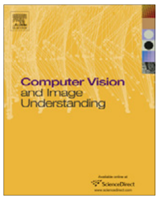
Computer Vision and Image Understanding 115 (2011) 1044–1061

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu



Bone graphs: Medial shape parsing and abstraction

Diego Macrini [a,*], Sven Dickinson [b], David Fleet [b], Kaleem Siddiqi [c]

[a] *School of Information Technology and Engineering, University of Ottawa, 800 King Edward Av., Colonel By, Room B407, Ottawa, Ontario, Canada K1N 6N*
[b] *Department of Computer Science, University of Toronto, 6 King's College Rd, Room PT 283, Toronto, Ontario, Canada M5S 3H5*
[c] *McGill University, McConnell Eng., 3480 University Street, Room 318, Montreal, Quebec, Canada H3A 2A7*

ARTICLE INFO

Article history:
Available online 17 March 2011

Keywords:
Medial shape representation
Shape parsing
Graph-based object representation

ABSTRACT

The recognition of 3-D objects from their silhouettes demands a shape representation which is stable with respect to minor changes in viewpoint and articulation. This can be achieved by parsing a silhouette into parts and relationships that do not change across similar object views. Medial descriptions, such as skeletons and shock graphs, provide part-based decompositions but suffer from instabilities. As a result, similar shapes may be represented by dissimilar part sets. We propose a novel shape parsing approach which is based on identifying and regularizing the ligature structure of a medial axis, leading to a *bone graph*, a medial abstraction which captures a more stable notion of an object's parts. Our experiments show that it offers improved recognition and pose estimation performance in the presence of within-class deformation over the shock graph.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

A skeletal description expresses shape as a set of symmetry-based parts, and has a long history in the shape recognition community. Binford's generalized cylinders [5] represent a 3-D object in terms 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 [6]. Skeletonization algorithms map a closed 2-D shape to a set of medial branches that terminate at endpoints or branch junctions. These branches can then serve to decompose the object into parts to be used for shape matching.

Unfortunately, the branching structure of the medial axis has been shown to be very sensitive to perturbations of the boundary, and to the addition or deletion of object parts [2]. This instability has limited the use of skeletons in the recognition community, since skeletal branches do not always map, in a one-to-one fashion, to meaningful shape parts. Often there are many more branches than meaningful parts (over-segmentation), and occasionally there are fewer branches than meaningful parts (under-segmentation). For example, the shorter rear leg of the dog in Fig. 1a results in an incident branch that oversegments the representation of the dog's body into two parts (i.e., skeletal branches). An enlarged view of the junctions also reveals a similar situation where the front legs meet the body. In addition, the representations of each of the four legs and the tail are undersegmented in the sense that the associated skeletal branches extend well past the locations of the part attachments with the dog's body. The net result is that a one-to-one mapping of skeletal branches to shape parts may not always be stable and intuitive.

To cope with skeletal instability, which leads to similar shapes having skeletons with dissimilar branching structure, matching algorithms are required to find many-to-many assignments of parts between a model and a test shape [11,10], or they 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 [30,27]. In fact, in [27] the transitions of the medial axis [15] are themselves incorporated in edit-distance operations which allow topologically distinct skeletal structures due to similar shapes to be matched.

This paper advocates a more stable representation as an alternative to passing the instability to the matcher. In what follows, we shall use the term *skeletal branch* to refer to the complete locus of skeletal points between two branch points (or an endpoint and a branchpoint), and the term *segment* to refer to a closed interval of skeletal points within a branch. We employ *ligature* analysis [6] to devise a skeletal representation in which skeletal branches map in a one-to-one fashion to stable shape parts. As developed in Section 3.1 in detail, ligature regions (the green[1] curves in Fig. 1) are segments of the skeleton that 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. 1a, where much

* Corresponding author.
E-mail address: dmac@cs.toronto.edu (D. Macrini).

[1] For interpretation of color in Figs. 1–3 and 5–28, the reader is referred to the web version of this article.

1077-3142/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2010.12.011

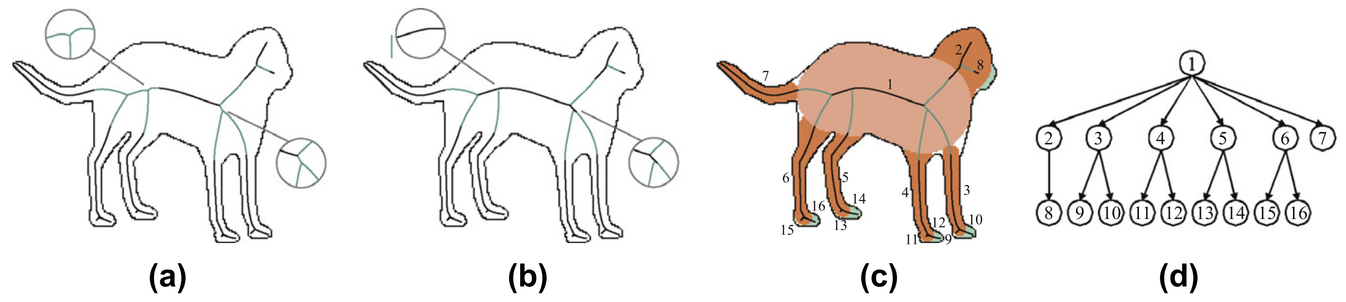


Fig. 1. Intuitive part decomposition. (a) A one-to-one mapping of skeletal branches to parts can lead to over-segmentation and under-segmentation. For example, the medial axis of the dog's body is given by two skeletal branches (instead of one) due to the junction point that represents the connection 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 called *ligature* regions, and they contribute little to the shape of the object (they are defined more formally in Section 3.1). 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. The ligature regions also gives rise to part under-segmentation, exemplified by the skeletons of the legs and tail extending well beyond their attachments to the body (in order to meet up with the body's skeleton, thereby preserving connectedness). (b) Our algorithm for detecting and removing ligature-induced skeletal instability uses a novel local ligature analysis to first identify and rectify the branch oversegmentation due to part protrusions. (c) A second ligature analysis then idenifies and rectifies branch under-segmentation, yielding 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

of the dog's body skeleton (near the back legs) is labeled as ligature.

In our analysis of ligature (Fig. 1a), we first identify the cases of part protrusions, and proceed to rectify the oversegmentation caused by these protrusions through a branch merging process. This yields branches that adhere to the geometric properties of the medial axis (Fig. 1b). A second stage of ligature analysis yields a new set of ligature segments arising from skeletal branches that undersegment the medial axes of the shape parts (Fig. 1c). Finally, the correction of undersegmented medial axes is shown in Fig. 1c. 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, and together they yield a reconstructed shape that is similar to the original. For example in Fig. 1c each colored part is the reconstruction of one non-ligature (black) branch, the union of which is close to the original shape. This final set of non-ligature segments determines the salient medial parts of the object.

In addition to a new framework for skeletal-based shape parsing, this paper also advocates a novel parts-based shape representation that takes advantage of our ligature-based analysis. We assemble the restored non-ligature segments into a *bone graph* (in Section 4), 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. We evaluate the bone graph in Section 5 by comparing it to the popular shock graph [30,27] 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 variation in a silhouette's shape than the shock graph, leading to significantly improved recognition performance.

2. Related work

The medial axis transform (MAT) [6] is a method for parsing a silhouette into its symmetric parts (skeletal branches) and their adjacency relations (branch connectivity). The skeleton is a complete representation of the silhouette, and its symmetry axes are thought to play an important role in human shape perception [31]. These attractive properties have spawned an entire research community, yet, despite their appeal, skeletal-based representations face some significant challenges due to skeletal instability.

The more manageable form of instability, namely the removal of skeletal "noise" due to small boundary perturbations, has been effectively addressed using boundary smoothing and/or skeletal pruning techniques, e.g., [28,33,3]; we will not address this form of skeletal instability in this paper. The second form of instability, namely structural instability due to ligature, is more challenging. While analysis of this instability has yielded stability measures ranging from very local (skeleton point) [18] to semi-local (skeleton branches) [33], most efforts can be viewed as a form of *skeleton processing*, mapping input skeletons to output skeletons, as opposed to a form of abstraction, i.e., mapping input skeletons to higher level shape representations. Moreover, their evaluation is typically anecdotal (visual), lacking the context of a particular (e.g., recognition) task. The one exception is the shock graph [30,27], but there the one-to-many mapping of skeletal segments to abstract parts unfortunately carries forward this skeletal instability.

The notion of ligature and its relation to skeletal instability was first proposed by Blum and Nagel [7]. It was later revisited by August et al. [2]. Using several examples, they demonstrated that non-ligature segments of the skeletal branchs appear to be stable. Hence the development in [2] focused on ligature removal since shape reconstruction without ligature appeared to cause little degradation 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 [16], could swiftly lead to topological changes in ligature regions, providing further motivation for their removal. However, these developments fell short of an algorithm, based on ligature analysis, that produces a more stable, abstract representation. Furthermore, no direct attempt was made to apply these ideas to the then emerging techniques for skeletal graph matching.

A concept related to ligature has been used by Katz and Pizer [18] where measures of *connection* and *substance* are associated with each skeletal point. 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, captured by a continuous connection value of each skeletal point. However, thus far this representation has not been evaluated in the context of an object recognition task.

There are more recent approaches to dealing with topological instabilities induced by ligature via a type of skeletal simplification. Telea et al. [34] propose a Bayesian framework for skeletal simplification/smoothing which seeks to find an optimal balance between structural simplification and the reconstruction error that results from simplification. The method seeks to collapse small skeletal branches while preserving salient object parts, but does not explicitly use ligature properties or cope with ligature-induced oversegmentation effects. van Eede et al. [35] extended this approach by basing the simplification on an ad-hoc type of ligature analysis. To preserve connectivity, contiguous ligature substructures are replaced with linearly interpolated skeletons. Whereas this provides a degree of regularity, ligature structures are not explicitly encoded and oversegmentation effects could remain.

The above approaches which use ligature properties share the limitation that they assume that these can be computed independently of the local influence of nearby boundary protrusions. This is problematic as it is violated often, even by simple shapes such as that in Fig. 1. This problem was addressed by Rom and Medioni [26] and by Juengling and Prasad [17] 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 [33] 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.

Mi and DeCarlo [21] also remove parts iteratively, by detecting *transitional areas* between adjacent branches in the Smoothed Local Symmetries (SLS) representation [8]. The influence of the order in which parts are removed in the part decomposition 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 reflects a desire to remove smaller parts first. This is similar to the dependency graph that we use to determine the order in which skeletal branches are merged (Section 3.4.2).

Mi et al. [22] extend the above approach in order to account for the influence of boundary intrusions on the part decomposition process by also computing and analysing the external skeleton of the shape (i.e., the skeleton of the background). A part decomposition heuristic is used to identify intrusions, which are then iteratively removed in order to obtain an unperturbed boundary. Since the removal of such parts alters the boundary of the shape, the skeletons of foreground and background shapes are recomputed at each step. The detection and removal of intrusions is an important contribution and is a key difference with other approaches, including ours.

Zhu and Yuille [36] construct an object modeling and recognition system based on medial-based parts. Utilizing deformable templates, they estimate the skeletons of an object's mid-grained parts, constrained to belong to two classes, deformable *worms*, or constant cross-sectioned elongated parts, and *circles*, or the compact parts that model short parts and joints between parts. A bottom-up process recovers a set of mid-grained parts and their connections. However, like the vast majority of other skeletal-based descriptions, no attempt is made to rectify the skeletal instability (part over- and under-segmentation) due to medial axis bifurcations. Instead, this instability is passed on to the matching algorithm to resolve, under the direction of an object model. In contrast, we seek to overcome this instability *without* the aid of a target model, instead identifying the salient parts and connections in an object-independent manner.

There is also a probabilistic approach to dealing with the oversegmentation of skeletal branches, which is proposed by Singh and Feldman [14]. In their work, the concept of ligature does not play a role, and they instead construct a Bayesian probabilistic model to estimate the set of skeletal branches that are most likely to have produced a 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 their work is to reduce the presence of spurious branches due to boundary noise, the oversegmentation problem is also addressed by merging two of the branches incident at a branchpoint. This approach may result in non-intuitive decompositions, e.g., in situations where the limbs connect with the body in Fig. 1.

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 [24], but instead of associating a shape with a family of skeletons, it selects a single scale 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 [32]. The result is a disconnected skeleton that has fewer branch junctions than a regular skeleton, but still captures the coarse medial properties of shape parts. However, the resulting branches may be significantly longer than the parts they represent and their adjacency relations are not clearly defined.

The approaches based on iterative part removal discussed above address both the under- and oversegmentation instabilities of the medial shape representation, and are similar to the shape parsing approach introduced in Section 3. However, since they do not propose a representation and matching framework for recognition, they cannot be directly compared to the bone graph (Section 4). In contrast, Siddiqi et al. [30] propose a shape abstraction with recognition in mind. The shock graph is a directed acyclic graph (DAG) encoding a coarse-to-fine decomposition of a shape into skeletal parts; similar variants are described in [19]. 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. It therefore inherits the oversegmentation instabilities of the medial axis. In Section 3.5, we discuss the partitioning method of shock graphs, and in Section 5 we compare shock graphs and bone graphs in recognition experiments.

3. Medial shape parsing

The junctions and endpoints of the medial axis have been used in the shape community to decompose shapes, wherein each branch of the medial axis is used to define one or more shape parts. However, as noted by August et al. [2], as protrusions are introduced or removed or are perturbed, the number of skeletal branches may not reflect the number of salient parts.

We argue that a one-to-many relation between skeletal branches and parts can be obtained by eliminating certain junctions. To this end we show how skeletal and boundary properties can be used to determine which branches should be merged to remove unwanted junctions. We further show how to preserve branch adjacency relations in order to maintain a modified skeleton which both approximates the original shape, and for which the remaining branches coincide with salient parts and part connections. In contrast to previous methods (e.g., [30,27]), we argue that not every skeletal point reflects properties of a salient part. Rather, we identify two types of skeletal segments, namely, *bones* and *ligaments* (e.g., see Fig. 2). Bones map one-to-one to the medial axes of shape parts, while the ligaments connect bones to other

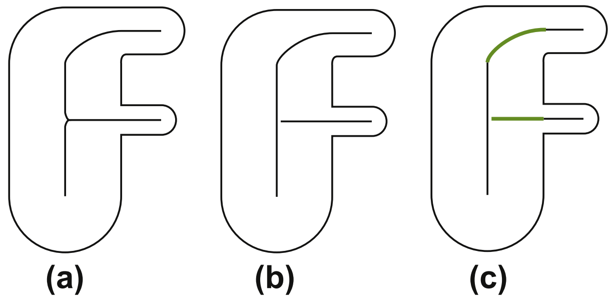


Fig. 2. Bones and ligaments. (a) We begin the parsing process with a given medial axis. (b) Next, we detect the branch junctions that correspond to boundary protrusions and merge the medial axes of the parts that host such protrusions. The result is a skeleton whose branches now map one-to-many to the 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

bones. In this way, 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.

3.1. Local geometry of the medial axis

We begin with a very brief review of several properties of the medial axis. Let Ω be the set of all points (x,y) within the interior of a 2-D object delimited by a simple closed curve. Let $\mathcal{S}(\Omega)$ be its interior medial axis [6]. Each skeletal point in $\mathcal{S}$ is characterized by a position $\mathbf{p}$ and a radius r. The relationship between the object angle θ, the spoke vectors $b^{\pm 1}$, and the direction of the unit tangent vector $\mathbf{T}$ is depicted in Fig. 3 where

$$\theta = \arccos\left(-\frac{dr}{d\tilde{s}}\right), \tag{1}$$

and $\tilde{s}$ 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 [29]. The variation of the radius along a curve is also used to define *flow* [15]. The direction of flow is the direction of increasing radius, and when necessary it is indicated as an arrow on the medial curve (e.g., see Fig. 4).

The degree of a skeletal point is determined by the number of points of intersection between a disk of radius ϵ centered at that point and the skeleton, as $\epsilon \to 0$. Endpoints have degree one. Junctions have degree three or higher. A *skeletal branch* is a continuous

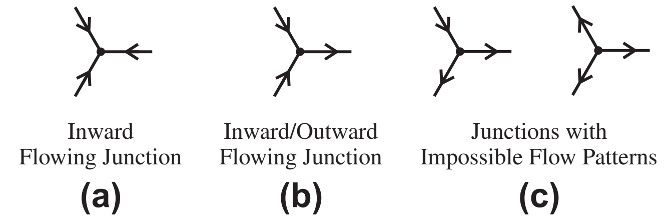


Fig. 4. The two generic cases of branch junctions (adapted from [15]). The arrows correspond to the direction in which the radius functions increase and represent a notion of flow direction. (a) The radius functions of all branches increase toward the junction. (b) The radius function of one branch increases 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 [15].

curve of skeletal points that terminates at either end at an endpoint or a junction. With the exception of these terminal points, each point in a skeletal branch has degree two.

The adjacency relations between branches define the *branching topology* of the medial axis [29]. We focus on *generic* junctions (i.e., junction points whose degree remains unchanged under small perturbations of the boundary curve [15]). There are two types of generic junctions in the medial axis (see Fig. 4): (1) junctions of degree three with only one outward flowing branch; and (2) junctions of degree three, with three inward-flowing branches.

The ratio of boundary length to medial axis length also plays an important role in our medial parsing approach. An open interval about a medial axis point in a skeletal branch (i.e., a segment) is associated with two intervals on the boundary curve, one on either side. The endpoints of these boundary curves are determined by the spokes at the corresponding endpoints of the segment on the medial axis (see Fig. 3). If one considers the limiting ratio of the associated boundary length to medial segment length, as this interval on the skeleton shrinks to zero, one obtains the two *boundary-to-axis ratios* (BARs) for the skeletal point [7]. In what follows, we will take a slightly different interpretation where we shall consider the BAR of a particular segment to be the ratio of associated boundary length to segment length, i.e., we will not use the differential process.

3.2. Intuition and goals

One of the key goals of the algorithm below for parsing the medial axis concerns the detection of protruding parts and attachment relations between parts. The relation of the boundary protrusion and the medial axis is twofold. First, a boundary protrusion can be associated with a branch junction in the medial axis. Second, the boundary concavities on either side of the protrusion lead to special skeletal points known as *ligature* [6,2]. These are skeletal points whose spokes end at a boundary concavity and can be grouped into segments that have a BAR smaller than one. Ligature points are shown in green in Figs. 1 and 5.

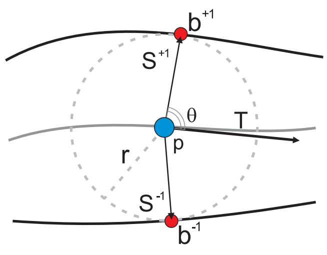


Fig. 3. Local geometry of a skeletal curve (adapted from [29]). The maximum inscribed disk at a regular skeletal point $\mathbf{p}$ with radius r touches the boundary at two bitangent points b^{+1} and b^{-1}, defining two spokes emanating from $\mathbf{p}$. The angle between the unit tangent $\mathbf{T}$ (to the medial curve) and either spoke is θ, the *object angle*.

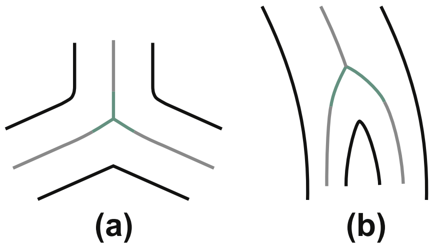


Fig. 5. Examples of part attachment that do not correspond to boundary protrusions (ligature points are colored green). In these two examples, the number of branches agrees with the number of intuitive shape parts. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Using the conventional terminology in the literature [6,2], a segment of skeletal points can be labeled as *non-ligature*, *full-ligature*, or *semi-ligature*. Fig. 6 shows examples of full-ligature segments (red) and semi-ligature segments (blue). A full-ligature segment is a set of connected points of a skeletal branch associated with a pair of opposing concave corners on the shape's boundary (Fig. 6). 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. We consider boundary protrusions to be nested if they induce nested ligature segments.

Part attachments are labeled according to 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. Other attachment types, like those in Fig. 5, do not define distinct roles in the attachment, and therefore, correspond to *undirected* relations. Our parsing algorithm produces an effective partitioning of the medial axis into non-ligature and ligature segments that we call *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.

Fig. 6a and b illustrate examples in which boundary concavities cause medial branches to be partitioned into ligature and non-ligature segments. Fig. 6c–h illustrate examples of different types of protrusions and the pattern of junctions and ligature points associated with them. In particular, Fig. 6c and d show simple protrusions, while Fig. 6e–h show nested protrusions. Since ligature configurations are complex when nested, an important component of our parsing algorithm is the untangling of such cases.

The main steps of our shape parsing algorithm are as follows:

1. Compute the interior skeleton (or medial axis) of the input (solid or closed) shape (or contour).
2. Identify the skeletal points that form ligature (Section 3.3).
3. Label each branch junction according to whether it is the result of the protrusion of a shape part from another or not (Section 3.4). This step requires the recursive processing of nested protrusions (Section 3.4.2) and the merging of the branches representing the medial axes of the host parts (Section 3.4.3). The result is a skeleton in which the junctions associated with protrusions are removed (Section 3.4.1).
4. Partition the branches of the new skeleton into non-ligature and ligature segments in order to determine bones and ligaments. A maximal segment of non-ligature points defines a bone, while a maximal segment of ligature points defines a ligament (Section 3.5).

The following sections give the details of each step. Section 3.6 concludes with a complete procedural specification.

3.3. Ligature detection

The algorithm for ligature detection in [2] first identifies points of minimal negative curvature along the boundary, and then labels those skeletal points whose bitangent points b^{+1}, b^{-1} fall within an ϵ-ball of the curvature minima points. 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, Fig. 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.

The approach to ligature detection we advocate begins not with curvature minima, but rather with the boundary-to-axis ratio (BAR). We look for segments of skeletal branches that have a BAR less than one for the associated boundary on one side of the branch or the other, and whose associated boundary points all have negative curvature. The first step of the algorithm searches for all maximal intervals of skeletal branches with a BAR significantly less than 1. For each branch the search is performed using the BAR computed for the boundaries on one side of the branch, and then for the other side. In the experiments below, we use a BAR threshold of 0.75, which admits smooth concavities. A smaller threshold leads to fewer smooth concavities inducing ligature points,

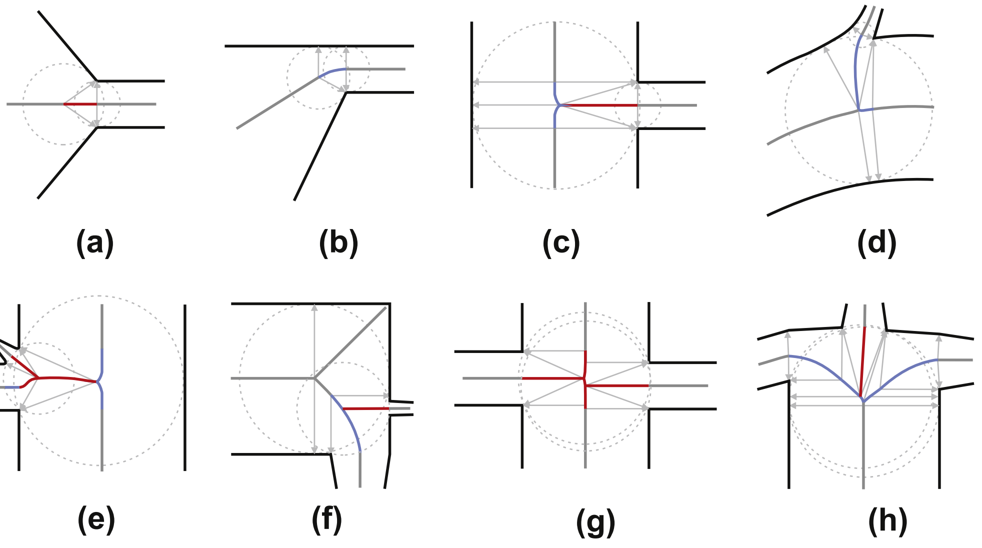


Fig. 6. Examples of ligature segment configurations (the basic cases a, b, and c are adapted from [2]): (a) full-ligature (red) segment (induced by a pair of concave corners), and (b) semi-ligature (blue) segment (induced by a single concave corner) that partition a branch into ligature and non-ligature 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). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

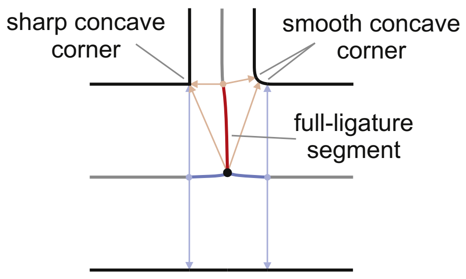


Fig. 7. Ligature from sharp and smooth concave corners. A sharp concave corner (left) has the property that the spoke vectors associated with the ligature segment coincide at it. In contrast, a smooth concave corner (right) is traced out by the endpoints of non-coincidental spoke vectors. In both situations the boundary-to-axis ratio is less than one.

potentially resulting in fewer junctions labeled as protrusions and fewer branches merged.

The points along the boundary that correspond to (or are mapped by) a ligature segment must also have negative curvature. For each branch segment found in step 1, with BAR less than the threshold, we search for intervals of maximal length within which all points have negative curvature. To compute boundary curvature, we use the approach of Chetverikov and Szabo [9], modified to account for the scale information provided by the candidate ligature interval in the first step; for details on the modified approach, see [20]. Each interval has negative curvature and a BAR less than one, and is therefore deemed to be a ligature segment. For intervals of ligature with a BAR less than one and negative curvature on boundaries on both sides of the branch, the points are full-ligature. The remaining points are semi-ligature.

Finally, when an interval found in step 1 (with BAR less than threshold) includes a junction point at the end of the branch, one must also consider the adjoining branch on the other side of the junction. This is needed because the ligature segments on both sides of the junction map to the same corner; both are required to determine the scale for curvature estimation. For details, see the caption in Fig. 8.

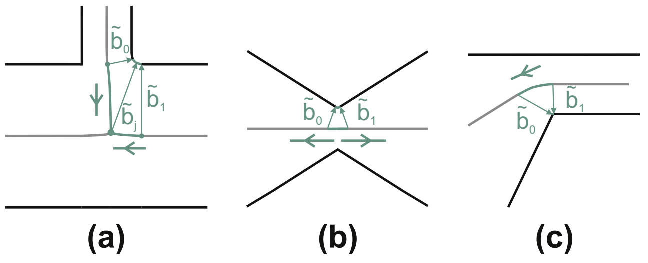


Fig. 8. The three configurations of concave corners and ligature segments considered by our ligature detection algorithm. We identify these configurations according to the relative flow directions of adjacent ligature segments (shown as arrows next to each segment). (a) The spokes emanating from two 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 segments 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.

3.4. Detecting protrusions

The detection of part protrusions relies on the analysis of branch junctions. A specific type of junction, which we call a *P-junction*, signals the existence of a part protrusion (e.g., see Fig. 9a). P-junctions also define a directed binary relation between shape parts; i.e., one branch of a P-junction corresponds to the protruding part, while the other two branches are deemed to belong to the host shape part from which the protrusion emerges. Our algorithm will eventually merge the host branches at P-junctions so the entire host part is explicitly represented. Other junctions, like that in Fig. 9b, correspond not to a protrusion but to a point where three parts connect. We refer to these junctions as *Y-junctions*.

We hypothesise that not all junction/ligature configurations occur. Fig. 10 shows those that are hypothesized to be impossible. The table in Fig. 11a depicts the remaining configurations that are possible, several of which occur in the example shapes shown in Fig. 11b. The junction/ligature patterns in Fig. 11 are classified as either Y-junctions, P-junctions, or *nested* junctions. The nested junctions correspond to junctions connected by ligature points (see Section 3.4.2). In many cases the same junction/ligature pattern corresponds to several different boundary shapes, and most of these are cases of nested ligature.

The main steps of our junction labeling algorithm that detects and labels part protrusions are as follows:

1. All junctions are given a label of Y, P, or nested (see Section 3.4.1).
2. We analyze nested ligature/junction configurations in order to determine an ordering for merging the host-part branches within the nested junctions (Section 3.4.2).

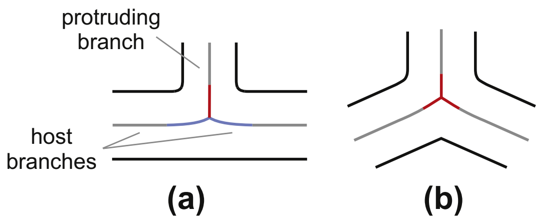


Fig. 9. Examples of *P-* and *Y-junctions*. (a) A P-junction represents a relation between two shape parts in which the medial axis of one part protrudes from the medial axis of a *host* part. In this case, the medial axis of the host part is formed by the two horizontal branches, while that of the *protruding* 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.

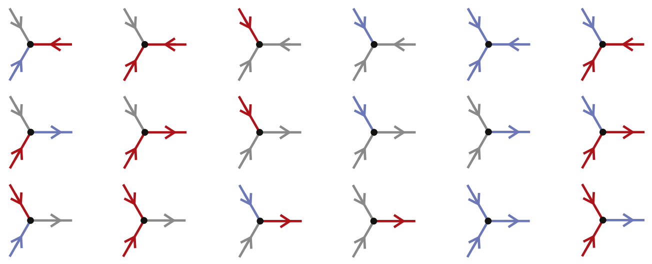


Fig. 10. Impossible junction and ligature configurations. Red, blue and gray colors correspond to full-ligature, semi-ligature and non-ligature segments, respectively. We consider these configurations impossible due to the assumption that every concave corner associated with a junction induces ligature 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

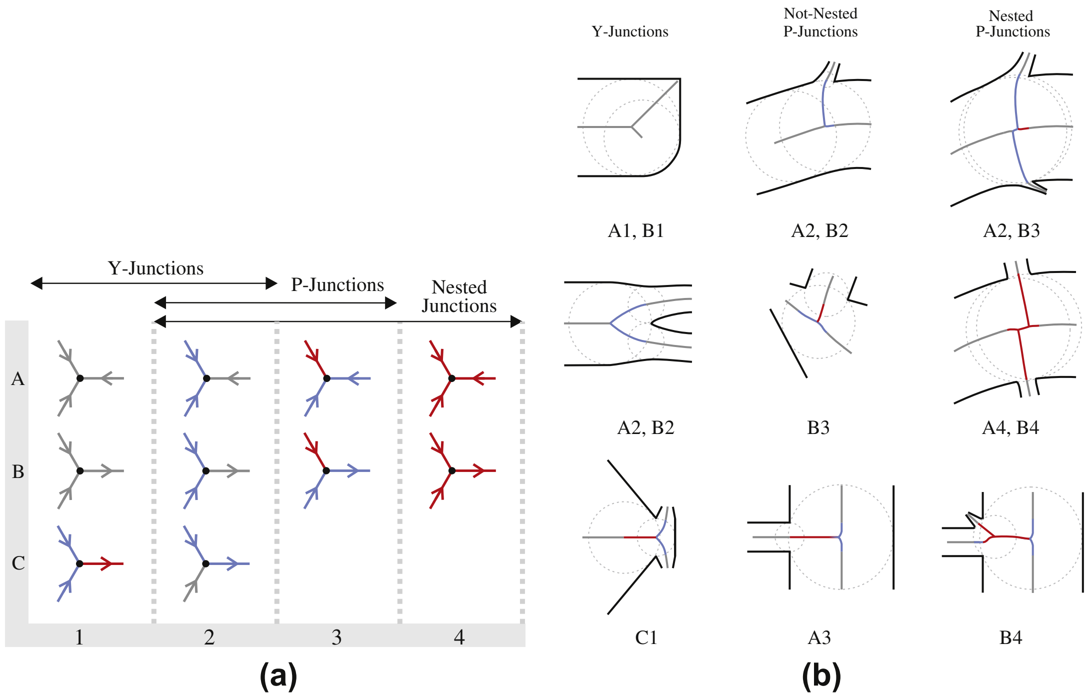


Fig. 11. Possible junction/ligature configurations. (a) Row A in the table corresponds to the left junction type in Fig. 4a, and rows B and C correspond to the right junction type in that figure. The headings of each column show that the same junction/ligature configurations occur within different contexts in the examples shown in (b), and may lead to different shape part interpretations (see Section 3.4.1 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.

3. We then merge branches in that order, thereby labeling junctions and identifying parts in a recursive fashion (see Section 3.4.3).

3.4.1. Labeling junctions as Y, P or nested

The main cues we use for labeling junctions are derived from the ligature properties of the incident branches. Frequently, the presence of a single full-ligature segment is a strong cue for detecting a protrusion. This is consistent with perceptual studies that suggest that humans use nearby concave corners on opposite sides of a medial axis as a cue for decomposing a shape into parts [31]. In this case, the full-ligature segment identifies the branch associated with the protruding part. The other two incident branchs correspond to the *host* part, from which the protruding part emerges.

For non-nested junctions, the presence of *three* full-ligature segments is a strong cue that the junction does *not* correspond to a protrusion (Fig. 12f). That is, in this case there is no salient branch that can be labeled as a protruding part, so the junction is labeled as a *Y*-junction.

For nested junctions (i.e., those connected to other junctions solely by ligature points), the labeling is difficult. In particular, where there are protrusions, it is difficult to determine the branches that correspond to the host and protruding parts of the shape. We therefore label such junctions as *nested*, and postpone their analysis until the next step of the algorithm.

What remains is the labeling of non-nested junctions whose incident ligature segments only contain semi-ligature points. This case is challenging because there is only one boundary concavity, and not all concavities are formed by protrusions (e.g., Fig. 12c–e). In these cases, to evaluate the hypothesis of a P-junction, in addition to ligature properties, we also evaluate both the relative thickness of the candidate protruding part and the curvature of the medial curve produced by merging the host branches (e.g., see Section 3.4.3). 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. 13). 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. If the junction meets the thickness condition and the merged (host) medial curve does not contain points of high curvature (i.e., above a given threshold), then it is deemed to be a P-junction. Otherwise it is a Y-junction.

3.4.2. Nested protrusions

The ligature/junction configurations can be quite complex when nested. When two junctions are nested 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. 15a). 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.

The identification of host and protruding branches associated with a P-junction is relevant for the analysis of nested protrusions. In particular, we are interested in the *boundary gap* defined by the ligature segments associated with a P-junction. 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 Fig. 14.

Two nested protrusions can create boundary gaps on the same side, or on opposite sides, of a host branch. If the boundary gaps are

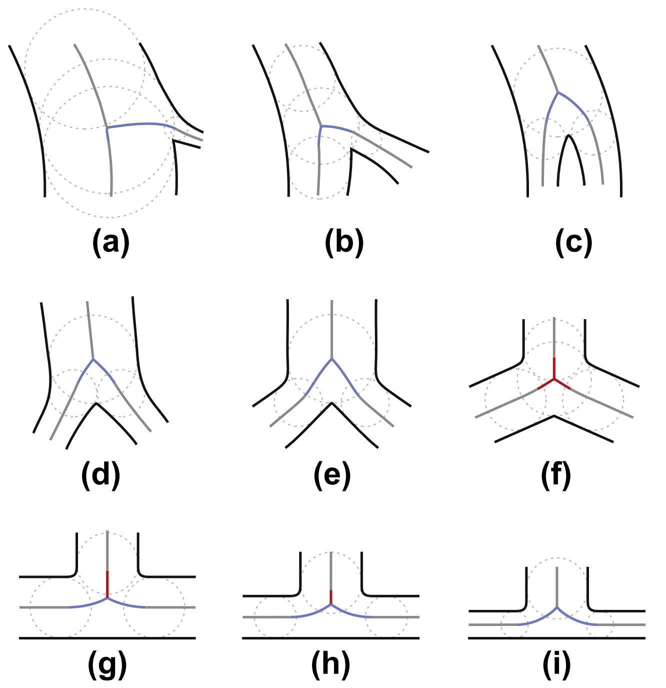


Fig. 12. Example of the evolution of a junction and the limiting cases of its ligature configuration. A P-junction (a) can be transformed into a Y-junction (c) without changing the flow and ligature pattern of the junction. The interpretation of limiting cases, such as (b), as Y- or P-junctions is sensitive 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 section, 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.

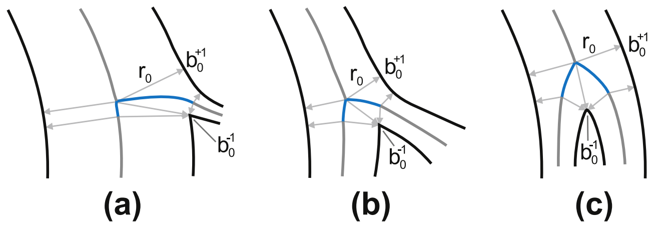


Fig. 13. Example of Y- and P-junctions with similar ligature properties. In each example the junction point is associated with one concave corner, b_0^{-1}, which induces two semi-ligature segments (blue points). In order to label the junction, the relative thickness of the protruding and host parts 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

on the same side of the branch, one of them must be fully contained within the other (Fig. 15a), or they must be adjacent (Fig. 15b), because medial axis spokes do not intersect [29] (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 the following conditions:

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. 15 a);

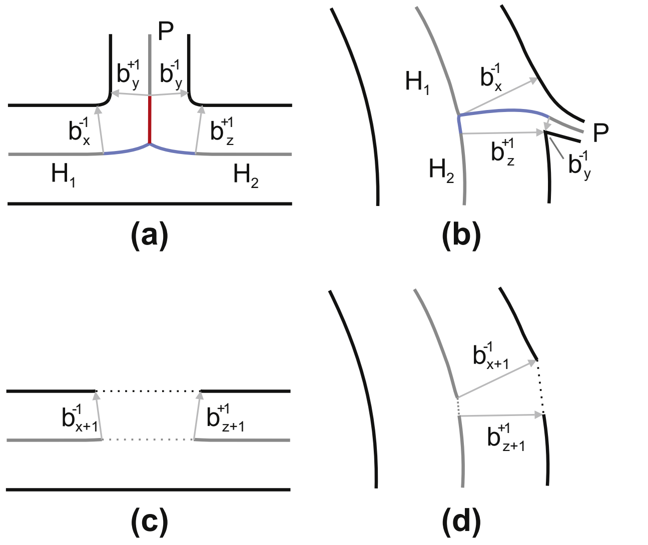


Fig. 14. Examples of boundary gaps induced by protrusions. A *boundary gap* is created by removing the boundary points that form the concave corner(s) associated with a protrusion and the boundary points mapped by the spokes of the protruding branch. The corner points are given by the interval of connected 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 $\left[b_x^{-1}, b_y^{+1}\right]$ and $\left[b_y^{-1}, b_z^{+1}\right]$, while the corner of the protrusion in (b) is given by the interval $\left[b_y^{-1}, b_z^{+1}\right]$. 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}^{+1} in (c) and (d). Then, the boundary gap interval is given, clockwise, by $\left(b_{x+1}^{-1}, b_{z+1}^{+1}\right)$. 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 3.4.3.

2. the ligature segments incident on the junctions map to concave corners on the same skeletal side of the host shape part and define adjacent boundary gaps (see Fig. 15b);
3. the ligature segments incident on the junctions map to concave corners on opposite sides of the host shape part (see Fig. 15c).

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. 19a–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 provide useful information about 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 Fig. 16b and e). This configuration may also be labeled as two P-junctions or two Y-junctions (see Fig. 16c and f).

The labeling of configuration (3) depends on the perceptual preferences that are appropriate for the domain. For example, Fig. 16a and d show two similar shapes and skeletons that differ only in the thickness of the skeletal branches. This 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 Section 3.4.1 to label the junctions as either P or Y, and leave the possible interpretation as overlapping medial axes for future work.

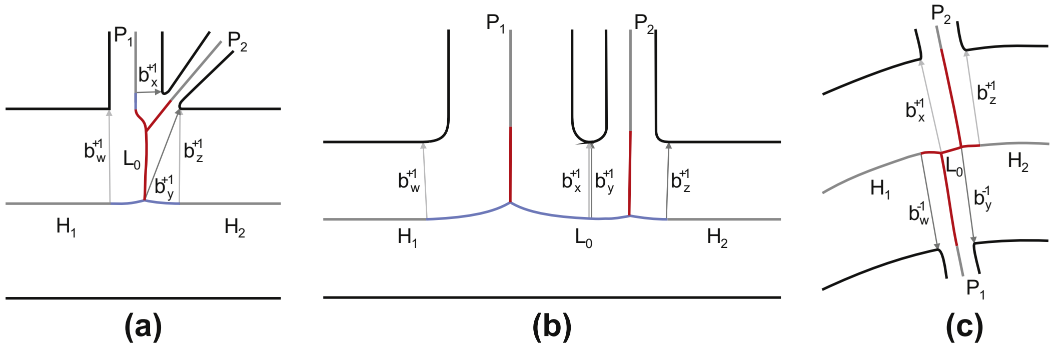


Fig. 15. Nested protrusions. A pair of junctions is said to be nested if they are connected by ligature points (L_0 branches in (a–c)). The type of nesting configurations between junctions is determined by the relative location of the boundary gaps defined by each protrusion (see Fig. 14 for details 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 $\left[b_x^{+1}, b_y^{+1}\right]$ is contained within the parent gap $\left[b_w^{+1}, b_z^{+1}\right]$; (b) the boundary gaps are adjacent and connected by ligature points with opposite flow direction. This is the case with gaps $\left[b_w^{+1}, b_x^{+1}\right]$ and $\left[b_y^{+1}, b_z^{+1}\right]$ 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 $\left[b_x^{+1}, b_z^{+1}\right]$ and $\left[b_w^{-1}, b_y^{-1}\right]$ are located on opposite sides of the host branches H_1 and H_2.

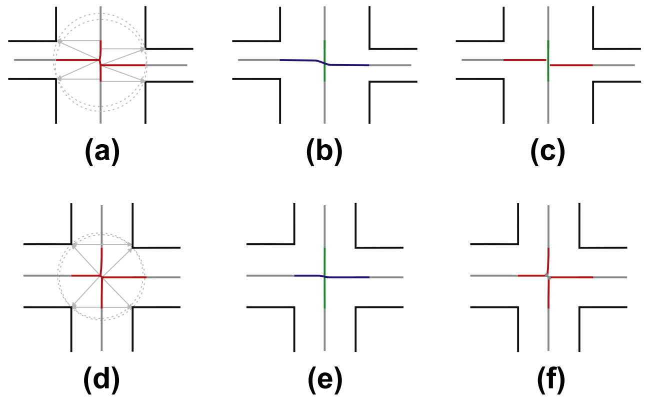


Fig. 16. Interpretations of nested protrusions. The ligature configuration in (a) can be naturally interpreted as two (imaginary) overlapping medial axes (b), or as two protrusions on the same medial axis (c). Similarly, the ligature configuration in (d), can be interpreted as two overlapping medial axes (e), or, given the comparable width of all the branches, as the attachment of four parts to a center part (f).

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 create a dependency graph in which every node represents a nested junction, 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 v is a protrusion whose host branches should be merged before those of junction 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 independent (i.e., we do not add a dependency edge between them). If the resulting graph contains cycles (a seldom occurring but possible event), we break them by removing an arbitrary edge from each cycle. Finally, given the junction dependency graph of a skeleton (see Fig. 17), we merge the branches of each P-junction node with no dependencies (i.e., with zero out-degree), detect the ligature points in the merged branches, and relabel the junction dependent on them. Next, we remove all nodes with zero out-degree from the graph, and process the graph recursively until it is empty.

3.4.3. 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. 18a).

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 Eq. (1), which relates the object angle to the first derivative of the 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 Fig. 18. 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 boundary 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 matching, 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. 18c). 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 section.

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

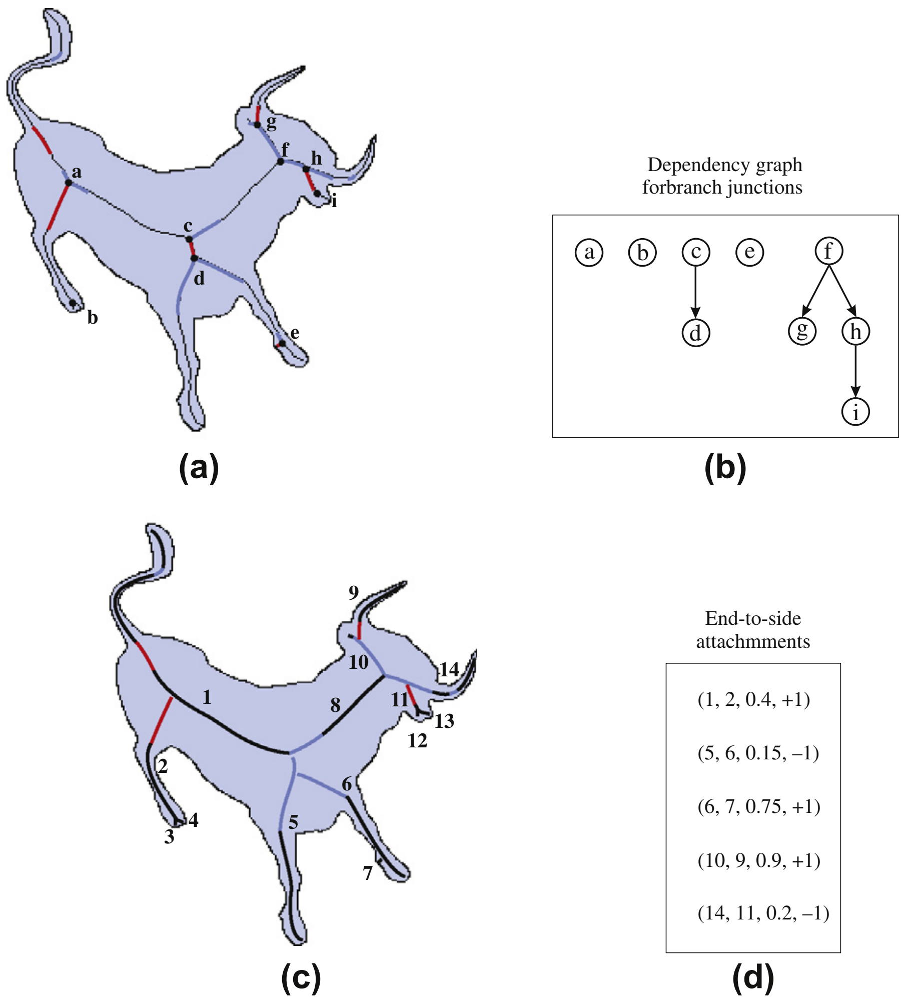


Fig. 17. The dependency graph of nested protrusions. (a) Every branch junction, a–i, is labeled as either Y, P, or nested. (b) The parent-child dependencies between nested junctions are represented by a dependency graph. Graph nodes with out-degree equal to zero (i.e., independent nodes) are processed 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.

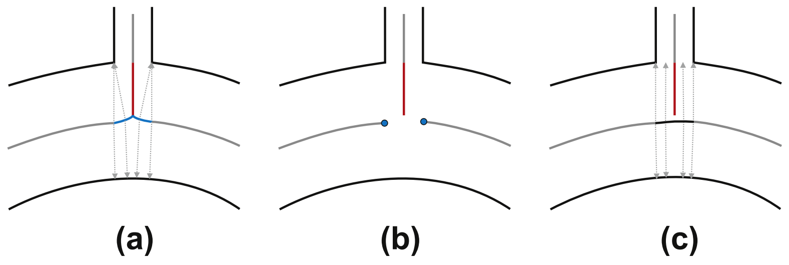


Fig. 18. The branch merging operation. (a) In this example, the semi-ligature "arms" of the junction define the skeletal gap points. (b) The position and radius of the gap endpoints are interpolated by smooth radius and axial functions while preserving tangent continuity. (c) The spokes of each interpolated point are computed from the relation between radius, medial curve and object angle given by Eq. (1).

Fig. 19. 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.

3.5. 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 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 [30]. 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 Fig. 20a–d). This partition function

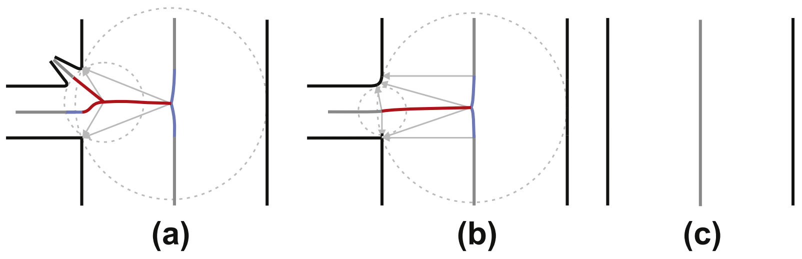


Fig. 19. Example of branch merges helping in the restoration of nested ligature. (a) Nested boundary gaps form nested ligature. (b) The child protrusion is restored first so that the inner boundary gap is filled and ligature properties can be recomputed. (c) The parent protrusion is not nested anymore and can be processed recursively.

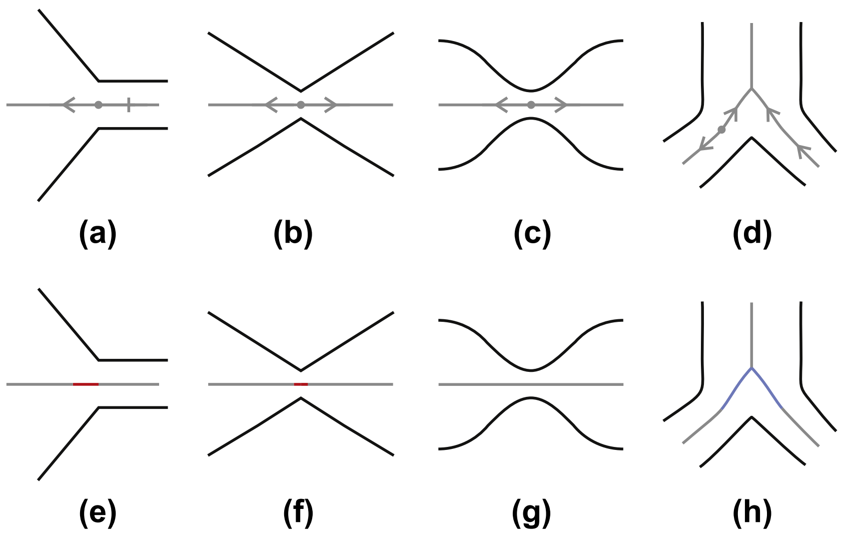


Fig. 20. Shock graph partitions and ligature-induced partitions. TOP ROW: Shock graph partitions. (a) A two-part decomposition induced by a segment with monotonically decreasing radii and a segment of constant radius. (b) Two parts induced by the segments with monotonically varying radii (a third part 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). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

does not account for concave corners explicitly, but is still able to identify shape parts induced by them (e.g., Fig. 20a and b), as well as other natural parts induced by smooth boundary deformations (e.g., Fig. 20c). 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 Fig. 20d 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 3.3. 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 Fig. 20e–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 sub-partitioning the bones. For example, the shock graph partition of Fig. 20c can be applied to the single bone of Fig. 20g in order to capture the perceptual relevance of the smooth neck.

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 Fig. 20e are joined by a ligament formed by a single ligature segment, while the non-ligature segments in Fig. 20f 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 under-segmentation and for discovering the attachment relationships that are not related to branch junctions.

3.6. 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
  $\mathscr{I}$: shape image
  $\mathscr{S}_0$: set of skeletal points
  $\mathscr{S}_1$: set of skeletal points with P-junctions removed
  $\mathscr{P}$: set of branch adjacency relations recovered from branch merges
  $\mathscr{L}$: set of ligature segments $[s_0,s_1]$, for $s_0, s_1 \in \mathscr{S}_1$
  $\mathscr{B}$: set of bone segments $[s_0,s_1]$, for $s_0, s_1 \in \mathscr{S}_1$
procedure $(\mathscr{B}, \mathscr{L}, \mathscr{P})$ = shapeParsing($\mathscr{I}$)
  $\mathscr{S}_0$ = computeSkeleton($\mathscr{I}$)
  $(\mathscr{S}_1, \mathscr{P})$ = detectProtrusions($\mathscr{S}_0$)
  $(\mathscr{B}, \mathscr{L})$ = partitionBranches($\mathscr{S}_1, \mathscr{P}$)
end
procedure $(\mathscr{S}_1, \mathscr{P})$ = detectProtrusions($\mathscr{S}_0$)
  $\mathscr{L}_0$ = analyzeLigatureAroundJunctions($\mathscr{S}_0$) ; see Section 3.3
  DG = createDependencyGraph($\mathscr{S}_0, \mathscr{L}_0$) ; see Section 3.4.2
  $\mathscr{P} = \emptyset$ ; let $\mathscr{P}$ be the empty set of end-to-side branch adjacency relations
  $\mathscr{S}_1 = \mathscr{S}_0$
  while DG not empty
    v = getIndependentNode(DG) ; any node (i.e., junction point) with out-degree equal to zero
    $\mathscr{L}_0$ = updateLigature($\mathscr{S}_1, \mathscr{L}_0$, v) ; recompute ligature for all branches incident on v
    type = labelJunction(v, $\mathscr{L}_0$) ; type is either 'Y' or 'P' (see Section 3.4)
    if type = 'P'
      $(\mathscr{S}_1, \mathscr{P})$ = mergeHostBranches(v, $\mathscr{S}_1, \mathscr{P}$) ; see Section 3.4.3
    removeNode(DG, v) ; removes the node and all its associated dependencies (edges)
end
procedure $(\mathscr{B}, \mathscr{L})$ = partitionBranches($\mathscr{S}_1$)
  $\mathscr{B} = \emptyset$ ; let $\mathscr{B}$ be the empty set of bones
  $\mathscr{L} = \emptyset$ ; let $\mathscr{L}$ be the empty set of ligaments
  for every branch b in $\mathscr{S}_1$
    $\mathscr{L}_0$ = findAllMaximalLigatureSegments(b) ; see Section 3.5
    let $\mathscr{B}_0$ be the complement set of $\mathscr{L}_0$ ; i.e., the set of non-ligature points in b
    $\mathscr{B} = \mathscr{B} \cup \mathscr{B}_0$ ; add the new bones to the set of all bones
    $\mathscr{L} = \mathscr{L} \cup \mathscr{L}_0$ ; add the new ligaments to the set of all ligaments
end
```

4. Bone graphs: medial abstraction for object recognition

The shape parsing approach presented in the previous section yields two types of skeletal parts and two types of adjacency relations between them (Fig. 21a). 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. 21) or *end-to-side* (e.g., parts 5 and 8 in Fig. 21). 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 section, 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 [23,30,27,12]. A part hierarchy is a powerful tool for simplifying the shape matching 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 [13], 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 Section 2.

In this section, 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. 21c). The bone graph is a parts-based abstraction of a shape whose boundary is a simple closed curve,[2] 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 [30], 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. 22). This allows the bone graph to be less sensitive to perturbations to a silhouette caused by viewpoint changes than the shock graph. In Section 5, 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.

4.1. 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 Fig. 21b). 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 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. 22). 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.

[2] In our experiments, we ensure that a shape's boundary is a closed curve by ignoring holes within the shape.

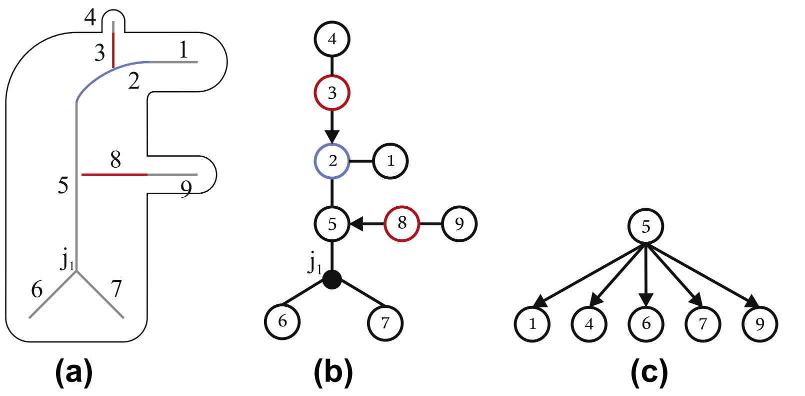


Fig. 21. Example output of the shape parsing algorithm and its bone graph representation. (a) The shape parsing algorithm presented in Section 3 yields a skeleton partitioned into bones (gray points) and ligaments (red and blue points), and the adjacency relations between them. (b) This output can be 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

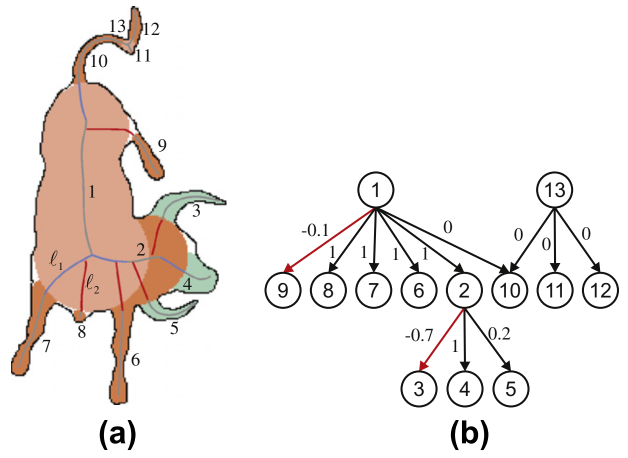


Fig. 22. Example of a bone graph. (a) The ligament-to-ligament attachments, such as that formed by ligaments ℓ_1 and ℓ_2, are expressed in (b) as edges between the parent node 1 and the two child nodes 7 and 8. The shape areas associated with each bone are colored differently for each level of the graph, 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

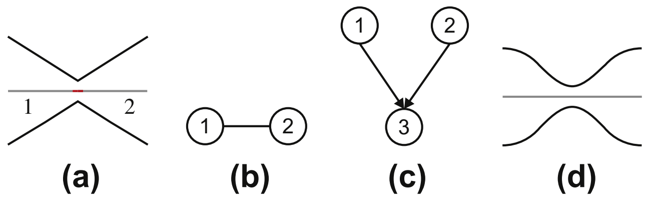


Fig. 23. The representation of neck shapes in the bone graph. (a) Example of a neck shape that induces a ligament connecting two bones. The radii of the ligament points decrease, from each endpoint, toward the ligature point closest to the concave corners. In this case, there is no clear hierarchical 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 3.5, 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.

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. 23), 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 [30]. We take the latter approach, as it allows us to simplify the matching problem by focusing on directed graphs. Then, let $\mathscr{E}$ be the set of directed edges. An ordered bone relation $(i,j) \in \mathscr{E}$, 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. 21a).
- 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

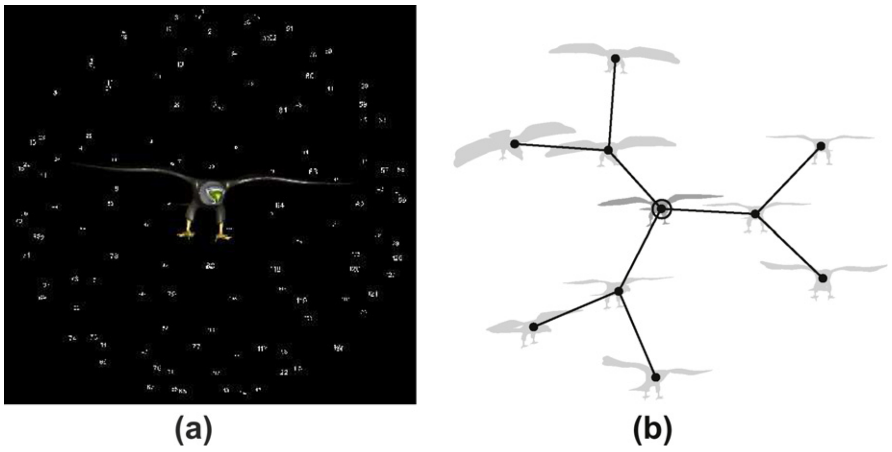


Fig. 24. The structure of the viewsphere. (a) Configuration of the nine closest neighbors of the query view (center) on the viewsphere; (b) one of the query's neighbors, as seen on the 3-D viewsphere.

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 [4].

We can now specify the bone graph: the bone graph of a 2-D shape bounded by a simple closed curve, $\mathbf{BG}(\Omega)$, 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 Section 3;
- *edges* $(i,j) \in E \subseteq V \times V$ directed from node i to node j iff $(i,j) \in \mathscr{E}$;
- *node attributes* $\lambda : V \mapsto \mathscr{S}$, where $\mathscr{S}$ 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 $L_i \subseteq \mathscr{S}$; and
- *edge attributes* γ: $E \mapsto [-1,1]$, 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 Section 3.1). This information can be used at matching time to compare the geometrical properties of two shape parts encoded by two nodes.

5. Evaluation

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 [30], 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.

[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.

We begin with a dataset of 1664 silhouette views of 13 3-D models (Fig. 25) with 128 uniformly spaced views per object around its viewsphere (Fig. 24a), 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 one of the nine closest neighbors (Fig. 24b) 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 form 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.

Fig. 26 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.

Fig. 27 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

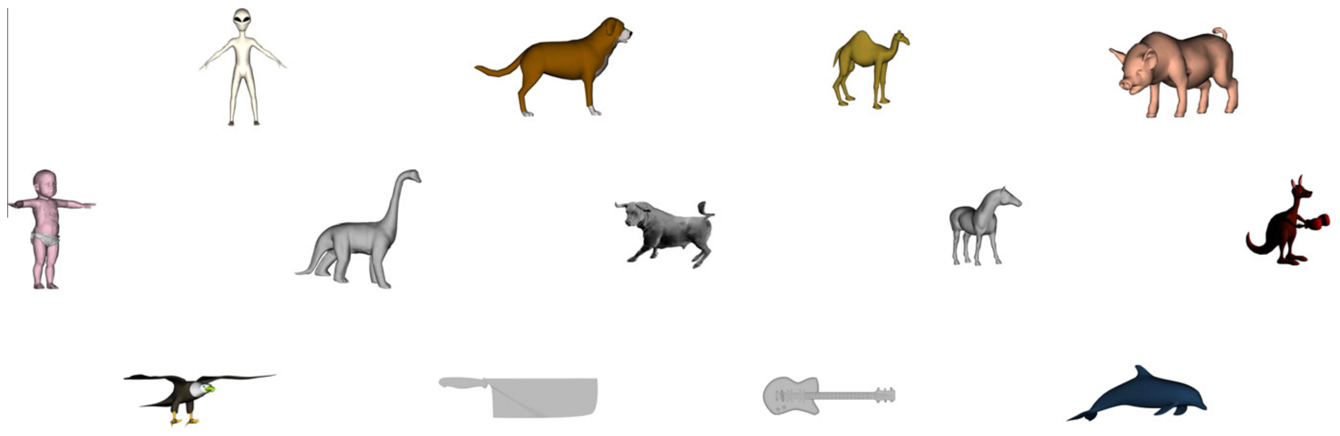


Fig. 25. The 13 models used in the experiments.

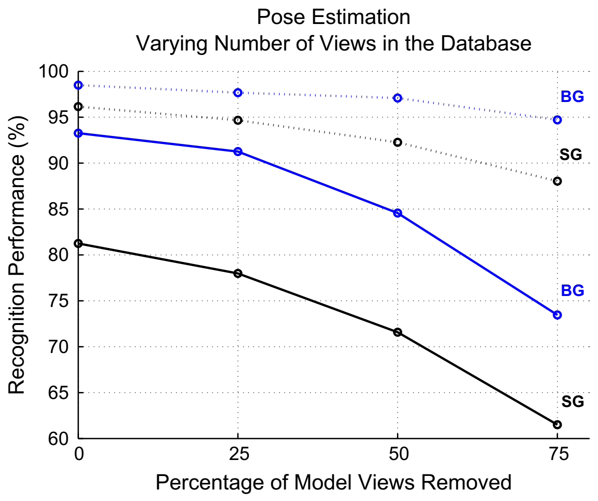


Fig. 26. Correct recognition (dashed curves) and pose estimation (solid curves) rates for bone graphs (blue, labeled BG) and shock graphs (black, labeled SG) as a function of decreasing viewsphere sampling resolution. The bone graph clearly exhibits superior stability for both tasks, with dramatic improvement for the more stringent pose estimation task. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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.

It is also interesting to analyze examples of unsuccessful matches. Fig. 28 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.

Finally, we provide a sense of how the framework performs under part articulation by matching a subset of the shapes in the Virtual Human Action Silhouette Data [25]. More specifically, we selected 24 silhouettes of six different subjects (two men, two women, one Viking and one humanoid) imaged from different viewpoints performing four different actions (punching, waking, kicking in the air, and landing after kicking) taken from different viewpoints. We matched all shapes against each other and selected six queries to show as examples. Fig. 29 depicts a table in which each row corresponds to a query and each column corresponds to the first, second, and third best matches, from left to right. The query is always on the left within each cell. These results show that the part decompositions of articulated silhouettes are indeed similar, and that the correspondences found by the matcher are correct for most most shapes. However, there are some incorrect correspondences, such as the arm-to-leg correspondence in row 5, column 1, in which the parts have similar contours but different attachment positions on their parents. In future work, we expect to exploit the edge attributes of the bone graph in order to add additional constraints to the selection of part correspondences. This would allow us to penalize the shape similarity measure when the correspondences define part relations with inconsistent edge attributes, such as those between arms and legs.

6. 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., [27,30]). 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.

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. Finally, 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

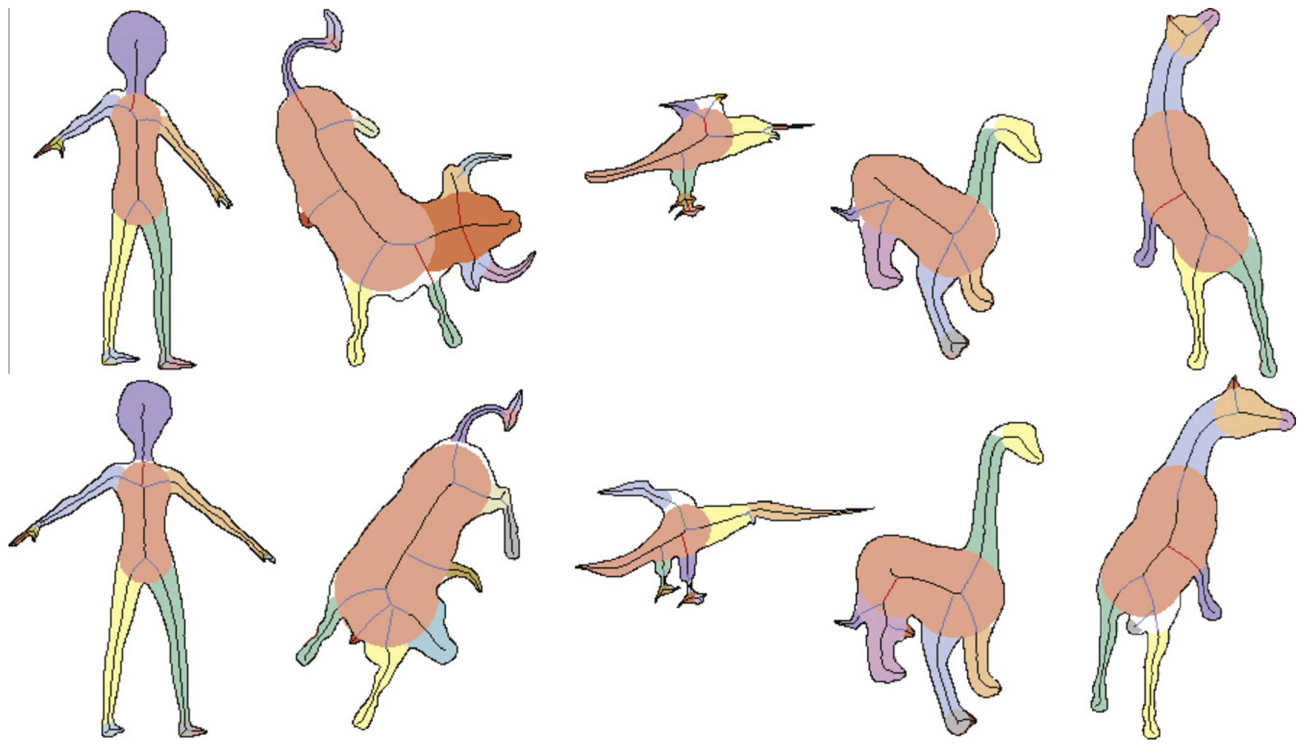


Fig. 27. Matching bone graphs. In each pair of shapes, the top shape represents the query while the bottom shape represents the closest matching database view. Each shape includes its final restored skeleton, along with the shaded bones defined by the non-ligature segments. Corresponding bones between 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

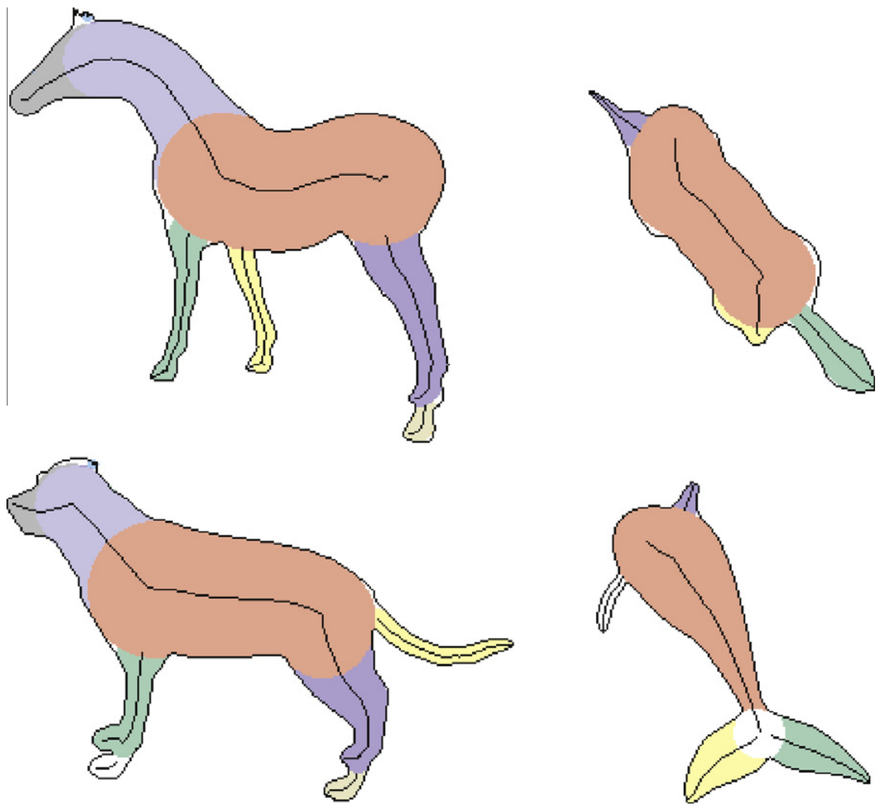


Fig. 28. Incorrect matching examples for bone graphs. Here we show two cases in which the most similar model views (bottom row) to the query views (top row) do not belong to the same object. LEFT: since edge attributes are ignored, the dog's tail is assigned to one of the horse's leg without a penalty 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.

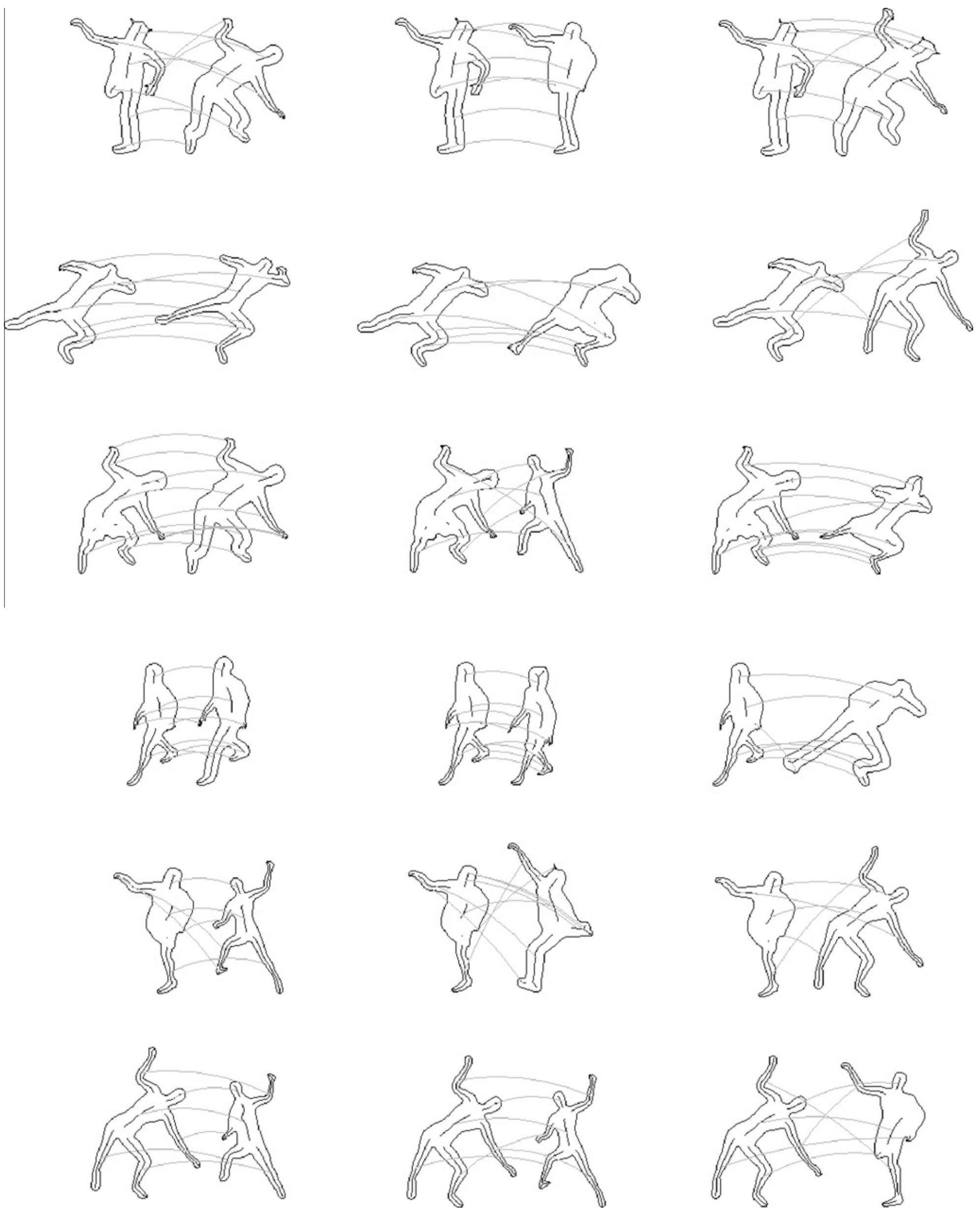


Fig. 29. Matching examples under part articulation and viewpoint changes. We show six queries and their three most similar models found within a set of 24 silhouettes of articulated objects (see text for details). Each row of the table corresponds to a query and each column corresponds to the first, second, and third best matches, from left to right. The query is shown on the left of each cell. These examples show that the articulated silhouettes of different actors have similar part decompositions, and that the correspondences assigned by the matching algorithm are mostly correct, even though edge attributes are not exploited to penalize for geometrically similar parts (e.g., arms and legs) that are attached at different positions along their parent part.

assembling the skeletal parts 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.

Acknowledgements

The authors would like to thank Allan Jepson and Ali Shokoufandeh for their thoughtful feedback and support over the course of this work. The authors would also like to gratefully acknowledge the support of OCE, NSERC, CIFAR, MRI, and DARPA.

References

[1] C. Aslan, S. Tari, An axis-based representation for recognition, in: International Conference on Computer Vision, 2005, pp. 1339–1346.
[2] J. August, K. Siddiqi, S. Zucker, Ligature instabilities in the perceptual organization of shape, Computer Vision and Image Understanding 76 (3) (1999) 231–243.
[3] Xiang Bai, Longin Jan Latecki, Wen-Yu Liu, Skeleton pruning by contour partitioning with discrete curve evolution, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (3) (2007) 449–462.
[4] I. Biederman, Human image understanding: recent research and a theory, Computer Vision, Graphics, and Image Processing 32 (1985) 29–73.
[5] T. Binford, Visual perception by computer, in: IEEE Conference on Systems and Control, Miami, FL, 1971.
[6] H. Blum, Biological shape and visual science, Journal of Theoretical Biology 38 (1973) 205–287.
[7] H. Blum, R.N. Nagel, Shape description using weighted symmetric axis features, Pattern Recognition 10 (1978) 167–180.
[8] Michael Brady, Haruo Asada, Smoothed local symmetries and their implementation, International Journal of Robotics Research 3 (3) (1984) 36–61.

[9] D. Chetverikov, Zs. Szabo, A simple and efficient algorithm for detection of high curvature points in planar curves, in: Workshop of the Austrian Pattern Recognition Group, 1999, pp. 175–184.
[10] F. Demirci, A. Shokoufandeh, S. Dickinson, Skeletal shape abstraction from examples, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (5) (2009) 944–952.
[11] F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, S. Dickinson, Object recognition as many-to-many feature matching, International Journal of Computer Vision 69 (2) (2006) 203–222.
[12] F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, S. Dickinson, Object recognition as many-to-many feature matching, International Journal of Computer Vision 69 (2) (2006) 203–222.
[13] M.A. Eshera, K.S. Fu, A similarity measure between attributed relational graphs for image analysis, in: International Conference on Pattern Recognition, 1984, pp. 75–77.
[14] J. Feldman, M. Singh, Bayesian estimation of the shape skeleton, Proceedings of the National Academy of Sciences 103 (2006) 18014–18019.
[15] Peter J. Giblin, Benjamin B. Kimia, On the local form and transitions of symmetry sets, medial axes, and shocks, International Journal of Computer Vision 54 (1-3) (2003) 143–156.
[16] M.A. Grayson, Shortening embedded curves, Annals of Mathematics 129 (1989) 71–111.
[17] Ralf Juengling, Lakshman Prasad, Parsing silhouettes without boundary curvature, in: International Conference on Image Analysis and Processing, Modena, Italy, 2007, pp. 665–670.
[18] Robert A. Katz, Stephen M. Pizer, Untangling the Blum medial axis transform, International Journal of Computer Vision 55 (2–3) (2003) 139–153.
[19] Benjamin B. Kimia, On the role of medial geometry in human vision, Journal of Physiology 97 (2–3) (2003) 155–190.
[20] Diego Macrini, Bone Graphs: Medial Abstraction for Shape Parsing and Object Recognition, PhD thesis, University of Toronto, 2010.
[21] Xiaofeng Mi, Doug DeCarlo, Separating parts from 2d shapes using relatability, in: International Conference on Computer Vision, 2007.
[22] Xiaofeng Mi, Doug DeCarlo, Matthew Stone, Abstraction of 2d shapes in terms of parts, in: International Symposium on Non-Photorealistic Animation and Rendering, 2009.
[23] S.M. Pizer, W.R. Oliver, S.H. Bloomberg, Hierarchical shape description via the multiresolution symmetric axis transform, IEEE Transactions on Pattern Analysis and Machine Intelligence 9 (4) (1987) 505–511.
[24] S.M. Pizer, K. Siddiqi, G. Székely, J.N. Damon, S.W. Zucker, Multiscale medial loci and their properties, International Journal of Computer Vision 55 (2–3) (2003) 155–179.
[25] H. Ragheb, S. Velastin, P. Remagnino, T. Ellis, Vihasi: virtual human action silhouette data for the performance evaluation of silhouette-based action recognition methods, in: Workshop on Activity Monitoring by Multi-Camera Surveillance Systems, Stanford University, California, 2008.
[26] H. Rom, G. Medioni, Hierarchical decomposition and axial shape description, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (10) (1993) 973–981.
[27] T. Sebastian, P.N. Klein, B. Kimia, Recognition of shapes by editing their shock graphs, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (5) (2004) 550–571.
[28] Doron Shaked, Alfred M. Bruckstein, Pruning medial axes, Computer Vision and Image Understanding 69 (2) (1998) 156–169.
[29] K. Siddiqi, S.M. Pizer, Medial Representations: Mathematics, Algorithms and Applications, Springer, 2008.
[30] K. Siddiqi, A. Shokoufandeh, Sven J. Dickinson, Steven W. Zucker, Shock graphs and shape matching, International Journal of Computer Vision 35 (1) (1999) 13–32.
[31] M. Singh, G. Seyranian, D. Hoffman, Parsing silhouettes: the short-cut rule, Perception and Psychophysics 61 (4) (1999) 636–660.
[32] S. Tari, J. Shah, H. Pien, Extraction of shape skeletons from grayscale images, Computer Vision and Image Understanding 66 (2) (1997) 133–146.
[33] Hüseyin Tek, Benjamin B. Kimia, Boundary smoothing via symmetry transforms, Journal of Mathematical Imaging and Vision 14 (3) (2001) 211–223.
[34] A. Telea, C. Sminchisescu, S. Dickinson, Optimal inference for hierarchical skeleton abstraction, in: International Conference on Pattern Recognition, Cambridge, UK, August 2004, pp. 19–22.
[35] M. van Eede, D. Macrini, A. Telea, C. Sminchisescu, S. Dickinson, Canonical skeletons for shape matching, in: International Conference on Pattern Recognition, Hong Kong, August 2006.
[36] Song Chun Zhu, A.L. Yuille, Forms: a flexible object recognition and modeling system, International Journal of Computer Vision 20 (3) (1996) 187–212.