# Object Detection and Segmentation using Discriminative Learning

Jingdan Zhang

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2009

Approved by:

Leonard McMillan, Advisor

Shaohua K. Zhou, Co-principal Reader

Stephen M. Pizer, Reader

J. Stephen Marron, Reader

Wei Wang, Reader

Hongtu Zhu, Reader

# Abstract

**Jingdan Zhang: Object Detection and Segmentation using Discriminative Learning.**
**(Under the direction of Leonard McMillan.)**

Object detection and segmentation algorithms need to use prior knowledge of objects' shape and appearance to guide solutions to correct ones. A promising way of obtaining prior knowledge is to learn it directly from expert annotations by using machine learning techniques. Previous approaches commonly use generative learning approaches to achieve this goal. In this dissertation, I propose a series of discriminative learning algorithms based on boosting principles to learn prior knowledge from image databases with expert annotations. The learned knowledge improves the performance of detection and segmentation, leading to fast and accurate solutions.

For object detection, I present a learning procedure called a Probabilistic Boosting Network (PBN) suitable for real-time object detection and pose estimation. Based on the law of total probability, PBN integrates evidence from two building blocks, namely a multiclass classifier for pose estimation and a detection cascade for object detection. Both the classifier and detection cascade employ boosting. By inferring the pose parameter, I avoid the exhaustive scan over pose parameters, which hampers real-time detection. I implement PBN using a graph-structured network that alternates the two tasks of object detection and pose estimation in an effort to reject negative cases as quickly as possible. Compared with previous approaches, PBN has higher accuracy in object localization and pose estimation with noticeable reduced computation.

For object segmentation, I cast deformable object segmentation as optimizing the conditional probability density function $p(C|I)$, where $I$ is an image and $C$ is a vector of model parameters describing the object shape. I propose a regression approach to learn the density $p(C|I)$ discriminatively based on boosting principles. The learned density

$p(C|I)$ possesses a desired unimodal, smooth shape, which can be used by optimization algorithms to efficiently estimate a solution. To handle the high-dimensional learning challenges, I propose a multi-level approach and a gradient-based sampling strategy to learn regression functions efficiently. I show that the regression approach consistently outperforms state-of-the-art methods on a variety of testing datasets.

Finally, I present a comparative study on how to apply three discriminative learning approaches - classification, regression, and ranking - to deformable shape segmentation. I discuss how to extend the idea of the regression approach to build discriminative models using classification and ranking. I propose sampling strategies to collect training examples from a high-dimensional model space for the classification and the ranking approach. I also propose a ranking algorithm based on Rankboost to learn a discriminative model for segmentation. Experimental results on left ventricle and left atrium segmentation from ultrasound images and facial feature localization demonstrate that the discriminative models outperform generative models and energy minimization methods by a large margin.

# Acknowledgments

I would like to express my sincere gratitude to my adviser Leonard McMillan. I am deeply grateful to him both for giving me freedom to explore my own ideas and for nudging me to finish when I needed motivation. His guidance, encouragement and support are always invaluable to me.

I would also like to thank my committee members: Dr. S. Kevin Zhou for his insightful advice on this research and his guidance on my career development; Dr. Stephen Pizer for his expertise in medical image analysis and his considerable effort on reading and revising my draft; Dr. Stephen Marron for his valuable feedback to keep my theory well-grounded and clearly elucidated; Dr. Wei Wang for her encouragement and for funding part of my PhD study; Dr. Hongtu Zhu for his insightful feedback.

I am indebted to all the people who have helped my research from different perspectives. I thank Dr. Martin Styner for providing the MR images, Dr. Yefeng Zheng for providing helpful suggestions, Dr. Dorin Comaniciu for his fully support, Dr. Marc Pollefeys for his valuable advice on my research, and Dr. Guido Gerig for providing a clear education on many essential image analysis concepts. I also would like to thank my fellow students in UNC, particularly Dr. Eric Bennett, Dr. Jingyi Yu, Dr. Marcelinus Prastawa, Dr. Seon Joo Kim, Dr. Hua Yang, Xiaoxiao Liu, Liangjun Zhang, Ja-Yeon Jeong, and many others. I am also grateful to the faculty and staff of the Computer Science department for providing an exciting and comfortable environment for study and research.

Finally, I would like to thank my wife Ning Deng and my parents Maoxun Zhang and Weiyi Zhu. I would not have gone this far without their constant support and unconditional love.

# Table of Contents

# List of Tables

# List of Figures

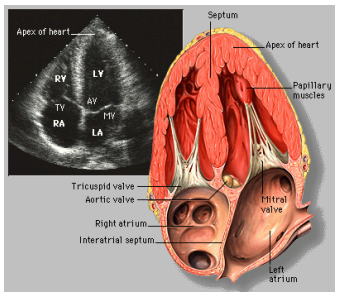
# Chapter 1

# Introduction

## 1.1 The problem

Object detection and segmentation are the tasks of retrieving knowledge of objects from images, including location, orientation, size and shape of the interested objects. These tasks are fundamental for understanding and analyzing the contents of images and they are important to many medical imaging and computer vision applications. Three example applications are presented below.

The first example application is endocardial wall segmentation in 2D echocardiograms. Electrocardiograms are ultrasound images that capture the structure of human hearts, and they are used by clinicians for diagnosing heart diseases. Figure 1.1(A) shows an echocardiogram and the corresponding anatomical diagram in the apical four-



(A)

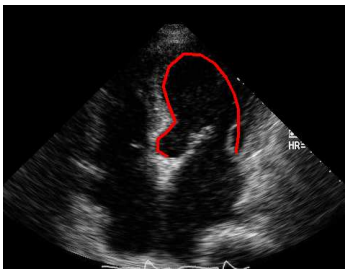
(B)

(C)

Figure 1.1: Endocardial wall segmentation in 2D echocardiogram: (A) an echocardiogram and the corresponding anatomical diagram in the A4C view (Image source: Yale atlas of echo), (B) and (C) endocardial wall segmentation of the left ventricle.

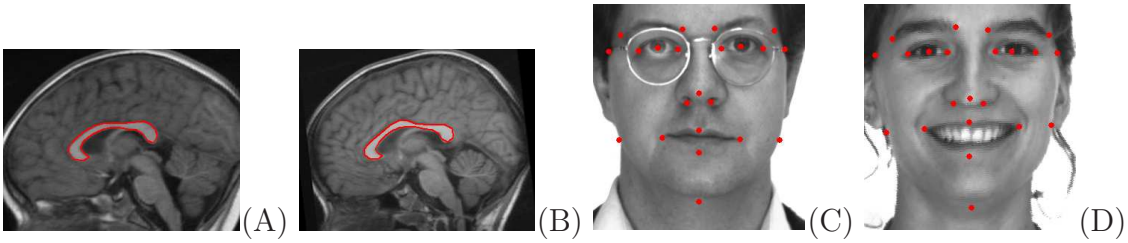



Figure 1.2: Corpus callosum segmentation and facial feature localization: (A) and (B) corpus callosum segmentation in mid-sagittal MR images, (C) and (D) feature localization in face images.

chamber (A4C) view, in which all four chambers, namely the left and right ventricles and left and right atria, are visible. In order to perform clinical analysis such as measuring the size of heart chambers, the structures indicated in the anatomical diagram (see Figure 1.1(A)) need to be identified and segmented out from echocardiograms. Figure 1.1(B) and (C) show endocardial wall segmentation of the left ventricle in echocardiograms captured from two patients. Traditionally, such segmentations are performed manually by sonographers, which is tedious and time-consuming. It is necessary to develop automatic tools for chamber detection and segmentation in order to automate the clinical work flow.

A second common clinical task is segmentation of the corpus callosum in mid-sagittal Magnetic Resonance (MR) images. Two segmentation examples are shown in Figure 1.2(A) and (B). Similar to the first application, it is desirable to have automatic tools for segmenting the object of interest.

A third real-world example task is the identification of features in face images for the purposes of registration, alignment, or tracking. Figure 1.2(C) and (D) show a set of facial features in two face images with different expressions. Automatically locating these features is a key task in many computer vision systems and medical imaging systems, for use in facial recognition, expression analysis, facial alignment, and tracking.

In the given example applications, automatic detection and segmentation are challenging due to scene complexity, large shape and appearance variation, signal dropout,

and image noise. In addition, the images in these applications are qualitatively and quantitatively different in terms of imaging modality and imaging content. It is of great interest to develop general-purpose algorithms which can cope with all these situations. In this dissertation, I present novel detection and segmentation algorithms to address these challenges.

## 1.2 Motivation

In real applications, image information is usually ambiguous and incomplete and objects and/or regions of interests have large shape and appearance variation. In order to guarantee the performance of detection and segmentation, it is often necessary to use prior knowledge or examples of target outcomes, which represent the expected shape and variability of the desired target and model the relation between an object shape and its appearance. These so called "priors" are helpful for resolving the potential confusion in images and to guide algorithms to a robust solution. For example, in Figure 1.1(A), the knowledge from the anatomical diagram can provide reliable guidance for identifying heart chambers from noisy echocardiograms.

Prior knowledge is traditionally defined by experts as explicit constraint formulations based on direct observation of object properties in images. Priors defined in this way can only encode simple knowledge, such as "boundary smoothness", "similar intensity or texture", and "high gradient boundary". They cannot capture complex shape and appearance variation of objects, and they can be easily violated when noise is present in images. Furthermore, different applications use different priors, and it is difficult to design a general detection and segmentation algorithm that can exploit all available priors. Moreover, many priors are not independent, thus assigning an appropriate weight to each is itself a non-trivial task. For example, the priors for the corpus callosum structure observed in Figure 1.2(A) and (B), such as high intensity of foreground and

a worm-like shape, are invalid in Figure 1.1 (B, C) and Figure 1.2 (C, D). Thus, a segmentation algorithm designed for corpus callosum segmentation would be difficult to apply in other applications. Because of these limitations, extensive human involvement is necessary in both defining prior knowledge and appropriately applying these defined priors to general detection and segmentation frameworks.

Alternatively, experts can implicitly define prior knowledge by annotating target objects in a set of training data. Prior knowledge can then be automatically extracted from training data and applied to new data via machine learning techniques. In this way, prior knowledge can be easily provided by experts through data annotation instead of through mathematical formulations, allowing more complicated and robust priors be learned through statistical models. Furthermore, it is possible to design general algorithms applicable to a verity of applications because prior knowledge provided by experts uses same kind of representation: images and their annotations. The Active Shape Model (ASM) [12] and the Active Appearance Model (AAM) [11] are typical examples of machine learning approaches, in which both shape and appearance priors are learned automatically. These algorithms have been applied to a wide variety of applications.

A common way to learn priors is to use generative learning, which attempt to fit a parameterized statistical model to the given data population. Several generative models are widely used in detection and segmentation applications because of their simplicity, such as Principal Component Analysis (PCA) [56], the Gaussian Mixture Model (GMM) [18], and the naive bayes [43]. However, these models have drawbacks that lead to poor performance. Generative models represent shape and appearance of objects statistically based on certain assumptions of data distribution. These assumptions are often too simple to describe the underlying complexity of real data. Also generative models focus exclusively on modeling positive examples (e.g., foreground objects). As a result, the learned models could have difficulty in discriminating between positive and negative

cases. Finally, when training generative models, it is difficult to force them to have desirable properties that might guide detection and segmentation algorithms effectively to the correct solution.

Discriminative models are trained explicitly to differentiate the foreground objects from their background. Given sufficient training examples, discriminative learning can construct models with greater discriminative power than generative models. Because of this, discriminative models are widely used in pattern recognition and many successful applications have been developed using methods like the Support Vector Machines (SVM) [84] or boosting [32]. However, in the object segmentation literature, discriminative models are less popular than generative models for two reasons. First, in training, discriminative learning needs to consider both foreground objects and their background, which is more complicated than generative learning, especially when the data's dimension is high. Second, discriminative learning needs much more training data than generative learning. Collecting sufficient training data in a high-dimensional space is a difficult task.

In this dissertation, I will develop a series of discriminative learning algorithms for object detection and segmentation. The learned priors are constrained to have certain properties that improve the performance of detection and segmentation, leading to fast and accurate solutions. To handle the complexity of discriminative learning, boosting, which is a general learning principle used to combine weak models into a stronger model, is used as a basis to develop a variety of algorithms. To demonstrate the performance of the proposed algorithms, I use several large image databases with expert annotations.

## 1.3 Thesis and contributions

Thesis statement: Boosting principles enable a new class of algorithms for fast and accurately detecting and segmenting deformable objects by learning discriminative models

from image databases with expert annotations.

Contributions:

1. I show how to use boosting to construct a variety of discriminative models. All algorithms discussed in this dissertation are based on the boosting principle.

2. I propose a network structure, named Probabilistic Boosting Network(PBN), for object detection in real time in both 2D and 3D.

3. I formulate deformable shape segmentation as a discriminative learning problem and applied three discriminative methods – classification, regression, and ranking – to solve the problem.

4. I propose a multilevel refinement approach to improve the segmentation performance.

5. I propose sampling algorithms for classification, regression, and ranking.

6. I compare and analyzed the performance of the proposed discriminate approaches using a variety of image data sets with different target objects and image modalities.

## 1.4 Overview of Chapters

The remainder of this dissertation is organized as follows:

Chapter 2 reviews the previous work of discriminative learning, object detection, and deformable object segmentation. This chapter also presents the background material for shape representation, boosting, and fast feature computation, which will be used in Chapter 3, Chapter 4, and Chapter 6.

Chapter 3 presents a learning procedure called probabilistic boosting network (PBN) for joint real-time object detection and pose estimation. The performance of PBN is

demonstrated on left ventricle detection from 3D volumes, left atrium detection from 2D images, and road sign detection.

Chapter 4 presents a regression approach to learn a conditional probability density for deformable shape segmentation. The learned density possesses a desired unimodal, smooth shape, which can be used by optimization algorithms to efficiently estimate a solution. The performance of the regression approach is demonstrated on corpus callosum segmentation and facial feature localization.

Chapter 5 extends the ideas developed in Chapter 4. It presents a comparative study on how to apply three discriminative learning approaches – classification, regression, and ranking – to deformable shape segmentation. The three discriminative learning approaches are all trained using the boosting principle. This chapter also addresses the challenge of learning in a high-dimensional space by using multi-level refinement and efficient sampling strategies. The performance of the proposed algorithms are compared and analyzed on left ventricle and left atrium segmentation from ultrasound images, and on facial feature localization.

Chapter 6 concludes with a summary of the contributions and discussion of possible future work.

# Chapter 2

# Previous work

In this chapter I provide an overview of the previous literature related to the dissertation; namely applying discriminative learning to object detection and segmentation. The literature comes from three areas: machine learning using discriminative models, object detection, and deformable object segmentation.

I first, in Section 2.1, review literature on discriminative learning, which gives a theoretic foundation to the dissertation. In this section, I present the basic idea of machine learning and discuss two common supervised learning approaches: generative learning and discriminative learning. I review the previous work comparing generative learning algorithms with discriminative learning algorithms. These comparisons justify why discriminative models should be used instead of generative models for robust object detection and segmentation when a large data set is available for training. I also present background on boosting, which is a general principle for training discriminative models. The boosting principle is used throughout the dissertation to train discriminative models for object detection and segmentation.

I then review the previous work on object detection in Section 2.2. Object detection is a common task in computer vision and medical imaging applications, which has led to a large number of algorithms. In this section, I focus on the detection algorithms based on Adaboost and its variants, which are most popular, due to their accuracy and

efficiency. I also review a technique for computing Haar-like features efficiently using integral images. I use this technique to compute image features in all the algorithms proposed in the dissertation.

I finally review the literature on deformable object segmentation in Section 2.3. In order to segment an object robustly, it is often necessary to use prior knowledge of the object. I discuss a variety of segmentation algorithms based on different kinds of shape and appearance priors. Some discussed algorithms are compared to the algorithms I propose in later chapters.

## 2.1 Discriminative learning

Machine learning is a broad subfield of artificial intelligence and it is studied in a variety of areas, including data mining, bioinformatics, and computer vision. It focuses on designing and implementing algorithms to let computers automatically "learn"/"discover" knowledge from given data and then apply the learned knowledge to analyze new data. Machine learning algorithms can be divided into two categories: supervised machine learning and unsupervised machine learning. For supervised learning, both input data and desired output data are available for learning. The learning goal is to learn a function mapping inputs to desired outputs. For unsupervised learning, only input data are available. The goal is to discover certain structure patterns conveyed by the input data and to cluster the data based on discovered patterns.

The algorithms proposed in this dissertation detect and segment object in supervised manner. In training, the input data are images and the expected outputs are expert annotations. The algorithms learn mappings between the images and their corresponding annotations. In this section, I first review two different approaches for supervised learning: generative learning and discriminative learning. I compare the two approaches and discuss why discriminative learning is used instead of generative learning in the dis-

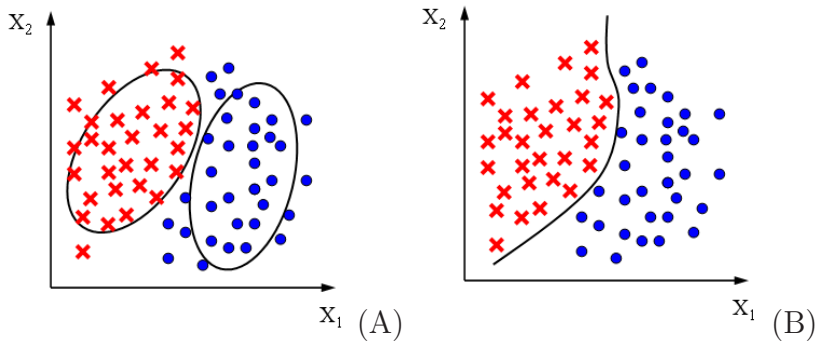



Figure 2.1: Generative learning v.s. discriminative learning for a two-class classification problem. (A) A generative model learns the data distribution of each individual class, where the distributions are represented as ellipses. (B) A discriminative model finds the boundary between the two classes.

sertation for object detection and segmentation. Then I review boosting, which is a general principle for training discriminative models.

### 2.1.1 Generative learning and discriminative learning

The goal of supervised learning is to learn a function $y = f(x)$, where input $x$ is usually a vector and output $y$ is either a discrete variable (class label) in classification or a continuous variable in regression. A typical supervised learning algorithm has two stages of computation. The first stage is the training stage (also called the learning stage), at which a function $f$ is learned from given training data, i.e., a set of training pairs $(x, y)$. The second stage is the testing stage (also called the inference stage), at which the learned function $f$ is applied to predict $y$ given a new input $x$.

From statistical point of view, both input $x$ and output $y$ can be considered as random variables. Supervised learning builds a conditional probability $p(y|x)$, describing conditional distribution of $y$ given $x$. In literature, the probability $p(y|x)$ can be learned either generatively or discriminatively.

A generative model attempts to learn a generation process for the data population of interest. Instead of modeling $p(y|x)$ directly, the model learns a conditional likelihood $p(x|y)$ and a prior probability $p(y)$. At the training stage, the likelihood $p(x|y_i)$, describ-

ing the distribution of the data population $x$ with a same class label $y_i$, is learned for every class label $y_i$. Figure 2.1(A) shows a two-class classification example, where the learned likelihood $p(x|y_1)$ and $p(x|y_2)$ are represented as ellipses. The prior probability $p(y)$ is also learned, describing the distribution of the labels in the training data. At the testing stage, the posterior probability $p(y|x)$ is computed via Bayes' theorem:

$$p(y_i|x) = \frac{p(x|y_i)p(y_i)}{p(x)} = \frac{p(x|y_i)p(y_i)}{\sum_i p(x|y_i)p(y_i)}. \tag{2.1}$$

Generative learning approaches assume a certain distribution of data and choose a corresponding statistical model to represent the distribution $p(x|y)$. In the training stage, the parameters of the model are estimated using some criteria, e.g., likelihood maximization. Based on different data distribution assumptions, a variety of models are proposed in literature, including the Gaussian Mixture Model (GMM) [18], Principal Component Analysis (PCA) [56], Independent Component Analysis (ICA) [9], and the Markov Random Field (MRF) [40].

Generally speaking, a generative model can represent data well when the data fit assumed distributions. The learned probability $p(x|y)$ is easily interpreted as the distribution of the data $x$ with the class label $y$. Along with the prior probability $p(y)$, the joint probability $p(x, y)$ can be estimated, which is useful to generate new data. The ability to generate new data is a major advantage of generative learning. In addition, the prior probability $p(y)$ is useful to constrain the estimation of the posterior $p(y|x)$ in inference. The drawback of the generative learning is that a learned model might not be accurate enough for classification problems because the model is not trained directly to classify data into different classes. Furthermore, generative learning often models details of the data distribution that may be irrelevant to classification. Another weakness is that generative learning makes assumptions about the distributions of data (e.g., the Gaussian distribution assumption in PCA), making it ineffective to model the complex

patterns of irregular distributions.

Discriminative learning focuses on boundaries partitioning classes, and it models a posterior probability $p(y|x)$ directly without the intermediate goal of learning $p(x|y)$. The probability $p(y|x)$ is usually a function of the distance of $x$ to the class boundaries. An example of a two-class classification is shown in Figure 2.1(B), where a boundary between the two classes is learned by a discriminative model. The typical discriminative models include Linear Discriminant Analysis (LDA) [22, 21], the Support Vector Machine (SVM) [84], boosting [29], the Conditional Random Field (CRF) [42], the logistic regression [33], and neural networks [1]

Discriminative models directly target finding separation boundaries between classes, and they usually have better predictive performance than generative models [35, 41, 85]. This is articulated by Vapnik [85] : "one should solve the [classification] problem directly and never solve a more general problems as an intermediate step [such as modeling $p(x|y)$]." A discriminative model does not rely on any assumption of the data distribution, and, thus, capable of handling data with complex distributions. The drawback of discriminative learning is that the learned probability $p(y|x)$ does not represent the true underlying distribution of data. It is difficult to generate new data using a discriminative model because the joint distribution $p(x, y)$ cannot be computed from $p(y|x)$. Also a discriminative model does not learn and use the prior probability $p(y)$, making it less flexible in modeling prior knowledge.

The common observation is that discriminative models need more training data than generative models to achieve a desired prediction accuracy. Discriminative learning requires sufficient training data to cover all data variations. Ng and Jordan [53] provided a theoretical and experimental comparison of Naive Bayes, a typical generative model, and logistic regression, a discriminative model having a similar mathematic form as naive bayes. They studied the relation between the number of training examples and the prediction accuracy of the models. They showed that when the training set is small,

Naive Bayes has better prediction accuracy than logistic regression. As the size of the training set increases, logistic regression catches up and overtakes the performance of Naive Bayes.

This dissertation targets applications having a large number of training images and annotations. For example, in the endocardial wall segmentation, more than 500 training images are available. Under such circumstances, discriminative models are expected to outperform generative models. In the following chapters I present several object segmentation algorithms using discriminative models and compare their performance to the generative-learning based algorithms.

Recently, hybrid approaches combining generative and discriminative learning have been proposed [63, 82, 44, 79]. These models are better than pure generative or discriminative models. In [63], a hybrid model is learned by training a large subset of model parameters generatively and the remaining small set of parameters discriminatively. The model requires less training data and achieves better prediction accuracy than either its purely generative or purely discriminative counterpart. In [82], a hybrid model is proposed for semi-supervised learning, in which both labeled and unlabeled data are used in training. Generative learning is used to label the unlabeled data and discriminative learning is used for classification. These hybrid approaches may benefit my future research in object detection and segmentation.

### 2.1.2 Boosting

Boosting is a general principle for training discriminative models. It creates a strong learner (or strong model) from a set of weak learners (or weak models), each of which is a simple model giving prediction only slightly better than random guessing. The strong learner is an ensemble of weak learners, capable to produce a more accurate prediction by using a voting mechanism. Boosting is based on an observation that, in a lot of machine learning problems, it is hard to find a suitable strong model with desired

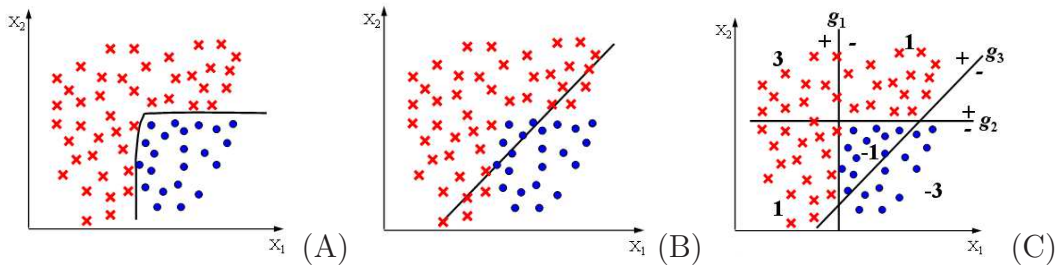



Figure 2.2: A simple example of two-class classification problem: (A) a complex strong classifier with a zigzag boundary; (B) a simple weak classifier with a linear boundary; (C) a strong classifier $f(x) = g_1(x) + g_2(x) - g_3(x)$, where $x$ is a two-dimensional vector and $g_1(x)$, $g_2(x)$, and $g_3(x)$ are linear classifiers. The value of $f(x)$ in each region is shown. It is easy to verify that $f(x)$ is all positive in the regions occupied by one class and is all negative in the regions occupied by another class.

prediction accuracy. On the other hand, it is relatively easy to find a set of weak models with moderate accuracy.

Figure 2.2 shows a simple example of a two-class classification to illustrate boosting. There are two classes in a $2D$ feature space in the figure. We want to build a classifier to discriminate the two classes. One extreme is to directly learn a strong classifier with a zigzag boundary, as shown in Figure 2.2(A), which partitions the two classes. Although finding such a classifier in 2D is not difficult, the problem is harder as the dimension of a feature space increasing. For a feature space with very high dimension (e.g., several thousand features), learning a strong classifier capable of separating classes well is a non-trivial problem. Another extreme is to use a weak classifier with a linear boundary as shown in Figure 2.2(B). Even though only having moderate classification accuracy, the linear classifier is simpler, scales better to higher dimensions, and is easier to evaluate than the strong classifier in Figure 2.2(A). In order to achieve a higher classification accuracy and still benefit from the simplicity of linear models, a strong classifier is built by combining several linear classifiers as shown in Figure 2.2(C). The strong classifier $f$ is a linear combination of three weak classifiers: $f(x) = g_1(x) + g_2(x) - g_3(x)$, where $x$ is a two-dimensional feature vector and the output of a weak classifier is either $+1$ or $-1$. The class label of an instance $x$ is determined by the sign of $f(x)$. Figure 2.2(C)

shows the value of $f(x)$ in regions partitioned by three weak classifiers. We can easily verify that $f(x)$ is all positive in the regions occupied by one class and it is all negative in the regions occupied by another class. The classifier has high classification accuracy while still maintaining the simplicity of a linear model.

Boosting provides a unified framework for building strong models using a set of weak models. This capability is especially valuable to image analysis applications, where a large number of features can be extracted from images. Due to image noise and appearance variations of objects, these features only provide weak evidence of the information we are interested in extracting. Boosting can build a robust model by combining these weak evidence. The strength of boosting to object detection and segmentation is demonstrated throughout the dissertation.

The boosting idea was introduced by Schapire [70] and Freud [25]. These original algorithms could not adapt to weak learners and suffered from practical drawbacks. Adaboost, from adaptive boosting, was proposed by Freud and Schapire in 1995 [28]. Adaboost solved many practical problems of early boosting algorithms and raised great interest in the machine learning research community. Since then, a variety of boosting algorithms have been proposed, including GentleBoost [29], LogitBoost [29], LPBoost [17], TotalBoost [87], BrownBoost [26], and MadaBoost [19].

Boosting works by iteratively selecting weak learners and adding them to a final strong learner. Although boosting algorithms use different mathematical formulas, they share the same computation procedure as shown in Figure 2.3.

The basic property of boosting is that it combines weak learners into a strong learner; namely a boosted strong learner is a more accurate predictor than any weak learners. Freund and Schapire [28] gave a bound of the training error (the fraction of a strong learner's prediction mistakes on the training set) to Adaboost, proving that the training error of Adaboost drops exponentially fast if weak learners have better prediction accuracy than random guessing. They further gave a bound of the generalization error (the

**The meta-algorithm of boosting**

1. Determine the initial weights of training data. The weight of a training instance represents the importance of the instance.
2. Train candidate weak learners independently using the weighted training data. In training, more emphasis is placed on instances with large weights.
3. Select a weak learner with the highest prediction accuracy. Take the weak learner out from the candidate pool and add it to the strong learner. The selected weak learner is weighted in terms of its accuracy. A weak learner with higher accuracy has larger weight, allowing it have a larger contribution to the strong learner.
4. Reweight the training data based on the updated strong classifier. The weights of instances with correct predictions are decreased. The weights of instances with incorrect predictions are increased.
5. Repeat step 2, 3, and 4, until desired accuracy is achieved or no weak learner is left in the candidate pool.

Figure 2.3: The boosting computation procedure.

fraction of prediction mistakes on the unseen test set), showing that the generalization error of a strong learner is bounded by its training error, the size of training sample, the number of boosting rounds, and the Vapnik-Chervonenkis (VC) dimension [1]. One observation is that Adaboost often does not overfit [5, 20, 62]. In order to explain this phenomenon, Schapire et al. [69] proposed an alternative approach to analyze the prediction error of a strong learner based on the margins of the training data. They proved that the generalization error is bounded by the margins of the training set, which are continuously reduced when the number of the boosting round increases. The margin theory builds a strong connection between boosting and the support vector machine [84].

Boosting can be interpreted as a computation procedure for fitting an additive model using a set of "basis" functions [32]. An additive model $f$ is a linear combination of

[1]VC dimension [4] is used in computational learning theory to measure the prediction capacity of a weak learner.

basis functions with the following form:

$$f(x) = \sum_{t=1}^{T} \alpha_t g_t(x; \eta_t); g_t(x; \eta_t) \in \mathcal{G} \tag{2.2}$$

where $x$ is a input feature vector, $\mathcal{G}$ is a weak learner set, $g_t(x; \eta_t)$ is a weak learner with parameters $\eta_t$, and $\alpha_t$ is the weight of $g_t(x; \eta_t)$.

Training a discriminative model is to define a loss function $L$ and estimate the parameters of the additive model $f$ by minimizing the loss function

$$\hat{f} = \arg \min_{(g_t \in \mathcal{G}, \alpha \in \mathbf{R}), t=1,\ldots,T} \sum_{n=1}^{N} L(y_n, \sum_{t=1}^{T} \alpha_t g_t(x; \eta_t)), \tag{2.3}$$

where $N$ is the number of training instances, and $y_n$ is the expected output for the input vector $x_n$. The loss function $L$ gives a large penalty when the output $f(x_n)$ is inconsistent to the desired $y_n$.

Equation 2.3 can be optimized by exhaustively selecting weak learners from a candidate pool and determining their weights. This is a combinational problem with high computational cost. Finding the optimal solution is computationally prohibitive when there are a large number of weak-learner candidates. Forward stagewise additive modeling is a practical alternative to find suboptimal solutions thanks to the additive nature of the model

$$f_t(x) = f_{t-1}(x) + \alpha_t g_t(x; \eta_t), \tag{2.4}$$

where at stage $t$, a weak learner $g_t(x; \eta_t)$ is selected and added to the strong learner without updating the parameters of the previous added weak learners. When selecting a weak learner, a stagewise loss function, which is more computational feasible, is used

$$(\hat{g}_t, \hat{\alpha}_t) = \min_{(g_t \in \mathcal{G}, \alpha_t \in \mathbf{R})} \sum_{n=1}^{N} L(y_n, f_{t-1}(x) + \alpha_t g_t(x; \eta_t)), \tag{2.5}$$

Forward stagewise additive modeling is essentially a greedy approach to minimize

the loss function 2.3 in the function space of the additive model 2.2. Many boosting algorithms can be explained using this framework. Friedman et al. [29] proved that Adaboost is equivalent to forward stagewise additive modeling using an exponential loss function. They showed that loss functions play an important role in a boosting algorithm's robustness. They further proposed the Logitboost algorithm using the logistic loss function, which is more robust for training data having outliers. Other robust loss functions were also proposed [32, 30].

## 2.2 Object detection

Object detection has been widely studied in a variety of computer vision and image analysis applications. In this section, I focus on learning based detection algorithms and pay particular attention to detection algorithms based on boosting. More comprehensive reviews of detection algorithms can be found in [91, 52]. I also introduce Haar-like features to represent image information.

### 2.2.1 Learning based object detection

The goal of object detection is to identify all image regions containing objects of interest from a given image or video. It is often a first step when extracting high level information from images. Due to a large number of objects in the real world, it is impractical to build a general object detection system that can detect and categorize arbitrary objects. So far, research on object detection focused on detecting specific objects, such as face detection [91], pedestrian detection [61], vehicle detection [52], and anatomic structure detection in medical imaging [31, 95, 3].

In the object detection literature face detection attracted most attention because it is a key component in many human computer interaction applications, such as face recognition, face tracking, and facial feature recognition. Face detection is challenging

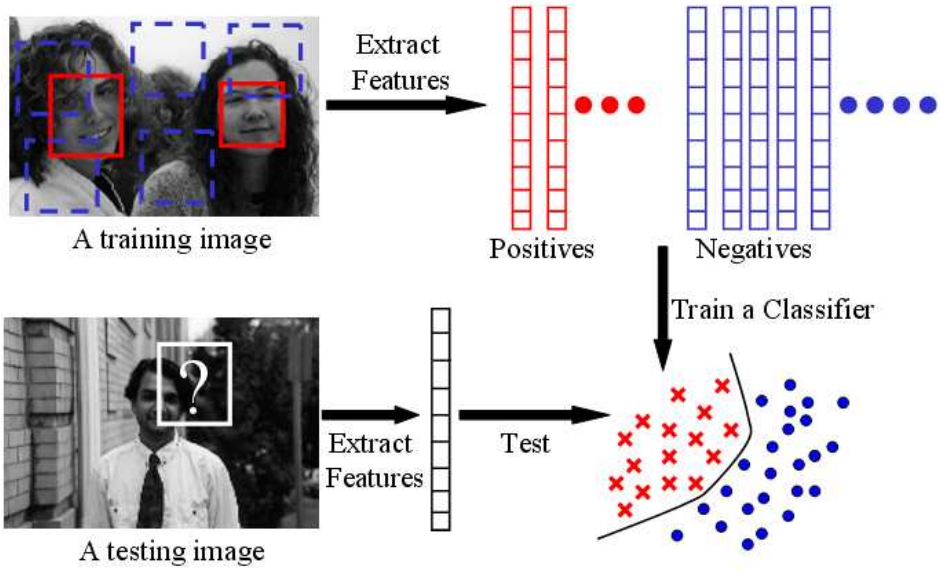



Figure 2.4: The workflow of learning based detection. In training, the sub-windows containing faces (red windows) are considered as positives, and the sub-windows without face (blue window) as negatives. Feature vectors are extracted from sub-windows and a classifier is trained to separate positive from negative. In testing, all possible sub-windows are checked by the classifier to find faces. The face images used in the figure is from the CMU frontal facial database [65].

because faces have high degrees of variability in size, shape, color, and texture. The variations of lighting, 3D face pose, and occlusion in capturing images further complicate the problem. On the other hand, face detection is a well defined problem in the sense that face images have a unique appearance and structure to differentiate themselves from background. For these reasons, face detection is a representative case of object detection and it servers as a testbed for object detection techniques. A lot of techniques have been proposed in the face detection literature. A detailed survey of face detection techniques can be found in [91].

As pointed out in the face detection survey [91], learning based detection algorithms have been most effective and have therefore have received most attention. These data-driven algorithms rely on training sets to learn rules for differentiating objects of interest from their background. The trained detectors show excellent performance when the training set is big enough to cover all the variations of objects of interest. Compared

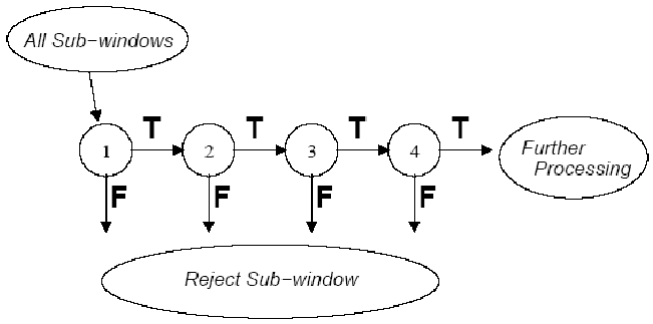



Figure 2.5: A cascade of boosted classifier [86]. A sub-window has to pass all the classifiers in the cascade to be considered as a positive. If a sub-window is failed by any classifier, it is regarded as a negative. A large number of sub-windows in an image contains background and they will be failed by the classifiers at the early stages of the cascade. This saves computation time.

with other detection algorithms, learning based algorithms make fewer assumption of the objects' properties, and they can easily be adapted to detect different objects. These algorithms work by formulating object detection into a two-class classification problem. The workflow of the algorithms is shown in Figure 2.4. In training, the sub-windows containing faces are considered as positives and the sub-windows containing background as negatives. Feature vectors are extracted from sub-windows, and a classifier is trained to separate positives from negative in the feature space. In testing, faces are detected by scanning a sub-window over the input image and applying the trained classifier to determine whether the sub-window contains a face or not.

Learning based detection algorithms differ in terms of how they extract image features and how they use classification algorithms. A wide range of image features can be used, including color, edge, histogram of intensity, et al. Due to the high dimensionality of a feature space, feature selection or dimension reduction is often necessary for the sake of computation efficiency. A variety of classification algorithms have been used in object detection. Rowley et al. [65] built a face detector based on neural network. Schneiderman [71] proposed a naive Bayes approach for detecting faces. Samaria [68] adopted a Hidden Markov Model (HMM) approach. Osuna et al. [55] used SVM to build a face detector.

Many learning based detection algorithms involve heavy computation, making them difficult to be applied to realtime applications, where the detection speed is important. Viola and Jones [86] introduced a fast detection system capable of detecting frontal-view faces in real time while it still has the detection accuracy equivalent to the best previous systems. The following three key ideas are presented in their seminal work.

1. Haar-like features are applied to represent image information. These features can be efficiently evaluated using integral images [86]. By varying the parameters of these Haar-like features, a large number of image features can be obtained. I also use Haar-like features in all the algorithms proposed in the dissertation. I will introduce them in details in the next section.

2. Adaboost is used to construct a classifier. Each Haar-like feature is associated with a weak learner, providing weak evidence of the objects' existence. Adaboost selects and combines weak learners to build a strong learner. In testing, only a small set of features are evaluated, rendering an efficient computation.

3. A cascade structure, as shown in Figure 2.5, is introduced to decrease computation by focusing on image regions having high probability of containing objects of interest. The cascade is formed by a set of successively more complex and hence more discriminative Adaboost classifiers. For a sub-window to be considered as a positive, namely containing an object of interest, it must pass all stages of the cascade. The computational efficiency is gained by the fact that the low-level classifiers can filter out a large number of negatives with a low computational cost. Only a small number of candidates, having high likelihood of containing objects of interest, advance to higher levels, which involve more computation to differentiate between positives and negatives.

The cascade of boosted classifiers introduced in [86] rapidly became popular in the computer vision community. It has been used in a number of detection applications

in addition to face detection, such as pedestrian detection [61], vehicle detection [39], and anatomic structure detection in medical imaging [31]. There are also extensive works to further improve the detection performance by using new methods to replace part of Viola's framework. Liu and Shum [46] proposed Kullback-Leibler boosting to achieve better detection accuracy through replacing Adaboost classifiers in the cascade with KLBoosting classifiers, which select features by maximizing Kullback-Leibler(KL) divergence between two classes. Xiao et al. [67] used a chain structure instead of the cascade structure to integrate historical knowledge of tested classifiers, enabling it to use fewer weak learners. Torralba et al. [77] proposed a feature sharing scheme to detect multiple objects efficiently.

The original system in [86] only detects objects with fixed pose (e.g., front face). In real applications objects of interest may be captured by cameras in arbitrary poses, causing large scale and rotation variations of objects in images. While the fast evaluation of local rectangle features is applicable when dealing with scale variation, it is not straightforward when applying it to the rotation variation. A simple approach is to train a classifier by pooling data from all orientations as positives. The variation introduced by rotation is treated as intra-class. However, as previously pointed out [34, 37, 45], this treatment causes a significant increase in appearance variation, rendering the detector complex and inaccurate. Also, it is impossible to recover rotation information. Another possibility [31] is to rotate the image for a set of chosen orientations and compute the integral image for each orientation. However it is computationally prohibitive when the size of an image is large.

A promising solution that requires only one integral image is to train a collection of binary classifiers to handle different poses. A variety of structures are proposed to combine these classifiers. The most straightforward way is a parallel structure (Figure 2.6(A)) that trains a classifier for each discretized pose [88]. In detection all classifiers are tested for every scanning window. The computation linearly depends on the number

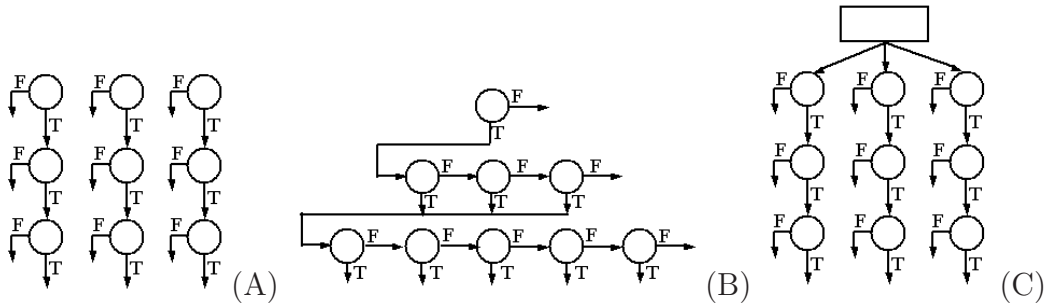



Figure 2.6: Different structures for multi-pose detection: (A) parallel structure, (B) pyramid structure, and (C)tree structure. The circle represents a binary foreground/background classifier. The rectangle represents a multiclass pose classifier.

of poses. To accelerate the detection speed, a pyramid structure (Figure 2.6(B)) was proposed by Li *et al.* [45]. This approach adopts a coarse-to-fine strategy to train classifiers with increasingly narrower pose ranges. In the parallel structure several classifiers might fire up at the same place when the actual pose is in-between the discretized poses. To estimate an accurate pose needs additional work due to the difficulty in comparing responses among these classifiers. To discriminate different poses explicitly, a tree structure (Figure 2.6(C)) that uses multiclass classifier as a differentiator is applied. In [37] a decision-tree is used to determine the pose of a face, followed by the binary classifier trained for that pose only. Most recently, Huang *et al.* [34] proposed a tree structure to divide the pose space into increasingly smaller subspaces. The foreground/background discrimination and pose estimation are performed at the same node using the Vector-Boost algorithm.

### 2.2.2 Haar-like features and integral image

In this section I introduce the image features used in the dissertation. There are many reasons for using features instead of raw pixel values to represent image appearance. The major one is that image features provide low-level descriptions of image contents, which are valuable for many image analysis applications. Many image features have been proposed in the literature, such as the Canny edge detector [6], steerable filters [24], and

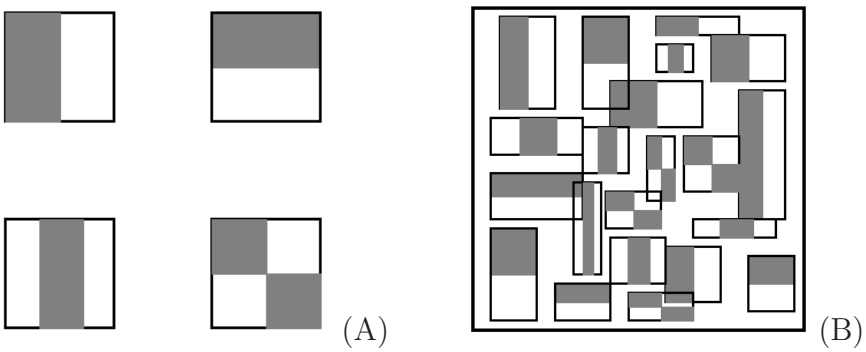
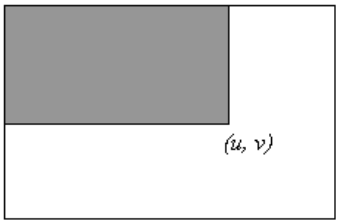



Figure 2.7: Haar-like features: (A) the four type of features used in [86]. (B) tens of thousands of possible Haar-like features in an image window, which are generated by varying the scales and positions of each type of features.

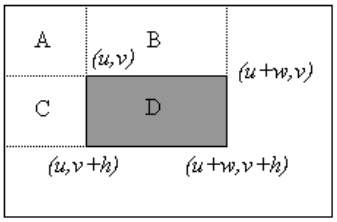
SIFT features [48]. Each type of feature represents a different kind of information and has a different computational cost. For example, Canny features represent low-level edge information with low computational cost, while SIFT features represent rotation and scale invariant information with high computational cost. When choosing image features, we always need to balance their representational capacities and computational costs.

In this dissertation I use Haar-like features proposed by Viola and Jones [86] for two reasons. First, Haar-like features can be computed very quickly, enabling efficient object detection and segmentation algorithms. Second, in an image region there are a large number of Haar-like features, each of which provides weak evidence of image contents. This well fits to the boosting framework. I will show how to use these features to build strong models in the later chapters.

The Haar-like features are similar to Haar basis functions. The four types of features defined by Viola and Jones are depicted in the Figure 2.7(A). These features are composed of rectangle regions, and they operate on grey level images. The feature values are defined as the difference between the sum over the dark regions and the sum over the light regions. In an image window tens of thousands of Haar-like features can be extracted by varying the scales and positions of each type of features, as shown in Figure




(A)




(B)

Figure 2.8: Computing a rectangular sum using an integral image. (A) The value of an integral image at the pixel index $(u, v)$ is computed as the sum of the pixels in the rectangle defined by the top-left corner of the image and $(u, v)$. (B) The rectangular sum $sum_{rect}(u, v, w, h) = II(u, v) + II(u + w, v + h) - II(u, v + h) - II(u + w, v) = A + A + B + C + D - (A + C) - (A + B) = D$. The computation only requires four array references of the integral image.

2.7(B). This feature set is overcomplete in the sense that the intrinsic dimension of the image pattern in the window is much less than the number of features.

The major benefit of using these rectangle features is that they can be computed efficiently using an intermediate image representation named the integral image, also known as the summed area table in graphics [15]. For an input image $I$, the value of its integral image $II$ at the pixel index $(u, v)$, as shown in Figure 2.8(A), is computed as the sum of the pixel values in the rectangle defined by the top-left corner of the image and $(u, v)$:

$$II(u, v) = \sum_{u' \leq u} \sum_{v' \leq v} I(u', v'). \tag{2.6}$$

The integral image $II$ can be computed efficiently by one pass scanning of the original image using the following recursive equation:

$$II(u, v) = II(u, v - 1) + II(u - 1, v) + I(u, v) - II(u - 1, v - 1). \tag{2.7}$$

With the help of the integral image, any rectangular sum can be computed in four array references:

$$sum_{rect}(u, v, w, h) = II(u, v) + II(u + w, v + h) - II(u, v + h) - II(u + w, v), \tag{2.8}$$

where $(u, v)$ is the top-left corner of the rectangle, $w$ is the width, and $h$ is the height. 2.8(B) illustrates the computation.

A Haar-like feature can be computed efficiently using rectangular sums. The computational cost is constant and independent of the feature scale. This is contrary to other features, whose computational costs are determined by the number of the covered pixels. The drawback of Haar-like features is that the cost of computing non-axis-aligned features is high. In the computation, instead of rotating Haar-like features, the input image is rotated, and a new integral image is computed. One way to reduce computational cost is to precompute a set of integral images with different orientations.

## 2.3 Deformable object segmentation

The purpose of image segmentation is to partition an image into several regions, each of which has a similar intensity, color, or texture property or contains objects of interest. Segmentation is widely used in computer vision and image analysis applications, such as measuring tumor or tissue sizes in medical imaging, locating objects in satellite images, and delineating the contours of faces for face recognition. During the last four decades, hundreds of segmentation algorithms have been proposed. Comprehensive reviews of these algorithms are given in [64, 57, 90].

In many real applications, segmentation only based on some simple rules, such as similar color or texture in a region, has poor performance due to confusions caused by the large shape and color variability of objects. Deformable shape segmentation uses prior knowledge of the the object's shape and appearance to resolve these confusions and to constrain the segmentation result to a plausible solution. This section focuses on reviewing such algorithms. There are two challenges in designing a deformable shape segmentation algorithm: (i) how to model deformable shape, and (ii) how to characterize the interaction between shape and appearance. I review different approaches in the

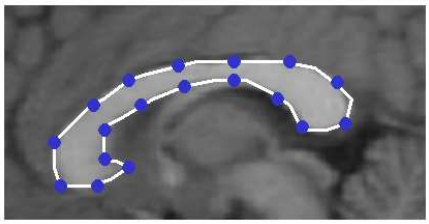
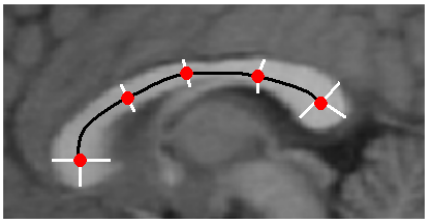

(A)

(B)

Figure 2.9: Representing the shape of the corpus callosum structure in a midsagittal MR image: (A) a boundary based representation, where a shape is depicted by a set of sampled point with connecting relation; and (B) a M-rep representation, which is specified by a set of medial atoms (red dots) and their associated spokes (while lines).

literature to address these two aspects.

### Deformable shape models

A straightforward way to describe an object shape in an image is to represent its boundary. Figure 2.9(A) shows a discretized boundary representation, where a shape is depicted by a set of connected points. Many algorithms use this representation due to its simplicity, such as the Active Contour Model (ACM) or snakes [38], the Active Shape Model (ASM) [12], and the Active Appearance Model (AAM) [11]. For such discretized representations, interpolation schemes are necessary to build smooth boundaries. In order to compute analytic expressions of differential qualities, such as normals and curvatures, on boundaries, a parametric form is often used, which describes boundaries using a set of local or global basis functions. Common parametric representations include B-splines [50], superquadrics [76], and spherical harmonics [75].

A shape can also be described by its occupied region [57]. A region based representation is capable of characterizing deformations happening not only at surfaces but also in the interior of objects. This capacity is valuable to many medical applications. Figure 2.9(B) shows an m-rep representation developed by Pizer et al [59], which is a typical region-based approach. An m-rep of an object consists of a set of medial atoms, each of which have two or three spokes to implicitly specify the object boundary. It has been shown that an m-rep is capable of describing relatively complicated object deformations,

including local translations, twisting, bending, and magnification.

Real applications are often required to deal with a large class of object shape variations. This leads to the idea of deformable shape models, which capture the essential characteristics of the shapes and have enough flexibility to describe the shape variations. Based on the discussion in [16], a good deformable shape model satisfies three criterions. First, it should be general, which means the model can generate any plausible shape variation. Second, it should be specific, which means the model can only generate valid shape variations. Third, it should be compact, which means the model can be described by a small number of parameters.

A deformable shape model is categorized into either a local model or a global model depending on the range of shape constraints introduced by the model. A local model imposes local constraints to shape deformation, such as the boundary smoothness constraint used in ACM [38] and the rigidity constraint in the viscous fluid based deformable template [8, 72]. Local constraints have been widely used, because they are easy to specify. The main weakness of local models is that they are not specific enough to guarantee valid shape structures; namely they might generate globally implausible shapes. A global model describes overall shape variations. Compared with local models, global models provide stronger constraints by controlling all plausible shape variations using a set of shape parameters. Global models are usually more specific and compact than local models. Typical algorithms based on global shape models include ASM [12], AAM [11], and segmentation based on m-reps [59].

Building a global model for a deformable shape is challenging because all plausible shape variations need be considered. A common way to construct a global shape model is to learn it from a set of training shapes [12, 11, 59]. For this kind of approach, the first step is to establish correspondence among the given shape population. The correspondence can be built by manually indicating salient points, or fiducial points, on the shapes, by using minimum description length [16], or by deforming a template

to each individual shape [66]. Then every shape is represented as a vector, and the shape population are considered as a point cloud in the vector space. These points usually reside within a high-dimensional manifold. Dimension reduction techniques, such as PCA [11, 59] or principal geodesic analysis (PGA) [23], are used to project the manifold to a low-dimensional model space. The shape populations are represented by the parameters defining the model space. Finally, the distribution of the shapes in the model space is estimated.

**Segmentation algorithms using deformable models**

Object segmentation can be considered to search in a model space for a shape that fits the target object in an image. In order to determine how well a hypothetical model matches to an image, we need to build a fitting function to characterize the relationship between the shape and image appearance. A good fitting function should satisfy two requirements. First, the function can well differentiate the correct solution from its background in the model space. Second, the function can effectively guide the search algorithms to the correct solution.

Fitting functions are often constructed as energy functions, which are, in turn, used in energy minimization approaches. A typical algorithm is the active contour model [38], which seeks a parameterized curve that minimizes the following energy function:

$$E = E_{\text{image}} + E_{\text{regularity}}, \tag{2.9}$$

where $E_{\text{image}}$ describes the fitness of the curve to the image appearance, and $E_{\text{regularity}}$ encodes local shape priors, including the elasticity and stiffness of the curve. The energy functions are usually set empirically by observing the properties of object boundary in images. Minimizing the energy function is formulated as a variational problem [14], which can be solved using Euler-Lagrange equations [38]. The parameterized curves used

the original ACM have difficulties in handling topological changes. Level set algorithms [54] represent a boundary implicitly as a level set of a higher-dimensional scaler function, allowing the topology of the boundary to change when minimizing the energy functions. Energy minimization approaches are widely used by many segmentation tools due to their long history and their simplicity in setting up energy functions. However, the fitting functions constructed in this way have difficulties satisfying the two requirements mentioned above. They cannot guarantee to produce the lowest energy at the ground truth position, and they are likely to have local minimums around strong image edges. Also, they use local shape models, which have too many parameters to be optimized.

Fitting functions can be learned from a given set of example images with ground truth annotations. It is common to learn both shape and appearance priors using generative models. Segmentation can be seen as minimizing the posterior probability defined by Bayes' rule:

$$\arg\max_C p(C|I) = \arg\max_C \frac{p(C)p(I|C)}{p(I)} = \arg\max_C p(C)p(I|C), \tag{2.10}$$

where $I$ is an input image and $C$ is a shape model. $p(C)$ is a shape prior learned using a generative model. $p(I|C)$ is a appearance likelihood describing the probability of the hypothesized model $C$ fitting to the image $I$. An example of such an approach is segmentation based on the m-rep [58], in which $C$ is a m-rep shape model, $p(C)$ is a Gaussian model computed using Principal Geodesic Analysis, and $p(I|C)$ is defined via PCA on a Discrete Regional Intensity Quantile Function(DRIQF). The function is minimized using conjugate gradient optimization. Generative models learn the relationship between the ground truth shapes and their appearance, and they provide useful information about the underlying generation process of the data population. However, they also suffer same problems faced by energy minimization approaches. A generative model is not trained explicitly to differentiate the ground truth shapes from their

background. Also generative learning has difficulties controlling the overall shape of the fitting function. The learned functions could have local extrema, causing difficulties for optimization algorithms.

The Active Shape Model (ASM) [12] and the Active Appearance Model (AAM) [11] are popular segmentation algorithms based on generative models. Both algorithms use the Point Distribution Model (PDM) as their shape models. ASM models appearance likelihood using Gaussian distributions of the local intensity profiles along the normals of object boundaries. In segmentation, the algorithm evolves object boundaries along their normal directions. AAM models the relation of a deformable shape and its appearance using PCA. To speed the minimization of the object function in segmentation, a 'difference decomposition' method is used to predict a search direction based on the current fitting error.

To improve the segmentation performance, discriminative learning has been recently incorporated into generative models and energy functions. In [95], boundary detectors were trained to replace the generative models in ASM for more accurately locating the boundary of heart chambers. In [81], a foreground/backgroud classifier was plugged into a energy function to provide the evidence of whether the current pixel belongs to the target object or not. The fitting functions built by these approaches improve the segmentation results. However, they could still have local extrema.

To avoid searching in high dimensions, a shape model can also be directly inferred from image appearance by searching for the most similar shape from a list of candidate shapes using a sample-based statistical model [31], a regression method [99], or a ranking method [96]. These methods need a large set of training examples for establishing the relationship between appearance and shape. Also, this kind of approach can only infer the non-rigid shape variation, and it relies on other algorithms to determine the global rigid transformation of the shape.

## 2.4 Summary

A wide range of literature related to discriminative learning, object detection, and deformable object segmentation was reviewed to serve as a background for the object detection and segmentation algorithms presented in the remainder of this dissertation. A main point revealed is that discriminative learning outperforms generative learning when there are sufficient training data. However the current object detection and segmentation techniques, especially the segmentation techniques, haven't fully exploited the capacities of discriminative learning. I now start to present a series of algorithms to further explore the potential of applying discriminative learning to object detection and segmentation.

# Chapter 3

# Joint Real-time Object detection and Pose Estimation Using a Probabilistic Boosting Network

Real-time object detection and pose estimation are important to many computer vision applications, ranging from face detection to segmentation of anatomical structures in medical images. One state-of-the-art object detector is described by Viola and Jones [86]. They train a binary classifier off-line that differentiates an object of interest from the background ,and then, online, they exhaustively slide a scanning window over the input image for object instances. Section 2.2.1 provides more details of this approach.

It is challenging, however, to build a real-time detector that incorporates accurate pose estimation using a detector like the one discussed in [86] (i.e., exhaustively scanning the space of all possible translations and poses). Consider detecting the left ventricle (LV) in a 3D echocardiogram, an ultrasound volume of the human heart (see Figure 3.1). Discovering the LV configuration is helpful for orienting the 3D volume. For example, from a known LV configuration, one can meet an emerging clinical need to automatically display canonical 2D slices. Because the LV can occur at an arbitrary location and orientation, one needs to search over six parameters [1] to fully align the LV

[1]Ideally, one also needs three additional parameter $(s_x, s_y, s_z)$ to compensate for the varying

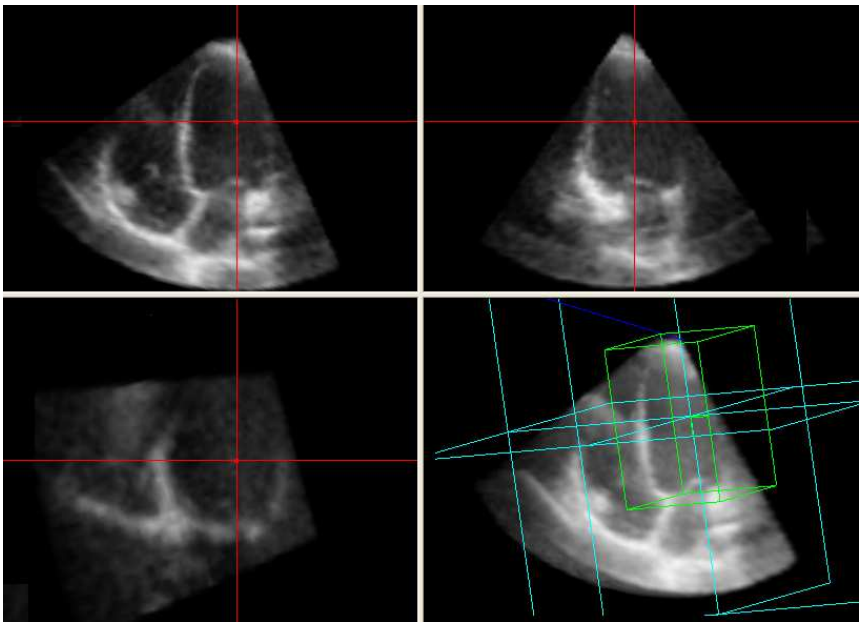


Figure 3.1: Detecting the LV and its pose in a 3D ultrasound volume is important to automatically navigate multiple canonical planes for clinical practices.

( $(t_x, t_y, t_z)$ for translation and $(\theta_x, \theta_y, \theta_z)$ for rotation). When extending the method of [86], the computational cost increases exponentially with the dimensionality of the parameter space. Furthermore, volume rotation and integral volume computations are time consuming because their computation is proportional to the number of voxels.

As reviewed in Section 2.2.1, a common way to avoid rotating images and volumes is to train a collection of binary classifiers to recognize different poses. A variety of structures are proposed to combine these classifiers, including the parallel structure [88] (Figure 2.6(A)), the pyramid structure [45] (Figure 2.6(B)), and the tree structure [37, 34] (Figure 2.6(C)). These approaches only test a sparse set of orientations and scales to meet the real-time requirement. However, anatomical structures in medical images can possess arbitrary orientations and scales, and the detection task needs to accurately determine them. Under such circumstances, it is challenging to build a rapid detector using the approaches above. In general, the speed is inversely proportional to the number of poses tested. In order to give an accurate estimate of the pose, speed

scale/size of patients.

must be sacrificed.

In this chapter I present a learning procedure called a Probabilistic Boosting Network (PBN) that is suitable for real-time object detection and pose estimation [94]. Based on the law of total probability, a PBN integrates evidence from two building blocks: a multiclass classifier for pose estimation and a detection cascade for object detection. Both the classifier and detection cascade employ boosting. By inferring the pose parameter, I avoid the exhaustive scan over pose parameters, which hampers the real-time requirement. In addition, I only use a single integral image/volume, avoiding the need for image/volume rotation. I implement PBN using a graph-structured network that alternates the two tasks of object detection and pose estimation in an effort to reject negative cases as quickly as possible. Compared with previous approaches, PBN has a higher accuracy in both object localization and pose estimation, with noticeably reduced computation.

In the remainder of this chapter I discuss the basic ideas of PBN (Section 3.1) and how to estimate object poses using multiclass classifiers (Section 3.2). In Section 3.3 I present the detailed graph structure of PNB. In Section 3.4 I demonstrate the performance of PBN on left ventricle detection from 3D volumes, left atrium detection from 2D images, and road sign detection. The chapter concludes in Section 3.5.

## 3.1 Joint object detection and pose estimation

A common way to detect an object in an image is to consider all the possible object configurations in the image. To avoid searching the whole configuration space, I break up the configuration space into two parts: For the first part, translation, I still use exhaustive search; and for the second part, rotation and scale, I directly estimate the parameters from the image's appearance. I refer to the parameters in the second part as *pose*. I couple exhaustive scanning with pose estimation. In this way, I successfully

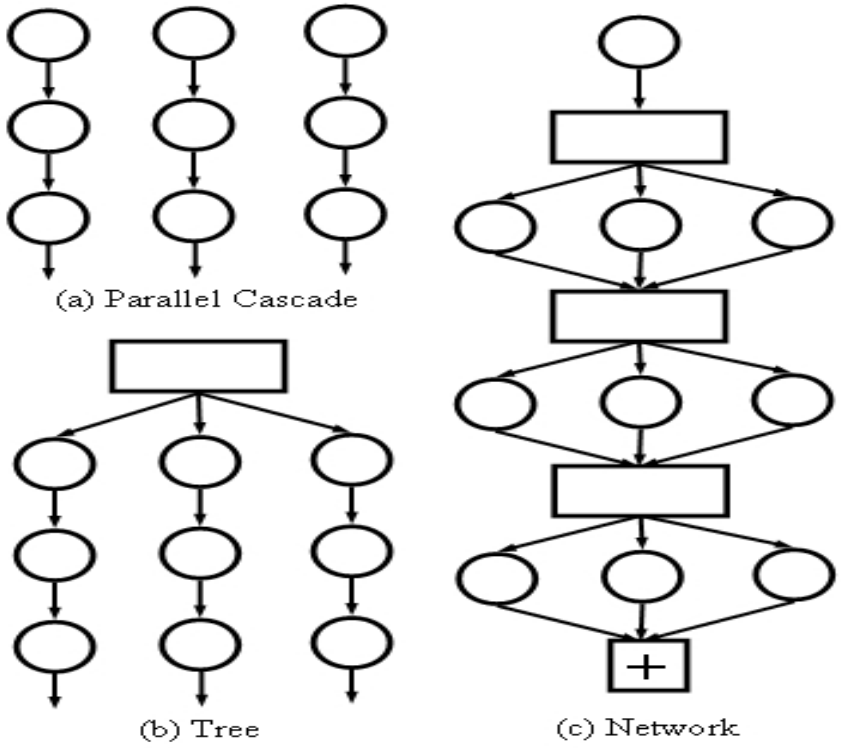



Figure 3.2: Different structures for multi-pose detection. The circle represents a binary foreground/background classifier. The rectangle represents a multiclass pose classifier.

eliminate the computational dependency proportionally to the number of poses.

My combined pose estimation and object detection framework employs a tree structure, where the root node is used to determine object poses, and the tree branches are used to verify the existence of objects with the determined poses. For the previous tree structure, usually only one branch is evaluated based on the decision of the multiclass classifier [37], and, as a result, the error made by the multiclass classifier has great influence on the detection result. In [34] several branches may be chosen, based on the hard decision provided by the VectorBoost algorithm, for the purpose of reducing the risk of possible errors in pose estimation. I handle the uncertainty of pose estimation by using probabilities as soft decisions. Let $I$ be an image patch that might contain an object of interest. The pose $\beta$ of the object is estimated via a multiclass approximation to the posterior probability $p(\beta|I)$. The probability is used to choose branches to evaluate. In addition, I obtain a better estimation of the pose by using the conditional mean of the probability.

For each branch, I train an object detector to compute $p(O|I, \beta)$, the probability of the object with pose $\beta$ given image $I$. I integrate the evidence from the pose classifier and object detectors using total probability. The final probability of being the object is computed as

$$p(O|I) = \int_{\beta} p(O|I, \beta)p(\beta|I)\, d\beta. \tag{3.1}$$

This integrating rule is an example of combining a generative mixture model with discriminative classification models [83]. In order to train the multiclass classifier and binary object detectors, I invoke boosting to perform feature selection based on a common feature pool. These weak classifiers rely on a single integral volume/image.

To further increase the efficiency of the tree structure, a graph-structured network (Figure 3.2(c)) is proposed to reject background cases as early as possible. The multiclass classifier is decomposed into several sub-classifiers by taking advantage of the additive nature of the boosted classifier. The sub-classifiers and binary detectors are

combined in a graph structure that alternates the two task of pose estimation and object detection. Furthermore, I include a binary classifier as a pre-filter of the graph to reject the background that can be easily differentiated from the foreground.

## 3.2 Pose Estimation

Given a window containing the object of interest, the goal is to estimate the pose parameter(s) of the object based on the image appearance $I$ in the window. I first discuss the algorithm for one-parameter estimation and then use the one-parameter estimation algorithm as a building block for multiple parameters.

### 3.2.1 One-Parameter Estimation

The object appearance variation is caused not only by the rigid transformation we want to estimate, but it is also influenced by noise, intensity variation, and nonrigid shape deformations (as well as other effects). As a result, a robust algorithm is needed to guarantee the accuracy of the pose estimation. I handle the uncertainty of the pose estimation by learning a probability of the parameter, $\beta$, subject to the given input, $I$, $p(\beta|I)$. This probability can be used to estimate $\beta$ accurately and to avoid errors in subsequent tasks.

In practice, $p(\beta|I)$ is approximated with discretized distribution $p(\beta_j|I)$, based on a discrete set of parameter values, $\{\beta_1, \beta_2, \ldots, \beta_J\}$. I implemented the image-based multiclass boosting algorithm proposed by Zhou *et al.* [98]. This algorithm is based on the multiclass version of the influential boosting algorithm proposed by Friedman *et al.* [29], the so-called LogitBoost algorithm, which fits an additive symmetric logistic model via the maximum-likelihood principle. This fitting proceeds iteratively selecting weak learners and combining them into a strong classifier. The output of the LogitBoost algorithm is a set of $J$ response functions $\{F_j(x);\ j = 1, \ldots, J\}$, where each $F_j(x)$ is a

linear combination of a subset of weak learners:

$$F_j^n(x) = \sum_{i=1}^{n} f_{j,i}(x), \tag{3.2}$$

where $f_{j,i}(x)$ is a weak learner, and $n$ is the number of weak learners. LogitBoost provides a natural way to calculate the posterior distribution of class label:

$$p_j^n(x) = \frac{\exp(F_j(x))}{\sum_{k=1}^{J} \exp(F_k(x))}. \tag{3.3}$$

Refer to [98] for more details.

One advantage of the LogitBoost algorithm is that the computed posterior probability asymptotically approximates the ground truth [29]. This means that a trade-off can be made between the approximation accuracy and the computational cost by adjusting the number of weaker learners in the response functions. This property is used to build the network structure to reject background cases more efficiently in the early stages.

I infer the parameter $\beta$ by using a conditional mean, which is a Minimum Mean Square Error (MMSE) estimator:

$$\hat{\beta}_{MMSE} = \int_{\beta} \beta p(\beta|I)\, d\beta \approx \sum_{j} \beta_j p(\beta_j|I). \tag{3.4}$$

This gives a better estimate than a Maximum A Posterior (MAP) estimate from a discrete set of possible values of $\beta$ because the MMSE estimate can interpolate between values in discrete set. The MAP estimator is defined as

$$\hat{\beta}_{MAP} = \max_{\beta} p(\beta|I) \approx \max_{\{\beta_1,\ldots,\beta_J\}} p(\beta_j|I). \tag{3.5}$$

Figure 3.3 compares the MMSE and MAP estimates for the rotation parameter in the second experiment (Section 3.4.2). I learned the multiclass classifier using the

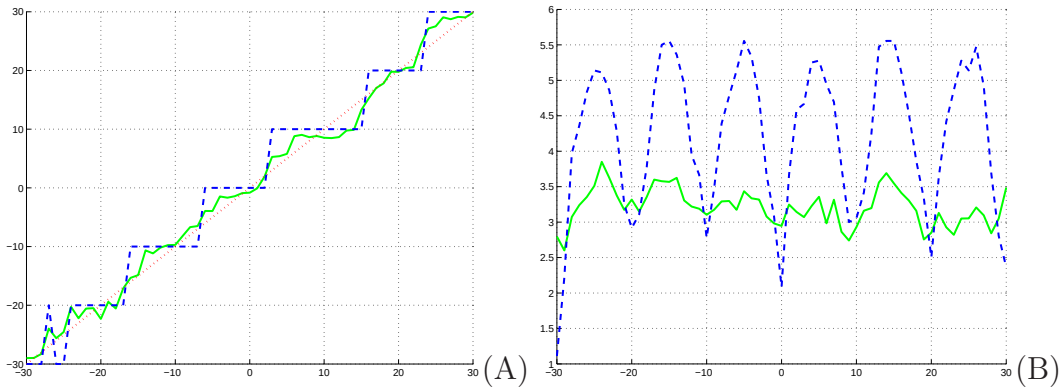



Figure 3.3: The estimation of the rotation parameter. (A) a single test image. (B) the MAE of all test data. The red line is the ground truth. The blue line is the MAP estimate. The green line is the MMSE estimate.

299 aligned training images rotated from -30 to 30 degrees in 10 degree steps. For comparison, I synthetically rotated 72 aligned test images from -30 to 30 degrees in 1-degree steps. Figure 3.3(A) displays the rotation estimation result of a test image. As the MMSE estimate can interpolate between angles, it gives a smoother curve. Figure 3.3(B) shows the Mean Absolute Error (MAE) in angle estimation of all test data. The MMSE estimate obviously outperforms the MAP estimate except when the angle is close to the quantized values.

### 3.2.2 Two-Parameter Estimation

The multiclass classifier that estimates one parameter in the previous section can be used as a building block to construct a graph structure for estimating two or more parameters. In this section I focus on two-parameter estimation, but the same principle can be applied to situations with more than two parameters. Suppose that the two unknown parameters are $\gamma$ and $\beta$, the goal is to estimate $p(\beta, \gamma|I)$ from the image $I$. Figure 3.4 gives a graphical illustration of three possible structures that can be used to model this estimation task.

For the type-$a$ structure, I treat the combination of $\beta$ and $\gamma$ as a single variable and train $p(\beta, \gamma|I)$ directly. This approach is structurally simple and has good performance

when the number of combined states is small. However, when the number of combined states is large, or the appearance variation caused by both parameters are too complex to learn using a single classifier, this approach performs poorly. In this case, a divide-and-conquer strategy is appropriate to estimate parameters sequentially by using multiple multiclass classifiers.

The type-$b$ structure assumes $\beta$ and $\gamma$ are independent. To train $p(\beta|I)$ (or $p(\gamma|I)$), I treat the variation in $\gamma$ (or $\beta$) as intra-class. The joint distribution is approximated as

$$p(\beta, \gamma|I) \approx p(\beta|I) * p(\gamma|I). \tag{3.6}$$

The drawback of this approach is the assumption of independence of $\beta$ and $\gamma$ is often invalid.

For the type-$c$ structure, I apply the exact conditional probability law:

$$p(\beta, \gamma|I) = p(\gamma|\beta, I) * p(\beta|I) \tag{3.7}$$

This can be represented as a tree structure. A root multiclass classifier is trained to learn $p(\beta|I)$ by treating the variation in $\gamma$ as intra-class. Each child node corresponds to the conditional probability $p(\gamma|\beta_j, I)$ for a discrete state $\beta_j$. To compute $p(\beta, \gamma|I)$ efficiently, I omit branches whose probability $p(\beta|I)$ is below a specified threshold.

The choice of the root parameter for the type-$c$ structure influences the overall performance, because the amount of the image appearance variation caused by the two parameters is not the same. Usually, the parameter that causes larger appearance variation should be the root node. This makes learning $p(\beta|I)$ easier, and leads to a better division of the pose space.

How to choose among these three types is determined by the data properties. An intuitive principle is that, if the number of the poses is small, and the appearance variation can be sufficiently captured by one classifier, I use the type-$a$ structure. If

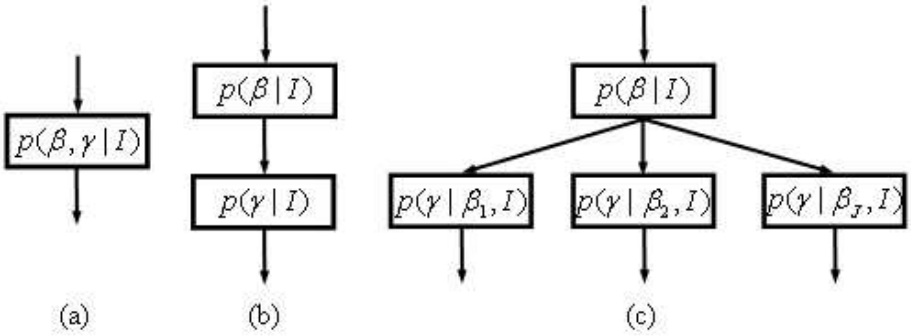



Figure 3.4: Different structures for estimating probability of two-parameter pose. In the type-$c$ structure, the parameter, $\beta$, is called the "root" parameter.

the parameters are justifiably independent, use the type-$b$ structure. Otherwise, use the type-$c$ structure.

## 3.3 Probabilistic boosting network (PBN)

In the previous section I discussed how to efficiently estimate the pose parameter using a scanning window. I now present PBN that integrates evidence from pose estimator and binary detectors. A PBN has three basic features:

1. It is *probabilistic.* A PBN leverages the fundamental total probability law to compute the probability of being object $O$. Assuming that the pose parameter $\beta$ is discretized into $\{\beta_1, \beta_2, \ldots, \beta_J\}$, we have

$$p(O|I) = \sum_{j=1}^{J} p(O|I, \beta_j) p(\beta_j|I), \tag{3.8}$$

   where $p(O|I, \beta_j)$ is the binary classifier specific to the parameter $\beta_j$. To compute (3.8) efficiently, I ignore branches whose pose probability is smaller than a pre-specified threshold $p_0$:

$$p(O|I) \approx \sum_{j:\ p(\beta_j|I) \geq p_0} p(O|I, \beta_j) p(\beta_j|I). \tag{3.9}$$

2. It uses *boosting.* As discussed in Section 3.2, the probability $p(\beta_j|I)$ is implemented using the multiclass LogintBoost algorithm [29]. The classifier $p(O|I,\beta_j)$ is implemented using the cascade of boosted binary classifiers [86], which are able to deal with numerous negative examples and eliminate them as early as possible during testing. To implement the binary classifier in the cascade, one can use AdaBoost [28], binary LogitBoost [29] or other variants. Suppose that the cascade has $S_j$ stages, then $p(O|I,\beta_j)$ is computed as

$$p(O|I,\beta_j) = \prod_{s=1}^{S_j} p_s(O|I,\beta_j), \tag{3.10}$$

where $p_s(O|I,\beta_j)$ is the binary classifier for the $s^{th}$ cascade. The complexity of the classifier $p_s(O|I,\beta_j)$ increases as the number of stages increases. Without loss of generality, assume that $S_1 = S_2 = ... = S_J = S$. If say $S_j < S$, I simply set $p_s(O|I,\beta_j) = 1$ for $s > S_j$.

3. It has a *network* structure. The total probability law:

$$p(O|I) = \sum_{j=1}^{J} \prod_{s=1}^{S} p_s(O|I,\beta_j) p(\beta_j|I), \tag{3.11}$$

can be implemented in a tree-structured network as shown in Figure 3.2(b). Using this structure, negatives are rejected quickly when they flow through the network while the positives traverse through several branches. By combining evidence from these branches, one is able to accurately estimate $p(O|I)$ using (3.11).

In [78] a learning procedure called a Probabilistic Boosting Tree (PBT) was presented. Both PBN and PBT are able to provide object detection probabilities using boosting, and both have a tree structure, but they also differ significantly. In the tree-structured PBN, each node corresponds to a specified parameter, while in PBT there is no specific parameter. PBN also estimates pose parameters explicitly. Finally, PBN

provides an efficient graph structure as shown next, which is not the case for PBT. In [7, 36] boosting is combined with a (dynamic) Bayesian network for parameter learning, structure selection, and audio-video speaker detection.

### 3.3.1 Efficient graph structure

The efficiency of the tree structure is due to two factors. First, instead of exploring all the branches in parallel, only a few promising branches, indicated by the multiclass classifier $p(\beta|I)$, are tested. There is an overhead for computing the probability $p(\beta|I)$. However, it is small compared with the cost of exploring all the branches. Second, the cascade structure rejects negatives quickly, which is important in the case where finding an object is a rare event. For common detection applications, there are usually only a few objects of interest present in the image. Everything other than the objects should be rejected as fast as possible.

The tree-structured PBN is not yet optimal in terms of computation because the overhead of computing the probability $p(\beta|I)$ is necessary for all background windows. These candidate windows are randomly sent to several branches of cascades and rejected by these branches. This creates a dilemma for the tree-structure: the purpose of a multiclass classifier is to select proper binary classifiers to reject background, but determining the pose of these background patches wastes computation.

One way to solve this problem is to discard as many background windows as possible via a focus-of-attention mechanism. This can be achieved by pooling together data from positives in different poses to train a pose-invariant classifier as a pre-filter. I tune the pose-invariant detector to have a 100% detection rate (though with a large number of false positives) by adjusting the threshold. This detector cannot offer a precise detection, but it is useful for rejecting a large percentage of the background candidates.

Even when the pre-filter classifier is used, there are still non-object windows passing the pre-filter, causing unnecessary overhead computations of $p(\beta|I)$. Following the idea

**To compute the probability** $p(O|I)$

1. Let $e_j$ be the number of the binary classifiers already evaluated in the $j^{th}$ branch. Start with $e_j = 0$, response functions $F_j(x) = 0$, and probabilities $p(O|I, \beta_j) = 1$, $j = 1, \ldots, J$.

2. Use the pose-invariant classifier to pre-filter. If $I$ is background, set $p(O|I) = 0$ and exit.

3. Repeat for $s = 1, 2, ..., S$:

   - Add $n_s$ weak learners to $F_j(x)$ and update $p(\beta_j|I)$.
   - For $j^{th}$ branch, $j = 1, \ldots, J$, if $p(\beta_j|I) \geq p_0$ and $p(O|I, \beta_j) > 0$:
     - Compute the probabilities $p_k(O|I, \beta_j)$, $k = e_j + 1, \ldots, s$. If it is background, set $p(O|I, \beta_j) = 0$.
     - Update:
     $$p(O|I, \beta_j) \leftarrow p(O|I, \beta_j) \prod_{k=e_j+1}^{s} p_k(O|I, \beta_j).$$
     - Update $e_j = s$.
   - If $p(O|I, \beta_j) = 0$ for all branches, set $p(O|I) = 0$ and exit.

4. Compute $p(O|I)$ based on the total probability law.

Figure 3.5: The PBN detection algorithm.

of the cascade structure for binary detector $p(O|I, \beta_j)$, which breaks its computation into several stages with increasing complexity, I also decompose the computation of $p(\beta|I)$ into several stages by taking the advantage of the additive model arising from the LogitBoost algorithm. The response functions at the $s^{th}$ stage is

$$F_j^s(x) = F_j^{s-1}(x) + \sum_{i=1}^{s_n} f_{j,i}(x), \tag{3.12}$$

where $s_n$ is the number of weak learners at the $s^{th}$ stage. For the type-$b$ and type-$c$ structures, the computation of the probability $p(\beta, \gamma|I)$ can also be decomposed by distributing the weak learners of the multiclass classifiers to several stages.

I organize the whole detector as a graph-structured network as shown in Figure 3.2(c), which alternates the two tasks of pose estimation and background rejection by hierar-

chically distributing the overhead of computing $p(\beta|I)$. Figure 3.5 shows the detection algorithm of PBN with a single pose parameter. PBN detection with multiple pose parameters can be implemented in a similar way. The network is evaluated top-to-bottom. More weak learners are added to update $p(\beta|I)$ at each new stage, which approximates the true posterior probability more accurately due to its asymptotic convergence property. Based on the newly estimated $p(\beta|I)$, the binary classifiers corresponding to large $p(\beta|I)$ are evaluated at this stage. If a new binary branch unexplored in earlier stages is selected, I trace back to the beginning and re-evaluate the whole branch. If the candidate fails all selected binary classifiers, it is considered as a background window and the computation stops; otherwise, it proceeds to the next stage.

More accurate pose estimation helps to determine whether the candidate window belongs to the foreground, while the binary classifiers help to determine if it is necessary to continue evaluating $p(\beta|I)$. This way, the positives are evaluated at a minimum number of branches and the negatives are quickly rejected by either the pre-filter or early detector cascades.

## 3.4 Experimental results

I applied the PBN classifier in three experiments. In the first experiment I detected the LV in a 3D echocardiogram and estimated the LV orientation using a one-parameter estimation. In the second I detected the left atrium (LA) in a 2D echoechocardiogram and estimated two pose parameters. And in the third I detected road signs from an unconstrained scene.

### 3.4.1 Left ventricle detection in 3D echocardiogram

I used 50 3D echocardiographic volumes for training and 22 volumes for testing. The LV configurations for all 72 volumes were annotated by experts. As a preprocessing step,

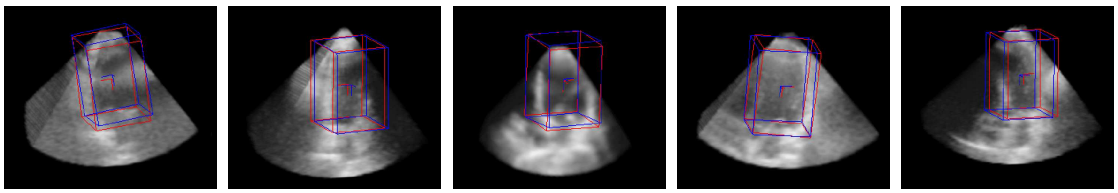


Figure 3.6: 3D LV detection results obtained by the PBN classifier. The blue box is the PBN estimation, and the red box is the ground truth.

each volume was down-sampled into a quarter size of its original size, giving a volume of $54 \times 62 \times 67$ voxels.

I built a local coordinate system for each volume, with origin at the LV center and $z$-axis along the long axis of the LV. Because the LV is roughly symmetrical along its long axis, I treated the variation resulting from the rotation along the $z$-axis as intra-class. I concentrated on detecting the object position $(t_x, t_y, t_z)$ and estimating two rotation parameters $(\theta_x, \theta_y)$.

The rotation along the $x$-axis was divided into three angles $\{-20, 0, 20\}$, and the rotation along the $y$-axis was also divided into three angles $\{-20, 0, 20\}$. Because the number of discrete poses is small, one multiclass classifier was used to estimate $p(\theta_x, \theta_y|I)$, which corresponds to the type-$a$ structure discussed in Section 3.2.2. I randomly perturbed each angle by a small amount (less than 3 voxels) around the LV center to collect more positives. The images caused by pure rotation along the $z$-axis were added as positive data in order to increase the robustness of the LV detection.

I implemented the following seven approaches: 1) a pose-invariant classifier. This is the same as using only the pre-filter but without adjusting the threshold; 2) a parallel-structured detector which makes decision based on the magnitude of response functions trained independently; 3) a tree-1 detector, which explored only one branch with the largest pose probability; 4) a tree-structured PBN detector exploring several of the most promising branches but without the pre-filter; 5) a graph-structured PBN detector without the pre-filter; 6) a tree-structured PBN detector with the pre-filter; and 7) a graph-structured PBN detector with the pre-filter. I constructed a common feature pool,

based on the 3D local rectangle features [80]. I selected, via boosting, relevant features from this pool to form the multiclass and binary classifiers. Because the ultrasound volume has a fan-like support region, it is likely that the ground truth LV window contains voxels outside the fan. To handle this, I used a so-called mask volume [31] which indicates one if a voxel is inside the fan and zero otherwise. To efficiently count the number of within-the-fan voxels in a specific window, I computed the integral volume of the mask volume. I empirically found that using the additional mask treatment introduces an extra 30% of computation when compared with [86]. Also, the area outside the known fan is not scanned. I used the same discretization of the pose parameter for all seven approaches, in order to fairly compare methods.

The test results are presented in Table 3.1. Denoting the estimated parameter by $(\hat{t}_x, \hat{t}_y, \hat{t}_x, \hat{\theta}_x, \hat{\theta}_y)$ and the ground truth by $(t_x, t_y, t_z, \theta_x, \theta_y)$, I calculated the error in translation as $\sqrt{\sum_{a=x,y,z}(\hat{t}_a - t_a)^2}$ and the errors in rotation as $|\hat{\theta}_x - \theta_x|$ and $|\hat{\theta}_y - \theta_y|$. Reported in Table 3.1 are the median values of the test errors. The computational speed was measured on a PC with a 2.4GHz CPU and 2GB memory.

From the results, observe that the pose-invariant detector is the fastest, but it lacks detection accuracy due to the appearance variation introduced by rotation. Also, it does not estimate the orientation. The parallel and tree-1 structures are comparable in terms of speed and translation, but the tree-1 structure has an edge over the parallel one in terms of orientation accuracy. The reason why the parallel structure, which evaluates all branches, is slightly faster than the tree-1 structure, which evaluates only one branch, is due to the overhead of computing pose estimation, the small number of branches, and the relative ease of rejecting background patches. However, both of their pose estimations are rough because of their inability to interpolate. The tree-structured PBN gives better accuracy, benefiting from the pose-probability computation, which enables proper branch selection, and the MMSE estimation of pose parameter, but it is slower than the parallel and tree-1 structures due to the previously mentioned reasons that lead

to more branches being evaluated. The graph-structured PBN achieves similar accuracy as the tree-structured PBN but significantly improves the speed as it distributes the overhead computation of pose estimation hierarchically. The pose-invariant pre-filter effectively rejects background candidates at early stages with no sacrifice of the final estimation. When it is coupled with the tree- and graph-structured PBNs, I achieved the real-time performance for 3D LV detection: processing about 10 volumes per second. To the best of my knowledge, this is the first real-time algorithm to detect an object in a 3D volume and estimate its pose. Figure 3.6 shows several detection and pose estimation results obtained by the PBN classifier.

| | time(ms) | T(voxel) | $\theta_x$ (degree) | $\theta_y$ (degree) |
|---|---|---|---|---|
| pose inv. | 83 | 3.593 | n/a | n/a |
| parallel | 223 | 2.549 | 9.109 | 6.976 |
| tree-1 [37] | 251 | 2.828 | 6.683 | 5.746 |
| tree | 295 | 2.236 | 5.481 | 3.967 |
| graph | 170 | 2.314 | 5.592 | 4.40 |
| tree+pre | 142 | 2.236 | 5.481 | 3.967 |
| graph+pre | 102 | 2.314 | 5.592 | 4.408 |

Table 3.1: Parameter estimation results of 3D LV detection.

### 3.4.2 Left atrium detection in 2D echocardiogram

I used 2D echoechocardiogram of Apical Two Chamber (A2C) view in the experiment. The A2C view captures only the left ventricle and left atrium (LA). LA detection in 2D echoechocardiogram is more difficult than LV detection [31] because the LA possesses more shape variation, and the LA, being in the far field of the ultrasound probe, is more difficult to image. Refer to Figure 3.8 for some A2C example images.

The data set contains 371 2D echoechocardiogram images with expert annotations of the LA, of which 299 randomly selected images were used for training and the remaining 72 for testing. I followed the procedure of Procrustes shape alignment in training. The rigid transformation between an original contour $\mathbf{x}$ and the mean contour $\mathbf{x_0}$ is defined as

$$\mathbf{x} = SRA\mathbf{x_0} + T, \tag{3.13}$$

where $S = \text{diag}[s, s]$ is a uniform scaling matrix, $R$ is a rotation matrix defined by rotation angle $\theta$, $A = \text{diag}[1, \alpha]$ defines the aspect ratio of the contour, and $T = [t_x, t_y]^T$ is a translation vector.

Among these five parameters $(s, \theta, \alpha, t_x, t_y)$, I treated the rotation $\theta$ and aspect ratio $\alpha$ as pose parameters. Based on range and precision requirements, I divided the rotation into seven angles $\{-30°, -20°, -10°, 0°, 10°, 20°, 30°\}$ and the aspect ratio into seven discrete values $\{-.76, -.84, -.92, 1, 1.08, 1.16, 1.24\}$. Because it is difficult to fit 49 classes into the memory to learn $p(\theta, \alpha|I)$ with one multiclass classifier, I used the type-$c$ structure discussed in Section 3.2.2. The root node computes $p(\theta|I)$ because the rotation causes more distinct variation. The training data were obtained by synthetically transforming aligned data with different rotation and aspect ratio parameters.

The other three parameters are searched exhaustively. A total of 49 binary detector cascades are trained for all poses. In order to allow objects with an in-between pose to pass these detectors, I augmented the positive training data with small pose perturbations.

I first compared the performance of the type-$b$ and type-$c$ structures for the two-parameter estimation. For each test image, I synthesized 49 examples with the discrete rotation and aspect ratio parameters. The probability $p(\theta, \alpha|I)$ is computed for the type-$b$ and type-$c$ structures, respectively. The estimate of the rotation parameter $\theta$ is the same for the two structures; the MAE is 3.4 degree. I compared the error of the aspect ratio parameter $\alpha$, as shown in Figure 3.7: The overall MAE error is 0.057 for type-$b$ and 0.050 for type-$c$, a 14% improvement over the type-$b$. The reason is that the variation caused by the rotation, which is quite large, interacts with the aspect ratio variation, making the independent assumption of the type-$b$ structure invalid.

I then compared the detection and pose estimation results of the seven approaches. Again, I used the mask image treatment. I implemented the exhaustive scanning algo-

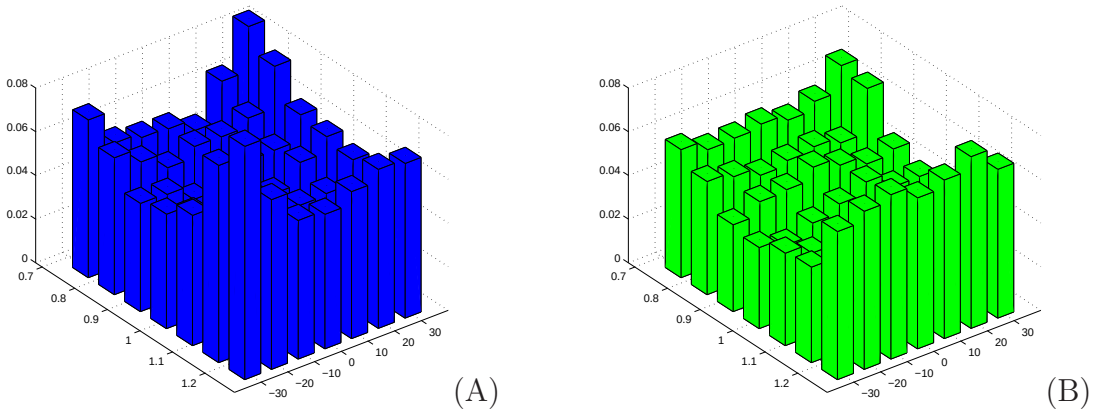



Figure 3.7: The mean absolute error of the aspect ratio in the two-parameter setting: (A) Type-$b$ structure and (B) Type-$c$ structure.

| | time(ms) | $T$(pix) | $\theta$(degree) | $s$ | $\alpha$ |
|---|---|---|---|---|---|
| exhaustive [31] | 619.5 | 5.018 | 7.082 | 0.086 | 0.099 |
| parallel | 339.3 | 5.849 | 7.351 | 0.096 | 0.104 |
| tree-1 | 145.2 | 7.925 | 7.225 | 0.158 | 0.147 |
| tree | 207.9 | 4.695 | 5.482 | 0.083 | 0.082 |
| graph | 141.3 | 4.695 | 5.482 | 0.083 | 0.082 |
| tree + pre | 104.4 | 5.027 | 5.767 | 0.083 | 0.083 |
| graph + pre | 76.1 | 5.027 | 5.767 | 0.083 | 0.083 |

Table 3.2: Parameter estimation results of 2D LA detection.

rithm in [31] for the LA[2], which trains one detector but rotates the image in testing, to replace the pose-invariant detector that is unable to estimate the pose. All the other six methods are the same. The detection is on half-sized images, which vary around the size $300 \times 270$ pixels. The test results are presented in Table 3.2. The error in translation is measured by the Euclidean distance, while the errors for other parameters are measured by the absolute value. Because there are a few outlier cases, the median error is reported. In each test image I selected the scanning candidate with the largest overall detection probability as the detection result and determined the pose using the MMSE estimate from PBN.

From Table 3.2, observe that the exhaustive approach is the slowest due to computation of integral images for all possible rotations and exhaustive search in the whole parameter space. The parallel structure saves the extra computation on integral images.

[2]Only the LV is originally addressed in [31].

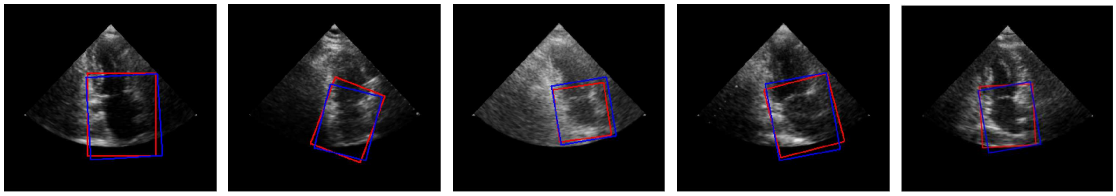


Figure 3.8: 2D LA detection results obtained by the PBN classifier. The blue box is the PBN estimation, and the red box is the ground truth.

However, comparing response functions of independently trained classifiers is problematic, and hence the accuracy is worse than the exhaustive approach. Unlike the 3D LV detection, the tree-1 structure [37], which evaluates only one branch even with the overhead computation of pose estimation, is faster than the parallel structure, which evaluates 49 binary detectors. But, it performs worse than the parallel structure, which is consistent to the observation made in [37]. The tree-structured PBN is slower than the tree-1 structure, but it significantly improves the accuracy: On the average, the error in translation is reduced by 3.23 pixels (40.8% improvement), the error in rotation by 1.743 degree (24.1% improvement), the error in scale by 0.075 (47.5% improvement), and the error in aspect ratio by 0.065 (44.2% improvement). The graph-structured PBN not only matches the same performance improvements induced by the tree-structured PBN over the tree-1 structure, but it also matches the speed of the tree-1 structure. Using the pose-invariant classifier as a pre-filter of PBN, which rejects about 65% of the candidates, the computation is effectively saved with a negligible performance degradation. The most efficient graph-structured PBN coupled with the pre-filter is able to process about 13 frames per second, more than eight times faster than the exhaustive approach [31].

### 3.4.3 Road sign detection

The proposed PBN is a generic learning procedure for object detection. I now present a preliminary experiment on detecting road signs in unconstrained scenes.

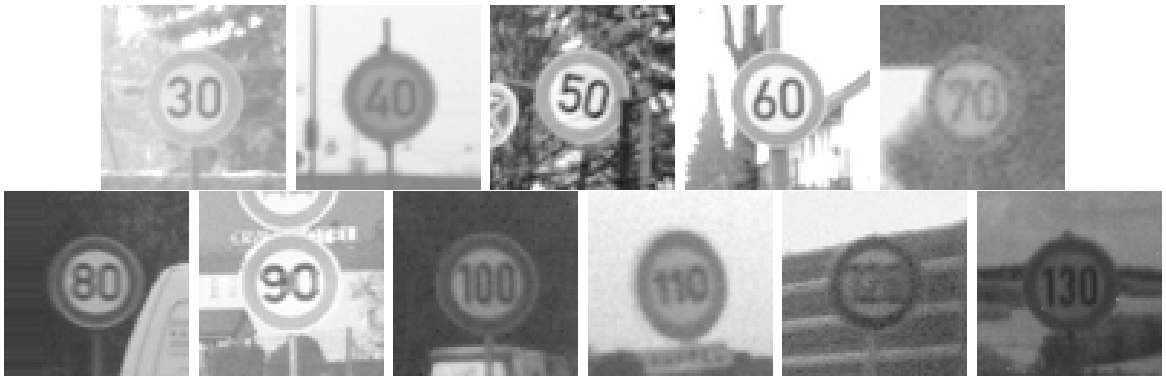



Figure 3.9: The positive image examples of road signs.

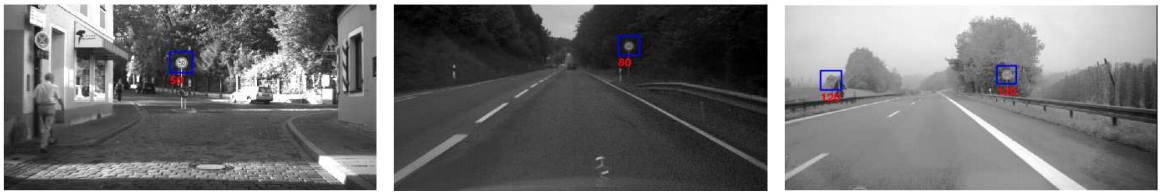


Figure 3.10: The road sign detection results obtained by the PBN classifier.

I trained the graph-structured PBN, along with the pre-filter, for 11 types of speed-limit road signs ranging from 30 to 130 kilometers per hour. I used one multiclass classifier to classify different road signs and 11 binary detector cascades to discriminate road signs from the background. Compared with the previous two experiments, the only difference is that the type of a road sign is determined by MAP estimation instead of MMSE estimation.

The amount of training data for each sign varies from 224 to 1227 images. I randomly cropped out negatives from images that contain no road signs. Figure 3.9 displays some of the training images of size $64 \times 64$ pixels, and Figure 3.10 shows the detection results on test images of size $400 \times 725$ pixels. The test images are captured in the city, under inclement weather, etc. The average running time is about 110ms.

## 3.5 Summary

In this chapter I have presented a learning algorithm called Probabilistic Boosting Network (PBN) for joint fast object detection and pose estimation. I have made the following contributions:

1. I have proposed a multiclass approximation for pose estimation and derived an accurate estimate for multiple continuous pose parameters. The multiclass classifier is trained via the LogitBoost algorithm that embeds feature selection and allows fast evaluation.

2. I have based the process of evidence integration on the total probability law. The posterior probability of being an object is analytically computed with no ad-hoc treatment.

3. I have implemented two network structures (tree and graph) to estimate pose and discriminate the foreground from the background efficiently. Coupled with a pose-invariant pre-filter, these structures enable fast rejection of background patches and wrong poses using only one integral volume/image.

4. I have invoked PBN to detect the left ventricle from a 3D ultrasound volume, processing about 10 volumes per second, and the left atrium and road signs from 2D images in real time.

# Chapter 4

# Deformable Shape Segmentation via Conditional Density Regression

Deformable shape segmentation is important to many computer vision and medical imaging applications. Three example applications are presented in Chapter 1. As discussed in Section 2.3, an object's shape in an image $I$ can be represented by a set of continuous model parameters, $C$, that define the shape and position of the object. Shape segmentation can be considered as optimizing a conditional probability density $p(C|I)$, which represents the probability of the model parameters describing the target object in the image. In any attempt to solve the above problem, two questions must be answered: (i) how to construct the density $p(C|I)$, and (ii) how to find the optimal solution for the given $p(C|I)$.

In general, finding a model that maximizes the probability is a difficult optimization problem because of the complexity of image appearance and high dimensionality of the model parameters. Usually, there can be an initial approximation to the correct solution that is provided by some prior knowledge or via some form of rigid object detection (e.g., PBN proposed in the previous chapter). Optimization techniques can then be used to refine the segmentation results. A variety of algorithms are proposed for this purpose, such as the Active Contour Model (ACM) [38], the Active Shape Model (ASM) [12], the Active Appearance Model (AAM) [11], and m-rep segmentation [59].

It is natural to use general-purpose optimization techniques such as gradient descent or the simplex method to find an optimal solution. In order to guarantee these kinds of algorithms converge to the optimal solution, $p(C|I)$ should be smooth and have only one global maximum, which is the correct solution. As reviewed in Section 2.3, previous approaches have difficulty satisfying this requirement.

In this chapter I propose a regression approach to learn the density $p(C|I)$ from the given training data [92]. The learned density $p(C|I)$ possesses a desired unimodal and smooth shape, which can be used by optimization algorithms to efficiently estimate a solution. To begin, I discuss the properties of the density $P(C|I)$ learned by the previous approaches and then motivate the regression approach.

## 4.1 Conditional density learning via regression

A common way to construct $p(C|I)$ is to learn this probability density from training data. The density $p(C|I)$ can be constructed either through generative approaches or discriminative approaches.

A generative approach learns a model $p(I|C)$ from the ground truth examples and calculates $p(C|I)$ using Bayes' rule, as shown in Equation 2.10. Although the learned model is sufficient to represent the ground truth examples, it cannot guarantee the convergence of the local optimization algorithms at the segmentation stage. When the model $C$ has only one parameter, a typical shape of the learned $p(C|I)$ is shown in Figure 4.1(A). The global maximum is usually near the ground truth. However, it is not smooth and could have several local maximums. The cost functions used in energy minimization techniques, such as ACM [38], also suffer from the same problem. Therefore the initial solution has to be close enough to the ground truth to make the local optimization algorithm effective.

One example of discriminative approaches is the classification method, which directly

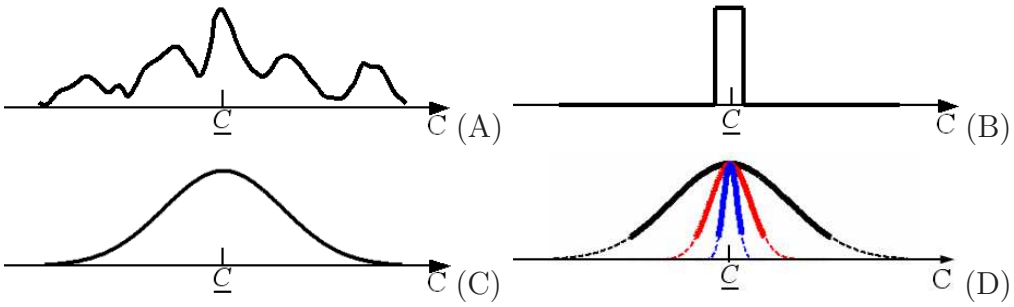



Figure 4.1: The learned $p(C|I)$ when $C$ is one dimensional: (A) a generative approach, (B) a classification approach, (C) the regression approach, and (D) the multilevel regression, the functions defined on the focus regions are drawn using thick lines. The ground truth of the model is $\underline{C}$.

models $p(C|I)$ by pooling ground truth examples as positives and the background examples as negatives. The learning focus is to build a model that distinguishes between foreground and background. Such classification methods are usually used in object detection applications. The learned $p(C|I)$ is like a boxcar function around the ground truth, e.g., a 1D example as shown in Figure 4.1(B). This model does not provide useful gradient information for local optimization. As a result, the solution is estimated by the exhaustive search, which is computationally prohibitive when the dimensionality of the model $C$ is high.

I construct $p(C|I)$ with a desired shape that guarantees the proper convergence of general-purpose local optimization techniques. I learn $p(C|I)$ from annotated training data by formulating a regression task. The learned density $p(C|I)$ is constrained to possess the desired unimodal and smooth shape (such as the bell-shaped curve of a normal density) in the model space, which can be used by local optimization algorithms to efficiently estimate the correct solution. Figure 4.1(C) shows the ideal shape of the 1D $p(C|I)$ learned in this way.

I employ boosting to learn a regressor by selecting relative features to form an additive committee of weak learners. Each weak learner, based on a Haar-like feature that can be computed rapidly, provides a rough fitness measurement of the object to the image's appearance. The learned regressor computes a robust measurement of fitness

by integrating the measurements of selected weak learners.

In most non-rigid segmentation applications, the object model $C$ has many parameters, including both pose and shape parameters. It is challenging to learn a function $p(C|I)$ via regression that approximates the target density sufficiently well across the whole parameter space because of insufficient sampling in the high-dimensional model space. To address this 'curse of dimensionality', the number of training examples should be an exponential function of the model's dimensionality to guarantee training accuracy. Furthermore, appearance variation and noisy imaging artifacts make the problem of insufficient sampling worse by introducing complexity to the regression input.

I make two contributions to tackle these learning challenges. First, I propose a multilevel approach that learns a series of conditional densities, each of which is defined on a focus region of the domain instead of the whole parameter space. By such a design, the regressors focus more and more on the region close to the ground truth. A one-dimensional multilevel example is shown in Figure 4.1(D). At the segmentation stage I perform a series of local optimizations based on the corresponding learned regressors to refine the segmentation result.

Second, when learning an individual regressor I propose a sampling algorithm for more effective training. The algorithm samples the most important regions in the model space for regression by leveraging the gradient information of the target probability function, which is essential for guiding the local optimization algorithms to find the correct solution.

In the remainder of the chapter I discuss the shape model used for segmentation in Section 4.2, the boosting-based regression in Section 4.3, and regression in high-dimensional model space in Section 4.4. In Section 4.5 I demonstrate the accuracy and robustness of the regression approach on corpus callosum border segmentation and facial feature localization. The chapter concludes in Section 4.6.

## 4.2 Shape model

I represent an object in an image $I$ by a set of continuous model parameters $C = (c_1, \ldots, c_D)$, where $D$ is the dimensionality of $C$. In different applications, $C$ can contain parameters for rigid transformation, or parameters for non-rigid shape deformation, or both. In this dissertation I use the model $C = (t_x, t_y, \theta, s, b_1, b_2, \ldots)$, where $(t_x, t_y, \theta, s)$ are rigid transformation parameters (2D translation, rotation, and scale), and $(b_1, b_2, \ldots)$ are parameters for a Point Distribution Model (PDM). In this section I start with a brief review of PDMs and then discuss the image appearance associated with the model.

### 4.2.1 Point Distribution Models

As discussed in Section 2.3, a variety of deformable shape models have been proposed to describe shape variation. I choose the PDM model because of its simplicity. For the experimental data used in this dissertation, the expert annotations are landmarks on objects of interest, making the application of PDM straightforward. Other deformable shape models can also be used. The regression approach focuses on modeling the relationship between a shape model and its image appearance. It is not dependent on a specific model.

PDM represents a shape with a set of $M$ control points specified by their image coordinate $\{u_m, v_m\}_{m=1,\ldots L}$. A shape vector $S = (u_1, v_1, u_2, v_2, \ldots u_M, v_M)^T$ is formed by concatenating the coordinates of the points. PDM learns the statistical variability of $S$ from a shape population. The training procedure is as follows:

1. Annotate the target objects in training images using a set of corresponding control points. The correspondence can be established either manually, by annotating the common fiducial points on objects, or automatically, by arranging points on object boundaries such that they minimize certain criterion (e.g., minimum description length [16]). In all the experiments I assume that the correspondences among

control points are available as inputs.

2. Align the annotated shapes to a common coordinate frame to remove the variation caused by rigid transformation, making the analysis focus on non-rigid deformation of the shapes. I use generalized Procrustes analysis [12] for this purpose.

3. Apply PCA to the aligned shapes to build a compact shape space. The aligned shapes are shape vectors corresponding to a set of points in the $2M$-dimensional space. These points reside on a high-dimensional manifold, and they can be projected into a low-dimensional shape space using PCA. The shape space is compact, and it is spanned by a reduced set of eigenvectors $(\phi_1, \phi_2, \ldots, \phi_{D_s})$ associated with the largest eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_{D_s})$. The dimensionality $D_s$ is chosen to explain a given fraction (e.g., 90%) of variation exhibited in the training data.

After training a PDM, a shape $S$ can be generated as

$$S = \overline{S} + \sum_{i=1}^{D_s} \phi_i b_i, \tag{4.1}$$

where $\overline{S}$ is the mean shape and $(b_1, b_2, \ldots, b_{D_s})$ are shape parameters controlling the amount of shape variation along directions defined by the eigenvectors. Each eigenvector is called as a shape mode of PDM. In order to ensure that the generated shape is a plausible shape (i.e., similar to those in the training set), the parameters $b_i$ are limited to be $\pm 3\sqrt{\lambda}$ [10]. Figure 4.2 shows the corpus callosum annotations in images and the effect of varying the shape parameters of the learned PDM. Observe that the learned PDMs capture the shape variation well. One advantage of PDM is that control points do not necessarily reside on the object boundary. This enables feature localization applications, such as facial feature localization, to be solved using the same framework as boundary segmentation. Figure 4.3 shows the facial feature annotations and the effect of varying the shape parameters of the learned PDM.

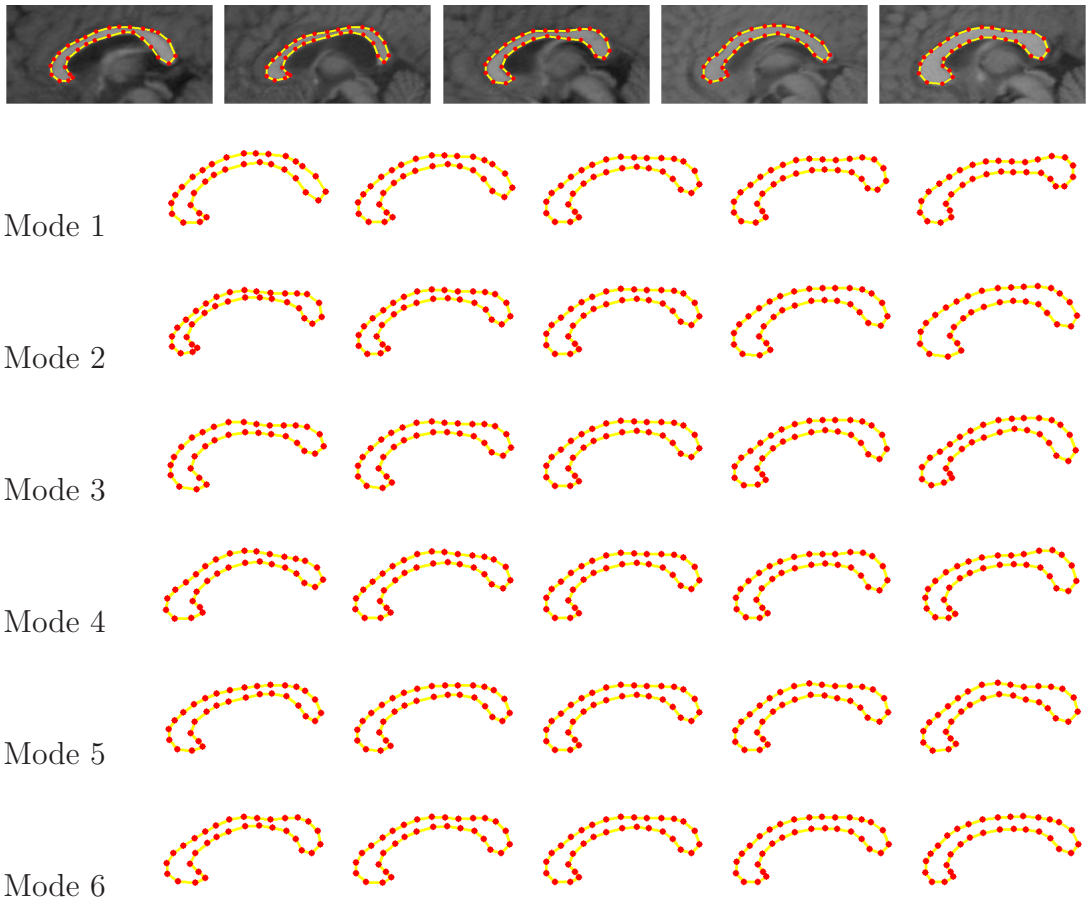



Figure 4.2: Corpus callosum annotations in images and the effect of varying the shape parameters of the learned PDM. The first row shows corpus callosum contours represented by 32 control points. The remaining rows show the effect of varying each of the first six PDM parameters between $\pm 3\sqrt{\lambda}$.

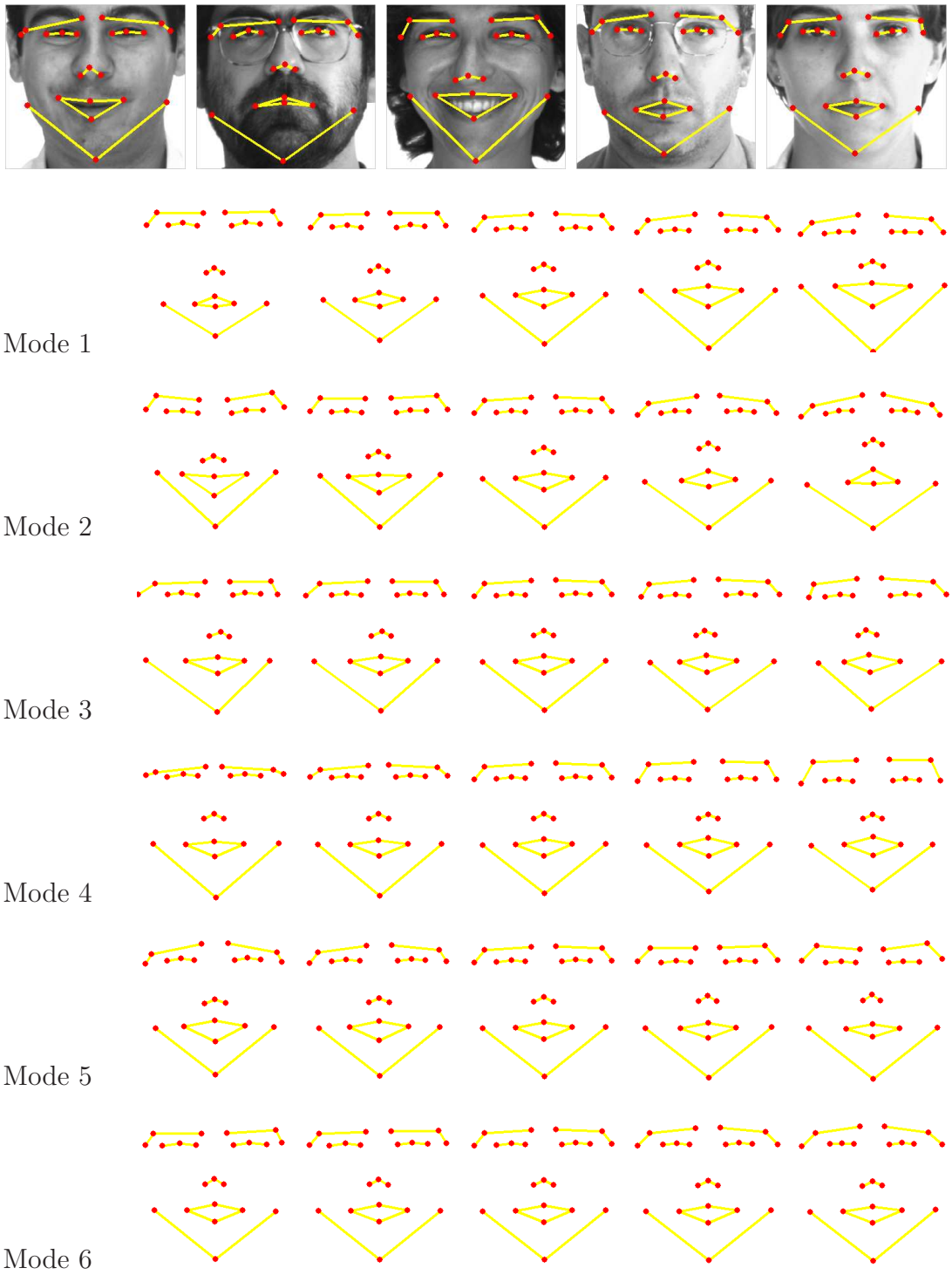



Figure 4.3: Facial feature annotations in images and the effect of varying the shape parameters of the learned PDM. The first row shows facial features indicated by 17 control points. The remaining rows show the effect of varying each of the first six PDM parameters between $\pm 3\sqrt{\lambda}$.

### 4.2.2 Model parameter normalization

When both rigid transformation parameters and shape parameters are included in a model $C$, these parameters usually have different units, e.g., translations versus rotations or translations versus shape parameters. A unit change in a specific parameter may cause varying amounts of shape variation in image space. To facilitate constructing focus regions presented in Section 4.4.2, I normalize parameters to exhibit roughly the same amount of shape variation as follows:

$$c_d = \bar{c}_d / \bar{r}_{1,d} * r_{1,d} \tag{4.2}$$

where $\bar{c}_i$ is an original model variable, $c_i$ is the variable after normalization, $\bar{r}_{1,d}$ is the initial error range of $\bar{c}_i$, and $r_{1,d}$ is the variation in the image domain caused by $\bar{r}_{1,d}$. The average Euclidean distance between corresponding control points is used in the experiments to indicate the variation in the image domain.

### 4.2.3 The feature image associated with a model

For a hypothesis model $C$ in a given image $I$, I extract a feature image $x(I, C)$ to represent the image appearance associated with the model $C$. For conciseness, I use $x$ instead of $x(I, C)$ when there is no confusion in the given context. In this section I address how to obtain $x$ efficiently.

There are two approaches for obtaining $x$, depending on the sampling range. The local approach samples information around the neighborhood of the current control points of the object in the image. For example, ASM [12] samples intensities along profiles normal to the boundary. The global approach samples the whole image in the area defined by the current shape, such as in AAM [11].

Generally speaking, the information from local sampling is good for measuring the fitness of the local regions, while the information from global sampling is useful for

measuring the fitness of the whole object. How to best use these two kinds of information effectively is application-specific. In previous approaches, heuristic rules were used to make decisions [12, 11], such as whether there are strong edges at boundaries, or whether the object has an overall unique appearance. It is not obvious how to combine such global and local information.

I let the algorithm decide which information is most useful by selecting the weak learners that most decrease the regression error. Therefore, an extracted feature image $x$ should contain both local and global information. I use a set of image patches associated with the current shape $C$ to represent $x$. Suppose that each shape is represented by $M$ control points; $M + 1$ subimages are extracted from the image as shown in Figure 4.4. Each subimage has its position, orientation and scale. The first subimage, indicated by the red box in Figure 4.4, contains the whole object. Its configuration is determined by the object pose. This image patch contains global information to indicate the fitness of the pose parameters. The remaining subimages, indicated by the green boxes in Figure 4.4, correspond to the control points, where the configurations are determined by the position, local orientation and scale of the control points. These patches are good indicators of the shape variation.

Other more computationally intensive approaches can also be used, such as image warping based on linear interpolation [11] or thin plate splines [96].

## 4.3 Regression using boosting method

I learn $p(C|I)$ via regression by determining a target density $q(C|I)$ that possesses a desired unimodal and smooth shape. This can be achieved by defining $q(C|I)$ as a normal density of $C$:

$$q(C|I) = N(C; \underline{C}(I), \Sigma), \tag{4.3}$$

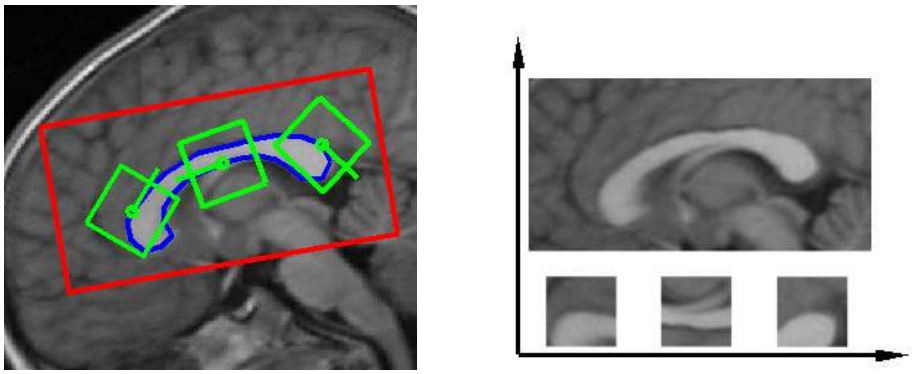


Figure 4.4: The feature image $x$ associated with a hypothesis model $C$. The contour represented by the model $C$ is plotted as a blue line. The subimage enclosed by the red box contains global fitness information. The subimges enclosed by the green boxes contain the local fitness information. The image $x$ is composed by subimages with normalized orientation as shown on the right.

where $\underline{C}(I)$ is the ground truth model for the training image $I$, and $\Sigma$ is an appropriate covariance matrix. In order to simplify computation, I assume the model parameters to be uncorrelated. Therefore, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_D)$. If there is a correlation among model parameters, a linear transformation can be applied to make them uncorrelated.

One additional benefit of using the normal distribution is that it enables the learned regressor to focus on part of the parameter space by setting $\Sigma$ properly. I discuss how to determine $\Sigma$ based on an initial error range in Section 4.4.

I apply regression to fit the density $p(C|I)$ to the target density $q(C|I)$. The density $p(C|I)$ learned in this manner gives an image fitting score for each hypothesis model $C$. This score reflects the distance between the hypothesis and the ground truth. As discussed in Section 4.2.3, a feature image $x(I, C)$ is extracted based on the hypothesis $C$ in the image $I$. The goal is to find a function $f(x(I, C))$ that serves as the density $p(C|I)$. I sample a set of training examples from the annotated input images. A training example is a pair $(x(I_j, C_n), q(C_n|I_j))$, where $I$ is a training image and $C_n$ is a point in the model space. I discuss how to sample $C_n$ in Section 4.4.1. Regression minimizes the

training error while solving the following minimization problem:

$$\hat{f}(x) = \arg\min_{f \in \mathcal{F}} \sum_{n=1}^{N} Loss(q(C_n|I_n), f(x_n)), \tag{4.4}$$

where $N$ is the number of training examples; $\mathcal{F}$ is the set of allowed regressors; and $Loss(\circ, \circ)$ is a function that penalizes the deviation of the regressor output $f(x)$ from the target probability density $q(C|I)$.

### 4.3.1 Boosting

As discussed in Section 2.1.2, when using boosting for regression, regressors take the following additive form:

$$f(x) = \sum_{t=1}^{T} g_t(x); g_t(x) \in \mathcal{G}, \tag{4.5}$$

where each $g_t(x)$ is a weak learner, and $f(x)$ is a strong (more accurate) learner. Furthermore, it is assumed that a weak learner $g(x)$ lies in a dictionary set or weak-learner set $\mathcal{G}$.

Boosting iteratively approximates the target function $q(C|I)$ by adding one more weak learner to the regression output:

$$f'(x) = f(x) + g(x). \tag{4.6}$$

At each round of boosting I select the learner $\hat{g}$ that most decreases the loss function by the following greedy choice:

$$\hat{g} = \arg\min_{g \in \mathcal{G}} \sum_{n=1}^{N} Loss(q(C_n|I_n), f(x_n) + g(x_n)). \tag{4.7}$$

In this chapter I used the quadratic loss function. The solution of Equation 4.7 is simply the weak learner that best predicts the current residuals $P(C_n|I_n) - f(x_n)$.

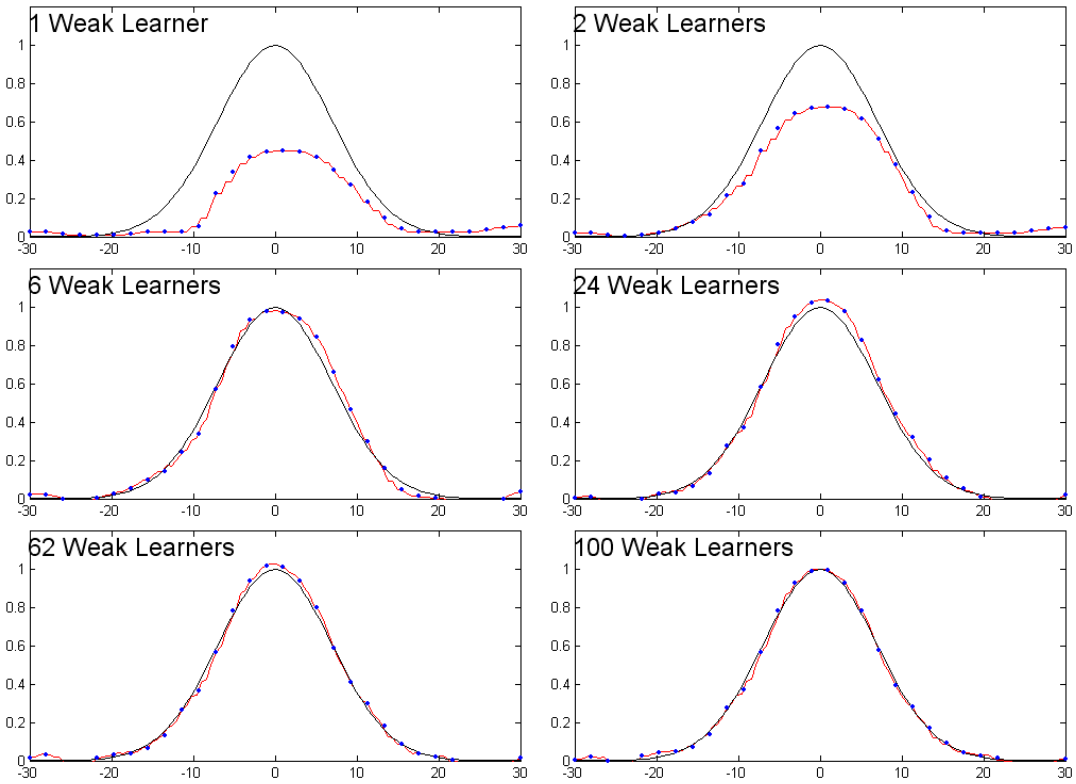



Figure 4.5: A 1D boosting regression example. The target function is a normal density, drawn as black curves. The regressors after the 1st, 2nd, 6th, 24th, 62nd, and 100th round of boosting are drawn as red. The blue dots are the sampled points used in regression.

Shrinkage [13, 30] is a technique for reducing overfitting in boosting methods. The idea is very simple: After each round of boosting, I scale the newly selected learner $g(x)$ by a shrinkage factor $\gamma \in [0, 1]$. The resulting update rule is

$$f'(x) = f(x) + \gamma \hat{g}(x), \tag{4.8}$$

where $\hat{g}$ is the optimal solution found in Equation 4.7. I found that a modest choice of $\gamma = 0.5$ gives good results.

Figure 4.5 illustrates a 1D regression example to illustrate the boosting regression procedure. In this example the model $C$ has only one parameter, and the target function is a 1D normal density, shown as black curves in Figure 4.5. The figure shows the

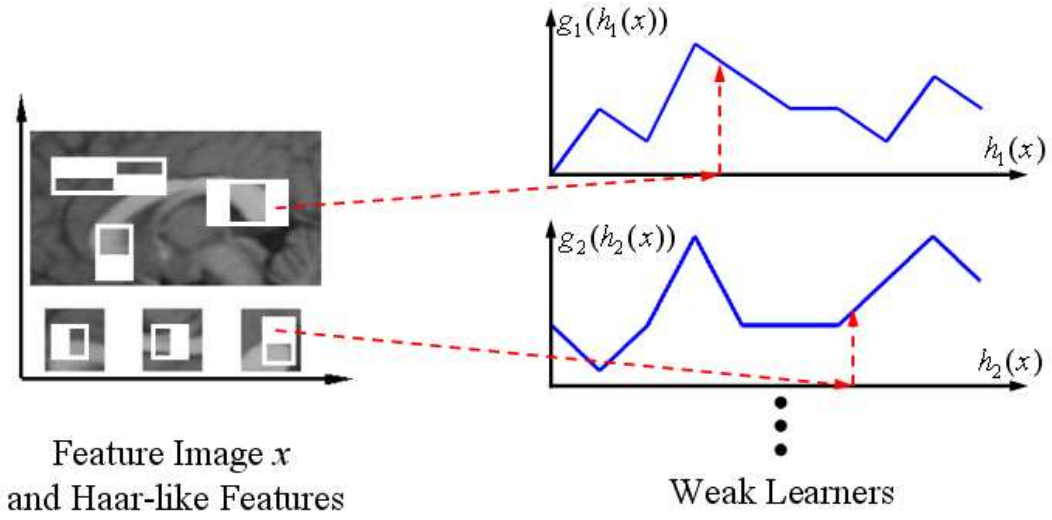



Figure 4.6: Haar-like features and their associated piecewise linear functions.

regressor after the 1st, 2nd, 6th, 24th, 62nd, and 100th round of boosting. The details of the weak learners used in boosting will be discussed in the next section. Observe that the boosting regression increasingly fits the regressor to the target function.

Regression and boosting [32, 29] are widely used for computer vision applications [86, 31, 94, 99, 97]. Image-based regression has been used to directly estimate the shape deformation [99, 97], where the regression output is more than one dimension. I only tackle a single-output regression problem, which is less complex than the regression problem posed in [99, 97].

### 4.3.2 Weak learner

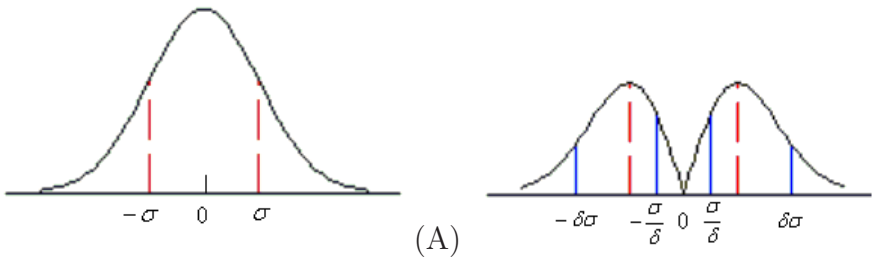
The dictionary set $\mathcal{G}$ contains weak learner candidates, each of which gives a rough fitness measurement of an object shape to the image's appearance. Intuitively, this set must be sufficiently large to enable rendering of the highly complex output function $f(x)$ through a linear combination of weak learners. Also, the computational cost of the feature image $x(I, C)$ and each $g(x(I, C))$ must be low enough for the fast evaluation of $f(x)$. I propose an efficient way of computing $g(x(I, C))$.

First I use Haar-like features to represent image appearance. A Haar-like feature $h(x;\zeta)$ is defined over a rectangular region of an image, where $\zeta$ specifies the attributes of the feature, including the feature type, window position, and window size. I further restrict that the features must be contained within one of the image patches in $x$. Figure 4.6 shows some examples of the Haar-like features. By choosing Haar-like features with different attributes, I obtain an over-complete feature representation of the image $x$. Haar-like features can be evaluated efficiently by pre-computing an integral image. See Section 2.2.2 for details. To further increase the computation speed, I use a set of pre-computed integral images with different orientations to compute $h(x)$, eliminating the need to resample $x$ at run time. This enables the very efficient computation of a weak learner.

I use the piecewise linear functions as the elements of the dictionary set. A piecewise linear function $g(x;\zeta)$ associated with a Haar-like feature models the feature response in a piecewise fashion:

$$g(x;\zeta) = a_{l-1} + \frac{h(x;\zeta) - \eta_{l-1}}{\eta_l - \eta_{l-1}}(a_l - a_{l-1}); \quad h(x;\zeta) \in (\eta_{l-1}, \eta_l], \tag{4.9}$$

where $\{\eta_l;\ l = 0, \ldots, L\}$ divide the range of the feature response equally with $\eta_0 = \min(h(x))$, and $\eta_L = \max(h(x))$ and $\{a_l;\ l = 0, \ldots, L\}$ are the values of the function at the corresponding nodes. At each boosting round $\{a_l\}$ is computed through solving a linear equation for an optimal least-square solution. Figure 4.6 shows the piecewise linear functions associated with the Haar-like features.




(A) (B)

Figure 4.7: The plots of (A) 1D normal distribution $q(C|I)$ and (B) the corresponding gradient magnitude $|\nabla q(C|I)|$.

## 4.4 Multilevel regression and gradient-based sampling

It is difficult to fit a strong learner $f(x)$ to $q(C|I)$ across the entire model space because of the complexity of the image appearance and the lack of sufficient training examples for representing the high-dimensional model space. Even if $f(x)$ approximates $q(C|I)$ well, the gradient magnitude of $f(x)$ may be too small in some regions of the model space to facilitate the fast convergence of local optimization algorithms.

I propose using multilevel regression to tackle this difficulty by training a series of regressors $f_k(x)$, $k = 1, \ldots, K$, with each focusing on a region that gradually narrows to the ground truth. Let $\Omega_k$ be the focus region of $f_k$ in the model space, which is defined as a $D$-dimensional ellipsoid:

$$\Omega_k = \{C = (c_1, c_2, ..., c_D) | \sum_{q=1}^{D} (c_q - \underline{c}_q)^2 / r_{k,q}^2 \leq 1\} \tag{4.10}$$

where $R_k = (r_{k,1}, \ldots, r_{k,D})$ defines extreme values of each dimension.

To gradually decrease segmentation error level by level, I design the focus regions to form a nested structure gradually shrinking to the ground truth:

$$\Omega_1 \supset \Omega_2 \supset \ldots \supset \Omega_K \supset \underline{\Omega} \ni \underline{C}(I), \tag{4.11}$$

where $\Omega_1$ defines the initial region of the model parameters, which can be determined by statistical analysis of the training data. $\Omega_1$ should be big enough to include all the possible solutions in the model space. The final region $\underline{\Omega}$ defines the desired segmentation accuracy. The focus regions of all levels have the same center $\underline{C}(I)$, which is the ground truth model for the training image $I$. In the following discussion I let $\underline{C}(I)$ be at the origin of the model space without loss of generality.

At the testing stage optimization algorithms are sequentially applied to the learned functions to refine the segmentation results. At the $k$th stage the solution falling within the region $\Omega_k$ should be able to be pushed into the region $\Omega_{k+1}$. This requires the learned function $f_k$ to provide effective guidance to the optimization algorithms in the region $\Omega_k - \Omega_{k+1}$. To achieve this goal, I let the function $q_k(C|I)$ exhibit a high gradient in the region $\Omega_k - \Omega_{k+1}$, and I sample more training examples in this region. This leads to a *gradient-based sampling* approach.

Multi-resolution approaches are widely used to improve the efficiency and robustness of the segmentation algorithms [38, 12, 11]. In a typical multi-resolution framework, an image pyramid is built, and the optimization is performed in a coarse-to-fine manner. It is generally true that the local maximums of $p(C|I)$, e.g., the local maximums in Figure 4.1(A), will be smoothed out in the coarse level. However, useful details might also be lost if the resolution of the image is too low. Determining a proper resolution for optimization is usually based on heuristics. I apply the coarse-to-fine principle in a different way, where a series of functions with nested focus regions are learned from a single resolution image. By explicitly defining the shape of each function, the multilevel refinements are more controllable.

Next, I illustrate the construction of focus regions and a related gradient-based sampling strategy using a 1D example and then extend it to higher dimensions.

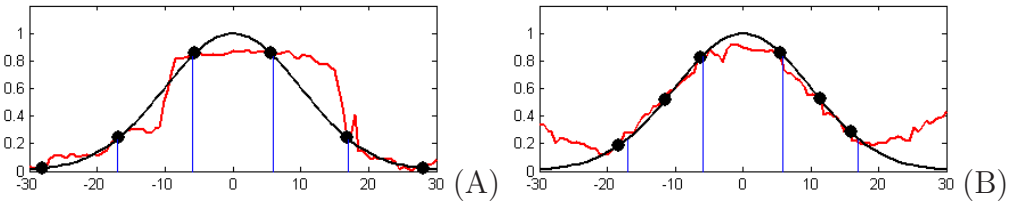



Figure 4.8: The target $q(C|I)$ (black line) and the learned $p(C|I)$ (red line) with (A) uniform sampling and (B) gradient-based sampling.

### 4.4.1 Multilevel regression in 1D

Figure 4.7 shows the plots of a 1D target function $q(C|I)$ and its gradient magnitude $|\nabla q(C|I)|$. From the figure we observe that the gradient magnitude of $q(C|I)$ reaches its maximum at $c_1 = \pm\sigma$ and approaches zero when $c_1 \to \pm\infty$ and $c_1 \to 0$.

Let $\sigma_k$ be the standard deviation of the function $q_k(C|I)$, and let $\delta$ be a pre-specified constant, which is set to be 1.7 for all experiments based on trial and error. I construct the nested focus regions as $\Omega_k = [-\sigma_k\delta, \sigma_k\delta]$ with

$$\sigma_k = r_k/\delta, \quad r_{k+1} = r_k/\delta^2, \tag{4.12}$$

where $r_1$ is the extreme initial value. This construction is based on the fact that the region $[-\sigma_k\delta, -\sigma_k/\delta] \cup [\sigma_k/\delta, \sigma_k\delta]$, centered around the maxima of the gradient magnitude (i.e., $\pm\sigma_k$), contains a large gradient magnitude.

Figure 4.8 compares two sampling strategies: uniform sampling and sampling based on gradient magnitude. Only six examples are used in the regression to simulate the sparsity of sampling in high dimensions. For the moment, I assume that sampling outside of the focus region is allowed. The standard deviation is $\sigma = 10$. In the region $[-\sigma\delta, -\sigma/\delta] \cup [\sigma/\delta, \sigma\delta]$ the regressor learned from sampling based on the gradient magnitude (Figure 4.8(B)) more faithfully captures the shape of the target density function than one learned by uniform sampling (Figure 4.8(A)).

In summary, the learned regressor $f_k(x)$ should fit the target density $q_k(C|I)$ well

enough in the region $\Omega_k - \Omega_{k+1}$ to provide good guidance in optimization. This is achieved by selecting examples primarily in this region that is constructed to have high gradient magnitude. I use the Metropolis sampling algorithm [51] to sample training examples in the focus region $\Omega_k$ with the sampling density function $|\nabla q_k(C|I)|$.

### 4.4.2 Multilevel regression in high dimension

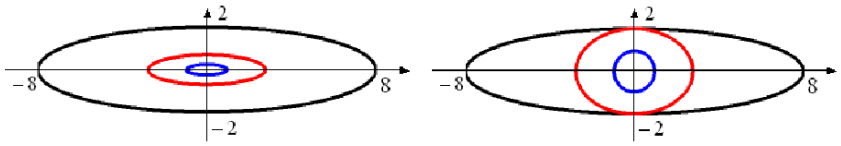
Following the idea developed in the 1D case, I design a series of target densities with nested focus regions. The shape of each target density $q_k(C|I)$ is defined by the covariance matrix $\Sigma_k = \text{diag}(\sigma_{1,k}, \ldots, \sigma_{D,k})$. Determining the optimal covariance matrices $\Sigma_k$ is a hard problem; here I consider two simple approaches.

The first approach is to learn a regressor for decreasing a focus region in all dimensions uniformly. Similar to Equation 4.12, this can be achieved by setting $\sigma_{k,d} = r_{k,d}/\delta$ and $r_{k+1,d} = r_{k,d}/\delta^2$. This approach works well when the elements in $R_1$, which define the initial error range, are roughly the same. However, in real applications, the range of model parameters varies greatly. Typically, the error range of pose parameters is larger than the error range of shape parameters. As a result, narrowing down the error ranges in all dimensions at the same rate is usually inefficient. A more practical solution is to first decrease error along dimensions with larger error ranges. For example, in [31], the pose parameters are determined before estimating shape parameters.

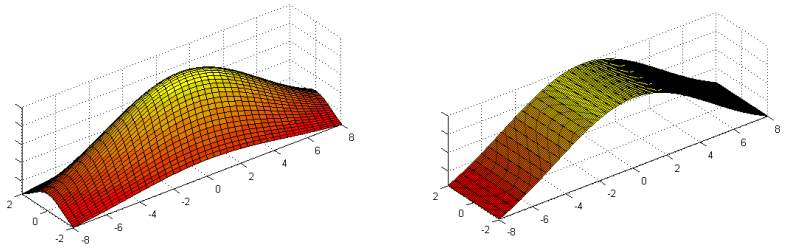
The second approach achieves this goal by allowing the target function to have a large gradient magnitude along the dimensions with large error ranges. The learned regressor therefore focuses on decreasing error along these dimensions. Let $r_k^{\max}$ be the largest element in $R_k$. I evolve $\sigma_{k,d}$ and $r_{k,d}$ as follows:

$$\begin{aligned} &\sigma_{k,d} = r_k^{\max}/\delta, \quad r_{k+1,d} = r_k^{\max}/\delta^2 \quad &&\text{if } r_{k,d} > r_k^{\max}/\delta^2 \\ &\sigma_{k,d} = \sigma_{\max}, \qquad r_{k+1,d} = r_{k,d} &&\text{otherwise} \end{aligned} \tag{4.13}$$

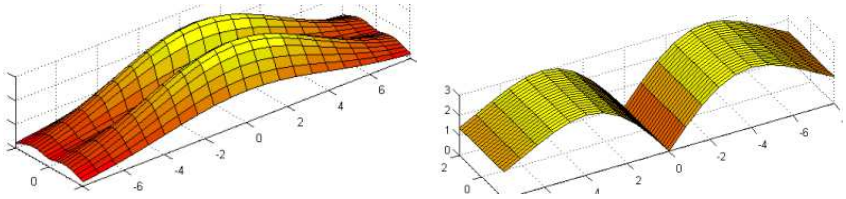
where $\sigma_{\max}$ is a constant, typically having a large value. The condition $\sigma_{k,d} = \sigma_{\max}$



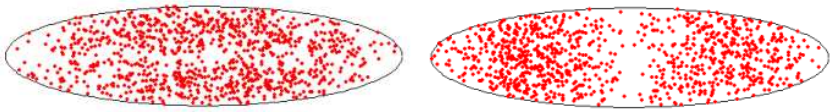

(A) The three nested focus regions defined by $R_1$(black), $R_2$(red) and $R_3$(blue).



(B) The target probability density function $q_1(C|I)$.



(C) The sampling density function $|\nabla q_1(C|I)|$.



(D) The sampled points in the model space based on $|\nabla q_1(C|I)|$.

Figure 4.9: Two sampling approaches. The left column: the first approach. The right column: the second approach.

means that the $k$th target density varies little along the $d$th dimension, and hence the learned regressor $f_k(x)$ makes no attempt to decrease the model error along this direction. Geometrically, the focus region gradually shrinks from a high-dimensional ellipsoid to a sphere and shrinks uniformly thereafter.

I use a 2D example to illustrate the rationale of this approach. Let $R_1 = (8, 2)$, where the error in the first dimension is much larger than that in the second one. For both approaches mentioned above, Figure 4.9 shows the three nested focus regions, the target function $q_1(C|I)$ for the first level, the sampling density function $|\nabla q_1(C|I)|$ and the corresponding sampling results. Both approaches gradually shrink the focus region.

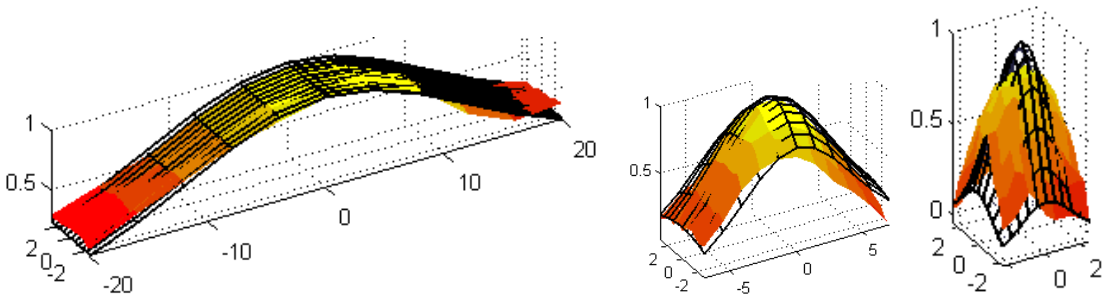



Figure 4.10: The 2D slices of $f_1(x)$(left), $f_2(x)$(middle), and $f_3(x)$(right) on a test image.

The shape of the focus region remains consistent in the first approach, while in the second approach it changes from an ellipse to a circle. The first approach samples many points around the short axis, making the regression less effective. The second approach based on Equation 4.13 is more effective.

## 4.5 Experiments

I present two experiments to test the regression approach and compare its performance with the performance of the other algorithms, including the Active Shape Model (ASM) [12] and the Active Appearance Model (AAM) [11]. In both experiments I used control points to represent objects in the image. The training inputs are a set of images with their associated annotations. In training I build Point Distribution Models (PDMs) following the description in Section 4.2. The initial error ranges of the shape parameters are assumed to be $3\sqrt{\lambda}$.

In testing I used a standard simplex algorithm [60] as the local optimization algorithm to maximize $f_k(x)$. The simplex method is insensitive to shallow maxima caused by image noise and regression error. The learned $f_k(x)$'s are used sequentially, and the converged solution of the current level is the initial value used in the next level. The segmentation errors are measured as the average Euclidean distance between corresponding control points of the segmented shape and the ground truth.

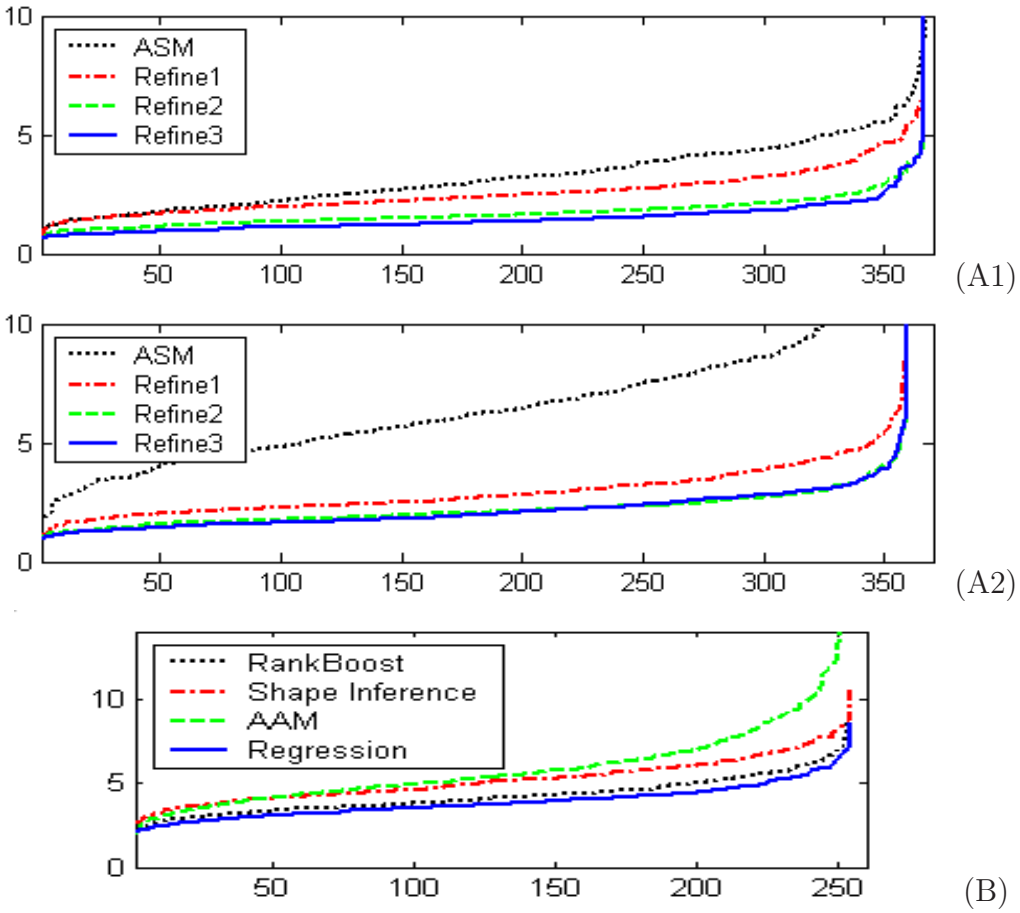



Figure 4.11: Sorted errors of the experiment results: (A1) corpus callosum border segmentation on noise free data, (A2) corpus callosum border segmentation on noisy data, and (B) facial feature localization.

### 4.5.1 Corpus callosum border segmentation

Segmentation of the corpus callosum structure in midsagittal MR images is a common task in brain imaging [74]. In this data set the corpus callosum has a clear border, making segmentation relatively easy. I collected a total of 148 midsagittal MR images with the corpus callosum border annotated by experts using contours with 32 control points. The corpus callosum roughly occupies $120 \times 50$ pixels. I used 74 images for training and the remaining 74 images for testing.

I assume that a rough initial position of the shape is obtained via object detection approaches, e.g., PBN presented in Chapter 3. The goal of segmentation is to locally refine a model $C$. The model $C$ has four rigid transformation parameters and six PDM shape parameters. The initial range of the four rigid transformation parameters is given as $[20, 20, \pi/9, 0.2]$, which means 20 pixels in translation, 20 degrees in rotation and 20% in scale. The six shape parameters account for 85% of the total shape variation. The model parameters were normalized using Equation 4.2 before training.

In training three levels of regressors are trained to approximate the target functions with $\Sigma$ defined in Equation 4.13. I sampled 3,000 examples from each image and set the maximum number of weak learners used by a regressor to be 500. The learned $f_k(x)$'s on a test image are shown in Figure 4.10. The overlaid wire-frame meshes are target functions. Because $f_k(x)$'s are high-dimensional functions, I plotted the 2D slices by varying the first and the fifth parameters of the model in the focus region while fixing the remaining parameters as the ground truth. The first parameter is a translation parameter, and the fifth is a shape parameter corresponding to the largest eigenvalue. The regressors predict the target density well, showing a conspicuous mode, and they are ready to be used in a local optimization algorithm.

In the testing I randomly generated five starting contours for each test image. The initial shape parameters were set to zero, i.e., using the mean shape. The initial pose parameters were randomly generated within the error range defined in the training.

| (A) | ASM | refine1 | refine2 | refine3 |
|---|---|---|---|---|
| w/out noise | 3.33±1.84 | 2.86±3.05 | 2.02±3.08 | 1.76±3.03 |
| | 3.07±1.16 | 2.44±0.74 | 1.63±0.46 | 1.37±0.41 |
| with noise | 6.77±3.16 | 3.66±4.33 | 2.98±4.63 | 2.94±4.68 |
| | 6.31±2.20 | 2.85±0.90 | 2.13±0.59 | 2.08±0.64 |

| (B) | AAM | Shape Inf. | RankBoost | Regression |
|---|---|---|---|---|
| AR face | 5.94±2.81 | 5.16±1.26 | 4.24±1.09 | 3.86±0.97 |
| | 5.50±1.69 | 4.99±1.06 | 4.09±0.88 | 3.72±0.77 |

Table 4.1: The mean and standard deviation of the segmentation errors. There are two rows in each cell: The first row reports the mean and standard deviation obtained using all test data, and the second row using 95% of the test data (excluding 5% outliers).

The average error of 370 starting contours is 12.65 pixels. The computation time of the regression approach is roughly three seconds per case.

I compared the regression approach with ASM. The discussion in [10] explains that ASM works well when the border is supported by strong edges[1]. I also applied multi-resolution searching and carefully tuned the parameters to achieve good performance.

Figure 4.12 shows some segmentation results on the test data. Table 4.1(A) shows the mean and standard deviation of the test errors. The proposed approach improves ASM by 47% in terms of mean error. Figure 4.11(A1) is a plot of the sorted errors, so the points on the curves at each horizontal position do not necessarily correspond to a same test case. There are outliers in the final segmentation results. If 5% of the test data is excluded as outliers, then the proposed approach improves ASM by 55% in terms of mean error, and it also reduces the standard deviation. Furthermore, each level improves the results from the previous level, proving the effectiveness of the multilevel approach.

I then added noise to make the segmentation more challenging. Gaussian noise with zero mean and 0.2 variance was added to both the training and test images with an intensity range $[0, 1]$. The regression approach and ASM were retrained and tested using the same parameter setting as in the noise-free case. Figure 4.13 shows some segmentation results. Observe that the regression approach has robust performance in

[1]I used the Matlab implementation of ASM by Dr. Ghassan Hamarneh, which is available at http://www.cs.sfu.ca/~hamarneh/software/asm/index.html.

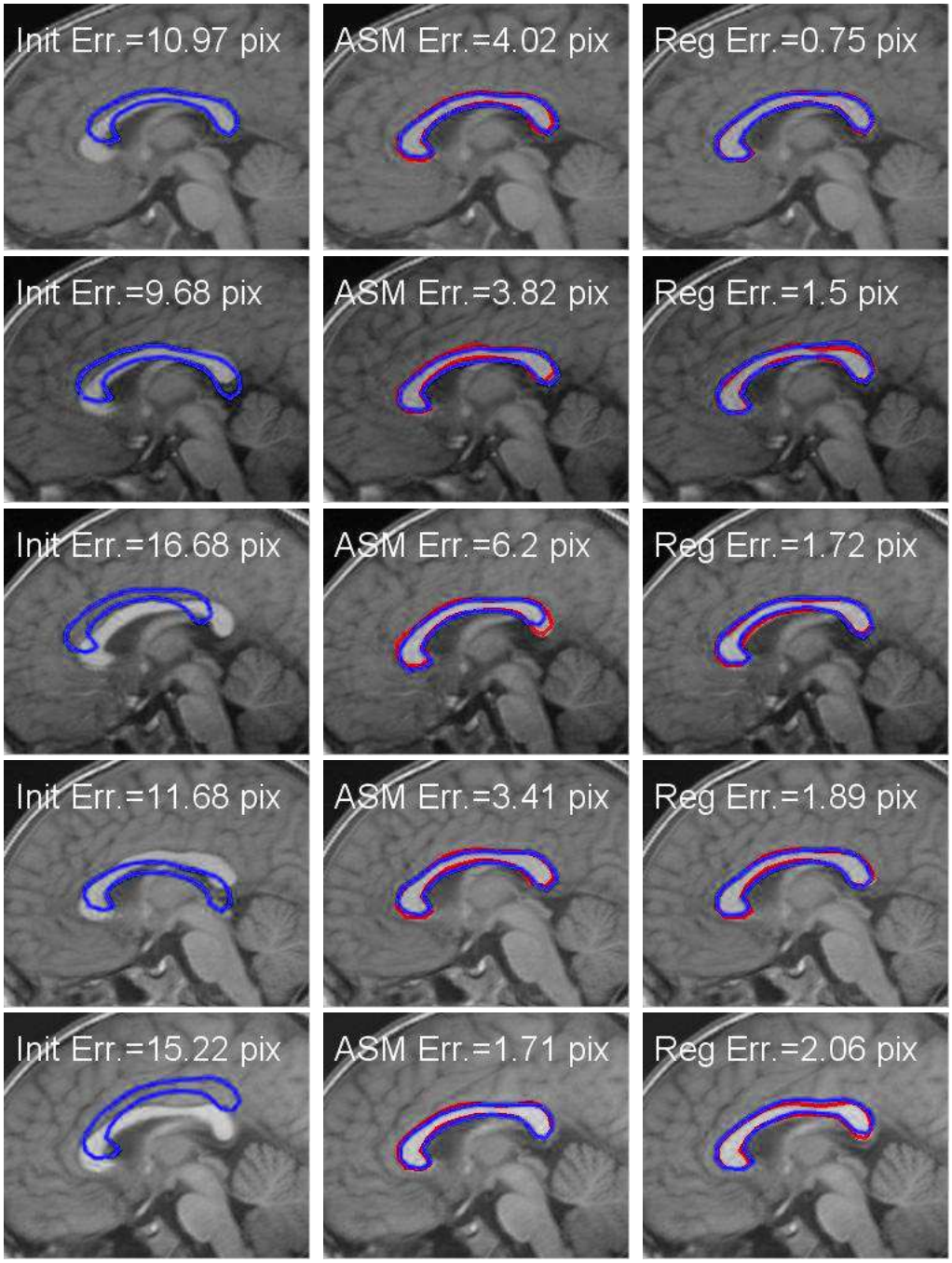



Figure 4.12: Some results of corpus callosum border segmentation. For each row, the left image shows the initial contour, the middle image shows the AAM segmentation result, and the right image shows the result obtained by the regression approach. The initial contours and the segmentation results are blue lines. The ground truths are red lines.

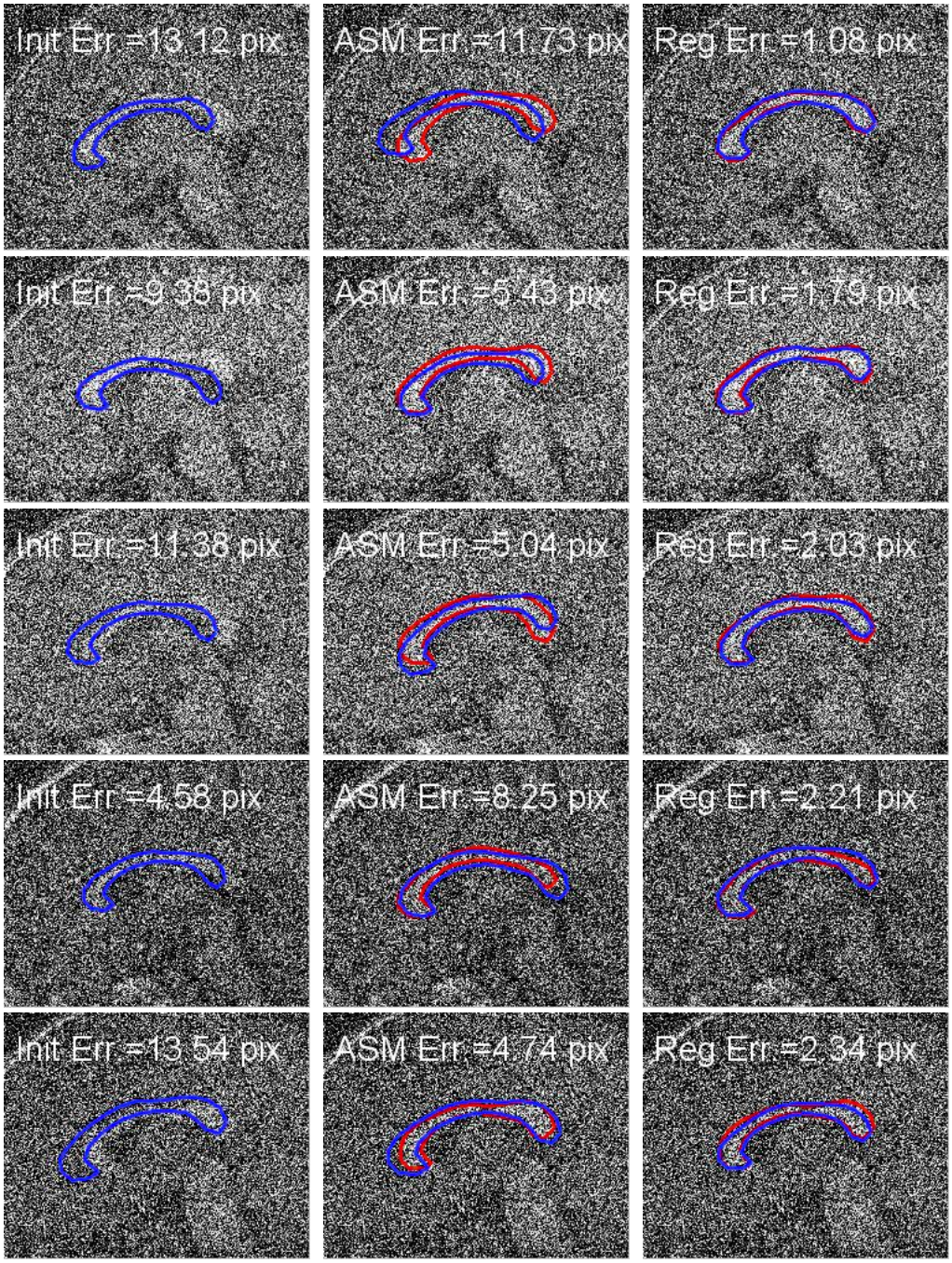



Figure 4.13: Some results of corpus callosum border segmentation in noisy images. For each row, the left image shows the initial contour, the middle image shows the AAM segmentation result, and the right image shows the result obtained by the regression approach. The initial contours and the segmentation results are blue lines. The ground truths are red lines.

such noisy conditions. The benchmarks are shown in Table 4.1(A) and Figure 4.11(A2). ASM has a poor performance because it cannot estimate accurate intensity profiles normal to the object boundary because of the noise. The performance of the regression suffers less because the added noise is considered at the training stage. The algorithm overcomes the noise corruption by selecting features less sensitive to the noise and hence provides robust segmentation. This proves that the learning approach can automatically adapt to noisy images as long as similar type and level of noise consistently exists in both training and test data. Note that in this experiment, the third level refinement cannot improve the segmentation further because of information loss caused by noise.

### 4.5.2 Facial feature localization

In the second experiment I tested the algorithms on facial feature localization using the AR face database [49]. There are a total of 508 images with annotations[2], including 76 males and 60 females with four expressions.

I used exactly the same training and test set as in [96]. Half of the data were used for training and half for testing. Examples of the same subject were not used in both training and test data. The color images were converted to gray-scale images. I also assumed that the rigid transformations of the faces are known; the focus is on localizing the non-rigid shape component. The model space is defined by ten shape parameters, which explain 86% of the total shape variation. I sampled 1,200 examples from an image and trained three levels of regressors. The mean shape is used as the starting point at the testing stage. The average initial error is 5.93 pixels.

I compared the performance of the regression approach to the algorithms listed in [96]: AAM [73], shape inference [31], and shape refinement based on rankboost [96]. Examples of localization is shown in Figure 4.14. The localization errors of the four

[2]The annotations are provided by Dr. Timothy Cootes, and they are available at http://www.isbe.man.ac.uk/~bim.

algorithms are shown in Table 4.1(B) and in Figure 4.11(B). Compared to shape inference and rankboost, which only consider the candidate shapes in the training set, the regression approach searches over the whole shape space. Again, the regression approach records the best performance.

## 4.6 Summary

I have presented a regression approach to learning a conditional density function. The target function can be artificially constructed to be both smooth and unimodal, and therefore easy to optimize. To learn the regressor, the boosting principle has been employed to select relevant features. I have also adopted a multilevel strategy to learn a series of conditional density functions, each of which guides the solution of optimization algorithms increasingly converging to the correct solution. I have then proposed a gradient-based sampling strategy to maximize the convergence in the directions of largest variation. I have successfully applied the regression approach to segmenting corpora callosa and localizing facial feature points. The results of the regression approach consistently outperform those obtained by state-of-the-art methods.


Figure 4.14: Some facial feature localization results. For each row, the first image shows the initial position, and the remaining images show the results obtained by AAM, shape inference[31], shape refinement based on rankboost [96], and the regression approach. The ground truth is red dots, and the localization result is blue dots.

# Chapter 5

# Deformable Shape Segmentation via Ranking and Classification

Deformable shape segmentation can be considered as searching through a model space for the model that best represents a target shape. In order to measure the fitness of a hypothesis model, fitting functions are built for characterizing the relationship between the model and image appearance. A desired fitting function should differentiate correct models from their background in the model space. As discussed in Section 2.1, discriminative models outperform generative models in terms of discriminative accuracy when the training data are sufficient.

In the previous chapter a regression-based approach was proposed to learn the fitting function. The target fitting function was constrained to be unimodal and smooth in the model space, which can be used by local optimization algorithms to efficiently estimate the correct solution. The algorithm demonstrated superior performance on segmenting corpus callosum borders from clean and noisy images and on localizing facial feature points. In addition to regression, two other discriminative learning approaches – classification and ranking – can also be used to learn fitting functions.

In this chapter I present a comparative study on how to apply three discriminative learning approaches – classification, regression, and ranking – to deformable shape segmentation [93]. By using discriminative learning in the model space, the fitting function

can be learned in a steerable manner. I discuss how to extend the classification approach from object detection to deformable object segmentation. I also propose a ranking-based approach for learning the fitting functions. The fitting function is trained to produce the highest score around the ground truth solution, and it also possesses a suitable shape to guide optimization algorithms to this solution. To address the high-dimensional learning challenges presented in the learning framework, I apply a multilevel approach, like that proposed in Section 4.4, to learn all discriminative models.

In Section 5.1 I discuss how to solve segmentation problems via classification, regression, and ranking approaches. I also compare these three approaches in terms of learning complexity and computational cost. In Section 5.2 I address a common challenge of the three approaches, which is to learn a discriminative function in a high-dimensional model space. In Section 5.3 I develop a ranking-based approach, and in Section 5.4 I compare the performance of the three approaches on various test data sets.

## 5.1 Discriminative learning approaches

Following the discussion of Section 4.2, a shape in an image $I$ can be parameterized by a set of continuous model parameters, $C$, which contains both rigid and non-rigid components. Given an image, $I$, and a hypothesis model, $C$, a feature image, $x(I, C)$, can be extracted to describe the image appearance associated with $C$. A supervised learning approach attempts to train a fitting function $f(x(I, C))$ based on a set of training images $\{I\}$ and their corresponding ground-truth shape models $\{\underline{C}\}$. The desired output of $f$ is specific to the discriminative approach.

### Classification

A classification approach learns a classifier $f$ to indicate whether a hypothesis shape $C$ matches the one seen in image $I$ or not. The desired output $y$ of $f$ is a signed binary

value. Whether a feature image $x(I, C)$ is positive or negative is determined by the distance between $C$ and the ground truth model $\underline{C}$ as provided in the image $I$. The desired classification output is

$$y = \begin{cases} 1 & \text{if } \| C - \underline{C} \| \leq \epsilon \\ -1, & \text{otherwise} \end{cases}, \tag{5.1}$$

where $\epsilon$ is a threshold that determines the aperture of $f$. The learned $f(x(I, C))$ is a boxcar function around the ground truth. Figure 5.1(A) shows an ideal learned function $f$ when $C$ is one dimensional. Because the learned $f$ only provides binary indication, an exhaustive search is necessary to estimate the solution, which is computationally prohibitive when the dimensionality of the model $C$ is high.

## Regression

The regression approach, presented in Chapter 4, learns a regression function $f$ with real-valued output, which indicates the fitness of a hypothesis model $C$ to an image $I$. In order to facilitate searching, the desired output $y$ of $f$ is constrained to be a normal distribution:

$$y = \mathcal{N}(C; \underline{C}, \Sigma), \tag{5.2}$$

where $\Sigma$ is a covariance matrix determining the aperture of $f$. The learned $f$ has a smooth and unimodal shape, e.g., a 1D example as shown in Figure 5.1(B). The regression function $f$ learned in this way can be effectively optimized by general-purpose local optimization techniques, such as gradient descent or simplex, due to $f$'s single maximum and smoothness. However, when compared with a classification approach, the desired output of $f$ is more complicated than necessary; more information is learned at the training stage to allow the regressor to produce a real value for each point in the model space.

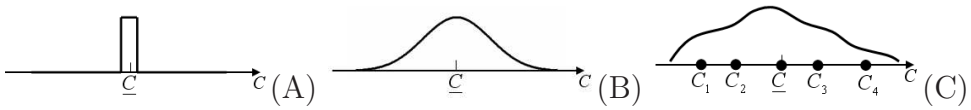



Figure 5.1: The learned $f(I, C)$ when $C$ is one dimensional: (A) a classification approach, (B) the regression approach, and (C) a ranking approach. The ground truth of the model is $\underline{C}$.

## Ranking

Discriminative learning via ranking was originally proposed to retrieve information based on user preference [27]. Ranking approaches were also used to retrieve candidate shapes from a database containing example shapes in image segmentation applications [2, 96].

In this chapter I propose a ranking approach for learning a partial ordering of points in the model space. The ordering learned by the ranking function provides the essential information to guide the optimization algorithm at the testing stage. Unlike a regression approach, which forces the regressor to produce an exact value at each point in the model space, ranking only tries to learn relative relations between point pairs in the model space. Let $(C_0, C_1)$ be a pair of points in the model space, and $(x_0, x_1)$ be its associated feature image pair. The ordering of $x_0$ and $x_1$ is determined by their shape distance to the ground truth: The one closer to the ground truth has a higher rank. I learn a ranking function $f$ to satisfy the following constraint:

$$\begin{cases} f(x_0) > f(x_1) & \text{if } \| C_0 - \underline{C} \| < \| C_1 - \underline{C} \| \\ f(x_0) < f(x_1) & \text{if } \| C_0 - \underline{C} \| > \| C_1 - \underline{C} \| \\ f(x_0) = f(x_1), & \text{otherwise} \end{cases} \tag{5.3}$$

Figure 5.1(C) illustrates the basic idea of the ranking approach. There are five points in the 1D model space, and $\underline{C}$ is the ground truth. At the training stage, a ranking function $f$ is learned to satisfy the ordering constrains: $f(x(I, \underline{C})) > f(x(I, C_2))$, $f(x(I, C_2)) > f(x(I, C_1))$, $f(x(I, \underline{C})) > f(x(I, C_3))$, and $f(x(I, C_3)) > f(x(I, C_4))$. Similar to the regression approach, the learned ranking function $f$ is unimodal, which

is desirable for local optimization techniques. However, the model of ranking is simpler, in the sense that the amount of information to be learned for ranking is less than the amount for regression. The regression approach learns a full ordering of points in the model space, while the ranking approach only learns a partial, pairwise ordering.

Similar to the boosting-based regression, I employ the boosting principle to learn the ranking function by selecting relative features to form an additive committee of weak learners. Each weak learner, based on a Haar-like feature that can be computed rapidly, provides a rough ranking. The learned ranking function combines the rough ranking from weak learners and provides the robust ordering information in the shape model space. I will discuss the detailed implementation of the ranking algorithm in Section 5.3.

## 5.2 Learning in a high-dimensional space

The first step toward learning a discriminative function is to sample training examples in the model space. As a result of the curse of dimensionality, the number of training examples should be an exponential function of the model's dimensionality to ensure training quality. This poses a huge challenge to applying discriminative learning to deformable segmentation applications, where the dimensionality of the model space is usually high. Another challenge is the increasing difficulty of discriminating the correct solution from its background when the background points get closer to the solution. In this situation the image appearance of the background points becomes more and more similar to that of the correct solution. As a result of these two challenges, learning a single function across the whole model space to accurately distinguish the optimal solution from its background is ineffective.

I employ the multilevel approach proposed in Section 4.4 to learn a series of discriminative functions $f_k$, $k = 1, \ldots, K$, each of which is defined over a focus region $\Omega_k$. I use

the second approach in Section 4.4.2 to evolve the focus region, where the region gradually shrinks from a high-resolution ellipsoid to a sphere and shrinks uniformly thereafter. In testing the $k$th function should be able to guide optimization algorithms to push a solution falling within the region $\Omega_k - \Omega_{k+1}$ into the region $\Omega_{k+1}$. Different data sampling strategies will be designed based on the properties of the three discriminative learning approaches discussed in this chapter.

## Classification

For the classification approach, the learned function $f_k$ should be able to differentiate the instances in the region $\Omega_{k+1}$ from those in the region $\Omega_k - \Omega_{k+1}$. In order to achieve this, the positive examples are sampled from the region $\Omega_{k+1}$ and the negatives from the region $\Omega_k - \Omega_{k+1}$. Figure 5.2 shows an example of two-dimensional sampling. The choice of shrinking rate $\delta$ used in Equation 4.13 is balanced by two factors. A large $\delta$ means a small positive region, which requires a fine search grid to detect the solution at the testing stage. This causes a high computational overhead at the testing stage. On the other hand, a small $\delta$ means a large positive region, where the image appearance of the instances has large variation. This leads to inaccurate classifiers in the training stage.

## Regression

For the regression approach, a gradient sampling is proposed in Section 4.4: The learned regressors provide guidance to optimization algorithms based on the local gradient. Because the regressor $f_k$ has a large gradient in the region $\Omega_k - \Omega_{k+1}$, more training examples are drawn from the region $\Omega_k - \Omega_{k+1}$ to ensure the training quality.

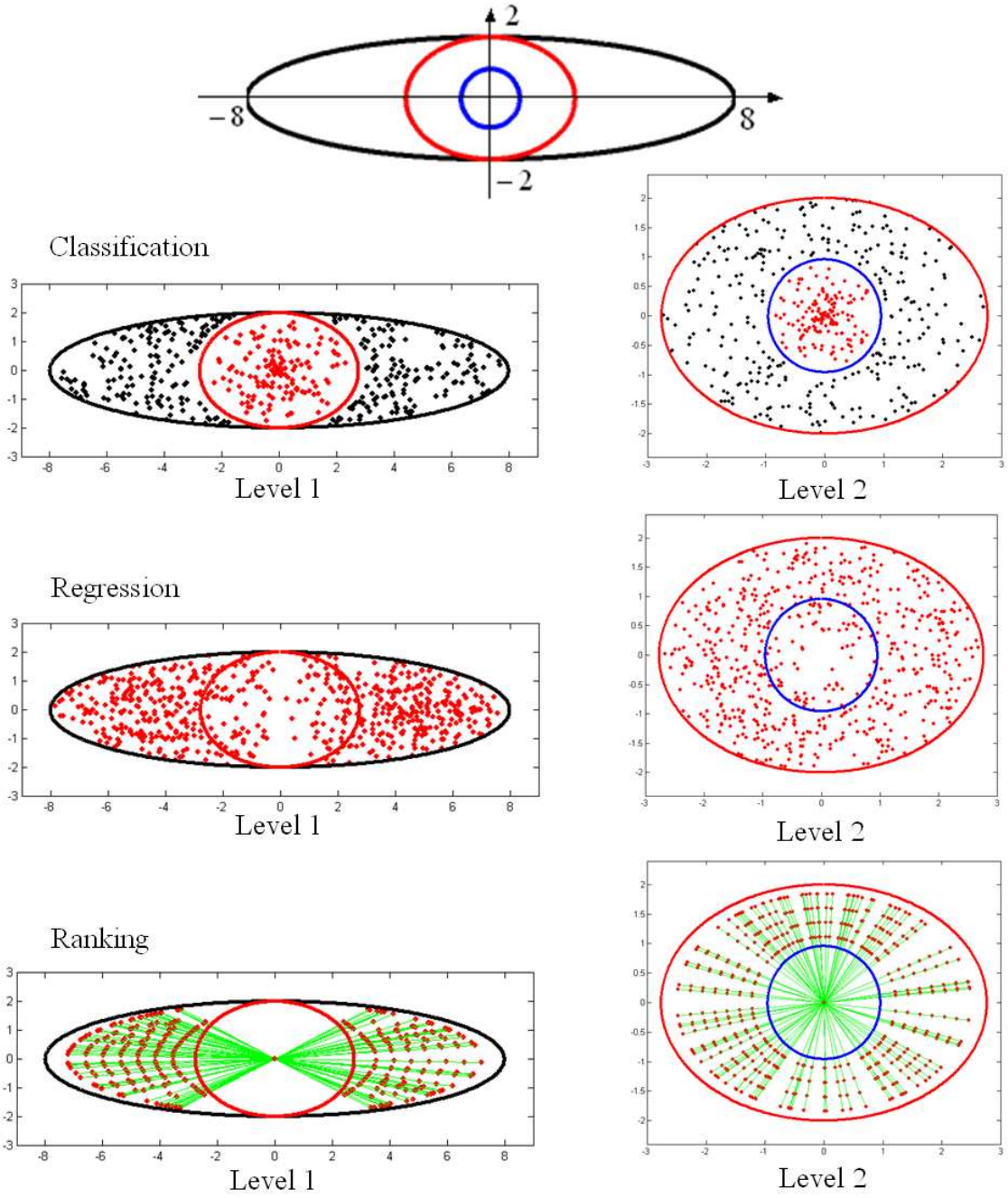



Figure 5.2: An example of sampling training data in a 2D model space. The first row shows the three nested focus regions defined by $R_1$(black), $R_2$(red), and $R_3$(blue). The second row shows the sampling results of classification, where positive examples are drawn as red dots and negative examples as black dots. The third row shows the sampling results of regression. The fourth row shows the sampling results of ranking.

### Ranking

The objective of the ranking approach is to learn a partial ordering of instances in the model space. Because I perform a line-searching optimization approach, the ordering is established along the rays starting from the ground truth. This ordering provides the essential information to guide the optimization algorithms to the ground truth. Also, by learning the ordering information from enough rays, the learned ranking function is unimodal with its global optimum at the ground truth.

I propose a sampling algorithm to choose the training pairs for learning the function $f_k$. First, select a ray starting from the ground truth with a random direction. Then sample $J+1$ points $\{C_0, C_1, \ldots, C_J\}$ on the selected ray, where $C_0$ is at the ground truth, and the remaining $J$ points are sampled from the line segment in the region $\Omega_k - \Omega_{k+1}$. These points are ordered based on the distance to the ground truth. The parameter $J$ is proportional to the length of the line segment. The reason for sampling only from the line segment is that the ordering on this part of the ray is most important for training $f_k$, which is used to push the solution from the region $\Omega_k - \Omega_{k+1}$ to $\Omega_{k+1}$. Finally, from the training image $I$, draw $J$ pairs of training examples $\{(x(I, C_j), x(I, C_{j-1})), j = 1, \ldots, J)\}$, where $x(I, C_{j-1})$ should be ranked above $x(I, C_j)$. This process is repeated to sample as many training pairs as available computer memory allows. Figure 5.2 shows the sampling result in a 2D model space.

## 5.3 Ranking using a boosting algorithm

In this section I present a ranking algorithm based on RankBoost [27] to learn the ordering of sampled image pairs. Mathematically, I learn a ranking function that minimizes the number of image pairs that are mis-ordered. Let $\Phi$ be the sampled training set and $(x_0, x_1) \in \Phi$ be a pair of images. Following the sampling strategy proposed in the previous section, $x_1$ should be ranked above $x_0$, otherwise a penalty $D(x_0, x_1)$ is

imposed. I use an equal-weighted penalty $D(x_0, x_1)$ in my experiments. The penalty weights can be normalized over the entire training set to a probability distribution $\sum_{(x_0,x_1)\in\Phi} D(x_0, x_1) = 1$.

The learning goal is to search for a ranking function $f$ that minimizes the ranking loss:

$$rloss_D(f) = \sum_{(x_0,x_1)\in\Phi} D(x_0, x_1)[\![f(x_0) \geq f(x_1)]\!], \tag{5.4}$$

where $[\![\Pi]\!]$ is defined to be 1 if the predicate $\Pi$ holds and 0 otherwise. In RankBoost the ranking function $f(x)$ takes the following additive form:

$$f_t(x) = f_{t-1}(x) + \alpha_t g_t(x) = \sum_{i=1}^{t} \alpha_i g_i(x), \tag{5.5}$$

where each $g_i(x)$ is a weak learner residing in a dictionary set $\mathcal{G}$. It maps a feature image $x$ to a real-valued ranking score. The strong learner $f(x)$ combines the weighted ranking scores from weak learners to obtain a robust ranking. Boosting is used to iteratively select weak learners by leveraging the additive nature of $f(x)$: At iteration $t$, one more additive term $\alpha_t g_t(x)$ is added to the ranking function $f_{t-1}(x)$. The weak learner $g_t(x)$ is selected from the set $\mathcal{G}$, and its associated weight $\alpha_t$ is computed to minimize the ranking loss:

$$(g_t, \alpha_t) = \arg \min_{g\in\mathcal{G},\alpha\in\mathbf{R}} \sum_{(x_0,x_1)\in\Phi} D(x_0, x_1)[\![f_{t-1}(x_0) + \alpha g(x_0) \geq f_{t-1}(x_1) + \alpha g(x_1)]\!]. \tag{5.6}$$

The RankBoost algorithm is shown in Figure 5.3. I now discuss the choice of weak learners.

1. Given: initial distribution $D$ over $\Phi$.
2. Initialize: $D_1 = D$.
3. For $t = 1, 2, \ldots, T$
   - Train the weak learner $g_t$ using distribution $D_t$.
   - Choose $\alpha_t \in \mathbf{R}$.
   - Update: $D_{t+1}(x_0, x_1) = \frac{D_t(x_0,x_1)\exp[\alpha_t(g_t(x_0)-g_t(x_1))]}{Z_t}$ where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution.)
4. Output the final ranking: $f(x) = \sum_{t=1}^{T} \alpha_t g_t(x)$.

Figure 5.3: The RankBoost algorithm [27].

## Weak learners

The input to a weak learner is a feature image $x$. I use the Haar-like feature set $\{h(x;\zeta)\}$ discussed in Section 4.3.2 to represent image information, where $\zeta$ specifies the attribute of the features.

For each feature $h(x;\zeta)$, I use a 1D binary function $g(x;\zeta)$ with $L$ bins to produce a weak ranking

$$g(x;\zeta) = \beta_l; \quad \text{if } h(x;\zeta) \in (\eta_{l-1}, \eta_l] \tag{5.7}$$

where $\{\eta_l; l = 0, \ldots, L\}$ evenly divide the output range of the feature $h(x;\zeta)$ to $L$ bins and $\beta_l \in \{0, 1\}$ is the value of the $l$th bin.

Based on the discussion in [27], when the output of a weak learner has the range $[0, 1]$, the weak learner should be trained to maximize the following function:

$$r = \sum_{(x_0,x_1)\in\Phi} D(x_0, x_1)(g(x_1;\zeta) - g(x_0;\zeta)). \tag{5.8}$$

Following the definition in Equation 5.7, the function $r$ can be rewritten as

$$r = \sum_{l=1}^{L} \beta_l e_l, \tag{5.9}$$

where

$$e_l = \sum_{h(x_1)\in(\zeta_{l-1},\zeta_l]} D(x_0, x_1) - \sum_{h(x_0)\in(\zeta_{l-1},\zeta_l]} D(x_0, x_1). \tag{5.10}$$

It is easy to show that, in order to maximize $r$, the $l$th bin value should be determined as

$$\beta_l = \begin{cases} 1 & \text{if } e_l > 0 \\ 0, & \text{otherwise} \end{cases}, \tag{5.11}$$

I exhaustively check all weak learners and find the one that produces the largest $r$ value at each boosting iteration. The associated weight $\alpha$ is then computed as

$$\alpha = \frac{1}{2} \ln \left( \frac{1 + r_{\max}}{1 - r_{\max}} \right). \tag{5.12}$$

## 5.4 Experiments

I tested the proposed discriminative learning approaches on three problems. The boosting principle is used in all three approaches to select and combine weak learners. For classification, I implemented the cascade of boosted binary classifiers based on Adaboost [86], which has been successfully applied to fast object detection. For regression, the approach described in the previous chapter was applied. To fairly compare the three algorithms, I used the same set of Haar-like features discussed in Section 4.3.2. I also compared the three algorithms with other approaches, such as ASM [12] and AAM [11]. In order to enhance the performance of ASM, I also implemented an enhanced ASM version that replaces the regular edge computation with boundary classifiers, which is similar to the approach proposed in [95, 81].

To handle the challenge of learning in a high-dimensional space, I used the multilevel approach to learn a series of discriminative functions. The initial error ranges of the parameters are set to control the sampling range in training. The initial error ranges of the shape parameters are assumed to be $3\sqrt{\lambda}$, where $\lambda$'s are eigenvalues from PCA.

| (A) LV | ASM | enhanced ASM | Classification | Regression | Ranking |
|---|---|---|---|---|---|
| level 1 | n/a | n/a | 15.85±5.51<br>15.15±4.65 | 11.09±4.31<br>10.43±3.11 | 10.12±3.26<br>9.69±2.69 |
| final level | 26.20±17.64<br>23.43±12.03 | 17.91±6.80<br>17.07±5.82 | 14.77±6.53<br>13.86±5.25 | 10.07±4.52<br>9.41±3.06 | 9.93±3.56<br>9.37±2.62 |
| time (s) | 0.94 | 1.43 | 18.7 | 3.15 | 2.86 |
| (B) LA | ASM | enhanced ASM | Classification | Regression | Ranking |
| level 1 | n/a | n/a | 16.37±5.96<br>15.60±4.95 | 14.72±13.59<br>12.10±4.02 | 11.66±6.14<br>10.79±3.69 |
| final level | 30.14±17.72<br>28.01±12.44 | 18.66±10.09<br>16.93±5.83 | 15.95±6.78<br>15.17±5.95 | 14.12±16.04<br>11.03±4.64 | 11.40±6.75<br>10.44±4.14 |
| time (s) | 0.75 | 1.26 | 18.1 | 2.80 | 2.18 |
| (C) AR | AAM | | Classification | Regression | Ranking |
| level 1 | n/a | | 18.28±7.07<br>17.23±5.29 | 17.40±6.34<br>16.59±5.30 | 14.80±5.63<br>13.99±4.49 |
| level 2 | n/a | | 12.94±6.11<br>11.88±3.60 | 8.39±4.09<br>7.72±1.92 | 6.84±2.48<br>6.47±1.87 |
| final level | 19.70±23.83<br>15.87±17.23 | | 11.66±6.77<br>10.53±4.35 | 5.76±3.99<br>5.10±1.38 | 5.79±2.95<br>5.31±2.07 |
| time (s) | 0.91 | | 29.4 | 4.70 | 3.49 |

Table 5.1: The mean and standard deviation of the segmentation errors. There are two rows in each cell: The first row reports the mean and standard deviation obtained using all of the test data, and the second row using 95% of the test data (excluding 5% outliers). For ASM and AAM, multi-resolution searching was applied, but only the benchmarks of the final results were reported.

I sampled as many training examples as available memory allowed. For a computer with 2GB memory, about 400K training examples are used. In testing, for each test image I randomly generated an initial contour, whose pose parameters are within the error range defined in training and whose shape parameters are zeros (mean shape). Starting from an initial solution, the learned functions are sequentially applied to refine the solution. For the classification approach, I searched exhaustively around the initial solution and found the candidate having the highest classification probability as the starting point for the next level. For the regression and ranking approaches, I used the simplex optimization method [60] due to its tolerance to shallow maxima caused by image noise.

### 5.4.1 Endocardial wall segmentation: left ventricle

In this experiment I focused on segmenting the endocardial wall of the left ventricle (LV) in the apical four chamber (A4C) view of the echocardiogram. The LV appearance varies

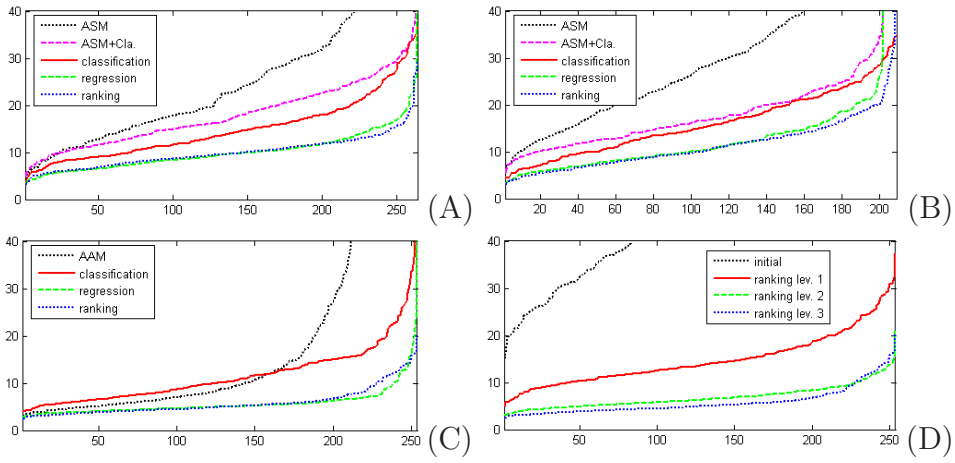



Figure 5.4: Sorted errors of the experimental results: (A) LV segmentation, (B) LA segmentation, (C) facial feature localization, and (D) the errors of the multilevel refinement using the ranking approach in the third experiment. The horizontal axes are the ranks of the test numbers. The vertical axes are segmentation errors.

significantly across patients, and ultrasound images often suffer from signal dropouts and speckle noise. There is no clear edge defining the endocardial wall. This combination of factors makes automatic LV segmentation challenging.

The data set has 528 A4C images from different patients. The LV walls are annotated by experts using contours with 17 control points. The size of the LV in an image is roughly $120 \times 180$ pixels. Half of the data set is used for training and the remainder for testing. The initial range of the pose is set as $[50, 50, \pi/9, 0.2]$, which means 50 pixels in translation, 20 degrees in rotation, and 20% in scale in the extreme. The model $C$ includes five shape parameters, which account for 80% of the total shape variation. Two levels of discriminative functions are learned in training.

The segmentation error is defined as the average Euclidean distance between corresponding control points of the segmented shape and the ground truth shape. In testing I set the initial pose of a test image to be a random perturbation of the ground truth within the initial range $[50, 50, \pi/9, 0.2]$. The average distance between the initial contours and the ground truths in all test images was 27.16 pixels. I tested the classification, regression, and ranking algorithms, along with ASM and its enhanced version. Table

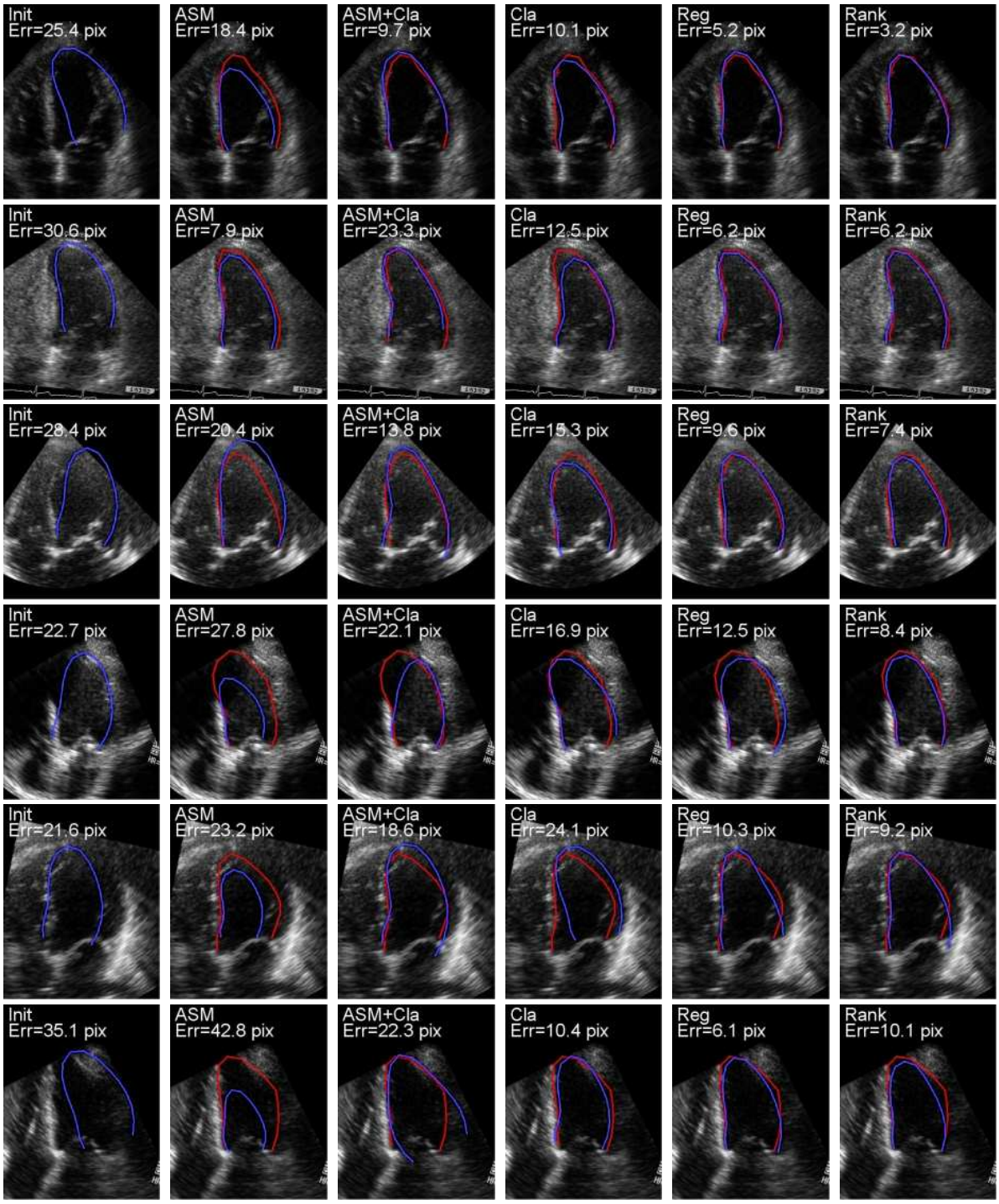



Figure 5.5: Some results of left ventricle segmentation. For each row, the left image shows the initial contour, and the subsequent images show the segmentation results obtained by ASM, enhanced ASM, the classification, the regression, and the ranking approach. The initial contours and the segmentation results are blue lines. The ground truths are red lines.

5.1(A) shows the mean and standard deviation of the test errors of the above algorithms. The average computational time is also listed. Figure 5.4(A) is a plot of the sorted errors, so the points on the curves at each horizontal position do not necessarily correspond to a same test case. Figure 5.5 shows some segmentation results. Section 5.4.4 presents the discussions of the experimental results.

### 5.4.2 Endocardial wall segmentation: left atrium

In this experiment I tested the algorithms on segmenting the endocardial wall of the left atrium (LA) in the apical two chamber (A2C) view. LA segmentation is harder than LV segmentation. The LA appearance is noisier because the LA, being in the far field of the ultrasound probe, is more difficult to image. I collected 417 A2C images with the LA walls annotated by experts using 17 control points. The LA roughly occupies $120 \times 120$ pixels in an image. I used 208 images in training and the remaining 209 images in testing. The initial range of the pose is set as $[50, 50, \pi/9, 0.2]$. The model $C$ includes four shape parameters, which account for 88% of the total shape variation. I trained two levels of discriminative functions. The average initial error is 26.82 pixels in testing. I used the same experimental setting as in the LV segmentation. Figure 5.6 shows some segmentation results. The benchmarks are shown in Table 5.1(B) and Figure 5.4(B). Section 5.4.4 presents the discussions of the experimental results.

### 5.4.3 Facial feature localization

In the third experiment I tested the performance of the algorithms on the AR face database [49]. The AR database was also used in the previous chapter. The difference is that the model $C$ used here includes rigid-transformation parameters, while $C$ only included deformable shape parameters in the previous chapter. There are 508 images with annotations, each of which is defined by 22 control points. The color images were converted to gray images. The size of a face is roughly $250 \times 300$ pixels in an

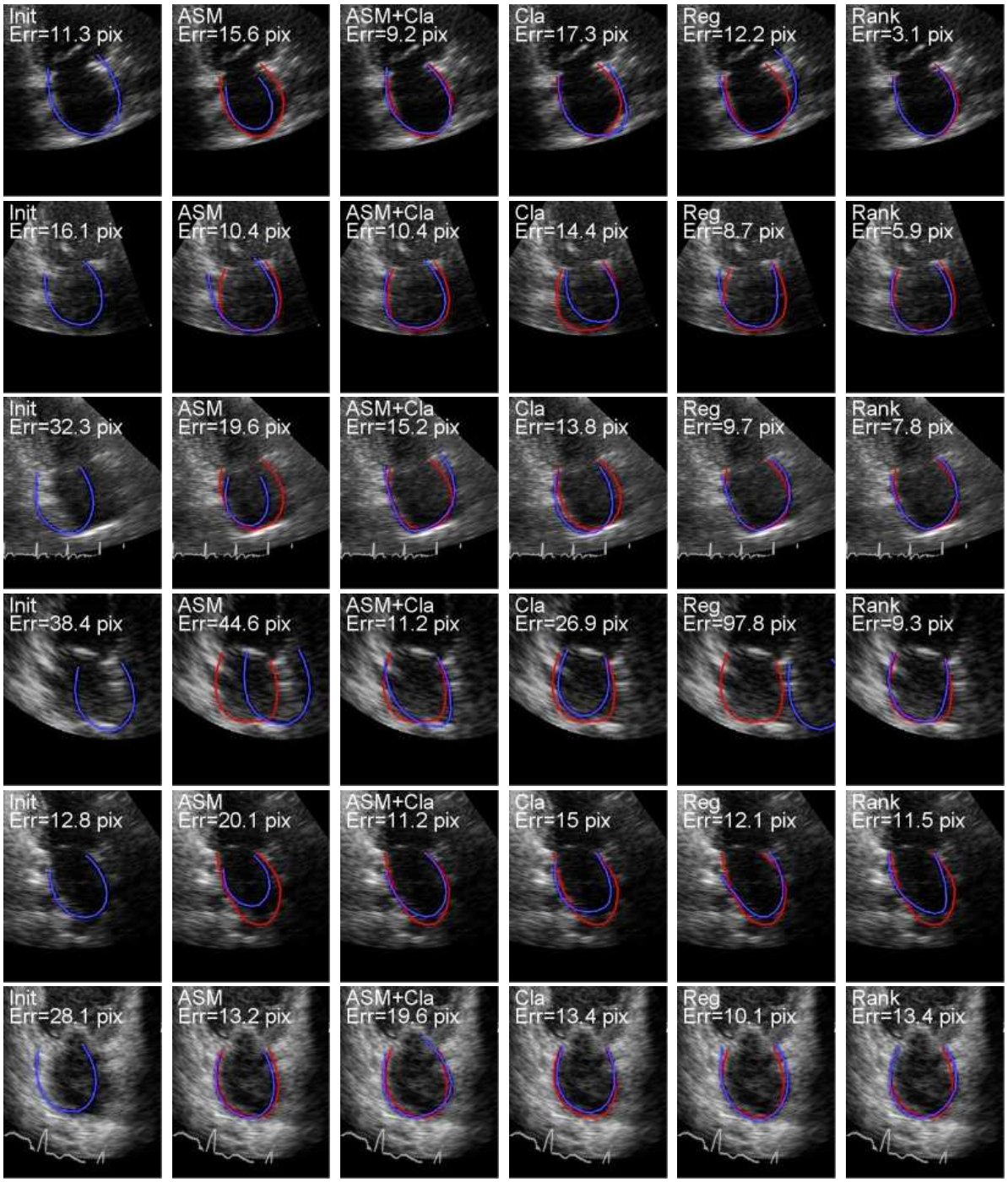



Figure 5.6: Some results of left atrium segmentation. For each row, the left image shows the initial contour, and the subsequent images show the segmentation results obtained by ASM, enhanced ASM, the classification, the regression, and the ranking approach. The initial contours and the segmentation results are blue lines. The ground truths are red lines.

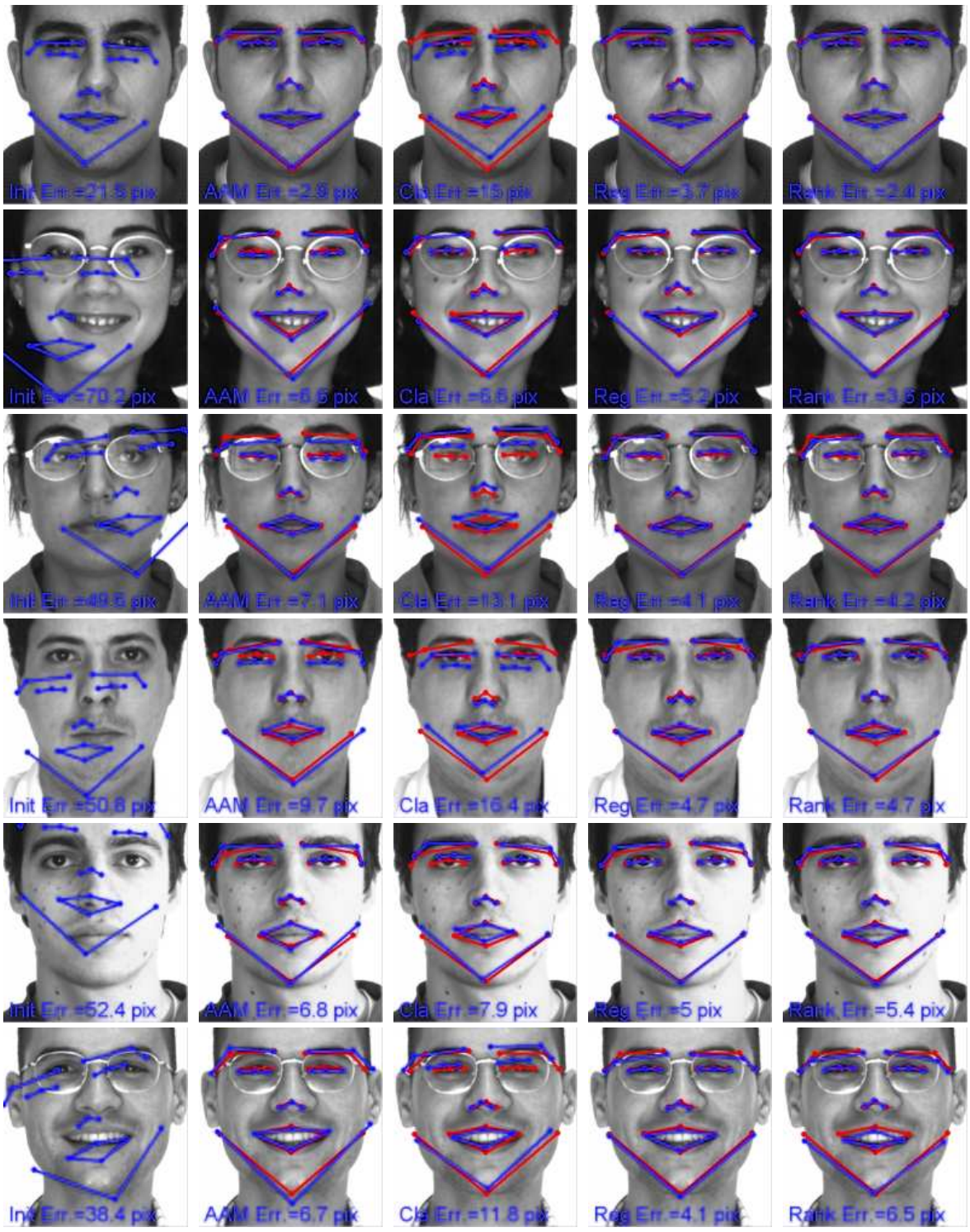



Figure 5.7: Some results of facial feature localization. For each row, the left image shows the initial locations, and the subsequent images show the localization results obtained by AAM, the classification, the regression, and the ranking approach. The initial locations and the localization results are drawn in blue. The ground truths are drawn in red.

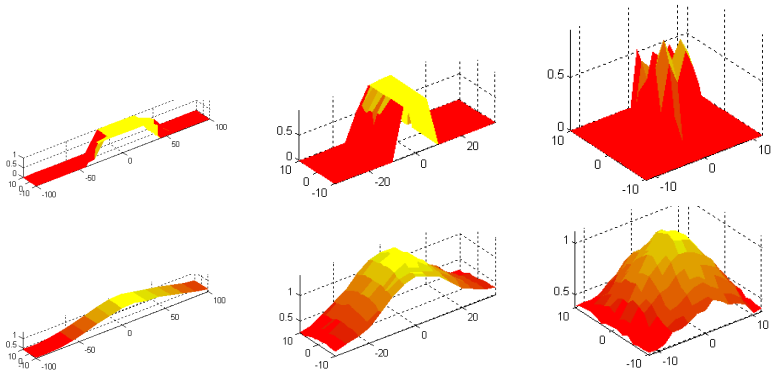


Figure 5.8: The 2D slices of the learned classifiers (top row) and ranking functions (bottom row) on a test image. The left column shows the first level, the middle shows the second level, and the right shows the third level.

image. I used half of the data for training and half for testing. Examples of the same subject were not used in both training and test data. The initial range of the pose is $[100, 100, \pi/9, 0.2]$. The model $C$ includes five shape parameters, which account for 73% of the total shape variation. I trained three levels of discriminative functions. The average initial error is 47.19 pixels in testing. I used AAM [73] for comparison. Figure 5.7 shows some facial feature localization results. The benchmarks are shown in Table 5.1(C) and Figure 5.4(C). Figure 5.4(D) shows the errors after each level of refinement when using the ranking functions in testing. Section 5.4.4 presents the discussions of the experimental results.

Figure 5.8 shows the 2D slices of learned classifiers and ranking functions on a test image. These slices are obtained by varying the first and the fifth parameters of the model in the error range while fixing the remaining parameters at the ground truth, where the first is a translation parameter and the fifth is a shape parameter corresponding to the largest eigenvalue. The learned functions have the desired shapes, which make the optimization algorithms perform well.

### 5.4.4 Discussion

The segmentation algorithms using discriminative fitting functions consistently outperform ASM and AAM by a large margin in the three experiments. The performance of ASM is improved by using discriminative boundary classifiers; however, it still falls into the local extremes because the boundary classifier is local. For the three discriminative learning approaches, the classification approach has relative poor performance due to its coarse search grid in the exhaustive search. If I use a fine search grid, the segmentation accuracy is expected to improve. It is interesting to see the performance of the algorithms specifically designed to train classifiers in a high-dimensional model space, such as marginal space learning [95]. The ranking approach converges to the correct solution faster than the regression approach, as indicated in the benchmarks of the first-level and the second-level refinement. The main reason might be that ranking only attempts to learn a partial ordering information in the model space, and hence its learning complexity is lower than regression. The learned ranking functions are more effective to guide the search algorithm to the correct solution. Verifying this conjecture is left to future work. Like all discriminative learning problems, the discriminative learning approaches suffer from the problem of overfitting, especially when the variation of training data does not cover the full variability. Furthermore, the number of sampled data points is hardly sufficient when the dimension of the model space is high. Because of these problems, the fitting function does not have the desired shape on some test data, and the local optimization algorithm fails to converge to the ground truth.

Recently, a ranking-based algorithm for face alignment was independently proposed [89]. It presents a ranking approach to learning an alignment score function, and compares the approach with a classification-based algorithm [47]. Compared with the method in [89], I used a different ranking algorithm and applied the multilevel approach to improve the segmentation accuracy. I also compared more algorithms on different kinds of data.

## 5.5 Summary

I have shown that all three discriminative methods – classification, regression, and ranking – can be applied for deformable shape segmentation. I have addressed how to sample a high-dimensional space for the classification and the ranking approach. I have also proposed a ranking algorithm based on Rankboost to learn a fitting function. The performance of the three discriminative learning approaches have been compared in the experiments on segmentation of the left ventricle and left atrium from ultrasound images and on localization of facial features. It has been demonstrated that the discriminative models outperform generative models and energy minimization methods by a large margin.

# Chapter 6

# Conclusion

I have proposed a series of learning-based object detection and segmentation algorithms to address my central thesis:

> Boosting principles enable a new class of algorithms for fast and accurate detection and segmentation of deformable objects by learning discriminative models from image databases with expert annotations.

In this chapter I begin to review the list of contributions presented in the first chapter. I then discuss future research possibilities and potential application areas.

## 6.1 Review of contributions

In this section I summarize the contributions of this dissertation. Each contribution is presented along with a discussion on how it was accomplished.

1. *I showed how to use boosting to construct a variety of discriminative models.*

   I first reviewed boosting in Chapter 2. I discussed the basic idea of boosting: building a strong learner from a set of weak learners by iteratively adding the weak learners to the strong learner. I explained why boosting is an effective way to select and combine weak models into a strong model. I further reviewed the computation

procedure of boosting and discussed the relationship between boosting and forward stagewise additive modeling.

I then developed a series of techniques to demonstrate that boosting is a general principle for designing learning based algorithms for object detection and segmentation applications. I showed that boosting is particularly useful for such applications, where a large number of weak learners can be obtained by associating each weak learner to an image feature. For all the algorithms proposed in the dissertation, I used Haar-like features as inputs of weak learners. These Haar-like features can be computed efficiently using the integral images. Although each Haar-like feature only gives weak representation of an objects' appearance, boosting combines hundreds of them to provide a robust representation.

I applied a variety of boosting algorithms in this dissertation. In Chapter 3 I used two types of boosted classifiers to construct the probabilistic boosting network: LogitBoost for pose classification and Adaboost for foreground/background classification. In Chapter 4 I used boosted regression to learn conditional density functions for deformable shape segmentation. In Chapter 5 I extended the idea developed in Chapter 4 to use both boosted ranking and Adaboost for learning fitting functions.

2. *I proposed a network structure, named the Probabilistic Boosting Network(PBN), for object detection in real time in both 2D and 3D.*

   In Section 2.2 I reviewed the learning based object detection algorithms. I also discussed several previous approaches for detecting objects with pose variation. In order to achieve real-time detection speed, these approaches only tested a sparse set of poses, which cannot meet an expected detection accuracy in medical applications.

   In Chapter 3 I first depicted a boosting-based multiclass procedure that estimates

an object pose more accurately. I discussed three approaches to combine several multiclass classifiers for estimating a pose with multiple parameters. I then presented a graph-structured PBN for joint object detection and pose estimation. Coupled with a pose-invariant pre-filter, the graph-structured PBN becomes an effective tool that quickly rejects negatives. In experiments I built PBNs that detect objects from 3D volumes and 2D images in real time.

3. *I formulated deformable shape segmentation as a discriminative learning problem and applied three discriminative methods, classification, regression, and ranking, to solve the problem.*

   I first compared generative models and discriminative models in Section 2.1.1. The main conclusion is that a discriminative model has better accuracy than a generative model when presented with sufficient training data. Based on this, I justified the purpose of this dissertation: applying discriminative learning to object detection and segmentation.

   I then pointed out that a key step of a deformable shape segmentation algorithm is to build a fitting function in a model space, which can differentiate the correct shape model from its background based on the image appearance. Fitting functions can be trained either generatively or discriminatively.

   In Chapter 4 I treated the conditional density function, $p(C|I)$, as a fitting function, which measures the fitting probability of a hypothesis shape model $C$ in a given image $I$. I used regression to learn $p(C|I)$ to approximate the ideal function $q(C|I)$, which is smooth and unimodal, and then I invoked a standard optimization algorithm, simplex, to optimize $p(C|I)$ for segmentation. I compared the regression approach with the previous generative learning approaches, including the Active Shape Model (ASM) and the Active Appearance Model (AAM).

   In Chapter 5 I applied two other discriminative learning approaches, classification

and ranking, to learn fitting functions. I discussed the connections and differences among the three discriminative learning approaches. I also analyzed the learning complexity and the computational cost of these three approaches.

4. *I proposed a multilevel refinement approach to improve the segmentation performance.*

   When the dimension of a model space is high, it is difficult to learn a single fitting function that is valid across the whole model space for distinguishing the correct solution from its background. In Chapter 4 I proposed a multilevel refinement approach to learn a series of fitting functions, each of which is defined on a focus region instead of the whole model space. I arranged the focused regions to be a nested structure gradually shrinking to the ground-truth model, enabling the learned functions to guide the solution of optimization algorithms to increasingly converge to the correct solution. In Section 4.4.2 I discussed two ways to construct the nested structure: One is to shrink the initial focus region uniformly, and the other is to first shrink the dimensions with a large error range.

5. *I proposed sampling algorithms for classification, regression, and ranking.*

   One of the major challenges for discriminative learning is to collect sufficient training data to prevent overfitting. Based on the proposed nested focus regions, I proposed the sampling strategies for the three discriminative learning approaches. These strategies are designed to maximize the performance of the learning algorithms. For classification, I randomly sampled positive and negative training samples defined by the focus regions. For regression, I sampled training samples based on the gradient magnitude of the target function. And for ranking, I sampled ranking pairs on the rays starting from the ground truth.

6. *I compared and analyzed the performance of the proposed discriminate approaches using large databases.*

I collected a variety of data sets, including 2D echocardiogram images, 3D echocardiogram volumes, mid-sagittal MR images, face images, and road sign images. These data sets are come from different applications, and they are captured by different imaging modalities. All these data sets are annotated by experts. Each data set contains a large number of training images, usually several hundred, enabling the algorithms to be tested on a large scale.

I tested the proposed algorithms on the collected data and compared them with the best-known previous algorithms. I demonstrated that the proposed algorithms outperform the previous algorithms, and they can be applied to a wide range of object detection and segmentation applications.

## 6.2 Future work

The task of detecting and segmenting objects is challenging and involves factors that are often domain-specific. This dissertation only explores some general techniques of applying discriminative learning to handle these tasks. In order to use these techniques in real applications, the following issues are worth additional study.

1. *How to use less training data by incorporating more domain knowledge or by combining discriminative learning and generative learning.*

   The proposed detection and segmentation algorithms employ pure discriminative models, which require a large volume of training data (enough to cover all data variation) for building the models with the desired accuracy. These algorithms demand a tremendous amount of data and need hundreds, if not thousands, of training images. Although the cost of capturing images continually decreases thanks to advances in imaging techniques, collecting a large training data set is still a burden for two reasons. First, manually annotating objects is tedious and time-consuming. This problem is especially obvious in 3D annotation, where it

normally takes more than 10 minutes to annotate an object with complex shape in a 3D volume (e.g., annotating liver in 3D CT volumes). Second, it is difficult to collect a large training data set for some objects. For example, when applying the proposed techniques to tumor segmentation, it is difficult to find enough training data because tumor growth is highly variable and rare.

As a result of these problems, it is worthwhile to study how to learn models using less training data. This might be achieved through two approaches. First, incorporate domain knowledge from experts into the training process. The proposed algorithms totally rely on machine learning to extract knowledge, requiring a large training data set to ensure that the learned knowledge is statistically meaningful. It is desirable to design learning algorithms which can incorporate knowledge from experts to reduce necessary training examples. Second, build hybrid models by combining generative learning and discriminative learning. As reviewed in Section 2.1, the hybrid models can achieve the same level of performance as discriminative models while using less training data. It is interesting to apply these models to object detection and segmentation.

2. *How to consider errors in expert annotations.*

   In this dissertation the expert annotations are served as ground truth in both training models and evaluating the trained models. The proposed algorithms assume that the annotations are consistent and accurate. However, this assumption seldom holds in real applications. Experts do not always annotate objects accurately, due to image noise and difficulty in navigating 3D volumes. Also, an expert might have his or her own bias in annotating objects, causing inconsistency among annotations from different experts. These annotation errors not only decrease the predication accuracy of the trained models, but also make the model evaluation questionable. It is necessary to understand how annotation errors affect training

and evaluation, and to further develop algorithms robust to such errors.

3. *How to apply the segmentation algorithms to other deformable shape models.*

   I used the Point Distribution Model (PDM) to demonstrate the strengths of discriminative learning. However, the proposed algorithms do not depend on a specific shape model, and other deformable shape models can be applied, such as the m-rep [59].

4. *How to combine a PBN and the proposed segmentation algorithms to a fully automatic object segmentation pipeline.*

   In this dissertation object detection and deformable object segmentation are studied separately. Detection and segmentation should be put together to form a fully automatic pipeline in real applications. The performance of detection and segmentation need be considered jointly in this situation.

5. *How to improve the computational efficiency of the segmentation algorithms.*

   The segmentation time of the proposed algorithm is several seconds. Although the speed is enough for off-line segmentation applications, it is not sufficient for online segmentation applications. Furthermore, when extending the algorithms to 3D, the segmentation speed is expected to be even slower. It is desirable develop schemes to accelerate the algorithms.

## 6.3 Closing Remarks

In this dissertation I have developed a series of discriminative learning algorithms for object detection and segmentation. The learned models are constrained to have certain properties that improve the performance of detection and segmentation, leading to fast and accurate solutions. To handle the complexity of discriminative learning, I have used boosting, which is a general learning principle used to combine weak models

into a stronger model, as a basis to develop the algorithms. I have demonstrated the performance of the proposed algorithms using a variety of data sets. By combining the algorithms proposed in this dissertation, a fully automatic object detection and segmentation pipeline can be built, which might hold great potential value for many computer vision and medical imaging applications.

# Bibliography


[1] M. A. Arbib, editor. *The Handbook of Brain Theory and Neural Networks*. MIT Press, 2002. 12

[2] V. Athitsos, J. Alon, and G. Kollios S. Sclaroff. Boostmap: A method for efficient approximate similarity rankings. In *Proc. CVPR*, 2004. 87

[3] A. Barbu, V. Athitsos, B. Georgescu, S. Boehm, P. Durlak, and D. Comaniciu. Hierarchical learning of curves application to guidewire localization in fluoroscopy. In *CVPR*, pages 1–8, 2007. 18

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965, 1989. 16

[5] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998. 16

[6] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986. 23

[7] T. Choudhury, J.M. Rehg, V. Pavlovic, and A. Pentland. Boosting and structure learning in dynamic bayesian networks for audio-visual speaker detection. In *Proc. ICPR*, 2002. 44

[8] G.E. Christensen, R.D. Rabbitt, and M.I. Miller. Deformable templates using large-deformation kinematics. *IEEE Transactions on Image Processing*, 5(10):1435–1447, 1996. 28

[9] P. Comon. Independent component analysis, a new concept? *Signal Process.*, 36(3):287–314, 1994. 11

[10] T. Cootes and C. Taylor. Statistical models for appearance for computer vision. In *unpublished manuscript*. 60, 78

[11] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. *IEEE Trans. PAMI*, 23(6):681–685, 2001. 4, 27, 28, 29, 31, 55, 63, 64, 71, 75, 94

[12] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models–their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995. 4, 27, 28, 31, 55, 60, 63, 64, 71, 75, 94

[13] H. Copas. Regression, and prediction, and shrinkage. In *J. R. Statist. Soc. B*, volume 45, pages 311–354, 1983. 67

[14] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 1. New York: Interscience, 1953. 29

[15] F. C. Crow. Summed-area tables for texture mapping. *SIGGRAPH Computer Graphics*, 18(3):207–212, 1984. 25

[16] R. H. Davies, T. F. Cootes, and C. J. Taylor. A minimum description length approach to statistical shape modelling. In *IPMI '01: Proceedings of the 17th International Conference on Information Processing in Medical Imaging*, pages 50–63, 2001. 28, 59

[17] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Journal of Machine Learning*, 46:225–254, 2002. 15

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. 4, 11

[19] C. Domingo and O. Watanabe. Madaboost: A modification of adaboost. In *COLT '00: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 180–189, 2000. 15

[20] H. Drucker and C. Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems 8*, pages 479–485, 1996. 16

[21] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. 12

[22] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936. 12

[23] P.T. Fletcher, C. Lu, S.M. Pizer, and S.C. Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, 23(8):995–1005, 2004. 29

[24] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991. 23

[25] Y. Freund. Boosting a weak learning algorithm by majority. In *COLT '90: Proceedings of the third annual workshop on Computational learning theory*, pages 202–216, 1990. 15

[26] Y. Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, 2001. 15

[27] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Machine Learning Research*, 4(6):933–970, 2004. 87, 91, 93

[28] Y. Freund and R. Schapire. A decision-theoretic generalization of online leaning and an application to boosting. *J. Computer and System Sciences*, 55(1):119–139, 1997. 15, 43

[29] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Ann. Statist.*, 28:337–407, 2000. 12, 15, 18, 38, 39, 43, 68

[30] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001. 18, 67

[31] B. Georgescu, X. S. Zhou, D. Comaniciu, and A. Gupta. Database-guided segmentation of anatomical structures with complex appearance. In *Proc. CVPR*, 2005. 18, 22, 31, 48, 49, 51, 52, 68, 73, 81, 83

[32] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York, 2001. 5, 16, 18, 68

[33] D. W. Hosmer and S. Lemeshow. *Applied Logistic Regression (2nd Edition)*. Wiley-Interscience, 2000. 12

[34] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *Proc. ICCV*, 2005. 22, 23, 34, 37

[35] T. Jebara and A. Pentland. Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 494–500, 1999. 12

[36] Y. Jing, V. Pavlovic, and J.M. Rehg. Efficient discriminative learning of bayesian network classifier via boosted augmented naive bayes. In *Proc. ICML*, 2005. 44

[37] M. Jones and P. Viola. Fast multi-view face detection. *MERL-TR2003-96*, July 2003. 22, 23, 34, 37, 49, 52

[38] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Computer Vision*, 1(4):321–331, 1988. 27, 28, 29, 55, 56, 71

[39] A. Khammari, F. Nashashibi, Y. Abramson, and C. Laurgeau. Vehicle detection combining gradient analysis and adaboost classification. In *Intelligent Transportation Systems, 2005. IEEE Proceedings*, pages 66–71, 2005. 22

[40] R. Kindermann and J. L. Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, 1980. 11

[41] J. Lafferty, K. Nigam, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999. 12

[42] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001. 12

[43] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *In Proceedings of the tenth national conference on artificial intelligence*, pages 223–228, 1992. 4

[44] J. A. Lasserre, C. M. Bishop, and T. P. Minka. Principled hybrids of generative and discriminative models. In *Proc CVPR*, pages 87–94, 2006. 13

[45] S. Li and Z. Zhang. FloatBoost learning and statistical face detection. *PAMI*, 26:1112–1123, 2004. 22, 23, 34

[46] C. Liu and H. Y. Shum. Kullback-leibler boosting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 587–594, 2003. 22

[47] X. Liu. Generic face alignment using boosted appearance model. In *Proc. CVPR*, 2008. 102

[48] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, pages 1150–1157, 1999. 24

[49] A. M. Martinez and R. Benavente. The AR face database. 1998. 81, 98

[50] S. Menet, P. Saint-Marc, and G Medioni. B-snakes: Implementation and application to stereo. In *DARPA Image Understanding Workshop*, pages 720–726, 1990. 27

[51] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6), 1953. 73

[52] R. Miller. On-road vehicle detection: A review. *IEEE Transactions ON Pattern Analysis and Machine Intelligence*, 28(5):694–711, 2006. 18

[53] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proceedings of Neural Information Processing Systems*, volume 13, 2001. 12

[54] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computation Physics*, 79:1249, 1988. 30

[55] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *CVPR*, pages 130–136, 1997. 20

[56] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901. 4, 11

[57] D. L. Pham, C. Xu, and J. L. Prince. Current methods in medical image segmentation. *Annual Review of Biomedical Engineering*, 2:315–338, 2000. 26, 27

[58] S. M. Pizer, R. E. Broadhurst, J. Levy, X. Liu, J. Y. Jeong, J. Stough, G. Tracton, and E. L. Chaney. Segmentation by posterior optimization of m-reps: Strategy and results. In *unpublished manuscript*, 2007. 30

[59] S. M. Pizer, P. T. Fletcher, S. Joshi, A. Thall, J. Z. Chen, Y. Fridman, D. S. Fritsch, A. G. Gash, J. M. Glotzer, M. R. Jiroutek, C. Lu, K. E. Muller, G. Tracton, P. Yushkevich, and E. L. Chaney. Deformable m-reps for 3D medical image segmentation. *Int. J. Comput. Vision*, 55(2-3):85–106, 2003. 27, 28, 29, 55, 110

[60] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. Numerical recipes in C (2nd edition). Cambridge University Press, 1992. 75, 95

[61] P.Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 734–741, 2003. 18, 22

[62] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996. 16

[63] A. Y. Ng R. Raina, Y. Shen. Classification with hybrid generative/discriminative models. In *Advances in Neural Information Processing Systems 16*, 2003. 13

[64] T. R. Reed and J. M. Hans du Buf. A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image Understanding*, 57(3):359–372, 1993. 26

[65] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998. 19, 20

[66] D. Rueckert, A.F. Frangi, and J.A. Schnabel. Automatic construction of 3-D statistical deformation models of the brain using nonrigid registration. *IEEE Transactions on Medical Imaging*, 22(8):1014–1025, 2003. 29

[67] R.Xiao, L. Zhu, and H. J. Zhang. Boosting chain learning for object detection. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 709–715, 2003. 22

[68] F.S. Samaria. Face recognition using hidden markov models. 1994. 20

[69] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998. 16

[70] R.E. Schapire. The strength of weak learnability. *Journal of Machine Learning*, 5(2):197–227, 1990. 15

[71] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 45–51, 1998. 20

[72] M. B. Skouson and M. E. Kowalski. A simple and efficient method for image segmentation with deformable templates. In *SSIAI '00: Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation*, page 31, 2000. 28

[73] M. B. Stegmann, B. K. Ersboll, and R. Larsen. FAME–a flexible appearance modeling environment. *IEEE Trans. Medical Imaging*, 22(10):1319–1331, 2003. 81, 101

[74] G. Székely, A. Kelemen, C. Brechbuehler, and G. Gerig. Segmentation of 2D and 3D objects from MRI volume data using constrained elastic deformations of flexible fourier contour and surface models. *MEDIA*, 1(1):19–34, 1996. 77

[75] G. Székely, A. Kelemen, C. Brechbühler, and G. Gerig. Segmentation of 3D objects from MRI volume data using constrained elastic deformations of flexible fourier surface models. In *CVRMed '95: Proceedings of the First International Conference on Computer Vision, Virtual Reality and Robotics in Medicine*, pages 495–505, 1995. 27

[76] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models for 3D object reconstruction. pages 269–276, 1987. 27

[77] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 762–769, 2004. 22

[78] Z. Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *Proc. ICCV*, 2005. 43

[79] Z. Tu. Learning generative models via discriminative approaches. In *Proc CVPR*, pages 1–8, 2007. 13

[80] Z. Tu, X. S. Zhou, A. Barbu, L. Bogoni, and D. Comaniciu. Probabilistic 3D polyp detection in CT images: The role of sample alignment. In *Proc. CVPR*, 2006. 48

[81] Z. Tu, X. S. Zhou, D. Comaniciu, and B. Luca. A learning based approach for 3D segmentation and colon detagging. In *Proc. European Conf. Computer Vision*, 2006. 31, 94

[82] I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In *Proc. CVPR*, pages 258–265, 2005. 13

[83] I. Ulusoy and C. M. Bishop. Comparison of generative and discriminative techniques for object detection and classification. In *Proc. Sicily Workshop on Object Recognition*, 2006. 37

[84] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. 5, 12, 16

[85] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998. 12

[86] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, 2001. 20, 21, 22, 24, 33, 34, 43, 48, 68, 94

[87] M.K. Warmuth, J.Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 1001–1008, 2006. 15

[88] B. Wu, H. AI, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *Proc. Auto. Face and Gesture Recognition*, 2004. 22, 34

[89] H. Wu, X. Liu, and G. Doretto. Face alignment via boosted ranking model. In *Proc. CVPR*, 2008. 102

[90] C. Xu, D. L. Pham, and J. L. Prince. Medical image segmentation using deformable models. *Handbook of Medical Imaging – Volume 2:Medical Image Processing and Analysis*, pages 129–174, 2000. 26

[91] M. H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002. 18, 19

[92] J. Zhang, S.K. Zhou, D. Comaniciu, and L. McMillan. Conditional density learning via regression with application to deformable shape segmentation. In *Proc. CVPR*, 2008. 56

[93] J. Zhang, S.K. Zhou, D. Comaniciu, and L. McMillan. Discriminative learning for deformable shape segmentation: A comparative study. In *Proc. ECCV*, 2008. 84

[94] J. Zhang, S.K. Zhou, L. McMillan, and D. Comaniciu. Joint real-time object detection and pose estimation using probabilistic boosting network. In *Proc. CVPR*, 2007. 35, 68

[95] Y. Zheng, A. Barbu, B. Georgescu, M. Scheuering, and D. Comaniciu. Fast automatic heart chamber segmentation from 3D CT data using marginal space learning and steerable features. In *Proc. ICCV*, 2007. 18, 31, 94, 102

[96] Y. Zheng, X. S. Zhou, B. Georgescu, S.K. Zhou, and D. Comaniciu. Example based non-rigid shape detection. In *Proc. European Conf. Computer Vision*, 2006. 31, 64, 81, 83, 87

[97] S. Zhou, B. Georgescu, X. Sean Zhou, and D. Comaniciu. Image based regression using boosting method. In *Proc. ICCV*, 2005. 68

[98] S. K. Zhou, J. H. Park, B. Georgescu, C. Simopoulos, J. Otsuki, and D. Comaniciu. Image-based multiclass boosting and echocardiographic view classification. In *Proc. CVPR*, 2006. 38, 39

[99] S.K. Zhou and D. Comaniciu. Shape regression machine. In *Proc. IPMI*, 2007. 31, 68