Probabilistic Procrustean Models for Shape Recognition with an Application to Robotic Grasping

by

Jared Marshall Glover

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

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author .....
Department of Electrical Engineering and Computer Science
February 15, 2008

Certified by.....
Daniela Rus
Professor
Thesis Supervisor

Certified by.
Nicholas Roy
Assistant Professor
Thesis Supervisor

Accepted by....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Probabilistic Procrustean Models for Shape Recognition with an Application to Robotic Grasping

by

Jared Marshall Glover

Submitted to the Department of Electrical Engineering and Computer Science
on February 15, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Robot manipulators largely rely on complete knowledge of object geometry in order to plan their motion and compute successful grasps. If an object is fully in view, the object geometry can be inferred from sensor data and a grasp computed directly. If the object is occluded by other entities in the scene, manipulations based on the visible part of the object may fail; to compensate, object recognition is often used to identify the location of the object and compute the grasp from a prior model. However, new instances of a known class of objects may vary from the prior model, and known objects may appear in novel configurations if they are not perfectly rigid. As a result, manipulation can pose a substantial modeling challenge when objects are not fully in view.

In this thesis, we will attempt to model the shapes of objects in a way that is robust to both deformations and occlusions. In addition, we will develop a model that allows us to recover the hidden parts of occluded objects (shape completion), and which maintains information about the object boundary for use in robotic grasp planning. Our approach will be data-driven and generative, and we will base our probabilistic models on Kendall's Procrustean theory of shape.

Thesis Supervisor: Daniela Rus
Title: Professor

Thesis Supervisor: Nicholas Roy
Title: Assistant Professor

Acknowledgments

First and foremost, I would like to thank my collaborators on this work: Kristian Kersting, whose machine learning prowess helped iron out the discriminative algorithms of chapter 5 and who was always full of new ideas; Christian Pedersen and Erik Taarnhøj, the upbeat Danish exchange students who contributed to the correspondence algorithms of chapter 3 and who made my Advanced Algorithms class a heck of a lot more fun than it would have been otherwise; Geoff Gordon from Carnegie Mellon University who helped introduce me to Kendall's shape space; and of course my tireless advisors, Nicholas Roy and Daniela Rus, who put countless hours of editing work into everything I wrote, and who pushed me to achieve much more than I ever would have achieved on my own.

I have greatly benefited from the many long discussions I have had with all of my fellow lab-mates, whose presence and mutual commiseration made the many long nights much more bearable. I must also thank Christian Plagemann, who first gave me the idea to try using local shape parts to prune the classification search. I am also grateful to the Linden Earth System Fellowship, which supported me in my first two semesters at MIT.

Contents

Chapter 1

Introduction

Shape modeling and analysis arises in many different disciplines, with numerous applications ranging from computer graphics and robotics to archeology and medicine. The specific problems one encounters in each field are as varied as they are important.

Consider a service robot in a restaurant which is tasked with unloading a dishwasher. In order to know where to put dishes it retrieves, the robot must be able to recognize each dish as an instance of a specific dish type (e.g. plate, fork, cup). This problem is known as *object recognition.* Furthermore, to plan the sequence of grasping actions it must perform in order to transport the dishes from the dishwasher to the cupboard, the robot must infer any hidden parts of objects which are not fully in view because they are occluded by other dishes or parts of the dishwasher. This is the problem of *object completion.* A dishwasher can be a particularly cluttered environment, which may make the resulting inference problems more challenging. Thus, it is imperative to leverage the information that is present in the environment to perform these tasks as quickly and as accurately as possible. Unfortunately, we cannot use the color or patterns on a dish to figure out what kind of an object it is. Many dishes are very uniform in color, with no discernible markings, while others have complicated patterns, but one cannot say "all red dishes are cups", or "all dishes with a striped pattern on them are plates". Instead, we must rely on the geometry of the dishes, in order to classify them and plan grasping actions.

In medicine, an AI system may be responsible for examining a CAT scan or an

X-ray in order to screen a patient for heart problems, or for determining if a tumor is likely to be cancerous. The AI must first determine which part of the image corresponds to the organ of interest, and which part corresponds to the surrounding tissue, or background. This problem is called *image segmentation*, and can be greatly aided by prior models of object shape. Once the object is segmented out of the image, the system must infer the probability of cancer, or of abnormal heart function, based on the appearance and shape of the object in one or more images. It can then display this information to a doctor, who can decide whether or not to perform surgery.

No one shape modeling technique will be able to answer all questions in every domain. However, within a specific domain, one may formulate a set of guidelines which roughly capture the properties any shape modeling method should have in order to be most useful to problems in that domain. In computer graphics, these properties may be things like ease of use in calculating lighting effects and collisions, ability to represent scanned objects from the real world, and smoothness/resolution. In robotics and computer vision, on the other hand, we care mostly about accuracy and the ability to perform complicated inference tasks, such as shape recognition ("what is it?") or shape completion ("what does the hidden part of that object look like?").

In robotics, we also want to use these models to accomplish practical tasks such as navigation and object manipulation, which require action-effect estimation (e.g. "how likely is it that I will drop this object if I place my fingers here, here, and here?" or "what is the probability that I can fit through that small opening in the wall?"). Both of these questions would be trivial (dynamics aside) in a world with no uncertainty, such as inside a computer game. However, in the real world we must deal constantly with uncertainty in not only sensing, but also in modeling and prediction.

In this thesis much of the discussion and methodology will be motivated by problems encountered in robotics–specifically, grasp planning with incomplete observations; however, it should be noted that the shape modeling framework and inference capabilities presented here are applicable to a wide range of fields.

1.1 Background

The ability to easily infer the geometry of the world around us is something that humans often take for granted, yet some of the most challenging problems in robotics and computer vision in recent years have dealt with precisely these issues. In mobile robotics, the problem of simultaneous localization and mapping (SLAM) has kept researchers busy for decades, while the corresponding problem in computer vision–known as structure from motion (SFM)–has also posed quite a challenge to scientists in the field. In each domain, the goal is to accurately estimate the geometry of the environment around the robot (or camera) from noisy sensor data, while at the same time recovering the trajectory of the robot (camera) through the world. In computer vision, the problem is compounded by the sheer volume of sensory data which must be processed. In both the SLAM and SFM communities, recent breakthrough techniques have increasingly used statistical tools to handle the errors in sensing and estimation [58, 10]. By modeling not only the best estimate of the world geometry, but also the covariances of those estimates, robust solutions to SLAM and SFM are becoming more attainable.

Neither SLAM nor SFM are considered to be shape modeling problems in the traditional sense, as they deal with the geometry of whole environments, not individual objects. However, we believe that the lessons from these two problems can and should be applied to modeling the shapes of objects.

In the computer vision community, a great deal of progress has been made in recent years on many shape recognition tasks, from retrieval to classification to segmentation. Several state-of-the-art approaches employ general-purpose shape descriptors (such as "shape contexts") that transform the original shape geometry into a more discriminative feature space and apply nearest-neighbor classification in this new feature space [2, 34, 50].

Although results of these state-of-the-art methods are impressive, there are two types of problems on which these discriminative approaches typically fail. The first is when the shape of objects within a shape class varies in a non-trivial way (for

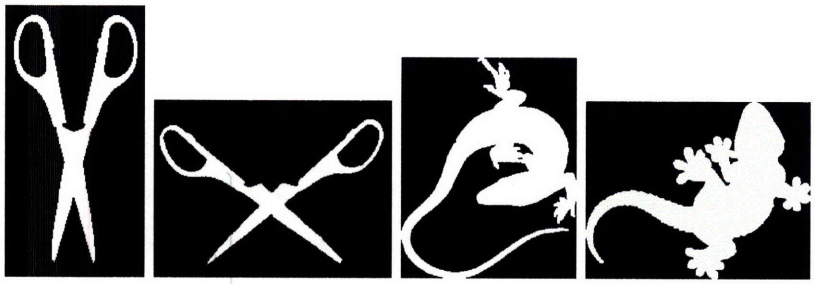


Figure 1-1: Shape classes with high variance, due to articulation (left) and biology (right). It is very difficult to construct a single shape metric which will enable robust, accurate discrimination of shape categories for classes such as these.

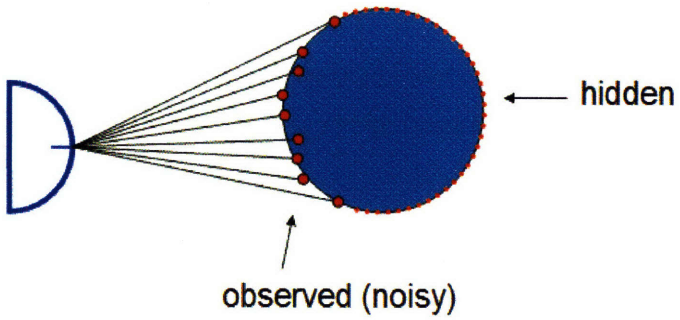
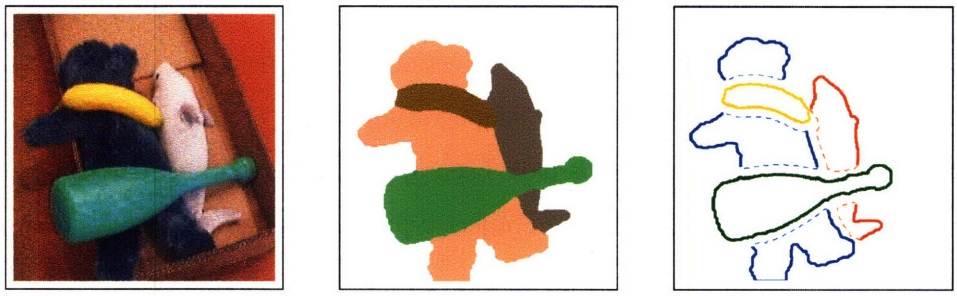


Figure 1-2: Boundary estimation. We want to predict the spatial location of every point on an object's boundary (both hidden and observed).

example, in an articulated object or an animal silhouette, as in Figure 1-1). In these cases, the same type of object can look very different depending on the instance, and either the shape distance must be modified in order to take these deformations into account[1], or multiple instances of the same object class must be stored in the shape database in order to encompass the full spectrum of variability within the class.

The second type of problem on which these discriminative approaches to shape analysis can fail is when we have incomplete observations of objects; either because of the limits of the sensor's field of view or because of object occlusions, as in Figure 1-3. For applications such as robotic grasping, we also want to perform additional inference

[1]It should be noted that Ling et. al's "inner distance" does do a very good job of modifying the shape context descriptor to handle articulated objects [34].



(a) Original image (b) Segmentation (c) Contours

Figure 1-3: (a) A collection of toys in a box. The toys partially occlude each other, making object identification difficult. (b) A segmentation of the image, based on color. (c) Partial object contours which must be used to perform geometric object recognition.

tasks other than shape recognition, such as shape completion or boundary estimation (Figure 1-2). In these cases, it is not enough to just recognize what an object is–we must also infer the correct geometry of the new object given the shape class it belongs to (and given a partial and/or noisy estimate of the new shape's geometry).

In this thesis, we will attempt to solve both of these canonical challenges–shape variation and occlusions–at the same time. Although the resulting inference problems are quite complex, we feel that handling these types of uncertainties is one of the key challenges which must be solved if we are to develop AI systems capable of operating in the real world.

In the classical computer vision literature, there is considerable work on recognizing occluded objects, e.g., [33, 28, 21]. Recognizing and localizing occluded objects when the objects are rigid is known as the "bin-of-parts" or "bin-picking" problem. Despite being described by early vision researchers as the most difficult problem in automatic assembly [20], there were many successful systems developed in the 1980's which solved the bin-of-parts problem in controlled environments. Most systems used greedy, heuristic search in a "guess-and-test" framework. Thus, in order to scale well with the number of object classes (and the number of objects in the image) they needed to be *especially* greedy, pruning away as much of the search space as possible to avoid an exponential running time. As a result, these approaches were especially

sensitive to variations in shape.

An explosion of interest in object detection and segmentation in recent years has led to many advances in modeling shape variability [9, 8, 15, 4]. However, most of these shape deformation models have been applied in constrained environments, detecting only a handful of prescribed object types–for example in medical imaging [38] or face detection [8].

1.2 Our Approach

We wish to model the shapes of objects in a way that is robust to both deformations and occlusions. In addition, we desire a model that allows us to recover the hidden parts of occluded objects (shape completion), and which maintains information about the object boundary for use in robotic grasp planning. Our approach can be summarized as follows:

1. **Dense:** We model an object's shape with a dense set of points along the object's boundary. This provides a level of redundancy and completeness which makes our model robust to many types of sensing noise (such as a poor image segmentation), while at the same time providing all the necessary information for planning grasps or other actions based on full object geometry.

2. **Data-driven:** In order to model the complex shape classes of figure 1-1, it is our belief that no one discriminative shape metric can handle all possible types of variation. Thus, we seek models which can be *learned* from training data.

3. **Generative:** Although we do explore the use of discriminative methods as a pre-processing step in object detection and classification (Chapter 5), we primarily seek generative shape models, as we believe these are the most amenable to robustly performing complicated inference tasks, such as shape completion, as well as being the most accurate in describing the variability of a class of objects.

Note that these tenets are not necessarily the most natural ones for every domain. For example, in the so-called "shape google" application–known as shape retrieval–it may be more desirable that the shape similarity measure is generic and consistent, applying to the widest range of objects in a predictable fashion, than that it can be tailored to perform well on any given dataset. Or, if we do not care about the original geometry of the object, but we wish only to classify its shape, then we may not care whether or not we can accurately capture the variability of a shape model in a generative way–in fact, discriminative approaches will often perform better than generative ones in such cases.

However, for our application–robotic grasping–it is clear that modeling shape variation, boundary estimation/shape completion, and the ability to tailor the system to improve performance over time are all key components when it comes to being able to grasp deformable, occluded objects in real world environments. We also speculate that these core capabilities will be increasingly important to computer vision algorithms for tasks such as image segmentation and class-level object recognition, although additional work is needed to fully explore these domains.

1.3 Thesis Statement

Learning dense, probabilistic models of object shape allows for accurate geometric inference of partially hidden, deformable objects.

1.4 Contributions

In this thesis, we provide several novel contributions to the areas of statistical shape analysis and robotic grasping:

1. We adapt the popular Procrustean shape model to model the dense probabilistic geometry of object boundaries.

2. We describe a novel graphical model for finding correspondences between the

points of two shapes, and show that this correspondence model improves the modeling accuracy of probabilistic Procrustean shape models.

3. We derive a solution to the shape completion problem, and use our shape models to infer the hidden parts of partially-occluded, deformable objects.

4. We demonstrate how a discriminative model of shape parts, also based on the Procrustean framework, can be trained to detect both full and partially-occluded shapes, and we show improved performance on an object retrieval task on a popular shape dataset over all previously published methods.

5. We apply our shape inference algorithms to the task of robotic grasping, and show that learning probabilistic models of deformable objects allows us to efficiently recognize and retrieve complex objects from a pile of objects in the presence of sensor noise and occlusions.

1.5 Outline

We begin in chapter 2 by developing a suitable model for 2-D shape, and reviewing existing models in the literature. In chapter 3 we discuss the correspondence, or "shape matching" problem, which must be solved in order to align shapes prior to inference. Chapter 4 is devoted to the task of shape completion, presenting our method for inferring the missing parts of occluded objects. In chapter 5 we present our work on "boosted shape parts", which we use as a pruning heuristic for large-scale shape recognition problems in which it is computationally prohibitive to find the likelihoods of a given shape under every possible shape model using the full, generative methods of chapters 2-4. Finally, in chapter 6 we tie everything together with an application to robotic grasping.

Chapter 2

2-D Shape Models

2.1 Background

There are currently three major approaches to 2-D shape modeling. The first approach is to use a set of points, or a *point cloud* representation. With this model, shapes are represented as a collection of point locations in the plane. Beyond that, each point cloud model handles inference differently. The most common inference tasks are *classification* and *registration*. Shape classification is the task of classifying a new shape given a set of shape classes, such as "triangle", or "fork". Registration is the task of aligning a prior shape model with a shape in a new scene, which may be represented by a set of points from a laser range scan, or a set of edge fragments extracted from an image. In order to align a point cloud with a new scene, one must figure out which points in the model correspond to which elements of the scene. This is an example of a "data association", or "correspondence" problem, which is the focus of chapter 3. Once a point correspondence is found, many point cloud models then use either a Euclidean distance (in the simplest cases), or more commonly a *Procrustes distance* [27] (which is invariant to position, scale, and orientation) or a *bending energy* [7, 2] (required to warp one shape into the other), in order to compute a shape dissimilarity metric. Correspondences are computed either by estimating the best alignment and the point correspondences jointly, as in the Iterated Closest Point algorithm [3], or by forming a pairwise assignment cost function between each pair of

points across the two point clouds and using either bipartite matching [2] or relaxation labeling [44] to find a good matching. Alternatively, an example of work where point correspondences are not directly computed is [23], where a *Hausdorff distance* is computed in order to perform both matching and registration simultaneously.

The second major approach to 2-D shape modeling is to represent the shape of an object by a continuous curve. These models can then be subdivided into those which represent only closed curves (contours) and those which represent either closed or open curves. The majority of widely-used work in this area has been done on closed contours, as most of the physical objects people wish to model have 2-D extent (although handwriting is a good counter-example). Many popular closed contour models in recent work have used variational methods (e.g. active contours or level sets) to perform image segmentation and tracking tasks [4, 9, 24, 45]. The basic idea in both active contours (snakes) and level sets is to use an energy minimization to evolve an initial curve to fit image intensity data, where energy terms include smoothness and degree to which the contour fits the image intensities. More general curves (both open and closed) have been modeled by convex polygon parts [30], fourier descriptors [33], algebraic curves [54, 57], and turning functions [56], among others. Another closed curve model which deserves mention is curvature scale space [39], where shape is modeled by the singularities of a curve evolution process as the curve is smoothed by a Gaussian kernel. Latecki's convex polygon parts model [30] also models shape across curvature scales, and this general multi-scale approach is a motivating idea for some of the work done in this thesis.

The third approach to 2-D shape modeling is found in shape skeletonization. Shape skeletons are limited to modeling closed contours, and in many cases can be thought of as simply an approximate dual representation for the object boundary. The advantage to skeleton models is that they can present a hierarchical decomposition of shape (e.g. the spine to the limbs to the fingers and toes), and are widely used on both articulated shapes and on shapes with distinct parts. Some examples of skeleton-based shape models are medial axis [5], shock graphs [50, 52], and shape axis trees [17].

Several good survey articles of 2-D shape modeling exist. For a good start, see [51, 60, 59].

2.1.1 A Simple Model for 2-D Shape

In order to accomplish the thesis goals set out in section 1.2 of handling shape deformations and occlusions in a way which preserves the original object geometry, we wanted to keep the model simple, in order to make the inference procedures as straightforward as possible. Therefore, we chose to use the tried and true method of representing an object's shape by a set of discrete points around its boundary. The drawbacks to this model are obvious–we are restricted to closed contours, we cannot represent (but we can approximate) continuous curves, and as with point clouds we must solve a correspondence problem in order to compare shapes. However, the advantages are (i) it can be made as dense as needed, (ii) we have a natural cyclic ordering of points around the contour which aids us in finding correspondences between shapes, and (iii) we can draw upon a vast amount of established literature on the statistical shape modeling of sets of ordered landmarks.

The most influential pioneer of statistical shape modeling was David Kendall, who developed a rigorous mathematical understanding of the topology of shape spaces, which we discuss next.

2.2 Shape Space

Let us represent a shape $\mathbf{z}$ as a set of n points $\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_n$ in some Euclidean space. We will restrict ourselves to two-dimensional points (representing shapes in a plane) such that $\mathbf{z}_i = (x_i, y_i)$, although extensions to three dimensions are feasible. We will assume these points are ordered (so that $\mathbf{z}$ can be defined as a vector), and represent a closed contour (such as the letter "O", as opposed to the letter "V"). In general, the shape model can be made *de facto* invariant to point ordering if we know correspondences between the boundary points of any two shapes we wish to compare.

In this thesis, we will use the following two conventions for representing a shape

$\mathbf{z}$, mathematically. First, we define the *real shape vector* $\mathbf{z}_{re}$ as

$$\mathbf{z}_{re} = [x_1, y_1, x_2, y_2, \ldots, x_n, y_n]^T, \tag{2.1}$$

where the 2-D points of $\mathbf{z}$ are stacked on top of each other, resulting in a real vector of length $2n$. Second, we define the *complex shape vector* $\mathbf{z}_{co}$ as

$$\mathbf{z}_{co} = [x_1 + iy_1, x_2 + iy_2, \ldots, x_n + iy_n]^T, \tag{2.2}$$

where each point of $\mathbf{z}$ shows up as a single complex element of $\mathbf{z}_{co}$, so that $\mathbf{z}_{co}$ is a complex vector of length n. Throughout the remainder of this thesis, we will drop the '*re*' and '*co*' subscripts from our shape vector notation, and will simply denote a shape vector by $\mathbf{z}$. If it is not clear whether $\mathbf{z}$ is real or complex from the context of the equations in which $\mathbf{z}$ appears, it will be specified.

In order to make our model invariant to changes in position and scale, we can normalize the shape so as to have unit length with centroid at the origin; that is,

$$\mathbf{z}' = \{\mathbf{z}'_i = (x_i - \bar{x}, y_i - \bar{y})\} \tag{2.3}$$

$$\tau = \frac{\mathbf{z}'}{\|\mathbf{z}'\|} \tag{2.4}$$

where $\|\mathbf{z}'\|$ is the L^2-norm of the real shape vector $\mathbf{z}'$. We call τ the *pre-shape* of $\mathbf{z}$. Since τ is a unit vector, the space of all possible pre-shapes of n points is a unit hyper-sphere, $\mathbb{S}_*^{2n-3}$, called *pre-shape space*[1].

Any pre-shape is a point on the hypersphere, and all rotations of the shape lie on an orbit, $\mathcal{O}(\tau)$, of this hypersphere. Thus, *shape* is defined as the orbit of a pre-shape under 2-D rotation, and the *shape space* Σ_2^n is defined as the space of all such orbits of 2-D pre-shapes of n points. If we wish to compare shapes using some distance metric between them, the spherical geometry of pre-shape space requires a geodesic distance rather than Euclidean distance. Additionally, in order to ensure this distance

[1]Following [53], the star subscript is added to remind us that $\mathbb{S}_*^{2n-3}$ is embedded in $\mathbb{R}^{2n}$, not the usual $\mathbb{R}^{2n-2}$.

is invariant to rotation, we define the distance between two shapes $\mathbf{z}$ and $\mathbf{w}$ as the smallest distance between their orbits:

$$d_p[\mathbf{z}, \mathbf{w}] = \inf[d(\phi, \psi) : \phi \in \mathcal{O}(\tau_z), \psi \in \mathcal{O}(\tau_w)] \quad (2.5)$$

$$d(\phi, \psi) = \cos^{-1}(\phi \cdot \psi) \quad (2.6)$$

Kendall [26] defined d_p as the *Procrustean metric*, where $d(\phi, \psi)$ is the geodesic distance between ϕ and ψ. Since the inverse cosine function is monotonically decreasing over its domain, it is sufficient to maximize $\phi \cdot \psi$, which is equivalent to minimizing the sum of squared distances between corresponding points on ϕ and ψ (since ϕ and ψ are unit vectors). For every rotation of ϕ there exists a rotation of ψ which will find the global minimum geodesic distance. Thus, to find the minimum distance, we need only rotate one pre-shape while holding the other one fixed. We call the rotated ψ which achieves this optimum ($\theta_{\alpha^*}(\tau_w)$) the *orthogonal Procrustes fit* of τ_w onto τ_z, and the angle α^* is called the *Procrustes fit angle.*

Representing the points of τ_z and τ_w in complex coordinates, which naturally encode rotation in the plane by scalar complex multiplication, the Procrustes distance minimization can be solved:

$$d_p[\mathbf{z}, \mathbf{w}] = \cos^{-1} |\tau_w^H \tau_z| \quad (2.7)$$

$$\alpha^* = \arg(\tau_w^H \tau_z), \quad (2.8)$$

where τ_w^H is the *Hermitian*, or complex conjugate transpose of the complex vector τ_w.

2.3 Shape Inference

Given a set of measurements of an object class, we would like to infer a distribution of possible object geometry. We will first derive an expression for the mean shape, and then derive the distribution covariance. After describing the model learning procedure, we will present an algorithm for computing the likelihood of a new shape

under this model.

2.3.1 The Shape Distribution Mean

Let us assume that our training data set consists of a set of measurements $\{\mathbf{m}_1, \mathbf{m}_2, \ldots\}$ of objects drawn from the same shape class, where each measurement $\mathbf{m}_i$ is a complete (but noisy) description of object geometry, a shape vector of length $2n$. We can normalize each measurement to be an independent pre-shape and use these pre-shapes to compute an estimate of mean shape. If each measurement $\mathbf{m}_i$ is normalized to a pre-shape τ_i, then the *Procrustean mean shape* of $\{\tau_1, \ldots, \tau_M\}$ is

$$\mu^* = \arg \inf_{\|\mu\|=1} \sum_i [d_p(\tau_i, \mu)]^2 \tag{2.9}$$

The pre-shape μ^* is formally called the *Fréchet mean*[2] of the samples $\tau_1, \ldots, \tau_M$ with respect to the distance measure 'd_p'. Note that the mean shape is *not* trivially the arithmetic mean of all pre-shapes; d_p is non-Euclidean, and we wish to preserve the constraint that the mean shape vector has unit length. Intuitively, what makes it difficult to find the Procrustean mean shape is that the pre-shapes $\tau_1, \ldots, \tau_M$ may not be rotated into alignment with each other to begin with. We must simultaneously optimize the rotational alignment of each pre-shape, while at the same time computing the mean shape μ^*, subject to the constraint that μ^* be a pre-shape–i.e. a unit-length shape vector.

Unfortunately, the non-linearity of the Procrustes distance in the $\cos^{-1}(\cdot)$ term leads to an intractable solution for the Fréchet mean minimization. Thus, we approximate the geodesic distance ρ between two pre-shape vectors, $\phi \in \mathcal{O}(\tau_1)$ and $\psi \in \mathcal{O}(\tau_2)$, as the projection distance, r,

$$r = \sin \rho = \sqrt{1 - \cos^2 \rho}. \tag{2.10}$$

[2]More precisely, μ^* is the Fréchet mean of the random variable ξ drawn from a distribution having uniform weight on each of the samples $\tau_1, \ldots, \tau_M$.

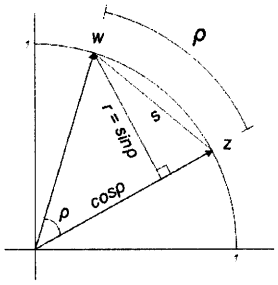


Figure 2-1: Distances on a hyper-sphere. r is a linear approximation to the geodesic distance ρ.

Figure 2-1 depicts this approximation to the geodesic distance graphically[3].

Using this linear projection distance r in the mean shape minimization yields an expression for the mean shape μ^*

$$\mu^* = \arg \inf_{\|\mu\|=1} \sum_i (1 - |\tau_i^H \mu|^2) \tag{2.11}$$

$$= \arg \sup_{\|\mu\|=1} \sum_i (\tau_i^H \mu)^H (\tau_i^H \mu) \tag{2.12}$$

$$= \arg \sup_{\|\mu\|=1} \mu^H (\sum_i \tau_i \tau_i^H) \mu \tag{2.13}$$

$$= \arg \sup_{\|\mu\|=1} \mu^H S \mu, \tag{2.14}$$

thus, μ^* is the complex eigenvector corresponding to the largest eigenvalue of S [12].

2.3.2 The Shape Distribution Covariance

With our measurements $\{\mathbf{m}_1, \mathbf{m}_2, \ldots\}$, we can now fit a probabilistic model of shape. In many applications, pre-shape data will be tightly localized around a mean shape, in such cases, the tangent space to the preshape hypersphere located at the mean shape will be a good approximation to the preshape space, as in figure 2-2. By linearizing our distribution in this manner, we can take advantage of standard multivariate statistical analysis techniques, representing our shape distribution as a Gaussian. In cases where

[3]The straight-line Euclidean distance, s, is another possible approximation to the geodesic distance, but it will not enable us to easily solve the mean shape minimization in closed form.

the data is more spread out, one can use a *complex Bingham distribution* [12].

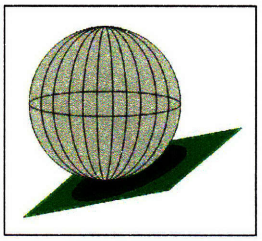


Figure 2-2: Tangent space distribution

In order to transform our set of pre-shapes into an appropriate tangent space, we first compute a mean shape, μ as above[4]. We then fit the observed pre-shapes to μ, and project each fitted pre-shape into the tangent space at μ. The *tangent space coordinates* for pre-shape τ_i are given by

$$\mathbf{v_i} = (I - \mu\mu^H)e^{j\theta_i^*}\tau_i, \tag{2.15}$$

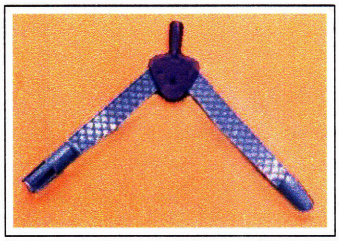
where $j^2 = -1$ and θ_i^* is the optimal Procrustes-matching rotation angle of τ_i onto μ. We then apply a dimensionality reduction technique, using principle components analysis (PCA) to the tangent space training data $\mathbf{v_1}, \ldots, \mathbf{v_M}$ to get a compact representation of estimated shape distribution (Figure 2-2).

Figure 2-3(a) shows one example out of a training set of images of a deformable object. Figure 2-3(b) shows sample objects drawn from the learned distribution. The red contour is the mean, and the blue and green samples are taken along the first two principal components of the distribution.

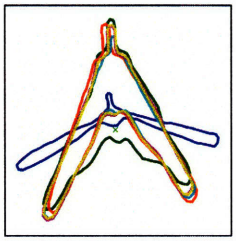
2.3.3 Shape Classification

Given N previously learned shape classes $C_1, \ldots, C_N$ with shape means $\mu_1, \ldots, \mu_N$ and covariance matrices $\Sigma_1, \ldots, \Sigma_N$, and given a measurement $\mathbf{m}$ of an unknown object shape, we can now compute the likelihood of a shape class given a measured

[4]We use μ to refer to the μ^* computed from the optimization in 2.9 for the remainder of the paper.



(a) Example Object



(b) Class Distribution

Figure 2-3: (a) An example image of a chalk compass. The compass can deform by opening and closing. (b) Sample shapes from the learned distribution along different eigenvalues of the distribution.

object: $\{P(C_i|\mathbf{m}) : i = 1 \ldots N\}$. The shape classification problem is to find the maximum likelihood class, $\hat{C}$, which we can compute as

$$\hat{C} = \arg\max_{C_i} P(C_i|\mathbf{m}) \tag{2.16}$$

$$= \arg\max_{C_i} P(\mathbf{m}|C_i)P(C_i). \tag{2.17}$$

Given the mean and covariance of a shape class C_i, we can compute the likelihood of a measured object given a class as follows. First, we compute τ, the pre-shape of $\mathbf{m}$, and we rotate τ to fit the mean shape μ_i and project into tangent space, resulting in a tangent space shape vector, $\mathbf{v}$. Then, we project $\mathbf{v}$ onto the top k principle components (eigenvectors), $\mathbf{b}_1, \ldots, \mathbf{b}_k$, of C_i's covariance matrix, and we compress $\mathbf{v}$ to a vector of length $k+1$, where the first k components are the projection scores of $\mathbf{v}$ onto the top k eigenvectors, and the final dimension is the projection distance from $\mathbf{v}$ to the top-k eigenvector subspace. The *compressed tangent space coordinates*, $\mathbf{v}^\perp$, of pre-shape τ with respect to the shape model C_i are

$$\mathbf{v}_i^\perp = \mathbf{b}_i \cdot \mathbf{v} \quad \forall i = 1..k \tag{2.18}$$

$$\mathbf{v}_{k+1}^\perp = \|\mathbf{v} - BB^T\mathbf{v}\| \tag{2.19}$$

where B is the matrix with eigenvectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$ in its columns.

In order to compute the likelihood of the compressed shape vector, $\mathbf{v}^\perp$, with

SHAPE_LIKELIHOOD($\mathbf{x}$, $\mathbf{S}$, k)

Input: Shape $\mathbf{x}$, shape model $\mathbf{S}$ (with mean μ and covariance Σ), and the number of principle components, k, to use in computing the likelihood.

Output: Likelihood $L = p(\mathbf{x}; \mathbf{S})$

- Compute $\tau \leftarrow$ PRESHAPE($\mathbf{x}$).
- Rotate τ to fit μ.
- Project τ into μ's tangent space to get tangent space coordinates, $\mathbf{v}$.
- Compress $\mathbf{v}$ to the $k+1$ dimensional vector $\mathbf{v}^{\perp}$ using the top k eigenvectors of the model covariance matrix, Σ, and compute (or look up) the compressed covariance matrix, $\Sigma^{\perp}$.
- Return $L \leftarrow \mathcal{N}(\mathbf{v}^{\perp}; \mathbf{0}, \Sigma^{\perp})$.

Table 2.1: Shape likelihood algorithm for computing the likelihood of a particular shape, $\mathbf{x}$ under a tangent space principle components model, $\mathbf{S}$.

respect to C_i's model, we must compress the mean and covariance as well. Since tangent space is perpendicular to the mean shape, the mean shape remains at the origin in compressed tangent space. The compressed covariance, $\Sigma_i^{\perp}$, is also easy to compute; it is a diagonal matrix with the first k diagonal elements being the top k eigenvalues of the full covariance, and the $k+1$st entry being the sum of the remaining $2n-k$ eigenvalues (since the $k+1$st dimension of the compressed shape model captures all of the variation that does not fall along the top k principle components). We then calculate the likelihood of $\mathbf{m}$ with respect to shape class C_i as

$$p(\mathbf{m}|C_i) = \mathcal{N}(\mathbf{v}^{\perp}; \mathbf{0}, \Sigma_i^{\perp}). \tag{2.20}$$

The full shape likelihood algorithm is shown in table 2.1.

Finally, assuming a uniform prior on C_i, we can compute the maximum likelihood class as

$$\hat{C} = \underset{C_i}{\operatorname{argmax}} \mathcal{N}(\mathbf{v}^{\perp}; \mathbf{0}, \Sigma_i^{\perp}). \tag{2.21}$$

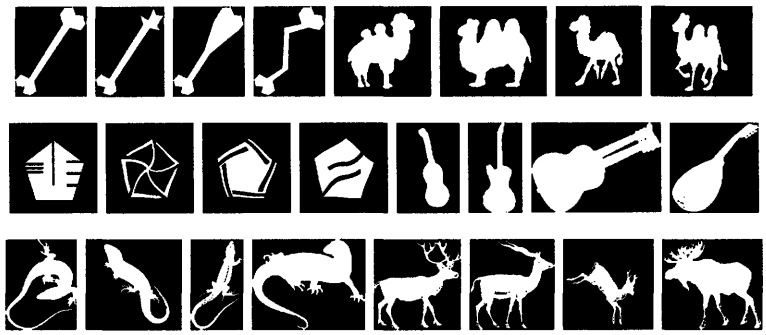


Figure 2-4: A sample from six classes of the MPEG-7 shape dataset. The many different types of shape variation present have made this dataset an extremely challenging one for shape recognition algorithms.

2.4 Experiments

In order to compare the performance of different shape metrics with one another, one needs a common dataset to test them on. In the past decade, the MPEG-7 Shape B dataset [31] has found great popularity among the shape recognition community, since being selected as the test set for the MPEG-7 video compression standard. The MPEG-7 dataset contains 70 classes of 20 shapes each, represented by black and white image silhouettes, as in figure 2-4.

2.4.1 Retrieval

One common performance test of shape similarity metrics is called the bullseye retrieval test. The bullseye test is a leave-one-out (LOO) test, which takes in a query shape, $\mathbf{v}$, and computes the percentage of matches out of the total number of shapes per class, m (including the query shape), within the top $2m$ matches which belong to the same class as the query shape. (For the MPEG-7 dataset, $m = 20$, since there are 20 shapes per class.) The bullseye retrieval rate is the average bullseye rate over all LOO tests (70×20 for the MPEG-7 dataset).

We wanted to see how the Procrustean shape distance compared to other popular shape metrics in the literature, so we tried running a Bullseye test on the MPEG-7

dataset. Unfortunately, the Procrustean metric is an extremely rigid form of shape metric. As was mentioned above, the Procrustean metric simply compares the configurations of two sets of ordered point sets, and requires that the two point sets are already in one-to-one correspondence with each other. However, since shapes are extracted from the MPEG-7 dataset from the boundaries of silhouette images, there is no guarantee that the starting points of each boundary will be the same, or that the boundaries will have the same number of points. The latter problem can be fixed by sub-sampling $n = 100$ evenly-spaced points around all of the boundaries, once they are extracted from the images. To solve the first problem, we must perform a minimization over all n possible starting points on one shape. Thus, if we let $\mathbf{z}^{(i)}$ be the permutation of $\mathbf{z}$ starting from the ith point, that is;

$$\mathbf{z}^{(i)} = (\mathbf{z}_i, \mathbf{z}_{i+1}, ..., \mathbf{z}_n, \mathbf{z}_1, ..., \mathbf{z}_{i-1}), \tag{2.22}$$

then we can define the *Shifted Procrustean Metric*, $d_{SP}(\cdot)$, as the minimum Procrustean distance over all possible starting points on one shape,

$$d_{SP}(\mathbf{z}, \mathbf{w}) = \min_i d_P(\mathbf{z}^{(i)}, \mathbf{w}). \tag{2.23}$$

Another challenge that the MPEG-7 dataset presents is that it contains several "mirror-image" shapes. Thus, shape metrics which are tested on the MPEG-7 dataset typically also take a minimum distance over reflections, which we do by defining the *Mirrored Shifted Procrustean Metric*, $d_{MSP}(\cdot)$, as the minimum Shifted Procrustean distance over reflections,

$$d_{MSP}(\mathbf{z}, \mathbf{w}) = min[d_P(\mathbf{z}, \mathbf{w}), d_P(\mathbf{z}', \mathbf{w})], \tag{2.24}$$

where we let $\mathbf{z}'$ be the reflection of $\mathbf{z}$ about the $x-axis$ (or the $y-axis$, equivalently).

With this modified Procrustean metric, $d_{MSP}(\cdot)$, we then ran the Bullseye test on the full MPEG-7 dataset, achieving an average retrieval rate of 77.04%. Impressively, this is slightly higher than the published results for the popular shape context

distance [2], which achieved a Bullseye retrieval rate of 76.51%. The best published result is 87.23%, due to the Dynamic Space Warping (DSW) of Alajlan et. al; however, in Chapter 5 we use the Procrustean metric to learn a local shape distance with AdaBoost which outperforms DSW, achieving a retrieval rate of 89.49%.

2.4.2 Classification

We also wanted to test the performance of the Procrustean shape distance for classification, as a baseline to compare against the performance of the full probabilistic models. We ran a 5-fold cross-validated classification test on the MPEG-7 dataset, using the Mirrored Shifted Procrustean (MSP) metric, together with nearest neighbor (NN) classification. The classification rate of this NN classifier was 95.86%.

We then tested the performance of the probabilistic Procrustean model of section 2.3. First, we computed the mean shape and covariance for every shape class in each CV training set, using the MSP metric. Then, we corresponded (with MSP) and classified all testing shapes according to equation 2.21, with different numbers of principle components, k. The highest classification rate of this MSP classifier was 63.86%, achieved at $k = 6$ PCs, which was far below the NN classification rate of 95.86%. One possible explanation for this drop off in performance is that, despite the high variance of many classes in the MPEG-7 dataset, there were enough training samples per class (16) so that similar shapes to each test shape were usually present in the training set. Still, the MSP classifier leaves much to be desired, since any shape class with a large amount of variation will learn models where training shape parts are incorrectly corresponded to one another, leading to garbled models, as in figure 2-5.

2.4.3 Procrustean Metric vs. Shape Contexts

The similar performance of the Procrustean metric to the shape context distance on the MPEG-7 Bullseye test is surprising, since the correspondence algorithm we used for the Procrustean metric is just a simple scan for a good starting point, whereas the

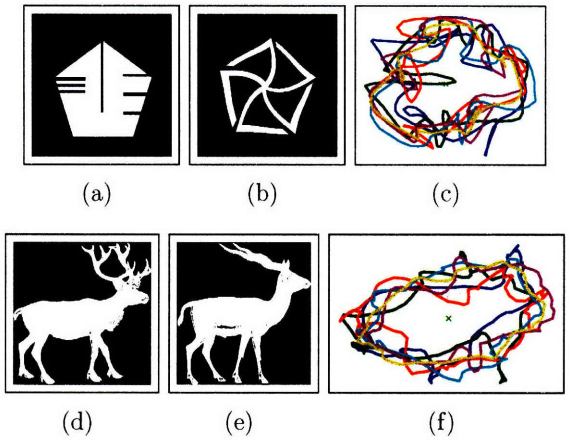


Figure 2-5: Shape classes from the MPEG-7 shape dataset which are modeled poorly with Mirrored Shifted Procrustean (MSP) correspondences. (a-b) Two examples of shapes from the "deer" class. (c) The mean shape (in red) plotted with shapes sampled along each of the top 5 principle components for the deer class. (d-e) Two examples of shapes from the "device6" class. (f) The mean shape (in red) plotted with shapes sampled along each of the top 5 principle components for the deer class.

shape context uses locally weighted shape signatures to determine correspondences using a bipartite graph matching algorithm. However, the Procrustean metric makes up for potentially poor correspondences by achieving full rotation invariance, whereas the shape context does not achieve this invariance. It is this orientation invariance (along with the position and scale invariance), which makes the Procrustean metric such an attractive shape distance on which to base a probabilistic model of object shape.

In the next chapter, we will seek to improve upon our basic scanning correspondence algorithm. This will become increasingly important as we attempt to capture the variation in a class of object shapes, since the same parts of an object must be matched up with each other in order to accurately model the distribution of shapes.

Chapter 3

Data Association and Shape Correspondences

Most shape modeling and recognition algorithms rely on being able to solve a correspondence problem between parts of two or more objects. Solving for the most likely correspondences between sets of data is an open problem in a number of fields, including computer vision and robot mapping. As object geometries vary due to projection distortions, sensor error, or even natural object dynamics, determining *which* part of an object image corresponds to *which* part of a previous image is non-trivial.

In 2-D, a shape is often represented as a closed contour [4, 19, 30, 34, 51]. Some methods use continuous representations of shape contours, matching curve segments on one contour to those on another [49], or matching the control points of two active contours [24]. Others use the skeleton of a contour to represent the shape of an object [50, 52]. For these methods, skeletal tree branches must be matched to each other, and a tree edit distance is often employed to determine the similarity between shapes. In this thesis, we represent the shape of an object by a densely-sampled, ordered set of discrete points around its contour, where one must find an optimal assignment of the points on one contour to the points on another contour. In each model, one must solve the correspondence problem in order to proceed with the computation of shape distances, distributions, and likelihoods.

We have already seen one example of a correspondence algorithm in chapter 2,

where the *Shifted Procrustean Metric* 2.23 was used to match up the points of one shape contour with the points on another contour by finding the best starting point, and assigning the rest of the points in one-to-one correspondence around the two contours. In this chapter, we will develop a more flexible framework for shape correspondences, and will show that by using this more general framework for data association, we can improve shape classification performance on the MPEG-7 data set over the simple Shifted Procrustean Metric.

In order to find correspondences between the points of two contours, one typically gives a cost function, $C(i,j)$, for assigning point i on the first contour to point j on the second contour, and these costs are usually assumed to all be independent. In addition, we may have constraints on the matching; for example, we may require an order-preserving matching (Figure 3-2). Scott and Nowak [48] define the Cyclic Order-Preserving Assignment Problem (COPAP) as the problem of finding an optimal matching such that the assignment of corresponding points preserves the cyclic ordering inherited from the contours. Alternatively, if we do not require the assignment to preserve contour ordering, yet we do desire a one-to-one matching, the problem may be formulated as a bipartite matching [2] or as a relaxation labeling [44].

3.1 Point Assignment Cost Functions

One recent approach which has had some success using the point-assignment cost framework is that of *shape contexts* [2]. The shape context of a point captures the relative positions of other points on the contour in a log-polar histogram. The point assignment cost is determined by a χ^2 distance between corresponding shape context histograms.

Another option is to simply align the two shapes as closely as possible, e.g. using a Hausdorff distance [23] or Iterated Closest Point (ICP) [3], and then to compute the cost function between two points as the Euclidean distance from one point to the other. Unfortunately, this Euclidean cost method can give poor results when the contours have large concavities on them, or in the presence of articulation (e.g. the

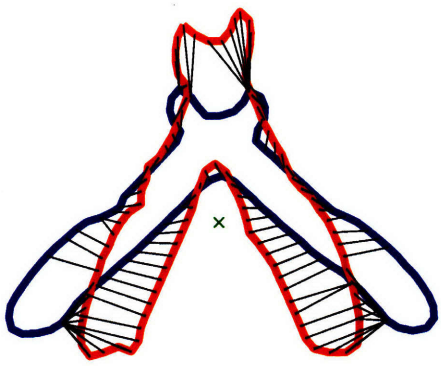
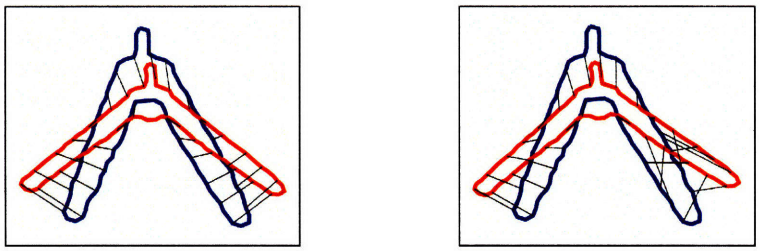


Figure 3-1: Tool articulation. Euclidean distance cost function is sensitive to articulation and concavities, as well as to the initial alignment (in this case by ICP).



(a) (b)

Figure 3-2: Order-preserving matching (a) vs. Non-order-preserving matching (b). The thin black lines depict the correspondences between points in the red and blue contour. Notice the violation of the cyclic-ordering constraint between the right arms of the two contours in the right image. Without the cyclic ordering constraint, these two shapes may appear to be extremely dissimilar.

opening and closing of a tool–figure 3-1). It is also extremely sensitive to the initial alignment, so such methods should only be used when it is known, *a priori*, that the two shapes will be very similar.

A third method, which we will use in this thesis, is to use local, rather than global shape information at each point, comparing the shapes of neighborhoods around the two points in order to determine the point assignment cost. To do this, we will use the Procrustean shape metric of chapter 2 to compute the shape distance between two point neighborhoods, and will then take the point-assignment cost as a weighted sum of neighborhood shape distances over varying neighborhood sizes.

By the nature of object contours, our specific shape correspondence problem con-

tains a *cyclic order-preserving* constraint, that is, correspondences between the two contours cannot "cross" each other. Scott and Nowak [48] define the Cyclic Order-Preserving Assignment Problem (COPAP) as the problem of finding an optimal one-to-one matching such that the assignment of corresponding points preserves the cyclic ordering inherited from the contours. Figure 3-2(a) shows an example set of correspondences (the thin black lines) that preserve the cyclic order-preserving constraint, whereas the correspondences in figure 3-2(b) violate the constraint at the right of the shape (notice that the association lines cross.) In the following sections, we show how the original COPAP algorithm can be written as a linear graphical model with the introduction of additional book-keeping variables.

Our goal is to match the points of one contour, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ to the points on another, $\mathbf{y}_1, \ldots, \mathbf{y}_m$. Let Φ denote a correspondence vector, where ϕ_i is the index of $\mathbf{y}$ to which $\mathbf{x}_i$ corresponds; that is: $\mathbf{x}_i \rightarrow \mathbf{y}_{\phi_i}$. We wish to find the most likely Φ given $\mathbf{x}$ and $\mathbf{y}$, that is, $\Phi^* = \text{argmax}_\Phi\, p(\Phi|\mathbf{x}, \mathbf{y})$. If we assume that the likelihood of individual points $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_j\}$ are conditionally independent given Φ, then

$$\Phi^* = \operatorname*{argmax}_\Phi \frac{1}{Z} p(\mathbf{x}, \mathbf{y}|\Phi) p(\Phi) \tag{3.1}$$

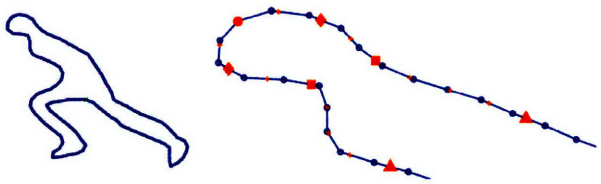
$$= \operatorname*{argmax}_\Phi \frac{1}{Z} \prod_{i=1}^{n} p(x_i, y_{\phi_i}) p(\Phi) \tag{3.2}$$

where Z is a normalizing constant.

3.2 Priors over Correspondences

There are two main terms to equation (3.1), the prior over correspondences, $p(\Phi)$, and the likelihood of object points given the correspondences, $p(\mathbf{x}_i, \mathbf{y}_{\phi_i})$. We model the prior over correspondences, $p(\Phi)$, as an exponential distribution subject to the cyclic-ordering constraint. We encode this constraint in the prior by allowing $p(\Phi) > 0$ if and only if

$$\exists \omega \;\; s.t. \;\; \phi_\omega < \phi_{\omega+1} < \cdots < \phi_n < \phi_1 < \cdots < \phi_{\omega-1}. \tag{3.3}$$



(a) (b)

Figure 3-3: Local shape neighborhoods. (a) The full contour of a running person. (b) Closeup of the top of the contour in (a), with local shape neighborhoods about the point • of size $k = 1$ (♦), $k = 2$ (■), and $k = 3$ (▲), where the original contour points are shown as small blue circles (○) and the interpolated neighborhood points are shown as small red +'s. The neighborhoods are chosen so that the length of the largest neighborhood (▲) is 20% of the full contour length.

We call ω the *wrapping point* of the assignment vector Φ. Each assignment vector, Φ, which obeys the cyclic-ordering constraint must have a unique wrapping point, ω.

Due to variations in object geometry, the model must allow for the possibility that some sequence of points of $\{\mathbf{x}_i, \ldots, \mathbf{x}_j\}$ do not correspond to any points in $\mathbf{y}$, for example, if sensor noise has introduced spurious points along an object edge or if the shapes vary in some significant way, such as an animal contour with three legs where another has four. We "skip" individual correspondences in $\mathbf{x}$ by allowing $\phi_i = 0$. (Points $\mathbf{y}_j$ are skipped when $\nexists i \; s.t. \; \phi_i = j$). We would like to minimize the number of such skipped assignments, so we give diminishing likelihood to Φ as the number of skipped points increases. Therefore, for Φ with k skipped assignments (in $\mathbf{x}$ and $\mathbf{y}$),

$$p(\Phi) = \begin{cases} \frac{1}{Z_\Phi} \exp\{-k(\Phi) \cdot \lambda\} & \text{if } \Phi \text{ is cyclic ordered} \\ 0 & \text{otherwise,} \end{cases} \tag{3.4}$$

where Z_Φ is a normalizing constant and λ is a likelihood penalty for skipped assignments.

3.3 Correspondence Likelihoods

Given an expression for the correspondence prior, we also need an expression for the likelihood that two points $\mathbf{x}_i$ and $\mathbf{y}_{\phi_i}$ correspond to each other, $p(\mathbf{x}_i, \mathbf{y}_{\phi_i})$, which we model as the likelihood that the local geometry of the contours match. Chapter 2 described a probabilistic model for global geometric similarity using the Procrustean metric, and we specialize this model to computing the likelihood of local geometries. We compute this likelihood by first forming a distance metric over local shape geometry, which we call the *Procrustean Local Shape Distance* (PLSD). Given such a distance, d, we compute the likelihood as the probability of d under a zero-mean Gaussian model with fixed variance, σ. Since σ is fixed for every local shape correspondence likelihood, we can simply write it as part of the normalization constant to ensure that the distribution $p(\mathbf{x}_i, \mathbf{y}_{\phi_i})$ sums to one. Thus,

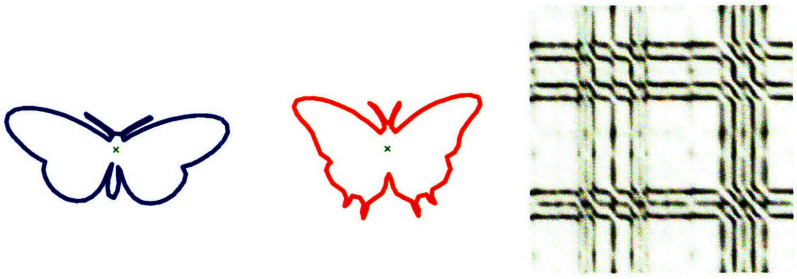
$$p(\mathbf{x}_i, \mathbf{y}_{\phi_i}) = \frac{1}{Z_{PLS}} \exp\left\{-[d_{PLS}(x_i, y_{\phi_i})]^2\right\} \tag{3.5}$$

In order to compute the Procrustean local shape distance, we first need a description of the local shape about $\mathbf{x}_i$. Instead of simply taking the p closest points to $\mathbf{x}_i$ on $\mathbf{x}$'s contour, we instead sample points evenly spaced about $\mathbf{x}_i$ so as to be robust to local placement of $\mathbf{x}$'s points. In other words, we define the *local neighborhood* of size k about $\mathbf{x}_i$ as:

$$\eta_k(x_i) = \langle \delta_x^i(-2^k\Delta), ..., \delta_x^i(0), ..., \delta_x^i(2^k\Delta) \rangle \tag{3.6}$$

where $\delta_x^i(d)$ returns the point from x's contour interpolated a distance of d starting from $\mathbf{x}_i$ and continuing clockwise for d positive or counter-clockwise for d negative. (Also, $\delta_x^i(0) = \mathbf{x}_i$.) The parameter Δ determines the step-size between interpolated neighborhood points, and thus the resolution of the local neighborhood shape. We have found that setting Δ such that the largest neighborhood is 20% of the total shape circumference yields good results on most datasets (Figure 3-3).

The Procrustean Local Shape Distance, d_{PLS}, between two points, x_i and y_j is



(a) (b) (c)

Figure 3-4: PLSD cost matrix for two butterfly contours (a-b). The cost matrix has a very regular structure.

the mean Procrustean shape distance over neighborhood sizes k:

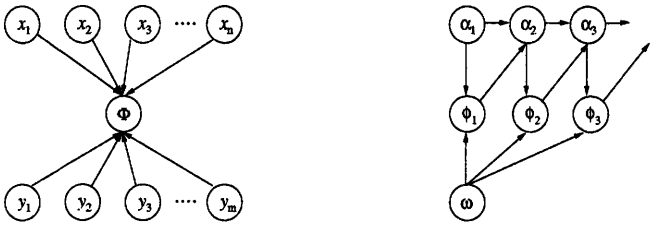
$$d_{PLS}(x_i, y_j) = \int_k \xi_k \cdot d_P[\eta_k(x_i), \eta_k(y_j)] \tag{3.7}$$

with neighborhood size prior ξ. No closed form exists for this integral so we approximate it using a sum over a discrete set of neighborhood sizes. Experimentally, we found that summing over $k = \{1, 2, 3\}$ and setting $\xi_1 = 4/7, \xi_2 = 2/7, \xi_3 = 1/7$ (so that each local neighborhood distance has half the weight of the neighborhood distance at the next smallest scale) yielded good results on most shape contours.

In figure 3-4, we see the matrix of squared Procrustean Local Shape Distances between all pairs of points on two butterfly contours. Note that the squared-distance matrix has a very regular structure. The dark, high cost rows and columns correspond to strong local features on each shape–for example, the tips of the wings, or the antennae; while the light, low-cost rows and columns correspond to flat, smooth portions of the two contours.

3.4 A Graphical Model for Shape Correspondences

Figure 3-5(a) shows the graphical model for inferring the correspondences assuming independence of local features $\mathbf{x}$ and $\mathbf{y}$. Unfortunately, although the local features are independent, the cyclic-ordering constraint leads to dependencies between the



(a) Full graphical model

(b) The Φ sub-graph as a chain

Figure 3-5: (a) A graphical model for the cyclic ordering problem. We assume independence of the local features $\mathbf{x}_i$ and $\mathbf{y}_j$, but the components ϕ_i of the correspondence vector Φ are fully connected. (b) The sub-graph of the Φ vector after breaking dependencies. Adding states $\{\alpha_i\}$ and ω, and choosing ω, allows us to express the model for Φ as a cyclic Markov chain. The dependence on $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_j\}$ is omitted for clarity.

assignment variables ϕ_i in a non-trivial way—in fact, the sub-graph of Φ is fully connected since each ϕ_i must know the values of all the other assignments, ϕ_j, in order to determine whether the matching is order-preserving or not. Computing the maximum likelihood Φ is therefore a non-trivial loopy graphical inference problem.

We can avoid this problem and break most of these dependencies by introducing variables α_i and ω, where α_i corresponds to the last non-zero assignment before ϕ_i and ω corresponds to the wrapping point from section 3.2. With these additional variables, each ϕ_i depends only on the wrapping point, which is stored in ω as well as the last non-zero assignment, α_i; the cyclic ordering-constraint is thus encoded by $p_{co}(\phi_i)$, such that

$$p_{co}(\phi_i) = \begin{cases} \frac{1}{Z_{co}} & : \text{if } \phi_i > \alpha_i \text{ or} \\ & \phi_i < \alpha_i \text{ and } \omega_i = i \text{ or} \\ & \phi_i = 0 \\ 0 & : \text{otherwise,} \end{cases} \tag{3.8}$$

which gives (3.9)

$$p(\Phi) = \frac{1}{Z_\Phi}(\exp\{-k(\Phi)\cdot\lambda\})\prod_i p_{co}(\phi_i). \tag{3.10}$$

where Z_{co} and Z_{Φ} are normalizing constants, to make the distributions sum to one. If we do not initially assign the wrapping point ω, the state vector $\{\alpha_i, \phi_i\}$ then yields a cyclic Markov chain (figure 3-5(b)), which leads to a computationally intractable inference problem. The standard approach to solving this cyclic Markov chain is therefore to try setting the wrapping point, ω, to each possible value from 1 to n. Given $\omega = k$, the cycle is broken into a linear chain (according to equation 3.3), which can be solved by dynamic programming. It is this introduction of the α_i and ω variables that is the key to the efficient inference procedure by converting the loopy graphical model into a linear chain.

In this approach, the point-assignment likelihoods are converted into a cost function $C(i, \phi_i)$ by taking a log likelihood, and Φ is optimized using

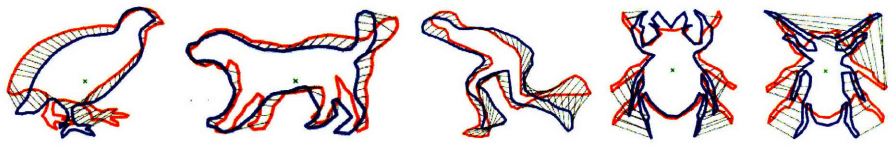
$$\Phi^* = \underset{\Phi}{\text{argmax}} \log \prod_i p(\mathbf{x}_i, \mathbf{y}_{\phi_i}) p(\Phi) \prod_i p_{co}(\phi_i) \tag{3.11}$$

$$= \underset{\Phi}{\text{argmin}} \left(\sum_i C(i, \phi_i) \right) + \lambda \cdot k(\Phi) \tag{3.12}$$

$$\text{s.t. } \forall \phi_i \; p_{co}(\phi_i) > 0$$

where $k(\Phi)$ is the number of points skipped in the assignment Φ. Solving for Φ using equation (3.12) takes $O(n^2 m)$ running time; however a bisection strategy exists in the dynamic programming search graph which reduces the complexity to $O(nm \log n)$ [36]. We follow Scott and Nowak [48] in referring to the cyclic Markov chain problem as the COPAP problem and solve it using this dynamic programming solution to the n "unwrapped" linear Markov chain inference problems.

Figure 3-6 shows examples of the inference process and correspondences between pairs of contours. Figure 3-6(a) is interesting because the correspondence algorithm has correctly associated all of the single leg present in the blue contour with the right-most leg in the red contour, and skipped any associations of the left-leg in the red contour. Figure 3-6(e), the "beetle" model, shows a failed correspondence at the top-right legs of two beetles; this is a challenging case because there are a number of similar structures for the correspondence to choose from.



(a) Bird (b) Dog (c) Running Person (d) Beetle (e) Beetle

Figure 3-6: Examples of shape correspondences found using the graphical model of figure 3-5(b). Note that in (e) the top-right legs of the beetles are incorrectly matched due to the local nature of the point assignment.

3.5 The COPAP Algorithm

3.5.1 LOPAP

Scott and Nowak refer to the n linear Markov chain problems–which occur as a result of setting the wrapping point, ω–as instances of the Linear Order-Preserving Assignment Problem (LOPAP), so called because solutions Φ to the LOPAP problem must be *linear-order preserving*, i.e. $\phi_1 \leq \phi_2 \leq \cdots \leq \phi_n$.

LOPAP has a natural formulation as a shortest paths problem, via the following graph construction (Figure 3-7). Construct an $(n+1) \times (m+1)$ matrix of nodes, with directed edges between neighboring edges in the EAST, SOUTH, and SOUTHEAST directions. Corresponding to each column of edges is a point $\mathbf{x}_i$ on contour $\mathbf{x}$, and corresponding to each row of edges is a point $\mathbf{y}_j$ on contour $\mathbf{y}$. In such a graph, every path from the top-left-most node to the bottom-right-most node corresponds to a (not necessarily unique) assignment vector, Φ, from $\mathbf{x} \rightarrow \mathbf{y}$. A SOUTHEAST step over $\mathbf{x}_i$'s column of edges and $\mathbf{y}_j$'s row of edges represents the point correspondence $\phi_i = j$. Thus, the cost of each SOUTHEAST edge is $C_{ij} = [d_{PLS}(\mathbf{x}_j, \mathbf{y}_i)]^2$. EAST and SOUTH steps translate to skipped assignments over x and y, respectively, and thus have a cost of λ.

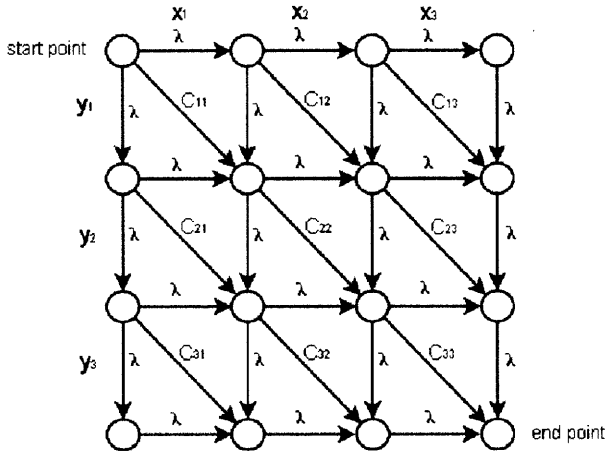


Figure 3-7: Standard LOPAP graph (with symmetric skipping costs). Diagonal edges correspond to point assignments, ϕ_i, while horizontal and vertical edges add skipping penalty, λ.

3.5.2 COPAP

COPAP can be solved naively with the solution to n LOPAPs, each with a different rotation of y's point ordering. The edge rows in the LOPAP graph would be labeled (from top to bottom) $y_1, \ldots, y_n$ in the first LOPAP, $y_2, \ldots, y_n, y_1$ in the second LOPAP, etc. These n searches could also be accomplished by adding another copy of the LOPAP graph below the original graph, as in Figure 3-8. We call this new graph the *COPAP graph*, and search from n consecutive starting locations at the top-left of the graph to n consecutive end points at the bottom-right. The running time of this solution to COPAP, where we solve each LOPAP with dynamic programming, is $O(n^2m)$, since each LOPAP solution takes $O(nm)$ time to complete, and we must perform n LOPAPs in total, each independently of the others. We call this algorithm "DP n-times".

Now consider the state of this "DP n-times" algorithm after finding a shortest path from s_i to e_i, but before finding the shortest path from s_{i+1} to e_{i+1} (Figure 3-9). The shortest path from s_{i+1} to e_{i+1} must lie below the shortest path from s_i to e_i, since any path from s_{i+1} to e_{i+1} which crosses over the previous shortest path would have been better off to have followed the shortest path between the intersection points rather than taking the detour above it. The same argument applies if we have already computed paths below $s_{i+1} \rightarrow e_{i+1}$, so that we can constrain our search space from

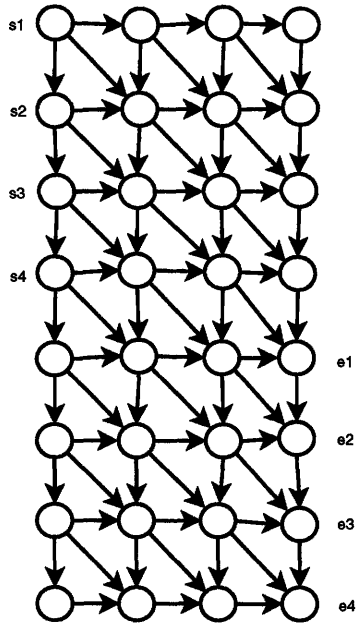


Figure 3-8: COPAP graph. The solution to COPAP can by found by computing n shortest paths, from s_k to e_k, and choosing the assignment vector, Φ^k, corresponding to the shortest one.

above and below, depending on the order in which paths are found in the COPAP graph.

From [36], we see that the best order in which to compute the shortest paths is a bisection ordering, starting with $s_{n/2} \rightarrow e_{n/2}$ at the first level, then $s_{n/4} \rightarrow e_{n/4}$ and $s_{3n/4} \rightarrow e_{3n/4}$ at the second level, and so on. There are $O(\log n)$ levels in total, and at each level we perform a series of disjoint bounded searches, with a total of $O(nm)$ work being done in the DPs at each level. Thus, the total running time of this bisection algorithm, which we call "DP bisection", is $O(nm \log n)$.

3.5.3 COPAP Variants

In the following sections we will need to slightly modify the basic COPAP algorithm to enforce constraints of the various modeling and inference problems we will face. None of the variants will be substantially different from the basic COPAP algorithm above, but it is worth pointing out and setting a naming convention for the variants

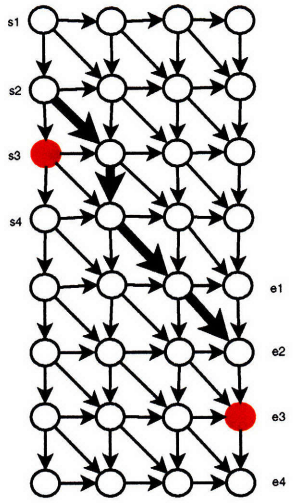


Figure 3-9: COPAP bisection strategy. Each shortest path from s_i to e_i is constrained to lie between the shortest paths below and above it.

to avoid confusion in later sections.

We will use four different variants in this work. Three of the variants modify only the skipping penalty, λ. The fourth variant is for correspondence between a partial shape and a complete shape, and modifies point assignment costs (likelihoods) as well. The variants (and their inputs) are:

1. COPAP_SYMMETRIC($\mathbf{x}$, $\mathbf{y}$): The standard COPAP algorithm, with symmetric spacing costs, λ.

2. COPAP_ASYMMETRIC($\mathbf{x}$, $\mathbf{y}$): Disallows skipped points on $\mathbf{x}$; skipped points on $\mathbf{y}$ have cost λ.

3. COPAP_SEMISYMMETRIC($\mathbf{x}$, $\mathbf{y}$, $\mathbf{h}$): Takes as input two shapes, $\mathbf{x}$ and $\mathbf{y}$, and also a binary mask $\mathbf{h}$; points $\mathbf{x}_i$ can be skipped only if $\mathbf{h}_i = 1$; skipped points (on $\mathbf{y}$ and where allowed on $\mathbf{x}$) have cost λ.

4. COPAP_WILD($\mathbf{x}$, $\mathbf{y}$, $\mathbf{h}$): Takes as input two shapes, $\mathbf{x}$ and $\mathbf{y}$, and also a binary mask $\mathbf{h}$; points $\mathbf{x}_i$ for which $\mathbf{h}_i = 1$ represent "wildcard" points, which can be skipped with cost λ or assigned with cost ϵ to any point $\mathbf{y}_j$; all other points on $\mathbf{x}$ cannot be skipped, and skipped points on $\mathbf{y}$ have cost λ.

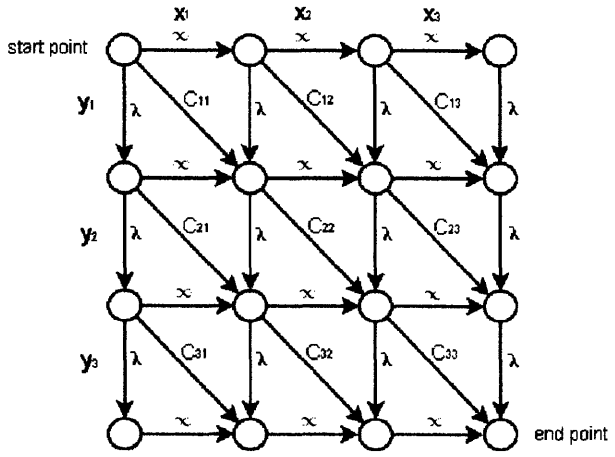


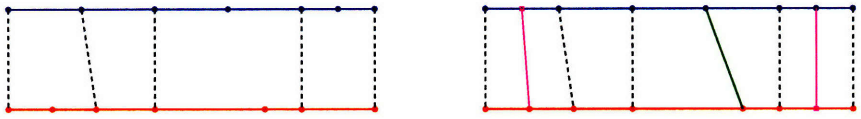
Figure 3-10: A LOPAP graph for the "COPAP_ASYMMETRIC" problem (with asymmetric skipping costs). Diagonal edges correspond to point assignments, ϕ_i, while vertical edges add skipping penalty, λ, and horizontal edges have infinite cost (and thus cannot be traversed). The remaining COPAP variants have a similar structure.

Each of the four COPAP variants require only a modification to the costs in the DP graph to solve; the graph topology remains the same, as can be seen in figure 3-10.

3.6 Skipped Assignments and Model Learning

In the variants of the COPAP algorithm we have described above, we have allowed skipped assignments, which gives the algorithm the power to skip over points on one contour which do not match up well with points on the other contour. The ability to skip assignments is a necessary feature, since it gives the algorithm flexibility which was lacking from the simple scanning correspondence algorithm of chapter 2. However, for the purposes of shape comparison and model building, we cannot simply "throw away" points which are skipped. There are some subtle differences between how one must handle skipped points for model building, and how one must handle skipped points for comparing two shapes (or computing the likelihood of a shape with respect to a tangent space Gaussian model). First we consider the task of model building, or learning a shape distribution, and then we consider the task of computing shape likelihoods.

Consider a class of dog shape contours (as seen from the side). Some contours will



(a) (b)

Figure 3-11: Adding skipped points to the matched contours. (a) Corresponded portions of two flat contours (red and blue solid lines), with correspondences given by dashed, black lines. (b) The same contours, after corresponding skipped points. First, where possible (i.e. where it would not violate the cyclic ordering constraint), skipped points are assigned to each other (the solid green line at center right). Second, new points are added where no point exists to correspond to an unmatched point on the opposite contour (solid pink lines at the left and right).

have four legs, while some will have three or even two legs, due to occlusions. If we correspond a contour with three legs to a contour with four legs, one of the legs will be skipped in the optimal assignment. If we simply ignore these skipped points, we will build a model for dog shape containing only three (or two) legs, while we would like to capture the fact that the model can contain a variable number of legs. Thus, we will add skipped points to two corresponded shapes, $\mathbf{x}$ and $\mathbf{y}$, with the following two rules (in succession):

1. For segments of points skipped on both $\mathbf{x}$ and $\mathbf{y}$, correspond the points on the segment with fewer points evenly to the points on the other segment.

2. For segments of points skipped only on $\mathbf{x}$ (or $\mathbf{y}$), add an equal number of points to the other contour interpolated evenly between the two nearest points.

These two rules will result in two new contours, $\mathbf{x}'$ and $\mathbf{y}'$, containing an equal number of points, in one-to-one correspondence with each other. $\mathbf{x}'$ and $\mathbf{y}'$ will have at least as many points as the original contours.

In order to build a shape model from a set of n training shapes, we must bring n shapes into correspondence with each other, not just two. In order to correspond n shapes to each other, one could (in theory) set up a giant optimization problem to find all n assignment vectors simultaneously. However, this approach is likely to be both complex as well as prohibitively slow. A simpler approach, which we will take,

LEARN_SHAPE_MODEL_COPAP($\mathbf{x}_1, \ldots, \mathbf{x}_M$)

Input: A set of M full shape contours, $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$.

Output: Shape model $\mathbf{S}$ consisting of mean shape μ and covariance Σ.

- Set $\mu \leftarrow \mathbf{x}_1$.
- For $i = 2, \ldots, M$:
  1. $\phi \leftarrow$ COPAP_SYMMETRIC($\mathbf{x}_i, \mu$).
  2. $(\mathbf{x'}_i, \mu') \leftarrow$ ADD_SKIPPED_POINTS($\mathbf{x}_i, \mu, \phi$).
  3. $\tau_x \leftarrow$ PRESHAPE($\mathbf{x'}_i$).
  4. $\tau_\mu \leftarrow$ PRESHAPE(μ').
  5. Rotate τ_x to fit τ_μ.
  6. $\mu \leftarrow \frac{(i-1)\tau_\mu + \tau_x}{i}$.
- For $i = 1, \ldots, M$:
  1. $\phi \leftarrow$ COPAP_ASYMMETRIC($\mathbf{x}_i, \mu$).
  2. $(\mathbf{x'}_i, \mu') \leftarrow$ ADD_SKIPPED_POINTS($\mathbf{x}_i, \mu, \phi$).
- Let $\mathbf{X}' = \{\mathbf{x'}_1, \mathbf{x'}_2, \ldots, \mathbf{x'}_M\}$.
- Return $(\mu, \Sigma) \leftarrow$ SHAPE_PCA($\mathbf{X}'$).

Table 3.1: The shape model learning algorithm, with unknown correspondences. In the first for-loop, an initial estimate of the mean shape, μ, is learned. In the second for-loop, all the training shapes are brought into alignment with μ, resulting in a dataset of matched shapes, $\mathbf{X}'$ which are fed into the basic tangent space PCA algorithm of section 2.3.

is to sequentially add in one shape at a time to the model, each time updating our estimate of mean shape by corresponding the current mean shape to the new training shape. At the end of this iterative process, we then re-correspond each training shape to the mean shape, this time adding skipped points only to the training shapes (and not to the mean shape), so that we then have n training shapes, all with the same number of points and in one-to-one correspondence with each other. We then use the tangent space PCA algorithm from chapter 2 to estimate mean shape and covariance. The entire model learning algorithm is shown in table 3.1.

SHAPE_LIKELIHOOD_COPAP($\mathbf{x}$,$\mathbf{S}$,k)

Input: A full shape contour, $\mathbf{x}$, a shape model $\mathbf{S}$ with mean μ and covariance Σ, and the number of principle components to keep, k.

Output: Likelihood, L.

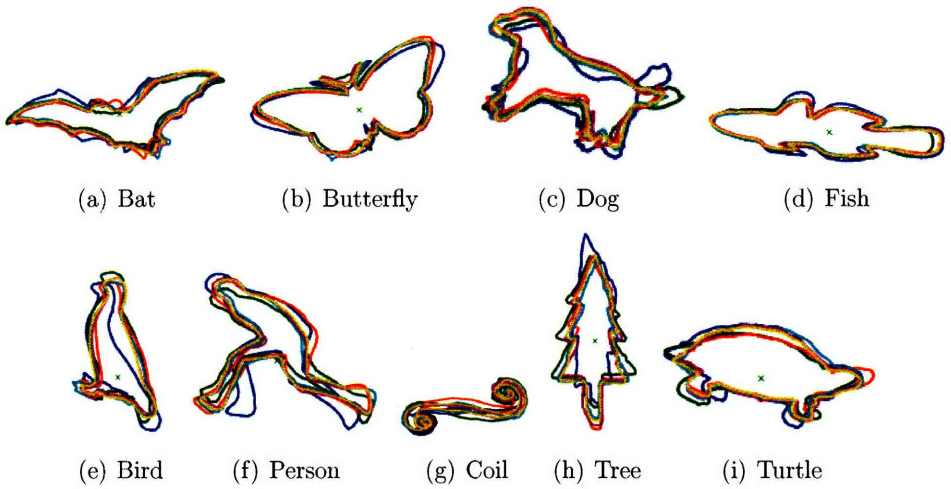
- $\phi \leftarrow$ COPAP_ASYMMETRIC($\mathbf{x}, \mu$).
- $(\mathbf{x}', \mu') \leftarrow$ ADD_SKIPPED_POINTS($\mathbf{x}, \mu, \phi$).
- Return $L \leftarrow$ SHAPE_LIKELIHOOD($\mathbf{x}', \mathbf{S}, k$).

Table 3.2: The shape likelihood algorithm, with unknown correspondences. The asymmetric variant of COPAP is used to ensure that none of $\mathbf{x}$'s points are skipped, so that the tangent space PCA likelihood computation of table 2.1 considers the full shape contour, $\mathbf{x}$.

In order to compute the likelihood of a shape, $\mathbf{z}$ with respect to a shape model, S, we must bring the shape $\mathbf{z}$ into correspondence with S's mean shape, μ. If any points are skipped on μ, we must add points to $\mathbf{z}$, as above. However, if points are skipped on $\mathbf{z}$, we face a dilemma. If we add points to μ, we must recalculate the entire shape distribution to account for the new number of points, which could be extremely computationally demanding. However, if we simply ignore $\mathbf{z}$'s skipped points, there is the very strong possibility that the resulting shape likelihood could be overly optimistic–we are allowing the likelihood function to simply ignore parts of shapes which don't match up well with the model! Instead, we choose to use the asymmetric variant of the COPAP algorithm to disallow skipped correspondences on $\mathbf{z}$. Thus, it is imperative that any new shape, $\mathbf{z}$, have fewer points than the model. The full shape likelihood algorithm is shown in table 3.2.

3.7 Experiments

Using the model learning and likelihood models of table 3.1 and table 3.2, which compute correspondences using the COPAP algorithm which we have explored in this chapter, we can now test our hypothesis that a more powerful and flexible corre-



(a) Bat (b) Butterfly (c) Dog (d) Fish

(e) Bird (f) Person (g) Coil (h) Tree (i) Turtle

Figure 3-12: Examples of learned models using our model learning algorithms. Each plotted shape is drawn according from the Gaussian model for that shape class.

spondence algorithm will increase the classification performance of our learned shape models. Recall that the cross-validated classification rate of our simple Mirrored Shifted Procrustean (MSP) metric from section 2.4 on the MPEG-7 dataset achieved a maximum value of 63.86% at $k = 6$ principle components. Unfortunately, since our algorithms were mostly written in un-optimized Matlab code, computing correspondences with COPAP on the full MPEG-7 dataset proved to be very time consuming, so we were unable to run the full cross-validated test with our improved correspondence model. (In section 7.2 we show how we hope to improve the speed of our correspondence algorithms in future work.) Thus, we ran the classification test on a single cross-validation set only, with 16 training and 4 testing shapes per class.

We learned models for each class of 16 training shapes using the LEARN_SHAPE_MODEL_COPAP algorithm (table 3.1), and we computed the maximum likelihood classes based on likelihoods computed from the SHAPE_LIKELIHOOD_COPAP function (table 3.2). Examples of learned shapes models are shown in figure 3-12. The classification rate on the test set of the tangent space PCA models learned with COPAP correspondences achieved a maximum of 79.29% with $k = 6$ principle components, compared to the MSP models' maximum classification rate on the same

	MSP-NN	MSP-PCA	COPAP-PCA
class%	95.71%	65.36%	79.29%
#pcs	-	9	6

Table 3.3: Maximum classification rates on the MPEG-7 dataset. The discriminative MSP-NN algorithm performs best, but cannot be used to perform probabilistic inference, as the generative methods can (MSP-PCA and COPAP-PCA). COPAP-PCA outperforms MSP-PCA with its more powerful correspondence algorithm, and achieves its maximum with a lower dimensional model, suggesting is has done a better job at modeling shape variation.

training and test sets of 65.36% with $k = 9$ principle components. Thus, we see a significant improvement–13.93% over the MSP correspondence model–by using COPAP correspondences with the PLSD point assignment likelihood. Table 3.3 shows the classification performance of the three classification algorithms we have seen so far: mirrored shifted Procrustean nearest neighbor (MSP-NN), mirrored shifted Procrustean tangent space principle components analysis (MSP-PCA), and COPAP tangent space principle components analysis (COPAP-PCA), all on the same training and testing sets of 16 and 4 shapes per class, respectively.

3.8 Related Work

The Procrustean Local Shape Distance described in this chapter is similar in nature to Belongie's shape contexts [2], in that both methods directly use the observed geometry of the contour in order to compute the shape signature of a point. However, shape contexts consider the relative global shape with respect to each point, while the PLSD captures only local shape information. We chose a local descriptor in this work because we wish to model global deformations in shape, so our cost function must be robust to global shape change. We do use a global model of deformation, which is in contrast to the local triangulation model of [15] or the pairwise model of [13]. The canonical shape model to use a multi-resolution technique is the curvature scale space model [39]. Our shape correspondence framework is directly taken

from [48]. The specific graphical model interpretations are our own contributions, as well as the use of the PLSD likelihood. Many other approaches have been taken to finding correspondences between shapes, for example using relaxation labeling [44]. Cremers et. al. [9] use kernel PCA to represent non-Gaussian shape distributions, while Felzenszwalb [15] uses Procrustean distributions over triangulations of object contours to achieve greater robustness with respect to elastic deformations. Our sequential model learning algorithm could be improved by using a method such as [13], who use hierarchical model merging to train shape models using Markov Random Fields (MRFs).

Chapter 4

Shape Completion

4.1 Shape Completion

We now turn to the problem of estimating the complete geometry of an object from an observation of part of its contour. We phrase this as a maximum likelihood estimation problem, estimating the missing points of a shape with respect to the Gaussian tangent space shape distribution, and we assume (for now) that the partial shape has already been brought into correspondence with the model via the algorithms of chapter 3.

Let us represent a shape as:

$$\mathbf{z} = [\mathbf{z}_1 \ \mathbf{z}_2]^T \tag{4.1}$$

where $\mathbf{z}_1 = \mathbf{m}$ contains the p points of our partial observation of the shape, and $\mathbf{z}_2$ contains the $n - p$ unknown points that complete the shape. Given a shape distribution $\mathbf{S}$ on n points with mean μ and covariance matrix Σ, and given $\mathbf{z_1}$ containing p measurements ($p < n$) of our shape, our task is to infer the last $n - p$ points which maximize the joint likelihood, $P_{\mathbf{S}}(\mathbf{z})$. (Thus, we implicitly assume that correspondences from the partial shape $\mathbf{z}$ to the model $\mathbf{S}$ are known–we later show how to compute partial shape correspondences in order to relax this assumption.)

In order for us to transform our completed vector, $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2)^T$, into a pre-shape, we must first normalize translation and scale. However, this cannot be done without

knowing the last $n - p$ points. Furthermore, the Procrustes minimizing rotation from $\mathbf{z}$'s pre-shape to μ depends on the missing points, so any projection into the tangent space (and corresponding likelihood) will depend in a highly non-linear way on the location of the missing points. We can, however, compute the missing points $\mathbf{z}_2$ given an orientation and scale. This leads to an iterative algorithm that holds the orientation and scale fixed, computes $\mathbf{z}_2$ and then computes a new orientation and scale given the new $\mathbf{z}_2$. The translation term can then be computed from the completed contour $\mathbf{z}$.

We derive $\mathbf{z}_2$ given a fixed orientation θ and scale α in the following manner. For a complete contour $\mathbf{z}$, we normalize for orientation and scale using

$$\mathbf{z}' = \frac{1}{\alpha} R_\theta \mathbf{z} \tag{4.2}$$

where R_θ is the rotation matrix of θ. To center $\mathbf{z}'$, we then subtract off the centroid:

$$\mathbf{w} = \mathbf{z}' - \frac{1}{n} C\mathbf{z}' \tag{4.3}$$

where C is the $2n \times 2n$ checkerboard matrix,

$$C = \begin{bmatrix} 1 & 0 & \cdots & 1 & 0 \\ 0 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & \cdots & 1 & 0 \\ 0 & 1 & \cdots & 0 & 1 \end{bmatrix}. \tag{4.4}$$

Thus $\mathbf{w}$ is the centered pre-shape. Now let M be the matrix that projects into the tangent space defined by the Gaussian distribution (μ, Σ):

$$M = I - \mu\mu^T \tag{4.5}$$

The Mahalanobis distance with respect to $\mathbf{S}$ from $M\mathbf{w}$ to the origin in the tangent

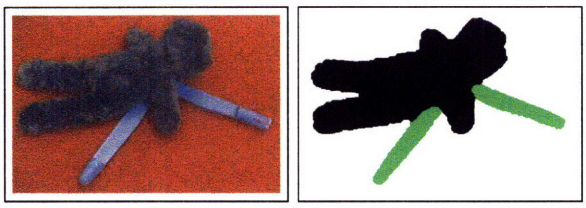


Figure 4-1: An example of occluded objects, where the bear occludes the compass. (a) The original image and (b) the image segmented into (unknown) objects. The contour of each segment must be matched against a known model.

space is:

$$d_\Sigma = (M\mathbf{w})^T \Sigma^{-1} M\mathbf{w} \tag{4.6}$$

Minimizing d_Σ is equivalent to maximizing $P_\mathbf{s}(\cdot)$, so we continue by setting $\frac{\partial d_\Sigma}{\partial \mathbf{z}_2}$ equal to zero, and letting

$$W_1 = M_1(I_1 - \frac{1}{n}C_1)\frac{1}{\alpha}R_\theta^1 \tag{4.7}$$

$$W_2 = M_2(I_2 - \frac{1}{n}C_2)\frac{1}{\alpha}R_\theta^2 \tag{4.8}$$

where the subscripts "1" and "2" indicate the left and right sub-matrices of M, I, and C that match the dimensions of $\mathbf{z}_1$ and $\mathbf{z}_2$. This yields the following system of linear equations which can be solved for the missing data, $\mathbf{z}_2$:

$$(W_1\mathbf{z}_1 + W_2\mathbf{z}_2)^T \Sigma^{-1} W_2 = 0 \tag{4.9}$$

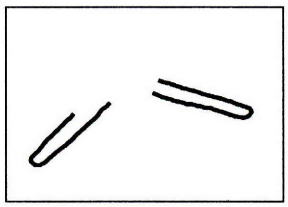
As described above, equation (4.9) holds for a specific orientation and scale. We can then use the estimate of $\mathbf{z}_2$ to re-optimize θ and α and iterate. Alternatively, we can simply sample a number of candidate orientations and scales, complete the shape of each sample, and take the completion with highest likelihood (lowest d_Σ).

To design such a sampling algorithm, we must choose a distribution from which to sample orientations and scales. One idea is to match the partial shape, $\mathbf{z}_1$, to the partial mean shape, μ_1, by computing the pre-shapes of $\mathbf{z}_1$ and μ_1 and finding the Procrustes fitting rotation, θ^*, from the pre-shape of $\mathbf{z}_1$ onto the pre-shape of μ_1. This

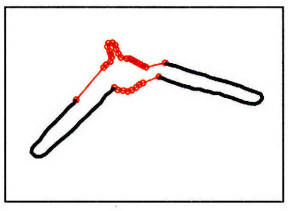
angle can then be used as a mean for a von Mises distribution (the circular analog of a Gaussian) from which to sample orientations. Similarly, we can sample scales from a Gaussian with mean α_0–the ratio of scales of the partial shapes $\mathbf{z}_1$ and μ_1 as in

$$\alpha_0 = \frac{\|\mathbf{z}_1 - \frac{1}{p}C_1\mathbf{z}_1\|}{\|\mu_1 - \frac{1}{p}C_1\mu_1\|}. \tag{4.10}$$

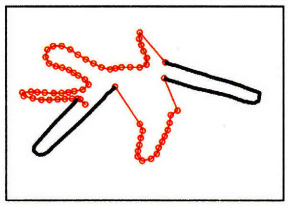
Any sampling method for shape completion will have a *scale bias*–completed shapes with smaller scales project to a point closer to the origin in tangent space, and thus have higher likelihood. In our experiments this scale bias has not appeared to provide any obvious errors in shape completion, although more testing and analysis are needed to determine the precise effect of the scale bias on the quality of shape completions.



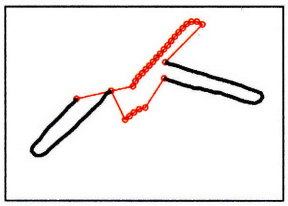
(a) Partial contour to be completed



(b) Completed as compass



(c) Completed as stuffed animal

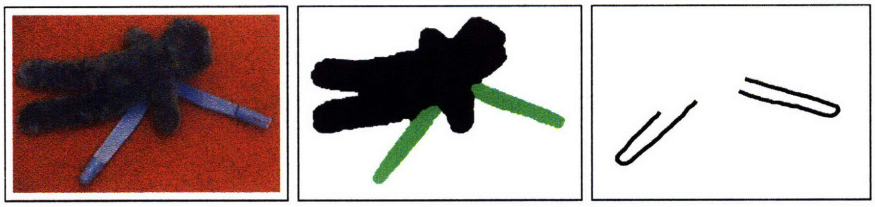


(d) Completed as jump rope

Figure 4-2: Shape completion of the partial contour of the compass in figure 4-1. Note that the correct completion (b) captures the knob in the top of the compass. The hypothesized completions in (c) and (d) lead to very unlikely shapes.

4.1.1 Partial Shape Class Likelihood

The most obvious approach to partial shape class likelihood is to simply complete the missing portion of the partial shape corresponding to $\mathbf{m}$ with respect to each shape



(a) (b) (c)

Figure 4-3: An example of occluded objects, where the bear occludes the compass. (a) The original image and (b) the image segmented into (unknown) objects. (c) Partial contour segments to be completed.

class as above (figure 4-2), then classify the completed shape. To make this concrete, let $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2\}$ be the completed shape, where $\mathbf{z}_1$ is the partial shape corresponding to measurement $\mathbf{m}$, and $\mathbf{z}_2$ is unknown. The probability of the class given the observed part of the contour $\mathbf{z}_1$ is then

$$P(C_i|\mathbf{z}_1) = \frac{P(C_i, \mathbf{z}_1)}{P(\mathbf{z}_1)} \propto \int P(C_i, \mathbf{z}_1, \mathbf{z}_2) d\mathbf{z}_2 \tag{4.11}$$

Rather than marginalize over the hidden data, $\mathbf{z}_2$, we can approximate this marginal with an estimate $\hat{\mathbf{z}}_2$, the output of our shape completion algorithm, yielding:

$$P(C_i|\mathbf{z}_1) \approx \eta \cdot P(\mathbf{z}_1, \hat{\mathbf{z}}_2|C_i) \tag{4.12}$$

where η is a normalizing constant (and can be ignored during classification), and $P(\mathbf{z}_1, \hat{\mathbf{z}}_2|C_i)$ is the complete shape class likelihood of the completed shape.

4.1.2 Unknown Correspondences in Shape Completion

Up until now we have assumed that the partial shape to be completed has already been brought into correspondence with the shape model. We will now show how to relax this assumption.

In order to calculate the maximum likelihood shape completion of a partial shape, $\mathbf{z}$ with respect to a shape model $\mathbf{S}$, we must know which points in the model correspond to the observed points of $\mathbf{z}$. In practice, $\mathbf{z}$ may contain multiple disconnected

contour segments which must be connected with hidden contour segments to form a complete contour–take for example, the two compass handles in figure 4-3. Before hidden contour segments can be inferred between the handles, the observable contours must be ordered. In many cases, there may not be a unique way to order the observed contour segments; however, we can constrain the connection ordering by noting that the interiors of all the observed object segments must remain on the interior of any completed shape. For most real-world cases, this topological constraint is enough to identify a unique connection ordering; in cases where the ordering of components is still ambiguous, a search process through the orderings can be used to identify the most likely correspondences.

Given a specific ordering of observed contour segments, we can adapt our graphical model to compute the correspondence between an ordered set of partial contour segments and a model mean shape, μ. First, we add a set of hidden, or "wildcard" points connecting the partial contour segments. This forms a complete contour, $\mathbf{z}_c$, where some of the points are hidden and some are observed. We then run a modified COPAP algorithm, where the only modification is that all "wildcard" points on $\mathbf{z}_c$ may be assigned to any of μ's points with small, fixed cost ϵ. (We must still pay a penalty of λ for skipping hidden points, however.) Recall from section 3.5.3 that we called this COPAP variant "COPAP_WILD".

In order to identify how large the hidden contour is (and therefore, how many hidden points should be added to connect the observed contour segments), we use the insight that objects of the same type generally have a similar scale. We can therefore use the ratio of the observed object segment areas to the expected full shape area to (inversely) determine the expected ratio of hidden points to observed points. In our experiments, we have found that the quality of partial shape completions depends on correctly estimating the number of hidden points; if no size priors are available, one may also perform multiple completions with varying hidden points ratios, and select the best completion using a generic prior such as the minimum description length (MDL) criterion.

The first shape completion algorithm for partial shapes with unknown correspon-

dences that we tried was to simply correspond the partial contour to the model according to the above method, and then to complete the partial contour given the computed correspondences[1]. However, we found that this procedure was often insufficient to ensure a good completion. Since we were relying on the algorithm perfecting the partial shape correspondences the first time around, any mistakes in the data association would show up in the resulting completion. Instead, we found that an iterative procedure worked best–first, compute the partial shape correspondences and initial completion as above; then re-compute the shape correspondences given the completion; then complete the shape with respect to the new correspondences, etc. This iterative procedure was found to significantly improve the performance of our shape completion algorithm. The full shape completion algorithm with unknown correspondences is shown in table 4.1.

[1]Note that the entries of the shape distribution mean and covariance can always be transposed in order to ensure that the first p points are observed and the last $n-p$ points are hidden.

COMPLETE_SHAPE_SEGMENTS($\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, $\mathbf{S}$, k, R)

Input: Set of M partial shape contours (polylines) $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, shape model $\mathbf{S}$ (with mean μ), number of principle components, k, and size ratio R.

Output: Completed shape, $\mathbf{y}$.

- $\mathbf{x} \leftarrow$ CONNECT_PARTIAL_CONTOURS($\mathbf{x}_1, \ldots, \mathbf{x}_M$).
- Add hidden (wildcard) points between the partial contours in proportion to the size ratio, R, resulting in $(\mathbf{x}, \mathbf{h})$, where $\mathbf{x}$ is the full shape contour with both observed and hidden points, and $\mathbf{h}$ is the hidden points mask.
- Sub-sample $\mathbf{x}$ down until it has some fraction (e.g. $1/2, 3/4, \ldots$) of the points that the model mean, μ has–taking care not to remove observed segment endpoints–and update $\mathbf{h}$.
- $\Phi \leftarrow$ COPAP_PARTIAL($\mathbf{x}, \mathbf{h}, \mu$).
- Set $\mathbf{x}' \leftarrow$ ADD_SKIPPED_POINTS($\mathbf{x}, \mu, \Phi$), and update $\mathbf{h} \rightarrow \mathbf{h}'$.
- $\mathbf{y} \leftarrow$ COMPLETE_SHAPE($\mathbf{x}', \mathbf{h}', \mathbf{S}$).
- $L \leftarrow$ SHAPE_LIKELIHOOD($\mathbf{y}, \mathbf{S}, k$).
- $\mathbf{y}_{best} \leftarrow \mathbf{y}$.
- $L_{best} \leftarrow L$.
- While $L \geq L_{best}$:
  1. $\mathbf{x} \leftarrow \mathbf{y}$.
  2. Sub-sample $\mathbf{x}$ down to a fraction of μ's points; update $\mathbf{h}$.
  3. $\Phi \leftarrow$ COPAP_SEMISYMMETRIC($\mathbf{x}, \mathbf{h}, \mu$).
  4. $\mathbf{x}' \leftarrow$ ADD_SKIPPED_POINTS($\mathbf{x}, \mu, \phi$); update $\mathbf{h} \rightarrow \mathbf{h}'$.
  5. $\mathbf{y} \leftarrow$ COMPLETE_SHAPE($\mathbf{x}', \mathbf{h}', \mathbf{S}$).
  6. $L \leftarrow$ SHAPE_LIKELIHOOD($\mathbf{y}, \mathbf{S}, k$).
  7. If $L > L_{best}$:
     - $\mathbf{y}_{best} \leftarrow \mathbf{y}$.
     - $L_{best} \leftarrow L$.
- Return $\mathbf{y} \leftarrow \mathbf{y}_{best}$.

Table 4.1: The partial shape completion algorithm.

Chapter 5

Boosted Shape Parts

Recall from section 1.2 that in this thesis we are primarily interested in shape inference tasks which require us to accurately model the variation of a class of objects. In particular, we have spent a great deal of time developing a framework for the completion and classification of partially occluded shapes. However, performing partial shape classification has thus far required the correspondence, completion, and likelihood computation of a given partial shape with respect to *each and every* model in a database of shape models. In this chapter, we explore the use of learned, discriminative models, for pruning away unlikely classifications before we must perform the full, probabilistic analysis.

Specifically, we present a method for learning a discriminative classifier of partial object shape, called *Procrustean Boosted Shape Parts (PBSP).* We use a variation of AdaBoost [16] based on Procrustean shape analysis, with one-vs-all logistic regression as the weak classifier. As an added bonus, we will show that by learning models from training data, we can achieve improved performance on shape retrieval over all previous (non-learning) methods. We will also set new benchmarks for a partial shape classification task. Ultimately, the work in this chapter fits into the overall shape analysis system in this thesis as a first-pass pruning method towards detecting, segmenting, and completing the boundaries of partially occluded objects, as in Figure 5-1. Once our partial shape detector is used to prune away unlikely classifications, more sophisticated methods which take into account the whole scene

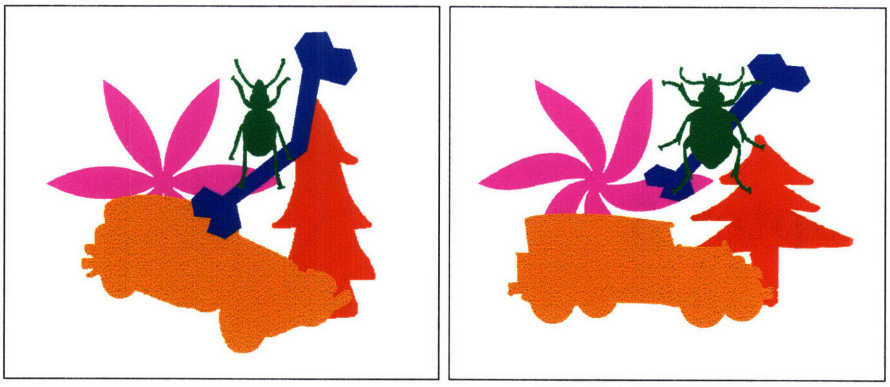


Figure 5-1: Two cluttered scenes, containing the same set of object types (drawn from the MPEG-7 shape dataset), but which vary in shape. We wish to classify all of the partially-hidden objects, based purely on shape geometry.

geometry can be used to form a final analysis of the image.

5.1 A Boosted Detector of Shape Parts

We will use a parts-based model, where the *part match* of a shape $\mathbf{x}$ to a part $\mathbf{y}$ is given by the minimum Procrustean shape distance between $\mathbf{y}$ and all the local (contiguous) features on $\mathbf{x}$. In order to be robust to noise and differently-sized features, we consider parts of varying sizes, and across multiple levels in image scale space. However, in contrast to many previous approaches to modeling shape parts using properties of a single contour (such as curvature), we will use machine learning (AdaBoost) to pick the most discriminative shape parts for a given training set.

5.1.1 Part Match

The Procrustean metric of chapter 2 assumes that $\mathbf{z}$ and $\mathbf{w}$ are two complete shapes. We can generalize the same metric to the distance between a contour segment or part and a complete shape. Given a shape contour part $\mathbf{z}$ with p points, and a complete shape $\mathbf{x}$ with n points, we define the *shape part match* of $\mathbf{x}$ to $\mathbf{z}$, $r(\mathbf{x}, \mathbf{z})$, to be the minimum Procrustean shape distance from $\mathbf{z}$ to any contiguous contour fragment of

length p on $\mathbf{x}$, or

$$r(\mathbf{x}, \mathbf{z}) = \min_i d_P((\mathbf{x_i}, ..., \mathbf{x_{i+p-1}}), \mathbf{z}) \tag{5.1}$$

(We assume that indexing is modular to avoid ugly wrap-around notation.) Note that a *low* shape part match here indicates a good match, while a *high* shape part match indicates a bad match.

5.1.2 AdaBoost

We are inspired in our approach by the Viola-Jones object detector [61], which uses boosting to select discriminative features in a cascaded, one-vs-all classification system. The Viola-Jones classifier uses millions of simple, rectangular features (blocks of 0's and 1's) which are convolved with images to generate feature responses–the higher the response, the better the match to a given feature. AdaBoost is used to learn a classifier based on boosting weak classifiers trained on the image features. The Viola-Jones classifier specifically uses decision stumps (a decision tree with depth 0) as the weak classifier, although we will see that decision stumps are prone to over-fitting in the shape classification problem. At each iteration in the boosting algorithm, the most discriminative feature on a weighted set of training images is selected for the next decision stump, and the decision stump is added to the classifier with weight in proportion to the discriminative power of the selected feature. The weights on the images in the training set are then updated based on the feature that was just added.

The power of Adaboost lies in how the weights on the training observations are updated; observations that are classified correctly receive *lower* weight on the next iteration, while observations that are classified incorrectly receive *higher* weight. The weights bias the next weak learner towards the observations which are close to the margin. The generic, discrete version of Adaboost is summarized in table 5.1.

5.1.3 Procrustean Boosted Shape Parts

To begin with, let C be the boosting class, i.e., the class we are trying to detect, and let all the training shapes which belong to C be labeled with a "1" (and all the

ADA_BOOST($\mathbf{X}, \mathbf{c}, A$)

Input: Training data $\mathbf{X}$, labels $\mathbf{c} \in \{0, 1\}$, and weak learning algorithm, A.

Output: A classifier, $H(x)$.

- Initialize weights $\mathbf{w}$ on training data.
- For $t = 1, 2, \ldots, T$:
  1. Generate a weak classifier, H_t, by training the weak learner, A, on the weighted training data.
  2. Update the weights, $\mathbf{w}$, based on the performance of H_t on the training data; elements which are correctly classified get their weight reduced, while elements which are misclassified by H_t get increased weight.
  3. Add H_t to the ensemble of weak classifiers, $\{H_1, \ldots, H_{t-1}\}$ with weight α_t in proportion to H_t's classification rate.
- Output the final classifier: $H(x) = \sum_{t=1}^{T} \alpha_t H_t(x)$.

Table 5.1: A high-level overview of the Adaboost algorithm.

rest with a "0"). Additionally, the weights of the training images w_i are initialized to be uniform. At each iteration of the PBSP boosting algorithm, we consider *every* possible contour fragment of every training shape in the boosting class, C, at a range of sizes and levels in image scale space. For each contour fragment, or feature $\mathbf{z}^{(\mathbf{i})}$, we compute a decision stump that classifies the images with respect to class C based on the Procrustean shape part metric (equation (5.1)) between $\mathbf{z}^{(\mathbf{i})}$ and each complete training image. We retain the decision stump that has the highest weighted accuracy.

Logistic Regression

At each round of boosting, we select the most discriminative shape part to add to the model, given the current weights on the training data. The weak learner must therefore handle two responsibilities: (1) select the most discriminative feature, and (2) fit a class membership likelihood function to the weighted training data if *only* that single feature was used for classification. In Viola-Jones, both (1) and (2) are

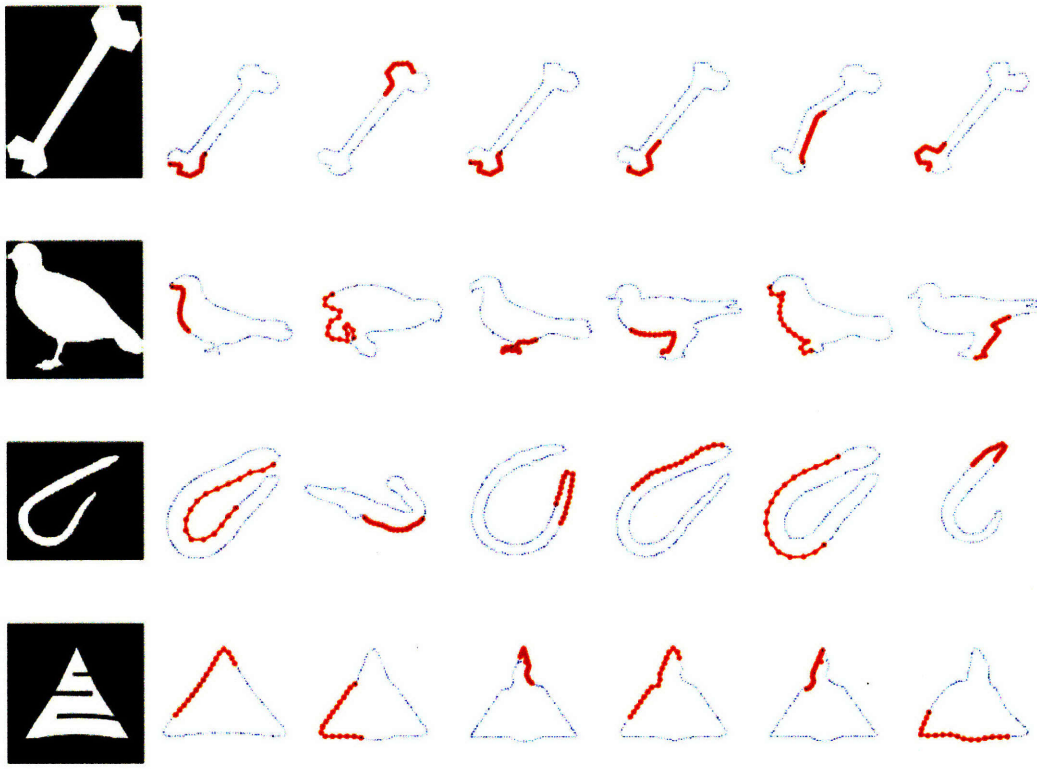


Figure 5-2: Boosted shape parts. In the leftmost column are four shapes from different categories in the MPEG-7 Shape B dataset: "Bone", "bird", "snake", and "device4". In the columns to the right are the first six boosted shape parts chosen by the PBSP algorithm. Shape parts are shown with thick red points, while the remaining portions of the contour they were chosen from is shown in the thin, dotted blue line. Note that in the triangle (device4) class, the top five most discriminative features were chosen to be at a lower, smoothed resolution in image scale space.

computed using decision stumps. Unfortunately, although decision stumps are a good choice for identifying discriminative features, they are also prone to over-fitting without large training data sets. In Figure 5-3, we see that the decision stump classifier incorrectly classifies two validation shapes in the detection class, placing two green circles to the right of the decision boundary.

Experimentally, we found that the smooth probabilities given by logistic regression improved the final, boosted classification rates in contrast to decision stumps. In Figure 5-3, logistic regression yields a probability curve which gives probability of at least 30% to the two validation set outliers that were incorrectly rejected by the

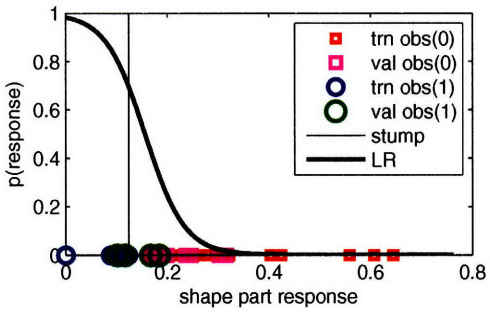


Figure 5-3: Logistic regression vs. decision stumps as a weak classifier of shape part responses, learned for the first "Bone" shape part in Figure 5-2. The decision stump classifier incorrectly classifies two validation shapes in the detection class for this shape, placing two green circles to the right of the decision stump's boundary.

decision stump. Decision stumps can be trained with only one pass over the sorted list of responses, while logistic regression requires a slower, iterative algorithm. We therefore use decision stumps to identify the best feature to boost on each iteration, and then fit a Logistic Regression (LR) model to the weighted set of training image responses. Although logistic regression is slower to learn, by using a better learner we see improved overall learning performance.

Finally, given N object classes, we learn a single boosted one-vs-all classifier for each class, then compute the probability that a new shape, $\mathbf{x}$, belongs to a given class, C, in proportion to the class membership likelihood,

$$P(C|\mathbf{x}) = \alpha \cdot p(\mathbf{x}; M_C) = \alpha \cdot \prod_i \exp f_i(\mathbf{x}; M_{i,C}) \tag{5.2}$$

where $f_i(\mathbf{x})$ is the feature response of the learned logistic, $p(\mathbf{x}; M_C)$ is the corresponding one-vs-all probability of shape $\mathbf{x}$ under logistic regression model M_C, C is the indicator variable for class membership of the random variable $\mathbf{x}$, and α is a normalization constant.

One must be careful, however, as logistic regression can also be prone to overfitting if the training observations are well-separated. Thus, we use a regularized version of logistic regression, which penalizes high "slopes" in the probability curve [47].

Logistic regression (for binary classification) attempts to fit a probability curve of the form

$$P(C = 1|r) = \frac{1}{1 + e^{-(a+br)}}$$

to the training data, where r is the shape part match distance between the shape $\mathbf{x}$ and the selected feature, as in equation (5.1). Regularized logistic regression has data log-likelihood

$$\ell_\lambda(a, b) = \sum_{i=1}^{N} [c_i(a + br_i) - \log(1 + e^{a+br_i})] - \frac{\lambda}{2} b^2$$

where the c_i's are the training observation labels (1 or 0), and the r_i's are the shape part match distances to individual features $\mathbf{x}_i$ of the training data. A validation training set is used to select the best penalty, λ (by learning LR models with several different λ's and choosing the model which maximizes the validation set data likelihood). From a Bayesian point of view, the regularization penalty, λ, defines a zero-mean Normal prior on parameter b with variance $\frac{1}{\lambda^2}$.

5.1.4 The PBSP Algorithm

The PBSP algorithm is shown in Table 5.2. Given training and validation observations, $(\mathbf{x_t^{(1)}}, c_t^{(1)})$, ... , $(\mathbf{x_t^{(N)}}, c_t^{(N)})$ and $(\mathbf{x_v^{(1)}}, c_v^{(1)})$, ... , $(\mathbf{x_v^{(M)}}, c_v^{(M)})$, where $\mathbf{x_t^{(i)}}$ and $\mathbf{x_v^{(i)}}$ are shapes and $c_t^{(i)}$ and $c_v^{(i)}$ are class membership labels, and given N_0 and N_1 are the number of negative and positive training observations, respectively, the PBSP algorithm uses Real AdaBoost [16] with penalized logistic regression [47] as the weak learner to learn a set of feature response functions, $f_1(\mathbf{x})$, ..., $f_T(\mathbf{x})$ in lines 1-5. The total response function $f(\mathbf{x})$ is simply the sum $\sum_{t=1}^{T} f_t(\mathbf{x})$ of the T feature responses. The final class likelihood function is computed using a second round of LR on the training and validation set total responses.

5.1.5 Boosted Ensembles

Our weak learner (regularized logistic regression) requires a validation set in addition to the training set. This requirement yields a natural method for generating an

LEARN_PBSP($\mathbf{X_t}, \mathbf{c_t}, \mathbf{X_v}, \mathbf{c_v}$)

Input: Training shapes and labels $\mathbf{X_t} = \{\mathbf{x_t^{(1)}}, \ldots, \mathbf{x_t^{(N)}}\}$ and $\mathbf{c_t} = \{c_t^{(1)}, \ldots, c_t^{(N)}\}$, and validation shapes and labels $\mathbf{X_v} = \{\mathbf{x_v^{(1)}}, \ldots, \mathbf{x_v^{(N)}}\}$ and $\mathbf{c_v} = \{c_v^{(1)}, \ldots, c_v^{(N)}\}$.

Output: Set of feature response functions, $\{f_1(\mathbf{x}), \ldots, f_T(\mathbf{x})\}$.

- Initialize training data weights $w_i = \frac{1}{2N_0}, \frac{1}{2N_1}$ for $c_i = 0, 1$, respectively.
- Repeat for $t = 1, 2, ..., T$:
  1. For each positive training shape part, $\mathbf{z_j} \subset \mathbf{x_t^{(i)}} \forall i$ s.t. $c_t^{(i)} = 1$, compute the shape part match scores $r_j^{(i)} = r(\mathbf{x_t^{(i)}}, \mathbf{z_j})$.
  2. Fit a decision stump to the weighted match scores for each $\mathbf{z_j}$ and choose the shape part $\mathbf{z^*}$ whose decision stump has the lowest weighted training set error.
  3. Using the validation data to select λ, fit a penalized LR classifier on the weighted training responses to $\mathbf{z^*}$ to obtain a class probability estimate $p_t(\mathbf{x}) = \hat{P}_w(c = 1|\mathbf{x}; \mathbf{z^*}) \in [0, 1]$.
  4. Set $f_t(\mathbf{x}) \leftarrow \frac{1}{2} \log \frac{p_t(\mathbf{x})}{1 - p_t(\mathbf{x})} \in \mathbb{R}$.
  5. Set $w_i \leftarrow w_i \exp[-c_i f_t(\mathbf{x_i})], i = 1, 2, ..., N$, and renormalize so that $\sum_i w_i = 1$.
- Set the total response function, $f(\mathbf{x}) \leftarrow \sum_{t=1}^{T} f_t(\mathbf{x})$.
- (Optional) Fit a penalized LR classifier to the combined validation and training total responses $f(\mathbf{x^{(i)}})$ to obtain the binary class likelihood function $p(\mathbf{x}) = \hat{P}(y = 1|\mathbf{x})$.

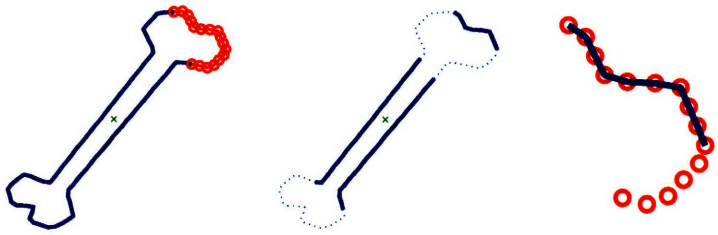
Table 5.2: The PBSP Algorithm learns a one-vs-all shape classifier using training and validation observations.

ensemble of boosted classifiers. We use cross-validation (CV) to break up our training data into 5 pairs of training and validation sets. For each training-validation pair (T_i, V_i) we run boosting to generate a strong classifier, B_i. Given a new shape, $\mathbf{x}$, which we wish to classify as either belonging to the boosting class, C, or not, we compute the probability of class membership, $P(C|\mathbf{x})$ as the average class membership probability over the 5 CV boosted classifiers.

5.2 Shape Classification

To classify a full shape, $\mathbf{x}$, the feature responses, $f_t(\mathbf{x})$, are computed using the minimum Procrustean shape part distance $r(\mathbf{x}, \mathbf{z})$ of equation (5.1). Since shapes are fully observed, this shape part match function is always well defined, and we can compute the class probabilities according to equation (5.2). However, if $\mathbf{x}$ is the partially occluded contour of some shape, then we cannot easily compare a learned feature $\mathbf{z}$ with every local point on $\mathbf{x}$. Recall that the distances are computed from segments of local contours that strongly predict shapes in our training data. Even though feature $\mathbf{z}$ is evaluated at a point on $\mathbf{x}$, the feature $\mathbf{z}$ is still being matched against p points on $\mathbf{x}$. When evaluating $\mathbf{z}$ against a point in $\mathbf{x}$ that is near one of the ends of an incomplete contour, there may not be p points to match against.

We can therefore proceed by either computing the part match using only those local features on $\mathbf{x}$ which are completely observed (all p points are visible), or by computing a *partial shape part match* function, which computes the similarity between a fully-observed shape part, $\mathbf{z}$, and *all* local (contiguous) features on $\mathbf{x}$, whether they are partially hidden or not. The main difficulty with matching only the fully-observed features on $\mathbf{x}$ is that if $\mathbf{x}$ is short, large (and highly informative) deformations of the shape towards the ends of the partial contour will tend to be ignored in the classification process. As a result, short partial contours tend to resemble all other shapes equally. Consider the example in Figure 5-4, where the shape part on the left is a good match to the top of the partially-hidden shape in the center. However, since there are not enough observed points at the top of the center shape, we must use a



(a) shape part (b) occluded shape (c) part match

Figure 5-4: Partial shape part match. (a) A shape part which we wish to detect, (b) a partially-occluded shape (occluded parts shown with a thin, dotted line), and (c) the correct part match. We would like to form a shape similarity metric which can handle missing/occluded points.

partial shape part match function in order to see that the given shape part is similar to the top of the occluded shape.

5.2.1 Similarity of Partially-Occluded Shapes

Let $\mathbf{y}$ denote a local point vector on $\mathbf{x}$, and $\mathbf{z}$ is a learned feature. Assuming $\mathbf{y}$ and $\mathbf{z}$ are fully observed vectors of p points each, we can normalize their positions and scales, and compare them using the Procrustean distance. However, if $\mathbf{y}$ has k missing points (out of p), then we can still compute the distance between $\mathbf{y}$ and $\mathbf{z}$ by taking an expectation over the missing parts. Let $\mathbf{y}_{obs}$ be the observed portion of $\mathbf{y}$, and let $\mathbf{z}_{obs}$ be the corresponding portion of $\mathbf{z}$. Similarly, $\mathbf{y}_{hid}$ and $\mathbf{z}_{hid}$ are the hidden part of $\mathbf{y}$, and corresponding part of $\mathbf{z}$. (Note that $\mathbf{z}_{hid}$ is actually observed–the *hid* subscript is simply to indicate the correspondence to $\mathbf{y}_{hid}$.) The expected distance between $\mathbf{y}$ and $\mathbf{z}$ can be written as

$$E_{\mathbf{y}_{hid}|\mathbf{y}_{obs},\mathbf{z}}[d_p(\mathbf{y},\mathbf{z})] = d_P(\mathbf{y}_{obs},\mathbf{z}_{obs}) + E_{\mathbf{y}_{hid}|\mathbf{y}_{obs},\mathbf{z}}\left[d_P(\mathbf{y},\mathbf{z}) - d_P(\mathbf{y}_{obs},\mathbf{z}_{obs})\right],$$

that is, the Procrustean distance between the observed portions of the two shapes, and an expectation over the additional distance incurred by occluded components of

the shape feature.

Note that the expectation is taken with respect to $\mathbf{y}_{hid}$, and the distribution is conditioned on $\mathbf{y}_{obs}$ and the feature $\mathbf{z}$. This distribution is high dimensional and requires considerable training data to learn accurately. In practice, we found a good approximation to the expected distance was to learn a function approximator for the expected distance directly, conditioned only on the number of visible and occluded points. Piece-wise linear regression performed well at approximating this function, such that for k missing points out of p total,

$$E_{\mathbf{y}_{hid}|\mathbf{y}_{obs},\mathbf{z}}\left[d_p(\mathbf{y},\mathbf{z}) - d_p(\mathbf{y}_{obs},\mathbf{z}_{obs})\right] \approx \gamma(\hat{d})\left(\frac{k}{p}\right)$$

where $\hat{d} = d_P(\mathbf{y_{obs}}, \mathbf{z_{obs}})$. Thus, we define the *partial Procrustean shape distance* between a partially observed shape $\mathbf{y_{obs}}$ and a fully observed shape $\mathbf{z}$ as

$$d_P^*(\mathbf{y_{obs}}, \mathbf{z}) = \hat{d} + \gamma(\hat{d})\left(\frac{k}{p}\right). \tag{5.3}$$

We call $\gamma(\cdot)$ the *occlusion penalty*, and we model $\gamma(\cdot)$ as a piece-wise linear function by scaling the linear regression model with the distance between the observable components to ensure that the occlusion penalty is independent of the overall distances involved. To estimate the occlusion penalty model from training data, we take m randomly sampled pairs of local shape parts, $(\mathbf{y^{(1)}}, \mathbf{z^{(1)}}), ..., (\mathbf{y^{(m)}}, \mathbf{z^{(m)}})$, from the training set and compute the difference in shape distances,

$$\Delta^{(i)} = d_P(\mathbf{y^{(i)}}, \mathbf{z^{(i)}}) - d_P(\mathbf{y_{obs}^{(i)}}, \mathbf{z_{obs}^{(i)}})$$

for each possible number k occluded points out of p total points. We occlude points in a single contiguous segment, at the end of $\mathbf{y^{(i)}}$, since most real scenes have occlusions which extend over multiple neighboring boundary points. (In other words, the event that $\mathbf{x_{i+1}}$ is occluded is highly correlated with the event that $\mathbf{x_i}$ is occluded.)

We discretize $\hat{d}$ into a large set of evenly-spaced values over the full range of Procrustean distances between the two features (from 0 to $\pi/2$), and for each $\hat{d}$ we

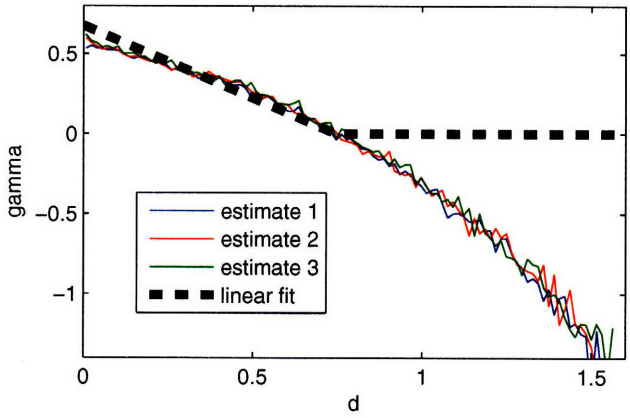


Figure 5-5: Piece-wise linear regression of the occlusion penalty, γ, with respect to the observed (partial) shape distance, d.

regress Δ as a function of $\frac{k}{p}$, with slope $\gamma(\hat{d})$ and intercept zero:

$$\Delta = \gamma(\hat{d}) \cdot \left(\frac{k}{p}\right).$$

Finally, to avoid negative occlusion penalties (in the case of high observed shape distances, $\hat{d}$), we introduce a second linear component to the model, such that for some value d_γ, if $\hat{d} \geq d_\gamma$, then $\gamma(\hat{d}) = 0$. We find the lowest $\hat{d}$ value, d_γ, such that $\gamma(\hat{d}) \leq 0$, and we fit a two-piece linear model to $\gamma(\hat{d})$, resulting in

$$\gamma(\hat{d}) = \begin{cases} s_\gamma \cdot (\hat{d} - d_\gamma) & \text{if } \hat{d} < d_\gamma \\ 0 & \text{o.w.} \end{cases}.$$

Experimentally, we found that this two-stage linear fit for γ is well-justified. In Figure 5-5, we see the second stage regression of γ vs. $\hat{d}$, with three estimates from the first stage regression of Δ vs. $\frac{k}{p}$ shown together to illustrate the low variance of these estimates.

To compute the partial shape part match function, we simply replace the Procrustean distance, $d_P(\cdot)$, with the partial Procrustean distance, $d_P^*(\cdot)$, in equation (5.1),

yielding

$$r^*(\mathbf{x}, \mathbf{z}) = \min_i d_P^*((\mathbf{x_i}, ..., \mathbf{x_{i+p-1}}), \mathbf{z}) \tag{5.4}$$

where some of the points $\mathbf{x_j}$ may be hidden.

5.3 Shape Retrieval

Although the main focus of our parts-based model is on the detection and classification of partially-hidden shapes, we can also use our learned shape models for the task of shape retrieval. Given a new shape, $\mathbf{v}$, the problem of shape retrieval is to find all the shapes in a database, $\mathbf{X}$ which are most similar to $\mathbf{v}$. Typically, a generic shape metric is specified ahead of time (such as the Procrustean shape distance of equation (2.7)). However, by inducing a learned shape distance on shape space, we will show improvement upon previous retrieval benchmarks on a standard shape retrieval dataset (MPEG-7).

For any shape, $\mathbf{v}$, we compute the total class response function, $f^C \leftarrow f(\mathbf{v}; C)$ (Table 5.2), with respect to each shape class, C. We then define the *shape part signature* of the shape $\mathbf{v}$ by the vector $\xi((\mathbf{v})) = \left[f^1, f^2, ..., f^N\right]$. The dissimilarity between two shape part signatures is then computed with the Euclidean distance metric.

5.4 Experimental Results

We evaluated our algorithm on the MPEG-7 shape dataset [31], the de facto standard for comparison of 2-D shape recognition algorithms. Recall that the MPEG-7 dataset has a total of 70 shape classes, with 20 shapes per class, for a total of 1400 shapes. Within many classes, there is a great deal of shape variability. Some of the variability, such as in the "device-6" (pentagon) class, may be seen as high-frequency noise in morphological image scale space, as the boundaries become very similar at a lower resolution, when the cuts are filled in by image dilation (Figure 5-6).

We extracted 100 evenly-spaced points from the boundaries of each shape in the

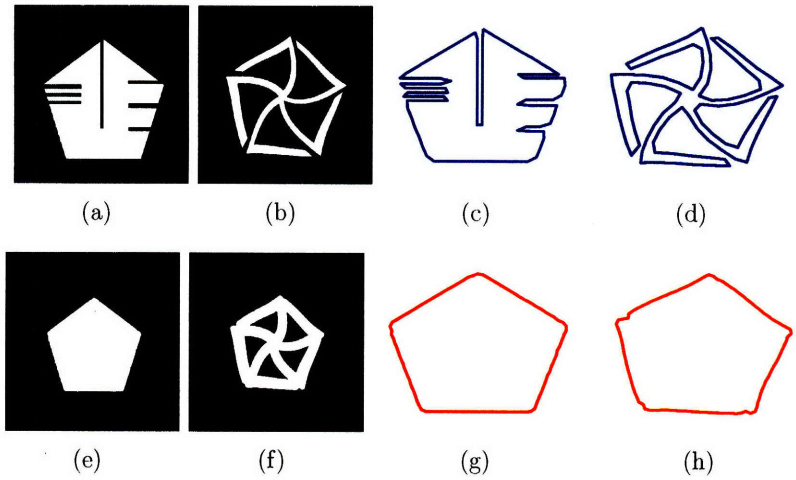


Figure 5-6: Two shapes taken from the "device6" class of the MPEG-7 dataset, which become very similar after performing an image dilation. (a-b) The original shape images. (c-d) Boundary contours extracted from the original images. (e-f) The smoothed shapes, after performing image dilation. (g-h) The boundary contours of the smoothed shapes.

MPEG-7 dataset to form our level 1 shape dataset (in image scale space). We also extracted 50 evenly-spaced points from each shape boundary after dilating each image by a 15-pixel-wide Gaussian dilation mask, to form a level 2 shape dataset.

We then trained our Procrustean Boosted Shape Parts (PBSP) recognition algorithm using shape part features of 15 points at both levels in scale space, and performed a five-fold cross-validation (CV) to estimate classification and detection/false positive rates of both full and partially-observed shapes. For classification, maximum likelihood estimation was used, while detection was performed by thresholding the class membership likelihood function for each class. In addition, we used the boosted shape models to learn a distance metric for shape retrieval.

5.4.1 Retrieval

The standard shape retrieval task on the MPEG-7 dataset is called the bullseye test. The bullseye test is a leave-one-out (LOO) test, which takes in a query shape, $\mathbf{v}$, and computes the percentage of matches out of the total number of shapes per class, m

CPF [55]	IDSC [34]	DSW [1]	**PBSP**
84.05%	85.40%	87.23%	**89.49%**

Table 5.3: Bullseye retrieval results on the MPEG-7 dataset. PBSP with shape part signatures outperforms all previous, non-learning methods.

(including the query shape), within the top $2m$ matches which belong to the same class as the query shape. (For the MPEG-7 dataset, $m = 20$, since there are 20 shapes per class.) The bullseye retrieval rate is the average bullseye rate over all 70×20 LOO tests.

After training boosted shape parts models on our five cross-validation training sets, we ran a bullseye test on the test shapes in each CV set, comparing each test shape to all 70×20 shapes in the MPEG-7 dataset. This is equivalent to the LOO bullseye test, since the CV test sets form a partition over the full MPEG-7 dataset. Using shape part signatures with Euclidean distance to compare each given test shape with all 70×20 shapes in the MPEG-7 dataset, we achieved a bullseye rate of 89.49%, an improvement over the best previously published results, which used prescribed, rather than learned distances (Table 5.3).

5.4.2 Classification

In Figure 5-7, the correct classification rate is plotted vs. the number of shape part features (rounds of boosting) used to represent each shape class. With only 3 shape parts per class, the average CV classification rate is already 90%. The highest rate is 94.07%, achieved with 11 features. As a baseline for full shape classification, we used the Mirrored Shifted Procrustean Metric, $d_{MSP}(\cdot)$ with a nearest neighbor (NN) classifier from section 2.4. This baseline classifier achieved an average CV classification rate of $95.86\% \pm 1.17\%$, which is higher than the PBSP classification rate; however, the PBSP classifier is unique in that it is designed to be robust to occlusions. The Procrustean NN classifier cannot be used to compare partially-occluded shapes.

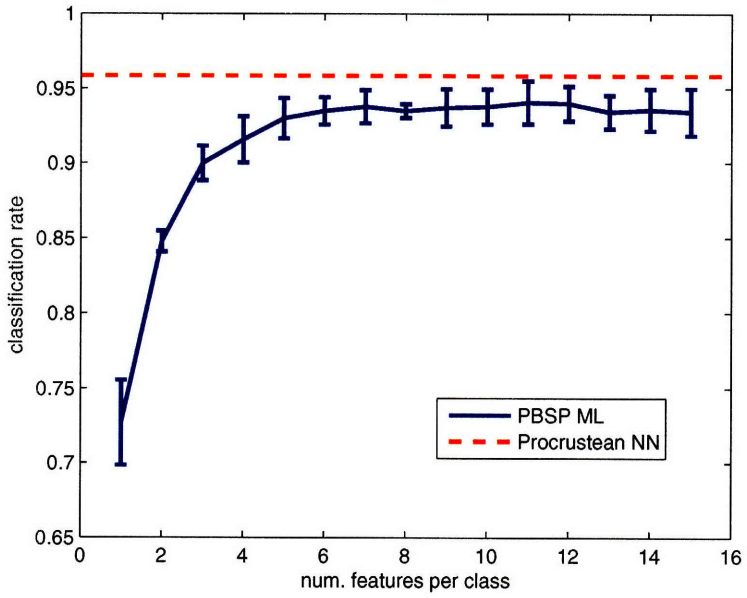


Figure 5-7: Full shape classification. Correct classification rate plotted vs. number of features (rounds of boosting) on the MPEG-7 dataset for PBSP and Procrustean NN. Error bars are drawn at 1 standard deviation.

5.4.3 Detection

Detection differs from classification in that some small number of false positives are acceptable–it is the true positives which we are trying to maximize, first and foremost (while minimizing false positives as much as possible). It makes sense to optimize detection (rather than classification rates) in the partial shapes setting of Figure 5-1. In this case, partial shape detection can be used as a pre-processing step towards a more complete image analysis (which uses the entire scene geometry), by pruning away unlikely classifications until only a few candidates remain, thereby making the search over object configurations efficient.

For detection, we investigated three different methods for setting the class likelihood thresholds from PBSP. First, we set a fixed threshold directly on the total feature responses, $f(\mathbf{x})$, from Table 5.2. Second, we used validation sets to find the minimum perfect detection rate (MPDR) threshold on the validation set, again di-

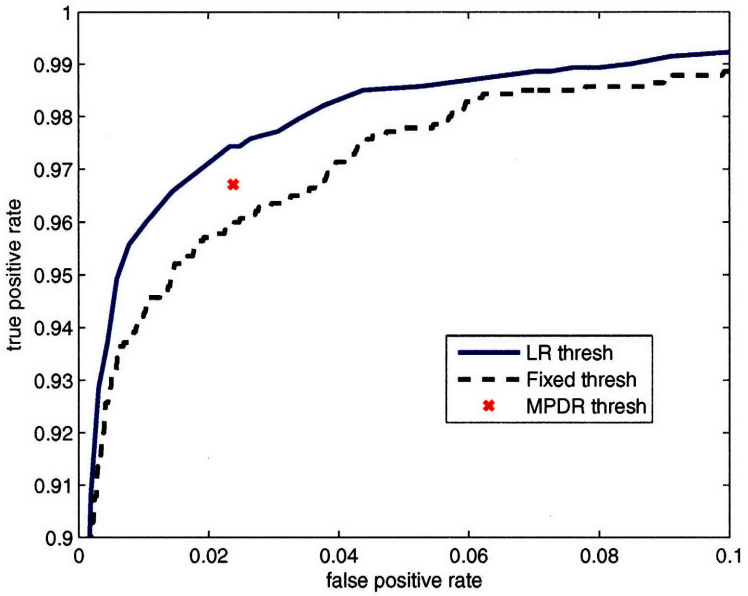


Figure 5-8: Full shape detection/false alarm rates. A comparison of three different methods for setting detection thresholds (MPEG-7).

rectly on the total feature responses. And third, we used logistic regression on the validation sets to fit likelihood functions on the total feature responses, and instead of thresholding the feature response functions, we set a threshold on the likelihoods. This third method (LR) outperformed the first two on the MPEG-7 full shape dataset (Figure 5-8).

We also used the models learned with PBSP on the full shape dataset to detect partially occluded shapes. We generated occluded shapes by taking each shape in the test set and removing a contiguous segment of points at random with size between 10% and 50% of the contour. We generated one partial shape dataset for each percentage of occluded points and then used the PBSP models together with LR thresholding to detect the partial shapes in each dataset. In Figure 5-9, we see the false positive rates plotted for each percentage of occluded points, where we fix the true positive rate at 90%. One improvement we can make to the PBSP learning algorithm for partial shape detection is to generate partially occluded shapes from the training validation

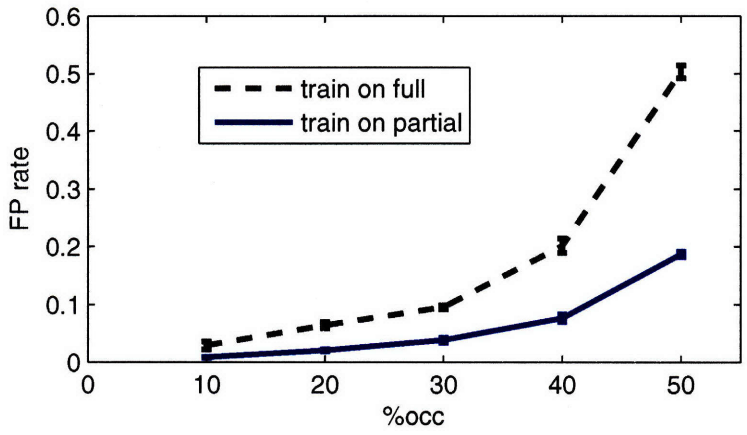


Figure 5-9: Detecting partial shapes. When fixing the true positive rate to 90%, the false positive rate is plotted vs. percent occlusion for full-shape LR validation (black dotted line), and partial-shape LR validation (blue solid line) on the MPEG-7 shape dataset. Incorporating partial shapes into the training algorithm reduces false positive rates.

sets, and then to use these partial shapes to validate the weak logistic classifiers learned in Table 5.2. In Figure 5-9 we see the effect of this partial shape validation on the resulting false positive rates. Further improvements in recognition may be made by incorporating partial shapes into the feature selection phase of boosting, as well.

5.5 Discussion and Related Work

Several attempts have been made previously at boosting shape parts for partial object detection. In [62], edgelet features are boosted to detect pedestrians and cars from images. In [42], boosting is used to generate a "Boundary-Fragment-Model" detector, and is used to detect cars, airplanes, cows, horses, and bottles. Shape contexts are boosted in [41] to learn models of hand shapes for sign language and gesture recognition.

The only other work that we are aware of on boosted shape parts to have evaluated their algorithm on the MPEG-7 dataset is [29]. However, they used only a 10-class subset of the full MPEG-7 dataset, and achieved a lower classification rate (90%)

than our algorithm achieved on the entire dataset of 70 classes. To our knowledge, we are the first to apply boosting to learn a parts-based shape classifier on such a large dataset. In [18], Ghosh and Petkov compare shape contexts [2] with distance multisets on partial shape detection under varying types of occlusions. However, they only report results on a subset of the MPEG-7 dataset, so we are unable to compare our method to theirs directly. They did show that shape contexts perform poorly under many types of occlusions, while their own method–distance multisets–performs well on a subset of the MPEG-7 dataset. But, since their results were evaluated using training data only, the actual generalization performance of their algorithm may be below what is reported. Their evaluation of shape contexts–a global shape descriptor–on partial shape recognition tasks lends further support to our assertion that local shape descriptors are most useful for partial shape analysis.

Chapter 6

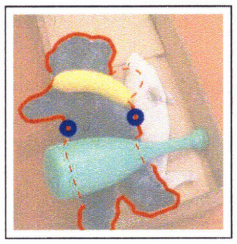
Manipulation using Probabilistic Models of Object Geometry

Robot manipulators largely rely on complete knowledge of object geometry in order to plan their motion and compute successful grasps. If an object is fully in view, the object geometry can be inferred from sensor data and a grasp computed directly. If the object is occluded by other entities in the scene, manipulations based on the visible part of the object may fail; to compensate, object recognition is often used to identify the location of the object and compute the grasp from a prior model. However, new instances of a known class of objects may vary from the prior model, and known objects may appear in novel configurations if they are not perfectly rigid. As a result, manipulation can pose a substantial modeling challenge when objects are not fully in view.

Consider the camera image[1] of four toys in a box in figure 6-1(a). Having a prior model of the objects is extremely useful in that visible segments (such as the three visible parts of the stuffed bear) can be aggregated into a single object, and a grasp can be planned appropriately as in figure 6-1(b). However, having a prior model of the geometry of every object in the world is not only infeasible but unnecessary. Although an object such as the stuffed bear may change shape as it is handled and

[1]Note that for the purposes of reproduction, the images have been cropped and modified from the original in brightness and contrast. They are otherwise unchanged.

(a) Original Image



(b) Recovered Geometries

Figure 6-1: (a) A collection of toys in a box. The toys partially occlude each other, making object identification and grasp planning difficult. (b) By using learned models of the bear, we can identify the bear from the three visible segments and predict its complete geometry (shown by the red line; the dashed lines are the predicted outline of the hidden shape). This prediction of the complete shape can then be used in planning a grasp of the bear (planned grasp points shown by the blue circles).

Algorithm 1 The Manipulation Process.

Require: An image of a scene, and learned models of objects

1: Segment the image into object components
2: Extract contours of components
3: Determine maximum-likelihood correspondence between observed contours and known models
4: Infer complete geometry of each object from matched contours
5: Return planned grasp strategy based on inferred geometries

placed in different configurations, the general shape in terms of a head, limbs, etc. are roughly constant. Regardless of configuration, a single robust model which accounts for deformations in shape should be sufficient for recognition and grasp planning for most object types.

6.1 The Manipulation Process

Our goal is to manipulate an object in a cluttered scene–for example to grasp the bear in figure 6-1(a). Our proposed manipulation process is given in algorithm 1. The input to the algorithm is a single image which is first segmented into perceptually similar regions. The boundaries or contours of the image segments are extracted, and we analyze them with the methods of chapters 2-4 of this thesis.

6.1.1 Grasp Planning

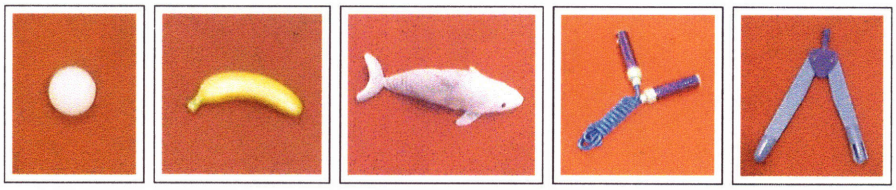
Given an estimate of the geometry of a detected object, we can plan a grasp for a manipulator. We have developed a grasp planning system for our mobile manipulator (shown in figure 6-2), a two-link arm on a mobile base with an in-house-designed gripper with two opposable fingers. Each finger is a structure capable of edge and surface contact with the object to be grasped.

Figure 6-2: Our mobile manipulator with a two link arm and gripper.

The input to the grasp planning system is the object geometry with the partial contours completed as described in Section 4.1. The output of the system is two regions, one for each finger of the gripper, that can provide an equilibrium grasp for the object following the algorithms for stable grasping described in [40]. Intuitively, the fingers are placed on opposing edges so that the forces exerted by the fingers can cancel each other out.

The grasp planner is implemented as search for a pair of grasping edges that yield maximal regions for the two grasping fingers using the geometric conditions derived by Nguyen [40]. Two edges can be paired if their friction cones are overlapping. Given two edges that can be paired we identify maximal regions for placing the fingers so that we can tolerate maximal uncertainty in the finger placement using Nguyen's criterion [40].

(a) ring (b) bat (c) rat (d) bear (e) fish (f) wand



(g) ball (h) banana (i) dolphin (j) jumprope (k) compass

Figure 6-3: Examples of one shape from each of the 11 classes of toys. Several of the classes contain objects which deform, either because of articulations (compass, jumprope) or because of the object's soft material (rat, bear, fish, dolphin). Two other classes (ring, bat) contain multiple instances of rigid objects.

6.2 Results

We built a shape dataset containing 11 shape classes. One example object from each class is seen in figure 6-3. We collected 10 images of each object type, segmented the object contours from the background, and used the correspondence and shape distribution learning algorithms of chapters 2 and 3 to build probabilistic shape models for each class, using contours of 100 points each. Our training data set is relatively sparse and therefore prone to over-fitting. In order to avoid this problem, we reduced the dimensionality of the covariance using Principal Components Analysis (PCA). Reducing the covariance to three principal components led to 100% prediction accuracy of the training set, and 98% cross-validated ($k = 5$) prediction accuracy. In figure 6-4, we show the effects of the top 3 principle components on the mean shape for each class.

In figures 6-5 and 6-6 we show the results of two example manipulation experiments, used to set algorithm parameters. In each case we seek to retrieve a single type of object from a box of toys, and we must locate and grasp this object while using the minimum number of object grasps possible. In both cases, the object we

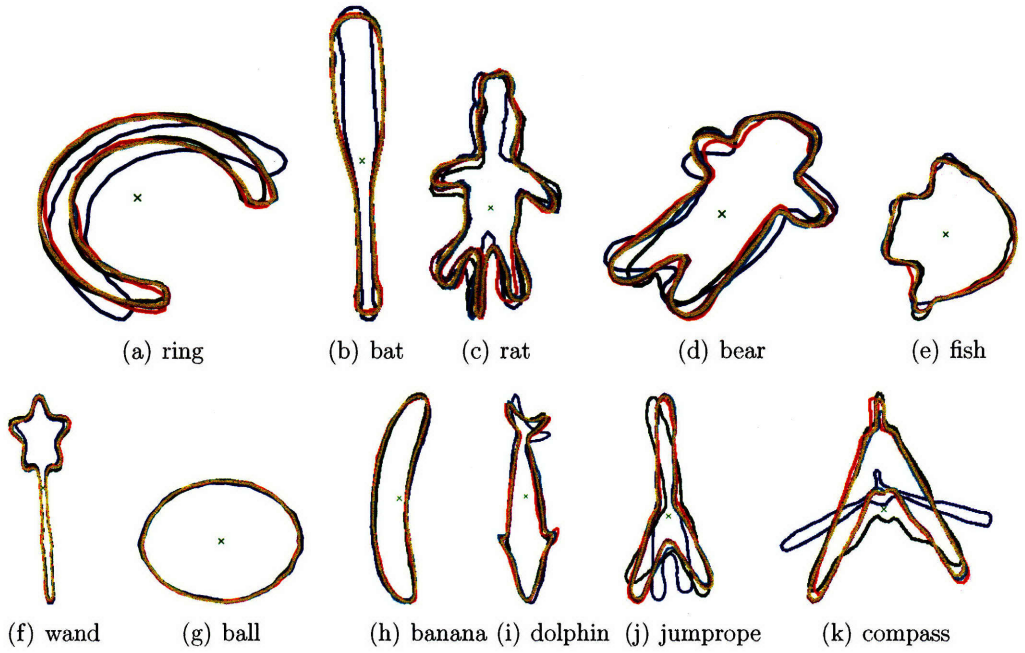
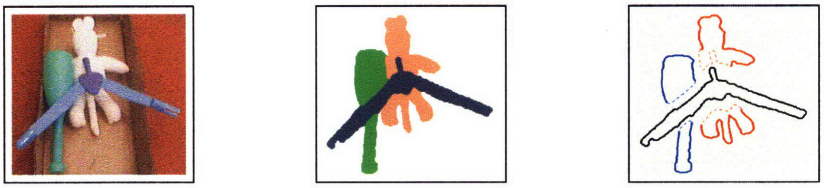


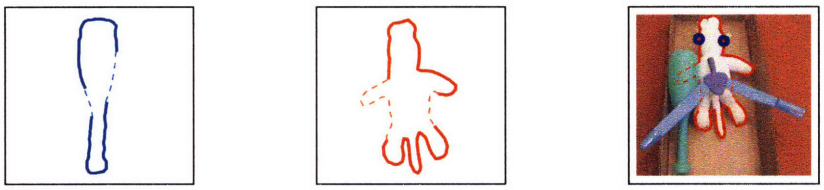
Figure 6-4: Shape models learned for each of the object classes. The red contours are the mean shapes, and the others are sampled along each of the top three eigenvectors.

wish to retrieve is occluded by other objects in the scene, and so a naive grasping strategy would first remove the objects on top of the desired object until the full object geometry is observed, and only then would it attempt to retrieve the object. Using the inferred geometry of the occluded object boundaries to classify and plan a grasp for the desired object, we find in both cases that we are able to grasp the object immediately, reducing the number of grasps required from 3 to 1. In addition, we were able to successfully complete and classify the other objects in each scene, even when a substantial portion of their boundaries was occluded. The classification of this training set of 7 object contours (from 6 objects classes) was 100% (note the correct completions in figures 6-5 and 6-6 of the occluded objects).

For a more thorough evaluation, we repeated the same type of experiment on 20 different piles of toys. In each test, we again sought to retrieve a single type of object from the box of toys, and in some cases, the manipulation algorithm required several grasps in order to successfully retrieve an object, due to either not being able to find the object right away, or because the occluding objects were blocking access to a



(a) Original image (b) Segmentation (c) Contours



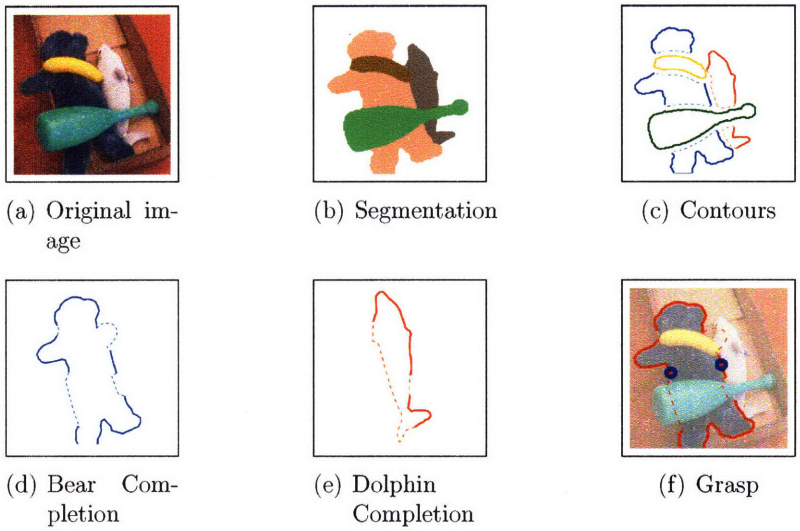
(d) Bat Completion (e) Rat Completion (f) Grasp

Figure 6-5: An example of a very simple planning problem involving three objects. The chalk compass is fully observed, but the stuffed rat and green bat are partially occluded by the compass. After segmentation (b), the image decomposes into five separate segments shown in (c). The learned models of the bat and the rat can be completed (d) and (e), and the complete contour of the stuffed rat is correctly positioned in the image (f). The two blue circles correspond to the planned grasp that results from the computed geometry.

stable grasp of the desired object. Figures 6-7 and 6-7 show 2 of the 20 trials in our experiment. Both trials are examples in which it took the robot more than one grasp to retrieve the desired object.

In figure 6-7, the object to be retrieved is the purple fish, which is initially occluded by the green bat. After segmentation and contour completions, the algorithm is able to recognize the fish (Figure 6-7(d)), but it realizes that the bat is in the way, and so it plans a grasp of the bat (Figure 6-7(e)) and removes it. This time, the fish is again completed (Figure 6-7(i)) and successfully classified as a fish, and a grasp is planned and executed (Figure 6-7(j)). All contours all correctly classified throughout the experiment.

In figure 6-8, the object to be retrieved is the yellow ring, which is initially occluded by both the blue bear and the green bat. After segmentation and contour completions, the algorithm is able to recognize the ring (Figure 6-8(d)), but it realizes that it must remove the bat before it can access the ring, so it plans a grasp of the bat (Figure 6-



(a) Original image (b) Segmentation (c) Contours

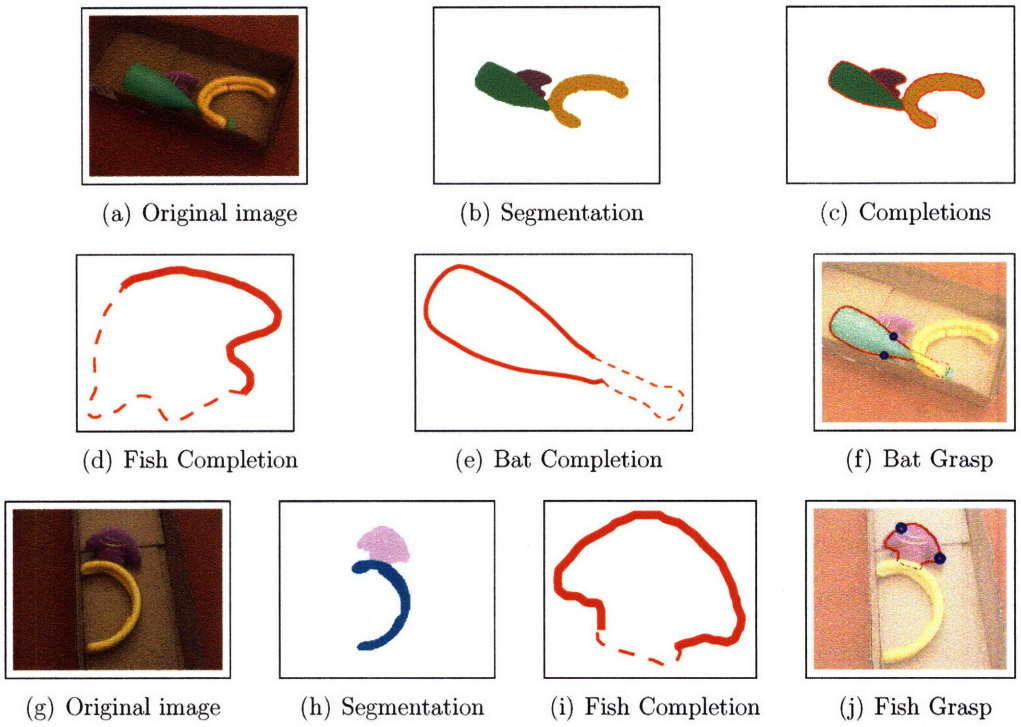
(d) Bear Completion (e) Dolphin Completion (f) Grasp

Figure 6-6: A more complex example involving four objects. The blue bat and the yellow banana are fully observed, but the stuffed bear and dolphin are significantly occluded. After segmentation (b), the image decomposes into five separate segments shown in (c). The learned models of the bear and the dolphin can be completed (d) and (e), and the complete contour of the stuffed bear is is correctly positioned in the image (f). The two blue circles correspond to the planned grasp given the geometry.

8(g)) and removes it. This time, the ring is again correctly completed and identified (Figure 6-8(k)), and a grasp is executed (Figure 6-8(l)), but fails due to the weight of the bear lying on top of the ring. After another round of image analysis, the ring is successfully retrieved (Figure 6-8(p)). Note that the rat was misclassified as a bear throughout this experiment; however this classification error had no effect on the retrieval of the ring.

In total, 52 partial and 49 complete contours were classified, 33/35 grasps were successfully executed (with 3 failures due to a hardware malfunction which were discounted). In table 6.1, we show classification rates for each class of object present in the images. Partially-observed shapes were correctly classified 71.15% of the time, while fully-observed shapes were correctly classified 93.88% of the time. Several of the errors were simply a result of ambiguity–when we examine the $> 5\%$ detection rates (i.e. the percentage of objects for which the algorithm gave at least 5% likelihood to the correct class), we see an improvement to 80.77% for partial shapes, and 97.96%

for full shapes. While a few of the detection errors were from poor or noisy image segmentations, most were from failed correspondences from the observed contour to the correct shape model. The most common reason for these failed correspondences was a lack of local features for the COPAP algorithm to latch onto with the PLSD point assignment cost. These failures would seem to argue for a combination of local and global match likelihoods in the correspondence algorithm, which is a direction we hope to explore in future work.



(a) Original image (b) Segmentation (c) Completions

(d) Fish Completion (e) Bat Completion (f) Bat Grasp

(g) Original image (h) Segmentation (i) Fish Completion (j) Fish Grasp

Figure 6-7: Example of an experiment where more than one grasp is required to retrieve the desired object (the purple fish). The robot is able to immediately recognize the fish, but it is required to first remove the bat before a good grasp of the fish is available.

6.3 Related Work

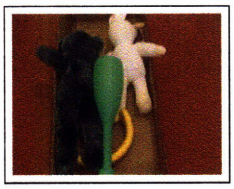
In addition to the related work on shape analysis discussed throughout the previous chapters of this thesis, we also build on classical and recent results on motion plan-

Object	Partial	Complete
ring	3/8	15/15
bat	7/10	8/10
rat	9/13	4/4
bear	7/7	7/7
fish	9/9	6/6
banana	-	1/2
dolphin	1/2	-
compass	1/3	5/5
totals	37/52	46/49
	71.15%	93.88%
detect > 5%	42/52	48/49
	80.77%	97.96%

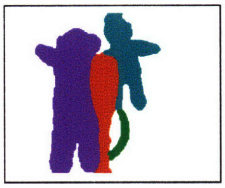
Table 6.1: Classification rates on test set.

ning and grasping, manipulation, uncertainty for modeling in robot manipulation, POMDPs applied to mobile robots, kinematics, and control. The initial formulation of the problem of planning robot motions under uncertainty was the preimage backchaining paper [35]. It was followed up with further analysis and implementation [11, 14], analysis of the mechanics and geometry of grasping [37], and grasping algorithm that guarantees geometrically closure properties [40]. Lavalle and Hutchinson [32] formulated both probabilistic and nondeterministic versions of the planning problem through information space. Our manipulation planner currently does not take advantage of the probabilistic representation of the object, but we plan to extend our work to this domain.

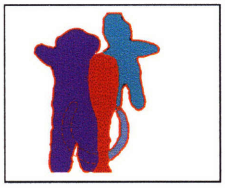
More recently, Grupen and Coelho [22] have constructed a system that learns optimal control policies in an information space that is derived from the changes in the observable modes of interaction between the robot and the object it is manipulating. Ng et al. [46] have used statistical inference techniques to learn manipulation strategies directly from monocular images; while these techniques show promise, the focus has been generalizing as much as possible from as simple a data source as possible. It is likely that the most robust manipulation strategies will result from including geometric information such as used by Pollard and Zordan [43].



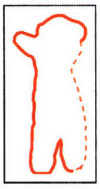
(a) Original image



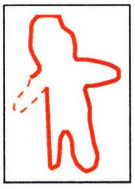
(b) Segmentation



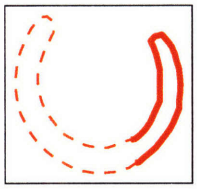
(c) Completions



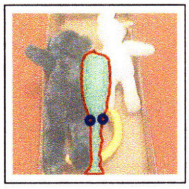
(d) Bear Completion



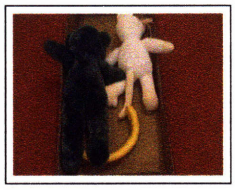
(e) Rat Completion



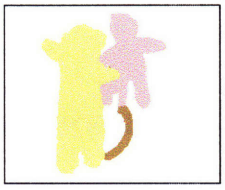
(f) Ring Completion



(g) Bat Grasp

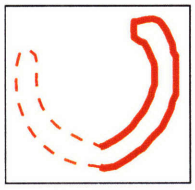


(h) Original image

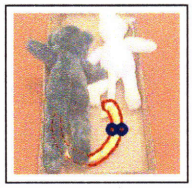


(i) Segmentation

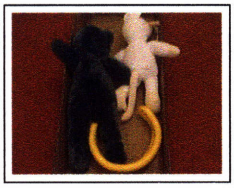
(j) Rat Completion



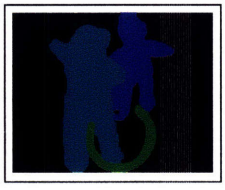
(k) Ring Completion



(l) Ring Grasp (Failed)



(m) Original image



(n) Segmentation

(o) Rat Completion

(p) Ring Grasp (Succeeded)

Figure 6-8: In this experiment, three grasps were necessary to retrieve the desired object (the yellow ring). The robot is able to immediately recognize the yellow ring, but it is required to first remove the bat and the bear before a successful grasp of the ring is achieved.

Chapter 7

Conclusion

7.1 Accomplishments

In this thesis, we have used the Procrustean shape metric to develop dense probabilistic models of object geometry. In chapter 2, we started out with a simple shape similarity metric–the Procrustean shape distance, and we saw how it achieved invariance to position, scale, and orientation. We then used this shape metric to build a probabilistic model of object geometry, based on tangent space principle components analysis. We experimented with our first correspondence model as part of the Mirrored Shifted Procrustean (MSP) metric, which scanned for the best starting point to align the points of two contours according to the Procrustean metric, and we showed that many shapes were not aligned properly with this simple correspondence method, resulting in poorly modeled shape classes and a relatively low classification rate on the MPEG-7 shape dataset.

In chapter 3, we developed a novel graphical model for shape correspondences, based on the Cyclic Order-Preserving Assignment Problem (COPAP) framework. We presented a new likelihood model for local shape similarity, based on the Procrustean Local Shape Distance (PLSD), and we described in detail how COPAP-PLSD could be solved with dynamic programming by breaking dependencies in the graphical model. We then gave detailed algorithms for combining the COPAP-PLSD correspondences with the tangent space PCA shape models of chapter 2, and we showed improved

shape classification over the basic scanning (MSP) correspondences on the MPEG-7 dataset.

In chapter 4, we derived a solution to the shape completion problem, and we used our shape models to infer the hidden parts of partially-occluded, deformable objects.

In chapter 5, we demonstrated how a discriminative model of shape parts, also based on the Procrustean framework, could be trained to detect both full and partially-occluded shapes, and we showed improved performance on an object retrieval task (the "bullseye" test) on the MPEG-7 shape dataset over all previously published methods.

Finally, in chapter 6, we applied our shape inference algorithms to the task of robotic grasping, where we demonstrated the applicability of our algorithms by showing that learning probabilistic models of deformable objects allows us to efficiently recognize and retrieve complex objects from a pile in the presence of sensor noise and occlusions.

7.2 Lessons Learned and Future Work

7.2.1 Algorithmic

Unfortunately, the improved correspondences of COPAP-PLSD in chapter 3 came at a cost: speed. A correspondence between two shape contours of 100 points each took around 1 second on average (on an Intel Pentium 4, 3.2 GHz computer), which is several orders of magnitude slower than would be necessary for real-time shape analysis in the complex scenes of chapter 6. Of course, by moving our code base from Matlab to C and optimizing our code, we should be able to achieve at least an order of magnitude in speed-ups. However, we also believe that there is room to streamline the algorithm itself. While the MSP correspondence algorithm is linear in the number of points on each contour, $O(n+m)$, COPAP is super-quadratic: $O(nm \log n)$. Furthermore, although computing the PLSD cost matrix has a theoretical time bound of $O(nm)$ since both the number of neighborhoods and the number of points in each

neighborhood are fixed, the constants are often quite high, meaning that it usually takes more time to fill up the cost matrix than it does to solve the resulting dynamic program! This leads naturally to the idea of using a different shortest paths algorithm–rather than dynamic programming–to find solutions in the COPAP search graph. Unfortunately, after experimenting with multiple heuristics in the A* search algorithm which attempt to exploit the regular structure of the PLSD cost matrix, we have found that there is simply no way to guarantee an optimal solution without exploring most of the nodes in the COPAP search graph, on average. This is because COPAP is solved by trying each and every wrapping point and solving the resulting linear correspondence problems. However, we have observed in practice that most wrapping points lead to extremely sub-optimal solutions–if the tail of one dog is matched to the other dog's head initially, how can one hope to match the rest of the animals' body parts correctly? It is this line of reasoning that may ultimately lead to better heuristics for the COPAP algorithm which prune entire wrapping points from the COPAP search in order to reduce time spent on unlikely correspondences.

Another line of reasoning which seems promising is to compute correspondences hierarchically, starting from a small set of high-importance "anchor" correspondences and then filling in between them with LOPAP searches. We have already noted–by looking at the structure of the PLSD cost matrix–that unique, high-curvature local shape features tend to dominate the point assignment cost terms in the correspondence likelihood. Thus, it makes sense to try to assign these high-cost points first, and then to fill in correspondences on the flat regions between them. The difficulty with this approach is that a single mistake in the "anchor" correspondences can ruin the chance for a good overall correspondence; thus a search over anchor correspondences might be necessary in order to provide robustness to these types of errors.

Luckily, if the correspondence algorithm is made to be more efficient, there is hope that all of the shape modeling and inference procedures can be made quite speedy. Since we use principle components analysis to reduce the dimensionality of our shape models, computing the likelihood of a shape under a tangent space PCA model is linear in the number of points on the shape. And if it turns out to be impossible to

speed up correspondence-finding; well then, there's always Moore law!

In order to analyze the scenes in chapter 6 for our grasping experiments, we made a number of key assumptions to focus our evaluation on the performance of our shape analysis algorithms. In particular, we assumed that a prior segmentation was available, and that object segments were already grouped together, so that no search over possible segments groupings was necessary. Although shape models have been shown to be useful in performing image segmentation, we feel that segmentation is out of the scope of this work. However, a search over segment groupings likely to be unavoidable in any real world scenarios when objects vary in appearance (unlike the objects chosen for our experiments, which were all composed of solid colors). In cases where this search is necessary, the shape part detection scheme of chapter 5 can be used to prune away unlikely classifications. However, it will probably also be necessary to add additional pruning steps in order to make the algorithm more feasible, for example by including negative information, since knowledge of where an object *isn't* located is often just as important as where an object is located.

7.2.2 Experimental

A quick look at the top ten most misclassified shape classes in the MPEG-7 dataset according to the COPAP-PCA model reveals a few key weaknesses of our correspondence algorithm. Four of the top ten most misclassified shape classes are "device3", "device4", "device6", and "device9", which all contain widely varying deep cuts in their shape contours. Skipping over these cuts will incur a large skipping penalty; furthermore, some portions of the cuts are actually flat, locally, and do match well with other flat portions, which may lie on the exterior of the shape. Five other classes in the top ten are examples of classes with varying protrusions: "ray", "bird", "butterfly", "camel", and "horse". Some bird contours have two feet, while others have only one; some camel contours have two humps and four legs, others have one hump and two legs, etc. In fact, the two problems–cuts and protrusions–can be seen as highly-related dual problems. While the skipping penalty in COPAP encourages the smoothing out of assignments, the correct correspondence of two shapes with cuts or

protrusions may be to skip over large portions of the contours. This is because such portions of contours contain a disproportionate number of points in comparison with the ratio of area of the cut or protrusion to the area of the entire shape.

One potential solution to the cuts and protrusions problem which we have explored is to model the object at multiple scales in image scale space. If an image is blurred (or dilated), cuts will be filled in, and protrusions will lose detail, seeming more uniform. For example, using such scale space techniques we have managed to raise the MPEG-7 classification rate of COPAP-PCA from 79.29% to 86.79%, an improvement of 7.5%. However, in order to really solve the cuts and protrusions problem, it will most likely be necessary to make a more fundamental change to our shape models, either by using a skeletonized, parts-based model (where each part is modeled by one of our probabilistic contour models) to learn a hierarchical model of shape, or by incorporating additional information into the correspondence model, such as orientation, curvature, or spacing.

We would also like to investigate the use of other probability densities for modeling shapes in shape space. While tangent space PCA has many benefits–including simplicity–many of the shape classes we have encountered in our work would be better modeled by a multi-modal distribution, such as a Gaussian mixture model or a complex Bingham. Additionally, we expect that the sequential model learning algorithm of chapter 3 could be vastly improved by using a hierarchical model merging algorithm, which would merge similar shapes until merging no longer improves the model; such a technique would lend itself quite nicely to generating mixture models for shape densities.

As we mentioned above, shape completion could also be improved by incorporating negative information. In addition, the current method of partial shape correspondences for shape completion can be quite brittle; to become robust to more complex shapes than the ones we encountered in the grasping experiments in chapter 6, one would need to find a way to (approximately) marginalize over correspondences by keeping around multiple hypotheses of correspondences and trying each one until a completion is found which satisfies the scene geometry. We also believe a brute-force

sampling strategy or search for scene geometry may be beneficial as a back-up to full probabilistic inference when there is simply not enough geometry present to make good correspondences to the prior models.

In the process of testing our grasping algorithms, we discovered that our hardware–which consisted of a basic two-link arm and a servo-powered gripper which could open and close, and which relied on rubber bands to maintain contact with its target object–was simply not up to the task of manipulating many complex, small, or heavy objects. Thus, our primary concern in choosing a dataset of objects (other than that it included different types of shape variation) was whether or not the robot was able to pick them up with its simple gripper configuration. Even after carefully choosing our dataset, we still encountered numerous failed grasps, due to the hardware of the system. Therefore, in order to focus on the performance of the geometric algorithms in this thesis, we have left out these hardware failures from our results. However, in order to make a system which can take full advantage of the sophisticated inference procedures which we have developed in this work, it is imperative that the hardware be made more robust.

Finally, it should be pointed out that we did not actually use the probabilistic models of object geometry to compute grasps; we simply assumed that our estimate of the maximum likelihood geometry of the scene was correct, and planned grasps accordingly. Ultimately, to be really useful for grasp planning, we will need to find a way to use the probabilistic, local geometry of objects to compute grasps which are robust to object deformations and scene uncertainty.

Bibliography

[1] Naif Alajlan, Ibrahim El Rube, Mohamed S. Kamel, and George Freeman. Shape retrieval using triangle-area representation and dynamic space warping. *Pattern Recogn.*, 40(7):1911–1920, 2007.

[2] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002.

[3] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[4] A. Blake and M. Isard. *Active Contours.* Springer-Verlag, 1998.

[5] H. Blum. Biological shape and visual science. *Theoretical Biology*, 38:205–287, 1973.

[6] F.L. Bookstein. A statistical method for biological shape comparisons. *Theoretical Biology*, 107:475–520, 1984.

[7] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(6):567–585, 1989.

[8] 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.

[9] Daniel Cremers, Timo Kohlberger, and Christoph Schnorr. Shape statistics in kernel space for variational image segmentation. *Pattern Recognition*, 36(9):1929–1943, 2003.

[10] F. Dellaert, S.M. Seitz, C. Thorpe, and S. Thrun. EM, MCMC, and chain flipping for structure from motion with unknown correspondence. *Machine Learning*, 50(1-2), 2003.

[11] Bruce Donald. A geometric approach to error detection and recovery for robot motion planning with uncertainty. *Artificial Intelligence*, 37:223–271, 1988.

[12] I. Dryden and K. Mardia. *Statistical Shape Analysis.* John Wiley and Sons, 1998.

[13] G. Elidan, G. Heitz, and D. Koller. Learning object shape: From drawings to images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[14] Michael Erdmann. Using backprojection for fine motion planning with uncertainty. *IJRR*, 5(1):240–271, 1994.

[15] P. Felzenszwalb. Representation and detection of deformable shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2), 2005.

[16] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting, 1998.

[17] Davi Geiger, Tyng-Luh Liu, and Robert V. Kohn. Representation and self-similarity of shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(1):86–99, 2003.

[18] Anarta Ghosh and Nicolai Petkov. Robustness of shape descriptors to incomplete contour representations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(11):1793–1804, 2005.

[19] Jared Glover, Daniela Rus, Nicholas Roy, and Geoff Gordon. Robust models of object geometry. In *Proceedings of the IROS Workshop on From Sensors to Human Spatial Concepts*, Beijing, China, 2006.

[20] Paul G. Gottschalk, Jerry L. Turney, and Trevor N. Mudge. Efficient recognition of partially visible objects using a logarithmic complexity matching technique. *International Journal of Robotics Research*, 8(6):110–131, December 1989.

[21] W. E. L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(4):469–482, 1987.

[22] Roderic A. Grupen and Jefferson A. Coelho. Acquiring state from control dynamics to learn grasping policies for robot hands. *Advanced Robotics*, 16(5):427–443, 2002.

[23] D.P. Huttenlocher, G.A. Klanderman, and W.A. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[24] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *European Conf. Computer Vision*, pages 343–356, 1996.

[25] V. Jain and H. Zhang. Robust 2d shape correspondence using geodesic shape context. *Pacific Graphics*, pages 121–124, May 2005.

[26] D.G. Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bull. London Math Soc.*, 16:81–121, 1984.

[27] D.G. Kendall, D. Barden, T.K. Carne, and H. Le. *Shape and Shape Theory.* John Wiley and Sons, 1999.

[28] M. W. Koch and R. L. Kashyap. Using polygons to recognize and locate partially occluded objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(4):483–494, 1987.

[29] R. Lakmper, L. J. Latecki, V. Megalooikonomou, Q. Wang, and X. Wang. Learning descriptive and distinctive parts of objects with a part-based shape similarity measure. In *IASTED Int. Conf. on Signal and Image Processing (SIP)*, 2004.

[30] L. J. Latecki and R. Lakämper. Contour-based shape similarity. In *Proc. of Int. Conf. on Visual Information Systems*, volume LNCS 1614, pages 617–624, June 1999.

[31] L. J. Latecki, R. Lakämper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 424–429, 2000.

[32] S. M. LaValle and S. A. Hutchinson. An objective-based stochastic framework for manipulation planning. In *Proc. IEEE/RSJ/GI Int'l Conf. on Intelligent Robots and Systems*, pages 1772–1779, September 1994.

[33] C. C. Lin and R. Chellappa. Classification of partial 2-d shapes using fourier descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):686–690, 1987.

[34] Haibin Ling and David W. Jacobs. Using the inner-distance for classification of articulated shapes. *IEEE Computer Vision and Pattern Recognition*, 2:719–726, June 2005.

[35] Tomás Lozano-Pérez, Matthew Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1), 1984.

[36] Maurice Maes. On a cyclic string-to-string correction problem. *Inf. Process. Lett.*, 35(2):73–78, 1990.

[37] Matthew T. Mason and J. Kenneth Salisbury Jr. *Robot Hands and the Mechanics of Manipulation.* MIT Press, Cambridge, Mass., 1985.

[38] T. McInerney and D. Terzopoulos. Deformable models in medical images analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[39] F. Mokhtarian and A. K. Mackworth. A theory of multiscale curvature-based shape representation for planar curves. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 14, 1992.

[40] Van-Duc Nguyen. Constructing stable grasps. *I. J. Robotic Res.*, 8(1):26–37, 1989.

[41] E. Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *IEEE Intern. Conf. on Face and Gesture Recognition*, pages 889–894, 2004.

[42] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *Proceedings of the European Conference on Computer Vision*, 2006.

[43] N. S. Pollard and Victor B. Zordan. Physically based grasping control from example. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, Los Angeles, CA, 2005.

[44] S. Ranade and A. Rosenfeld. Point pattern matching by relaxation. *Pattern Recognition*, 12:269–275, 1980.

[45] Tammy Riklin-Raviv, Nahum Kiryati, and Nir Sochen. Prior-based segmentation by projective registration and level sets. In *Proc. of the International Conference on Computer Vision ICCV*, volume 1, pages 204–211, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[46] Ashutosh Saxena, Justin Driemeyer, Justin Kearns, Chioma Osondu, and Andrew Y. Ng. Learning to grasp novel objects using vision. In *Proc. International Symposium on Experimental Robotics (ISER)*, 2006.

[47] M.G. Schimek. Penalized logistic regression in gene expression analysis. In *Proc. The Art of Semiparametrics Conf.*, 2003.

[48] C. Scott and R. Nowak. Robust contour matching via the order preserving assignment problem. *IEEE Transactions on Image Processing*, 15(7):1831–1838, July 2006.

[49] Thomas Sebastian, Philip Klein, and Benjamin Kimia. On aligning curves. *PAMI*, 25(1):116–125, January 2003.

[50] Thomas Sebastian, Philip Klein, and Benjamin Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5), 2004.

[51] Thomas B. Sebastian and Benjamin B. Kimia. Curves vs skeletons in object recognition. *Signal Processing*, 85(2):247–263, February 2005.

[52] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker. Shock graphs and shape matching. In *ICCV*, pages 222–229, 1998.

[53] C. G. Small. *The statistical theory of shape.* Springer, 1996.

[54] J. Subrahmonia, D. B. Cooper, and D. Keren. Practical reliable bayesian recognition of 2d and 3d objects using implicit polynomials and algebraic invariants. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(5), May 1996.

[55] Boaz J. Super. Retrieval from shape databases using chance probability functions and fixed correspondence. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(8):1117–1138, 2006.

[56] Mirela Tanase and Remco C. Veltkamp. Part-based shape retrieval. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 543–546, New York, NY, USA, 2005. ACM Press.

[57] Jean-Philippe Tarel and David B. Cooper. The complex representation of algebraic curves and its simple exploitation for pose estimation and invariant recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(7):663–674, 2000.

[58] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* MIT Press, Cambridge, MA, 2005.

[59] R. C. Veltkamp and L. J. Latecki. Properties and performance of shape similarity measures. In *10th IFCS Conf. Data Science and Classification*, 2006.

[60] Remco C. Veltkamp. Shape matching: Similarity measures and algorithms. Technical Report UU-CS-2001-03, Utrecht University, 2001.

[61] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[62] B. Wu and R. Nevatia. Simultaneous object detection and segmentation by boosting local shape feature based classifier. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.