eye (which at the larger resolution is mostly blurred out) is not as important as having access to the surrounding features (nose, eyebrows, corners of eyes, etc). This may be because for the larger receptive field, dark shadows, closed versus open eyes, or specularities from glasses have less impact on the overall visual appearance of the pixels under consideration than for the detector that only focuses only on the pixels generated by the eyeball and eyelid. On the other hand, a receptive field that is much larger than the face seems to lose the ability to discriminate much detail in the face, and may be confused by many background pixels which actually have no information about the location of the eye within the face.

The performance on the data set generated from the 3-D database illuminates how performance changes as head-pose changes. As seen in figure 2.16, the face detector achieves about 92% for fully frontal faces (comparable to its performance on CMU-MIT), and falls off smoothly as the head deviates from frontal view. However, provided the head is detected in the first place, accuracy on eye-detection is not strongly degraded from 1/3 of an iris width as pose changes from frontal. Indeed, if elevation and azimuth is kept between ± 20 degrees, median distance from the center of the labeled eye position remains nearly constant.

Table 2.2: Results on the BioID data set of eye detection under different choices of patch size, offset and post-processing (mean or max of log-likelihood or log-posterior ratio). Each cell displays the mean distance, in irises, from the true center of the eye to the estimated center of the eye. The ± terms indicate standard error of the mean. The post-processing is explained in Section 5.2. The patch conditions are described in Section 6.2.

post processing	$q = .11$ eye-centered	$q = .22$ eye-centered	$q = .5$ eye-centered	$q = 1$ eye-centered	$q = 1$ face-centered	$q = 1.5$ eye-centered	$q = 1.5$ face-centered	$q = 2.5$ eye-centered
max log-lik ratio	4.66 ± 0.19	2.25 ± 0.14	0.30 ± 0.03	0.27 ± 0.01	0.41 ± 0.02	0.35 ± 0.02	0.59 ± 0.05	1.33 ± 0.06
mean log-lik ratio	3.40 ± 0.23	2.07 ± 0.16	0.24 ± 0.04	0.21 ± 0.02	0.33 ± 0.02	0.31 ± 0.03	0.65 ± 0.04	1.26 ± 0.06
max log-post ratio	10.43 ± 0.34	2.68 ± 0.11	0.29 ± 0.02	0.26 ± 0.01	0.41 ± 0.02	0.36 ± 0.01	0.55 ± 0.02	0.96 ± 0.03
mean log-post ratio	9.47 ± 0.45	2.81 ± 0.16	0.24 ± 0.03	0.21 ± 0.01	0.31 ± 0.02	0.28 ± 0.02	0.55 ± 0.02	0.89 ± 0.04

Table 2.3: Results on the Web data set of eye detection under the same conditions as Table 2.2

post processing	$q = .11$ eye-centered	$q = .22$ eye-centered	$q = .5$ eye-centered	$q = 1$ eye-centered	$q = 1$ face-centered	$q = 1.5$ eye-centered	$q = 1.5$ face-centered	$q = 2.5$ eye-centered
max log-lik ratio	4.64 ± 0.38	2.13 ± 0.19	0.38 ± 0.04	0.37 ± 0.03	0.48 ± 0.05	0.52 ± 0.05	0.67 ± 0.06	1.35 ± 0.10
mean log-lik ratio	4.01 ± 0.46	1.82 ± 0.24	0.34 ± 0.05	0.33 ± 0.03	0.40 ± 0.05	0.47 ± 0.06	0.69 ± 0.06	1.38 ± 0.10
max log-post ratio	6.28 ± 0.75	2.81 ± 0.23	0.38 ± 0.05	0.36 ± 0.03	0.43 ± 0.03	0.50 ± 0.04	0.60 ± 0.04	1.00 ± 0.07
mean log-post ratio	5.78 ± 0.71	2.73 ± 0.22	0.32 ± 0.04	0.31 ± 0.02	0.36 ± 0.03	0.42 ± 0.04	0.57 ± 0.03	0.94 ± 0.06

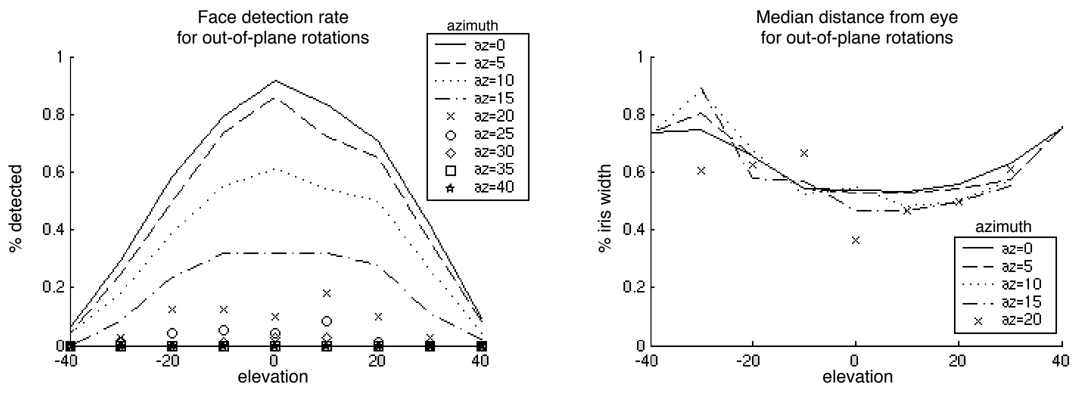


Figure 2.16: Performance for face detection and eye detection as pose changes. Each curve shows performance for heads at a fixed azimuth as the elevation is varied from -40 to 40 degrees. (Left) Face detection rate falls off as pose deviates from frontal. (Right) Median distance from the true eye label remains nearly constant for heads between ±20 degrees from frontal.

2.5.3 Blink detection

The best performing eye detection, with scale parameter $q = 1$ and zero offset from the center of the eye, was used to automatically crop, scale and rotate 120 examples of closed eyes and open eyes. These examples were used to train a blink detector. We stopped training after 500 wavelets and tuning curves had been chosen. The resulting classifier was then used to classify an additional 120 eyes-open and eyes-closed faces taken from the web and labeled by hand.

To assess the effects of precise localization of the eyes we compared systems that found the eyes based on the output of Stage I alone (face detection) and systems that located the eyes using Stage I and II. The effects were dramatic: adding stage II increased performance from $56.53\% \pm 8\%$ to $83.48\% \pm 6\%$.

2.6 Conclusions

The study of the representations that sustain face perception in humans has recently become a subject of interest in cognitive science [Cottrell et al., 2003]. One heated debate centers on whether these representations are holistic in nature or whether they are are feature based [Farah et al., 1988]. In line with the methodological stand of probabilistic functionalism [Movellan and Nelson, 2001] instead of positioning ourselves in this debate we focus on understanding the nature of the problem of detecting faces and facial features. To do so we developed an image generation model and derived its corresponding optimal inference algorithm. The algorithm was implemented and tested with an emphasis on robustness under natural conditions. We learned several important lessons:

(1) We found that it is difficult to analyze facial behavior (e.g., blinks) without explicitly localizing the eyes. Based on our previous work on expression recognition we think eye localization with precision on the order of 1/4 of an iris width may be necessary for reliable recognition of facial expressions. Thus it seems reasonable to expect that the brain may allocate resources to precisely locate facial features, including the eyes.

(2) We found that it is difficult to develop detectors that are both robust (i.e., work in very general conditions) and spatially accurate. There seems to be a trade-off between robustness and accuracy. Eye detectors that localize the eyes precisely within the face exhibit unacceptable false-alarm rates when operating outside the face. Eye

detectors that avoid false-alarm rates in cluttered environments, are not sufficiently precise about the location of the eyes. We explored a solution to this tradeoff, based on a cascade of detectors that operate at different levels in the robustness/localization trade-off. Some of these detectors capture the general context in which one may find eyes. By doing so they minimize false alarms at the cost of precise position information. Precise spatial localization is achieved by detectors that operate in specific contexts. If this is the strategy adopted by the brain, one would expect to find at least two types of neurons. The first type would respond to large contextual regions (e.g., faces). Neurons of this type are expected to be robust to changes in illumination but also to provide poor spatial resolution. We also expect to find a second type of neurons specialized on precise spatial localization of features in specific contexts. For example, neurons of this type may be maximally excited by eyes precisely aligned and maximally inhibited by small deviations from alignment. This second type of neurons may exhibit a large number of false alarms when operating out of context, making it very difficult for neuroscientists to ascertain what they respond to.

(3) In this paper we developed the necessary likelihood-ratio and prior models using supervised learning methods. It would be of interest to investigate whether such models can be learned using unsupervised learning methods. Another possibility is that evolution took care of developing such models. Provided a set of useful wavelets is available, our face detector would require in the order of 50 Kbytes to be encoded by the genome. It takes an additional 2 KBytes to encode eye detectors within faces.

(4) We focused on a system specialized on detection of eyes in a particular pose: upright frontal. In many cases (e.g., detection of fatigue in car drivers) analysis of upright-frontal views is all that is needed since frontal orientations are nominal and deviations from such orientation typically indicate fatigue or lack of attention [Ji and Yang, 2002]. In-plane rotation invariance can be easily achieved by scanning across rotations, in the same way we scan across scales and in-plane locations. There are several ways one could generalize the system to work under rotations in depth. One approach we experimented with in the past fits 3D morphable models and warps them into frontal views [Bartlett et al., 2003]. While this method is very effective under controlled illumination conditions, it is expensive computationally and brittle when exposed to outdoor conditions. Another approach we are pursuing is a mixture of experts architecture, where each expert specializes on specific face views. Indeed there is experimental evidence for

the existence of view specific face detection neurons in infero-temporal cortex (IT) in monkeys [Logothetis and Poggio, 1994]. Due to rotational symmetry of the face, pose invariance can be achieved by covering an octant of the sphere of possible face orientations, i.e., $\pi/2$ steradians. Assuming each pose expert can handle $\pm 5^\circ$, as is the case for the system presented here, it would take approximately $1/(2\tan(5)) \approx 6$ experts to cover an octant. This is certainly not an unreasonable number of experts, thus making mixtures of pose experts a very attractive architecture for future systems. Development of systems specialized in non-frontal views is currently difficult due to the lack of labeled data sets that include sufficient number of images in multiple poses and illumination conditions. Collecting such databases is critical to accelerate progress in this field.

The text of this chapter, in part, is a reprint of the material as it appears in *Computer Vision and Image Understanding*, Ian Fasel, Bret Fortenberry, Javier Movellan, 2005, vol.1. Ian Fasel was the primary author and the second co-author listed in this publication directed and supervised the research which forms the basis for this chapter.

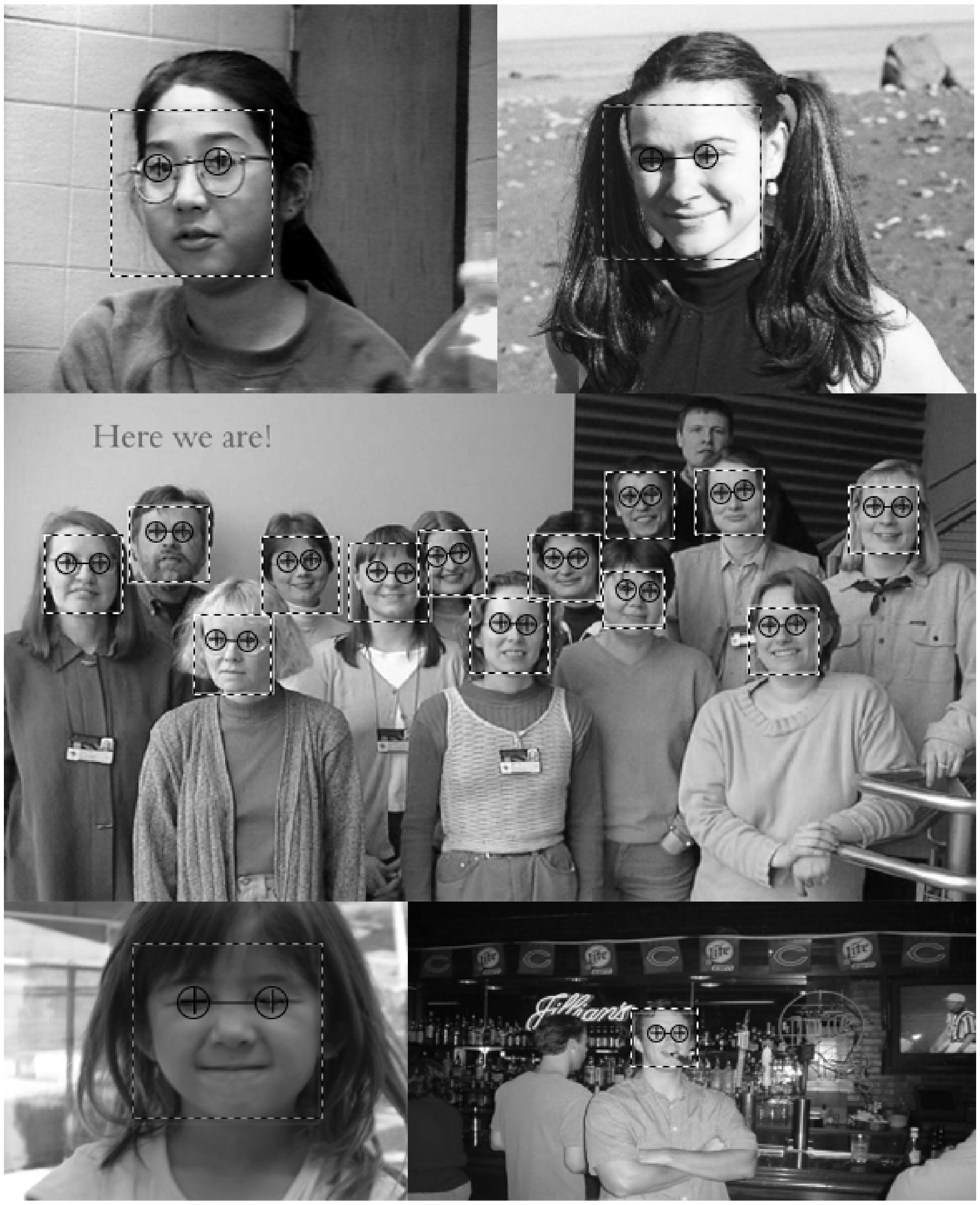


Figure 2.17: Examples of the eye detector results

2.A Gaussian Confidence Regions

Let Z be n-d Gaussian, zero mean with covariance I_n. Let σ a covariance matrix, with eigenvectors p and eigenvalues λ, i.e. $\sigma = p\lambda p^T$. Let $\mu \in \mathbb{R}^n$. Let $Y = p(\lambda)^{1/2}Z + \mu$. Thus Y is Gaussian with covariance Σ and mean μ.

For a given $\alpha > 0$ We want the probability that $(Y-\mu)^T\Sigma^{-1}(Y-\mu)$ takes values smaller or equal to α. Now note

$$(Y-\mu)^T\Sigma^{-1}(Y-\mu) = Z^TZ = \sum_{i=1}^{n} Z_i^2 \tag{A-1}$$

which is a chi-square random variable with n degrees of freedom. This is the key to obtaining confidence intervals.

2.A.1 Example

Suppose $n = 3$, Y is Gaussian with mean μ and covariance σ, and we want to calculate the value α such that

$$P((Y-\mu)^T\sigma^{-1}(Y-\mu) < \alpha) = 0.001,$$

i.e., we want a volume that captures 99.9 % of the probability. First we go to the chi-square distribution with 3 degrees of freedom and find that the critical value for 1/1000 is 16.27. Thus

$$P((Y-\mu)^T\sigma^{-1}(Y-\mu) < 16.27) = P(Z^TZ < 16.27) = 1/1000. \tag{A-2}$$

Thus the 99.9 % confidence region for Y is given by the set of values y such that

$$(y-\mu)^T\sigma^{-1}(y-\mu) \leq 16.27. \tag{A-3}$$

3

Weakly Supervised Robust Real-Time Object Detection

3.1 Introduction

In Chapter 2, we described a method for learning real-time object detectors using a set of example images in which a human had labeled the locations of the objects in each image. In this chapter, our goal is to develop a system which learns to detect the presence and location of objects in real-time using example images labeled only as containing or not containing the object of interest. This problem has been referred to as an "unsupervised" learning problem [Weber et al., 2000; Fergus et al., 2003] to contrast it with supervised learning approaches in which the location of objects in training examples must be known. Because there is some label information present during training we instead refer to this as a "weakly-supervised" learning task.

Most principled approaches to unsupervised or weakly-supervised learning are based on probabilistic generative models. Unfortunately, the complexity of such models can quickly become intractable as the size of problems grow, thus in practice underlying model distributions are typically restricted to e.g., mixtures of Gaussian distributions (for continuous variables) or Dirichlet distributions (for discrete variables, e.g., Weber et al. [2000]; Barnard et al. [2003]; Sudderth et al. [2005]), and it is often necessary to resort to approximate techniques such as variational EM [Ghahramani and Beal, 2000] or loopy belief propagation [Pearl, 1988; Weiss, 1997] to perform learning and inference [Agosta, 1988; Fei-Fei et al., 2004; Jojic and Frey, 2001; Winn and Jojic, 2005]. These issues greatly limit the application of fully generative techniques in many realistic computer vision problems.

Discriminative methods such as SVMs [Vapnik, 1995] or AdaBoost [Freund and Schapire, 1996a] are a useful alternative to generative approaches as they are explicitly designed to solve classification tasks without the simplifying assumptions needed by generative models. Discriminative approaches have been used very successfully to develop highly-accurate, real-time performance in problems such as face and car detection [Sung and Poggio, 1998; Rowley et al., 1998; Schneiderman, 2000; Jones and Viola, 2003; Dalai and Triggs, 2005]. The drawback of these methods is that to be successful large amounts of labeled data must be provided, which may be very costly or impossible to obtain. Due to this fact only a few high-quality systems capable of deployment in relatively unconstrained environments have been developed for a handful of very specific vision problems.

In this chapter, we explore a way to use discriminative approaches within a generative framework. We refine the model presented in Chapter 2 and show how it can be used for learning in a weakly-supervised manner. Under the model, scenes consist of sets of pixels each of which are generated by a different object category. The scene generation consists of a random partitioning of the image into segments, assignment of object categories to each segment, and independent rendering of each segment given the assigned object categories. This rendering is controlled by a family of parameterized distributions which, following the Bayesian approach, are also treated as random variables. This process is illustrated in Figure 3.1.

Inference consists of discovering the causes underlying an observed collection of images. Within the Bayesian framework this simply requires application of Bayes rule

$$p(causes \mid observed\ image) \propto p(causes)p(observed\ image \mid causes) \tag{3.1}$$

where $p(causes)$ represent beliefs about the distribution of causes prior to the observation of the data, $p(causes \mid observed\ image)$ is the re-evaluation of those beliefs after the image has been observed, and $p(observed\ image \mid causes)$ is given by the generative model. There are two types of inference problems of interest:

- Discovery of the underlying distribution parameters. This is typically referred to as the *learning problem*. In this case we are typically given a large collection of images and our goal is to find the most probable distribution parameters, marginalizing over all the other causal variables. Because this process typically operates on large image data sets, it is seen as operating at a long time scale (e.g., minutes to days).

- Image classification and segmentation. In this case we are typically given an image and a fixed set of parameters and our goal is to infer how the image was generated. The classification task requires deciding whether or not the image contains an object of a given category. The segmentation task requires deciding where the objects are located and what type of objects they are. Under this model, the segmentation problem consists of finding collections of dependent pixels which are independent of all the other pixels. This abstract definition of the segmentation task contrasts with other approaches that rely on concepts such as edges, texture, connected-components, etc., and is easy to generalize to non-visual domains. e.g., audition, olfaction, touch, or proprioception.

3.2 Formalization of the Scene Generation Model

Notational Conventions: *Unless otherwise stated, capital Roman letters represent random variables and lower-case Roman letters represent specific values taken by those variables. When possible, we use informal shorthand notation to suppress explicit reference to the probability space on which random variables and probability densities are defined. We indicate the domain of random variables using the "$\in$" symbol, for instance, $X \in \mathbb{R}$ means the function $X : \Omega \to \mathbb{R}$ for an outcome space Ω. When unambiguous, we also identify probability functions by their arguments, for example, $p(y)$ is shorthand for the probability (or probability density) that the random variable Y takes the specific value y, i.e., $p(y) \overset{def}{=} P(Y = y) \overset{def}{=} P(\{\Omega : Y(\omega) = y\})$.*

Let an *image* X be a collection of $n_p \in \mathbb{Z}_+$ pixel intensities

$$X = (X^1, X^2, ..., X^{n_p}), \qquad X \in \mathbb{R}_+^{n_p} \tag{3.2}$$

Let s_i be a partition of $\{1, ..., n_p\}$. We will refer to each partition as a *segmentation*. Let

$$\mathcal{S} = \{s_1, ..., s_{n_k}\}, \qquad n_k \in \mathbb{Z}_+ \tag{3.3}$$

be a collection of possible segmentations. The elements of each segmentation s_i are sets

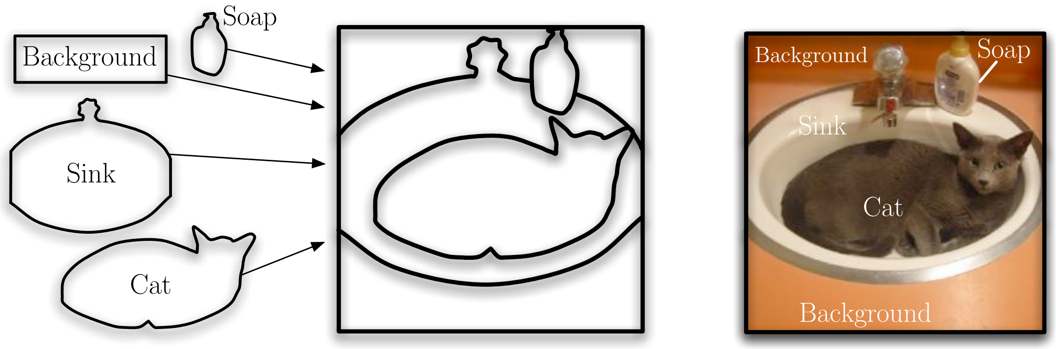


Figure 3.1: The scene generation process. First a set of objects are chosen and arranged with respect to the camera, inducing a *segmentation*. Then, pixel intensities are chosen by drawing from a distribution conditioned on the segment category and shape. The definition of an image segment is thus a set of pixels that are dependent, but are independent of the rest of the pixels in the image.

of pixel indexes, which we call *segments* (See Figure 3.2). Let

$$\mathcal{A} = \{a_1, ..., a_{n_s}\} = \bigcup_{i=1}^{n_k} s_i. \tag{3.4}$$

be the collection of all the different segments used by all of the segmentations in $\mathcal{S}$, thus each a_i is a segment. (Note that $n_s \leq 2^{n_p}$, since at most $\mathcal{A}$ could be the power set of $\{1, ..., n_p\}$). Let $|a_i|$ be the number of elements in a_i. Let

$$X_i = \{X^j : j \in a_i\}, \qquad X_i \in \mathbb{R}^{|a_i|} \tag{3.5}$$

be the set of pixels in X indexed by a_i. We will refer to X_i as an *image segment* to distinguish it from the *segment* a_i, which is a collection of pixel indexes, however when the context makes it clear we may simply use the term *segment*. Note under our notation X_i is not a scalar but a vector with as many pixels as elements in the segment a_i. Let $n_c \geq 2$ be an integer number of object categories, and let

$$O = (O_1, ..., O_{n_s}), \qquad O \in \{0, ..., n_c\}^{n_s} \tag{3.6}$$

represent *object assignments*. The object assignments identify which object category is responsible for rendering each image segment. If $O_i = c \neq 0$, then the segment a_i is rendered by object category c. If $O_i = 0$, we say that the segment a_i is not rendered (See Figure 3.3).

To simplify presentation, we will hereafter focus on the case of $n_c = 2$, and refer to object category 1 as the *foreground* and object category 2 as the *background*. Gen-

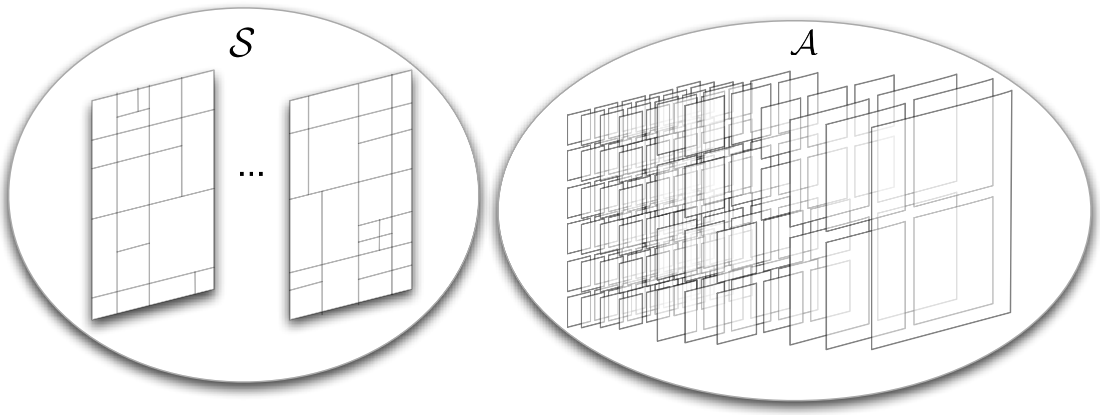


Figure 3.2: (Left) The set $\mathcal{S}$ consists of partitions of the pixel indexes $\{1, ..., n_p\}$, i.e, segmentations of the image. (Right) The set $\mathcal{A}$ is the collection of all the different segments in all segmentations in $\mathcal{S}$.

eralization to more than 2 categories is straightforward but obfuscates the presentation. Define the *pixel assignments* as

$$Y = (Y_1, ..., Y_{n_p}), \qquad Y \in \{1, ..., n_c\}^{n_p} \tag{3.7}$$

If $Y_i = c$, then pixel X^i is rendered by an object of category c. Note that Y_i cannot equal 0 because every pixel *must* be rendered by exactly one object (under the model objects are not transparent).

Let F represent parameters of the foreground rendering model, and let B represent parameters of the background rendering model. We treat each as a random variable which take values f and b on parameterized families of functions $\mathcal{F}$ and $\mathcal{B}$ respectively. Together f and b determine the probability distribution of pixel intensities given image segments and object category assignments. These are described in section 3.3.4. We are now ready to formalize the image generation process.

1. Choose a foreground rendering function f with uniform probability from the set of allowable rendering functions $\mathcal{F}$, and a background rendering function b with uniform probability from the set of allowable rendering functions $\mathcal{B}$.

2. Choose a *segmentation* $s \in \mathcal{S}$ with probability $p(s)$. Unless otherwise stated, let

$$p(s) = \frac{1}{|\mathcal{S}|} \tag{3.8}$$

where $|\mathcal{S}|$ is the number of segmentations. In other words, all segmentations have equal prior probability.

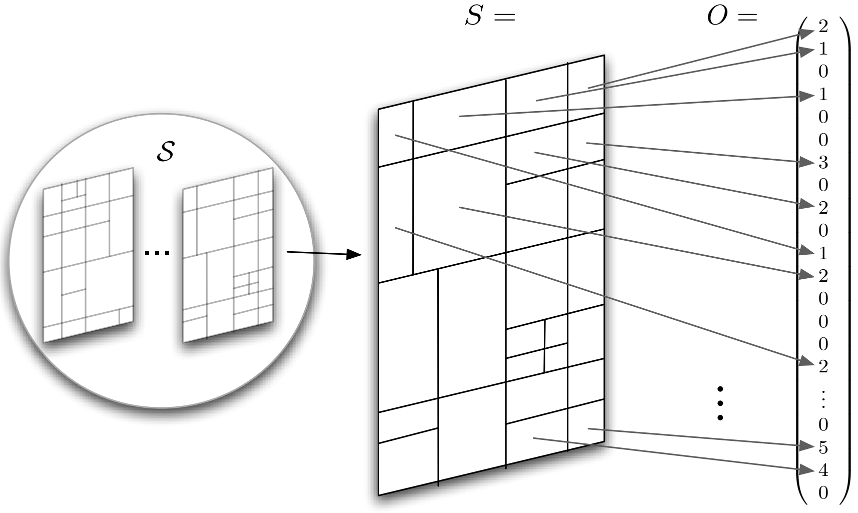


Figure 3.3: To create a scene, first a segmentation s is drawn from $\mathcal{S}$, then object assignment o is chosen given s. The object assignment is zero for any segment that is not part of the segmentation, while the nonzero entries indicate from which conditional distribution the pixel intensities in that segment are drawn. For this illustration we let $n_c = 5$, however for the remainder of this document we will restrict $n_c = 2$, i.e., O_i can only take on values $0, 1$ or 2.

3. Choose an *object assignment* o with probability $p(o|s)$, where

$$p(O_j = o_j|s) = \begin{cases} 1 & \text{if } \ o_j = 0 \text{ and } a_j \notin s, \\ \frac{1}{|s|} & \text{if } \ o_j = 1 \text{ and } a_j \in s, \\ 1 - \frac{1}{|s|} & \text{if } \ o_j = 2 \text{ and } a_j \in s \end{cases} \tag{3.9}$$

   where $|s|$ is the number of segments in s. In other words, all segments which are part of the segmentation s have equal prior probability of containing the foreground object.

4. Given object assignments, pixel category assignments are determined by

$$Y_j = \sum_{k=1}^{n_s} I_{a_k}(j)\ O_k \tag{3.10}$$

   where $I_{a_k}(j)$ is an indicator function, i.e.,

$$I_{a_k}(j) \overset{\text{def}}{=} \begin{cases} 1 & \textit{if } \ j \in a_k, \\ 0 & \text{otherwise,} \end{cases} \tag{3.11}$$

5. For all a_j, if $a_j \in s$, draw the *image segment* x_j from $p(x_j|o_j, f, b)$ as defined in Section 3.3.1.

We used uninformative priors for the segmentation and object locations in order to simplify the presentation, however generalizing to other priors is straightforward.

3.3 Likelihood function for entire images

Within a Bayesian framework, learning is simply a form of probabilistic inference. The goal of learning is to discover some model parameters f and b based on a set of example images. Critical to this process is finding an expression for $p(x \mid f, b)$, the likelihood of image x given the rendering functions f and b. Arguably the main contribution of our work is the specification of the form that this function takes under the proposed generative model. This allows us to use standard probabilistic methods to solve what nowadays are considered difficult computer vision problems. We will derive the likelihood function in several steps, first formulating image likelihoods given known object assignments, then marginalizing over object assignments given segmentations, and finally marginalizing over segmentations and object assignments.

3.3.1 Known segmentation, known object category assignments

Knowing the object category assignment o entails knowledge of the image segmentation s and the specific object category rendering each segment. Thus, under the model,

$$p(x|o, s, f, b) = p(x|o, f, b) = \prod_{\{j:o_j \neq 0\}} p(x_j|o_j, f, b) \tag{3.12}$$

$$= \prod_{\{j:o_j=1\}} p(x_j|O_j = 1, f, b) \prod_{\{j:o_j=2\}} p(x_j|O_j = 2, b) \tag{3.13}$$

That is, for each segment x_j, multiply by $p(x_j|O_j = 1, f, b)$ if the segment is rendered by foreground and multiply by $p(x_j|O_j = 2, b)$ if the segment is rendered by background. Note that in this model pixels are not independent – only *sets* of pixels corresponding to different segments are independent.

3.3.2 One foreground object, known segmentation, unknown object category assignments

In this case we know the segmentation s, and we know that exactly one of the segments is rendered by the foreground object, and the remaining segments are rendered by the background. However we do not know which specific segment is rendered by the object. Then

$$\begin{aligned} p(x|s,f,b) &= \sum_{\{j:a_j\in s\}} P(O_j=1,x|s,f,b) && (3.14)\\ &= \sum_{\{j:a_j\in s\}} P(O_j=1|s)p(x|O_j=1,s,f,b) && (3.15)\\ &= \frac{1}{|s|}\sum_{\{j:a_j\in s\}} p(x_j|O_j=1,f,b)\prod_{\{k:a_k\in s,k\neq j\}} p(x_k|O_k=2,b) && (3.16)\end{aligned}$$

where we used the fact that under the model the object can appear in all segments with equal probability, i.e., $P(O_j=1|s,f,b)=1/|s|$, where $|s|$ is the number of segments in the segmentation s. Provided $p(x_j|O_j=1,f)$ is absolutely continuous with respect to $p(x_j|O_j=2,b)$, we can multiply by $p(x_j|O_j=2,b)/p(x_j|O_j=2,b)$ to get

$$\begin{aligned} p(x|s,f) &= \frac{1}{|s|}\sum_{\{j:a_j\in s\}} \frac{p(x_j|O_j=1,f,b)}{p(x_j|O_j=2,b)}\prod_{\{k:a_k\in s\}} p(x_k|O_k=2,b) && (3.17)\\ &= \frac{1}{|s|}\sum_{\{j:a_j\in s\}} l(x|a_j,f,b)K^{(s)}(b) && (3.18)\end{aligned}$$

where

$$K^{(s)}(b) = \frac{1}{|s|}\prod_{k:a_k\in s} p(x_k \mid O_k=2,b) \geq 0 \qquad (3.19)$$

is the probability of the image given segmentation s and no foreground objects, and

$$l(x|a_j,f,b) = \frac{p(x_j|O_j=1,f,b)}{p(x_j|O_j=2,b)} \qquad (3.20)$$

is the likelihood ratio under the models f and b of image segment x_j.

3.3.3 One foreground object, unknown segmentation, unknown object category assignments

In this case we need to marginalize $p(x \mid s,f,b)$ over all possible segmentations

$$p(x|f,b) = \sum_s p(s)p(x|s,f,b) \qquad (3.21)$$

Thus, using (3.18) for $p(x|s,f,b)$, we have

$$p(x|f,b) = \frac{1}{|\mathcal{S}|}\sum_s K^{(s)}(b) \sum_{\{j: a_j \in s\}} l(x|a_j,f,b) \tag{3.22}$$

$$= \frac{1}{|\mathcal{S}|}\sum_s K^{(s)}(b) \sum_{j=1}^{n_s} l(x|a_j,f,b) I_s(a_j) \tag{3.23}$$

$$= \frac{1}{|\mathcal{S}|}\sum_{j=1}^{n_s} l(x|a_j,f,b) \sum_s K^{(s)}(b) I_s(a_j) \tag{3.24}$$

$$= \sum_{j=1}^{n_s} l(x|a_j,f,b) K_j(b) \tag{3.25}$$

where I_s is an indicator function,

$$I_s(a_j) \stackrel{\text{def}}{=} \begin{cases} 1 & \textit{if} \;\; a_j \in s, \\ 0 & \text{otherwise}, \end{cases} \tag{3.26}$$

and

$$K_j(b) = \frac{1}{|\mathcal{S}|}\sum_s K^{(s)}(b) I_s(a_j) \geq 0 \tag{3.27}$$

describes how well the image can be explained assuming that it contains no objects and that the segment a_j is rendered.

3.3.4 Likelihood ratio model

As described in Section 3.2, the image generation model requires specifying for each function $f \in \mathcal{F}$ and $b \in \mathcal{B}$ a family of probability distributions, one for each possible combination of segment and object categories:

$$\{ \; p(\cdot \mid o_j, a_j, f, b) : j = 1, \cdots, n_s; \; o_j = 1, 2 \; \} \tag{3.28}$$

Given a family of background distributions

$$\{ \; p(\cdot \mid O_j = 2, a_j, b) : j = 1, \cdots, n_s; \;\; \} \tag{3.29}$$

we then derive the corresponding background distribution via likelihood ratio models. To do so let $\mathcal{H}$ be a family of functions $h : \mathbb{R}^r \to \mathbb{R}$, which we call *feature detectors*. These functions take as input a patch of pixels of size r, and output a real number. Each function f in $\mathcal{F}$ has the form

$$f(x) = \sum_{i=1}^{n} \alpha_i h_i(\phi(x)), \text{ for } x \in \mathbb{D} \tag{3.30}$$

where $h_1, \cdots, h_n \in \mathcal{H}$, $\mathbb{D} = \{\mathbb{R}^r \bigcup \mathbb{R}^{r+1} \cdots \bigcup \mathbb{R}^{n_p}\}$, i.e., the set of all real valued vectors of any length from $r = \min_j |a_j|$ to $n_p = \max_j |a_j|$, and $\phi : \mathbb{D} \to \mathbb{R}^r$ is a function that scales image segments of any size to patches of size r. Thus, each function $f \in \mathcal{F}$ is defined by an integer n, a set of feature detectors $\{h_1, \cdots, h_n\}$ and a vector of scalars $\alpha_1, \cdots, \alpha_n$. Then for any $a_j \in \mathcal{A}$ and $x \in \mathbb{R}^{|a_i|}$ we let

$$l(x|a_j, f, b) = \frac{p(x_j|O_j = 1, f, b)}{p(x_j|O_j = 2, b)} \propto e^{f(x_j)} \tag{3.31}$$

and therefore

$$l(x|a_j, f, b) = \frac{e^{f(x_j)}}{Z(a_j, f, b)} \tag{3.32}$$

$$Z(a_j, f, b) = \sum_{x_j} p(x_j|O_j = 2, b)\, e^{f(x_j)}, \quad x_j \in \mathbb{R}^{|a_j|} \tag{3.33}$$

where $Z(a_j, f, b) \in \mathbb{R}$ is a partition function which ensures that $p(x_j|O_j = 1, f, b)$ sums to one.

3.3.5 Modeling translation and scale invariance

Here we describe how we define the image generative model so as to achieve translation and scale invariance. First we define the family of background distributions via a seed distribution b for a reference size r, and a collection of distributions for generating patches of size s given patches of size r. Patches y of size $s > r$ are generated by sampling patches x of size r with probability $b(x)$ and then sampling patches y of size s with probability $p(y \mid x)$, where

$$p(y \mid x) \overset{\text{def}}{=} \delta(\phi(y), x)\, g(y, x), \text{ for } x \in \mathbb{R}^r,\ y\mathbb{R}^s \tag{3.34}$$

where the function g is any function such that

$$\sum_y p(y \mid x) = \sum_y \delta(\phi(y), x)\, g(y, x) = 1 \tag{3.35}$$

In other words, given a “seed” patch x of size r we put the constraint that a patch y of size $s > r$ can only be derived from x if the scaled down version of y looks like x. In Appendix 3.A, we show that under this model, the partition function $Z(a_j, f, b)$ is shift and scale invariant, i.e., constant with respect to a_j, and independent of the function g. Hereafter we will assume translation and scale invariant models as described in this section and thus drop the argument a_j from Z, i.e.,

$$l(x|a_j, f, b) = \frac{e^{f(x_j)}}{Z(f, b)}, \quad \text{for } a_j \in \mathcal{A} \tag{3.36}$$

3.4 Learning model parameters from examples

We treat learning as a Bayesian inference problem. We want to optimally update our beliefs about the distribution of the model parameters f and b by using all the available information in a dataset of training images. Ideally this would involve computing the posterior distributions of f and b given the dataset. Since this is not tractable computationally, we instead require choosing a single value for f, and a single value for b. A minimum error strategy calls for choosing the peak of the posterior probability distribution of f and b given the dataset. We use a uniform prior distribution over f and b, in which case the problem reduces to finding values of f and b that locally maximize the function $p(x \mid f, b)$. This is the focus of this section.

To begin with, suppose we are given a sample of n image segments $\{\tilde{x}_1, \cdots \tilde{x}_n\}$ of size r drawn from the background distribution. The log probability of the sample is

$$\log p(\tilde{x}_1 \cdots, \tilde{x}_n \mid f, b) = \sum_{i=1}^{n} \log p(\tilde{x}_i \mid O_i = 2, b) \tag{3.37}$$

With no additional constraints on b, the maximum likelihood estimate of b is simply the empirical distribution of the background sample. Thus

$$p(x \mid a_j, O_j = 2, \hat{b}) = \frac{1}{m} \sum_{i=1}^{m} \delta(\phi(x_j), \phi(\tilde{x}_i)) \tag{3.38}$$

Hereafter, since $\{\tilde{x}_1, \cdots \tilde{x}_n\}$ is fixed during learning, we hold $\hat{b}$ fixed and use it as a reference measure for the corresponding foreground distribution f.

Suppose we are now given a single image x which we know contains a foreground object, but we do not know where the object is located. Our goal is to maximize the probability of the observed images with respect to f, holding $\hat{b}$ fixed. The log probability of the training data is

$$\log p(x, \tilde{x}_1 \cdots, \tilde{x}_n \mid f, \hat{b}) = \log p(x \mid f, \hat{b}) + \sum_{i=1}^{n} \log p(\tilde{x}_i \mid O_i = 2, \hat{b}) \tag{3.39}$$

Since the background distribution is fixed and treated as a reference measure, we just need to focus on the likelihood of the image with a foreground object

$$\log p(x|f, \hat{b}) = \log \sum_{j=1}^{n_s} l(x_j \mid f) K_j(\hat{b}) \tag{3.40}$$

$$= \log \sum_{j=1}^{n_s} e^{f(x_j)} K_j(\hat{b}) - \log Z(f, \hat{b}) \tag{3.41}$$

where

$$Z(f,\hat{b}) = \frac{1}{n}\sum_{k=1}^{n} e^{f(\tilde{x}_k)} \tag{3.42}$$

Thus the log likelihood of the foreground image takes the following form

$$L(x|f,\hat{b}) = \log\sum_{j=1}^{n_s} e^{f(x_j)}K_j(\hat{b}) - \log\sum_{k=1}^{n} e^{f(\tilde{x}_k)} + \log(n) \tag{3.43}$$

It is convenient to express this function in terms of "weights" assigned to image segment values. This can be done by rearranging terms, then collecting common terms into summary variables. Let $z = \{z_1, ..., z_\xi\}$ be the set of ξ unique values of all observed image segments (i.e., the union of all example image segments). Now let u and v, and w be $1 \times \xi$ vectors, with elements defined as:

$$u_j(x,f) \stackrel{\text{def}}{=} \sum_{k=1}^{n_s} e^{f(z_k)}\delta(x_k, z_j) \tag{3.44}$$

$$v_j(f) \stackrel{\text{def}}{=} \sum_{k=1}^{n} e^{f(z_k)}\delta(\tilde{x}_k, z_j) \tag{3.45}$$

$$w_j(x,\hat{b}) \stackrel{\text{def}}{=} \sum_{k=1}^{n_s} K_k(\hat{b})\delta(x_k, z_j) \tag{3.46}$$

Thus u and v are weighted histograms – each entry u_j is the count of how often segments in image x take on the value z_j, multiplied by $e^{f(z_j)}$. Similarly, each entry v_j is the count of how often an image segment in the background sample takes on the value z_j, multiplied by $e^{f(z_j)}$. Now we can write the log-likelihood as

$$L(x|f,\hat{b}) = \log\sum_{j=1}^{\xi} u_j(x,f)\, w_j(x,\hat{b}) - \log\sum_{k=1}^{\xi} v_k(f) + \log n \tag{3.47}$$

We will try to find a local maximum of L by taking the gradient with respect to the function f and choosing a function that has a positive inner product with the gradient, a method which we will refer to as *functional gradient ascent.*

3.4.1 Functional gradient ascent

We use an iterative procedure for inferring f from data. At time $t-1$ we have a rendering function f_{t-1} with $t-1$ feature detectors

$$f_{t-1} = \sum_{\tau=1}^{t-1} \alpha_\tau h_\tau \tag{3.48}$$

Our goal is to find a feature detector h_t and constant α_t that improves the log-likelihood of the data, i.e. we want

$$f_t = f_{t-1} + \alpha h_t \tag{3.49}$$

such that

$$L(x \mid f_t, \hat{b}) > L(x \mid f_{t-1}, \hat{b}) \tag{3.50}$$

To do so we apply a functional gradient ascent approach [Friedman, 1999; Mason et al., 2000]. Functional gradient ascent generalizes the idea of gradients to functions of functions. We define the z_k component of the gradient of $L(x \mid f, \hat{b})$ with respect to f as follows

$$\begin{aligned} G_f(z_k) &\overset{\text{def}}{=} \left.\frac{\partial L(f + \epsilon\delta(z_k, \cdot), \hat{b})}{\partial \epsilon}\right|_{\epsilon=0} && (3.51)\\ &= \frac{\partial}{\partial \epsilon}\left.\left(\log \sum_{i=1}^{\xi} u_i(x, f)\, w_i(x, \hat{b}) e^{\epsilon\delta(z_i, z_k)} - \log \sum_{j=1}^{\xi} v_j(f) e^{\epsilon\delta(z_j, z_k)}\right)\right|_{\epsilon=0} && (3.52)\\ &= \frac{u_k(x, f)\, w_k(x, \hat{b})}{\sum_{i=1}^{\xi} u_i(x, f)\, w_i(x, \hat{b})} - \frac{v_k(f)}{\sum_{j=1}^{\xi} v_j(f)}, \quad \text{for } z_k \in z && (3.53)\\ G_f(y) &= 0, \text{for } y \in \mathbb{D}, y \notin z && (3.54) \end{aligned}$$

This can be viewed as taking the difference between the weighted histograms of foreground and background segments.

If instead of a single foreground image we have a collection of m independently drawn images $\mathbf{x} = (x^{(1)}, ..., x^{(m)})$ known to contain an object, then joint log probability of the collection is

$$\log p(\mathbf{x}|f, \hat{b}) = \sum_{i=1}^{m} \log p(x^{(s)}|f, \hat{b}) \tag{3.55}$$

and the functional gradient is

$$G_f(z_k) = \sum_{l=1}^{m} \left[\frac{u_k(x^{(l)}, f) w_k(x^{(l)}, \hat{b})}{\sum_{i=1}^{\xi} u_i(x^{(l)}, f) w_i(x^{(l)}, \hat{b})} - \frac{v_k(f)}{\sum_{j=1}^{\xi} v_j(f)}\right] \tag{3.56}$$

This can be written more concisely if we define $\mu = (\mu_1, ..., \mu_\xi)$, $\nu = (\nu_1, ..., \nu_\xi)$ as:

$$\mu_k(f) \overset{\text{def}}{=} \sum_{j=1}^{m} \frac{u(x_k^{(j)}, f) w_k(x^{(j)}, \hat{b})}{\sum_{i=1}^{\xi} u_i(x^{(j)}, f) w_i(x^{(j)}, \hat{b})} \tag{3.57}$$

$$\nu_k(f) \overset{\text{def}}{=} m \frac{v_k(f)}{\sum_{j=1}^{\xi} v_j(f)} \tag{3.58}$$

Then we can write

$$G_f(z_k) = \mu_k(f) - \nu_k(f), \text{ for } z_k \in z \tag{3.59}$$

$$G_f(y) = 0, \text{ for } y \in \mathbb{D}, y \notin z \tag{3.60}$$

The functional gradient ascent approach requires finding a function f with a positive inner product with the gradient. This is justified by the fact that to first order,

$$L(\mathbf{x}|f + \epsilon h, \hat{b}) = L(\mathbf{x}|f, \hat{b}) + \epsilon < G_f, h > \tag{3.61}$$

Note however that we cannot compute the gradient because we do not know the w terms. Fortunately, these terms are positive and constant with respect to f. In addition if the sets of foreground segments and background segments are disjoint, we can find a function which has the same sign as the gradient for all its terms. We call this function the pseudogradient G'_f and define it as follows

$$G'_f(z_k) = \mu'_k(f) - \nu_k(f), \text{ for } y \in z \tag{3.62}$$

$$G'_f(y) = 0, \text{ for } y \in \mathbb{D}, y \notin z \tag{3.63}$$

where $\mu'(f) = (\mu'_1(f), ..., \mu'_\xi(f))$ is defined as

$$\mu'(f) \overset{\text{def}}{=} \sum_{l}^{m} \frac{u_k(x^{(l)}, f)}{\sum_{i=1}^{\xi} u_i(x^{(l)}, f)} \tag{3.64}$$

Note for a segment z_k from the foreground images

$$G_f(z_k) = \mu_k \tag{3.65}$$

$$G'_f(z_k) = \mu'_k \tag{3.66}$$

which have the same sign. For a segment z_k from the background images

$$G_f(z_k) = -\nu_k \tag{3.67}$$

$$G'_f(z_k) = -\nu_k \tag{3.68}$$

which are equal and thus have the same sign. And for any other segment, both the gradient and pseudogradient take value zero. Thus the following theorem follows:

Theorem: PseudoGradient *If the set of segments from background images and from foreground images are disjoint, then each component of the pseudogradient has the same sign as the corresponding component of the gradient.*

The theorem guarantees that there is a constant ϵ such that if the inner product between h and G' is positive and greater than ϵ then the inner product between h and G is also positive. We can use this to find functions $h \in \mathcal{H}$ that can be used to incrementally improve $L(\mathbf{x}|f, \hat{b})$ in a gradient ascent procedure. We describe such a procedure for a specific family of functions $\mathcal{H}$ in Section 3.4.3.

3.4.2 Absolute continuity

The absolute continuity assumption in Section 3.3.2 means that any segment which appears in the foreground images must have a nonzero probability of appearing in the background. However quite often the *empirical* probabilities from the training sample violate the absolute continuity assumption. This can lead to degenerate solutions during learning, because $L(x|f, \hat{b})$ can be trivially increased by simply by finding feature detectors for which $\alpha_t h_t(z_k)$ is very large for any z_k that appears only in foreground examples.

We can avoid degenerate solutions by adopting a background model in which the foreground segments from the training set have a small but non-zero probability of occurrence, i.e.,

$$p^{\beta}(x \,|\, a_j, O_j = 2, \hat{b}) = (1-\beta)\frac{1}{n}\sum_{i=1}^{n} \delta(\phi(x), \phi(\hat{x}_i)) + (1-\beta)\sum_{i=1}^{m}\sum_{j=1}^{n_s} \delta(\phi(x), \phi(x_j^{(i)})) \tag{3.69}$$

where $0 \leq \beta \leq 1$ is the *foreground contamination* parameter. Then

$$\hat{Z}(f, \hat{b}, \beta) = (1-\beta)Z(f, \hat{b}) + \beta\frac{1}{n_s m}\sum_{i=1}^{m}\sum_{j=1}^{n_s} e^{f(x_j^{(i)})} \tag{3.70}$$

$$\hat{v}_j(f, \beta) \overset{\text{def}}{=} (1-\beta)v_j(f) + \frac{\beta}{m}\sum_{i=1}^{m} u_j(x^{(i)}, f) \tag{3.71}$$

$$\hat{\nu}_j(f, \beta) \overset{\text{def}}{=} m\frac{\hat{v}_j(f, \beta)}{\sum_{k=1}^{\xi} \hat{v}_k(f, \beta)} \tag{3.72}$$

$$\hat{G}_f(z_k, \beta) = \mu_k(f) - \hat{\nu}_k(f, \beta) \tag{3.73}$$

$$\hat{G}'_f(z_k, \beta) = \mu'_k(f) - \hat{\nu}_k(f, \beta) \tag{3.74}$$

Note that while $\beta > 0$ guarantees absolute continuity between the foreground and background models, it also breaks the orthogonality condition needed for the PseudoGradient Theorem. However given a fixed training set, it is always possible to find a value of $\beta > 0$ which guarantees absolute continuity, and such that all the components of the Gradient and PseudoGradients have the same sign. While we may not know what that value is, in practice, simply fixing $\beta = 0.0001$ was effective in all of our experiments.

3.4.3 Feature Selection

We now describe some details of the gradient ascent procedure for a specific family of functions $\mathcal{H}$. Let ψ^i be a partition of $\mathbb{R}^r$ into $n_b \in \mathbb{Z}_+$ subsets, i.e.,

$$\psi^i = \{\psi^i_1, ..., \psi^i_{n_b}\} \tag{3.75}$$

We will refer to each partition as a *feature*. Let

$$\Psi = \{\psi^1, ..., \psi^{n_f}\} \tag{3.76}$$

be a collection of n_f features. Let $h^i \in \mathcal{H}$, $h : \mathbb{D} \to \mathbb{R}$ have the form

$$h^i(z) \overset{\text{def}}{=} \sum_{j=1}^{n_b} \eta^i_j \; I_{\psi^i_j}(\phi(z)), \quad z \in \mathbb{D} \tag{3.77}$$

where $\eta^i = (\eta^i_1, ..., \eta^i_{n_b}) \in \mathbb{R}^{n_b}$, ϕ is the scaling function, and

$$I_{\psi^i_j}(x) = \begin{cases} 1 & \text{if } x \in \psi^i_j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.78}$$

We refer to η^i as a *tuning curve*. Thus h^i takes an image patch, scales it to size r, and outputs a number based on the feature ψ^i and tuning curve η^i.

For any particular h^i, the inner product with the pseudogradient is

$$\begin{aligned} < G'_f, h^i > &= \sum_{k=1}^{\xi} G'_f(z_k) h^i(z_k) && (3.79) \\ &= \sum_{k=1}^{\xi} G'_f(z_k) \sum_{j=1}^{n_b} \eta^i_j \, I_{\psi^i_j}(z_k) && (3.80) \\ &= \sum_{j=1}^{n_b} \eta^i_j \sum_{k=1}^{\xi} G'_f(z_k) \, I_{\psi^i_j}(z_k) && (3.81) \\ &= \sum_{j=1}^{n_b} \eta^i_j \lambda^i_j(f) && (3.82) \end{aligned}$$

where

$$\lambda_j^i(f) = \sum_{k=1}^{\xi} G'_f(z_k)\, I_{\psi_j^i}(z_k) \tag{3.83}$$

We want to minimize the angle between h^i and G'_f, therefore we find the values of η^i where $< G'_f, h^i >$ takes a maximum subject to a normalization constraint on η^i, i.e., $\sum_{j=1}^{n_b} {\eta_j^i}^2 = 1$. Let γ be a Lagrange multiplier, then taking the derivative with respect to each η_j^i and setting it to zero,

$$0 = \frac{\partial < G'_f, h >}{\partial \eta_j^i} = \frac{\partial}{\partial \eta_j^i}\left[\sum_{k=1}^{n_b} \eta_k^i\, \lambda_k(f) + \gamma\left(1 - \sum_{l=1}^{n_b} {\eta_l^i}^2\right)\right] \tag{3.84}$$

$$= \lambda_j(f) - 2\gamma\eta_j^i \tag{3.85}$$

The maximum of $< G'_f, h^i >$ occurs at

$$\eta_j^i = \frac{\lambda_j(f)}{\sqrt{\sum_{k=1}^{n_b} \lambda_k^2(f)}} \tag{3.86}$$

3.4.4 Lower bound on the likelihood

Gradient ascent methods typically require either taking a small, fixed step size α, or optimizing α to give a maximum improvement in the objective function. Taking the latter approach, the procedure would be: for each feature ψ^i, pick η^i to maximize $< G'_f, h^i >$ subject to a norm on η^i, then pick α to maximize $L(\mathbf{x}|f_{t-1} + \alpha h^{(i)}, \hat{b})$.

Because we do not have access to the w terms, we cannot directly optimize $L(\mathbf{x}|f_{t-1} + \alpha h^{(i)}, \hat{b})$. Instead, we will optimize a lower bound on the log-likelihood. Let

$$L(\mathbf{x}|f, \hat{b}) = \sum_{i=1}^{m} \log \sum_{j=1}^{\xi} u_j(x^{(i)}, f)\, w_j(x^{(i)}, \hat{b}) - \log \sum_{k=1}^{\xi} v_k(f) + \log n \tag{3.87}$$

$$\geq \sum_{i=1}^{m} \log \sum_{j=1}^{\xi} u_j(x^{(i)}, f)\hat{w}^{(i)} - \log \sum_{k=1}^{\xi} v_k(f) + \log n \tag{3.88}$$

where

$$\hat{w}^{(i)} = \min_{j \in \{1,..,\xi\}} w_j(x^{(i)}, \hat{b}) \tag{3.89}$$

is a constant with respect to f. Then

$$L(\mathbf{x}|f,\hat{b}) \geq \sum_{i=1}^{m} \log \sum_{j=1}^{\xi} u_j(x^{(i)}, f) - \log \sum_{k=1}^{\xi} v_k(f) + \log n + \sum_{l=1}^{m} \log \hat{w}^{(l)} \tag{3.90}$$

$$= \sum_{i=1}^{m} \log \sum_{j=1}^{\xi} u_j(x^{(i)}, f) - \log \sum_{k=1}^{\xi} v_k(f) + C = L'(\mathbf{x}|f,\hat{b}) \tag{3.91}$$

Where C is constant and $L'(\mathbf{x}|f,\hat{b})$ is the pseudo log-likelihood

$$L'(\mathbf{x}|f,\hat{b}) = \sum_{i=1}^{m} \log \sum_{j=1}^{\xi} u_j(x^{(i)}, f) - \log \sum_{k=1}^{\xi} v_k(f) \tag{3.92}$$

We can now specify a gradient ascent procedure which maximizes a lower bound on the log-likelihood of the data:

- For a given function f_{t-1} compute the pseudogradient $G'_{f_{t-1}}$
- For each $\psi^i \in \Psi$,
  - Set tuning curve $\hat{\eta}^i$ using (3.86), i.e.,

$$\lambda^i_j(f_{t-1}) = \sum_{k=1}^{\xi} G'_{f_{t-1}}(z_k)\, I_{\psi^i_j}(z_k) \tag{3.93}$$

$$\hat{\eta}^i_j = \frac{\lambda_j(f_{t-1})}{\sqrt{\sum_{k=1}^{n_b} \lambda^2_k(f_{t-1})}} \tag{3.94}$$

  - Set $\hat{h}^i$ to use feature ψ^i and tuning curve $\hat{\eta}^i$
  - Find step size α_i to maximize a lower bound on the log likelihood, i.e.,

$$\alpha^i = \operatorname*{argmax}_{\alpha \in \mathbb{R}} L'(\mathbf{x}|f_{t-1} + \alpha \hat{h}^i, \hat{b}) \tag{3.95}$$

$$g^i = \max_{\alpha \in \mathbb{R}} L'(\mathbf{x}|f_{t-1} + \alpha \hat{h}^i, \hat{b}) \tag{3.96}$$

- Choose $j = \operatorname*{argmax}_i \; g_i$
- Choose feature $h_t = \hat{h}^j$ and step size $\alpha_t = \alpha_j$, then update f:

$$f_t = f_{t-1} + \alpha_t h_t \tag{3.97}$$

To decide when to stop updating f, we test performance on inference on a validation set of images after each update of f.

3.5 Inference on images given model parameters

In the previous section we studied the problem of making inferences about the model parameters f and b given a training set of images. In this section we assume model parameters have been learned and our goal is to make inferences about the unobserved causes of an image, e.g., whether or not an object of interest is present and, if so, where the object is located.

3.5.1 Inferring pixel category assignments

Given a new image x and models f and b, our goal is to infer the probability that a particular pixel i renders a foreground object. First we analyze the case in which the segmentation is known and we know there is only one foreground segment. In this case

$$p(y|x,s,f,b) = \sum_{o} p(o|x,s,f,b)p(y|o,x,s,f,b) = \sum_{o} p(o|x,s,f,b)p(y|o,s,f,b) \quad (3.98)$$

because Y is conditionally independent of X given O. To find $p(o|x,s,f)$, we find $p(x|s,f,b)$ using equation (3.18), then using Bayes' rule we can find:

$$\begin{aligned} P(O_i = 1|x,s,f,b) &= \frac{P(O_i = 1|s,f,b)p(x|O_i = 1,s,f,b)}{\sum_o p(x|o,s,f,b)} & (3.99)\\ &= \frac{K^{(s)}(b)l(x|a_i,f,b)}{K^{(s)}(b)\sum_{\{a_j \in s\}} l(x|a_j,f,b)} & (3.100)\\ &= \frac{l(x|a_i,f,b)}{\sum_{\{a_j \in s\}} l(x|a_j,f,b)} & (3.101)\end{aligned}$$

Finally, because the pixel assignments are fully determined by the segments' object category assignments, we can simplify (3.98) as

$$P(Y_i = 1|x,s,f,b) = \sum_{\{k:k \in a_k\}} P(O_k = 1|x,s,f,b) \quad (3.102)$$

$$P(Y_i = 2|x,s,f,b) = 1 - P(Y_i = 1|x,s,f,b) \quad (3.103)$$

A pixel-wise posterior probability "image" can now be rendered by setting corresponding pixels in a raster to these pixel category assignment probabilities. This image can be interpreted intuitively as an object-specific "saliency map". If the segmentation is unknown, then

$$p(Y_i = 1|x, f, b) = \sum_s p(s)p(Y_i = 1|x, s, f, b) = \sum_s p(s) \sum_{k:k\in a_k} \frac{l(x|a_j, f, b)}{\sum_{\{a_k\in s\}} l(x|a_j, f, b)} \tag{3.104}$$

$$= \sum_{\{k:i\in a_k\}} \sum_s \frac{p(s)l(x|a_j, f, b)}{\sum_{\{a_k\in s\}} l(x|a_j, f, b)} \tag{3.105}$$

$$= \frac{1}{|\mathcal{S}|} \sum_{\{k:i\in a_k\}} l(x|a_k, f, b) \sum_s \frac{I_s(a_k)}{\sum_{\{a_j\in s\}} l(x|a_j, f, b)} \tag{3.106}$$

$$= \sum_{\{k:i\in a_k\}} l(x|a_k, f, b)Q_k \tag{3.107}$$

where $I_s(\cdot)$ is the indicator function defined in (3.26) and

$$Q_k = \frac{1}{|\mathcal{S}|} \sum_{\{s:a_k\in s\}} \frac{1}{\sum_{\{a_j\in s\}} l(x|a_j, f, b)} \tag{3.108}$$

Unfortunately there is no simple way to compute this term, therefore in practice we estimate Q_k as constant with respect to k, and refer to the resulting value as an approximate posterior probability. In practice the resulting approximate posterior probability maps do look like "saliency maps" with respect to the object of interest. We are currently developing methods for getting better estimates of Q_k.

3.5.2 Inferring presence versus absence of objects

If we know the segmentation and models f and b, the likelihood-ratio of an image containing one versus no foreground objects is simply

$$\frac{p(x|s, f, b, \text{one foreground object})}{p(x|s, f, b, \text{no foreground objects})} = \sum_{\{a_j\in s\}} l(x|a_j, f, b) \tag{3.109}$$

We can then choose a threshold $\tau \in \mathbb{R}$ to minimize classification error over the set of training images if we predict "object present" only if the sum of likelihood ratios exceeds it.

When the segmentation is unknown, the likelihood ratio of an image given one object present versus no objects present cannot be computed exactly due to the Q_k terms which depend on being able to . However, it is straightforward to apply a simple discriminative technique to classify images as containing or not containing the object of interest using the learned model f. First, compute f for every segment in the image.

Then sort the outputs in descending order into a vector $v = (v_1, ..., v_{n_s})$. Finally, use the labels in the training images to choose a number n_l and a threshold τ which minimizes the total classification error across all training images, when prediction $H_{\mathit{final}}(x_i)$ is given by

$$H_{\mathit{final}}(x_i) = \begin{cases} \mathit{present} & \text{if } \sum_{i=1}^{n_l} V_i > \tau \\ \mathit{absent} & \text{otherwise} \end{cases} \tag{3.110}$$

3.5.3 Inferring object location

We define the best inference for the object location as the one which minimizes the probability of misclassifying the image pixels. For a given hypothesis foreground region, the pixel misclassification probability is proportional to the sum of the probability that each pixel *inside* the region belongs to background, plus the sum of the probability that each pixel *outside* the region belongs to foreground. Therefore, we want to choose classification region R from the allowed segmentation regions in $\mathcal{A}$ as follows:

$$R = \operatorname*{argmin}_{a_j} \sum_{j \in a_i} P(Y_j = 2|x, f, b) + \sum_{j \notin a_i} P(Y_j = 1|x, f, b) \tag{3.111}$$

$$= \operatorname*{argmin}_{a_j} \sum_{j \in a_i} (1 - P(Y_j = 1|x, f, b)) + \sum_{i=1}^{k} P(Y_j = 1|x, f, b) - \sum_{j \in a_i} P(Y_j = 1|x, f, b) \tag{3.112}$$

$$= \operatorname*{argmin}_{a_j} |a_j| - 2\sum_{j \in a_i} P(Y_j = 1|x, f, b) + \sum_{i=1}^{k} P(Y_j = 1|x, f, b) \tag{3.113}$$

$$= \operatorname*{argmin}_{a_j} |a_j| - 2\sum_{j \in a_i} P(Y_j = 1|x, f, b) \tag{3.114}$$

where k is the total number of pixels in the raster, and $|a_j|$ is the number of pixels in the region. If the regions are sufficiently restricted, e.g., to rectangles, this can be computed efficiently using summed area tables. Finally note if a segment of size zero gives the minimum pixel classification error, then it is possible to interpret this as a "no object present" hypothesis.

3.6 Neural Network interpretation

At runtime, the object classification, localization, and posterior probability map can be implemented as a three hidden layer convolutional neural network with lateral inhibition. To do so, the set of feature detectors and their linear combination (via the α terms) are first replicated, once for each overlapping window in the input image. The replicated feature detectors can be viewed as a first hidden layer, and the combination of feature detectors for each segment is then a second hidden layer. The term "convolutional" refers to the fact that the feature detectors and their weighted sum are applied to all segments of the image at multiple scales in parallel.

The transfer function for the second hidden layer units is the exponential function, to convert them from log-likelihood ratios to likelihood ratios. These outputs are then normalized so that they sum to one (which can be implemented via lateral inhibitory connections), and these outputs are fed into a third hidden layer, the posterior probability map, which has the same number of units as the number of input pixels. Each second hidden layer node output is added to each third-hidden layer node corresponding to a pixel in the input region for that segment. The sum of the posterior probability map values over all candidate hypothesis windows is performed by an output layer, and the maximum of this output layer can be taken as the target location decision. Finally, an additional output unit which takes the second hidden layer as input can be used to compute the final *presence* vs. *absence* decision.

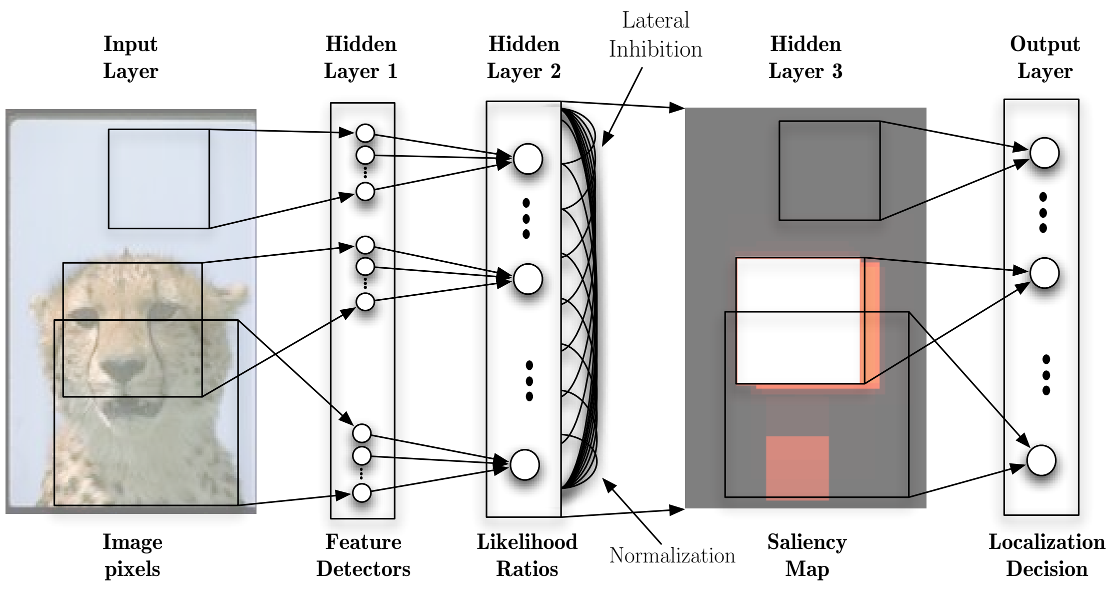


Figure 3.4: Implementation as a convolutional neural network. For images of size 640 $\times$ 480 pixels and learning ten feature detectors, this system can be implemented using 5 million neurons.

An illustration of the system implementation as a convolutional neural network is shown in figure 3.4. The number of units required depends on the resolution of the input image and the number of features learned. For a 640×480 image with a base window size of 24×24 and increasing window scale factor of 1.2 per scale (rounded up to the nearest integer), this gives 400,000 windows. For a model requiring 10 features, this results in 4 million nodes in the first hidden layer, 400,000 units in the second hidden layer, about 300,000 units in the third layer, and finally 400,000 units in the output layer, for a total of about 5 million units.

3.7 Implementation Details

3.7.1 Weighted feature histograms

For efficiency reasons, it is useful to introduce a set of intermediate variables during feature selection. Given m foreground images $\mathbf{x} = (x^{(1)}, ..., x^{(m)})$, a sample of n background patches $\tilde{x} = (\tilde{x}_1, ..., \tilde{x}_n)$, a pool of n_f features $\Psi = \{\psi^1, ..., \psi^{n_f}\}$, and a background corruption constant β, let $\tilde{u}, \tilde{v}$, and $\tilde{\lambda}$ be defined as

$$\tilde{u}_k^i(x^{(j)}, f) \overset{\text{def}}{=} \sum_{l=1}^{n_s} e^{f(z_l)} I_{\psi_k^i}(x_l^{(j)}) \tag{3.115}$$

$$\tilde{v}_k^i(f, \beta) \overset{\text{def}}{=} \frac{1-\beta}{n} \sum_{j=1}^{n} e^{f(z_k)} I_{\psi_k^i}(\tilde{x}_j) + \frac{\beta}{n_s m} \sum_{l=1}^{m} \tilde{u}_k^i(x^{(l)}, f) \tag{3.116}$$

$$\tilde{\lambda}_k^i(f, \beta) \overset{\text{def}}{=} \sum_{j=1}^{m} \frac{\tilde{u}_k^i(x^{(j)}, f)}{\sum_{l=1}^{n_b} \tilde{u}_l^i(x^{(j)}, f)} - m \frac{\tilde{v}_k^i(f, \beta)}{\sum_{s=1}^{n_b} \tilde{v}_s^i(f, \beta)} \tag{3.117}$$

Now (3.93) can be written as

$$\tilde{\eta}_j^i = \frac{\tilde{\lambda}_j^i(f_{t-1}, \beta)}{\sum_{k=1}^{n_b} (\tilde{\lambda}_k^i(f_{t-1}, \beta))^2} \tag{3.118}$$

and the pseudo-log likelihood (3.92) becomes

$$L'(\mathbf{x}|f_{t-1} + \alpha \hat{h}^i, \hat{b}, \beta) = \sum_{j=1}^{m} \log \sum_{k=1}^{n_b} \tilde{u}_k^i(x^{(j)}, f_{t-1}) e^{\alpha \tilde{\eta}_k^i} - m \log \sum_{l=1}^{n_b} \tilde{v}_l^i(f_{t-1}, \beta) e^{\alpha \tilde{\eta}_l^i} \tag{3.119}$$

When n_b is much smaller than ξ, this function is significantly faster than (3.92) to evaluate. This speedup is critical to make the optimization of α in equation (3.95) practical.

This also provides an intuitive interpretation of what each feature detector is doing. We can interpret $\tilde{u}^i(x^{(j)}, f) = (\tilde{u}^i_1(x^{(i)}, f), ..., \tilde{u}^i_{n_b}(x^{(i)}, f))$ as a weighted histogram of the feature ψ^i over all segments in the jth foreground image, and $\tilde{v}^i(f, \beta) = (\tilde{v}^i_1(f, \beta), ..., \tilde{v}^i_{n_b}(f, \beta))$ as a weighted histogram of the feature ψ^i over all segments in the background sample (with some corruption from the foreground). Average the foreground histograms across images, and the tuning curve is then the difference between foreground and background weighted histograms, times a constant.

3.7.2 Kernel smoothing

To prevent the tuning curves used by feature detectors in f from overfitting the example data, we use the Naradaya-Watson kernel regression technique to smooth the tuning curves. Let $\varphi_\sigma(j,k)$ be a kernel weighting function with bandwidth σ, where $\sum_{j=1}^K \varphi_\sigma(j,k) = \sum_{k=1}^K \varphi_\sigma(j,k) = 1$. Given a tuning curve η, if we now set

$$\tilde{\eta}_k = \sum_{j=1}^{n_b} \varphi_\sigma(j,k)\eta_j \tag{3.120}$$

then we obtain a smoothed tuning curve. For a Gaussian kernel, let

$$\varphi_\sigma(j,k) = \frac{e^{(j-k)^2/(2\sigma)}}{\sum_{i=1}^{n_b} e^{(i-k)^2/(2\sigma)}} \tag{3.121}$$

It is possible to interpret this probabilistically – if we assume Gaussian additive noise has corrupted the pixels, and the features are discretized linear filters, then with the right choice of σ this method yields an improved estimate of the likelihood for uncorrupted data. However even without this probabilistic interpretation, this kernel regression technique is a useful method for limiting the "effective degrees of freedom" of h [Hastie et al., 2001], which often helps generalization in practice. The trade-off is that there is now a free parameter σ.

3.7.3 Computational complexity

During feature selection, if there are n_t total training segments, then for each feature ψ^i, the cost of computing the histograms in (3.115) and (3.116) is $O(n_t)$. The cost of computing the tuning curve $\hat{\eta}^i$ and α^i is negligible if n_t is much larger than n_b. Therefore the time to add one feature to the model is linear in the number of training segments. Since the whole process is repeated for n_f features, the total cost for one round of inference is $O(n_f \cdot n_t)$.

In most of our experiments on real images, we set the number of training patches $n_t = 200,000$, number of histogram bins $n_b = 128$, and number of features $n_f = 1,000,000$. During learning, most of the computation involved in evaluating all features on all image segments can be computed once and stored in memory – for each segment, we must store several integral image representations (due to the choice of features described below), bringing the memory requirement to about 4GB. Also, because n_f was large, we employed an additional gradient ascent step in the parameter-space of features to avoid having to calculate all candidate features at each step.

With this approach, it takes about 11 minutes to select one feature on a 2GHz PowerMac G5 using a combined Matlab and C++ implementation. This actually places our algorithm among the fastest for this problem – much faster than Fergus et al. [2003] who require 36 hours to learn 6 features, about the same speed as our eye detector from Chapter 2, and also about the same as the generative system in Ulusoy and Bishop [2005], which had some strongly labeled training data. On the negative side, the memory requirements are enormous, requiring us to create a special 64-bit process that communicates with the primary implementation in Matlab (which is 32-bit) using shared memory. We are currently developing a technique to reduce these memory requirements by using sampling techniques for estimating the histograms used in the model updates.

3.7.4 Choosing candidate segments $\mathcal{A}$

For all images the set of candidate segments $\mathcal{A}$ is derived from the multi-scale sliding-window approach described in 2.4.1. It includes many locations and many sizes. First, a base size of $k = 24$ is chosen. At scale $s = 1$, each $k \times k$ pixel segment is considered a candidate segment. We increase the scale by multiplying the previous scale s by 1.2, and rounding up to the closest integer. Now each $k \cdot s \times k \cdot s$ segment is chosen, spaced apart by s pixels – i.e., for scale $s = 3$, the segments are of size 72×72 pixels, and each neighboring segment is shifted by 3 pixels. This process is repeated until the segment size does not fit within the image bounds. For a 640×480 pixel image, this produces over $400,000$ total patches.

3.7.5 Image Features

In order to make the sliding window approach efficient and capable of real-time performance, we use fast features that are known to be effective for learning object detectors in the supervised learning scenario. We use two types of features: contrast features, and dominant orientation features, illustrated in Figure 3.7.5. The contrast features, based on [Viola and Jones, 2001], are described in Chapter 2. The dominant orientation features are from [Levi and Weiss, 2004], with minor modifications, and described below. In [Levi and Weiss, 2004], these features were used in a supervised context to train a state-of-the-art face detector using only a few hundred training examples.

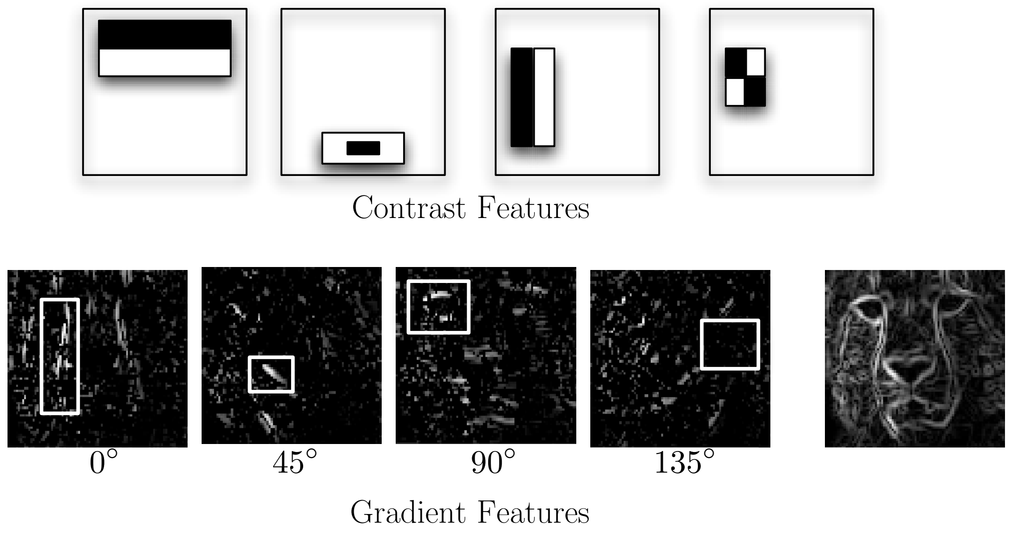


Figure 3.5: Features used for natural images. (Top:) Haar wavelet-like contrast features adapted from Viola and Jones [2001]. For each feature, the sum of the pixels in the black regions are subtracted from the sum of the pixels in the white regions. (Bottom:) The gradient orientation features adapted from Levi and Weiss [2004]. For each feature, an orientation range is chosen, and the sum is taken of the magnitude of the gradient for every pixel whose gradient orientation falls in this range. This sum is then divided by the total gradient in the same region (an example of the total gradient image is shown on right).

A dominant orientation feature estimates the proportion of the gradient energy within a sub-region of the detection window that occurs in a particular range of orientations. The features are computed quickly as follows. First, the gradients at pixel j in an image x are computed by convolving 3×3 Söbel masks with the image, resulting in two horizontal and vertical edge images:

$$G^h = Sobel_h * x \tag{3.122}$$

$$G^v = Sobel_v * x \tag{3.123}$$

where $Sobel_h$ and $Sobel_v$ are the horizontal and vertical Söbel masks respectively, and $*$ is the convolution operator. At each location j in the image, the orientation of the edge is

$$\phi_j = \arctan(\frac{G_j^h}{G_j^v}) \tag{3.124}$$

We then divide the unit circle into n_o orientation bins, and create n_o auxiliary images $\Theta = (\Theta^1, ..., \Theta^{n_o})$ where the value at each jth pixel is

$$\Theta_j^i = \begin{cases} \sqrt{{G_j^h}^2 + {G_j^v}^2} & \text{if } \phi_j \in bin_i \\ 0 & \text{otherwise} \end{cases} \tag{3.125}$$

Using the integral image, it is efficient to compute the sum of pixels within any sub-region of any of these auxiliary orientation-bin images with only four lookups and three arithmetic operations. Let r be a sub-regions within an image. Then given a sub-region r, orientation k, and orientation images Θ, let a dominant orientation feature f be

$$f(r, k, \Theta) = \frac{\sum_{j \in r} \Theta_j^k + \epsilon}{\sum_{l=1}^{n_o} \sum_{j \in r} \Theta_j^l + \epsilon} \tag{3.126}$$

where ϵ is a small regularization constant (we set this to 1 for images with pixel values in the range $[0, 255]$). In addition to these basic dominant orientation features, we also created a set of symmetry features which were simply the difference of a normal orientation feature and it's reflection with respect to the detection window about either the horizontal or vertical axis, or about origin.

For both contrast and dominant orientation features, the range of possible output values varies depending on the statistics of the image database. Because we use tuning curves instead of arbitrary real-valued functions, it is important to fix the minimum and maximum bins of the tuning curve to cover most of the possible outputs of the associated feature across the training dataset. Therefore in each experiment, we find a minimum and maximum value for each feature so that 99% of the training data lies between these two values. This can be considered step zero of learning, since it is driven by the data. A bad choice of tuning-curve input ranges can lead to poor performance.

3.8 Experiments

Given a set of images labeled as containing or not containing an object of interest, our goal is to learn a model that allows us to identify the presence or absence, as well as the location, of the object of interest in unseen images. The first two data sets are commonly in use in the computer vision community – the Caltech-4 and Caltech-101 data sets. We then perform an experiment on a large dataset of faces collected from the web, GENKI06, which is more difficult and more realistic than the faces in the Caltech data sets.

3.8.1 Experiment 1: Caltech-4

The most commonly used and reported on dataset for the object discovery task is the "Caltech-4" dataset, containing between 400-1200 images for each of 4 categories: faces, airplanes, motorbikes, and cars (viewed from behind), plus a fifth "background" category containing snapshots from around the Caltech campus. The faces and background dataset are from Weber et al. [2000], and the additional object categories were introduced by Fergus et al. [2003]. Each image in the dataset contains an example of the object of interest at a particular orientation – i.e., faces are always frontal upright, airplanes are always viewed from the side, pointing to the right. The objects are typically near the center of the image, and the variation in scale is about 1 to 1.5 octaves, although in the faces category the size is fixed.

For each object category, we trained an object detector using 35 positive images, and sampled segments from 200 background images. For each foreground image, we collected a random sample of up to 3000 segments, and for each background image we collected a random sample of a maximum of 200 segments. To reduce the influence of the overall size of the image, we restrict the minimum size of each segment in an image to no smaller than about 18% of the total area of the image. This means that the object of interest is assumed to be somewhat prominent in the image – i.e., images containing only tiny examples of the object would likely not be able to learn the object category in this experiment. We also immediately reject any segment whose standard deviation is below 0.02, which prevents the classifier from using large numbers of redundant near-uniform colored segments. The total number of image segments collected this way was $100,000$ image segments from each category.

For all the image experiments reported in this paper, we fixed the foreground contamination term $\beta = 0.0001$. The tuning curves had $n_b = 128$ bins, and we used a Gaussian kernel with bandwidth $\sigma = 0.2$ for smoothing. We used these same parameter settings for all the experiments in this paper.

For each category, we trained for 50 steps, and then used a holdout-validation set to find a minimum number of features required to achieve peak accuracy in terms of the area under the ROC curve. For the Caltech-4 dataset, it was typically possible to achieve best accuracy using only about $3-8$ features. We then computed final results by testing on all of the remaining images in the dataset.

Table 3.1: Caltech-4 equal point classification rates for our system, along with the best reported results for each year since 2003. Although other systems did not include confidence intervals, we show the standard deviation of the mean. Quattoni et al. [2004] did not report runtime speed.

	Fergus et al. [2003]	Quattoni et al. [2004]	Serre et al. [2005]	Current system
Faces	96.4	99.0	98.2	$\mathbf{99.8 \pm 0.2}$
Cars rear	90.3	94.6	**99.8**	$\mathbf{99.8 \pm 0.2}$
Airplanes	94.0	96.0	96.7	$\mathbf{98.4 \pm 0.5}$
Motorbikes	95.0	95.0	**98.0**	$\mathbf{97.8 \pm 0.6}$
Time per image:	10-15 seconds	–	15-20 seconds	**0.02-0.06 seconds**

Table 3.8.1 shows the correct classification rates for our classifier, compared to those of several other recent algorithms for the same data sets. Our system achieves state-of-the art detection accuracy on this dataset. In addition to a high detection accuracy rate, our system is also able to perform detection in real-time – i.e., a 500×300 image can be evaluated in less than 60 milliseconds. This is several orders of magnitude faster than previous systems, which typically require a minimum of several seconds to detect the presence or absence of an object in the scene (however it is not known to what degree these other systems were optimize for speed). Finally, this system provides a belief about the location of the object in the image, making it the first weakly supervised system that could be useful for object tracking in a real video application.

3.8.2 Experiment 2: Caltech 101

The Caltech-101 dataset, from Fei-Fei et al. [2004], is similar to the Caltech-4 data sets, however the number of object categories has been expanded to 100, plus a new "background" dataset which contains the result of typing the string "Things"

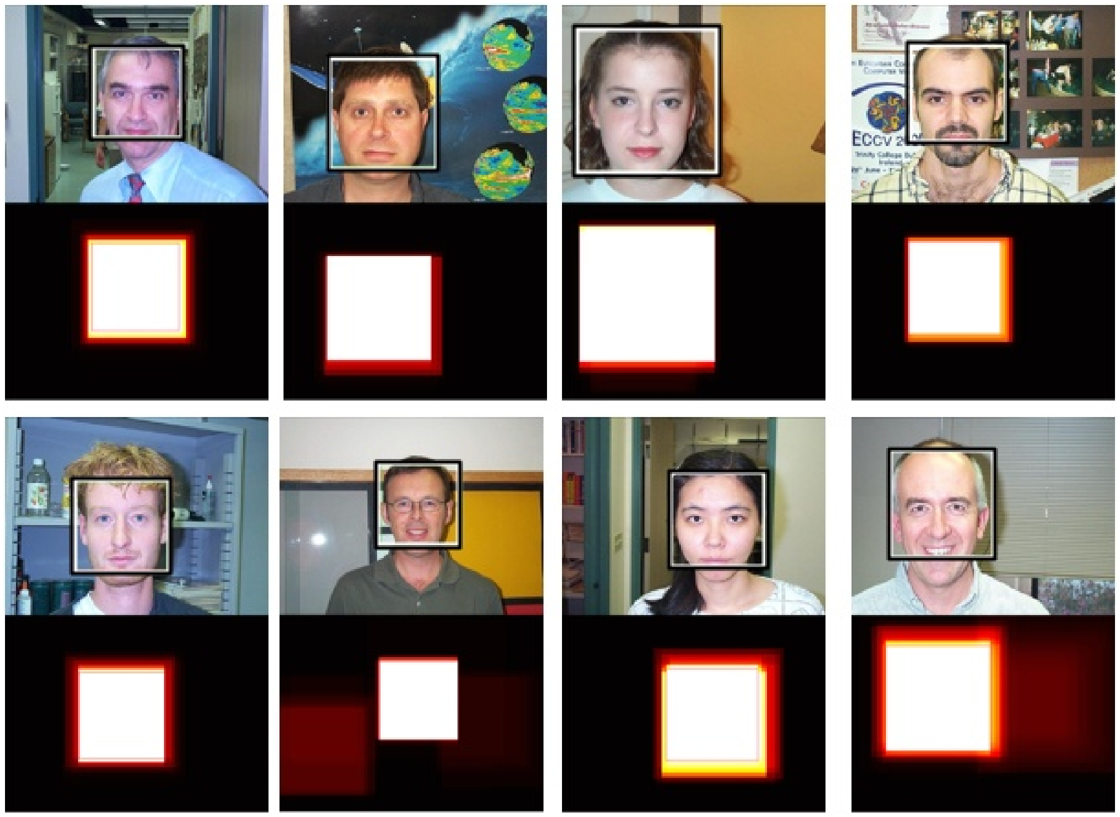


Figure 3.6: Examples of inference on novel Caltech-4 Faces. On top of each image pair is the original image, with the bounding box showing the minimum classification error region inferred by the model. On the bottom is the saliency map, showing the approximate posterior probability under the model that each pixel was generated by the foreground object.

into the Google images internet search engine. Each object category contains between 30 and 60 examples of the object of interest. Some categories are more difficult than others, containing greater variation in scale, orientation, and variety – for instance, the category "cannons" contains both photographs and hand-drawn images, and contains a great variety of styles of cannons e.g., cannons with and without wheels, and pointed at different elevations – though the images were prepared so that the cannons always point generally to the right. Figures 3.7 – 3.10 show some examples from these data sets.

We followed the same procedure for learning each category as described for the Caltech-4 dataset. We did not perform any optimizations for the parameters, simply using the exact same settings as in the previous experiment. One issue that made this task different from the Caltech-4 dataset is that some categories do not contain many images

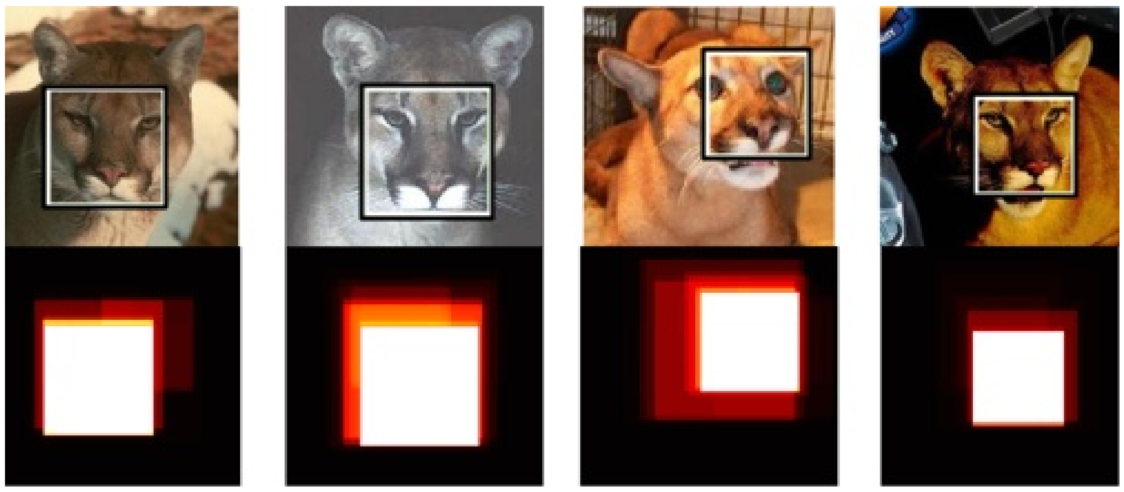


Figure 3.7: Examples of inference on novel images in the "cougar faces" category. The image above shows the best classification window inferred by the model. On the bottom is the saliency map, showing the approximate posterior probability under the model that each pixel was generated by the foreground object.

– sometimes no more than 34 images. In order to get reasonable confidence intervals for performance, we automatically limited the number of images used for training so that at least 20 images were available for validation. Therefore, some training data sets used fewer than 30 training images.

The average area under the ROC curve for these data sets was 92.7 and the average equal point error was 89.1. This compares favorably with other results on recent systems. Serre et al. [2005], which is among the best current systems (indeed has the best reported results on Caltech-4), did not report the average AUC, however they did give histograms of the AUC for their system under several different conditions. We superimpose a histogram of our own system over theirs in Figure 3.8.2. Tables 3.2 and 3.8.2 give a view of the specific values of the AUC for the top 10 and worst 10 categories. We can see that in raw accuracy performance, our system is an improvement over Serre et al. [2005]. Our system offers the added advantage of being very fast, and offering good localization.

3.8.3 Experiment 3: GENKI database

The GENKI database contains 75,000 images of people collected from the internet. Labels for the corners of the eyes, nose, mouth and head pose (pitch, yaw, and roll) are available, however we did not use any of these labels for training. Although the images

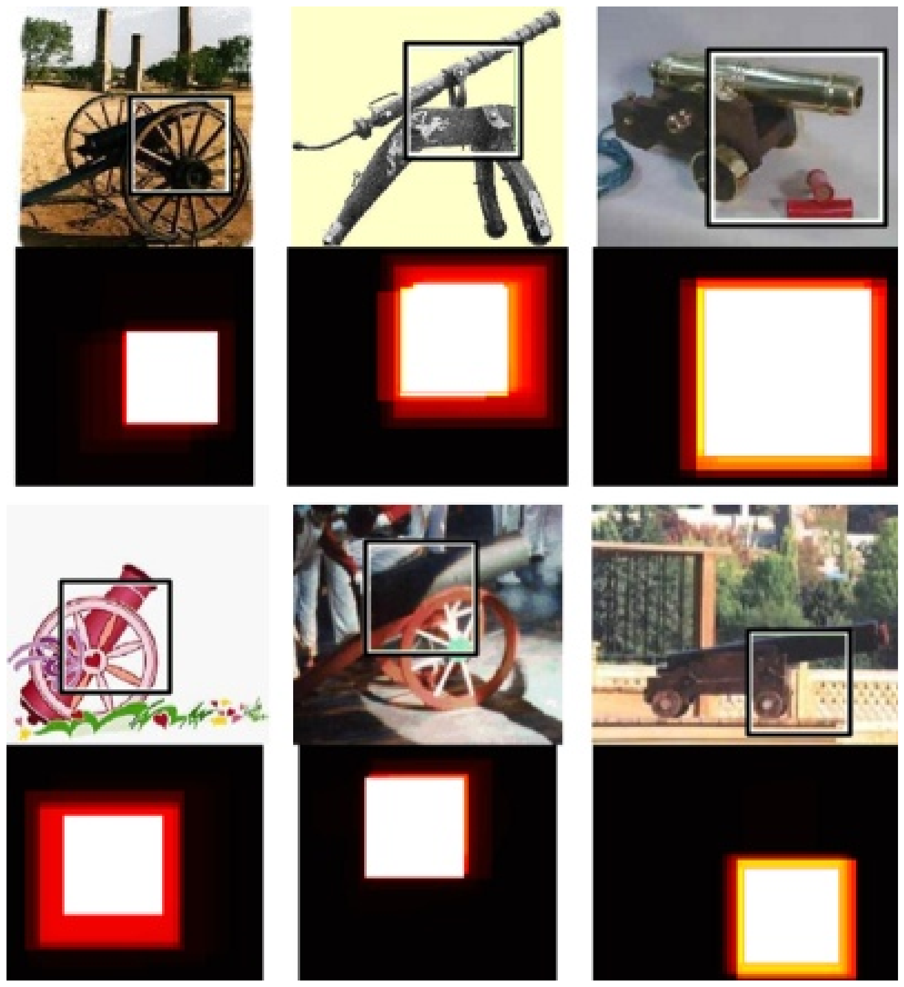


Figure 3.8: Examples of inference on novel images in the ”cannon” category. The image above shows the best classification window inferred by the model. On the bottom is the saliency map, showing the approximate posterior probability under the model that each pixel was generated by the foreground object.

tend to contain frontal upright faces near the center of the image, the variation in scale of the head across images is about three octaves and there are a considerable number of examples that are rotated both in and out of plane, contain both the head and the body, and are not centered in the image (see Figure 3.14).

For training, we used 45 positive images and 200 negative images. Training segments were collected using the same procedure as described in section 3.8.1. This resulted in about 200,000 image segments per category. Figure 3.12 shows a subset of the training segments resulting from one positive image and several negative images. We used the

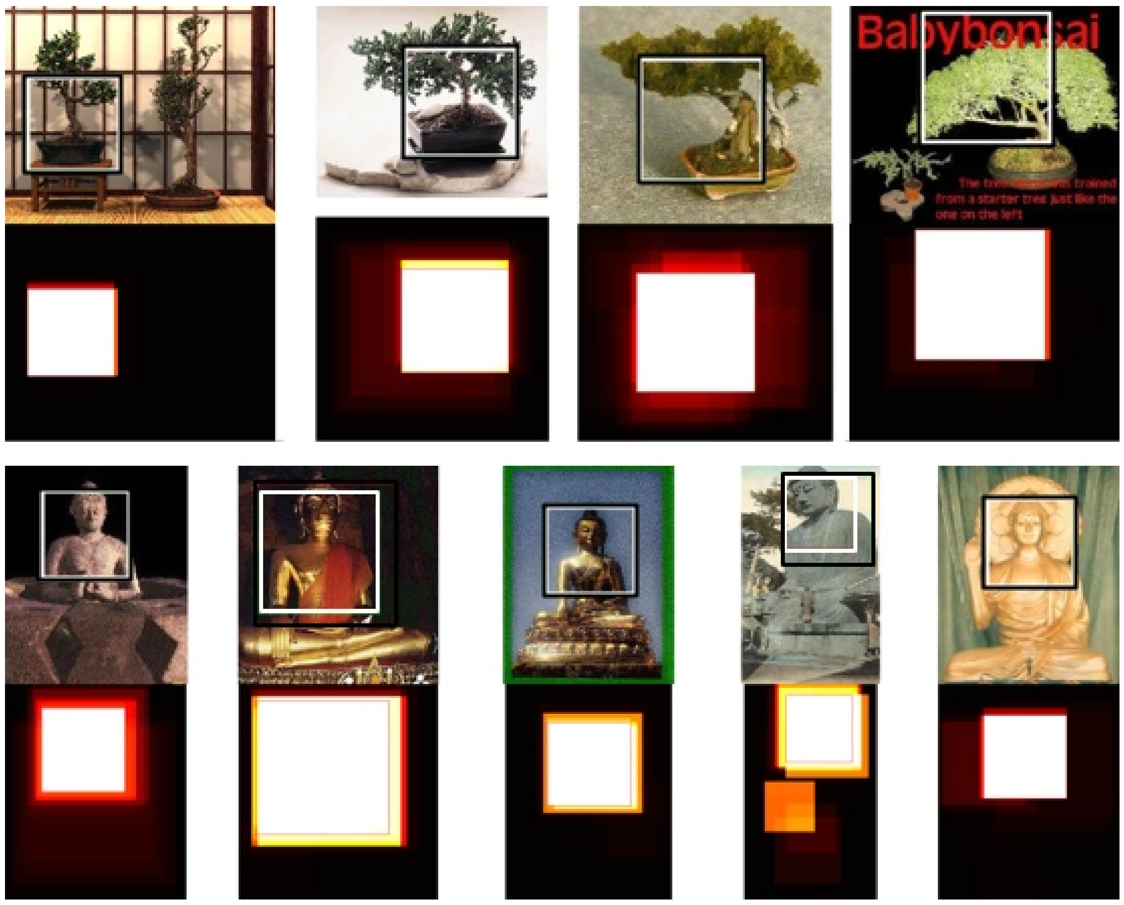


Figure 3.9: Examples of inference on novel images in the "bonsai" and "buddha" categories. The image above shows the best classification window inferred by the model. On the bottom is the saliency map, showing the approximate posterior probability under the model that each pixel was generated by the foreground object.

same parameter settings as in the previous experiments. Training 50 features took 6 hours in Matlab on a 2GHz PowerMac G5.

We performed ten cross-validation rounds of training and testing on 1000 randomly selected images from this dataset. The resulting performance was 95.0% area under the curve, and 89.0% equal error rate. The confidence intervals for these measurements were all smaller than ±0.01%. Because the parameters for learning were the same as on the Caltech-4 face dataset, the lower performance on this dataset is indicative that it is indeed much more difficult than the Caltech faces. However we were surprised that we were able to achieve even this level of performance. Figure 3.14 shows several images and their saliency maps, giving a sense of how much variety in size, head orientation, facial structure, skin color, hair, and background clutter exists in this dataset.

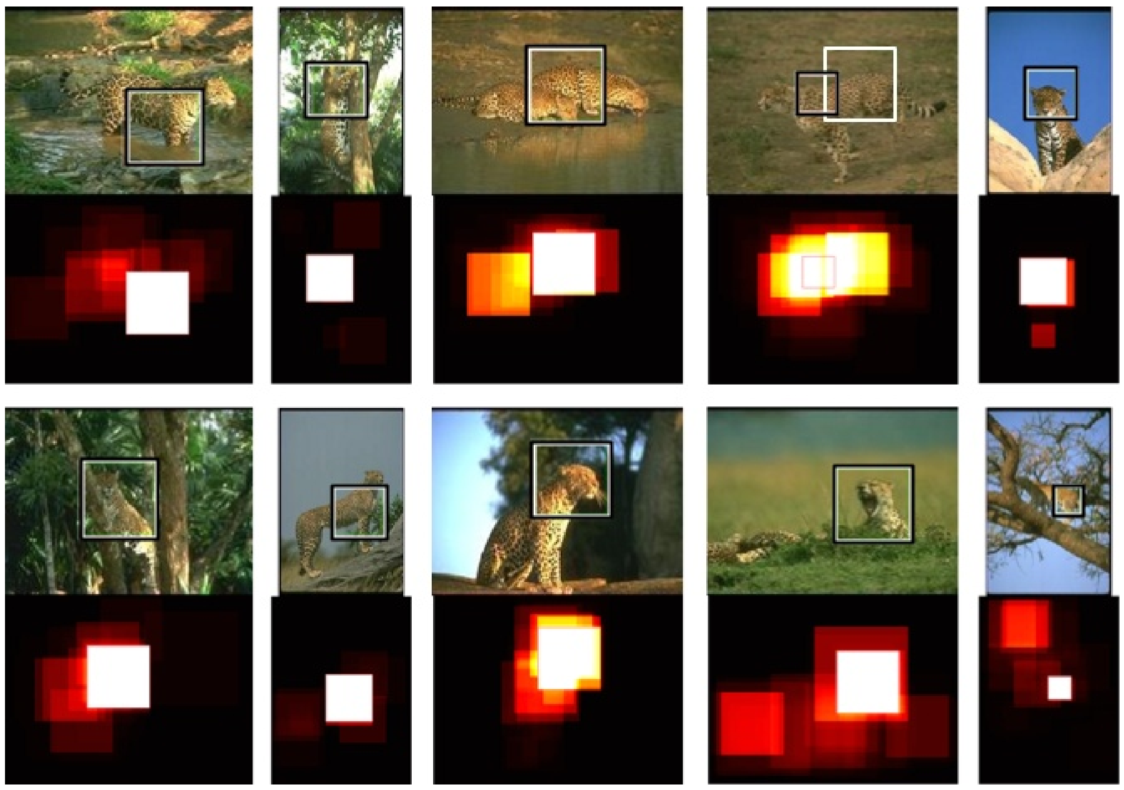


Figure 3.10: Examples of inference on novel images in the "Leopards" category. The image above shows the best classification window inferred by the model. On the bottom is the saliency map, showing the approximate posterior probability under the model that each pixel was generated by the foreground object.

In figure 3.13, we show the first ten features and tuning curves learned for this dataset.

GENKI represents a new kind of challenge for this problem, compared to the Caltech-101 object categories, because of the degree of variability in the data sets, particularly in size. While images in the Caltech-101 dataset tends to be fairly well centered, cropped and oriented in roughly the same direction, GENKI contains many more images in which the faces are far off center, extremely large or extremely small, partially occluded, etc.

3.9 Conclusions

The system we have presented uses a generative modeling approach to derive a weakly supervised learning algorithm for discovery of object categories from images labeled only as containing or not containing an object of interest. The resulting algorithm

Table 3.2: Ten best performing classifiers for Caltech-101 dataset for the current system compared to those of Serre et al. [2005].

Current system

% error (eq. pt)	category
99.25 ± 0.8	'pagoda'
99.18 ± 0.3	'car side'
99.06 ± 1.3	'accordion'
98.88 ± 0.7	'trilobite'
98.69 ± 1.9	'panda'
97.82 ± 1.8	'cellphone'
97.75 ± 1.1	'windsor chair'
97.70 ± 1.0	'minaret'
97.43 ± 0.3	'airplanes'
97.38 ± 1.6	'metronome'

Serre et al. [2005]

% error (eq. pt)	category
100.0	'metronome'
99.5	'inline skate'
98.3	'scissors'
98.1	'pagoda'
97.9	'trilobite'
97.3	'faces'
97.2	'accordion'
96.2	'minaret'
95.7	'faces easy'
95.7	'car side'

Table 3.3: Ten worst performing classifiers for Caltech-101 dataset for the current system compared to those of Serre et al. [2005].

Current system

% error (eq. pt)	category
79.71 ± 5.6	'dragonfly'
79.53 ± 6.4	'wheelchair'
79.48 ± 8.2	'cougar body'
79.04 ± 9.0	'anchor'
78.48 ± 2.4	'watch'
78.43 ± 6.4	'nautilus'
78.26 ± 9.0	'strawberry'
78.00 ± 6.4	'chair'
75.65 ± 4.0	'umbrella'
74.32 ± 5.2	'starfish'

Serre et al. [2005]

% error (eq. pt)	category
73.0	'chair'
72.1	'barrel'
72.1	'ibis'
71.6	'octopus'
71.3	'cup'
71.1	'cannon'
70.8	'wheelchair'
70.6	'lamp'
68.4	'flamingo'
62.9	'ewer'

achieves state-of-the-art accuracy on two public data sets of images containing multiple object categories, as well as a new dataset of 75000 face images, while also achieving much

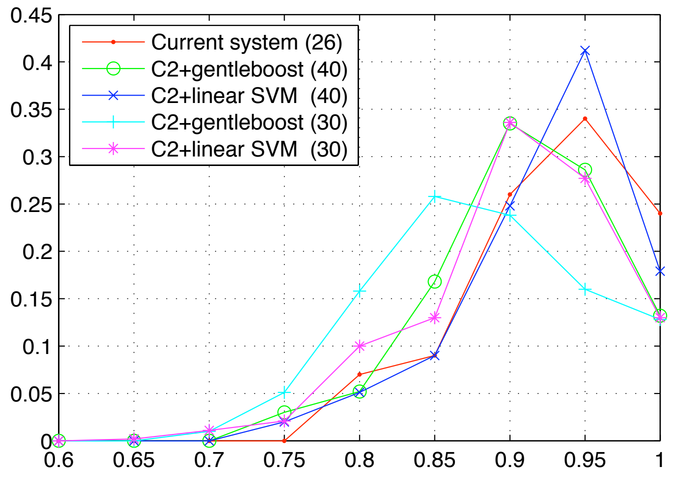


Figure 3.11: Histograms of area under the ROC curve for generalization on the Caltech-101 object categories for different systems. We show the current system compared to the C2 systems from [Serre et al., 2005] with different classification methods and numbers of training examples. The parentheses indicate the number of positive training examples. The best performing system is our system, which used an average of 26 positive images per category. The C2 systems were trained with 30 or 40 examples, using GentleBoost or linear SVM classifiers.

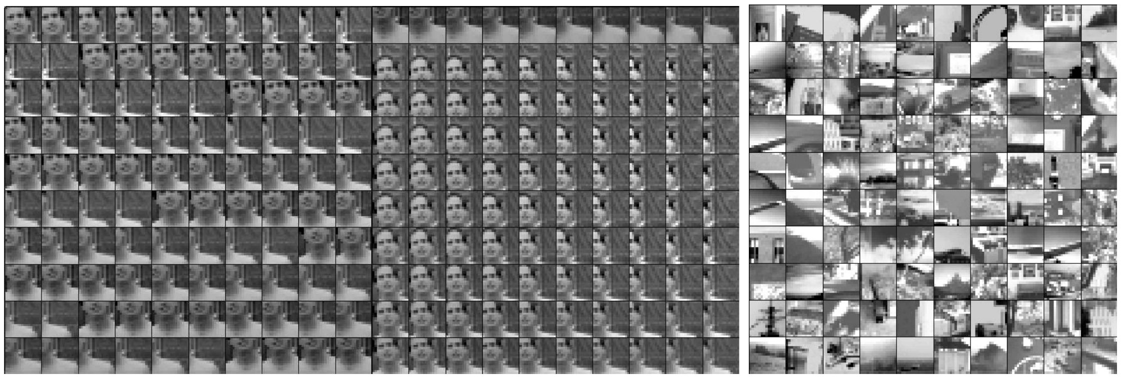


Figure 3.12: (left) Subset of segments from a single example face image, (right) examples of segments from the nonface images

faster run-time performance than any other weakly-supervised system we are aware of. It has an intuitive interpretation as a convolutional neural network with 5 million hidden units. While our emphasis has been on visual object detection, the technique in fact relies on a much more general concept of an "object" as a subset of dependent data that is conditionally independent from the other subsets within the container set. This

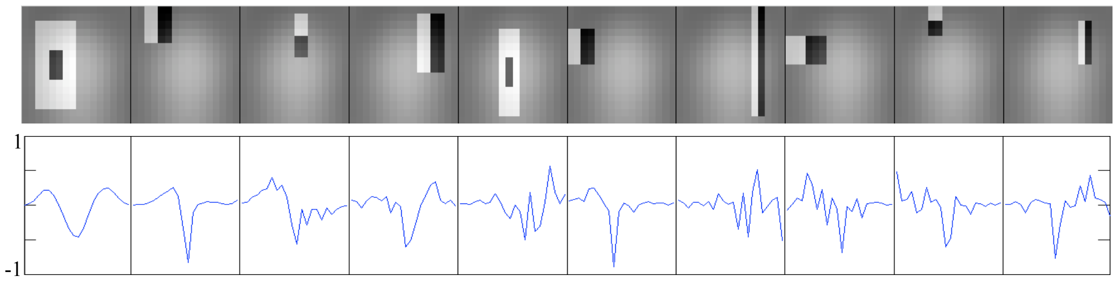


Figure 3.13: (top) The first ten features and (bottom) their respective tuning curves

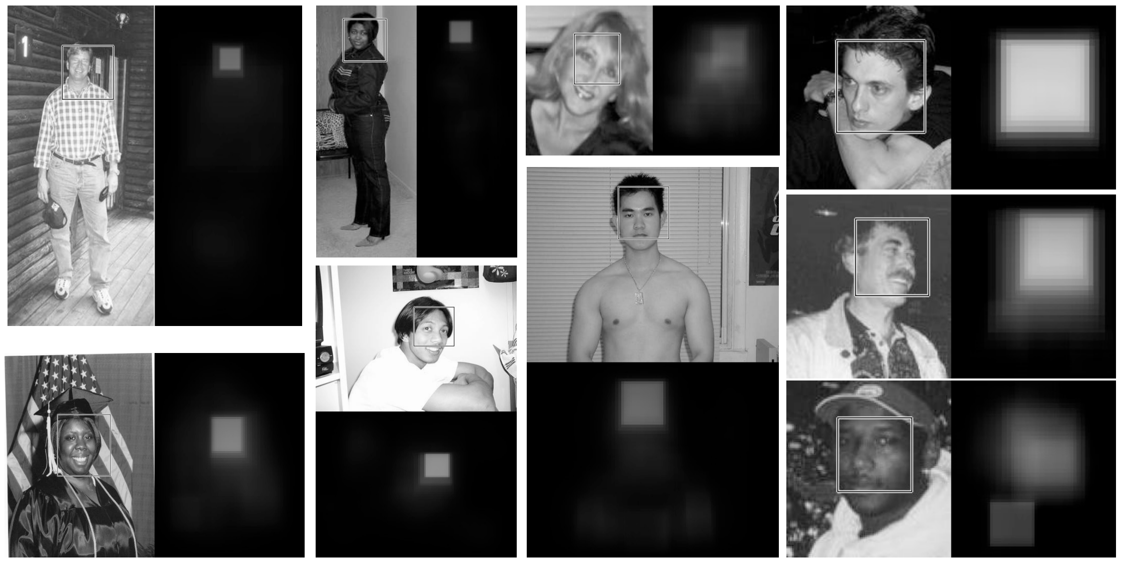


Figure 3.14: Face images and their saliency maps

makes it well suited to other domains in which the underlying cause making two streams of data “different” is an unknown, arbitrary subset of that stream which is assumed to be dependent, but independent from the other subsets in the stream. For instance, this could be useful in analysis of touch or auditory data, sequences of DNA or RNA, security systems, financial data, and many other areas.

The ability to learn from a very weak training signal suggests that it may be possible to move to a completely unsupervised system by using other modalities to provide the weak training signal. For instance, color or motion in the video stream could be used to indicate the presence of a person. This is also highly suggestive of biological systems that do not receive strong supervision but also have access to sensory cues such as motion detection, smell, or contingency. The approach presented here suggests these

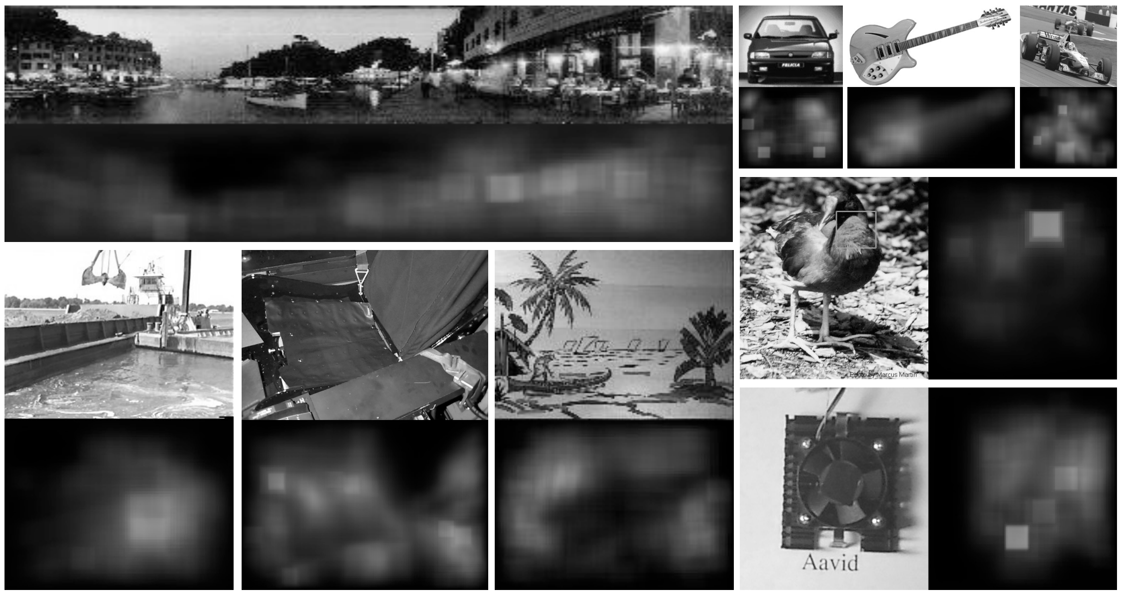


Figure 3.15: Some correctly rejected nonface images, and one false alarm (the chicken on the right), along with their saliency maps

basic features may be sufficient for organisms to discover the appearance of objects of interest, without the need to posit any specific innate knowledge about these objects. In the next chapter, we test this theory in an interactive robot which uses auditory contingency detection to generate its own weak labels about the presence or absence of a person.

3.A Theorem: Scale Invariance of the Partition Function

The background model is determined by a background distribution b for a reference size r, a collection of scaling functions $\phi_s : \mathbb{R}^s \to \mathbb{R}^r$ for $s > r$, and by a collection of distributions for generating patches of size s given patches of size r. Under the model a patch y of size s is generated by sampling a patch x of size r with probability $b(x)$ and then sampling a patch y of size s with probability $p(y \mid x)$, where

$$p(y \mid x) \overset{\text{def}}{=} \delta(\phi(y), x)\, g(y, x), \text{ for } x \in \mathbb{R}^r,\ y\mathbb{R}^s \tag{A-1}$$

where the function g is such that

$$\sum_y p(y \mid x) = 1 \tag{A-2}$$

In other words, given a "seed" patch x of size r we put the constraints that a patch y of size $s > r$ can only be derived from x if the scaled down version of y looks like x. Thus,

$$p(y) = \sum_x b(x) p(y \mid x) = \sum_x b(x) \delta(\phi(y), x) g(y, x) = b(\phi(y))\, g(y, \phi(y)) \tag{A-3}$$

and

$$Z(s, f, b) \overset{\text{def}}{=} \sum_{y \in \mathbb{R}^s} p(y) e^{f(\phi(y))} = \sum_{y \in \mathbb{R}^s} b(\phi(y))\, g(y, \phi(y)) e^{f(\phi(y))} \tag{A-4}$$

$$\sum_{y \in \mathbb{R}^s} \sum_{x \in \mathbb{R}^r} \delta(\phi(y), x) b(x)\, g(y, x) e^{f(x)} \tag{A-5}$$

$$= \sum_{x \in \mathbb{R}^r} b(x) e^{f(x)} \sum_{y \in \mathbb{R}^s} \delta(\phi(y), x)\, g(y, x) \tag{A-6}$$

$$= \sum_{x \in \mathbb{R}^r} b(x) e^{f(x)} \sum_{y \in \mathbb{R}^s} p(y \mid x) = Z(r, f, b) \tag{A-7}$$

which does not depend on s, or g.

3.B Summary of terms

Constants:

$\mathbb{Z}_+$	positive integers
$\mathbb{R}_+$	positive real numbers
$n_p \in \mathbb{Z}_+$	number of pixels
s_i	a segmentation, a partition of $\{1, ..., n_p\}$
$\mathcal{S} = \{s_1, ..., s_{n_k}\}$	allowable segmentations
$n_k \in \mathbb{Z}_+$	number of allowed segmentations
$\mathcal{A} = \{a_1, ..., a_{n_s}\} = \bigcup_{i=1}^{n_k} s_i$	all allowed segments
$n_s \in \mathbb{Z}_+$	number of segments
$\|a_i\|$	size of set a_i
$n_c \in \mathbb{Z}_+$	number of object categories
$r \in \mathbb{Z}_+$	smallest segment size
$\mathbb{D} = \{\mathbb{R}^r \bigcup \mathbb{R}^{r+1} \cdots \bigcup \mathbb{R}^{n_p}\}$	real valued vectors of lengths r to n_p
β	foreground contamination
n_f	number of feature detectors

Specification of the model:

$X = (X^1, X^2, ..., X^{n_p}) \in \mathbb{R}_+^{n_p}$	an image, i.e., collection of pixel intensities
$X_i = \{X^j : j \in a_i\} \in \mathbb{R}^{\|a_i\|}$	image segment
$O = (O_1, ..., O_{n_s}) \in \{0, ..., n_c\}^{n_s}$	object assignments
$Y = (Y_1, ..., Y_{n_p}) \in \{1, ..., n_c\}^{n_p}$	pixel assignments
$F \in \mathcal{F}$	foreground model parameters
$B \in \mathcal{B}$	background model parameters
$p(s)$	prior probability of segmentation s
$p(o_j\|s)$	probability object assignment O_j takes value o_j given segmentation s
$p(x_j\|O_j = o, f, b)$	probability image segment X_i takes value x_j if rendered by object category o under model parameters f and b

Other important terms:

$l(x \mid a_j, f, b)$	foreground vs. background likelihood ratio of x_j
$\phi_r(z) \in \mathbb{D} \to \mathbb{R}^r$	scaling function, scales patches from size $\rho \in \mathbb{D}$ to r
$h(z) \in \mathbb{R}^r \to \mathbb{R}$	feature detector
$f(x_j)$	foreground model parameters
$Z(a_j, f, b)$	partition function ensuring $p(x_j \mid O_j = 1, f, b)$ sums to one
$\hat{b}$	estimate of background distribution
$L(x \mid f, b)$	log likelihood of image x under models f, b
$G_f(z)$	functional gradient of L with respect to f
$L'(x \mid f, b)$	pseudo-log likelihood, a lower bound of log likelihood
$G'_f(z)$	pseudogradient of L with respect to f
$u(x, f)$	weighted histogram of segments in foreground image x
$v(f)$	weighted histogram of segments in background sample
$w(x, \hat{b})$	background weights
$\Psi = \{\psi^1, ...\psi^{n_f}\}$	features
η^i	tuning curve for feature detector using feature ψ^i
$\tilde{u}(x, f)$	feature weighted histogram of segments in foreground image x
$\tilde{v}(f, \beta)$	feature weighted histogram of segments in background sample
$\tilde{\lambda}(f, \beta)$	feature weighted histogram of pseudogradient

4

Unsupervised Object Discovery

In the previous chapter, we developed a technique by which object categories such as faces can be learned without requiring that a human indicates the location of objects in example images. All that was needed was access to a label indicating whether or not the object of interest is present or absent.

In this chapter we explore whether it is possible to learn to detect the presence and location of objects without *any* human supervision if we combine the techniques developed in the previous chapter with a system that generates its own *presence* vs. *absence* labels based on another sensory domain. Multiple researchers have pursued the idea of combining multiple, low-level cues in one modality to bootstrap learning in another modality. For example Hershey and Movellan [2000] showed that it is possible to locate faces by focusing on regions of the image plane that correlate highly with the acoustic signal. Beal et al. [2003] developed a probabilistic generative model under which a common cause generates both auditory and visual data, and used this model to infer templates of human appearance from video without supervision. Triesch and von der Malsburg [2001] used multiple low-level visual features for unsupervised person tracking in video, and de Sa [1994] developed an unsupervised learning method in which audio and video systems trained each other to classify spoken syllables. Finally, Blum and Mitchell [1998] used a similar technique for classifying web pages, in which the links to web pages and the words in the page are treated as separate modalities.

Some have suggested that the combination of multiple, low-level cues may also be a basic mechanism used in infant learning [Cohen and Cashon, 2001]. Rather than innate visual biases, simple low-level cues such as auditory, tactile, or proprioceptive input may be used as evidence for the presence or absence of objects. The features needed to perform this discrimination are learned due to their ability to help make this discrimination, not because of any implicit bias.

Watson [1972] suggested that the infant brain may be particularly sensitive to the presence of contingencies between sensory channels, and this contingency drives the definition and recognition of caregivers. Watson hypothesized that human faces in particular are learned because they tend to occur in high contingency situations. This idea developed over decades of research, but originated from an a study in which 2-month-old infants spontaneously exhibited intense social responses toward mobiles that did not look particularly human, but that responded to the infant's head movements [Watson, 1972].

To investigate the process of visual learning in systems without external supervision, we built an inexpensive robot out of off-the-shelf parts, with the aim of letting it interact with the environment and then learn about objects from self-generated labels based on multimodal input. Our goal was to explore whether auditory contingency information would be sufficient for the robot to develop preferences for human faces, to get a sense for the time scale of the learning problem, and to test whether those preferences would transfer to abstract stimuli, like 2-D drawings. We found our robot was able to rapidly learn face preferences from only minutes worth of visual data. Furthermore, it learned to identify and locate people in the visual scene reliably, even when their faces are not present.

4.1 Infant robot

We created a simple interactive robot by fitting a plush baby doll with an IEEE1394a webcam, a microphone, and a loudspeaker (Figure 4.1). These components were connected to a computer which ran a social contingency detection algorithm from Movellan [2006]. The algorithm is grounded in the theory of stochastic optimal control [Movellan, 2005], and consists of (1) an *Infomax controller* which schedules vocalizations so as to maximize the information gained about the presence or absence of contingencies, and (2) a Bayesian inference algorithm that computes the probability that a contingency is present given the observed sequences of auditory signals.

The controller periodically makes vocalizations and listens to the environment to determine as quickly as possible if a contingent agent is present. The continuous audio input was converted to binary auditory “events” by thresholding the instantaneous power from the microphone. Whenever an auditory event occurred and the posterior probability of social contingency given by the contingency detector was simultaneously above 97.5%, an image was saved with the label “contingent”. Whenever an auditory event occurred and the posterior probability of social contingency was simultaneously below 2.5%, an image was saved with the label “not contingent”.

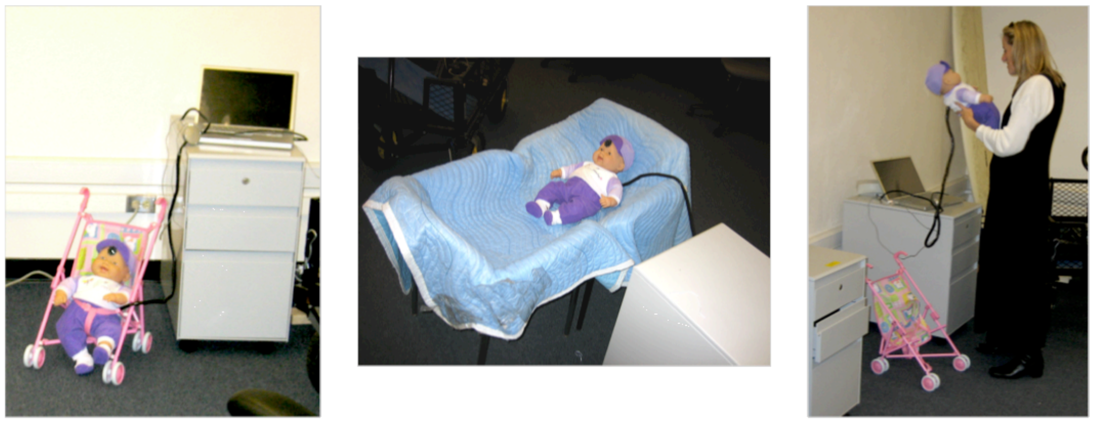


Figure 4.1: The robot used in our experiments. Two types of beginning experimental conditions, “stroller” and “crib”, are shown (left and middle respectively). The robot infant did not remain in a constant position as subjects were allowed to pick it up if they liked (right).

4.2 Data collection

We allowed the robot to run continuously for a total of 88 minutes across two sessions. During this time, the robot made vocalizations and collected images labeled by the contingency detector. During the 88 minutes of the experiment the robot was placed in three different conditions: a *chair* condition, a *stroller* condition, and a *crib* condition. For each condition the baby robot was moved so as to face one of three different backgrounds. Each condition was presented in one of two lighting conditions *bright* and *dim*. This provided 18 different background conditions (see Figure 4.1 for two example starting conditions). Within each background condition subjects could move the robot, thus constant backgrounds could not be assumed.

Nine members of the Machine Perception Laboratory at UCSD were asked to interact with the baby robot and instructed to try to make it “excited”. They were told that the robot would make excited noises if it thought somebody was responding to it, and would make bored noises otherwise. The robot could play 5 different sounds, ranked in level of excitation by the experimenter, each corresponding to a different level of the posterior probability of contingency as estimated by the contingency detector.

The subjects interacted with the robot for 2 trials of 2 minutes each; each trial began in a different background condition chosen randomly. The robot ran continuously during the 88 minutes of the experiment, including times that it was being moved to different starting conditions and as subjects entered and left the room. The experimental room was noisy due to a computer cluster in the same room, the background conversations from adjacent offices, and occasional conversations between the subjects and the experimenter.

Table 4.1: Disagreement between contingency detector vs. experimenter labels

Experimenter Label	Internal Contingency Label			
	Training Set		Validation Set	
	“Contingent”	“Not Contingent”	“Contingent”	“Not Contingent”
“Face”		21%(41/200)		16%(99/624)
“No Face”	18%(6/34)		15%(421/2843)	
“Person”		29%(58/200)		25%(154/624)
“No Person”	9%(3/34)		4%(126/2843)	

4.3 Discovery of a “Person” category

Over the course of the 88 minutes of interactions, a total of 2877 images labeled “contingent” and 824 images labeled “not contingent” were collected. From these images, a training set of 34 contingent images and 200 not contingent images were chosen at random, comprising 6.3% of all images. Since the images were collected over 88 minutes, this training set represents just under 5 minutes and 34 seconds of visual experience. All subsequent testing was done with the remaining 3467 images not used for training.

In order to better understand what the system was learning, we provided *post-hoc* labels of whether or not a face was present in the image, and whether or not a person was present in the image, to supplement the original self-generated “contingent” vs. “not-contingent” labels. Images labeled as “face” contained either a whole or part of a face. Images labeled “person” had at least some portion of a person visible. Note that all images labeled“face” were also labeled “person” and all images labeled“no person” were labeled “no face”. Table 4.1 shows the relative frequencies of the three labels. As expected, we found that the labels provided by the contingency detector were only weakly informative about the presence or absence of people in the images. For example, 29% of the images which were labeled as “contingent” did not contain people, and 9% of the images labeled as “not contingent” did contain people (see Table 4.1).

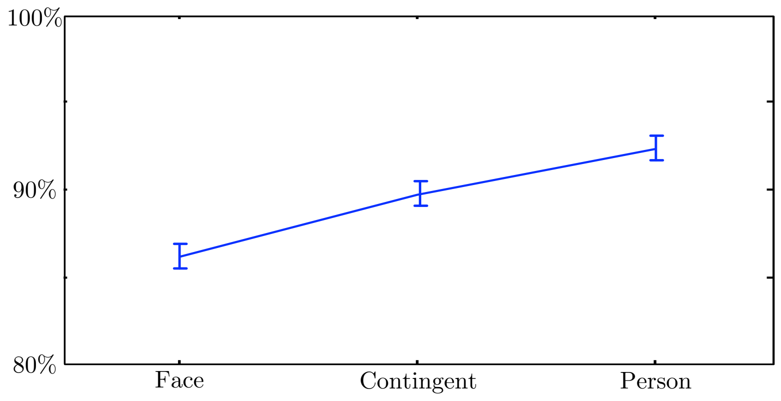


Figure 4.2: Generalization performance for “face” vs. “no face”, “contingent” vs. “not contingent”, and “person” vs. “no person” tasks. The system was only trained with the noisy “contingent” vs. “not contingent” labels. The plot shows the area under the ROC curve.

4.4 Person identification and localization

For each image in the testing set, the system made an inference about whether or not a foreground object is “present” or “absent”’ using the procedure described in 3.5.2. We then tested how well this inference predicted three different types of labels: contingency, face, and person labels. For each task, we measured the area under the ROC (AUC), which can also be interpreted as the correct classification rate on the

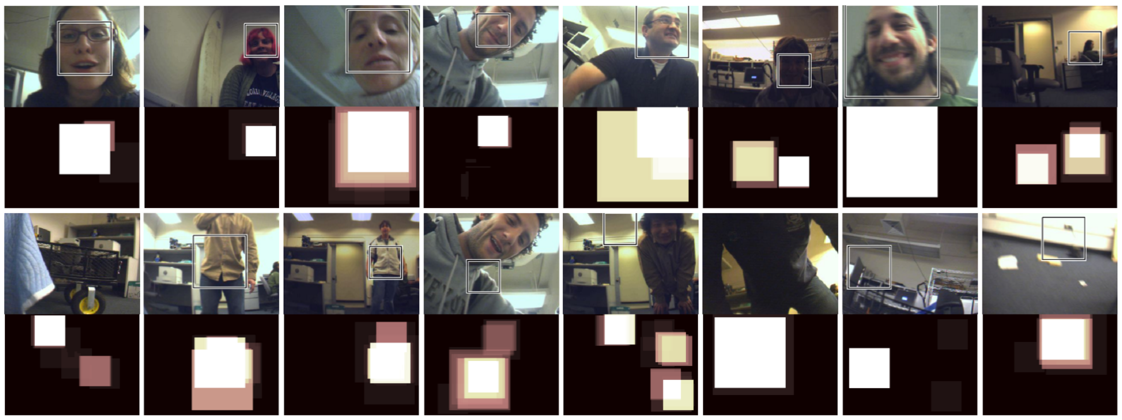


Figure 4.3: Examples images and their saliency maps. On the top row are good localization and detection results, despite variations in lighting, scale, gender, pose, and facial expression. Note that the top right image is an example that was originally labeled “not contingent”. From left to right on the bottom row: (1) correct rejection, (2)-(4) correct detections, where the body was preferred over the face, (5) the most probable location was incorrect, however the image was correctly classified, (6) an incorrect rejection, (7)-(8) incorrect detections.

two-alternative-forced-choice task (2AFC) [Cortes and Mohri, 2004]. The system's performance was 86.17% correct if we asked how well the output predicted the presence or absence of a face, 89.7% correct if we asked how well the output predicted the contingency label given to the image by the contingency detector, and 92.3 % correct if we asked how well the output predicted the presence or absence of a human (see Figure 4.2).

These performance levels are quite interesting. Although the detector was trained with image labels provided by the contingency detector, which in fact disagreed with the human labels of the presence versus the absence of people by about 26%, the detector actually achieved 92% accuracy in person detection. The labels did not provide any information about *where* people were located in the image, and there were no constraints on the views and poses of people, which sometimes contained faces, sometimes only bodies, and had wide variability in orientation, scale, and lighting conditions. Interestingly, although the system was generally quite good at finding faces, it also seemed to be very good at finding other parts of the human body. Some examples of the system's predictions are shown in Figure 4.3, showing that both head and body tend to be identified as more likely to have been caused by the foreground category.

Figure 4.4 shows several of the learned feature detectors superimposed on a sketch face. On top are the features, and on bottom are the tuning curves. For each tuning curve, the horizontal axis is the feature output, and the vertical axis is the tuning curve output corresponding to that feature output. Many of the learned feature detectors are easily interpretable. Bilateral symmetry and edges around the mouth were among the earliest selected features, and features that might be called “hairline” and “bridge of nose” detectors were also present. For example, the fourth feature gives a high output response if there is strong light-dark contrast from left to right over the bridge of nose area, low response if there is little contrast, and stays close to zero if there is strong right to left contrast.

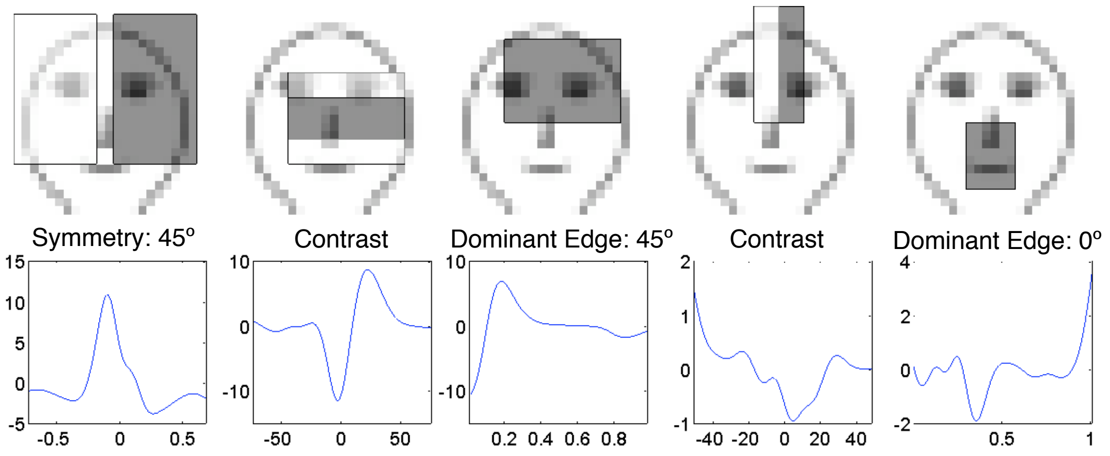


Figure 4.4: (top) Several of the most informative features superimposed on a sketch face, and (bottom) their tuning curves.

4.5 Preferences for face stimuli

Johnson et al. [1991] presented 40 minute old human infants with 3 types of visual stimuli to study their visual preferences (See Figure 4.5): (1) A drawing of a frontal face; (2) A drawing with the same features of the face but scrambled arrangement while maintaining symmetry; (3) An empty face-outline. They found that infants showed an order of tracking preference in favor the face stimulus, followed by the scrambled stimulus, followed by the empty stimulus.

We presented the baby robot with the same three stimuli used in [Johnson et al., 1991]. Recall that this system had been trained only with real visual scenes labeled by the InfoMax contingency detector as containing or not containing a contingent agent. It had never seen line drawings of faces. Despite this the approximate posterior probability map reproduced the preference order reported in Johnson et al. [1991] perfectly. The area around the face drawing was given the highest probability of coming from the contingency category, the area around the scrambled face was given somewhat less probability, and the area around the empty face was given even less probability.

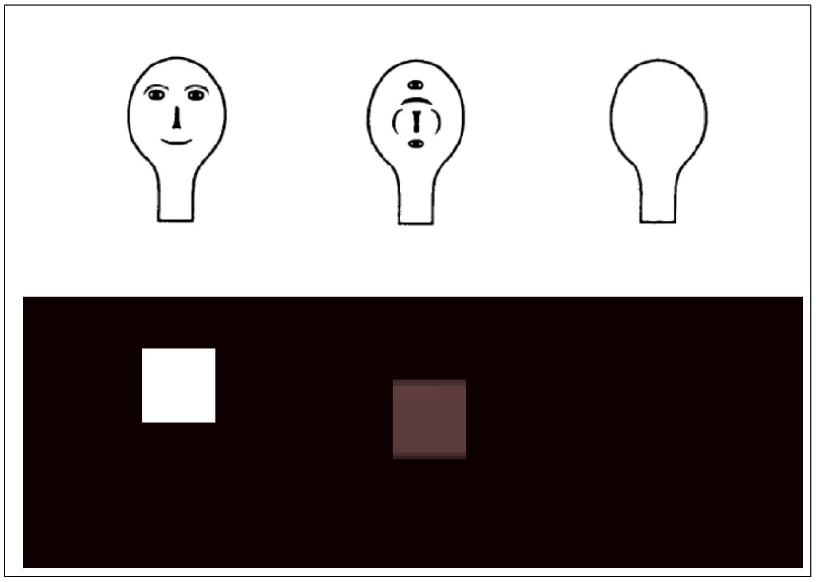


Figure 4.5: (top) The stimuli used by Johnson *et al.* to test whether neonate infants showed preferential face tracking. Infants tracked the first stimulus the most, and the third the least. (bottom) Posterior probability map from the learned model, indicting estimates of the probabiltiy that the pixel was generated by a "contingent" object. There is relatively high probability around the first image, and decreasing probability from left to right, following the order of infant preferences exactly.

4.6 Developmental implications

From a sample of only 34 images labeled as "contingent" and 200 images labeled as "not contingent", the robot's visual system was capable of detecting the presence of people in novel images with high accuracy (over 90 % correct). In doing so it developed a preference for human faces that was detectable in 2d-face drawings it had never been exposed to. The robot was never told by a human whether or not people were present in the images, or whether people were of any particular relevance at all. However it discovered that the only consistent visual explanation for two sets of scenes with differing auditory response statistics was a combination of feature detectors that happened to discriminate the presence of people. There was nothing special about the auditory contingency domain – similar results could undoubtedly be obtained using other modalities. The results illustrate that from a computational point of view, the visual preferences of the type typically investigated in human neonates can be acquired very quickly, in a matter of minutes. Previous studies that were thought to provide evidence for innate cognitive modules may actually be evidence for rapid learning mechanisms in a neonate brain exquisitely tuned to detect the statistical structure of the world. This further adds to a body of evidence that simple cues from one or several other modalities are sufficient to learn visual concepts without supervision (e.g., Triesch [2001]; de Sa [1994]).

The results show that rapid learning is a viable explanation for empirical results that had previously been thought to require innate "*units of mental architecture.*" They provide computational credibility to John Watson's views about the role of contingency on infant development Watson [1972]. Most importantly the results illustrate the importance of understanding the problems faced by the developing brain via computational experiments with real-world images and sounds.

4.7 Acknowledgments

Special thanks to collaborator Nicholas Butko who contributed significantly to the work described in this chapter.

Bibliography

John Agosta. The structure of bayes networks for visual recognition. In *Proceedings of the 4th Annual Conference on Uncertainty in Artificial Intelligence (UAI-88)*, New York, NY, 1988. Elsevier Science.

Kobus Barnard, Pinar Duygulu, Nando de Freitas, David Forsyth, David Blei, and M. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3: 1107:1135, 2003.

S. Baron-Cohen. *Mindblindness*. MIT Press, Cambridge, MA, 1995.

M. Stewart Bartlett, B. Braathen, G. Littlewort, E. Smith, and J. R. Movellan. A prototype for automatic recognition of spontaneous facial actions. In S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, number 15, pages 1271–1278. MIT Press, Cambridge, Massachusetts, 2003.

Matthew J. Beal, Nebojsa Jojic, and Hagai Attias. A graphical model for audiovisual object tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(7): 828–836, 2003.

Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH'99 conference proceedings*, pages 187–194. 1999.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998. URL `citeseer.nj.nec.com/blum98combining.html`.

Leslie B. Cohen and C. H. Cashon. Infant object segregation implies information integration. *Journal of Experimental Child Psychology*, 78:75–83, 2001.

J. F. Cohn, J. Xiao, T. Moriyama, Z. Ambada, and T. Kanade. Automatic recognition of eye blinking in spontaneously occurring behavior. *Behavior Research Methods, Instruments, and Computers*, in press.

Corinna Cortes and Mehryar Mohri. Confidence intervals for the area under the ROC curve. In *Advances in Neural Information Processing Systems*, 2004.

G. W. Cottrell, M. N. Dailey, C. Padgett, and R. Adolphs. Is all face processing holistic? the view from UCSD. In M. Wenger and J. Townsend, editors, *Computational, Geometric, and Process Perspectives on Facial Cognition: Contexts and Challenges.* Erlbaum, 2003.

N. Dalai and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 20-25 June 2005.

V. R. de Sa. Learning classification with unlabeled data. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 112–119. Morgan Kaufmann, San Francisco, CA, 1994.

S. Edelman and L. M. Vaina. David marr. *International Encyclopedia of the Social and Behavioral Sciences*, 2001.

P. Ekman. *Telling Lies: Clues to Deceit in the Marketplace, Politics, and Marriage.* W.W. Norton, New York, 1st edition, 1985.

P. Ekman and W. Friesen. *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Consulting Psychologists Press, Palo Alto, CA, 1978.

M. J. Farah, K. D. Wilson, M. Drain, and J. N. Tanaka. What is special about face perception? *Psychological Review*, 105(3):482–498, 1988.

T. Farroni, G. Csibra, F. Simion, and M. H. Johnson. Eye contact detection in humans from birth. *Proceedings of the National Academy of Sciences*, 99:9602–9605, 2002.

I. R. Fasel, E. Smith, M. R. Bartlett, and J. R. Movellan. A comparison of Gabor filter methods for automatic detection of facial landmarks. In *Proceedings of the 7th Symposium on Neural Computation.* California Institute of Technology, 2000.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Generative-Model Based Vision*, 2004.

Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.

Y. Freund and R. Schapire. A short introduction to boosting, 1999.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996a.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996b.

J Friedman. Greedy function approximation: a gradient boosting machine. Technical report, Stanford University, 1999.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting, 1998. URL `citeseer.nj.nec.com/friedman98additive.html`.

R. Frischholz and U. Dieckmann. BioID: A multimodal biometric identification system. *IEEE Computer*, 33(2), February 2000.

N. George, J. Driver, and R. J. Dolan. Seen gaze-direction modulates fusiform activity and its coupling with other brain areas during face processing. *Neuroimage*, 6(13): 1102–12, 2001.

Zoubin Ghahramani and Matthew J. Beal. Variational inference for bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000.

A. Haro, M. Flickner, and I. A. Essa, editors. *Detecting and Tracking Eyes by Using Their Physiological Properties, Dynamics, and Appearance.*, 2000. IEEE Computer Society. ISBN 0-7695-0662-3.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer, New York, 2001.

J. Hershey and J. R. Movellan. Audio-vision: Using audio-visual correlation to locate sound sources. In S. A. Solla, T. K. Leen, and K. R. Muller, editors, *Advances in Neural Information Processing Systems*, number 12, pages 813–819. MIT Press, Cambridge, Massachusetts, 2000.

M. K. Holland and G. Tarlow. Blinking and mental load. *Psychological Reports*, (31): 119–127, 1972.

J. Huang and H. Wechsler. Eye detection using optimal wavelet packets and radial basis functions (RBFs). *International Journal of Pattern Recognition and Artificial Intelligence*, 7(13), 1999.

O. Jesorsky, K. Kirchberg, and R. Frischholz. Robust face detection using the hausdorff distance. In J. Bigun and F. Smeraldi, editors, *Audio and Video based Person Authentication - AVBPA 2001*, pages 90–95. Springer, 2001.

Q. Ji and X. Yang. Real time visual cues extraction for monitoring driver vigilance. *Second International Workshop on Computer Vision Systems (ICVS2001)*, 2001.

Q. Ji and X. Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, (8):1077–2014, 2002.

M. H. Johnson. The developmental and neural basis of face recognition: Comment and speculation. *Infant and Child Development*, 10:31–33, 2001.

Mark H. Johnson, Suzanne Dziurawiec, Hadyn Ellis, and John Morton. Newborns' preferential tracking of face-like stimuli and its subsequent decline. *Cognition*, 40: 1–19, 1991.

N. Jojic and B. Frey. Learning flexible sprites in video layers. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition, Maui, HI.*, 2001.

Michael J. Jones and Paul Viola. Fast multi-view face detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2003.

C. N. Karson. Physiology of normal and abnormal blinking. *Advances in Neurology*, 25-37(49):119–127, 1988.

R. Kawashima, M. Sugiura, T. Kato, A. Nakamura, K. Hatano, K. Ito, H. Fukuda, S. Kojima, and K. Nakamura. The human amygdala plays an important role in gaze monitoring: A PET study. *Brain*, 122(4):779–783, 1999.

R. Kothari and J. Mitchell. Detection of eye locations in unconstrained visual images. *ICIP96*, 1996.

T. K. Leung, M. C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. *Fifth Intl. Conf. on Comp. Vision*, 1995.

K. Levi and Y. Weiss. Learning object detection from a small number of examples: the importance of good features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, 2004.

N. K. Logothetis and T. Poggio. Viewer-centered object recognition in monkeys. Technical Report A.I. Memo 1473, Artificial Intelligence Laboratory, M.I.T., 1994.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Functional gradient techniques for combining hypotheses. In A Smola, P Bartlett, B Schölkopf, and D Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 2000.

J. R. Movellan and J. Nelson. Probabilistic functionalism: A unifying paradigm for the cognitive sciences. *Behavioral and Brain Sciences*, 24(4):690–692, 2001.

Javier R. Movellan. Infomax control as a model of real time behavior: Theory and application to the detection of social contingency. Technical Report 1, Machine Perception Laboratory, 2005.

Javier R. Movellan. Infomax control as a model for active real-time learning and development. Unpublished, 2006.

J. Pearl. *Probabalistic reasoning in intelligent systems : networks of plausible inference.* Morgan Kaufmann, San Mateo, CA, 1988.

J. Phillips. Personal communication. DARPA Symposium on Human ID, Washington, D.C., September 29–30, 2003.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems, 17*, 2004.

H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1(20):23–28, 1998.

H. Schneiderman. A statistical approach to 3d object detection applied to faces and cars. Technical Report CMU-RI-TR-00-06, Carnegie-Mellon University, 2000.

Thomas Serre, Lior Wolf, and Tomaso Poggio. Object recognition with features inspired by visual cortex. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 994–1000, 2005.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, 1986.

Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *Proc. IEEE International Conference on Computer Vision*, volume 2, pages 1331–1338, 2005.

Kah Kay Sung and Tomaso Poggio. Example based learning for view-based human face detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20:39–51, 1998.

Jochen Triesch. The role of a priori biases in unsupervised learning of visual representations: a robotics experiment. In *Developmental Embodied Cognition DECO-2001, Workshop at the University of Edinburgh*, Edinburgh, 2001.

Jochen Triesch and Christoph von der Malsburg. Self-organized integration of visual cues for face tracking. *Neural Computation*, 13(9):2049–2074, 2001.

Ilkay Ulusoy and Christopher M. Bishop. Generative versus discriminative models for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

K. Van-Orden, T. P. Jung, and S. Makeig. Eye activity correlates of fatigue. *Biological Psychology*, 3(52):221–40, 2000.

Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer Verlag, Heidelberg, DE, 1995.

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001.

John S. Watson. Smiling, cooing, and "the game.". *Merrill-Palmer Quarterly*, 18:323–339, 1972.

Markus Weber, Max Welling, and Pietro Perona. Unsupervised learning of models for recognition. In *ECCV (1)*, pages 18–32, 2000. URL `citeseer.ist.psu.edu/weber00unsupervised.html`.

Yair Weiss. Belief propagation and revision in networks with loops. Technical Report AIM–1616,CBCL–155, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1997.

J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. In *Proc. IEEE International Conference on Computer Vision*, volume 1, pages 756–763, 2005.

L. Wiskott, J. M. Fellous, N. Krüger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.