Abstract

Bone Graphs: Medial Abstraction for Shape Parsing and Object Recognition

Diego A. Macrini
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2010

The recognition of 3-D objects from their silhouettes demands a shape representation which is invariant to minor changes in viewpoint and articulation. This invariance can be achieved by parsing a silhouette into parts and relationships that are stable across similar object views. Medial descriptions, such as skeletons and shock graphs, attempt to decompose a shape into parts, but suffer from instabilities that lead to similar shapes being represented by dissimilar part sets. We propose a novel shape parsing approach based on identifying and regularizing the ligature structure of a given medial axis. The result of this process is a bone graph, a new medial shape abstraction that captures a more intuitive notion of an objects parts than a skeleton or a shock graph, and offers improved stability and within-class deformation invariance over the shock graph.

The bone graph, unlike the shock graph, has attributed edges that specify how and where two medial parts meet. We propose a novel shape matching framework that exploits this relational information by formulating the problem as an inexact directed acyclic graph matching, and extending a leading bipartite graph-based matching framework introduced for matching shock graphs. In addition to accommodating the relational information, our new framework is better able to enforce hierarchical and sibling constraints between nodes, resulting in a more general and more powerful matching framework. We evaluate our matching framework with respect to a competing shock graph matching framework, and show that for the task of view-based object categorization, our matching
framework applied to bone graphs outperforms the competing framework. Moreover, our matching framework applied to shock graphs also outperforms the competing shock graph matching algorithm, demonstrating the generality and improved performance of our matching algorithm.
Acknowledgements

Firstly, I would like to thank my advisor, Sven Dickinson, for giving me the academic and human support that I needed throughout my studies. He proved countless times to care deeply about me both as a student and as a person. His passion for solving hard vision problems and his tireless approach to work and life set a high bar for me. I would also like to thank my co-advisor, David Fleet, for his kind guidance and his good cheer. His comments and thoughts have improved this thesis in numerous ways.

I am deeply grateful to Kaleem Siddiqi, who was an integral part of this thesis and whose work on shape representations was an important source of inspiration. Moreover, I would like to thank him for all the trips between Montreal and Toronto that he took to attend the meetings of my Ph.D. committee. I am also grateful to Allan Jepson, who was an exemplary chair of my committee meetings, and whose thorough reading of my work and tough questioning set a high standard. I want to express my gratitude toward Ali Shokoufandeh for his insightful comments, and for being always ready to give me a helpful hand on short notice. Lastly, I would like to thank my external committee member Doug DeCarlo, who impressed me with his thoughtfulness and kindness.

I must also thank my friends and colleagues in the computer vision group: Marcus Brubaker, Gustavo Carneiro, Chakra Chennubhotla, Matthijs van Eede, Francisco Estrada, Fernando Flores-Mangas, Sam Hasinoff, Mike Jamieson, Alex Levinshtein, and Pablo Sala. I cannot thank them enough for all the times in which they helped me improve my numerous presentations over the years. Together they created a stimulating group with whom I always felt comfortable sharing all aspects of our challenging research experiences.

If felt at home in the computer science department as much as I did, it is only because of my friends Patricio Simari, Pablo Sala, Blas Melissari, Flavio Rizzolo, Ariel Fuxman, and Sebastian Sardina. Our passionate lunch-time discussions covered all imaginable topics, and the range of different views in the group ensured that we all learned something about each topic. By the same token, my graduate student life would not have been complete without sharing it with my two good friends Josh Buresh-Oppenheim and Jennifer Listgarten.

I would like to conclude by expressing my deep love toward my family, and by thanking them for their unconditional support. My wife, Luisa, gave me the understanding that I needed through the course of my studies, while our son took care of lifting my mood each
day. My mother and sister were always there to encourage me and to provide a positive view to each challenge, whereas my father, Alejandra and my brother put the essential cheerful spin on every experience.
# Contents

1 Introduction 1
   1.1 Coping with Skeletal Instabilities 3
   1.2 Matching Bone Graphs 6

2 Related Work 9
   2.1 Skeleton-Based Shape Parsing 9
      2.1.1 Part Attachment Instability of the Medial Axis 10
   2.2 Parts-Based Shape Representations 13
   2.3 Inexact Graph Matching 16

3 Shape Parsing Using Skeletal Information 21
   3.1 Introduction 21
   3.2 Local Geometry of the Medial Axis 23
   3.3 Intuition and Goals 26
   3.4 Detecting Ligature 29
   3.5 Detecting Protrusions 33
      3.5.1 Labeling Junctions as Y or P 35
      3.5.2 Nested Protrusions 39
      3.5.3 Merging Host Branches 43
   3.6 Recovering Shape Parts 45
   3.7 The Shape Parsing Algorithm 48
   3.8 Conclusions 49

4 Bone Graphs: Medial Abstraction for Object Recognition 51
   4.1 Introduction 51
   4.2 Bone Graph Construction 53
Chapter 1

Introduction

View-based object recognition demands a parts-based shape representation that gracefully accommodates both articulation and minor changes in viewpoint. The process of recovering such parts from an input shape can be compared to the parsing of natural language sentences. In that context, to parse means “to break (a sentence) down into its component parts of speech with an explanation of the form, function, and syntactical relationship of each part” [27]. By analogy, the problem of parsing a shape is defined in this thesis as the decomposition of an object’s silhouette into meaningful parts with an explanation of the role of each part and its relationships with other parts. For the task at hand, we consider a shape part to be meaningful if it is largely invariant to the natural deformations undergone by silhouettes, such as those shown in Figure 1.1.

Skeletal descriptions decompose a shape into a set of symmetry-based parts, and have a long history in the shape recognition community. In 3-D, Binford’s generalized cylinders [10] decomposed an object into a set of elongated parts defined by sweeping a 2-D cross section through a 3-D space curve. The concept of an axial description of shape was proposed even earlier in 2-D through Blum’s medial axis transform, or skeleton [11]. Skeletonization algorithms map a closed 2-D shape into a set of medial branches that terminate at endpoints or branch junctions. Unfortunately, the number of branches used to describe a shape has been shown to be overly sensitive to the addition or deletion of parts [2], which has limited the use of skeletons in the recognition community. In the first part of this thesis, we argue that this instability can be overcome, and moreover, that the process of producing stable skeletons is an effective method for shape parsing. We introduce a new algorithm that takes an input skeleton and rectifies the oversegmentation and undersegmentation of shape parts produced by the branching topology of the
Chapter 1. Introduction

Figure 1.1: Shape Deformation due to Viewpoint Changes: the silhouette of a 3D object can change drastically when seen from different viewpoints.

skeleton. The result is a new and powerful graph-based shape abstraction, called a *bone graph*, that offers significant advantages over the ubiquitous shock graph.

Given this new shape representation, our next challenge is to recognize a query bone graph from a database of model bone graphs. We will not address the bone graph indexing problem in this thesis, i.e., using the query bone graph to efficiently prune the model database down to a small number of candidates. Rather, we focus on the matching problem, which computes a node-to-node correspondence between two bone graphs, along with a similarity score that can be used to rank order the models. This is an inexact graph matching problem, and we build on our previous work [83] on matching hierarchical structures in the presence of noise, occlusion, and clutter. In order to take full advantage of the new features and properties encoded in the bone graph, we extend our earlier work to consider graphs in which both nodes and edges are attributed. Moreover, we propose a novel graph matching framework that allows for the enforcement of domain-dependent constraints between the node correspondences. Our framework provides a contribution to the problem of matching *generic* directed acyclic graph-based representations, and therefore can be applied to other domains in pattern recognition.

We conclude by instantiating our graph matching framework for the problem of comparing bone graphs. This requires the definition of similarity functions for nodes and edges, and the specification of the domain assumptions about the relative importance of the structural differences between two graphs. Our proposed node similarity function is a robust measure that can gracefully account for the deformations produced by perspective transformations, part articulation, and within-class deformation. Our proposed edge similarity function is able to measure the position and orientation of each part attachment in terms of its local context. This allows our matching framework to be globally
Chapter 1. Introduction

invariant to 2-D rotation, reflection, and scaling, but sensitive to local perturbations in the positions and orientations of parts.

The contributions of the thesis therefore begin with a new shape parsing approach and a novel medial shape abstraction, and conclude with a powerful graph matching algorithm and its application to the bone graph domain. The result is a recognition pipeline based on a novel 2-D medial axis-based shape representation. We evaluate each component of this pipeline extensively, and compare it to a competing shock graph recognition framework. In the following sections, we expand on these contributions in more detail.

1.1 Coping with Skeletal Instabilities

An important criterion for any effective shape representation is locality of representation, i.e., the impact of a local boundary perturbation on the representation should also be local [59]. Unfortunately, shape skeletons fail to meet this criterion, because skeletal branches do not always map, in a one-to-one fashion, to meaningful shape parts. As an example, the shorter rear leg of the dog in Fig. 1.2(a) results in an incident branch that oversegments the representation of the dog’s body into two skeletal branches. An enlarged picture of the junction also reveals a local perturbation of the body branch, which arises due to the attachment. A similar situation arises near the two front legs. The representation of each of the four legs, as well as the tail, is also undersegmented in the sense that the associated skeletal branches extend well past the locations of the part attachments with the body. The net result is a skeleton topology that does a poor job of reflecting the salient part structure of the shape.

One alternative to mitigating the sensitivity of the topological structure of the skeleton is to pass this instability to the recognition (indexing followed by matching) algorithm. The fact that two very similar shapes may yield topologically dissimilar skeletons means that the matching algorithm must be able to establish correspondences at higher levels of abstraction, i.e., levels above the structural instability. A number of graph matching frameworks have evolved to address this challenging problem [89, 77]. In fact, in [77] the transitions of the medial axis [35] are themselves incorporated in edit-distance operations which allow topologically distinct skeletal structures due to similar shapes to be matched.

Unfortunately, passing the representational instability to the recognition stage makes the matching problem much harder, as it demands the solution of a challenging many-to-
Figure 1.2: Intuitive Part Decomposition: (a) skeletal instability arises from part oversegmentation and undersegmentation. For example, the medial axis of the dog’s body is given by two skeletal branches (instead of one) due to the junction point that represents the connection between these branches and the skeletal segment extending from the shorter rear leg. A similar situation occurs near the front legs. The vicinity of the part oversegmentation is enlarged in each case, showing the resulting perturbation of the skeleton. Those skeletal segments shown in green are ligature regions, and they contribute little to the shape of the object. A purely local analysis of ligature is problematic in the presence of such oversegmentation, as illustrated by the non-intuitive labeling of the body part in the vicinity of the oversegmentation as ligature. (b) Our algorithm for detecting and removing ligature-induced skeletal instability uses a novel local ligature analysis to first identify and rectify the medial representations of part protrusions. (c) A second ligature analysis then yields a set of salient parts, called bones (shown in black). The bones capture the coarse part structure of the object, as indicated by the colored parts reconstructed from the bones. (d) The bones give rise to a bone graph, an intuitive and stable representation whose nodes represent the salient parts and whose edges, derived from the final ligature analysis, capture part attachment.

many assignment of parts [24, 23]. Instead, we argue for a more stable shape representation that eases the burden on the matcher, allowing it to focus on finding one-to-one part correspondences. To this end, we use ligature analysis [11] to devise a more stable skeletal representation, where skeletal segments map in a one-to-one fashion to meaningful shape parts. Intuitively, ligature regions (shown as green curves in Fig. 1.2) are segments of the skeleton that represent the symmetry axes of boundary points that form one or two concave corners. Moreover, the ligature points contribute little to the representation of the boundary. One could simply remove these portions of the skeleton, as suggested in [2]. However, not every ligature point is a good candidate for removal, as illustrated in Fig. 1.2(a), where much of the dog’s body skeleton (toward its back legs) is labeled as
ligature. This shows that the part oversegmentation problem cannot be separated from the ligature analysis.

Our approach to dealing with skeletal instabilities can be described as follows. In a first stage of ligature analysis (Fig. 1.2(a)), we identify the cases of part protrusions, and proceed to rectify the oversegmentation caused by the medial representation of the part hosting the protrusion through a branch merging process. This yields a set of branches that adhere to the geometric properties of the medial axis (Fig. 1.2(b)). Once the oversegmentation is corrected, a final ligature analysis yields a new set of ligature segments arising from skeletal branches that undersegment the medial axis of the shape parts (Fig. 1.2(c)). The correction of oversegmented medial axes is shown in Figure 1.2(b). As can be seen in the zoom-ins on the junctions, the two cases of medial axis oversegmentation have been identified and corrected. The correction of undersegmented medial axes is shown in Figure 1.2(c). The skeletal branches of the corrected skeleton are partitioned into ligature and non-ligature segments. Each non-ligature segment now maps to the medial axis of a distinct shape part.

The final set of non-ligature segments are used to determine the salient medial parts of the object. In fact, they yield a reconstructed shape that is very similar to that of the original object, as shown in Fig. 1.2 (c) by the union of the colored regions, each of which is the reconstruction of one non-ligature (black) branch. We assemble the restored non-ligature segments in a novel, parts-based shape representation called the bone graph, whose nodes represent stable, intuitive skeletal parts (bones), and whose attachment edges are derived from the ligature segments. The edges of the bone graph are directed according to a local estimate of relative part size, encoding hierarchical relations between nodes that can be exploited as constraints during matching. In the construction of the bone graph, we assume that the boundary of a shape is given by a simple closed curve. This leads to a directed acyclic graph (DAG) representation of the shape parts and their relationships. In Chapter 4, we evaluate the improved stability of our new representation by comparing it to the popular shock graph [89, 77] in a set of view-based 3-D object recognition and pose estimation trials. Experimental evidence demonstrates that the bone graph is less sensitive to viewpoint change-induced perturbations to a silhouette than the shock graph, leading to significantly improved recognition performance (Sections 4.3 and 6.4).
1.2 Matching Bone Graphs

The bone graph forms a hierarchy of parts with edges spanning adjacent parts. Like the shock graph [89, 77], a bone graph node encodes the geometry of a part. However, unlike the shock graph, a bone graph edge is attributed, specifying the attachment position (on each part) and the orientation of the attachment. The view-based recognition of 3-D objects using bone graphs requires a matching algorithm that can compare the attributes of nodes and edges as well as the structures of graphs in the presence of spurious and missing nodes. Figure 1.3 illustrates a typical example of similar views of two horses, which are represented by similar, but not isomorphic, bone graphs. One important difference between these two graphs is that in one of them, the front legs are connected directly to the body of the horse, while in the other, there is small shape part between the legs and the body. This small part is not noise, as it represents a true part between the legs and the torso. However, there is no natural corresponding part on the other shape, and the algorithm must be able to leave it unassigned, yet still establish correspondences between the front legs. Figure 1.4 shows the node correspondences found by our algorithm for this example.

The comparison of bone graphs defines an inexact graph matching between DAGs with high-dimensional node and edge attributes. There is a large body of work in the area of inexact graph matching, but most of it is focused on graphs in which either the nodes or the edges are attributed. Our experience with the inexact graph matching algorithm proposed by Shokoufandeh et al. [89, 84, 56, 98, 83] for DAGs with attributed nodes motivates our extension of this approach in order to incorporate edge information into the matching problem. We propose a generalization of this algorithm that expands the range of constraints that can be accounted for at matching time, leading to a general framework for representing domain knowledge about the relevance of structural differences between the graphs. Because the matching problem is intractable, the addition of domain knowledge is especially important to guide the search for approximate solutions that are relevant to the task at hand.

Our novel graph matching algorithm provides a contribution to the problem of matching generic graph-based representations, and therefore can be applied to a wide range of domains in computer vision and pattern recognition. We demonstrate the algorithm for the task of object categorization. This task is concerned with the recognition of 2-D views of novel 3-D exemplars of known object categories, and is a pivotal prob-
Figure 1.3: Example of missing non-terminal nodes in the bone graph. We show the bone graph representations of two different horse exemplars as seen from similar viewpoints (only the non-ligature points of the medial axes are drawn). In this example, the left horse has internal shape parts between its torso and its front/back legs and between its neck and ears that have no natural correspondence on the right horse. The representation of these parts in the graph leads to differences in the parent-child relations of some nodes. For example, the torso (node 1) is the grandparent of the front legs (nodes 3 and 6) on the top graph, whereas on the bottom graph, the torso is the parent of the front legs (nodes 2 and 9). In order to find the desired correspondences between the legs and ears, the graph similarity function that drives the matching process must favor solutions in which the missing parts are left unassigned.

In conclusion, we present three main contributions in this thesis: a shape parsing approach, a novel shape representation (the bone graph), and a generic graph matching algorithm. Our shape parsing approach recovers stable and intuitive shape parts and attachment relations from a given skeletal description while preserving its medial axis properties (Chapter 3). This approach is a contribution to the area of shape analysis, and can enable the construction of novel shape descriptors. In turn, our bone graph is a
Chapter 1. Introduction

Figure 1.4: Matching result for the horse example. The correspondences found by the algorithm are shown as arcs connecting shape parts, and as matching node colors in the graphs. In this case, the three internal shape parts in between the left horse’s torso and legs, and neck and ears are left unassigned (white nodes) at matching time.

A novel shape representation that assembles the shape parts and relations recovered by our parsing algorithm into a graph, and offers improved stability and within-class deformation invariance over previous skeletal-based representations (Chapter 4). Finally, our graph matching approach is a contribution to the generic problem of matching hierarchical structures based on domain knowledge of graph structure and graph attributes (Chapter 5). These three contributions are independent of one another, but also form a coherent framework for view-based object recognition, which we demonstrate in our experiments (Chapter 6).
Chapter 2

Related Work

2.1 Skeleton-Based Shape Parsing

The notion of parsing shapes has been used with different meanings in the last thirty years. For example, in early work, such as [81, 102], parsing is used in the context of generative grammars, and refers to a shape analysis technique in which a formal grammar is used to accept or reject alternate part decompositions of shape. This approach is known as syntactic shape recognition. More recently, the idea of parsing has been used in the psychophysics community to mean the perception of shape parts. As one example, Singh et al. [90] argue that humans parse shapes by locating and connecting negative curvature minima along a shape’s contour under a minimum distance rule. This idea of parsing is borrowed from its application to natural languages, and captures the notion that the perception of shape parts is influenced by many contextual factors, such as the relative size of parts [75, 91], their degree of protrusion [45, 46], and their convexity properties [54, 71]. Our use of the term in this thesis is closer to this latter idea. More specifically, we regard the term as an analogy that emphasizes the importance of recovering not only parts but also their roles and inter-relationships.

The medial axis transform (MAT) [11] can be seen as a computational method for parsing a silhouette into its symmetric parts (skeletal branches) and their adjacency relations (branch connectivity). The skeleton is also a transformation of the original shape that preserves all the information in it. These properties make the skeleton an attractive method for representing shape and, consequently, provide strong motivation to overcome its limitations. In fact, the symmetry axes of shapes have been shown to play an important role in human shape perception [90], providing further motivation for
research in this area. The medial axis is also a natural way of capturing the length and width of parts, which are shape features that are meaningful for indexing and matching shapes at varying levels of abstraction.

### 2.1.1 Part Attachment Instability of the Medial Axis

The instability of the skeleton due to part attachment has long been acknowledged by the shape representation community.\(^1\) While analysis of this instability has yielded stability measures ranging from very local (skeleton point) \([49]\) to semi-local (skeleton branches) \([92]\), most efforts can be viewed as skeleton processing, i.e., mapping input skeletons to output skeletons rather than mapping input skeletons to higher level shape abstractions. Moreover, evaluation is typically anecdotal (visual), lacking the context of a particular (e.g., recognition) task. The one exception is the shock graph \([89, 77]\), whose one-to-many mapping of skeletal segments to abstract parts unfortunately carries forward this skeletal instability.

The notion of ligature and its relation to the structural instability of the skeleton representation was first proposed by Blum and Nagel \([12]\). Much later, it was revisited in \([2]\) with several qualitative examples provided to demonstrate that with ligature regions labeled, the remaining portions of the skeleton appeared to be stable. In fact, the development in \([2]\) was guided by a principle of ligature removal: reconstructions without ligature appeared to lose little in terms of boundary detail. One such example was a set of hands with distinct skeletal topologies due to articulating fingers that became similar when ligature regions were labeled and ignored. It was also shown that certain deformations of the boundary, such as those resulting from evolution by curvature \([41]\), could swiftly lead to topological changes in ligature regions, providing further motivation for their removal. However, these developments fell short of a precise method by which ligature analysis could lead to a higher level of abstraction computationally. Furthermore, no direct attempt was made to apply these ideas to the then emerging techniques for skeletal graph matching, apart from the notion that ligature regions should be viewed as less salient.

The concept of skeletal ligature was also addressed by Katz and Pizer \([49]\) by associating measures of connection and substance to each skeletal point. In this work, the

\(^1\)The structural instability problem is different from that of handling “noise” due to small boundary perturbations, for which a number of effective pruning techniques have been developed \([80, 92, 5]\).
connection degree of a point corresponds to a continuous (rather than binary) measure of ligature that is computed as a sum of Gaussian-weighted distances from the skeletal points to their closest boundary points. In turn, the substance degree of a point is defined as the complement of the connection value. These measures are combined with rules of “visual conductance” to connect branches that can be perceived as belonging to the same part. The result is a fuzzy decomposition of a shape into potential parts, which is captured by a continuous connection value of each skeletal point. It is argued that by avoiding a binary decision between substance (i.e., parts) and connection (i.e., ligature) points, the uncertainty of part boundaries is better represented. Unfortunately, no experimental results for a recognition task are provided to evaluate this claim.

There are more recent approaches to dealing with topological instabilities induced by ligature via a type of skeletal simplification. Telea et al. [93] propose a Bayesian framework for skeletal simplification/smoothing which seeks the optimal balance between structural simplification and the reconstruction error that results from simplification. The method tends to collapse small internal and external skeletal segments, leading to a more stable skeleton that still preserves salient object parts. However, because the simplification was not based on a formal ligature analysis, ligature branches could survive. Moreover, branches were only removed and not merged, so that the ligature-induced oversegmentation problem was not addressed. van Eede et al. [98] extended this optimization work by basing the simplification on an ad-hoc ligature analysis. To preserve connectivity, contiguous ligature substructures were simply removed and replaced with linearly interpolated skeletons to preserve skeletal connectivity. Although producing more regular skeletons, this last step violated geometric constraints on the skeleton. Furthermore, ligature structure was still implicitly encoded, and ligature-induced oversegmentation was not overcome.

All the above approaches share one limitation in their detection of ligature. They assume that the ligature properties of skeletal points can be computed independently from the local influence of nearby boundary protrusions. This assumption is limiting because it is often violated even by simple shapes, as is shown in the example of Figure 1.2. This problem was addressed by Rom and Medioni [70] and by Juengling and Prasad [48] using a hybrid approach in which shape parts are found using boundary analysis and rules for concave corner associations, and then removed iteratively from the shape in order to compute their unperturbed medial axes. Tek and Kimia [92] follow a similar but non-hybrid approach based on a ligature-like analysis of a shape’s skeleton. In this
approach, parts are found and removed iteratively using the original skeletal information. Next, the medial axes of the removed parts and the last remaining part are recomputed to obtain their unperturbed medial axis.

Mi and DeCarlo [60] also take the approach of removing parts iteratively, but use the Smoothed Local Symmetries (SLS) representation [13] instead of the MAT. In this work, a similar concept to the MAT’s ligature is captured by detecting transitional areas between adjacent branches in the SLS representation. The influence of the order in which parts are removed in the part decomposition results is taken into account by constructing a dependency graph. The dependencies between candidate parts are determined by comparing the relative radii associated with the axial points in the transitional areas connecting adjacent parts, and represent the desire of removing smaller parts first. This approach is similar to the dependency graph that we use in our work to determine the order in which skeletal branches are merged (Section 3.5.2).

Mi et al. [61] extend the above approach in order to account for the influence of boundary intrusions on the part decomposition process. The proposed detection of boundary intrusions requires the computation and analysis of the shape’s background skeleton (i.e., the skeleton of the negative shape). This additional information, together with a novel part decomposition heuristic, allows for the detection of boundary intrusions. The intrusion “parts” are iteratively removed in order to obtain an unperturbed boundary, which is needed to correctly identify the underlaying shape parts. Since the removal of any shape part alters the boundary of the shape, the skeleton of both foreground and background shape is recomputed after each part removal. The detection and removal of intrusions is an important contribution of this approach, and a key difference with our own, which does not account for them (we discuss a possible solution to this problem in Appendix A).

There is also a probabilistic approach to dealing with the oversegmentation of skeletal branches, which is proposed by Singh and Feldman [31]. In this work, the concept of ligature does not play a role. Instead, a Bayesian probabilistic model is constructed in order to estimate the set of skeletal branches that are most likely to have produced the shape. The selection of branches is based on a maximum a posteriori approach and a prior probability distribution that expresses a preference for straight axes. While the main focus of this work is to reduce the presence of spurious branches due to boundary noise, the oversegmentation problem is also addressed by assuming that every junction point between three or more branches is an oversegmentation point, and eliminating the
junctions by merging two of the branches incident on each of them. However, since some
junctions may not be true oversegmentation points, the result is a skeleton that reduces
oversegmentation at the expense of introducing undersegmentation.

Aslan et al. [1] address the problem of skeletal oversegmentation by computing a
disconnected medial axis using a regularization procedure. This approach is related
to the multiscale computation of the medial axis [58, 69], but instead of associating a
shape with a multitude of skeletons, it selects a single scale. The scale is selected by
letting the smoothing of the boundary tend to infinity during the computation of the
propagating front whose singularities (or shocks) lead to the medial axis [50]. The result
is a disconnected skeleton that has fewer branch junctions than a regular skeleton, but
still captures the medial axes of all shape parts. Unfortunately, the resulting branches
tend to be significantly longer than the parts they represent and their adjacency relations
are not clearly defined.

The approaches based on iterative part removals discussed above address both the
under- and oversegmentation instabilities of the medial representation of shape, and
therefore are the most similar work to the shape parsing approach presented in Chapter
3. However, since they do not propose a representation and matching framework for
recognition, they cannot be directly compared to the bone graph (Chapter 4). In contrast,
Siddiqi et al. [89] focus on the undersegmentation problem of skeletons, and propose a
shape abstraction with recognition in mind. The undersegmentation of skeletal branches
is addressed by partitioning each branch into segments of monotonically varying radius
values. The resulting set of parts is used to construct the shock graph representation. The
shock graph is a directed acyclic graph (DAG) encoding a coarse-to-fine decomposition
of a shape into skeletal parts. Since these parts correspond to a partitioning of skeletal
branches, the shock graph can be regarded as a one-to-many mapping between branches
and parts, and therefore, it unfortunately inherits the oversegmentation instabilities of
the medial axis. In Section 3.6, we discuss the partitioning method of shock graphs,
and in Section 4.3 we compare shock graphs and bone graphs in a set of recognition
experiments.

2.2 Parts-Based Shape Representations

The recovery of shape parts is often followed by the construction of a graph-based repre-
sentation of the attributes and relations of the parts. The approaches in this area differ in
the types of graphs and in the attributes of nodes and edges that are considered, which, in turn, lead to different solutions to the shape matching problem. In our review of the related work, we focus on parts-based representations that are based on the medial axis. We begin by briefly reviewing the main differences between the alternate definitions of the medial axis, and then discuss the parts-based representations that are based on them.

The medial axis transform (MAT) of Blum [11] is the most studied and widely used medial shape description, and is the one that we consider in our work. When seen as a part decomposition method, the branching topology of the MAT can suffer from instabilities due to part attachment (as discussed in Sec. 2.1.1) and to small boundary perturbations. There is a significant body of work concerned with this latter type of instability. The predominant approach is to detect and prune the branches associated with small boundary perturbations [80, 92, 5]. However, there is also relevant work on alternate definitions of the medial axis that lead to a branching topology that is less sensitive to this type of instability [103, 38, 99]. In addition, there is relevant work on generalizing the MAT to include external symmetries (i.e., symmetries that are not entirely contained within a shape’s boundary) [13, 14, 36].

A popular and straightforward approach to shape parsing is to take the branches of a skeleton (Blum’s MAT or its variants) and assume that they map one-to-one to the medial axes of shape parts. The result is a parts-based representation called a skeleton graph [103, 72, 76, 4], which is an instance of an attributed relational graph (ARG) [30]. In this graph, the edges are attributed and undirected and each encodes a skeletal branch, while the nodes encode the endpoints (junctions and terminal points) of the branches. The object recognition approaches based on the skeleton graph address the instability of the representation due to part attachments at matching time. For example, Zhu and Yuille [103] use an error-correcting tree search approach (see Sec. 2.3 for details) that edits the query skeleton graph during matching by cutting, merging, shifting and concatenating nodes according to a set of rules and costs. Bai et al. [4, 3] take a different approach, and simply avoid comparing the branching topology of skeleton graphs. Instead, each terminal branch (i.e., an edge incident on a node with degree one) is associated with a shape descriptor that encodes the arc lengths of the paths that connect the branch with every terminal point on the skeleton. During matching, the similarity of edge descriptors is measured using the optimal subsequence bijection (OSB) algorithm [53], which is able to find correspondences among descriptors (i.e., vectors of path distances) with different numbers of components. The best correspondence between the terminal branches of two
Chapter 2. Related Work

graphs is found by solving a maximum weight bipartite matching, and used to compute a measure of graph similarity.

Connell and Brady [19, 20] represent the generalized medial axis of Brady and Asada [13] as a hierarchical semantic network (HSN) in which the nodes represent either shape parts or relational predicates. The predicate nodes with degree one encode the attributes of shape parts, while those with greater degree encode the attributes of the relations between shape parts. The predicate nodes also encode whether a relation is directed, e.g., *part A is attached to part B*, or undirected, e.g., *part A and B are connected*. It is interesting to note that the HSN is conceptually similar to our bone graph representation (Chapter 4), as it can be seen as an attributed mixed graph with nodes encoding shape parts and edges encoding the relations between parts. However, the two representations differ in that the relational attributes of the HSN attempt to capture qualitative properties such as “is bigger than”, “it has” and “it is attached to”, while the edges of the bone graph encode only the parameters of attachment relations. The HSNs are matched using an interpretation tree search [43].

An alternative to the skeleton graph representation is the shock graph [89]. The shock graph is a hierarchical representation of the part structure of a shape, and is usually encoded either as a rooted tree [66] or as a DAG [89, 57]. The nodes of a shock graph correspond to the partitioning of skeletal branches into segments with constant or monotonically varying radii, whereas the edges of the graph encode the branch segment connectivity. The edge orientations capture a hierarchical ordering of the nodes, which is determined as a function of the relative sizes of the shape parts represented by each node [86]. Like the skeleton graph, the shock graph inherits the part attachment instability of the medial axis, and passes it to the graph matching stage.

Goh [37] proposes a parts-based representation that encodes an over-partitioning of a skeleton into subparts, which are merged at matching time. In contrast to the shock graph, the subparts do not correspond to the partitioning of skeletal branches, but instead are obtained by partitioning the result of merging skeletal branches according to orientation continuity criteria. The skeletal subparts become nodes in a graph that is transformed into a rooted tree at matching time. Moreover, the edge attributes of this graph are also determined at matching time, and encode the relative orientations of the subparts with respect to the root nodes (of each graph) that lead to maximum graph similarity. The merging of subparts during matching depends on the information provided by the multiresolution gradient vector field (MGVF) of Goh and Chan [38], which
is also used to compute the medial axes of the shapes. The MGVF provides a measure of skeletal ligature that is used to guide the merging of branches during the matching process. The MGVF-based parsing and representation is similar to the work presented in Chapters 3 and 4. However, our work uses generic properties of the MAT to parse a shape, and constructs a representation that does not vary during the matching process.

In summary, a common aspect of the representations discussed above is that they pass the problem of dealing with part attachment instabilities to the matching algorithm. The drawback of this approach is twofold. First, it poses a more challenging matching problem that, in addition to viewpoint and within-class shape deformation, must accommodate part attachment instability. Second, it makes it more difficult to exploit part structure information for indexing and learning, as the parsing of a shape is only fully determined with respect to the (parsing of) other shapes to which it is compared. In contrast, we propose a parts-based representation that abstracts out the part attachment instabilities during the construction of a shape’s representation, allowing the matching algorithm to focus on other types of shape differences.

### 2.3 Inexact Graph Matching

The problem of matching graph representations of shape has received considerable attention in the vision research community. The graph matching task demands an algorithm that combines differences in graph topology and node and edge attributes into a continuous measure of graph similarity. Since this problem (subgraph isomorphism) is NP-complete (maximum common isomorphic subgraph is NP-hard), the choice is either to seek an approximate solution or to consider graphs that can be matched in polynomial time (e.g., rooted trees). In addition, domain-dependent constraints may be assumed to simplify the matching problem. For example, Umeyama [97] proposes a matching algorithm for graphs of equal size using an eigen-decomposition method to find the permutation matrix that minimizes the difference of the edge weights. Another example is the work of Gold and Rangarajan [39], who use a graduated assignment heuristic to find the largest isomorphic subgraph between two attributed graphs. Unfortunately, these specific constraints are too limiting for the task of matching shape representations, since the graphs representing similar shapes might not have the same number of nodes or large isomorphic subgraphs.

Conte et al. survey the field of exact and inexact graph matching in pattern recog-
inition in [21] and [22]. In this section, we review the body of work in the area that is most relevant to the problem of matching graph-based shape representations. The most popular classes of graphs in this area are given by trees, directed acyclic graphs (DAG), and attributed relational graphs (ARG) [30]. The attributed tree- and DAG-based representations encode hierarchical relations between nodes (i.e., shape parts), which, in turn, provide structural constraints that simplify the matching problem. On the other hand, an ARG is a more general structure than a tree or a DAG (it is an attributed undirected graph that might contain cycles), but provides weaker constraints to exploit during matching.

Pelillo et al. [64, 65] match node-attributed trees by constructing a maximum weight clique problem. This algorithm looks for the set of nodes in the query and in the model graphs that preserve the hierarchical constraints imposed by the representation while maximizing the pairwise node similarities among the nodes. Another matching approach for node-attributed trees is the work of Sebastian et al. [78], which measures the edit distance between two graphs (this is also known in the literature as error-correcting or error-tolerant algorithms [21]). The graph edit distance is defined as the minimum cost of the deformation path that makes two graphs isomorphic. Four edit operations are defined to deform a graph representation of one shape into another. Three of these operations allow for different types of merges and deletions of nodes, while the fourth operation allows for altering node attributes. The merge operation allows for assigning many-to-many correspondences between nodes (or one-to-many, if merges are applied only to the query graph) and can be used to find similarities at higher levels of abstraction. However, this flexibility comes at high computational cost, as finding a common graph between largely dissimilar graphs can define an extremely large space of possible edit operations. This is particularly problematic for cluttered or occluded scenes, where much of the edit cost may be due to the removal of extraneous clutter.

Pelillo et al. [67] also propose a solution to the many-to-many matching of node-attributed trees by reducing tree isomorphism to the problem of solving a maximum weight clique in an association graph. Their solution to the matching problem uses replicator dynamical systems from evolutionary game theory. In more recent work, Demirci et al. [25] present a framework for many-to-many matching, where features and their relations are represented using directed edge-weighted graphs. The method begins by transforming the graph into a metric tree. Next, using graph embedding techniques, the tree is embedded into a normed vector space. This two-step transformation reduces the
problem of many-to-many graph matching to a simpler problem of matching weighted distributions of points in a normed vector space. The distance between two weighted point distributions is computed using the Earth Mover’s distance [18, 73].

For the case of node-attributed DAGs, Shokoufandeh et al. [84, 83] propose a recursive bipartite matching algorithm in which a node similarity function measures both the attribute similarity of nodes and the topological similarity of the subgraphs rooted at each node. The result is a one-to-one assignment of node correspondences that yields a similarity value between arbitrary DAGs. This approach has been used to match shock graphs [84] as well as multiscale blob decompositions of shape [83]. In this latter approach, edge attributes are not accounted for explicitly, but can be embedded in the node similarity function. For example, in [83] the relative orientation of parts (i.e., edge attributes) is seen as a parameter of the node similarity function. However, since edge attributes are not compared explicitly, the ability of the matcher to independently measure structural and geometrical similarity is reduced.

All the above matching approaches suffer from one important limitation; they either ignore edge attributes or allow only for scalar edge weights. A popular method for matching fully-attributed graphs is to treat the problem as a tree search with backtracking. The basic mechanism in this family of approaches is to iteratively grow a solution set (initially empty) by adding node correspondences that are compatible with the mappings already in the set. The search is usually guided by the cost (or similarity weight) of the partial mapping spanned by the solution set and a heuristic that estimates the cost of matching the remaining nodes. The heuristic allows the matching algorithm to prune unfruitful search paths (and backtrack the population of the solution set), and/or to determine the order in which the search tree is traversed. The search is usually performed using a depth-first or best-first strategy.

In the pattern recognition literature, the tree search approaches are most frequently applied to ARGs, and seek to find the edit operations with minimum cost that make a model graph isomorphic to (a subgraph of) a query graph. The seminal work in this area is due to Fu et al. [95, 96, 74, 29, 30]. Later, Wong et al. [101] improve the search heuristic in [96] by taking into account, in addition to node mappings, the future cost of edge mappings. In turn, several authors improve the computational performance of the search by providing admissible heuristics and using the A* algorithm [44]. Berretti et al. [7, 6, 8] propose an admissible heuristic based on estimating the future matching cost by solving a maximum-weight bipartite graph matching problem, which leads to
an improvement in computational performance when compared to the work in [101]. Gregory and Kittler [42] propose a computationally efficient admissible heuristic that only accounts for the future cost of unmatched nodes. Finally, Dumay et al. [28] also cast the matching problem as an A* search.

Our algorithm for matching fully-attributed DAGs is a tree search approach, and in that respect it is similar to the previous work on ARGs. However, it differs from such work in the way it treats noisy nodes and edge attributes. In the case of ARG-based methods, noisy nodes are matched (with a cost) to a special null node that represents the node deletion operation in a graph edit distance. The addition of null nodes (one for each graph) increases considerably the number of possible solutions that must be investigated. In our approach, the hierarchical structure of DAGs is used to constrain the node correspondences at each iteration of the algorithm, which allows for solutions with unmatched noisy nodes without adding null nodes. Another important difference is that the ARG matching approaches seek a mapping that preserves node adjacency by editing the graphs to create an isomorphism, while this is not the case in our approach (unless node adjacency preservation is specified by the domain constraints). This results in a different treatment of edge similarity, since in our case there is no strict one-to-one correspondence between edges if node adjacency is not preserved.

A different family of methods is based on formulating the problem as a continuous non-linear optimization. Fischler and Elschlager [32] propose a relaxation labeling approach that assigns a label to each node in the model ARG that determines its correspondence to a node in the query ARG. The algorithm begins by computing the probability of each possible label assignment as a function of the node attributes and any other local information, such as the attributes of the edges incident on the model and query nodes. In subsequent iterations, the probabilities are updated by taking into account the assignment probabilities of neighboring nodes until either convergence or a maximum number of iterations are reached. A similar approach is proposed by Kittler and Hancock [51]. Christmas et al. [17] extend the relaxation labeling approach to consider edge attribute information at each iteration (i.e., not just in the initialization step). In the same spirit, Wilson and Hancock [100] propose an error-correcting method that exploits structural constraints by defining a probabilistic dictionary of consistent node mappings for neighborhoods of nodes (augmented with null nodes to represent deletions). The matching solution in this formulation is given by the maximum a posteriori (MAP) estimate of a Bayesian formulation of the node correspondence probabilities. Huet and Hancock [47]
extend this approach to consider edge attributes in the probability of node neighborhood transformations encoded by the consistency dictionary in [100]. Similar probabilistic error-correcting approaches based on relaxation labeling are proposed by Myers et al. [63] and by Torsello and Hancock [94] (for trees). Finally, Luo and Hancock [55] use the EM algorithm [26] (instead of relaxation labeling) to find the MAP solution to a probabilistic graph matching problem in which the query graph is treated as observed data and the model graph acts as hidden random variables.

The probabilistic methods described above do not enforce a one-to-one mapping between nodes (they allow for one-to-many correspondences), and yield matches that are not symmetric (the results depend on whether each graph takes the role of the model or the query). These properties are appropriate for some domains but are not desirable in others. In contrast, the matching algorithm that we present in Chapter 5 yields a one-to-one node mapping that does not depend on the role of each graph. Our approach does bear some resemblance to the probabilistic methods in that the similarity of node attributes may be updated at each iteration of the algorithm as a function of the current matching information, whereas previous tree search approaches assume that the node attribute similarities are constant.
Chapter 3

Shape Parsing Using Skeletal Information

3.1 Introduction

The junctions and terminal points of the medial axis have been used in the shape community to decompose a shape into parts. The predominant approach in the literature is to assume that the skeletal branches, i.e., segments of skeletal points delimited by junctions or terminal points, correspond to the medial axes of one or more shape parts. This assumption reduces the part decomposition problem to that of finding the partitioning of branches that yields a one-to-one mapping between medial axis segments and shape parts. However, as noted by August and Siddiqi [2], in the case of part protrusions, the branch structure of the skeleton has more branches than the number of shape parts perceived by a human. For example, a human might represent the shape in Figure 3.1 (a) as composed of three parts, while the skeleton of the shape has five branches (Fig. 3.1 (b)). Thus, in this case the one-to-many mapping of branches and parts can be seen as an oversegmentation of the desired part structure. This phenomenon may create an instability in the part decomposition of shapes, as is shown in Figure 1.1, where similar views of a 3-D object are represented by different numbers of skeletal branches.

In this chapter, we argue that a one-to-many relation between medial axis segments and parts can be obtained by eliminating the junctions associated with protrusions from the medial axis. Moreover, we show that skeletal and boundary properties can be used to determine which branches must be merged in order to remove the unwanted junctions. The removal of junctions leads to new branches that correspond to the medial axes of
Figure 3.1: Parts and skeletons. (a) A human observer is most likely to decompose this shape into three parts, but the skeleton of the shape has five branches (b), which defines a many-to-one correspondence between the branches and (the medial axes of) the desired parts. (c) We seek a merging of branches that creates a (possibly disconnected) skeleton in which each branch corresponds to the medial axis of one or more parts. In this case, the two junctions that connect the vertical and horizontal branches end-to-end in (b) are replaced by a smooth interpolation of the vertical branches, which defines the medial axis of the host shape part. The adjacency information of the now disconnected branches is preserved as a list of end-to-side branch attachments (i.e., protrusions), which augments the information contained in the skeleton. Once the junctions associated with the protrusions are removed from the skeleton, the part decomposition problem is cast as the partitioning of the branches into segments that map one-to-one to the medial axes of the desired shape parts.
one or more parts. Each junction removal disconnects the skeleton – the connections between the branches incident on the removed junctions are eliminated (Fig. 3.1 (c)). In order to preserve the branch adjacency information, we augment the skeletal description with a list of the branches that now have an end-to-side adjacency relation due to the branch merges.

We complete our shape parsing algorithm by showing that the same cues used to detect protrusions can be used to partition branches into skeletal segments that correspond one-to-one with the medial axes of shape parts. In contrast to previous medial partitioning algorithms (e.g., [89, 77]), we do not require every skeletal point to belong to the medial axis of a part. Instead, we identify two types of skeletal segments, which we call bone and ligament (Fig. 3.2). The bones map one-to-one to the medial axes of shape parts, while the ligaments connect bones to other bones (i.e., like the fibrous tissue with the same name in the anatomy literature). Hence, the bones provide a decomposition of a shape into parts, and the ligaments, together with the branch adjacency information, describe the attachment relations between these parts. In Chapter 4, we use the parts and relations recovered by our shape parsing algorithm to construct a novel parts-based representation of shape.

3.2 Local Geometry of the Medial Axis

In order to discuss the detection of protrusions using the information provided by the medial axis, we must first present the skeletal properties that we shall use. Let \( \Omega \) be the set of all points \((x, y)\) within the interior of a 2-D object, with \(S(\Omega)\) its medial axis [11]. Each skeletal point in \(S\) is characterized by a position \(p\) and a radius \(r\). The relationship between the object angle \(\theta\), the spoke vectors \(b^{\pm1}\), and the direction of the unit tangent vector \(T\) (see Fig. 3.3) is given by

\[
\theta = \arccos \left( -\frac{dr}{ds} \right),
\]

where \(s\) is the arc length along the medial curve. The object angle is expressed with respect to the unit tangent in the direction of decreasing radius along the curve [87]. The variation of radii along the curve is also used in the literature to define a sense of flow [35]. The direction of flow is given by the direction in which the radius increases and, when necessary, it is indicated as an arrow on the medial curve (e.g., see Fig. 3.4).
Figure 3.2: Bones and Ligaments. (a) We begin the parsing process with a given medial axis. (b) Next, we detect the branch junctions that correspond to boundary protrusions and merge the medial axes of the parts that host such protrusions. The result is a skeleton whose branches now map one-to-many to the desired shape parts; here two branches undersegment the medial axes of three parts. (c) Finally, we partition the skeletal branches into ligature (green points) and non-ligature (black points) segments. Intuitively, a ligature segment represents the symmetry axis of boundary points that form one or two concave corners and contributes little to the reconstruction of the boundary. In this partitioning, the non-ligature segments, called bones, map one-to-one to the medial axes of shape parts, while the ligature segments, called ligaments, connect bones to other bones.

The degree of a skeletal point is given by its number of adjacent points. Endpoints have degree one and junctions have degree three or greater. A branch is a segment of skeletal points that terminates at endpoints or junctions, and whose internal points have degree two. The adjacency relations between branches define the branching topology of the medial axis [87]. Junction points connect three or more branches. For simplicity, and without loss of generality, in this chapter we will focus exclusively on generic junctions. A junction is said to be generic if it maintains the degree of incident branches under arbitrarily small perturbations of the boundary curve [35]. There are two types of generic junctions in the medial axis: (1) junctions of three branches with only one outward flowing branch; and (2) junctions of three branches, all of which are inward-flowing. These two flow patterns are shown in Figure 3.4.

The ratio of boundary length to medial axis length defined by a skeletal branch or a subsegment of it is another property that we shall use in this chapter. Any branch segment defines two boundary-to-axis ratios (BAR), each obtained by dividing the length
Figure 3.3: Local geometry of a skeletal curve (adapted from [87]). The maximum inscribed disc at a regular skeletal point $p$ with radius $r$ touches the boundary at two bitangent points $b^{+1}$ and $b^{-1}$, defining two spokes emanating from $p$. The angle between the unit tangent $T$ (to the medial curve) and either spoke is $\theta$, the object angle.

Figure 3.4: (a) The two generic cases of branch junctions (adapted from [35]). The arrows correspond to the direction in which the radius functions increase and represent a notion of flow direction. (a) the radius functions of all branches increase toward the junction. (b) the radius function of one branch increases away from the junction, while the radius functions of the other two branches increase toward the junction. (c) The remaining flow patterns are not possible [35].
of the associated boundary on each side by the length of the segment [12]. The boundary associated with a skeletal segment is defined by the spoke endpoints corresponding to the first and last point of the skeletal segment.

3.3 Intuition and Goals

The perception of protrusions, i.e., the identification of a relation between shape parts in which one part is seen as protruding from another, depends largely on the domain (e.g., human silhouettes, typographic characters, etc.). Thus, it is difficult to formulate a method for detecting protrusions that is appropriate for all tasks. Instead, our goal is to show that the medial axis provides important information for the detection of protrusions, and that the detection process can be parameterized according to the domain. The relation of the boundary protrusion and the medial axis is twofold. First, any significant boundary protrusion can be associated with a branch junction in the medial axis. Second, the boundary concavities created by the protrusion lead to special skeletal points known as ligature [11, 2]. These are skeletal points whose spokes end at a boundary concavity and can be grouped into segments that have a BAR smaller than one.

The ligature points of the medial axis provide a powerful cue for identifying part protrusions and other types of part attachments, as they combine boundary concavity and boundary symmetry information. Moreover, they can be used to decompose a shape into parts by identifying the skeletal segments that represent the shape areas where parts

---

1Some small protrusions may not create a branch junction, as we show in Appendix A.
attach to one another. This leads to a shape parsing approach in which ligature information is used to reason about part attachments and to account for their interdependencies. In this approach, part attachments are labeled according to both the number of parts and the role of each part in the attachment. For example, a boundary protrusion represents a directed relation between two shape parts, since one part is considered to be protruding from another. In contrast, other attachment types, such as those shown in Figure 3.5, do not define distinct roles in the attachment, and therefore, correspond to undirected relations. The result of our parsing approach is a partitioning of the medial axis into non-ligature and ligature segments, which are called bones and ligaments, respectively. A bone is a segment of non-ligature points representing the medial axis of one shape part, while a ligament is a segment of ligature points acting as the medial “tissue” between bones.

In order to discuss the relation between the attachment of parts and the ligature properties of the medial axis in more detail, it is convenient to introduce the appropriate terminology used in the literature [11, 2]. A segment of skeletal points can be labeled as non-ligature, full-ligature, or semi-ligature. A full-ligature segment is the set of connected points of a skeletal branch associated with a pair of opposing concave corners on the shape’s boundary (Fig. 3.6). Similarly, a semi-ligature segment is the set of connected points of a skeletal branch associated with a single concave corner on the shape’s boundary (to one side of the segment). Ligature segments are said to be nested when they share a branch junction point. Moreover, we consider boundary protrusions to be nested if they induce nested ligature segments. Examples of full-, semi- and nested ligature are shown in Fig. 3.6, with full-ligature segments shown in red and semi-ligature segments shown in blue.

Figures 3.6 (a) and (b) illustrate examples in which the boundary concavities of the shapes lead to the partitioning of a branch into ligature and non-ligature segments. This results in the decomposition of the shapes into two parts whose medial axes are given by the non-ligature points. Figures 3.6 (c–h) illustrate examples of different types of protrusions and the pattern of junctions and ligature points associated with them. In particular, Figures 3.6 (c) and (d) represent cases of simple protrusions, while Figures 3.6 (e–h) represent cases of nested protrusions. Since ligature configurations can be quite complex when they are nested, an important component of our parsing algorithm is a procedure to untangle such cases.

We can summarize the main steps of our shape parsing algorithm as follows:
Figure 3.6: Examples of ligature segment configurations (the basic cases a, b, and c are adapted from [2]): (a) Full ligature (red) segment (induced by a pair of concave corners), and (b) Semi-ligature (blue) segment (induced by a single concave corner) that partition a branch into ligature and non-ligature segments; (c) Full ligature (red) and semi-ligature (blue) segments (induced by a pair of concave corners) associated with a boundary protrusion – the blue cusp actually represents two adjacent semi-ligature segments that meet at the apex of the cusp; (d) A pair of semi-ligature (blue) segments (induced by a single concave corners) associated with a protrusion that creates only one acute boundary concavity; (e) and (f) Examples of complex configurations of full-on-full and full-on-semi nested ligature (induced by three concave corners). (g) and (h) Examples of other forms of nested ligature (induced by four concave corners).
Chapter 3. Shape Parsing Using Skeletal Information

1. Compute the skeleton of the input shape;

2. Identify the skeletal points that form ligature (Sec. 3.4);

3. Label each branch junction according to whether it is the result of the protrusion of a shape part from another or not (Sec. 3.5). This step requires the recursive processing of nested protrusions (Sec. 3.5.2) and the merging of the branches representing the medial axes of the host parts (Sec. 3.5.3). The result is a skeleton in which the junctions associated with protrusions are removed (Sec. 3.5.1);

4. Partition the branches of the new skeleton into non-ligature and ligature segments in order to find their bones and ligaments. A maximal segment of non-ligature points defines a bone, while a maximal segment of ligature points defines a ligament (Sec. 3.6).

The output of the algorithm is a set of skeletal segments, each labeled bone or ligament, and the adjacency relations between the segments. In the following sections, we present the details of each step of the algorithm. In Section 3.7, we conclude the presentation of the algorithm with a complete procedural specification of it.

3.4 Detecting Ligature

The algorithm for ligature detection developed in [2] works by first identifying strong negative curvature minima on the boundary (by using a curvature measure) and then labeling those skeletal points whose bitangent points \( b^{+1}, b^{-1} \) fall within an \( \epsilon \)-ball of the same negative curvature minima. In this approach, the negative curvature minima are computed at a fixed (boundary length) scale. We seek a more robust approach in which local shape properties dictate the scale of negative curvature minima detection. For example, Figure 3.7 illustrates two types of corners defined by boundary segments of different lengths. The boundary length of the “sharp” corner is significantly smaller than that of the “smooth” corner.

A general notion of ligature must account for the scale of the boundary corners that give rise to the ligature points, such that both sharp and smooth corners can induce skeletal ligature. As a guiding principle, it is useful to consider the following necessary condition of ligature points:
Terminology 3.1. A ligature segment is a skeletal segment with the property that at least one side has boundary-to-axis ratio smaller than one, and whose spokes map to a boundary interval of negative curvature points.

This general notion of ligature can be used to search for ligature segments by first evaluating their boundary-to-axis ratios, and then detecting the presence of curvature minima at scales appropriate for the boundary intervals spanned by the segments.

We propose a novel algorithm for detecting ligature that accounts for both sharp and smooth concave corners. Our algorithm begins by searching for candidate ligature segments whose boundary-to-axis ratios (BAR) are significantly smaller than one (we use a BAR threshold of 0.75 in our experiments). Next, we identify three different ways in which the spokes emanating from one or two ligature segments may sweep the boundary points that form a concave corner. For simplicity of exposition, we discuss the details of these three configurations in the caption of Figure 3.8. By identifying these configurations, we are able to determine the boundary interval $[\hat{b}_0, \hat{b}_1]$ that is expected to be a concave corner. For configurations (a) and (b) in Figure 3.8, the boundary interval is defined by the spokes emanating from two adjacent candidate ligature segments. For configuration (c) in Figure 3.8, the boundary interval is spanned by the spokes emanating from a single candidate ligature segment.

Given a boundary interval $[\tilde{b}_0, \tilde{b}_1]$, we evaluate whether it forms a concave corner by following the approach of Chetverikov and Szabo [16]. However, we modify this procedure in order to account for the scale information provided by the candidate ligature segment.
Figure 3.8: The three configurations of concave corners and ligature segments considered by our ligature detection algorithm. We identify these configurations according to the relative flow directions of adjacent ligature segments (shown as arrows next to each segment). (a) The spokes emanating from two adjacent ligature segments on different branches sweep a set of connected boundary points that form a concave corner. In this case, the ligature points flow toward the branch junction. Since each of these two ligature segments may map only to a subset of the corner points, we consider the union of their spokes (on the corner sides) when defining the boundary interval that is expected to form a corner. (b) A similar phenomenon occurs when two adjacent ligature segments on the same branch have a radius function that increases away from their adjacent point (i.e., they form a neck shape). In this case, it is also necessary to consider the union of their spokes (on the corner sides) when defining the boundary interval to evaluate. In contrast, the spokes of the ligature segment with homogeneous flow direction in (c) sweep the entire set of boundary points forming the concave corner. Our ligature detection algorithm begins by evaluating all candidate ligature segment with homogeneous flow in a skeleton. This requires the evaluation of adjacent segments in order to determine whether the candidate ligature segment is part of configurations (a) or (b). If the segment is not part of either configuration, we assume the configuration (c) for it.
Chapter 3. Shape Parsing Using Skeletal Information

Figure 3.9: Boundary Corner Detection. We aim to determine whether the boundary interval $[\tilde{b}_0, \tilde{b}_1]$ forms a concave corner or not. Each point $\tilde{b}$ within the boundary interval $[\tilde{b}_0 - d, \tilde{b}_1 + d]$ is considered the middle vertex of all the triangles that can be inscribed in the interval. For example purposes, assume that only three triangles can be inscribed in this example, and that their middle vertices have angles $\theta_1$, $\theta_2$, and $\theta_3$, respectively. Following the corner detection algorithm presented in [16], we let the angle $\theta_{\tilde{b}}$ of boundary point $\tilde{b}$ be the smallest angle formed by its inscribed triangles, i.e., $\theta_{\tilde{b}} = \min(\{\theta_1, \theta_2, \theta_3\})$ in this example. The interval $[\tilde{b}_0, \tilde{b}_1]$ is considered a corner if the angle $\theta_{\tilde{b}}$ associated with each point is: (1) less than the angles associated with every point in $[\tilde{b}_0 - d, \tilde{b}_1 + d] - [\tilde{b}_0, \tilde{b}_1]$, and (2) less than a given (concavity) threshold $\theta_\tau$.

The parameter $d$ is related to the scale of the corner, and we define it as $d = \frac{\tilde{b}_1 - \tilde{b}_0}{2} + d_0$. Then, we consider the interval $[\tilde{b}_0, \tilde{b}_1]$ to be a concave corner if the angle $\theta_{\tilde{b}}$ associated with each point in the interval is: (1) smaller than the angle $\theta_{\tilde{b}}$ of every point in the surrounding neighborhood of the interval, i.e., the set of points $[\tilde{b}_0 - d, \tilde{b}_1 + d] - [\tilde{b}_0, \tilde{b}_1]$, and (2) smaller than a given (concavity) threshold $\theta_\tau$ (as suggested in [16], we use a threshold of 150 degrees in our experiments).

Finally, we assume that any concave corner mapped by a junction point (e.g., spoke $\tilde{b}_j$ in Fig. 3.8 (a)) must induce ligature on two branches incident on the junction in order

---

2If a point does not correspond to the middle vertex of any inscribed triangle with valid side lengths, we let its angle $\theta_{\tilde{b}}$ be equal to 180 degrees.
Figure 3.10: Impossible junction and ligature configurations. Red, blue and gray colors correspond to full ligature, semi-ligature and non-ligature segments, respectively. We consider these configurations impossible due to the assumption that every concave corner associated with a junction induces ligature points on two of the adjacent branches incident at the junction. Hence, the configurations of full- and semi-ligature segments must be consistent with the concave corners spanned by their ligature sides. For example, here the top-left configuration has a full ligature segment adjacent to a non-ligature segment, which cannot occur given that if there is a corner, both segments must have ligature properties.

to be detected as a corner. While this assumption may not be appropriate for some very smooth corners (i.e., they may induce ligature on a single branch instead of two), it is convenient in order to rule out a significant number of otherwise possible ligature configurations around junctions, and, in turn, to simplify the detection of protrusions presented below. Such configurations are discussed in the caption of Figure 3.10.

3.5 Detecting Protrusions

We can now analyze the detection of junctions between the medial axes of protruding parts and their host parts. These junctions define a directed binary relation between parts; one shape part protrudes from another (Fig. 3.11 (a)). We refer to these junctions as P-junctions. In contrast, a junction that is not a protrusion corresponds to the connection between the medial axes of three shape parts (Fig. 3.11 (b)). These junctions define an undirected tertiary relation between parts; the parts are attached to one another, but none of them play a distinct role in the attachment relation. We refer to these junctions as Y-junctions.

To evaluate whether or not a junction is a P-junction, we shall analyze the ligature
Figure 3.11: Examples of P and Y junctions. (a) A P-junction represents a relation between two shape parts in which the medial axis of one part protrudes from the medial axis of a host part. In this case, the medial axis of the host part is formed by the two horizontal branches, while that of the protruding part is formed by the vertical branch. (b) A Y-junction represents a relation between three shape parts whose medial axes terminate at the junction point.

properties of the branch segments incident on the junction. In some cases, ligature information is not sufficient to discriminate whether a junction corresponds to a protrusion or not, or to unambiguously label the protruding branch and the host branches incident on a P-junction. For these cases, the detection of P-junctions requires the use of appropriate criteria that capture other perceptual cues, such as the relative thickness of attached parts and/or the continuity of skeletal branches (see Fig. 3.13). Thus, ligature information provides us with an important cue to detect protrusions, but must be combined with other cues in order to label junctions as P or Y.

We capture the information provided by the ligature segments incident on junctions in a taxonomy of all possible junction and ligature configurations. Such a taxonomy is shown as a table in Figure 3.12 (a), while some associated example shapes are shown in Figure 3.12 (b). The junction/ligature patterns (Fig. 3.12 (a)) are grouped according to whether they correspond to Y-junctions, non-nested P-junctions, or nested P-junctions in the example shapes (Fig. 3.12 (b)). The P-junctions considered nested in these figures correspond to junctions connected by ligature points (see Subsection 3.5.2). This taxonomy shows that in some cases, the same junction and ligature pattern may be associated with different boundary phenomena, and that most such cases correspond to cases of nested ligature. That is, when ligature is not nested, it becomes a powerful cue for identifying boundary protrusions. This observation motivates our approach for untangling the nested ligature configurations before labeling a junction as Y or P. Our procedure for detecting nested protrusions attempts to merge branches following an order that recovers
new and more reliable ligature information for labeling the nested junctions.

Our junction labeling algorithm can be summarized as follows. We perform a preliminary labeling of all junctions as either Y or P. This is discussed in Subsection 3.5.1 below. Next, we analyze the nested ligature and junction configurations in order to determine an ordering for merging the host branches based on the parent-child relations of the nested junctions. This is discussed in Subsection 3.5.2. The key property of this approach is that the merging of branches yields a smooth medial axis whose spokes meet the object angle property specified by Equation 3.1, and that has new ligature properties that may change the preliminary labeling of the parent junctions. The branch merging operation is discussed in Subsection 3.5.3.

### 3.5.1 Labeling Junctions as Y or P

We cast the problem of detecting protrusions as that of labeling junctions as Y or P, and then merging the two host branches associated with each P-junction (e.g., Fig. 3.11 (a)). The main cues that we consider when labeling junctions are the ligature properties of their incident segments. Frequently, the presence of a single full-ligature segment is a strong cue for detecting a protrusion. This fact is consistent with the results in the psychophysics community, which suggest that humans consider two near concave corners on opposite sides of a medial axis as a strong cue for decomposing a shape into parts [90]. For non-nested junctions, the presence of three full-ligature segments is a strong cue that the junction is not a protrusion (Fig. 3.13 (f)). That is, there is no segment acting as the protruding branch, and so there is a tertiary relation between the branches, which is represented by a Y-junction.

Labeling junctions using ligature information is more difficult when a junction has only semi-ligature points, or when its ligature points are nested with those of another junction. The former is the case in which the possible protrusion has a single boundary concavity, which is a weak cue since concavities are not formed exclusively by protrusions (e.g., Figs. 3.13 (c–e)). The latter is also a weak cue because the ligature points around the junction are perturbed by the presence of nearby boundary concavities. Thus, before labeling a junction, we must determine whether its incident ligature segments provide enough information to select an appropriate label for it or not. To this end, we consider the ligature information of a junction to be ambiguous if the junction is nested or if its incident ligature points form only semi-ligature. Otherwise, we consider its ligature
Figure 3.12: Taxonomy of junction and ligature configurations. (a) Row A in the table corresponds to the left junction type in Fig. 3.4 (a), and rows B and C correspond to the right junction type in that figure. The headings of each column show that the same ligature configuration may be labeled differently (see text for details). (b) Examples of the junction and ligature configurations presented in (a), with references to their corresponding rows and columns. Branches with constant radius, such as the non-ligature branch in row 2 column 1 in (b), can be seen as having decreasing and increasing radius if minor boundary perturbations are applied.
Figure 3.13: Example of the evolution of a junction and the limiting cases of its ligature configuration. A P-junction (a) can be transformed into a Y-junction (c) without changing the flow and ligature pattern of the junction. The interpretation of limiting cases, such as (b), as Y- or P-junctions is sensitive to specific perceptual preferences. Other limiting cases, such as (e), exist between changes of ligature geometry and are less sensitive to varying interpretations. For example, (d), (e) and (f) are all naturally interpreted as Y-junctions. In this chapter, we suggest labeling the cases (g-i) using only ligature information, which leads to the labeling of (g) and (h) as P-junctions and (i) as a Y-junction. However, the labeling of these cases could be made dependent on domain preferences by also considering other cues, such as the relative thickness of parts and/or the good continuation of the candidate branches for merging.
Figure 3.14: Example of Y- and P-junctions with similar ligature properties. In each example the junction point is associated with one concave corner, $b_0^{-1}$, which induces two semi-ligature segments (blue points). In order to label the junction, the relative thickness of the protruding and host parts is evaluated by comparing the distance between the spoke endpoints $b_0^{+1}$ and $b_0^{-1}$ against the radius of the junction, $r_0$. The condition that $\|b_0^{+1} - b_0^{-1}\|_2 < r_0$ is only met by case (a), which is labeled a P-junction. The junctions in (b) and (c) are labeled Y-junctions.

information unambiguous. In this model, the unambiguous configurations are given by the junction and ligature patterns in columns one, three, and four of the table in Figure 3.12 (a), as long as they are not nested.

Once we determine whether a junction and ligature configuration is unambiguous or ambiguous, we proceed to label the junction. We label the unambiguous patterns in column three as P-junctions, and those in columns one and four as Y-junctions. For the case of ambiguous junction patterns, i.e., those in the second column of the table and those formed by a pair of junctions connected by ligature points, we evaluate both the relative thickness of the protruding part and the curvature of the medial curve produced by merging the host branches. We evaluate relative thickness by comparing the distance between the spoke endpoints of the candidate protruding branch at the junction point against the radius of the junction point (see Fig. 3.14). We assume that a P-junction must have a distance between its spoke endpoints that is smaller than the medial axis radius at the junction (this heuristic is similar to the short-cut rule proposed by Singh et al. [90]). We assign the label P to a junction if it meets this relative thickness condition, and if the interpolated medial curve segment created by merging the host branches (Sec. 3.5.3) does not contain points of high curvature (for a given curvature threshold). Otherwise, we label it as a Y-junction.

The labeling of a junction as P also assigns the corresponding labels to the protruding
and host branches that define the identified protrusion. In turn, these labels and the ligature segments incident at the junction define a *boundary gap* formed by the protruding part, which is relevant for the labeling of nested protrusions (discussed below). Intuitively, a boundary gap is created by the (imaginary) removal of the boundary points that form the concave corner(s) associated with the protrusion and the boundary points represented by the protruding branch. The endpoints of the boundary gap are given by the two boundary points of the host part that had a neighboring point removed. The *boundary gap interval* is the open interval defined by the gap endpoints, and is defined such that it contains the boundary points of the protruding part. A more detailed depiction of the boundary gap associated with a protrusion is provided in Figure 3.15.

### 3.5.2 Nested Protrusions

The ligature and junction configurations can be quite complex when they are nested. This suggests that they must be analyzed carefully in order to choose an appropriate order for merging the branches that represent host parts. Two junctions are nested when the labeling of one of them (as P or Y) may depend on the labeling (and branch merging) of the other. This occurs when the spokes of a pair of junctions map to a common concave corner on the shape boundary (e.g., see Fig. 3.16 (a)). In this case, the two junctions are connected by a branch formed exclusively by ligature points, since the spokes between the pair of spokes mapping to the common corner must “sweep” the boundary points that form the corner.

Two nested protrusions can create boundary gaps on the same side or opposite sides of a host branch (a branch has exactly two sides). If the boundary gaps are along the same branch side, either one of them must be fully contained within the other (Fig 3.16 (a)) or they are adjacent (Fig 3.16 (b)) since medial axis spokes do not intersect [87] (i.e., two gaps sharing a corner cannot be partially contained within one another). Then, the nested configurations between two junctions can be divided into three cases according to the relative positions of the shared concave corner and the other concave corners associated with the junctions. We label each nested configuration according to which one of the following conditions they meet:

1. the boundary gap defined by the spokes of a junction is included within the boundary gap interval induced by the other junction (see Fig. 3.16 (a));

2. the ligature segments incident on the junctions map to concave corners on the same
Figure 3.15: Examples of boundary gaps induced by protrusions. A boundary gap is created by removing the boundary points that form the concave corner(s) associated with a protrusion and the boundary points mapped by the spokes of the protruding branch. The corner points are given by the interval of connected boundary points spanned by the endpoints of the spokes emanating from the ligature points on the protruding branch (labeled P) and the host branches (labeled $H_1$ and $H_2$) incident at the junction. In a clockwise ordering of the boundary points, the two corners of the protrusion in (a) are given by the intervals $[b_{x-1}^{-1}, b_{y+1}^+]$ and $[b_{y-1}^{-1}, b_{z+1}^+]$, while the corner of the protrusion in (b) is given by the interval $[b_{y-1}^{-1}, b_{z+1}^+]$. The endpoints of the boundary gaps are given by the two boundary points of the host part that had a neighboring point removed. Assuming that the first point of each branch corresponds to the junction point, we label the gap endpoints as $b_{x+1}^{-1}$ and $b_{z+1}^+$ in (c) and (d). Then, the boundary gap interval is given, clockwise, by $(b_{x+1}^{-1}, b_{z+1}^+)$. Finally, the removal of the skeletal points whose spokes map to the concave corners associated with the protrusion creates a skeletal gap. The interpolation of these points is discussed in Section 3.5.3.
Figure 3.16: Nested Protrusions. A pair of junctions is said to be nested if they are connected by ligature points ($L_0$ branches in Figures (a–c)). The type of nesting configurations between junctions is determined by the relative location of the boundary gaps defined by each protrusion (see Fig. 3.15 for details on boundary gaps). There are three possible cases: (a) one boundary gap is contained within the other defining a parent-child relationship. In (a), the child gap $[b_i^{+1}, b_j^{+1}]$ is contained within the parent gap $[b_k^{+1}, b_l^{+1}]$; (b) the boundary gaps are adjacent and connected by ligature points with opposite flow direction. This is the case with gaps $[b_w^{+1}, b_x^{+1}]$ and $[b_y^{+1}, b_z^{+1}]$ in (b). In (c), the boundary gaps are located on opposing sides of the shape part identified as the host of the two protrusions. For example, in (c), the gaps $[b_x^{+1}, b_z^{+1}]$ and $[b_w^{-1}, b_y^{-1}]$ are located on opposite sides of the host branches $H_1$ and $H_2$. 
skeletal side of the host shape part and define adjacent boundary gaps (see Fig. 3.16 (b));

3. the ligature segments incident on the junctions map to concave corners on opposite sides of the host shape part (see Fig. 3.16 (c)).

Configuration (1) defines a parent-child relation between P-junctions in which the boundary gap associated with the parent junction contains the boundary gap associated with the child junction. In this case, the branches of the child P-junction must be merged first, because the ligature properties of the merged branch are relevant for processing the parent branch (see Fig. 3.20 (a-c)). In contrast, the host branches of nested P-junctions with configuration (2) can be merged in an arbitrary order, because the ligature properties of the merged branches of one junction do not add new information for the processing of the other junction. Finally, configuration (3) presents the possibility of a special interpretation in which the protrusions correspond to two imaginary overlapping medial axes (see Figs. 3.17 (b) and (e)). This configuration may also be labeled as two P-junctions or two Y-junctions (see Figs. 3.17 (c) and (f)).

The labeling of configuration (3) depends on the perceptual preferences that are appropriate for the working domain. For example, Figures 3.17 (a) and (d) show two similar shapes and skeletons that differ only in the thickness of the skeletal branches, and that may induce a different part decomposition depending on perceptual preferences. In our experiments, we use the rules of relative thickness and good medial axis continuation discussed in Subsection 3.5.1 to label the junctions as either P or Y, and leave the possible interpretation as overlapping medial axes for future work. The complete set of possible interpretations of configuration (3) and the part attachment relations induced by them is discussed in Appendix B.

Since nested ligature cases may be formed by more than two junctions, we propose a procedure to label cases involving a multitude of junctions. We begin by labeling all junctions as either Y or P using the rules discussed in Subsection 3.5.1. Next, we create a dependency graph in which every node represents a junction point and its label, and every directed edge represents a dependency in the merging of host branches. That is, an edge from node $u$ to node $v$ implies that the host branches of the P-junction represented by $v$ must be merged before those of the P-junction represented by $u$. We add an edge in the graph for every nested P-junction labeled as configuration (1), and direct it from the child junction to the parent junction. We treat the nested configurations (2) and (3) as
Chapter 3. Shape Parsing Using Skeletal Information

Figure 3.17: Interpretations of Nested Protrusions. The ligature configuration in (a) can be naturally interpreted as two (imaginary) overlapping medial axes (b), or as two protrusions on the same medial axis (c). Similarly, the ligature configuration in (d), can be interpreted as two overlapping medial axes (e), or, given the comparable width of all the branches, as the attachment of four parts to a center part (f).

3.5.3 Merging Host Branches

The goal of merging host branches is to form a medial axis that represents a boundary of the host part without the gap formed by the protruding part. This boundary gap is, in turn, related to the skeletal gap formed by the skeletal points with spokes ending at the boundary concavities of the attachment (see Fig. 3.19 (a)).

The merging operation replaces the skeletal gap by a smooth interpolation of the position and radius values of the gap’s endpoints while respecting tangent continuity. The boundary gap is bridged by calculating the spokes of the interpolated points as specified by Equation 3.1, which relates the object angle to the first derivative of the
Figure 3.18: The Dependency Graph of Nested Protrusions. (a) Every branch junction, a–i, is labeled as either Y-junction, simple P-junction, or nested P-junction. (b) The parent-child dependencies between nested P-junctions are represented by a dependency graph. Graph nodes with out-degree equal to zero (i.e., independent nodes) are processed first and eliminated from the graph. This creates new independent nodes. The junctions represented by the new independent node are relabeled to account for the ligature information of the merged branches. This process is repeated until the dependency graph is empty. (c) The merging of host branches leads to an end-to-side adjacency relation between branches, which is encoded as a list of tuples (d) whose elements are the indices of the merged host branch and the protruding branch, the normalized position of the point in the host branch closest to the removed P-junction, and the side \{+1, −1\} of the protrusion on the host branch.

Figure 3.19: The branch merging operation: (a) In this example, the semi-ligature “arms” of the junction define the skeletal gap points. (b) The position and radius of the gap endpoints are interpolated by smooth radius and axial functions while preserving tangent continuity. (c) The spokes of each interpolated point are computed from the relation between radius, medial curve and object angle given by Equation 3.1.
radius function at each gap point. In turn, the object angle is used to rotate the point’s
tangent along the medial curve to obtain the spokes’ directions. These steps are depicted
in Figure 3.19. In our implementation, we perform a cubic polynomial interpolation of
the gap’s medial curve, and a linear interpolation of the gap’s radius function. Other
smooth functions can also be considered.

The interpolation method presented above is an efficient approach for merging branches.
However, this method can introduce small perturbations along the original shape bound-
ary opposite to the boundary segment being filled in. The reason for this is that the
spokes of the interpolated medial axis on the opposite side of the protrusion are not
constrained to terminate at the original shape boundary. If the exact preservation of the
input shape boundary is required, an iterative method could be used instead to obtain
tangents and radius values that meet these constraints. For the problem of shape match-
ing, we found that, in practice, the potential errors introduced by the simple interpolant
above are too small to justify the additional computational effort.

The skeleton of a shape becomes disconnected as a result of each merge operation,
since the original branch junction points are not necessarily interpolated (e.g., see red
and black points in Fig. 3.19 (c)). We preserve the original branch connectivity by
keeping track of the adjacency relations associated with each branch merging. The branch
adjacency information of all the removed P-junctions is preserved as a list, which also
specifies the closest point in the merged branches to the junction points removed, as well
as the side of the merged branches from which the parts protrude. This information is
later used to construct the graph-based representation discussed in the next chapter.

Finally, we note that the merging of branches, together with the preservation of medial
axis properties, is important for part segmentation and for shape recognition. For part
segmentation, the restoration of the boundary gap is necessary to compute the BAR
of the parent branch in nested ligature cases, as is shown in Figure 3.20. For shape
recognition, the restoration of skeletal information simplifies the comparison between
similar shapes with missing parts, as the individual parts now encode a similar boundary
contour.

## 3.6 Recovering Shape Parts

The branch merging process presented in the previous section yields a skeleton in which
each shape part is expected to map to exactly one skeletal branch. In this section, we
Figure 3.20: Example of branch merges helping in the restoration of nested ligature: (a) Nested boundary gaps form nested ligature; (b) The child protrusion is restored first so that the inner boundary gap is filled and ligature properties can be recomputed; (c) The parent protrusion is not nested anymore and can be processed recursively.

complete the shape decomposition process by partitioning each branch into segments that map one-to-one to the medial axes of shape parts. Unlike branch merging, skeletal branch partitioning is a well-studied problem in the related literature, where the predominant approach is that of shock graphs [89]. In such an approach, a branch is partitioned into maximal segments of either constant or monotonically varying radii, which produces shape parts with homogeneous flow directions (see Figs. 3.21 (a–d)). This partition function does not account for concave corners explicitly, but it is still able to identify shape parts induced by them (e.g., Figs. 3.21 (a) and (b)), as well as other natural parts induced by smooth boundary deformations (e.g., Fig. 3.21 (c)). However, the lack of an explicit account of concave corners is also a shortcoming of the approach, as their presence is not handled consistently. For example, the difference in flow direction of the bottom branches in Figure 3.21 (d) induces two different partitions, even though both branches relate to the same concave corner.

We seek a branch partition function induced by concave corners. A natural candidate for this function is the ligature analysis presented in Section 3.4. Under this scheme, a branch is partitioned into ligature and non-ligature segments, which represent skeletal parts induced by the negative curvature minima along the shape’s boundary (see Figs. 3.21 (e–h)). This partition leads to a natural association of roles for the two types of skeletal parts, in which the non-ligature segments provide the “support” of each shape part and the ligature segments provide the “glue” that holds the parts together. We refer to these roles as bones and ligaments, respectively. The ligature-based partition can also be combined effectively with other partition criteria, such as that of shock graphs, by
Figure 3.21: Shock graph partitions and ligature-induced partitions. Top Row: Shock graph partitions. (a) a two-part decomposition induced by a segment with monotonically decreasing radii and a segment of constant radius. (b) two parts induced by the segments with monotonically varying radii (a third part is given by their common point). (c) the same part decomposition of (b) is applied, even though no concave corners are formed. (d) the radius variation of the left “leg” induces a partition, while that of the right “leg” does not. Bottom Row: Ligature-induced partitions. (e) and (f) both depict a two-bone-and-a-ligament decomposition induced by the full ligature segments (red points). (g) one-bone decomposition induced by the lack of ligature segments. (h) three-bone-and-two-ligament decomposition induced by the two semi-ligature segments (blue points).
sub-partitioning the bones. For example, the shock graph partition of Figure 3.21 (c) can be applied to the single bone of Figure 3.21 (g) in order to capture the perceptual relevance of the smooth neck.

**Terminology 3.2.** Every branch of a restored skeleton is partitioned into segments, such that the medial axes of different shape parts map, one-to-one, to non-overlapping segments. This partitioning creates two types of skeletal parts, which are called bones and ligaments. A bone is a maximal segment of non-ligature points encoding the medial axis of a shape part. A ligament is a maximal segment of ligature points representing the connection of its adjacent bones.

It should be noted that a ligament can be formed by more than one ligature segment since, when chained together, these segments create a set of connected ligature points. For example, the non-ligature segments of Figure 3.21 (e) are joined by a ligament formed by a single ligature segment, while the non-ligature segments in Figure 3.21 (f) are joined by a ligament formed by two ligature segments with opposing flow direction. Furthermore, a ligament connecting two bones from the same branch, i.e., an internal ligament, defines a part attachment relation in which the bones of the shape parts are connected via skeletal points of degree two (i.e., skeletal points with two neighboring points). In contrast, the ligaments at the end of a branch, i.e., the external ligaments, are the result of part attachment relations that are already known from the labeling of Y-junctions and P-junctions. Thus, the labeling of internal ligaments is a mechanism for coping with part undersegmentation and for discovering the attachment relationships that are not related to branch junctions.

### 3.7 The Shape Parsing Algorithm

The following is a structured definition of the overall parsing algorithm. The goal here is to present the major steps of the algorithm along with the references to their corresponding sections in the text.

**Declaration of variables**

- $I$: shape image
- $S_0$: set of skeletal points
- $S_1$: set of skeletal points with P-junctions removed
- $P$: set of branch adjacency relations recovered from branch merges
- $L$: set of ligature segments $[s_0, s_1]$, for $s_0, s_1 \in S_1$
- $B$: set of bone segments $[s_0, s_1]$, for $s_0, s_1 \in S_1$
procedure \((B, L, P) = \text{shapeParsing}(I)\) 
\(S_0 = \text{computeSkeleton}(I)\) 
\((S_1, P) = \text{detectProtrusions}(S_0)\) 
\((B, L) = \text{partitionBranches}(S_1, P)\) 
end

procedure \((S_1, P) = \text{detectProtrusions}(S_0)\) 
\(L_0 = \text{analyzeLigatureAroundJunctions}(S_0)\); see Section 3.4 
DG = createDependencyGraph\((S_0, L_0)\); see Section 3.5.2 
\(P = \emptyset\); let \(P\) be the empty set of end-to-side branch adjacency relations 
\(S_1 = S_0\) 
while DG not empty 
\(v = \text{getIndependentNode}(DG)\); any node (i.e., junction point) with out-degree equal to zero 
\(L_0 = \text{updateLigature}(S_1, L_0, v)\); recompute ligature for all branches incident on \(v\) 
type = labelJunction\((v, L_0)\); type is either ’Y’ or ’P’ (see Section 3.5) 
if type = ’P’ 
\((S_1, P) = \text{mergeHostBranches}(v, S_1, P)\); see Section 3.5.3 
removeNode\((DG, v)\); removes the node and all its associated dependencies (edges) 
end

procedure \((B, L) = \text{partitionBranches}(S_1)\) 
\(B = \emptyset\); let \(B\) be the empty set of bones 
\(L = \emptyset\); let \(L\) be the empty set of ligaments 
for every branch \(b\) in \(S_1\) 
\(L_0 = \text{findAllMaximalLigatureSegments}(b)\); see Section 3.6 and Appendix A 
let \(B_0\) be the complement set of \(L_0\); i.e., the set of non-ligature points in \(b\) 
\(B = B \cup B_0\); add the new bones to the set of all bones 
\(L = L \cup L_0\); add the new ligaments to the set of all ligaments 
end

3.8 Conclusions

Previous medial descriptions of shape assume that there is a one-to-one or one-to-many relationship between skeletal branches and shape parts (e.g., [77, 89]). In contrast, we allow for this relationship to be many-to-many. This is motivated by our observation of the effect that part protrusions have on the medial axis. In the presence of part protrusions, the number of branches in a skeleton is greater than the number of shape parts perceived by a human observer. A protrusion is a relation between two parts in which one of the parts is perceived as protruding from the side of the other part. The medial axis of a protruding part and a host part should connect end-to-side, but that connection cannot be represented in a skeleton, since all branch junctions are end-to-
end connections between medial axis segments. We deal with this limitation by merging the branches labeled as *host*, and augmenting the skeletal information with a list of the resulting end-to-side branch connections.

Our algorithm for detecting protrusions is based on the ligature properties of the medial axis. It differs from previous ligature-based approaches (see Sec. 2.1) in that the complex cases of nested ligature are untangled by following simple rules and a clear processing order, which, in turn, is based on a branch merging operation that respects medial axis properties (allowing for the recursive processing of branches). In addition, we propose a novel method for detecting ligature points that accounts for the presence of both sharp and smooth concave corners on the shape boundary.

Finally, we recover shape parts and part attachment relations from the medial axis by partitioning the unprotruded branches into segments of ligature and non-ligature points. The segments of non-ligature points correspond to the medial axes of shape parts, while the junctions, ligature points and the list of protrusions provide the attachment relations between such parts. In the next chapter, we use this information to construct a hierarchical representation of the part structure of a given shape.

We leave as future work the evaluation of alternate rules for detecting protrusions. For example, it would be interesting to study whether a decomposition into parts with no apparent perceptual quality can, in fact, lead to better recognition performance. In addition, it would be fruitful to examine the sensitivity of recognition performance as a function of varying the protrusion detection threshold.
Chapter 4

Bone Graphs: Medial Abstraction for Object Recognition

4.1 Introduction

The shape parsing approach presented in the previous chapter yields two types of skeletal parts and two types of adjacency relations between them (Fig. 4.1 (a)). The skeletal parts are called bones and ligaments. The relations between these parts are given by the way in which they are attached, which can be either end-to-end (e.g., parts 2 and 5 in Fig. 4.1) or end-to-side (e.g., parts 5 and 8 in Fig. 4.1). The attributes of both types of relations encode the points on each of the parts defining the attachment, and, in the case of the end-to-side relation, also the side of the (host) medial axis associated with the attachment. The information recovered by the parsing process can be extremely useful for comparing shapes and finding part correspondences, but it needs to be represented in a way that helps solve the shape matching problem. In this chapter, we seek an abstraction of this information that makes explicit the salient parts of a shape and yields a stable encoding of their attachment relations.

Parts and relations can be represented naturally by an attributed graph. Three classes of attributed graphs are chiefly considered in the shape literature: rooted trees, directed acyclic graphs (DAG), and undirected graphs. Rooted trees and DAGs represent shapes as hierarchical structures defined with respect to the saliency or the scale of the parts [68, 89, 77, 25]. A part hierarchy is a powerful tool for simplifying the shape matching problem.

1These are the only two attachment types produced by our implementation of the parsing algorithm. In general, it is also possible to have side-to-side attachment, which is discussed in Appendix B.
Figure 4.1: Example output of the shape parsing algorithm and its bone graph representation. (a) The shape parsing algorithm presented in Chapter 3 yields a skeleton partitioned into bones (gray points) and ligaments (red and blue points), and the adjacency relations between them. (b) This output can be better appreciated by encoding it as a mixed graph in which: (1) each bone and ligament maps to a node, (2) the edges of the graph encode the binary adjacency relations, (3) special relational nodes encode adjacency relations of higher order (e.g., \( j_1 \) in the figure), and (4) the edge directions encode whether a relation is end-to-end (EE) or end-to-side (ES). The EE is an undirected relation between two or more skeletal parts connected by their terminal points, while the ES is a directed relation between exactly two skeletal parts in which the terminal point of one part is considered to be connected to the side of the other part. (c) In order to simplify the problem of comparing the bone and ligament parsing of a shape, we propose to abstract out the non-salient skeletal parts (the ligaments) and to assemble the salient parts (the bones) into a hierarchical structure, called a bone graph. In this graph, the node attributes encode the geometrical properties of each shape part, while the edge attributes encode the relational properties of each part attachment.
problem, as it provides global node dependencies that become meaningful constraints at matching time. On the other hand, undirected graphs, such as the ARG representation [30], are limited to providing local part-attachment constraints, as they only encode node adjacency information. The lack of global constraints leads to a computationally expensive matching approach, but can be advantageous if a part hierarchy cannot be constructed reliably. The related work in this area is discussed in Chapter 2.

In this chapter, we propose a novel graph-based shape abstraction, called a bone graph, which assembles the skeletal parts recovered by our shape parsing algorithm into a hierarchical structure (Fig. 4.1 (c)). The bone graph is a parts-based abstraction of a shape whose boundary is a simple closed curve, and is encoded as a DAG in which the edges represent hierarchical relations between the salient parts of a skeleton (the bones). The rules governing the edge directions are inspired by those of the shock graph grammar [89], but offer significant advantages over them. In particular, the edges of the bone graph abstract out the non-salient parts of a skeleton (the ligaments) and its branching topology, which can be quite complex (Fig. 4.2). This allows the bone graph to be less sensitive to perturbations to a silhouette caused by viewpoint changes than the shock graph. In Section 4.3, we evaluate the new representation by comparing it to the shock graph in a set of view-based 3-D object recognition and pose estimation trials.

### 4.2 Bone Graph Construction

In the construction of the bone graph, we seek an encoding of the attachment types and the abstraction of the uninformative complexity of ligament-to-ligament attachments (e.g., edge (3,2) in Figure 4.1 (b)). Furthermore, we seek to represent the hierarchical relations between the salient parts (the bones) of a skeleton by letting the bones map to graph nodes, and the ligaments and junction points map to graph edges. The edges of the graph represent bone-to-bone attachments, which are recovered from the adjacency relations between bones and ligaments given by the shape parsing algorithm. Recovering bone-to-bone attachments requires the transformation of end-to-end (EE) and end-to-side (ES) attachments between ligaments into binary relations between the bones connected to them.

The ligament-to-ligament attachments correspond to tertiary or higher order relations.

\(^2^2\)In our experiments, we ensure that a shape’s boundary is a closed curve by “filling” every hole within the shape.
Figure 4.2: Example of a bone graph. (a) The ligament-to-ligament attachments, such as that formed by ligaments $\ell_1$ and $\ell_2$, are expressed in (b) as edges between the parent node 1 and the two child nodes 7 and 8. The shape areas associated with each bone are colored differently for each level of the graph, and drawn following the edge directions in bottom-up order. The edge attributes encode the attachment position of a child bone along its parent bone. The sign of the position specifies the side of the attachment on the parent bone. For display purposes, an edge is colored black if it encodes a position with a positive sign and red if it encodes a position with a negative sign.

between bones. We transform them into binary bone relations by selecting one of the bones as the parent of the others. To this end, we define the parent bone as the bone with the skeletal point that is closest to the ligature point with largest radius (Fig. 4.2). The attributes of each parent-child edge of a bone graph are given by the skeletal points of the ligature segment that is immediately adjacent to each child bone. As a result, every ligature point is uniquely associated with one edge, and every edge connects two bones.

Like the shock graph, we direct the edges of the bone graph according to a local estimate of relative part size. A hierarchical relation between bones can be associated with the flow direction of the ligaments between them. However, in the case of a “neck” shape (Fig. 4.3), the ligament between the bones is formed by two ligature segments
that flow away from the ligature point closer to the boundary. One solution to this hierarchical-order uncertainty is to represent necks as undirected edges, which leads to a mixed graph. Another solution is to treat the neck’s ligament point(s) with the smallest radius as a bone. This solution is similar to the type 2 node of a shock graph [89]. We take the latter approach, as it allows us to simplify the matching problem by focusing on directed graphs. Then, let $\mathcal{E}$ be the set of directed edges. An ordered bone relation $(i, j) \in \mathcal{E}$, directed from bone $i$ to bone $j$, reflects one of the following conditions:

- bones $i$ and $j$ are incident to a junction point, and the radius function of bone $i$ is constant or increases away from the junction. This is the case where a larger bone branches out to form a series of smaller bones (e.g., see junction $j_1$ in Fig. 4.1 (a)).

- bones $i$ and $j$ share a junction point at which their respective radius functions are local maxima. In this case, bone $i$ in fact has the junction point as its only skeletal point.

- bones $i$ and $j$ are connected by ligaments whose radius function decreases monotonically from $i$ to $j$. This is the case where there is an end-to-side attachment between bones $j$ and $i$, or where the bones are connected end-to-end by one single ligature segment or by nested ligature.

We let the edge attributes encode the position, $p_{i,j}$, along the parent bone $i$ of the point closest to the nearest end of the child bone $j$. For convenience, we normalize the length of each bone’s medial curve to the interval $[0, 1]$, with the “0” end chosen arbitrarily for root bones and bones with two parents. For bones with a single parent, the “0” end is chosen to be the endpoint closest to the parent bone. For ES attachments, and assuming a clockwise traversal of the branch from the “0” end, we specify attachments on the left side as positive values in the open interval $(0, 1)$ (a value of 0 or 1 would imply an EE attachment), and attachments on the right side as negative values in the open interval $(0, -1)$.\(^3\) Such an attachment specification allows us to qualitatively distinguish whether attachments are near one of the ends or the middle of a bone, whether multiple attachments are on the same or opposite sides of a bone, or whether the attachments on the same side of a bone are near or far apart. The ability to facilitate such qualitative attachment judgments is inspired by Biederman’s RBC theory [9].

\(^3\)In fact, there are two possible attachment specifications, depending on the choice of endpoint, and both have to be considered if the signs of attachment positions are used in matching or other tasks.
Figure 4.3: The representation of neck shapes in the bone graph. (a) Example of a neck shape that induces a ligament connecting two bones. The radii of the ligament points decrease, from each endpoint, toward the ligature point closest to the concave corners. In this case, there is no clear hierarchical relation between bones 1 and 2. In general, a ligament with a radius function that is not monotonically varying represents an attachment with no clear direction. A natural graph representation of such attachments is as undirected edges, as shown in (b). However, this leads to a graph in which some edges are directed and others are not, which can be more difficult to match. As a solution, we relabel the ligament point(s) with the smallest radius as bone point(s). This results in two directed attachments between three bones (c). Finally, it is important to note that, as discussed in Section 3.6, not all neck shapes induce ligaments between bones. (d) An example of a neck shape represented by a single bone. In such cases, the neck shape is encoded by the attributes of a bone, which are evaluated during matching.
We can now specify the bone graph:

**Terminology 4.1.** The Bone Graph of a 2-D shape bounded by a simple closed curve, $\text{BG}(\Omega)$, is an attributed directed acyclic graph $G = (V, E, \lambda, \gamma)$ with

- **nodes** $V = \{1, \ldots, n\}$, representing the bones obtained from parsing the shape using the algorithm presented in Chapter 3;

- **edges** $(i, j) \in E \subseteq V \times V$ directed from node $i$ to node $j$ iff $(i, j) \in E$;

- **node attributes** $\lambda : V \mapsto S$, where $S$ is the set of all bone points, and $\lambda(i) = L_i$ is the set of bone points represented by node $i \in V$, for $L_i \subseteq S$; and

- **edge attributes** $\gamma : E \mapsto [-1, 1]$, with $\gamma(i, j) = p_{i,j}$ encoding the attachment position of bone $j$ onto bone $i$, for $(i, j) \in E$.

The node attributes encode the position, radius, tangent, and object angle of the skeletal points represented by them (see Sec. 3.2). This information can be used at matching time to compare the geometrical properties of two shape parts encoded by two nodes.

### 4.3 Experiments

We evaluate the bone graph representation by comparing it against the shock graph in a set of view-based object recognition experiments. We provide a meaningful comparison by evaluating both types of graphs under the same graph matching framework and by using the same node similarity function. We follow the matching framework of [89], and construct a node similarity function for bone graphs by partitioning each bone into shock parts. While this matching framework ignores the edge attributes of the bone graph, it does allow us to directly compare the stability of these two medial descriptions by ensuring that nodes and edges are interpreted identically.

We begin with a dataset of 1664 silhouette views of 13 3-D models (Fig. 4.5) with 128 uniformly spaced views per object around its viewsphere (Fig. 4.4 (a)), and populate a database of shock graphs and a database of bone graphs. Each view is successively removed and compared to the remaining views. If the 3-D model from which the closest matching view was generated is the same as that of the query, then recognition (identification) is said to be successful. If recognition is successful and the best matching view is
Figure 4.4: The structure of the viewsphere. (a) Configuration of the 9 closest neighbors of the query view (center) on the viewsphere; (b) one of the query’s neighbors, as seen on the 3-D viewsphere.

Figure 4.5: The 13 models used in the experiments.
one of the nine closest neighbors (Fig. 4.4 (b)) of the removed view, then pose estimation is said to be successful.

In the next set of trials, each of the 1664 views is again used as a query. However, the database of views is subsampled by randomly removing 25% of the views, leading to subsampled databases of shock graphs and bone graphs. The same experiment is repeated (the query view, if present, is removed from the model database), measuring correct recognition rates for shock graphs and bone graphs. This subsampling/evaluation process is repeated down to, and including, databases containing only 32 views of each object (75% model view removal). At each iteration, we compute three separate random viewsphere subsamplings and aggregate the results. In this fashion, 16,640 recognition trials are conducted in total.

Figure 4.6 plots both the recognition and pose estimation success rates for both shock graphs and bone graphs as a function of decreasing viewsphere sampling resolution. For the recognition task, the improved stability of the bone graph over the shock graph is clearly visible. The results show an improvement of approximately 3% with no model views removed, and this improvement increases steadily to approximately 7% with 75% of the model views removed. The pose estimation results, reflecting a far more stringent recognition task, show a dramatic (13%) improvement in stability over the shock graph at all sampling resolutions. We remind the reader that these experiments do not exploit the full power of the bone graph in that the relative locations of attachments (edge attributes) are ignored so as to put the bone graph on the same footing as the shock graph for each trial. Exploiting such constraints in the matcher should lead to further improvement in the results.

Figure 4.7 illustrates a number of successful matches drawn from the experiment. In each pair, the shape on top represents the query while the shape underneath represents the closest matching database view. For both shapes, the recovered bones are displayed (shaded) over the restored skeletons, with the final ligature/non-ligature analysis reflected in the coloring of the skeletons. In addition, corresponding bones between query and model, as computed by the matcher, are colored the same. These examples illustrate the fact that while viewpoint changes may induce significant structural variation in the skeleton, due to skeleton over- and under-segmentation, the final bone decomposition is less sensitive to viewpoint changes than the shock graph. Whereas the shock graph is forced to explicitly encode this structural instability, the bone graph captures the salient shape at a higher and more stable level of abstraction.
Figure 4.6: Correct recognition (dashed curves) and pose estimation (solid curves) rates for bone graphs (blue, labeled BG) and shock graphs (black, labeled SG) as a function of decreasing viewsphere sampling resolution. The bone graph clearly exhibits superior stability for both tasks, with dramatic improvement for the more stringent pose estimation task.

It is also interesting to analyze examples of unsuccessful matches. Figure 4.8 illustrates two incorrect recognition results in which the absence of edge attributes and the weakness of the node similarity function do not penalize sufficiently the salient differences between the shapes. Here, the views of the horse and dog are in fact similar, but we would like other horse views to rank before any view of a different object. In the case of the dinosaur, the query is a view from the top of the viewsphere and is missing many of the parts present in other views of the object. This makes the matching algorithm depend more strongly on the geometrical differences between parts, which in this case, leads to an incorrect match. A different node distance function might help correct this type of error.

4.4 Conclusions

We introduce a novel shape abstraction based on the skeleton, where the goal is to map skeletal segments to intuitive shape parts. We do so by assembling the skeletal parts
Figure 4.7: Matching Bone Graphs. In each pair of shapes, the top shape represents the query while the bottom shape represents the closest matching database view. Each shape includes its final restored skeleton, along with the shaded bones defined by the non-ligature segments. Corresponding bones between query and model, as computed by the matcher (which ignores part order), are colored the same. Close examination reveals that while skeleton topology (encoded explicitly in a shock graph) may change significantly due to changes in viewpoint, bone graph topology is far more stable.

recovered by our shape parsing algorithm into a hierarchical abstraction of a shape’s structure. The result is the bone graph, a powerful parts-based shape abstraction whose stability is demonstrably better than the shock graph for the task of view-based object recognition. In addition, the bone graph, unlike the shock graph, not only captures parts and their adjacency, but a set of attributed attachment relations between the parts. In the next chapters, we develop a matching framework that exploits the edge attributes of the bone graph (reflecting “where” parts are attached) and explore the bone graph as a tool for object categorization.
Figure 4.8: Incorrect Matching Examples for Bone Graphs. Here we show two cases in which the most similar model views (bottom row) to the query views (top row) do not belong to the same object. **LEFT:** since edge attributes are ignored, the dog’s tail is assigned to one of the horse’s leg without a penalty for the differences in their relative position with respect to each respective torso. **RIGHT:** the query is a view from the top of the dinosaur’s viewsphere in which some shape parts are either occluded or their projections are significantly deformed with respect to other views of the object. In this case, the node similarity function employed by the matching algorithm fails to penalize for the geometrical differences between the matched parts, and leads to a dolphin’s view with the highest similarity score.
Chapter 5

Inexact Graph Matching for Bone Graphs

5.1 Introduction

In this chapter, we explore the problem of matching graph-based representations in which both the nodes and edges of the graphs have arbitrary sets of attributes. In the previous chapter, we performed graph matching experiments with bone graphs and shock graphs using the matching algorithm for node-attributed (NA) DAGs proposed by Shokoufandeh et al. [83]. Since this algorithm shows promising results when applied to bone graphs, we propose a generalization of it for the problem of matching fully-attributed (FA) DAGs. In addition, our generalization of the algorithm expands the type of domain knowledge that can be incorporated into the matching process, and allows us to improve on the results obtained with the previous version of the algorithm.

The problem of finding node correspondences between graphs that might not be isomorphic is known as inexact graph matching. In our case, we are also interested in computing a measurement of graph similarity that can be used to rank order model graphs with respect to a query graph. Given two graphs $G(V, E)$ and $G'(V', E')$, our problem is to find the values of node assignment variables $a_{vv'} \in \{0, 1\}$, for $v \in V$ and $v' \in V'$, that maximize some function of graph similarity, $F(G, G')$. In the case of attributed graphs, the graph similarity measure is a function of the attribute similarity of nodes and/or edges and the structural similarity of the underlying graphs. For example, in [62, 52, 88]
Figure 5.1: Natural Violations to Node Adjacency Relations. We show the bone graph representations of two different horse exemplars (only the skeletons’ bones are drawn). In this example, the enforcement of node adjacencies would exclude a node correspondence solution in which the torsos and front legs of the horses are assigned to one another. This is because the torso of the left horse (node 1) is not adjacent to the front legs (nodes 6 and 3) due to the presence of a shape part between them (node 2), but the torso of the right horse is adjacent to the horse’s front legs.

the problem is stated as

$$F(G, G') = \max_{M \in \mathcal{M}} \frac{1}{2} \sum_{v \in V} \sum_{v' \in V'} M(v, v') N_a(v, v'), \quad (5.1)$$

where $N_a(v, v')$ are constant weights representing measures of node attribute similarity, and $\mathcal{M} = \{M\}$ is the set of all $|V| \times |V'|$ binary matrices $M(v, v') = a_{vv'}$ whose nonzero entries represent node assignments that satisfy some set of constraints. Frequently, the constraints enforce one-to-one node mappings that preserve node adjacency, and therefore define a maximum common isomorphic subgraph problem, which, for general graphs, is NP-hard [34].

In some domains, requiring the preservation of node adjacency may exclude natural solutions to the node correspondence problem. For example, Figure 5.1 shows that a minor shape difference can make a mapping between salient bone graph nodes invalid if node adjacency constraints are enforced. As a workaround, inexact graph matching can be defined in terms of edit operations that add or eliminate nodes with the objective of finding a graph isomorphism with minimum edit cost [15]. In this case, the similarity
weight between nodes $v$ and $v'$ is not simply a function of their attributes (like $N_a(v, v')$ in Eq. 5.1), but also accounts for the costs of the edit operations needed to make $v$ and $v'$ respect their adjacency to any previously matched nodes. The minimization of edit costs is usually posed as tree search with backtracking, in which a solution set (initially empty) is iteratively grown by adding node correspondences that are compatible with the mappings already in the set. We discuss the related work in this area in Chapter 2.

The tree search approach to graph matching provides a simple mechanism for measuring the similarity of attributed graphs, but is limited to exploiting only node adjacency constraints. Our algorithm for graph matching is similar to the tree search approach in that we: (a) iteratively grow a solution set of node correspondences, (b) use heuristics to find an approximate solution in polynomial time, and (c) define the similarity between two nodes as a function of both their attributes and the set of previously selected correspondences. However, unlike tree search, we consider constraints beyond adjacency in the evaluation of node similarity. This allows us to account for edge attribute similarities and for the hierarchical dependencies between the nodes of a DAG.

Our matching algorithm seeks to approximate the result of evaluating all possible ways of populating a solution set of node assignments, and selecting the set with the maximum sum of assignment weights. One simple way of expressing this function is to assume that every node in $V$ is mapped to every node in $V'$ with some weight, which is nonzero only if the nodes are assigned to each other. Then, if we order the set of $|V| \times |V'|$ correspondences $\{(v, v')\}_{v \in V, v' \in V'}$ as a list, and let $\mathcal{L}$ be the set of all possible permutations of that list, the graph similarity function can be expressed as

$$F(G, G') = \max_{A \in \mathcal{L}} \frac{1}{N} \sum_{v \in V} \sum_{v' \in V'} w(v, v', A_{vv'})$$  \hspace{1cm} (5.2)$$

where $A_{vv'}$ is the set of node correspondences that appear before $(v, v')$ in the totally ordered set $A \in \mathcal{L}$, and $w(v, v', A_{vv'}) \in [0, 1]$ are node similarity weights, which are nonzero if and only if $v$ is assigned to $v'$. $N$ is a normalization constant that penalizes for unmatched nodes (i.e., nodes of a graph whose similarity weights are zero for every node of the other graph), and is needed for rank ordering matching results. We let $N = \max(|V|, |V'|)$ for bone graph and shock graph matching, which is an appropriate normalization constant when the nonzero solutions are constrained to be one-to-one node mappings.

The dependency of node correspondence weights on $A_{vv'}$ allows for measuring node similarity with respect to structural and attribute constraints given by the node assign-
ments already in the solution. Generally, these constraints render some permutations redundant (e.g., they yield node mappings with zero weights), which can be exploited by the matching algorithm. In Section 5.3, we propose a novel definition of similarity weights that leads to a general approach for graph matching, which includes the approach of Shokoufandeh et al. [83] as a special case. Our approach allows us to naturally incorporate domain knowledge that cannot be expressed when the node correspondence weights are treated as constants. We present an efficient algorithm that exploits the structure of the problem to recompute only a small set of similarity weights at each iteration of the algorithm. Furthermore, in Section 5.3.3, we describe a straightforward extension to our algorithm in order to explore a bounded number of matching solutions. Later, in Chapter 6, we evaluate empirically the benefit of considering multiple solutions for the case of bone graphs and shock graphs.

5.2 Notation

We assume that we are given a pair of fully-attributed DAGs of the form $G(V, E, \lambda, \gamma)$, where $V$ is a set of nodes, $E$ is a set of edges $e = (u, v)$ directed from node $u \in V$ to node $v \in V$, $\lambda(v)$ is a function that maps each node $v \in V$ to a domain-dependent set of node attributes, and $\gamma(e)$ is a function that maps each edge $e \in E$ to a domain-dependent set of edge attributes. The given graphs are constant throughout the matching process, and so need not be passed as arguments to the functions that require access to their node adjacency matrices, or to the attributes of their node and edges.

We also assume that, in addition to the graphs, we are given a domain-dependent function of node attribute similarity $N_a(v, v')$, for $v \in V, v' \in V'$, and a domain-dependent function of edge attribute similarity $E_a(e, e')$, for $e \in E, e' \in E'$. All measures of similarity referred to throughout this chapter, whether they involve graphs, nodes, or edges, are assumed to be values in the interval $[0, 1]$. A similarity value equal to zero represents total dissimilarity, while a value of one represents equality.

The algorithms discussed below make frequent references to the problem of maximum-weight bipartite matching (MWBM) [33]. Then, it is convenient to define special notation for this problem. To this end, let $M = MWBM(A, B, W)$ be the solution to the MWBM problem, where $A$ and $B$ are disjoint sets of elements, and $W$ is an $|A| \times |B|$ matrix of similarity weights between each pair of elements spanning $A$ and $B$. The solution to this problem is the set $M = \{(a, b)\}$ of one-to-one correspondences with nonzero weights.
between the elements of $A$ and $B$ that yields the maximum sum of similarity weights. In the case of ties, we assume that $M$ is any set with maximum cardinality among all possible sets with maximum weight.

In some cases we are only interested in the value of the weight sum rather than the set of correspondences, and so we define an appropriate function:

$$\bar{M}(A, B, W) = \sum_{(a,b) \in MWBM(A, B, W)} W(a, b),$$

where $W(a, b)$ is the weight associated with matching element $a \in A$ and $b \in B$. Finally, it should be assumed that whenever we evaluate $\bar{M}$, the set of correspondences $M$ can also be recovered from the solution if needed.

## 5.3 The FA-DAG Matcher

We begin by reviewing the matching algorithm of Shokoufandeh et al. [83] for node-attributed DAGs (NA-DAG), and then motivate our generalization of this work to the problem of matching fully-attributed DAGs (FA-DAG) and the exploitation of a wider range of domain knowledge. The NA-DAG matcher measures the similarity between two DAGs by searching for the node correspondences that maximize the sum of pairwise node similarities, while attempting to respect the hierarchy imposed by the edge directions in each graph. In each iteration of the algorithm, one node correspondence is added to the solution set in a greedy fashion. The selection of each correspondence is given by first solving the $M = MWBM(V, V', W)$ problem, where the weights for each possible correspondence $(v, v')$ are a function of the attribute similarities of the nodes and a measure of the structural similarity of the subgraph rooted at each node. Then, the node correspondence $(v, v') \in M$ with the largest weight is added to the solution set (initially the empty set). Next, the selected node correspondence is used to split the graphs into two subgraphs formed by the descendants of $v$ and $v'$, and two subgraphs formed by their respective non-descendants. Finally, the algorithm proceeds recursively by matching the pairs of descendant and non-descendant subgraphs independently until there are no more possible correspondences left to select. This algorithm is depicted in Figure 5.2.

The measure of structural similarity encoded in the correspondence weights is a low-dimensional spectral signature computed from the adjacency matrix of the subgraph rooted at each node. This measure is an attempt to let the matcher account for the
Figure 5.2: The NA-DAG matcher. (a) Given a pair of directed acyclic graphs $G$ and $G'$, (b) form a bipartite graph in which the edge weights are the pairwise node similarities. Then, (c) compute a maximum weight matching and add the best edge to the solution set. Finally, (d) split the graphs at the matched nodes and (e) recursively descend.
structure underneath nodes before committing to a node correspondence. In practice, the signature vectors can be seen as part of the node attributes, since they are constant and can be precomputed before the matching process takes place. We take this view, and assume that the same technique can be used in our generalization of the algorithm, if desired.

The pairwise node correspondence weights in each iteration of the algorithm remain constant and correspond to a possibly suboptimal solution to Equation 5.1 \[84\]. In \[83\], variable weights are considered to overcome the problem that whenever the graphs are split and each pair of subgraphs is matched, the subgraphs of non-descendants can lead to correspondences that violate the hierarchical constraints imposed by the DAGs. For example, a pair of sibling nodes in one graph can be assigned to a pair of parent-child nodes in the other graph. To prevent this, some of the weights in \( W \) are penalized in each iteration of the algorithm. Unfortunately, the update of weights can lead to a solution that is not consistent with Equation 5.1, which assumes that the correspondence weights are constant.

We generalize the NA-DAG matching algorithm to fully-attributed DAGs by considering the objective function given by Equation 5.2, and defining correspondence weights \( w(v, v', A_t) \) that are a function of the node and edge attributes of nodes \( v \) and \( v' \), and the solution set, \( A_t \), at each iteration, \( t \), of the algorithm. This definition of correspondence weights also allows us to encode the graph split operation of the NA-DAG matcher as a structural constraint of the problem. In fact, by encoding structural constraints as weights, we provide a more general approach for incorporating domain knowledge into the matching problem. The weights can be used to penalize correspondences that violate different types of node relations, whereas the graph split operation of the NA-DAG matcher is restricted to a single type of node relation (i.e., descendant), and to a binary decision on whether or not the node correspondences satisfy the relation.

We decompose \( w(v, v', A_t) \) into four factors, which measure similarity as a function of

- **Graph structure**, such that the violation of a hierarchical relation with respect to nodes in \( A_t \) is penalized;

- **Edge attributes**, where the attribute similarity of all edges incident on nodes \( v \) and \( v' \) is evaluated. This similarity is computed while accounting for the paths that connect nodes in \( A_t \) to both nodes \( v \) and \( v' \);
assignment uniqueness, such that correspondences involving nodes already in \( A_t \) are assigned a zero weight;

node attributes, where the similarity of node attributes is measured using a domain-dependent function.

Our objective is to let any of these factors veto a node assignment by yielding a similarity value equal to zero. This can be expressed as a product over all the factors. Then, we let the similarity weights be

\[
w(v, v', A_t) = S(v, v', A_t)E(v, v', A_t)U(v, v', A_t)N_a(v, v', A_t),
\]

where \( S, E, \) and \( U \) are the structural, edge, and uniqueness similarity functions, and \( N_a \) is a domain-dependent similarity function for node attributes. The structural and edge similarity functions deserve special consideration and are defined in Sections 5.3.1 and 5.3.2, respectively. The uniqueness similarity is simple, and can be defined as

\[
U(v, v', A_t) = \begin{cases} 
1 & \text{if } \forall u' \in V'(v, u') \notin A_t \text{ and } \forall u \in V(u, v') \notin A_t, \\
0 & \text{otherwise}. \end{cases}
\]

The matching algorithm is simply given by the successive iteration of two main steps. In the first step, we create/update a complete bipartite graph with node sets \( V \) and \( V' \), and edge weights \( W_t \) computed as a function of the current solution set \( A_t \) (initially the empty set). In the second step, we solve the MWBM problem, \( M = \text{MWBM}(V, V', W_t) \), and add the correspondence \((v, v')^* \in M\) with largest weight to the solution set, such that \( A_{t+1} = A_t \cup \{(v, v')^*\} \). The algorithm terminates when there are no more node correspondences to consider, i.e., \( W_t = 0 \). In Section 5.3.3, we present this algorithm in more detail and also present a less greedy variation of it.

An example of the first three iterations of the algorithm is shown in Figure 5.3. In this example, we assume that the structural similarity \( S(v, v', A_t) \) is equal to zero if the union of \( \{(v, v')\} \) and \( A_t \) is a set in which an ancestor-descendant assignment is inverted, i.e., a set in which a pair of ancestor-descendant nodes in one graph is assigned to a pair of ancestor-descendant nodes in the other graph with the ancestors assigned to the descendants.
Figure 5.3: The generalized DAG matching algorithm. (a) The given pair of DAGs. (b) In the first iteration of the algorithm, we form a bipartite graph with similarity weights computed as a function of the empty solution set $A_0$. Then, we solve the MWBM problem and add the correspondence with largest weight, in this case $(3, 3')$, to the solution set. (c) In the next iteration, we recompute the similarity weights as a function of the augmented solution set $A_1$. Here, all correspondences of the form $(3, v')$ and $(v, 3)$ have a zero weight due to the $U(v, v', A_t)$ term in Equation 5.3. Similarly, the $S(v, v', A_t)$ term evaluates to zero for the correspondences that violate structural constraints. For example, the correspondence $(1, 4')$, if added to $A_1$, would have an ancestor of node 3 assigned to a descendant of node $3'$, and so $S(1, 4', A_1) = 0$. Only nonzero correspondences need be considered when solving the MWBM problem. (d) A possible third iteration of the algorithm. The algorithm terminates when all the possible correspondences between unassigned nodes have zero weights.

### 5.3.1 Measuring Structural Similarity

The structural similarity of a node assignment $(v, v')$ given a set of previous assignments $A_t$ can be thought of as a measure of how much the node relationships in each graph
Figure 5.4: Example of hierarchical similarity. Here we assume that the node correspondence \((x, x')\) is in the current solution set \(A_t\), and want to compute the structural similarity for the node correspondence \((v, v')\). Since \(x\) is an ancestor of \(v\), but \(x'\) is a descendant of \(v'\), the addition of \((v, v')\) to the solution would create a node mapping that does not respect the hierarchical ordering between the nodes of each DAG. A structural similarity equal to zero would prevent this assignment from being added to the solution.

would differ if node \(v\) is assigned to node \(v'\). For example, Figure 5.4 shows an example in which an assignment between an ancestor of node \(v\) and a descendant of node \(v'\) has been previously established. In this case, if node \(v\) is assigned to node \(v'\), the resulting set of node assignments would not respect the hierarchical ordering imposed by the DAGs.

There are three types of hierarchical relations in which we are interested for bone graphs: ancestor, descendant, and sibling. However, we can describe our approach more generally by assuming a set of node relations \(R = \{r_k\}_{k=1}^K\) that we are interested in maintaining, and defining predicates \(r_k(a, b)\) that are true iff node \(a\) and \(b\) satisfy relation \(r_k\). Briefly, our goal is simply to ensure that both nodes \(v\) and \(v'\) have similar relations with the previously matched nodes \((u, u') \in A_t\). Whenever this is not the case, i.e., \(r_k(u, v) \neq r_k(u', v')\) for any \(k = 1, \ldots, K\), we penalize the assignment \((v, v')\) according to a domain-dependent penalty \(p_k \in [0, 1]\) for the relation \(k\). This can be expressed as a product over all relations of interest between nodes \(v\) and \(v'\) and the nodes in the current solution set. Then, we define the structural similarity function as

\[
S(v, v', A_t) = \prod_{(u, u') \in A_t} \prod_{r_k \in R} [p_k^{r_k(u, v) \neq r_k(u', v')}],
\]

(5.5)

where \([\mathcal{P}]\) is the square bracket notation, which is equal to one if the predicate \(\mathcal{P}\) is true, and zero otherwise.

In our experiments, we specify the penalties for the ancestor, descendant, and sibling
Figure 5.5: Example of the ancestor relation. Here we assume that the node correspondence \((x, x')\) is in the current solution, and want to compute the structural similarity for the candidate correspondence \((v, v')\). Since \(v'\) is an ancestor of \(x'\), but \(v\) is not an ancestor of \(x\), the addition of \((v, v')\) to the solution would create a node mapping that does not respect the ancestor relation. In this case, we apply a penalty to the weight of the candidate correspondence \((v, v')\) to make it less likely to be added to the solution. In contrast, the NA-DAG matcher does not penalize for the violation of this node relation.

relations as follows. For the ancestor and descendant relations, we let their penalties be \(p_1 = 0, p_2 = 0\), so that violations of such relations are not allowed in the solution. This differs from the NA-DAG matching approach of Shokoufandeh et al. [83], which does not penalize for violations to the ancestor relation (see Figure 5.5). For the sibling relation (see [83] for a discussion on this relation), we let the penalty be \(p_3 = 0.8\), which encodes the fact that shape parts frequently switch parents when the shapes are represented by bone graphs or shock graphs.

In summary, our structural similarity function is a general approach to incorporating assumptions about the importance of preserving certain node relations when matching two graphs. Our formulation of the problem allows us to specify arbitrary types of node relations, and to penalize mismatches according to the importance of each relation.

5.3.2 Measuring Edge Similarity

The edge similarity of a node assignment \((v, v')\) given a set of previous assignments \(A_t\) is the normalized sum of pairwise edge attribute similarities for the inward and outward edges incident on nodes \(v\) and \(v'\). We compute this value by assuming that there is a one-to-one correspondence between inward edges and between outward edges, and find the set of edge correspondences that maximizes the sum of pairwise similarities. We
Figure 5.6: The edge similarity function. (a) We evaluate $E(3, 3', A_t)$ by summing the pairwise similarities of edges incident on nodes 3 and 3' as a function of the solution set $A_t$. In this example, we assume that the node correspondence (6, 7') is the only element in $A_t$. $E(3, 3', A_t)$ is equal to the maximal sum of nonzero one-to-one correspondences between edges. A correspondence $(e_i, e_j)$ may have a nonzero weight $W(i, j)$ iff: (1) both edges have equal direction, and (2) they are consistent with the contents of $A_t$. For example, here the only possibly nonzero correspondence for either edge $e_3$ or $e_4'$ is $(e_3, e_4')$, since there is a directed path from nodes 3 to 6 and from nodes 3' to 7'. In contrast, edges $e_4$, $e_2'$, and $e_3'$ have no ancestors or descendants in $A_t$, and have more than one possible correspondence. (b) $E(3, 3', A_t)$ is equal to the solution of the MWBM problem shown here, where the weights of the correspondences that meet conditions (1) and (2) are given by a domain-dependent measure of edge attribute similarity, $E_a(e, e')$.

account for the previous node matches $(u, u') \in A_t$ by ensuring that if both $u$ and $u'$ are connected to $v$ and $v'$ by a directed path, then the edges linking these paths to nodes $v$ and $v'$ are assigned to one another when we sum the pairwise edge similarities. An example of this constraint is illustrated in Figure 5.6.

We define the problem of measuring edge similarity as that of solving a MWBM problem for the set of edges incident on nodes $v$ and $v'$. For the special case in which both $v$ and $v'$ have empty sets of incident edges, we assume that their edge similarity is one. Otherwise, we represent the pairwise edge attribute similarity between edges, and the constraints on path information from previously matched nodes as the weights of the
Chapter 5. Inexact Graph Matching for Bone Graphs

MWBM problem. Thus, the edge similarity function becomes

\[ E(v, v', A_t) = \frac{\bar{M}(E_v, E_{v'}, W)}{\max(|E_v|, |E_{v'}|)}, \]  

(5.6)

where \( E_v = \{ e_k \} \) and \( E_{v'} = \{ e'_{l} \} \) are the sets of incident edges on nodes \( v \) and \( v' \), respectively, \( \bar{M} \) is the solution to the MWBM problem (Sec. 5.2), and \( W \) is the \(|E_v| \times |E_{v'}|\) matrix of edge correspondence weights defined below. The denominator of this equation normalizes the sum of edge correspondence weights to unity, yielding a similarity measure that penalizes for unmatched edges.

In order to define the edge correspondence weights, let \( P(a, b) \) be the (possibly empty) set of edges along the directed path that connects nodes \( a \) and \( b \) (i.e., a non-empty set means that \( a \) is either an ancestor or a descendant of \( b \)). Then, we let the edge correspondence weights \( W(k, l) \) be equal to zero if

- the edges \( e_k \) and \( e'_{l} \) have opposite direction with respect to nodes \( i \) and \( j \) (this condition is optional, as it may not be appropriate in some domains); or
- there exists a node correspondence \( (u, u') \in A_t \) such that the path \( P(u, v) \) includes \( e_k \) but the path \( P(u', v') \) does not include \( e'_{l} \), or conversely, that \( P(u', v') \) includes \( e'_{l} \) but \( P(u, v) \) does not include \( e_k \).

Otherwise, we let \( W(k, l) \) be equal to a domain-dependent function \( E_a(v, v') \) that measures the similarity of the edge attributes.

In conclusion, our edge similarity function incorporates edge attribute information in a nontrivial way by combining it with the information provided by the node correspondences in the solution. This measure of similarity exploits the structural dependencies between nodes to obtain an assignment of edge correspondences that is consistent with the node assignments that are in the solution set. As this set grows in each iteration of the algorithm, so too does the information available for assigning the edge correspondences. This, in turn, makes the measure of edge attribute similarity less ambiguous, and strengthens the overall measure of node similarity weights.

5.3.3 Searching the Solution Space

The matching algorithm of Shokoufandeh et al. [83] takes a greedy approach to establish node correspondence. This is motivated by the assumption that the measures of node attribute similarity should provide a strong indication of the best correspondences, and
make a broad search of the solution space less necessary. In our generalization of this algorithm, we propose to relax the greediness of the approach by incorporating a bounded queue to evaluate more than one possible solution to the problem. We form a queue of solution sets of size bounded by the maximum number of solutions that we want to evaluate. In each iteration of the algorithm, we add one node correspondence to a solution set in the queue. The algorithm terminates when the queue is empty.

We present the algorithm that considers a single solution first, and later introduce a queue to consider multiple solutions. The two main steps of the algorithm are the computation of node similarity and the selection of node correspondences. In the first iteration of the algorithm, $t = 0$, the weights of all possible pairwise node correspondences, $W_t$, are computed as a function of the empty solution set, $A_t = \emptyset$, according to Equation 5.3. Next, the MWBM problem, $M = MWBM(V, V', W_t)$, is solved and correspondence $(v, v')^* \in M$ with largest weight is added to the solution set, such that $A_{t+1} = A_t \cup \{(v, v')^*\}$. In the subsequent iterations, $t \geq 1$, the correspondence weights are updated to reflect the current contents of the solution set, and a new correspondence is added to the solution as previously described. The algorithm terminates when all possible new node correspondences have zero weight (i.e., they are incompatible).

When considering multiple solutions, we queue the alternate solution sets that can be obtained in the first few iterations of the algorithm. The motivation for this is that the first elements added to a solution set condition all subsequent correspondences, and so selecting the correct ones early on is important. We bound the maximum number of solution sets that can be considered by a given constant $K \geq 1$. We also let the maximum number of solution sets added to the queue in each iteration be a given constant $1 \leq \tau \leq K$. Then, we modify the matching algorithm such that at each iteration, we pop one solution set $A$ from the queue of active solution sets $Q$, and compute the correspondence weights as a function of it. If all correspondences have zero weight, we add $A$ to the set of completed solutions $S$. Otherwise, we solve a MWBM problem for the nodes of the graph using the current weights, and then select the top $n = \min(\tau, K - |Q| - |S| + 1)$ correspondences $(v, v')^*_k \in M$, for $k = 1, \ldots, n$. Each of these correspondences is used to create new solution sets $A^k = A \cup \{(v, v')^*_k\}$, which are added to the back to the queue. The algorithm terminates when the queue of active solution sets is empty. This algorithm is shown on the next page.
Algorithm 1 The FA-DAG matcher

Require: \( G(V, E, \gamma, \lambda), G'(V', E', \gamma', \lambda'), K \geq 1, \tau \leq K \) // for \( K, \tau \) defined in Sec. 5.3.3

1. \( Q \leftarrow \{\emptyset\} \) // let \( Q \) be a queue with an empty solution set in it
2. \( S \leftarrow \emptyset \) // let \( S \) be the empty set of completed solution sets
3. while \(|Q| \neq 0\) do
   4. \( A \leftarrow Q.Pop() \) // retrieve a solution set from the queue
   5. \( W \leftarrow f(G, G', A) \) // set node correspondence weights according to Eq. 5.3
   6. if \( W \neq 0 \) then
      7. \( M \leftarrow MWBM(V, V', W) \) // solve the MWBM problem (see Sec. 5.2)
      8. \( L \leftarrow Sort(M) \) // sort correspondences by decreasing weight
      9. \( n \leftarrow \text{Min}(\tau, K - |Q| - |S| + 1) \) // set max number of new solutions
     10. \( L \leftarrow \text{Prune}(L, n) \) // reduce the list to top \( n \) candidates
     11. for all \((v, v')\) in \( L\) do
        12. \( A' \leftarrow A \cup \{(v, v')\} \) // create a new solution set with \( A \) and \((v, v')\) in it
        13. \( Q.Push(A') \) // add the new solution set to the queue
     end for
   14. else
      15. \( S \leftarrow S \cup \{A\} \) // add \( A \) to the set of completed solution sets
   end if
17. end while
19. return \( \arg\max_{A \in S} \sum_{v \in V} \sum_{v' \in V'} w(v, v', A) \) // select the set with maximum weight sum
5.3.4 Algorithm Complexity

We focus our complexity analysis on the case in which the maximum number of solution sets is one, since a greater constant limit does not affect the algorithm’s complexity. We also assume that the algorithm is efficiently implemented such that only the weights that may change from one iteration to the next are updated. For the case of bone graphs, we update only the weights that involve ancestors, descendants, and siblings of the nodes \( v \) and \( v' \) that are added to the solution set in the previous iteration of the algorithm. In this case, we use appropriate data structures in order to efficiently visit the required weights, and to evaluate the structural and uniqueness similarity factor of the weight function. By recursively following the parents and children associated with each node correspondence added to the solution set, we can visit the weights that must be updated in time \( O(N) \), for \( N = \max(|V|, |V'|) \). The ancestor, descendant, and sibling relations can be evaluated in constant time by precomputing appropriate data structures for each graph. An efficient representation of the ancestor and descendant relations can be computed in linear time (they are given by the transitive closure of a graph [40]), while that of the sibling relation can be computed in quadratic time (it requires a combination of the transitive closure and the adjacency matrix of a graph [83]). We also use an appropriate data structure to evaluate the membership of a node to the solution set in order to compute the uniqueness similarity factor of the weight function in constant time.

The computation of the edge similarity factor demands more work, as it requires solving a MWBM problem. The complexity of this step is \( O(e^2 \log e) \), where \( e \) is the maximum number of incident edges for any node in either graph. Then, the complexity of updating the weights in each iteration is \( O(Ne^2 \log e) \). In addition to updating weights, each iteration of the algorithm must solve a MWBM problem for the nodes, which has complexity \( O(N^2 \log N) \). The number of iterations of the algorithm can be bounded by \( N \) if only one-to-one correspondences are allowed. Thus, the complexity of the algorithm is \( O(N^3 \log N + N^2e^2 \log e) \). In the case of hierarchical structures, such as bone graphs and shock graphs, the number of nodes grows much faster than the maximum number of incident edges on a node, and \( e^2 \) is generally smaller than \( N \) for DAGs of significant height, which results in \( O(N^3 \log N) \) complexity. Similarly, the complexity of the NA-DAG matcher algorithm is \( O(N^3 \log N) \) [83], since the edge similarity term need not be computed.
5.4 Similarity Functions for Bone Graphs

We present measures of similarity for the attributes of nodes and edges in the bone graph. These similarity measures are the domain-dependent components of the matching algorithms described in the previous sections. Our goal is to compare silhouettes obtained by the perspective projection of 3-D objects onto a plane. Moreover, since we aim to find similarities between the silhouettes of different exemplars of the same object class, we propose coarse measures of node and edge similarity that attempt to not overpenalize the within-class deformations of both shape parts and part relationships.

5.4.1 Node Attribute Similarity

We define the similarity $N_a(v, v')$ between the attributes of nodes $v$ and $v'$ as a function of the length and width of the shape part represented by each node. Our measure of similarity is inspired by that of shock graphs [57], which compares the radius functions of the skeletal segments encoded as the attributes of nodes. The radius function of a bone captures the variation of the width along a shape part and the length of the part. By comparing radius functions, we obtain a similarity measure that is invariant to relative rotation, translation, and bending. This measure is not scale invariant, and so we assume that the global scale of each shape is normalized. We obtain a continuous representation of the radius function by approximating its discrete values with a piecewise linear function. This approximation is the result of minimizing the number of line segments without surpassing a maximum fitting error (see [57] for details). Figure 5.7 illustrates examples of the radius functions of bone graph nodes, and their approximation by piecewise linear functions.

In the shock graph, the piecewise linear approximation of the radius function is used to create the shock graph nodes, and to compare their attributes at matching time. During the construction of a shock graph, this approximation is used to decompose the skeletal branches into segments with constant or monotonically varying radii, which then become nodes in the graph. At matching time, the approximation is used to compare the radius functions encoded by the nodes. For the bone graph, we only use the approximation of the radius function at matching time, since we do not partition skeletal segments according to radius. Thus, our problem is to compare radius functions that vary arbitrarily.

We propose a measure of similarity that accounts for the absolute radius differences and for the variations of the radii along the medial axes. We begin by subdividing the
radius functions into an equal number of segments for the part of the domain on which they are both defined. Given two piecewise linear functions, \( r(x) \) and \( r'(x) \), defined on the respective intervals \([0, L]\) and \([0, L']\), we subdivide the line segments that fall in the interval \([0, \min(L, L')]\) into \( N \) subsegments. The subdivision is performed such that the x-coordinates \( \{x_i\}_{i=0}^N \), for \( x_i < x_{i+1} \), correspond to the endpoints, intersection points, and knots between the segments of both functions (see Figure 5.8). Then, we compute the areas and slopes of the pair of linear functions defined on each interval \([x_i, x_{i+1}]\), and combine them into a piecewise measure of similarity.

First, we define the area-based similarity as a normalized measure of area difference. Let \( a_i \) and \( a'_i \) be the respective nonzero areas under \( r(x) \) and \( r'(x) \) on the interval \([x_i, x_{i+1}]\), and

\[
\alpha_i = 1 - \frac{|a_i - a'_i|}{a_i + a'_i} 
\]

be their area-based similarity. Next, we define the slope-based similarity as a Gaussian function of slope difference. Let \( m_i \) and \( m'_i \) be the respective slopes of \( r(x) \) and \( r'(x) \) along the interval \([x_i, x_{i+1}]\), and

\[
\beta_i = e^{-\frac{(m_i - m'_i)^2}{2\sigma^2}} 
\]

be their slope-based similarity (we let \( \sigma = 0.15 \) in our experiments). Finally, we combine the area- and slope-based similarities and the relative length of each interval into a measure of node attribute similarity

\[
N_a(v, v') = \sum_{i=0}^{N-1} \alpha_i \beta_i \frac{(x_{i+1} - x_i)}{\max(L, L')} 
\]
Figure 5.8: Example of the comparison of two radius functions. The piecewise linear functions are subdivided into $N$ segments for the part of the domain on which they are both defined. The segments are partitioned such that each $x$-coordinate $x_i$, for $i = 0, \ldots, N$, corresponds to an endpoint, intersection point, or segment knot on either function.

Our measure of node attribute similarity penalizes the differences in the lengths of the medial axes, and in the relative widths and width variations along the medial axes. In our experiments, we consider a value of $\sigma$ that is constant for all shapes and shape parts, but this could be replaced by a values obtained as a function of the model shapes or model parts being matched. We leave the problem of learning shape deformation parameters conditioned on object classes as the subject of future research.

5.4.2 Edge Attribute Similarity

We define the similarity $E_a(e, e')$ between the attributes of edges $e$ and $e'$ as a function of the relative side and position of the attachment represented by each edge. The attribute $\gamma(e) = p_{u,v}$ of edge $e = (u, v)$ is the normalized signed position of the attachment of node $v$ onto node $u$ as defined in Chapter 4. The sign of $p_{u,v}$ encodes the side of the attachment, and the absolute value of $p_{u,v}$ encodes the attachment position between endpoints 0 and 1 along the bone represented by node $u$. Figure 5.9 illustrates an example of the edge attributes of a bone graph.

The “0” end of the position along the medial axis encoded by a node is chosen
Figure 5.9: Example of the edge attributes of a bone graph. The attribute of an edge corresponds to the attachment position between a child bone and its parent bone. For clarity, we show the absolute value of each position and let the color of the edge represent its sign (i.e., the side of the attachment). The parameterization of edge attributes assumes that the lengths of all bones are normalized to the unit length. In the case of the root node 1, the “0” end is chosen arbitrarily. The “0” end of all other nodes is specified as the skeletal point closer to the attachment with their parent bone.

arbitrarily for root nodes and for nodes with multiple parents. In the case of nodes with a single parent, the “0” end is the point closer to the attachment point with the parent node. The sign of the position is specified by assuming a clockwise traversal of the skeletal points from the “0” end. We take this into consideration, and for the cases in which the parameterization is ambiguous, i.e., nodes whose in-degree is not one, we evaluate the node and edge attributes with respect to both possible choices for the “0” end, and keep the one that maximizes similarity.

We compute the edge attribute similarity as the product of sign similarity and position similarity. We define a constant penalty \( \omega \in [0, 1] \) for attachments on opposite sides, and let the sign similarity be

\[
\delta(p_{u,v}, p_{u',v'}) = \begin{cases} 
1 & \text{if } \operatorname{sign}(p_{u,v}) = \operatorname{sign}(p_{u',v'}) \\
\omega & \text{otherwise}
\end{cases}
\]

In our experiments, we found that \( \omega = 0 \) was too strict a penalty, and that any value in \([0.1, 0.8]\) yielded similarly good results (we let \( \omega = 0.6 \) in our experiments). Next, we let
the position similarity be the absolute difference in the positions of the attachments

\[ \psi(p_{u,v}, p_{u',v'}) = 1 - |p_{u,v} - p_{u',v'}|. \] (5.11)

The product of these two measures of similarity becomes our edge attribute function

\[ E_a(e, e') = \delta(\gamma(e), \gamma'(e')) \psi(\gamma(e), \gamma'(e')). \] (5.12)

Our measure of edge attribute similarity combines a linear penalty for the differences in the positions of the attachments with a constant penalty for the differences in the sides of the attachments. This represents a coarse measure of the similarity of part relations, and does not account for the relative ordering of child nodes that are attached to the same point of a parent node. Similarly to the case of node attributes, our measure of edge similarity could be improved by learning, at training time, the relative importance of the differences in the position and side of part attachments as a function of each object’s class. Then, this information could be exploited at matching time, when a novel exemplar is compared against a model of a known object class. We leave this problem as future research.

5.5 Conclusions

We introduce a novel algorithm for inexact matching that provides a general mechanism for incorporating domain knowledge about the relative importance of the structural and attribute differences between graphs. Our approach is a generalization of an existing method [83] for matching DAGs with attributed nodes. We define the matching problem in terms of four types of dependencies between node correspondences, which are based on graph structure, node attributes, edge attributes, and assignment uniqueness. Our measure of structural similarity allows us to capture a wider range of domain assumptions than is possible with the previous formulation of the approach. In turn, our measure of edge similarity reduces the ambiguity of edge correspondences by exploiting the conditional dependencies between the node correspondences in the solution at each iteration of the algorithm. Finally, we present a formulation of the matching algorithm in which the level of greediness in the selection of node correspondences can be specified. This option allows us in the next chapter to evaluate the practical cost of performing greedy steps by analyzing recognition performance as a function of computation time. Furthermore, we
compare our FA-DAG matching algorithm against the NA-DAG matching algorithm for the task of object categorization using bone graphs and shock graphs.
Chapter 6

Experiments

6.1 Introduction

In this chapter, we evaluate the bone graph representation together with the graph matching algorithm presented in Chapter 5 for the task of object categorization. The goal in this task is to recognize 2-D views of novel 3-D exemplars of known object categories. This task differs significantly from the exemplar-based object recognition and pose estimation experiments considered in Section 4.3. In particular, in our previous experiments, the task of estimating pose from unknown views of known exemplars allowed us to evaluate the benefits of addressing the over- and under-segmentation of skeletal parts, but provided little information about how the representation might cope with views of novel exemplars. The categorization problem, on the other hand, addresses this question directly. It also provides us with a more stringent task to evaluate the representation and the matching algorithms as part of a complete recognition framework.

Like the experiments in Chapter 4, we compare the bone graph representation to the shock graph. This comparison is meaningful because both representations take a skeleton as input and yield a graph of skeletal parts as output. This means that for the bone graph to outperform the shock graph, its structure and edge attribute properties must be more beneficial for matching than those of the shock graph. In our experiments, we measure the individual contribution of these properties by matching graphs with and without node and edge attributes.

In contrast to the experiments in Chapter 4, where for the purpose of comparison we sub-partitioned the nodes of the bone graph according to radius, here we are concerned with evaluating the bone graph as defined in Section 4.2. This definition of bone graph
leads to shapes that are represented in terms of fewer nodes than required by the shock graph, and has the advantage of significantly improving matching time, since the time complexity of the matching algorithms are functions that grow with the size of the input graphs. For example, in our dataset the average size of bone graphs is 10.8 nodes, while that of shock graphs is 20.5.

Finally, we note that our experiments evaluate the bone graph and our matching algorithm as components of a comprehensive approach for object categorization, and not as a complete framework for this task. We acknowledge that the solution to such a problem would demand a more ambitious approach that addresses the problems of learning class-specific features and parameters, efficiently indexing shapes, and selecting model views in a principled manner. In the next chapter, we discuss how the bone graph and the matcher can contribute to each one of these problems.

## 6.2 The Dataset

The dataset in our experiments is a subset of the Princeton Shape Benchmark (PSB) [82]. This dataset is publicly available and widely used for evaluating learning and recognition approaches using 2-D and 3-D shape representations. We select 8 classes and 5 3-D exemplars per class out of the 1,814 exemplars and 90 classes available in the PSB. Our selection of exemplars, shown in Figure 6.1, includes models whose part structure varies from simple (e.g., the hot air balloon class) to fairly complex (e.g., the walking human class). This subset provides enough data to compare the bone graph and shock graph representations and to discuss the limitations of our approach while performing a comprehensive set of experiments. In future research, we expect to consider larger datasets by combining our representation and matching approach with learning and indexing components.

We populate a model database of 2-D shapes by taking 25 uniformly-sampled views per exemplar in our dataset, which yields a total of 1000 shapes. We collect the sets of 2-D views of each 3-D exemplar by sampling a portion of its viewing sphere. We select the views that fall within the azimuth and elevation angles in the ranges $[-180, -90]$ and $[-30, 30]$ degrees, respectively. We take 5 evenly spaced views along each coordinate (i.e., 25 views per object). We found that this selection of views provides a sufficiently dense sampling of a 3-D object while capturing most of its non-degenerate views. As an example, the set of views of one of the horse exemplars is shown in Figure 6.2.
Figure 6.1: The Dataset of 3-D Models. It is formed by 8 classes with 5 exemplars per class. Half of the dataset is formed by “inorganic” objects, while the other half contains “organic” objects. This set of objects captures a wide range of part structure complexity, which ranges from very simple, in the case of the hot air balloon class, to fairly complex, in the case of the walking human and bird classes.

6.3 Experimental Set Up

We evaluate two graph matching algorithms in our experiments, and for simplicity, we refer to them as algorithms A and B. Algorithm A is the matching algorithm of Shoko-
Algorithm B is our generalization of algorithm A (Section 5.3), and accounts for edge attributes, as well as structural graph constraints that differ from those of algorithm A (Section 5.3.1). We consider the most greedy version of algorithm B, in which the size of the solution queue is bounded to one. Later, we evaluate the performance of the algorithm as a function of the size of the solution queue. Both matching algorithms require a domain-dependent node similarity function, while algorithm B also requires a domain-dependent edge similarity function. We use the node similarity function presented in [57] for shock graphs, and, when needed, let the edge similarity function be the identity function. For the bone graph, we use the node similarity function described in Section 5.4.1 and the edge similarity function described in Section 5.4.2.

The node similarity functions used for the bone graph and the shock graph are both based on comparing only the skeletal radius function represented by each node. That is,
these functions do not account for skeletal curvature or other skeletal properties encoded in the node attributes. The two functions differ mainly in that with the shock graph, the radius function of a node is assumed to be constant or vary monotonically, while with the bone graph, this need not be the case. Thus, the minor differences between the node similarity functions should not provide an unfair advantage for either representation.

In order to understand the influence of edge attributes of the bone graph, we carry out experiments with and without edge attributes (i.e., we let the edge similarity of the bone graph be the identity function) using algorithm B. Similarly, we evaluate the influence of graph structure in the matching process of both bone graphs and shock graphs by performing experiments with both node and edge similarity functions set to the identity function. In each figure plotting experimental results, we specify the attributes considered at matching time by labeling each representation as fully attributed (FA), node attributed (NA), edge attributed (EA), and unattributed (UA). For example, when matching the bone graph (BG) using both its edge and node attributes, we label the results as FA-BG.

We perform two sets of experiments in which all 1000 shapes are considered as queries while the contents of the model database vary as a function of the query class. In the first set of experiments, we evaluate recognition performance as a function of number of model exemplars for the query class. Here we expect the recognition performance to decrease as the shape variation of the exemplars in the query class becomes underrepresented by the reduction of its model exemplars. In the second set of experiments, we evaluate performance as a function of number of model views per exemplar of the query class. In this case, we also expect the performance to decrease as the query class is represented using fewer views per exemplar than the other classes. These experiments allow us to measure the performance of bone graphs and shock graphs relative to the performance associated with the numbers of exemplars and views of the query class.

In each experiment trial, we take an exemplar’s view and compare it against all shapes in the model database that belong to exemplars different from the query’s. That is, for each query view, the model database contains 4 exemplars of the query class and 5 exemplars of every other class (each exemplar is represented by 25 views). We say that class recognition is correct if the view that receives the highest similarity score belongs to the same object class as the query view. In the case of ties between a view of the query class and a view of a non-query class, we consider the outcome to be unsuccessful, since we seek a unique answer to the categorization question.
6.4 Object Categorization

We measure the influence of within-class shape deformation by evaluating recognition performance as a function of decreasing number of exemplars for the query class. In the first set of trials, each given query view is removed from the model database, along with all other views of the same query exemplar. In the next set of trials, we remove all the views of another exemplar of the query class. We repeat this procedure until there is only one exemplar of the query class remaining in the model database. Since the choice of which exemplars are removed influences the recognition performance, we try all possible combinations of exemplars and average the results. More specifically, there are 4, 6, and 4 ways of removing 1, 2, and 3 additional exemplars associated with each query class, respectively. The number of exemplars of the classes that are not that of the query view remains constant in each trial.

Figure 6.3 plots the recognition performance as the number of exemplars for each query class decreases from 4 to 1 (all the non-query classes are represented by 5 exemplars). The results exhibit the superiority of bone graphs and algorithm B over shock graphs and algorithm A. For the case of 4 exemplars, the score of the bone graph framework is 80.4% while that of the shock graph framework is 73.6%. This performance difference, 6.8 percentage points, is significant, as it is comparable to the decrease in performance due to matching bone graphs using 3 exemplars instead of 4, which has a score of 73.9%.

In order to evaluate the statistical significance of these results, we can consider the probability of obtaining the same results by randomly selecting each query’s most similar view from the model database. For each query view, the probability of selecting a view from the correct exemplar is \( \frac{n}{35+n} \), where \( n = \{4, 3, 2, 1\} \) is the number of exemplars from the query category in the model database. For instance, if we consider the case of \( n = 4 \) (i.e., the first column in Figure 6.3), the probability that 80.4% of the 1000 independent queries have a correct answer using this strategy is \( \left( \frac{4}{39} \right)^{804} + \left( \frac{35}{39} \right)^{196} = 6.14 \times 10^{-10} \). Thus, the probability of obtaining a performance comparable to that of bone graphs by chance is extremely low.

The incremental removal of exemplars (in addition to the query exemplar) from the model database characterizes the variability in performance associated with having different sets of models representing each query category. The similarity between the standard
error bars shown in the results suggest that both representations are comparably affected by the particular selections of model exemplars. It should be noted that, in practice, one would include all available model exemplars into the database in order to represent as much variability as possible for each object category.

We determine whether the performance improvement of the bone graph is due to the representation, the matching algorithm, or both, by evaluating the four combinations of matching algorithms and shape representations. Figure 6.4 plots these results and shows that algorithm B is the most effective algorithm for matching both bone graphs and shock graphs. The results also show that the superiority of bone graphs over shock graphs is not due to the matching algorithm alone, since, for example, in the case of 4 exemplars the algorithm used only accounts for less than half the performance difference between the two representations. The standard error bars shown in the plot suggests that both representations are sensitive to the choice of model exemplars that are left in the database.

We evaluate the contribution of graph structure stability to the matching process by considering the problem of matching bone graphs and shock graphs without their node and edge attributes. Figure 6.5 plots recognition performance for unattributed (UA) bone graphs and shock graphs using algorithms A and B. Here the structure of the bone graph seems to contribute little information. In the case of 4 exemplars, the recognition score is just 26.6% with algorithm B, which is slightly more than twice the chance level of 10.26% (i.e., 4 correct exemplars over 39 exemplars). On the other hand, the score for shock graphs is 34.1% with algorithm A, and 40.1% with algorithm B. One reason for this large performance disparity between bone graphs and shock graphs is that the partitioning of skeletal branches according to radius performed by the shock graph effectively encodes geometrical attributes of the shape parts into the structure of the graph. In contrast, the structure of the bone graph is influenced less by the geometrical properties of the shape parts, which leads to more dissimilar shapes having equal structure. This effect can be seen in Figure 6.6, where we plot the same results but count ties for first ranking position as successful trials. Now, the recognition score for bone graphs and algorithm B jumps from 26.6% to 42.9%, while that for shock graphs and algorithm B has a more modest increase from 40.1% to 42%. This confirms that bone graphs have a much larger number of ties in the similarity ranking than shock graphs.

---

1The standard error is computed as \( \frac{\sigma}{\sqrt{m}} \), where \( \sigma \) is the sample standard deviation and \( m \) is the number of samples.
Figure 6.3: Correct recognition rates for bone graphs (solid lines) and shock graphs (dash-dot line) as a function of decreasing number of model exemplars for each query class. Each of the 1000 views in the model database is considered as the query. Algorithm A is used to match node-attributed shock graphs (NA-SG), while our novel algorithm B is used to match fully-attributed bone graphs (FA-BG). The number of exemplars for each of the non-query classes is 5, while the number of exemplars for the query class decreases from 4 to 1 (the query exemplar is removed from the database prior to matching). All ways of removing one, two, and three exemplars out of four are considered, and the average and standard error of their recognition rate is plotted. That is, the values 3, 2 and 1 along the x-axis correspond to the average performance obtained from 4, 6, and 4 sets of 1000 trials, respectively. Each of such trials represents a different way of removing exemplars of the query category prior to matching.

We analyze the contribution of the edge attributes to the matching process by comparing the results of matching fully-attributed, node-attributed, edge-attributed, and unattributed bone graphs. Figure 6.7 plots these results and shows that the presence of edge information does not improve recognition performance when combined with node
Figure 6.4: Complete comparison between algorithms A and B applied to the bone graph and shock graph representations. See Figure 6.3 for more details.

attributes, but it does lead to a significantly better score than using graph structure alone. This suggests that the graph structure of bone graphs can become significantly more informative when combined with edge attributes. This result has important implications for the problem of indexing graphs based on their unattributed structure, such as the spectral approach proposed by Shokoufandeh et al. [85]. In the case of bone graphs, the results suggest that indexing features based on graph structure would indeed benefit from incorporating edge attribute information.

In the experiments above we treat all unsuccessful cases equally, i.e., regardless of whether the correct answer is close to the top or at the bottom of the ranking. However, it is also important to evaluate the actual ranking position of the correct answer that is best ranked. This is useful when the goal of the matching process is that of pruning the model classes down to a small set of candidates. Figure 6.8 plots recognition performance when the correct answer is ranked in the top three positions. These results show that with bone graphs, the correct answer is ranked within the top three positions for 90.3%
Figure 6.5: Correct recognition rates for bone graphs (solid lines) and shock graphs (dotted line) as a function of decreasing number of model exemplars for each query class. We repeat the experiments shown in Figure 6.3 using unattributed graphs and algorithms A and B. That is, only graph structure is matched. Here the shock graphs outperforms the bone graph regardless of which matching algorithm is used. These results suggest that the structure of a bone graph is less discriminating than that of the shock graph.

of the queries, while with shock graphs and algorithm A this happens only with 83.9% of the queries. In addition, the performance of the shock graph improves with algorithm B to 87.2%, which provides further evidence of the superior performance obtained by our matching algorithm regardless of the shape representation used.

We now evaluate recognition performance as a function of the number of views per exemplar. In this case, we fix the number of exemplars per class, and let the number of views representing the exemplars of the query class vary from 25 to 13 (the number of views representing the other classes is always 25). For simplicity, we remove the views closer to the center view of each exemplar in increments of 4. As in the experiments
Figure 6.6: Correct recognition rates for bone graphs (solid lines) and shock graphs (dotted line) as a function of decreasing number of model exemplars for each query class. We repeat the experiments shown in Figure 6.5, but this time count model views of the query class that are tied for first place in the similarity ranking as successful trials. This leads to a large increase in bone graph performance when compared with that shown in Figure 6.5, and suggests that the unattributed structure of the bone graph induces more ties, and therefore that it is a more ambiguous feature than the structure of the shock graph.

above, we used each of the 1000 views in the model database as the query. Figure 6.9 plots recognition performance for the case of a model database with 4 exemplars for each query class, and 5 exemplars for the non-query classes. These results show that the bone graph continues to outperform the shock graph as the number of views per exemplar decreases, which provides further evidence of the superior stability of the representation. The plots also suggest that the number of model views selected to represent each exemplar can have a dramatic impact on performance. In the case of bone graphs, the performance
Figure 6.7: Correct recognition rates for bone graphs as a function of decreasing number of model exemplars for each query class. We compare the performance of fully-attributed, node-attributed, edge-attributed, and unattributed bone graphs using algorithm B in order to determine the contribution of edge information to the matching process. These results suggest that the node attribute similarity is equally informative when used alone or combined with edge attributes. Moreover, it can be seen that the graph structure information does become significantly more discriminating when edge attributes are considered.

decreases 9.3 percentage points when nearly half of the views are removed. This suggests that a principled approach for selecting an optimal number of prototypical views per exemplar may lead to a significant increase in class recognition performance.

Finally, we evaluate the performance of algorithm B when the size of the solution queue grows up to 12. In particular, we conduct trials in which the constants \( K, \tau \) discussed in Section 5.3.3 are \((1, 1), (3, 1), (6, 2), (9, 3), \) and \((12, 4)\). We test the case of bone graphs when the query class is represented by four exemplars and 25 views per
Figure 6.8: Correct recognition rates for bone graphs (blue, labeled BG) and shock graphs (black, labeled SG) as a function of position in the similarity ranking. We compare the recognition performance for the case of 4 exemplars for the query class, and 5 exemplars for non-query classes in the database using algorithm B for matching fully-attributed (FA) bone graphs, and algorithm A for matching node-attributed (NA) bone graphs and shock graphs. The results show that with bone graphs, there is up to a 90.3% chance of finding a model view of the query class among the top three best ranked model views. In the case of the shock graph, the performance in this case is significantly less.

exemplar. In these trials, the performance remains constant at 80.4% up to a solution queue of size 6, and increases monotonically to only 80.6% for a queue of size 12. Since this represents a small performance increase at a high computational cost, it justifies the approach of considering only one greedy solution. Naturally, a more exhaustive search of the solution space could lead to better performance, but the computational cost would render the approach impractical.
Figure 6.9: Correct recognition rates for bone graphs (blue, labeled BG) and shock graphs (black, labeled SG) as a function of decreasing number of views per exemplar. We compare the recognition performance for the case of a model database with 4 exemplars for each query class, and 5 exemplars for the non-query classes. We use algorithm B for matching fully-attributed (FA) bone graphs, and algorithm A for matching node-attributed (NA) shock graphs. The results show that the bone graph consistently outperforms the shock graph as the number of views per exemplar decreases. Moreover, this experiment demonstrates that the selection of views representing each class exemplar can have a large impact on class recognition performance.

6.5 Examples of Node Correspondences

Figures 6.10, 6.11, and 6.12 illustrate a number of successful matches drawn from the experiments. The set of node correspondences found for each shape is shown by arcs connecting each pair of shapes, and by matching node colors between the corresponding bone graph representations of each shape. The examples demonstrate cases in which the matcher finds natural correspondences between parts with largely dissimilar geometries,
such as the torso and the legs. We have not measured the correctness of part correspondence among successful matches, but we have found that in practice, the correct matches generally yield mostly correct part correspondences.

It is interesting to analyze some examples of unsuccessful matches in order to determine the limitations of our approach. We show four such cases in Figures 6.13 and 6.14. Case (a) in Figure 6.13 illustrates the limitation of edge attributes for representing ordering and orientation of multiple end-to-end attachments. Here, the neck and tips next to it of the electric guitar have similar attachment (edge) attributes with the guitar’s body to the posts of the mailbox and its box. However, it can be seen that there is a strong visual difference in how these parts are attached to each other. Case (b) in 6.13 provides an example of two simple but very different objects with isomorphic graph structures. Here, in spite of the geometrical differences between the node attributes, the fact that the knife view is able to explain all nodes of the hot air balloon allows it to rank ahead of views of the correct class with more similar nodes. This illustrates that small details in a simple shape, such as convex corners, may have a disproportionate influence in the matching process. Finally, the examples in Figure 6.14 show that strong structural similarities can lead to matches between exemplars of object classes that may seem to have nothing in common, such as a fish and a knife, or a guitar and a bird.
Figure 6.10: Successful Matching Examples for Bone Graphs and Matching Algorithm B. In each pair of shapes and graphs, the one on the left is the query view and the one on the right is the best ranked model view. The part correspondences found by the matcher are illustrated as arcs between each pair of shapes, and as matching node colors between each pair of graphs. The number of each bone graph node corresponds to the shape part represented by it.
Figure 6.11: More Successful Matching Examples for Bone Graphs. See Figure 6.10 for details.
(a) Correct match for the class hot air balloon

(b) Correct match for the class knife

(c) Correct match for the class mailbox

Figure 6.12: More Successful Matching Examples for Bone Graphs. See Figure 6.10 for details.
Figure 6.13: Incorrect Matching Examples for Bone Graphs and Matching Algorithm B. Here we show two cases in which the graph structures and edge attributes are very similar, even though the shapes are not. The strong structural and edge attribute similarities in these cases help the wrong model exemplars rank well by compensating for the dissimilarity between the node attributes.
Figure 6.14: Incorrect Matching Examples for Bone Graphs and Matching Algorithm B. Here we show more cases in which two views of conceptually dissimilar classes have similar part structures.
Chapter 7

Conclusions

We present a novel shape parsing approach that recovers intuitive parts and relations from a shape’s medial axis. This approach is a contribution to the area of shape analysis, and can be used, for example, to recognize 3-D objects from their 2-D silhouettes. It is also a contribution to the subarea of medial representations, as it improves the stability of the medial branching topology of a shape’s skeleton in the presence of boundary protrusions. The parts and relations recovered by the parsing algorithm allow us to construct a novel hierarchical abstraction of a shape’s part structure. The result is the bone graph, a powerful parts-based shape representation that makes explicit the salient parts of a shape and their interrelations, and which provides structural constraints that are relevant to the matching problem.

We evaluate the part-structure stability of the bone graph by comparing it against the shock graph [89, 77] in a set of view-based object recognition experiments. The shock graph is a popular medial representation that does not alter the medial axis prior to decomposing the branches into parts, and which considers that every skeletal point is equally salient. We provide a meaningful comparison by evaluating both representations under the same matching framework and by using the same similarity functions. Our pose estimation experiments evaluate the benefits of addressing the over- and under-segmentation of skeletal parts, and of abstracting out the uninformative, and potentially complex, ligature structure of the medial axis. The results of the experiments show that the bone graph is significantly more stable and viewpoint invariant than the shock graph.

In order to exploit all the information encoded by a bone graph when comparing shapes, we explore the problem of matching directed acyclic graphs (DAG) with arbitrary sets of attributes for both nodes and edges. We build on our previous work on matching
hierarchical structures in the presence of noise, occlusion and clutter. We propose a novel generalization of the graph matching approach of Shokoufandeh et al. [83] that incorporates edge information, expands the range of domain constraints considered, and ensures (if desired) that node correspondences do not violate the ancestor/descendant relations between nodes. This framework is a contribution to the problem of matching generic DAGs, and therefore can be applied to a multitude of problems in pattern recognition.

The above contributions are largely independent of one another, but also form a coherent framework for view-based object recognition, which we evaluate for the task of object categorization. This task is a central problem in computer vision and requires an approach that can accommodate the within-category shape variation of object views. We compare the bone graph against the shock graph by matching them using both the algorithm of Shokoufandeh et al. [83] and our proposed generalization of this algorithm. This provides a relevant comparison since both representations take the same shape skeletons as input and yield graph-based encodings of their skeletal parts as output. This means that for the bone graph to outperform the shock graph, its structure and attributes must be more beneficial for matching than those of the shock graph. Our experimental results show that the bone graph significantly outperforms the shock graph for the object categorization task, and that the additional structural constraints exploited by our graph matching algorithm improve the recognition performance of both representations. The bone graph also leads to better computational performance as it represents the object silhouettes using, on average, half the number of nodes than the shock graph.

The experiments in Chapter 6 also evaluate whether the information provided by the edge attributes of the bone graph contribute to improving recognition performance. The results suggest that edge attributes improve performance in the absence of node attributes, but that when both node and edge attributes are considered, the edge attributes do not offer a significant benefit. A possible explanation for this is that the parameters that determine the penalty for edge attribute dissimilarities are sensitive to the category of the model being matched. That is, the positions and sides of part attachments (i.e., the edge attributes) may provide discriminating features for some object classes but not for others, which can lead to an over- or under-penalization of dissimilarities. We have found evidence of this phenomenon when analyzing a significant number of matching cases in our experiments.

A limitation of our view-based object recognition framework is that the rules to detect protrusions may fail to capture the part variability of some objects. When this problem
affects the decomposition of large shape parts (of similar silhouettes), it can lead to bone graphs with significantly different structures. Because the structures of the graphs are used to constrain the assignment of node correspondences, the matching algorithm cannot correct this type of parsing errors. A possible solution to this problem is to perform a denser sampling of the viewing sphere of each model object in order to ensure that all the representational variations of its views have a representative in the model database. Since this can lead to a large number of redundant model views, it must be integrated with a clustering procedure to find and eliminate views that are too similar. Another solution is to represent each shape using multiple bone graphs computed by varying the parameters that control the detection of protrusions. The disadvantage of this solution is its high computational cost, since a pair of query and model views would be represented by a multitude of graphs, which must be matched against each other.

7.1 Future Research Directions

In order to complete our framework for view-based object recognition, it is necessary to address the problems of: (a) learning the within-class variation of node and edge attributes, (b) eliminating redundant model views, and (c) organizing the model database such that it can be searched efficiently. These problems demand the clustering of model views according to similarity, and can benefit from the stable parts and attributed relations offered by the bone graph representation. More specifically, learning the attribute variability of the members of an object category can be used at matching time to penalize node and edge dissimilarities as a function of each model view.

The clustering of object views can also be used to study how the process of building and matching the bone graph can handle large scale changes between views. Furthermore, it can be used to improve search performance by organizing the model database as a hierarchy of prototypical views, e.g., as suggested by Sebastian et al. [79]. Such an approach creates a hierarchy of object classes by clustering the model views across object classes, and then selects prototypical views from each cluster.

Another promising avenue of research is to compute a low-dimensional abstraction of a bone graph (both query and model bone graphs) that can be used to prune all but a small set of candidate models with respect to a query. This is a form of indexing that is complementary to the hierarchical organization of the model database. An exciting approach can be to make the spectral signatures proposed by Shokoufandeh et al. [84]
more discriminating by encoding the edge attributes of a bone graph in them. These signatures are computed from the adjacency matrix of a DAG and encode no attachment information beyond adjacency. Our experimental results motivate the combination of edge attributes and graph structure of the bone graph by showing that together they provide more information for recognition than graph structure alone.
Appendix A

Partitioning with Contextual Information

The ligature-induced partition of a branch is a simple and unambiguous process that treats every boundary concavity the same way. However, in some domains, it can be beneficial to trade this simplicity and lack of ambiguity for a finer control over which ligature segments – and corresponding boundary concavities – are considered meaningful partition cues. In what remains of this appendix, we analyze three types of ligature configurations that allow for alternate interpretation to that of two bones joined by a ligament. These configurations are associated with intrusions, bends, and “soft” protrusions along the boundary, and are illustrated in Figures A.1 (a), (b) and (c), respectively.

We begin with the case of a single boundary intrusion, which can be interpreted as a partition cue or as an accidental “notch” depending on its size. For example, the depth of the intrusion suggests two bones in Figure A.1 (d) and a single bone in Figure A.1 (a). In turn, the concavity of an intrusion is locally similar to that of a bend which, for example, creates the “elbow” shape in Figure A.1 (b). Both intrusion and bend patterns arise as a pair of connected ligature segments that flow away from their meeting point. Interestingly, the geometrical difference between an intrusion and a bend can be recovered from skeletal properties alone by considering the BAR measure of the side opposed to the concavity. For a bend, such a BAR measure is expected to be greater than one, as the ligature segments map to a circular arc, while for an intrusion, the opposing boundary curve is expected to be straighter than the medial curve, and have a BAR measure smaller than or equal to one. In addition, a “perfect bend” can also arise
Appendix A. Partitioning with Contextual Information

Figure A.1: Examples of intrusions, bends, and soft protrusions: (a) and (d) are intrusions that may elicit different interpretations, where (a) is a bone with a notch, and (d) is two bones connected by a ligament; (b) is a generic bend in which the ligature segments flow away from the point closest to the concavity; (e), in contrast, is a “perfect” bend, as the radii of the ligature points are all equal; (c) is a soft protrusion, which can be transformed into an ES attachment by increasing the size of the bump, as shown in (f).

The size of an intrusion can be measured relative to its local context by comparing the radius of the point closest to the concavity against some estimate of the size of the connected non-ligature parts. A natural choice for this estimate is the average of the maximum radii of the two segments of non-ligature points that flow away from the intrusion (see details in Fig. A.2). In general, an intrusion with a small size relative to its surrounding is a weaker partition cue than that of a deeper intrusion. However, the final judgment on when to partition and whether or not to consider multiple interpretations depends on the domain. For example, in a many-to-many matching framework, the differences in the partition of a query and a model shape can be handled by the matcher, while in a one-to-one matching framework, these differences are best handled at the representation level.

The last configuration of interest is that of a soft protrusion (see Fig. A.1 (c)) which, in addition to being interpreted as an EE bone attachment or as a single bone, can also be seen as a degenerate ES attachment. A soft protrusion – or “bump” – arises as two connected ligature segments that flow toward their meeting point (see Fig. A.1 (c)).
Figure A.2: Measuring the relative size of an intrusion. The radius of the meeting point between the ligature segments, denoted $r_0$ in the figures, corresponds to the smallest radius value that is related to the intrusion. In turn, the largest radius values associated with the intrusion are given by the last points of each segment with monotonically increasing radii away from the ligature, which are denoted $r_1$ and $r_2$ in the figures. A natural size estimate is given by the quotient between the smallest radius and the average of the largest radii, such that the intrusion size becomes inversely proportional to $\frac{2r_0}{r_1 + r_2}$.

Figure A.3: Examples of opposing intrusions and soft protrusions: (a) two opposed and symmetric intrusions produce two full ligature segments that flow away from their meeting point; (b) two opposing and symmetric “bumps” produce a pair of full ligature segments that flow toward their meeting point; (c) opposing intrusions produce a similar ligature configuration to that of symmetrically opposed intrusions; (d) the configuration in (c) can also arise when two pairs of symmetrically opposed bumps are concatenated; (e–h) show examples of ligature configurations obtained by realigning the bumps and notches along the boundary.
Figure A.4: The complete spectrum of internal ligature configurations. Thick dots represent the endpoints of each segment with monotonically varying (arrow line) or constant (crossed line) radius values. The dotted line segments represent both ligature and non-ligature segments.

refer to this point as \textit{intra-ligature}. Under the ES attachment interpretation, the intra-ligature point is seen as a “third” branch of length one which, in turn, creates a three-branch junction point. Then, the “junction” is treated as a regular undersegmentation case, as discussed in Section 3.6. Moreover, as is the case with regular oversegmentation, the ES interpretation of bumps along a branch must be restored before a branch is partitioned, as the restoration process changes the skeletal information of the branch. For example, the middle boundary points of Figure A.3 (d) form an intrusion pattern that is eliminated after the restoration of the soft protrusions.

The alignment of intrusions and soft protrusions on opposing boundary sides can also influence the perception of shape parts. For example, two symmetrically opposed bumps form an “ankle” shape (see Fig. A.3 (b)), while an opposing notch and bump form a “knee” shape (see Fig. A.3 (c)). In general, chains of three or more ligature segments provide a strong partition cue, but as is the case with single intrusions and soft protrusions, this interpretation can be overruled to satisfy domain preferences. For instance, the non-generic case of two symmetrically opposed bumps, which arises as two connected \textit{full} ligature segments that flow toward their meeting point (Fig. A.3 (b)), can be considered a four-branch junction and interpreted as a form of branch \textit{crossing}, which is discussed in Section 3.5.
Appendix B

Side-To-Side Attachment of Parts

In this appendix, we discuss alternate interpretations of nested configurations that leads to bone attachment relations not considered by our shape parsing algorithm. The nested ligature configurations discussed in Section 3.5.2 are divided into three types. The third type is formed by concave corners on opposite boundary sides of the host part of a protrusion (Fig. B.1 (a)). This configuration is particularly interesting because it allows for several interpretations of shape parts with smooth boundaries and medial curves. Some of these interpretations define attachment relations that are different from the end-to-side and end-to-end types discussed in Chapter 4. For example, the shape in Figure B.1 (a) can be interpreted as two parts that are attached side-to-side (Fig. B.1 (e)), or as two parts that overlap one another (Fig. B.1 (f)). The solution to this ambiguity may depend on perceptual preferences for the relative size of the boundary gaps created by the protruding branches and the smooth continuation of the medial axes of the resulting parts. These preferences, in turn, may depend on the domain (e.g., human silhouettes, typographic characters, etc.). We leave as future work the evaluation of these interpretations during the parsing of a shape.
Figure B.1: Examples of nested protrusions that allow for several interpretations. (a) A shape with four concave corners that create two junctions connected by full ligature points. The labeling of this ligature configuration and the merging of branches identified as hosts defines the parsing of the shape. (b–c) The shape is parsed as three parts that are attached end-to-side. (d) The shape is parsed as five parts that are attached end-to-end. (e) The shape is parsed as two parts that are attached side-to-side. (f) The shape is parsed as two overlapping parts. The exploration of these possible parses in the representation of a shape is left as future work.
Bibliography


