Recognition of Shapes by Editing their Shock Graphs

Thomas B. Sebastian Philip N. Klein Benjamin B. Kimia*
Brown University, Providence RI 02912

Abstract

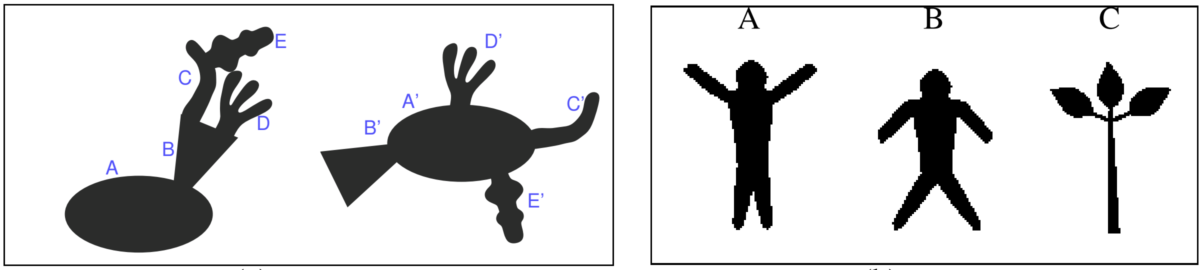
This paper presents a novel framework for the recognition of objects based on their silhouettes. The main idea is to measure the distance between two shapes as the minimum extent of deformation necessary for one shape to match another. Since the space of deformations is very high-dimensional, three steps are taken to make the search practical: (i) *define an equivalence class for shapes based on shock-graph topology;* (ii) *define an equivalence class for deformation paths based on shock-graph transitions; and* (iii) *avoid complexity-increasing deformation paths by moving towards shock-graph degeneracy. Despite these steps, which tremendously reduce the search requirement, there still remain numerous paths to consider. To that end, we employ an edit-distance algorithm for shock graphs which finds the optimal deformation path in polynomial time. The proposed approach gives intuitive correspondences for a variety of shapes, and is robust in the presence of a wide range of visual transformations. The recognition rates on two distinct databases of 99 and 216 shapes each indicate highly successful within category matches (*100% *in top three matches), which render the framework potentially usable in a range of shape-based recognition applications.*

1 Introduction

The recognition of objects in images is one of the central problems in computer vision. It is a challenging task mainly due to the *large degree of variations* which projections of objects in 2D images experience as a result of within-class variations, articulations and other object-based changes, as well as variations in position, viewing direction, illumination, *etc.* The extent by which representations of object shape capture such variations can significantly impact the effectiveness of recognition. In recognition applications, shapes have been represented as point sets [3, 17], outline curves [2, 10, 37, 34, 13], and by their medial axis [26, 32, 40, 43, 56, 51], among others.

When a shape is represented using a point set (unorganized cloud of points) [18], matching is typically done using an assignment algorithm [17]. The methods that use an assignment algorithm for

*Corresponding Author.



(a) (b)

Figure 1: (a) Shapes with matching sub-parts. An approach that is based on similarity of parts but which does not capture the arrangement of features, will have negligible cost in matching these two shapes, since the cost of matching pairs (A, A'), (B, B'), (C, C'), (D, D') and (E, E') is zero. (b) The shape B results from an articulation of parts in shape A. This similarity is not captured by shape context [3] since articulation (disproportionately) affects the global signature (shape context) of each point. In addition, changes in the geometry of the parts which do affect the perception of shape significantly do not cause a large change in the shape context.

matching have the advantage of not requiring *ordered* boundary points. However, if the similarity of two points is based on a local measure, the matching process does not necessarily capture the coherence of shapes in that the relationship among portions of shape may not be fully captured in the match, Figure 1a. Note, however, that the use of a global shape context where the relative location of all points is captured in a summary form alleviates this drawback to some extent [3]. Further, since these methods typically rely on an extrinsic reference frame [3, 9], they are not invariant to global rotations and are sensitive to partial occlusion and object-based changes like articulation, Figure 1b. Finally, an alteration of fine shape geometry of parts may not significantly affect the overall distance, Figure 1b.

In curve-based representations, the matching is typically based on either aligning ordered point sets by an optimal similarity transformation [1], or by finding a mapping from one curve to another that minimizes an "elastic" performance functional, penalizing the "stretching" and "bending" energies of deformation [2, 10, 37, 34]. The minimization problem in the discrete domain is often transformed into one of matching shape signatures with curvature, bending angle, or absolute orientation as attributes [13, 37]. The curve-based matching methods often suffer from one or more of the following drawbacks: asymmetric treatment of the two curves, sensitivity to sampling, lack of rotation and scaling invariance, and sensitivity to articulations, placement and deformations of parts.

Shapes have also been represented by their medial axis and variants. Zhu and Yuille match skeletal graphs of animate shapes using two deformable primitives, a tubular structure referred to as "worm" and a "circle" [56]. The similarity between instances of primitives is then defined in terms of the principal deformation modes in a Bayesian framework. The inherent instabilities of this skeletal representation are accounted for by defining skeleton operators and by using a user-specified model graph as an intermediary in the match process. This approach is shown to be effective under articulation, viewpoint variation, and occlusion. However, (i) its applicability to inanimate objects is limited due to the choice

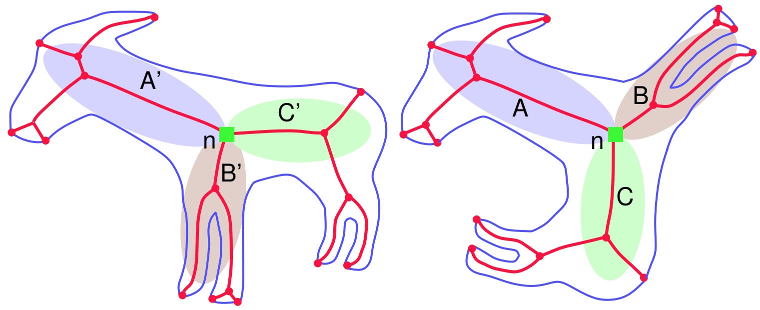


Figure 2: This figure illustrates the importance of capturing the *planar order* at nodes of a skeletal graphs in the match process. The shape on the left is reconstructed on the right after the ordering of branches (A', B', C') has been changed to (A, C, B). An approach that does not explicitly represent order at graph nodes cannot distinguish between the above shapes since there is a zero-cost match between portions (highlighted by ellipses for purposes of illustration) of the two shapes (A, A'), (B, B') and (C, C'). However, the two shapes are clearly different and should have a non-trivial match cost between them.

of primitives used; (ii) it difficult to extend the number of primitives used, as the matching implicitly assumes that the edges in the graph correspond to the worm and the nodes correspond to the circle; (iii) the proposed skeleton-operators are graph-based and are not motivated by changes in the underlying shape. This means that the costs of these operators, which are not independent of one another, may not be consistent with one another. For example, note that deleting an edge in the graph can be viewed as the limit of matching the edge to a zero-length edge.

In a related approach, Liu and Geiger [26] use the A* algorithm to match shape-axis trees, which are defined by the locus of midpoints of optimally corresponding boundary points. The cost of matching branches in shape-axis trees are computed in terms of the cost of matching corresponding boundary curves. As in [56], graph topology changing operations are used to deal with transformations like articulation and occlusion. However, the cost of the match can have discrete jumps because the cost of the topology-changing operations are not defined as limiting cases of deforming the corresponding edges. Further, their algorithm uses un-ordered trees, and does not necessarily preserve the ordering of edges at nodes, in the final match, Figure 2. In addition, the applicability of this method to large datasets of shapes is yet to be established. Another approach [51] with similar drawbacks models the topologically classified Euclidean skeleton as an attributed relational graph (ARG) and matches them using graduated assignment [17].

A representation derived from viewing the medial axis as singularities formed during a propagation from boundaries, $i.e.$, Blum's grassfire, is the *shock tree* or *shock graph*. In this approach isolated points like branch points, end-points, and those medial axis segments (links) where shocks are monotonically flowing give rise to a tree or a graph. The approach in this paper [38] uses a graph structure where shocks with isolated point topology (See Section 2 for details) are nodes, and those with curve topology

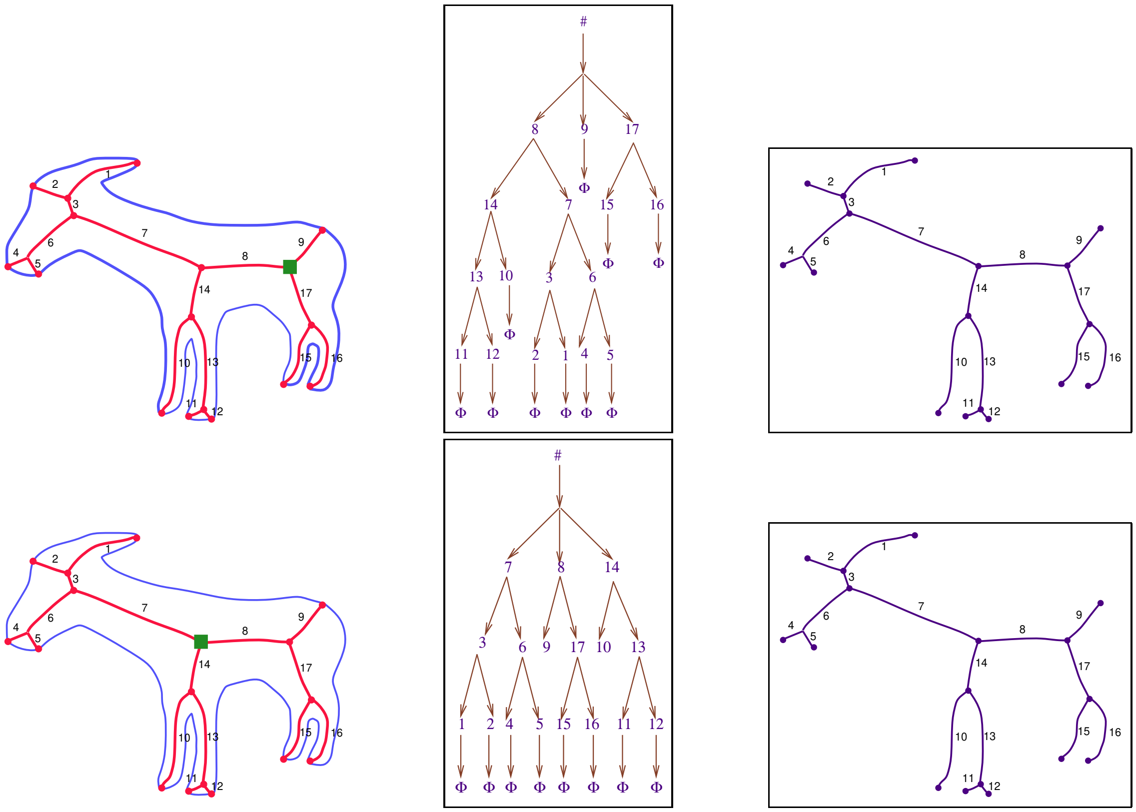


Figure 3: This figure compares the shock graph representations of Siddiqi *et al.* [43, 32] and Kimia *et al.* [40, 20]. The original shapes and shocks are shown in the left. The oldest shock is indicated by a square (green). Shock graphs of Siddiqi *et al.* and Kimia *et al.* are shown in the middle and right columns respectively. Note that a relatively small change in the shape causes the oldest shock to change, leading to a significant change in the topology of Siddiqi's shock representation, which may result in erroneous matches.

are links; we refer to this as the *edit-distance shock graph*. This representation was used in a similar form in Sharvit *et al.* [40, 20]. In an alternate approach [43, 32] the shock hierarchy is obtained by selecting the oldest shock as the root of a tree and then mapping connected shocks of the same type as nodes. Edges in this tree are placed between adjacent shock types such that the parent node has a higher time of formation than all the child nodes. We refer to this as the *Siddiqi et al.'s rooted shock tree* to facilitate a discussion on the two types of shock representations. While the Siddiqi et al.'s rooted shock tree represents the shock hierarchy well, this representation maps shock segments with directed flow to a point (node), where the dynamic information is implicit. For example, the planar order at nodes is not represented, Figure 2. In addition, the selection of the oldest shock as the root of the tree, leads to an instability in the shock-tree representation, as relatively small change in the shapes can lead to a shock tree of very different topology. This in turn leads to erroneous matches, Figure 3, unless explicitly accounted for by additional measures. We now present the various shape matching methods which take advantage of these shock-based representations.

Sharvit *et al.* [40, 20] considered the pairwise assignment of shock graph nodes and defined similar-

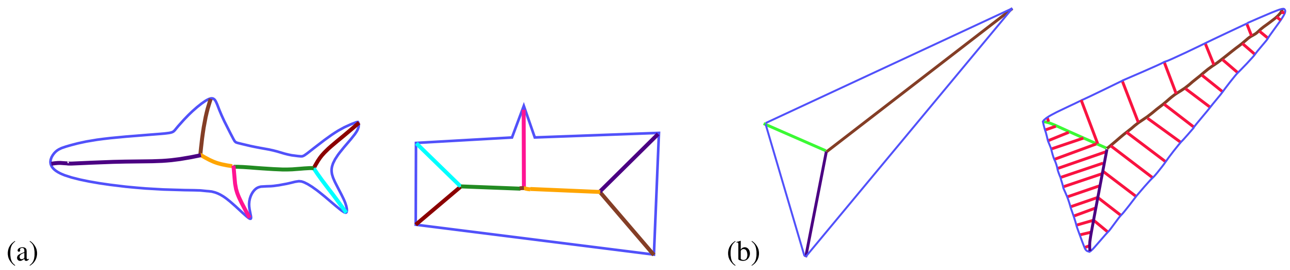


Figure 4: This figure illustrates a drawback of matching the graph topology, $e.g.$, [8, 53] without using geometrical information. Shapes that are very dissimilar can have the same shock topology (a) and vice versa (b).

ity for each pair of nodes based on the geometry of the corresponding shock segments linking them. The optimal assignment is found by minimizing the overall cumulative similarity using a graduated assignment approach [17]. This approach is fast, handles missing/extra nodes using slack variables, and gives fairly good matches for a rather small database of 25 shapes. However, the errors in the matching process also point out a fundamental flaw in that the coherence of a shape is not necessarily preserved in the match process, as illustrated in Figures 1a and 2.

The Siddiqi et al.'s rooted shock tree representation has been used in conjunction with several different schemes to index into shape databases. First, an approach based on subgraph isomorphism [43], characterizes the topology of a shock subtree by the sum of the eigenvalues of its adjacency matrix. In addition to this topological similarity, geometric similarity between nodes is measured as the Hausdorff distance between shock branches that are aligned by an optimal affine transformation. Figure 4 illustrates why geometrical similarity as well as topological similarity is required. These two similarity measures are combined in the tree matching, which starts at the roots of the trees and proceeds through the subtrees in a depth-first fashion. This matching strategy is thus sensitive to the choice of roots in the shock trees, Figure 3. In addition, the order at nodes is not preserved, which may lead to errors such as the one illustrated in Figure 2.

Another approach based on matching rooted shock trees relies on finding maximal cliques of an association graph [32]. The main idea behind this approach is that maximal subtree isomorphism between rooted trees induces a maximal clique in the corresponding tree association graph, which can be found as the global maximum of a quadratic function. The geometric similarity between nodes in the shock tree is defined in terms of differences in attributes of the shocks, namely, location, time of formation, orientation, and velocity. See Section 6.3 for a comparison. This approach allows only one-to-one and many-to-one correspondences, and is overly restrictive in cases where topological alterations are required on both trees, $e.g.$, nodes being deleted or added. As a remedy, the notion of many-to-many correspondences is introduced and computed using a similar association graph framework [33], where nodes with negligible weights are ignored in the match process. The matching of free trees where there

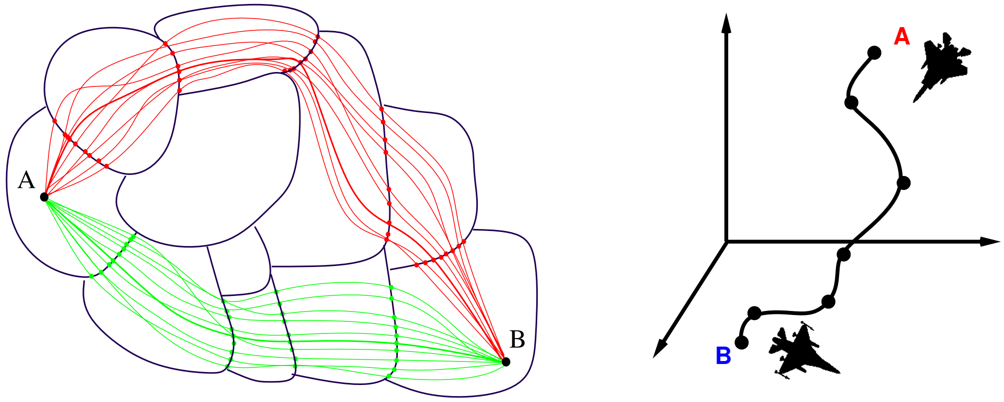


Figure 5: Every pair of shapes in the shape space is related by an infinite number of deformation paths, few of which are shown here. Each deformation path can be effectively characterized by the sequence of transitions (dots represent the transition from one shape cell to another) the shape experiences in the morphing process.

is no distinguished root node is also done based on the computational equivalence between the maximal clique of free tree association graphs and subgraph isomorphism of free trees [31].

In a third approach, Torsello *et al.* [49] compute approximate tree edit distance between shock trees, where the edit costs are restricted to be uniform. This approach relies on the computational equivalence between maximum common subgraph and tree edit distance assuming that the cost of relabeling an element is less than the cost of deleting it and re-inserting one with a different label [6]. The maximum common subgraph is then computed using maximal cliques of association graph and relaxation labeling [32]. The similarity of shock edges is measured in terms of the length of the corresponding boundary segments [50].

The above shock-based matching approaches [20, 40, 32, 43, 49] have been shown to perform well in indexing into small databases of shapes (less than 30 shapes), but their performance for larger databases has not been examined. We expect that for a recognition method to be effective for a much larger database, $i.e.$, one depicting an extensive range of visual transformations including occlusion, viewpoint variation, and articulation, the matching method should be able to handle the inherent *instabilities of the symmetry-based representations*, as classified in [15], Figure 9b. In other words, the matching strategy has to relate shapes on either side of an instability, which have different shock graph topologies but similar shape, with negligible matching cost.

We now present a brief overview of the proposed approach for comparing 2D shapes which addresses some of these issues. The main idea is to treat each shape as a point in a shape space and define the distance between two shapes in terms of the minimum-cost deformation path connecting them. Since there are infinitely many ways to deform a shape, the space of shapes and of deformations must be "discretized" to reduce the dimensionality of the search to practical limits. We address this issue by first defining an equivalence class on shapes, where all shapes with the same shock graph topology are equivalent. We then define as equivalent all the deformation paths which have the same set of

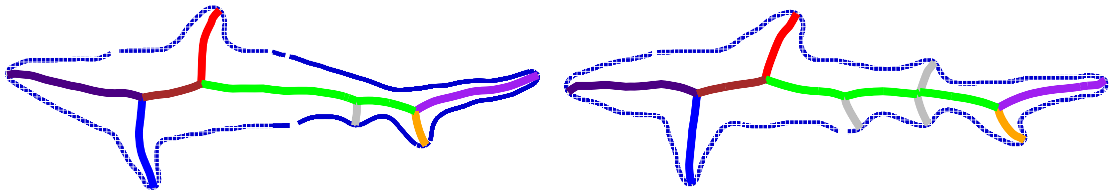


Figure 6: A preview of the correspondence between two fishes based on shock graph edit distance. Same colors indicate matching shock branches, and grey colored branches in the shock graphs have been pruned. This color scheme is used throughout the paper.

transition shapes (instabilities of the shock graph representation, that form the boundaries between shape equivalence classes), Figure 5. The representation of the deformation paths explicitly in terms of the transition shapes allows us to relate shapes on either side of the instability with negligible cost. We are able to recruit the shock transitions as a basic representation because they have been formally classified and enumerated [15], and can be effectively mapped to an "edit" operation on the shock graph in the discrete domain. A cost for each deformation path is defined by summing the costs of individual edit operations. Specifically, the cost of edit operations are computed in a consistent fashion by summing local shape differences, and by aligning the corresponding boundary curves. We employ a graph edit distance algorithm which was developed in [22, 23] to find the globally optimal path in polynomial time, Figure 6. The proposed approach is implemented and is shown to be robust in the presence of a variety of visual transformations such as articulation, partial occlusion, boundary perturbations, viewpoint variation *etc.*, and gives intuitive results for indexing into shape databases.

The main contributions of this paper are as follows: (i) develop and use a *planar ordered shock graph* representation where nodes and edges are distinct types of shocks as a basic representation for matching shapes; (ii) discretize the space of shapes and of deformation paths using shock transitions, which makes the problem of finding the optimal deformation path practical; (iii) assign costs for shock-graph edit operations such that they are consistent with each other; (iv) employ a previously developed edit distance algorithm to find the optimal solution of the discretized version of the problem; and (v) illustrate the effectiveness of the approach in the presence of various visual transformations, and for indexing into shape databases.

The paper is organized as follows. First, Section 2 reviews the basic concepts and develops the notion of a shock graph based on a formal classification of shock points, its instabilities under deformations, and its use in reconstructing the shape. In Section 3, the idea of discretizing the shape space based on defining equivalence classes on shapes and their deformations, and a restriction of deformation paths to pairs of simplifying paths is described. The edit distance algorithm for finding the optimal path is described in Section 4. Section 5 describes how the cost of each edit operation is assigned in a consistent fashion. Section 6 examines matching results under a variety of transformations, and shows indexing results for two databases, and finally compares the proposed approach to two recent methods,

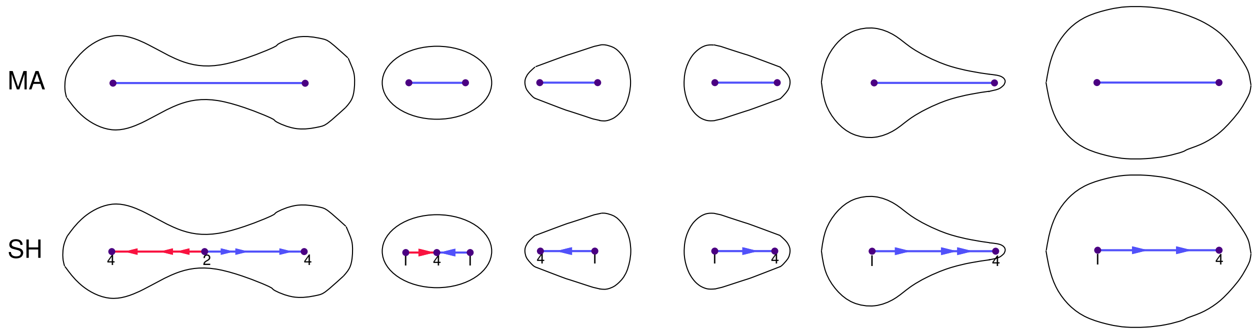


Figure 7: This figure from [15] illustrates that the dynamic interpretation of a medial axis as a flowing singularity (shock) gives rise to a more refined partitioning of the medial axis, which describes the qualitative aspects of the shape better.

Siddiqi et al rooted shock tree matching [32] and shape-context matching [3]. Section 7 concludes with a summary and discussion.

2 Shape Representation using Shock Graphs

Definition (Symmetry Set, Medial Axis, Shock Structure): The *symmetry set* (SS) of a shape is the locus of centers of bitangent circles [5]. The *medial axis* (MA) of a shape is defined as the locus of centers of maximal (bitangent) circles [4]. The *shock structure* (SH) of a shape arises from a "dynamic" interpretation of the medial axis, as the locus of singularities (shocks) formed in the course of wave propagation (as in Blum's grass-fire) from boundaries with an associated direction and speed of flow. *Shock segments* are curve segments of the medial axis with monotonic flow. Shock segments give a more refined partition of the medial axis segments [41]. These shock are obtained either by detecting the singularities of the evolving curve in a curve evolution/PDE approach [19, 42] or in an Eulerian-Lagrangian propagation which combines wave propagation and computational geometry concepts in a unified framework [48, 47, 46]. All shock graph used in this paper are from the latter approach.

Medial axis and shock points were formally classified in [15], based on the order of contact of circles with the shape boundary, as was originally used to classify the symmetry set [5]. The notation A_k^n refers to a circle with $k-$fold tangency at n distinct points, $e.g.$, A_1 means regular tangency, A_2 means an osculating circle, A_3 means tangency at a vertex, Figure 8. An implication of this classification together with a consideration of direction of flow [15] is that shock points generically consists of *sources*: (i) end-points corresponding to curvature extrema A_3, which always have inward flow, (ii) interior points with outward flow A_1^2-2 (second-order shocks), and (iii) junctions of three branches with only one outward flowing branch A_1^3-1, and of *sinks* (iv) interior points with inward flow A_1^2-4 (fourth-order shocks with two branches) and (v) junction of three branches all of which are inward-flowing A_1^3-4 (fourth-order shocks with three branches). All remaining points are A_1^2-1 (first-order shocks) connecting sources to sinks. The above five types of shocks are isolated from other shocks of the same type

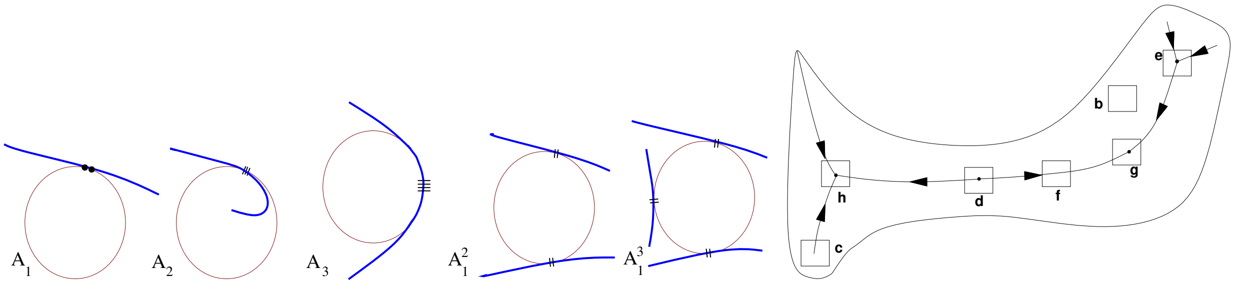


Figure 8: This figure from [15] and [47] illustrates the A_k^n notation in the context of the order of contact of the curves (thick blue) with circles (thin brown), which leads to a classification of generic possibilities for shock formation and propagation on the left, and in the context of a shape on the right: (b) no wave collision, (c) formation of an initial first-order shock, an A_3 point, (d) a second-order shock A_1^2-2, (e) a junction with an outgoing branch A_1^3-1, (f) monotonic propagation of a shockwave A_1^2-1 (first-order shocks) defining a shock segment, (g) termination of two branches at a fourth order shock A_1^2-4, (h) termination of three branches at a fourth order shock A_1^3-4.

and are naturally associated with nodes in a shock grass-fire, while the first-order shocks are contiguous and form links between the previous five types, resulting in a *directed, planar graph*, which is referred to as the *shock graph*.

Definition 1 *(Shock graph) Let the* A_1^2-2, A_1^2-4, A_3, A_1^3-1, *and* A_1^3-4 *types of shocks denote the* nodes *of a graph. Let the* A_1^2-1 *monotonic shock segments define the* links *of this graph. The attributes for each link are defined by the intrinsic geometry (curvature, acceleration, length, total time) and for each node by time of formation and direction of flow. We refer to this attributed relational graph as the shock graph.*

Note that the shock graph is a richer descriptor of shape than the medial axis graph, since its graph topology is more in accord with our perceptual notions of shape [41], Figure 7. Moreover, the shock graph of a simple, closed shape is an ordered unrooted planar tree. We now review the reconstruction of shape from a shock graph and the transitions of a shock graph, both of which will be used in later parts of the paper.

Reconstruction of Shape from a Shock Graph: Giblin and Kimia showed how a shape can be reconstructed from the shock graph [14]. For example, let $\gamma(s)$ denote a first-order A_1^2 shock curve, with arclength s, radius r (or time of formation), curvature κ, instantaneous velocity v, and acceleration a. The pair $(\kappa(s), a(s))$ and the length of the branch L are sufficient to intrinsically reconstruct the corresponding pair of boundaries $\gamma^{\pm}(s)$ of the shape and their differential properties [14]. Specifically, the boundary points $\gamma^{\pm}(s)$ are given by

$$\gamma^{\pm}(s) = \gamma(s) - \frac{r}{v}\vec{T} \pm \frac{r}{v}\sqrt{v^2 - 1}\vec{N}. \tag{1}$$

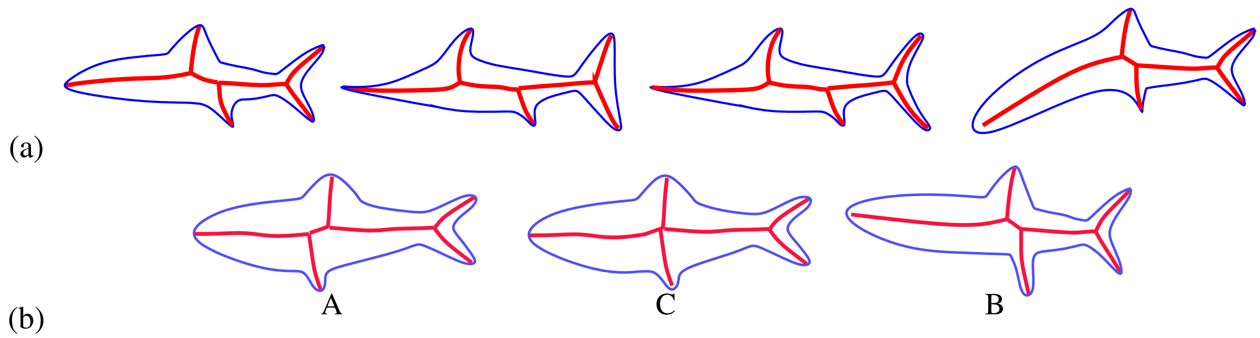


Figure 9: (a) A few examples of shapes having the same shock graph topology ($i.e.$, members of the same shape cell). (b) This figure illustrates one of the transitions (contract). A and B are the original shapes, and the C is the (degenerate) transition shape.

where $\vec{T}$ and $\vec{N}$ are the tangent and normal to the curve γ, respectively, where v and r are obtained by integrating a, and γ is obtained by integrating κ. The induced velocities $v^{\pm}$ and curvatures $\kappa^{\pm}$ of the shape boundary segments can be written as

$$\begin{cases} v^{\pm} = \mp \dfrac{\sqrt{v^2-1}}{v}(1 + r\bar{\kappa}^{\pm}) \\ \kappa^{\pm} = \dfrac{\bar{\kappa}^{\pm}}{1 + r\kappa^{\pm}}, \text{ where } \bar{\kappa}^{\pm} = \mp\kappa\dfrac{v}{\sqrt{v^2-1}} + \dfrac{a}{v(v^2-1)} \end{cases} . \quad (2)$$

This local intrinsic reconstruction is essential in computing the similarity between two shock segments, Section 5.

Transitions of the Shock Graph: The changes in the shock graph when the shape outline undergoes a one-parameter family of deformations has been studied [15]. In a majority of cases as a shape is deformed, the shock graph topology remains unaltered, Figure 9a. However, at certain points ($i.e.$, for certain shapes) an infinitesimal change in the shape can cause a large (abrupt) change in the shock graph topology, Figure 9b. The shapes where the graph topology is altered by a small perturbation are precisely the points of *instability* of the medial axis/shock graph. These instabilities, namely all generic *shock transitions* along a one-parameter family of deformations have been formally enumerated and classified [15], Figure 10. A similar situation exists in 3D, as recently studied in [16], where generic transitions are classified and enumerated in terms of the order of contact of spheres with the surface. In a related approach [30], structural changes that occur when an implicit surface undergoes a Gaussian diffusion process is studied.

In the next section we discuss how these transitions or instabilities are used to our advantage to partition the shape space into shape cells and to group deformation paths into groups of deformation bundles, in effect discretizing the search in the shape space for the optimal path between two shapes.

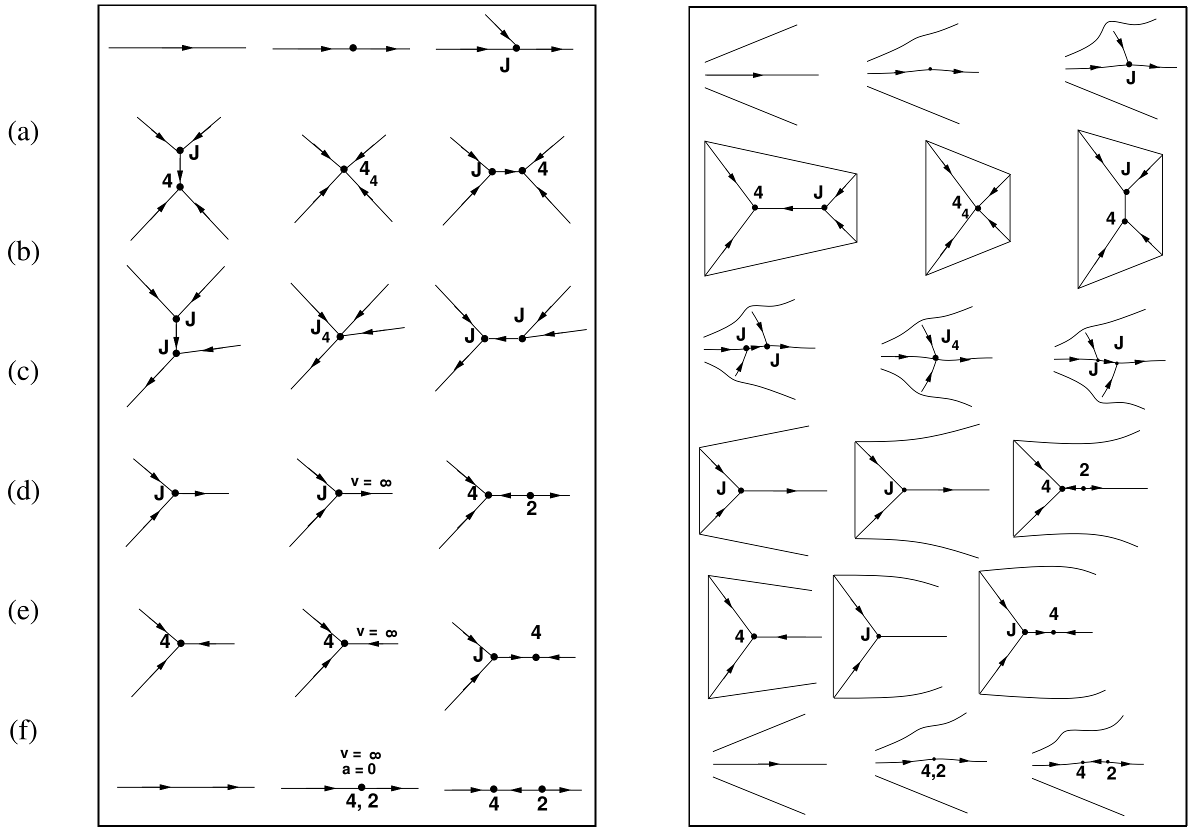


Figure 10: From [15]. The first three columns show a schematic description of the six possible transitions of the shock structure of a closed curve [15], and the last three columns show examples of corresponding shape deformations. The central column in each group represents the transition point in the deformation from left to right, or right to left columns. In the notation of [15] *(a)* is the A_1A_3 transition, *(b)* and *(c)* are the two types of A_1^4, *(d)* and *(e)* are the two types of A_1^3 with infinite velocity, and *(f)* is the A_1^2 point with infinite velocity and zero acceleration. The graph operations to make the right and left columns equivalent are: splice, contract (two types), and merge (three types).

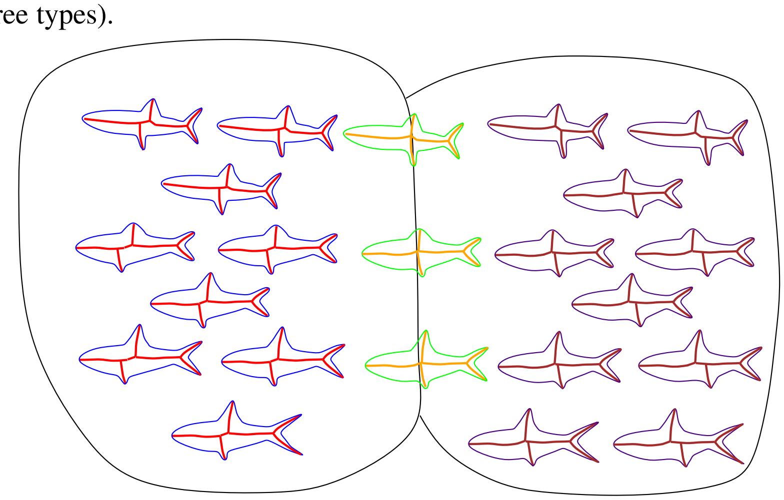
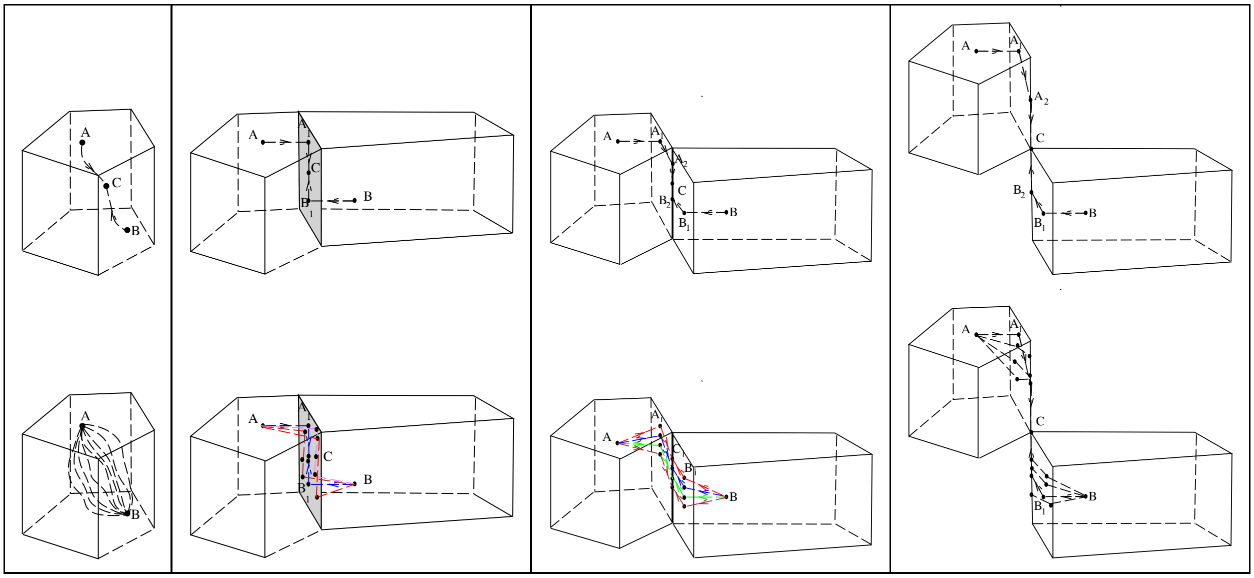


Figure 11: Two shape cells and the transition shapes that occupy the boundary between them.



(a) (b) (c) (d)

Figure 12: A highly schematic description of a *deformation bundle* by resorting to the analogy of connecting points embedded in polyhedral cells (a coarse analogy for shape cells). Two neighboring shape cells can meet at interfaces which upon arrival at the interface remove one, two, three or more degrees of freedom, as compared to that of the shape cell. Of course, the shape space is a much higher dimensional space than the 3D Euclidean polyhedral space. The path illustrates a projection of A and B to the transition interface at A_1 and B_1, respectively; from A_1 and B_1 to A_2 and B_2, *etc.* until both A_n and B_n belong to the same shape cell. Bottom row: each path is a representative of a collection of paths, or a deformation bundle which share the same sequence of transitions.

3 Partitioning the Shape Space

This section describes our approach to defining the distance between shapes. Each shape is viewed as a point in a *shape space*, the collection of all “shapes”. A deformation sequence between two shapes is a path in the shape space, Figure 5. We define the distance between two shapes A and B as the cost of the least-cost deformation path between them, and denote it by $D(A, B)$. However, there are infinitely many deformation paths between two shapes, and a key bottleneck in this deformation-based approach to matching shapes is the impracticality of the computational representation of this high-dimensional space of deformations. We propose three steps in “discretizing” this high-dimensional space such that the entire shape space is effectively and densely represented. First, we define an equivalence class on shapes based on shock graph topology. Specifically, we use the constancy of shock graph topology under deformations to define a local neighborhood for most shapes.

Definition 2 *A* shape cell *is a collection of shapes which have identical shock graph topology.*

At shock transitions, this neighborhood topology breaks down, $i.e.$, two shapes on either side of a transition shape are not connected using this notion of a neighborhood, while the perceptual distance

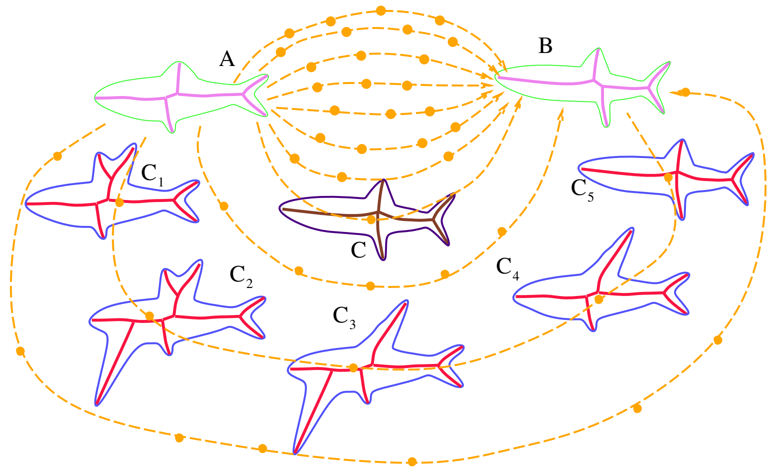


Figure 13: Two deformation paths among many are highlighted. The sequence (A, C_1, C_2, C_3, C_4,C_5, B) is not optimal since features are first added and then removed. In order to avoid such paths, we restrict each deformation path to those consisting of a pair of "simplifying" deformation paths from A and B leading to a common shape C.

between them is negligible, Figure 11. Note that this is primarily the reason why recognition based on a direct subgraph isomorphism of this representation breaks down. This motivates an explicit embedding of the transitions in the definition of a shape neighborhood as the "seams" of the shape space: Specifically, the remedy for arriving at a broader notion of a neighborhood requires discretizing the space of all deformation paths by considering all deformation paths that encounter the same set of transition shapes to be equivalent:

Definition 3 *Two one-parameter shape deformation sequences from shape A to shape B are equivalent if they pass through an identical sequence of shock transitions. A* shape deformation bundle *from shape A to shape B is the set of equivalent one-parameter families of deformations from A to B.*

In other words, two deformation paths are equivalent if they traverse through the same shape cells, Figure 5a. Further, we represent each deformation bundle using the set of transition shapes it passes through, Figure 5b.

We note further that among deformations paths, those where features are added and then removed are clearly not optimal, Figure 13. Hence, as a third measure, we avoid candidate deformation paths which unnecessarily venture into more complicated shapes. In order to avoid complexity-increasing deformation paths, we represent each deformation path by a pair of "simplifying" deformation paths from A and B to a "simpler" common shape C, Figure 14a. The notion of simplicity is derived from the transitions themselves, Figure 10. The deformation of a shape to the adjacent simpler transition shape restricts the shape in that a degree of freedom is removed, and leads to a shape cell of lower dimensionality, Figure 12: the first transition splices off branches, the second and third transitions "symmetrize" the shape, the fourth transition removes parts at "necks", the fifth transition moves the

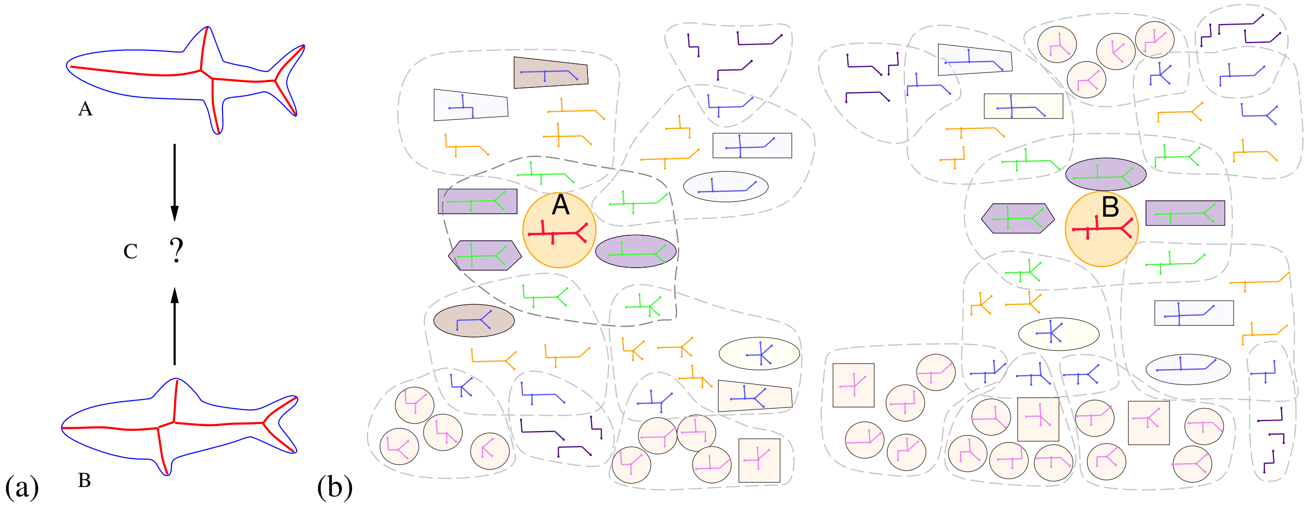


Figure 14: (a) The optimal path of deformation between two shapes is obtained by searching all pairs of simplifying deformation paths leading to a common shape C. (b) A hand-drawn sketch of how the space of one-parameter family of deformations for each shape can be discretized using transition shapes. Shape A the fish on the left (red) initially gives rise to seven transition shapes, each of which give rise to a finite number of other shapes enclosed in the dashed curves. A similar process applies to the second shape on the right. Common shapes with equivalent shock graph topology are marked by common icons with common shape/coloring. The optimal path in this case goes through the grey-hashed hexagonal surround.

shape to a rounder shape, and the last transition removes wavy patterns on the boundary. In the discrete domain, the deformation of a shape to the adjacent transition shape corresponds to an *edit operation* on the shock graph.

With the above discretizations the deformation paths between two shapes A and B can be represented by a finite number of transition shapes. For example, in Figure 14b seven transition shapes result from applying one simplifying transition on shape A (fish). In the second step of the transition sequence, each of these seven shapes undergoes a similar deformations towards the next applicable transition. In a few steps, the shock graph reaches highly simplified shapes, $e.g.$, elongated approximations of the object (only three steps are illustrated; additional steps are not shown). A complete path consists of a pair of simplifying deformation paths which reach two shapes in a common shape cell. These two deformed shapes are then related by a continuous transformation of the shock graph attributes using a "deform" edit operation, Figure 12.

In summary, the three steps of (i) partitioning of the shape space into *shape cells*, (ii) the definition of a *deformation bundle* as an equivalence class of deformations based on the sequence of transitions along each deformation, and (iii) the restriction of the search to paths consisting of two simplifying sub-paths from shapes A and B to a common shape C, constitute a discretization of the search space, and render the search of optimal deformation path practical.

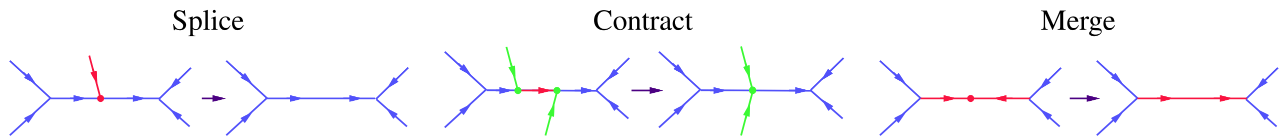


Figure 15: Three shock-graph edits that change the topology.

4 Edit-Distance Algorithm

Despite the above discretization and the tremendous reduction in dimensionality numerous paths remain to be considered, as illustrated in Figure 14. Thus we need an efficient algorithm to find the optimal path among all possibilities. We employ a polynomial-time edit-distance algorithm developed in [22, 23] for comparing ordered, unrooted planar trees, which we review in this section.

The notion of edit-distance was originally proposed to compare character strings [52], where there are three kinds of edit operations: (i) *deleting* a character, (ii) *inserting* a character, and (iii) *changing* one character into another; consider matching the word “HELLO” to “HELLOW”, “HELO” and “JELLO”, respectively. Once costs are assigned to each edit operation, edit-distance is defined to be the minimum cost of the sequence of operations required to convert one string to another, and is typically computed using dynamic programming. Note that the *delete* operation and the *insert* operations are inverses of one another. An equivalent way to describe the same measure is to omit the *insert* operation, and ask for the minimum cost set of operations applied to the two strings that will succeed in transforming them into a common string.

The notion of edit-distance has been generalized for comparing ordered, rooted trees [45, 55] and ordered, unrooted trees [21], where analogous edit operations are defined as: (i) *contract* an edge, (ii) *uncontract* an edge, and (iii) *change* the label of an edge. As before, one can omit the *uncontract* operation, and ask for the minimum-cost set of operations taking the two trees to a common tree. Edit distance has been used in many computer-vision applications such as handwritten character and word recognition [12] and stereo matching [28]. The computational equivalency of edit distance and maximum common subgraph problems under certain conditions [6, 7] has been used for matching shock trees [49].

In applying the edit-distance approach to comparing shock graphs, traditional tree edit operations do not suffice. Rather, four groups of edit operations which are derived from the shock transitions are necessary, Figure 15: (i) the *splice* operation deletes a shock edge that is a leaf and merges the remaining two; (ii) the *contract* operations deletes a shock edge connecting two degree-three nodes; (iii) the *merge* operations combines two edges at a degree-two node; (iv) we also define a *deform* edit to match two shock edges with different attributes. As in traditional edit-distance, once the costs for these edits are determined, the edit-distance is defined to be the minimum-cost of a set of operations that

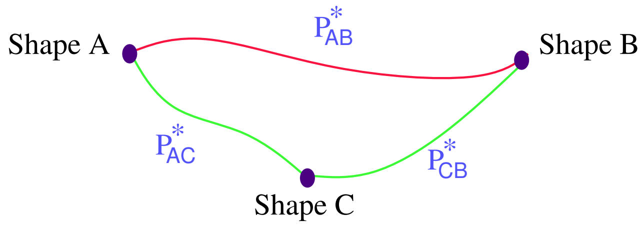


Figure 16: This figure illustrates that the distance satisfies the triangle inequality. Let P^*_{AB} (red curve) be the minimum-cost deformation path between shapes A and B. The deformation path between shapes A and B consisting of minimum-cost deformation paths P^*_{AC} and P^*_{CB} (green curves) is not typically optimal.

transforms the two graphs into a common graph.[1] The minimum-cost set of operations can be found using dynamic programming, as sketched in Appendix B. The definition and method of assigning costs to the edit operations is discussed in Section 5.

The computational complexity of the algorithm is $O(n_1^3 n_2^3)$ where n_1 and n_2 are the number of nodes in T_1 and T_2, respectively. However, the use of a heuristic allows for a reduction to $O(n^3 k^3)$ where $n = \max(n_1, n_2)$ and $k = \max(k_1, k_2)$, where k_i are the diameter of T_i. See Appendix B for details.

5 Cost of Edits

We now discuss how costs are assigned to each edit operation. The overall cost of the deformation path would then be computed as the sum of the cost of its subpaths. Observe that edit costs have to be consistent with each other, and also with our perceptual metrics of similarity. While the former can be developed mathematically, the latter can only be judged subjectively. Our basic approach is to first derive the cost of the *deform* edit, $i.e.$, the distance between two shapes within the same shape cell (identical shock topology), and then derive the cost of other edits as the limit of the deform cost as the shape moves to the boundary of the shape cell (transition shape). This will ensure that edit costs are consistent with each other.

We derive the deform edit cost by summing over local shape differences which are computed as differences between corresponding shock attributes. This approach is based on extending a previously developed metric for aligning curves [35, 37], which is reviewed below in Section 5.1. First, however, we observe that in many applications, such as clustering, and indexing into a large databases, the metric properties of the distance are critical for database organization. The following proposition shows that the deformation-based notion of distance between two shapes A and B $D(A, B)$ is a metric.

Proposition 1 *The distance $D(A, B)$ is a metric, $i.e.$, it satisfies the following properties for shapes A, B, and C: (i) $D(A, A) = 0$, (ii) $D(A, B) = D(B, A)$, and (iii) $D(A, B) \leq D(A, C) + D(C, B)$.*

[1] For the traditional formulation, one also includes operations that are inverse to *splice*, *contract*, and *merge*, and defines edit-distance to be the minimum cost of a sequence of edits to morph one graph to the other.

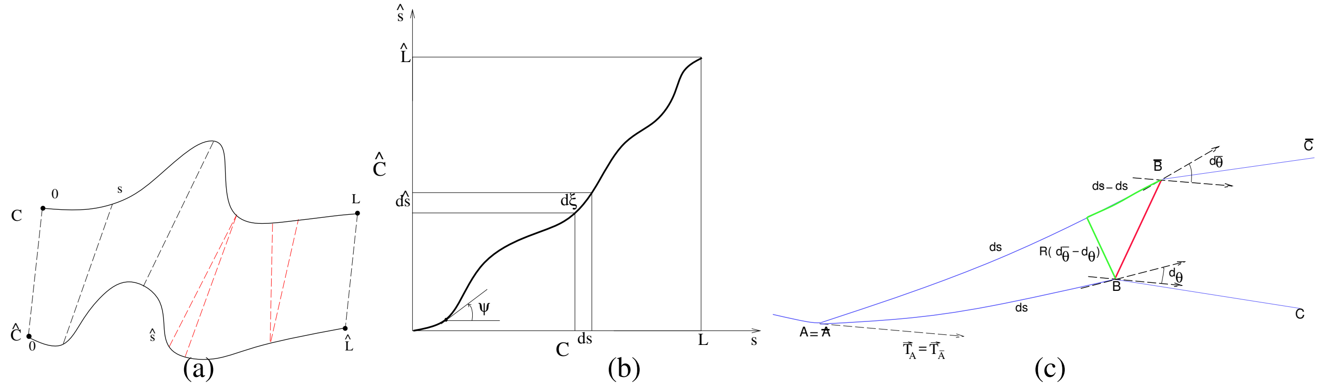


Figure 17: (a) An alignment of two curves $\mathcal{C}$ and $\hat{\mathcal{C}}$ is represented by a pairing of points on the two curves. (b) A symmetric treatment this pairing leads to a notion of an alignment curve. (c) The cost of deforming an infinitesimal segment AB to segment $\hat{A}\hat{B}$, when the initial points and the initial tangents are aligned ($A = \hat{A}$, $\vec{T}_A = \vec{T}_{\hat{A}}$), is related to the distance $B\hat{B}$, and is defined by $|d\hat{s} - ds| + R|d\hat{\theta} - d\theta|$ [35, 37].

Proof: (i) and (ii) follow directly from the definition. To prove (iii) let P^*_{AB}, P^*_{AC}, and P^*_{CB} be the minimum-cost deformation paths between (A, B), (A, C) and (C, B) with costs $D(A, B)$, $D(A, C)$, and $D(C, B)$ respectively. Consider the deformation path between A and B that passes through C (green curve in Figure 16), consisting of the minimum-cost deformation paths P^*_{AC} and P^*_{CB}. The cost of this path is $D(A, C) + D(C, B)$ by definition. From the optimality of the P^*_{AB} it follows that $D(A, B) \leq D(A, C) + D(C, B)$.

5.1 Matching Curves by Optimal Alignment

Our approach [35, 37] to comparing two curves $\mathcal{C}$ and $\hat{\mathcal{C}}$ consists of finding the minimum-cost deformation of one curve to another, where the cost is defined as the sum of "stretching" and "bending" energies [44, 54]. Let $\mathcal{C}$ and $\hat{\mathcal{C}}$ be parameterized by arclength s and $\hat{s}$ with tangent orientations $\theta(s)$ and $\hat{\theta}(\hat{s})$, respectively, Figure 17a. The basic premise of the approach is that the cost of matching entire curves can be expressed as the sum of the costs of matching infinitesimal subsegments, which is penalized by length and curvature differences, Figure 17c. Specifically, the cost of matching infinitesimal segments $d\mu$ is defined as $|d\hat{s} - ds| + R|d\hat{\theta} - d\theta|$, where R is a constant related to the average sample length. The problem is then cast as minimizing a functional over all possible *alignments* between the two curves. To ensure symmetric treatment of the curves, the alignment is represented as a pairing of points on the two curves leading to the notion of an *alignment curve* α, $\alpha(\xi) = (s(\xi), \hat{s}(\xi))$, where $\mathcal{C}(s(\xi))$ and $\hat{\mathcal{C}}(\hat{s}(\xi))$ denote the points on the two curves that correspond, Figure 17b. The optimal alignment curve α^* is found by minimizing the functional

$$\mu[\alpha] = \int_0^{\tilde{L}} \left[\left| \frac{d\hat{s}}{d\xi} - \frac{ds}{d\xi} \right| + R \left| \frac{d\hat{\theta}(\hat{s})}{d\xi} - \frac{d\theta(s)}{d\xi} \right| \right] d\xi. \tag{3}$$

where $\tilde{L}$ is the length of the alignment curve. A key advantage of using the alignment curve is that the alignment can be expressed in terms of a single function. We choose the angle between the tangent of

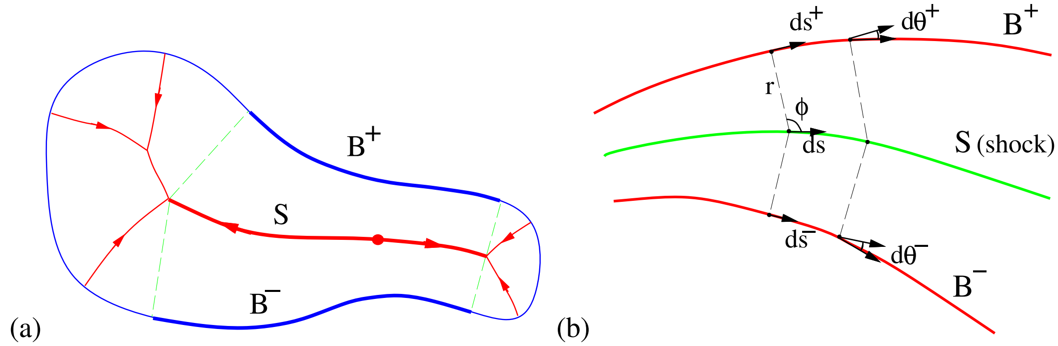


Figure 18: The shock $\mathcal{S}$ and the corresponding shape boundary segments $\mathcal{B}^+$ and $\mathcal{B}^-$.

alignment curve and the $x-$axis, $\psi(\xi)$, Figure 17(b). Then, Equation 3 can be rewritten in terms of ψ as

$$\mu[\alpha(\psi)] = \int_0^{\tilde{L}} \left[|\cos(\psi) - \sin(\psi)| + R|\kappa\cos(\psi) - \hat{\kappa}\sin(\psi)|\right] d\xi. \tag{4}$$

The optimal alignment curve is computed efficiently using dynamic programming [35, 37], and is reviewed in Appendix A. The distance $d(\mathcal{C}, \hat{\mathcal{C}})$ between the curves $\mathcal{C}$ and $\hat{\mathcal{C}}$ is then defined as the cost of the optimal alignment, $d(\mathcal{C}, \hat{\mathcal{C}}) = \mu[\alpha^*]$.

5.2 Deform Cost

The deform cost is defined as the sum of "local shape differences" between corresponding shock segments. This difference is defined by extending the metric between two curves to a metric between two shock segments. Observe that each shock segment represents a pair of segments on the boundary of the shape as shown in Figure 18a. Hence, we can view the problem of deforming one edge in a shock graph to an edge in another shock graph in terms of deforming the corresponding boundary segments, *i.e.*, as a "joint curve matching problem". As in the case of a single pair of curves, we penalize stretching and bending of each pair of boundary segments in terms of length and curvature differences. However, this notion of joint curve matching of shape boundary segment pairs must not only rely on simultaneous pairwise intrinsic differences between the boundaries, but also on changes in the relative pose of the two boundary segments, Figure 19.

Consider a pair of edges from a pair of shock graphs, $\mathcal{S}$ and $\hat{\mathcal{S}}$, each parameterized by s and $\hat{s}$, respectively. Let the boundary segments corresponding to $\mathcal{S}$, $\mathcal{B}^+$ and $\mathcal{B}^-$ be parameterized by s^+ and s^-, with tangent orientations $\theta^+(s^+)$ and $\theta^-(s^-)$, respectively, Figure 18b. The boundary segments of $\hat{\mathcal{S}}$ are similarly defined. The cost of matching infinitesimal segments of the shock edges are defined as in curve matching by length differences $|d\hat{s}^+ - ds^+| + |d\hat{s}^- - ds^-|$, and boundary curvature differences $|d\hat{\theta}^+ - d\theta^+| + |d\hat{\theta}^- - d\theta^-|$, but in addition now augmented with width differences $2(|\hat{r}_0 - r_0| + \int |d\hat{r} - dr|)$, and relative orientation differences $2(|\hat{\phi}_0 - \phi_0| + \int |d\hat{\phi} - d\phi|)$. As in the case of curves, the *shock alignment curve* is used to mediate the match, and the optimal alignment curve α^* is found by

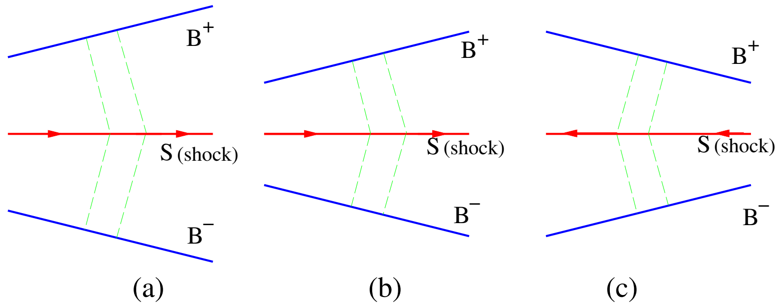


Figure 19: This figure illustrates why summing the pairwise differences in corresponding boundary segments is not sufficient to measure the distance between two shock segments. In all three cases the boundary segments are identical up to rotations and translations, and the cost of matching each individual pair *independently* is zero. However, the cost of matching the pair as a whole should not be zero, as (i) the segments in (a) are farther apart than those in (b) suggesting that shape in (a) is *wider* than shape in (b), and (ii) the segments in (b) are widening whereas those in (c) are narrowing.

minimizing the functional

$$\begin{aligned}\mu[\mathcal{S},\hat{\mathcal{S}};\alpha] =& \int_0^{\tilde{L}} [|\frac{d\hat{s}^+}{d\xi} - \frac{ds^+}{d\xi}| + |\frac{d\hat{s}^-}{d\xi} - \frac{ds^-}{d\xi}|]\, d\xi + R\int_0^{\tilde{L}} (|\frac{d\hat{\theta}^+}{d\xi} - \frac{d\theta^+}{d\xi}| + |\frac{d\hat{\theta}^-}{d\xi} - \frac{d\theta^-}{d\xi}|)\, d\xi \\ &+2|\hat{r}_0 - r_0| + 2\int_0^{\tilde{L}} |\frac{d\hat{r}}{d\xi} - \frac{dr}{d\xi}|d\xi + 2R|\hat{\phi}_0 - \phi_0| + 2R\int_0^{\tilde{L}} |\frac{d\hat{\phi}}{d\xi} - \frac{d\phi}{d\xi}|d\xi,\end{aligned} \tag{5}$$

where $\cos\phi = -\dfrac{dr}{ds}$ and $\tilde{L}$ is length of alignment curve. Note that the reconstruction equation (Equation 2 in Section 2) allows us to write μ in terms of shock properties. Specifically, using $ds^\pm = v^\pm ds$, and $d\theta^\pm = \kappa^\pm v^\pm ds$, we have

$$\begin{aligned}\mu[\mathcal{S},\hat{\mathcal{S}};\alpha] =& \int_0^{\tilde{L}} [|\hat{v}^+\frac{d\hat{s}}{d\xi} - v^+\frac{ds}{d\xi}| + |\hat{v}^-\frac{d\hat{s}}{d\xi} - v^-\frac{ds}{d\xi}|]\, d\xi \\ &+\int_0^{\tilde{L}} [R(|\bar{\hat{\kappa}}^+\frac{\sqrt{\hat{v}^2-1}}{\hat{v}}\frac{d\hat{s}}{d\xi} - \bar{\kappa}^+\frac{\sqrt{v^2-1}}{v}\frac{ds}{d\xi}| + |\bar{\hat{\kappa}}^-\frac{\sqrt{\hat{v}^2-1}}{\hat{v}}\frac{d\hat{s}}{d\xi} - \bar{\kappa}^-\frac{\sqrt{v^2-1}}{v}\frac{ds}{d\xi}|)]\, d\xi \\ &+\int_0^{\tilde{L}} |\frac{1}{\hat{v}}\frac{d\hat{s}}{d\xi} - \frac{1}{v}\frac{ds}{d\xi}|d\xi + |\hat{r}_0 - r_0| + \int_0^{\tilde{L}} R|\frac{\hat{a}}{\hat{v}\sqrt{\hat{v}^2-1}}\frac{d\hat{s}}{d\xi} - \frac{a}{v\sqrt{v^2-1}}\frac{ds}{d\xi}|d\xi + R|\hat{\phi}_0 - \phi_0|.\end{aligned}$$

Finally, the deform cost of the shock edges $\mathcal{S}$ and $\hat{\mathcal{S}}$ is defined as the cost of the optimal alignment,

$$d(\mathcal{S},\hat{\mathcal{S}}) = \min_\alpha \mu[\mathcal{S},\hat{\mathcal{S}};\alpha] \tag{6}$$

which is found by a dynamic-programming method described in [35, 37]. This method has $O(n^2)$ complexity, where n is the number of samples along the shock segments. The average running time on an SGI Indigo (195 MHz) for matching two shock segments with 50 points each is 0.17 seconds. Note that we currently compute the cost for all combinations of shock paths independently, leading to about 5 minutes per match. However, there is a great deal of redundancy which can be avoided by revising our implementation to avoid duplicate matches.

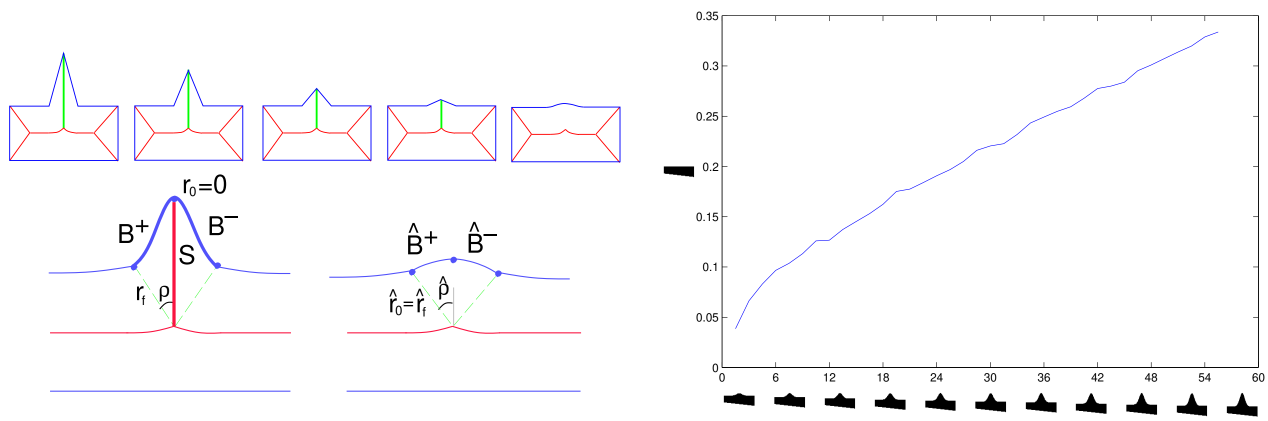


Figure 20: Left: Splice operation removes a protrusion, and can be viewed as the limit of shrinking the protrusion and replacing it with a circular arc. Right: The plot illustrates that splice cost behaves rather intuitively, and increases almost linearly when matching a sequence of shapes with a protrusion of increasing length. The x-axis is the length of the protrusion in pixels, and the y-axis is the normalized edit cost. Note that in all cases the optimal edit sequence involves the splice edit (pruning the protrusion).

5.3 Cost of Other Edits

Thus far, we have described how the *deform cost* between two shock segments is computed in an intrinsic manner. Other edits can be viewed as limiting cases of the corresponding deform operation, and penalized accordingly. The *splice* operation prunes a leaf edge in the shock graph and corresponds to a deformation sequence that shrinks a shock branch to a point. From the shape perspective, this is equivalent to replacing the boundary segment corresponding to the protrusion with a circular arc. Hence, the splice edit cost can be viewed as the cost of deforming the protrusion to the circular arc, and is derived from Equation 5. Observe that the alignment curve in this case is horizontal as the second shock edge is a point in this case. The first two terms reduce to the difference in lengths between the protrusion and the circular arc, $i.e.$, the difference in lengths between $\mathcal{B}^{\pm}$ and $\hat{\mathcal{B}}^{\pm}$ in Figure 20, which is given by $(\left|\int ds^{+}+\int ds^{-})-2\rho r\right|$. The third and fourth terms reduce to 0 as the orientation of the tangents at each endpoint remains the same. The fifth and sixth terms simplify to $2\hat{r}_0$ as $\int_0^{\tilde{L}} \left|\frac{d\hat{r}}{d\xi} - \frac{dr}{d\xi}\right| d\xi = \int_0^{\tilde{L}} |dr| = r_f = \hat{r}_0$ and $|\hat{r}_0 - r_0| = \hat{r}_0$. Finally, the last two terms reduce to 0 under the optimal pairing of boundary segments. Thus we get the splice cost to be

$$\mu_s = \left| \int ds^{+} + \int ds^{-} - 2\rho\hat{r}_0 \right| + 2\hat{r}_0. \tag{7}$$

The *contract* operation removes an edge between high degree nodes. Unlike other edit operations, the contract operation is not a local operation in that it affects the neighboring shock edges also. However, as an approximation we measure only the local effect of the contract operation by deforming the contracted edge to a point, and its cost is derived from Equation 5. The first two terms reduce to the lengths of the boundary segments $\mathcal{B}^{+}$ and $\mathcal{B}^{-}$, and is given by $\int ds^{+} + \int ds^{-}$. As the boundary

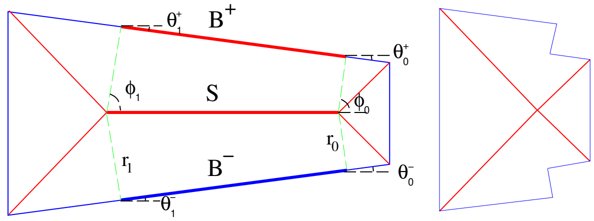


Figure 21: Contract operation

segments $\mathcal{B}^+$ and $\mathcal{B}^-$ are mapped to a point, the third and fourth terms reduce to the differences in orientations of the tangents at each endpoints, and is given by $R\big((\theta_0^+ - \theta_1^+) + (\theta_0^- - \theta_1^-)\big)$. The fifth and sixth terms reduce to zero and $2|r_0 - r_1|$, respectively. The seventh and eighth terms reduce to zero and $R(\phi_0^+ - \phi_1^+)$, respectively. Thus the contract cost is given by

$$\mu_c = \int ds^+ + \int ds^- + R\big((\theta_0^+ - \theta_1^+) + (\theta_0^- - \theta_1^-)\big) + 2|r_0 - r_1| + R\int d\phi. \tag{8}$$

Recall that the merge operation merges two edges adjacent to a degree-two node, and does not otherwise affect the topology of the shock graph. Hence, the decision whether or not to merge a set of adjacent edges is captured and integrated in the process of computing the deform cost.

5.4 Edit Distance between Two Shapes

Let the shapes to matched A and B be represented by their shock graphs T_A and T_B. For a shock graph T_A, let $e_{Ai} \in E_A, i = 1 \ldots N_A$ represent the set of edges. In the optimal edit sequence for matching A and B, let D_A denote the set of edges in T_A that corresponds with edges in T_B, and let S_A and C_A denote the set of edges that are spliced and contracted, respectively. Let $e_{Bj} = \phi_{AB}(e_{Ai}), e_{Ai} \in D_A, e_{Bj} \in D_B$ denote the corresponding edge of e_{Ai}. The edit distance of A and B is given by

$$\begin{aligned} D(A,B) = & \sum_{e_{Ai}\in D_A, e_{Bj}=\phi_{AB}(e_{Ai})} deformCost(e_{Ai}, e_{Bj})) + \sum_{e_{Ai}\in S_A} spliceCost(e_{Ai}) \\ & + \sum_{e_{Ai}\in C_A} contractCost(e_{Ai}) + \sum_{e_{Bi}\in S_B} spliceCost(e_{Bi}) + \sum_{e_{Bi}\in C_B} contractCost(e_{Bi}) \end{aligned}$$

6 Results of Shock-Graph Matching

In this section, we present the results of our approach. First we show that that shock-graph matching gives intuitive correspondences for a variety of shapes, and is effective in the presence of commonly occurring visual transformations like articulation, shadows and highlights, viewpoint variation, scale changes, boundary perturbations, and partial occlusion. Then, recognition results for two databases consisting of 99 shapes and 216 shapes are shown. Finally, shock-graph matching is compared to shock-tree matching [32] and shape-contexts matching [3].

Figure 22 shows that shock-graph matching results in an intuitive pairing of shock segments. In addition to giving the final correspondence, the edit distance algorithm gives a sequence of intermediate shock graphs that identify the optimal transformation of one input shock graph to another, Figure 23.

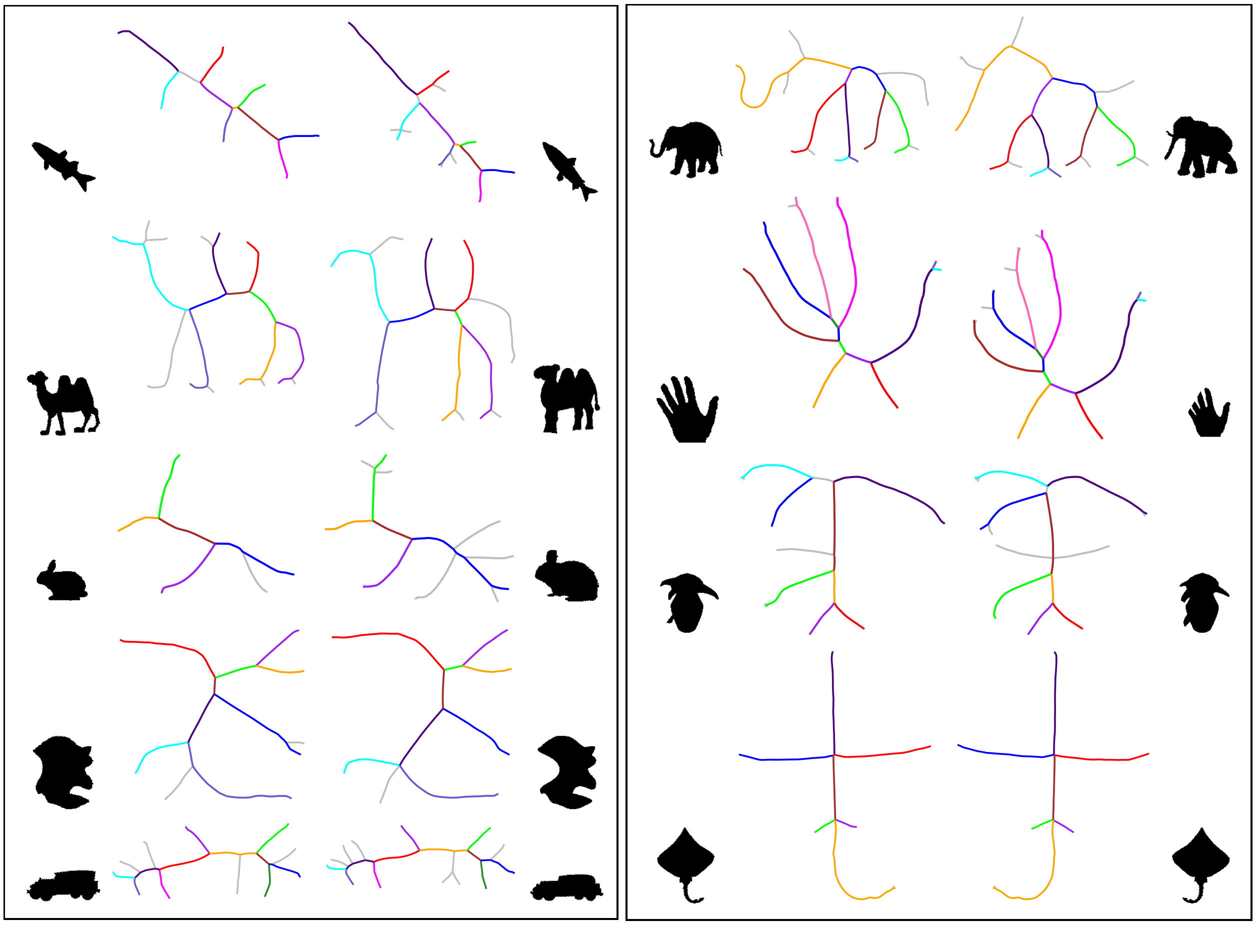


Figure 22: The result of matching the shock graphs for a few pairs of sample shapes in the database. Same colors indicate matching shock branches, while the grey colored edges in the shock graphs indicate that they are pruned. Observe that the correspondence obtained is intuitive in all cases. For example, the head, the fins, and tips of tail of the two fishes (top left) correspond. Also, note that when matching the elephants (top right), the shock edge corresponding to the tail of the elephant is pruned as the tail is occluded on the other.

6.1 Robustness of Recognition under Visual Transformations

We now examine the robustness of the match under six classes of visual transformations. First observe that shock computation is sensitive to **boundary perturbations**, which often introduce spurious shock edges and modify existing branches. While approaches relying on the medial axis have advocated either regularization in the detection process or after recovery to address such instabilities [29], one can alternatively view the *regularization as part of the recognition process*. This is illustrated in Figure 24, where one of the shapes being matched experiences discretization artifacts. The current approach considers all deformation paths between the noisy shape and the noiseless one, and finds that the optimal path consists of edits which prune the spurious branches. This is mainly because the cost of a single splice as well as the cumulative cost of a large sequence of splices is rather low as compared to large

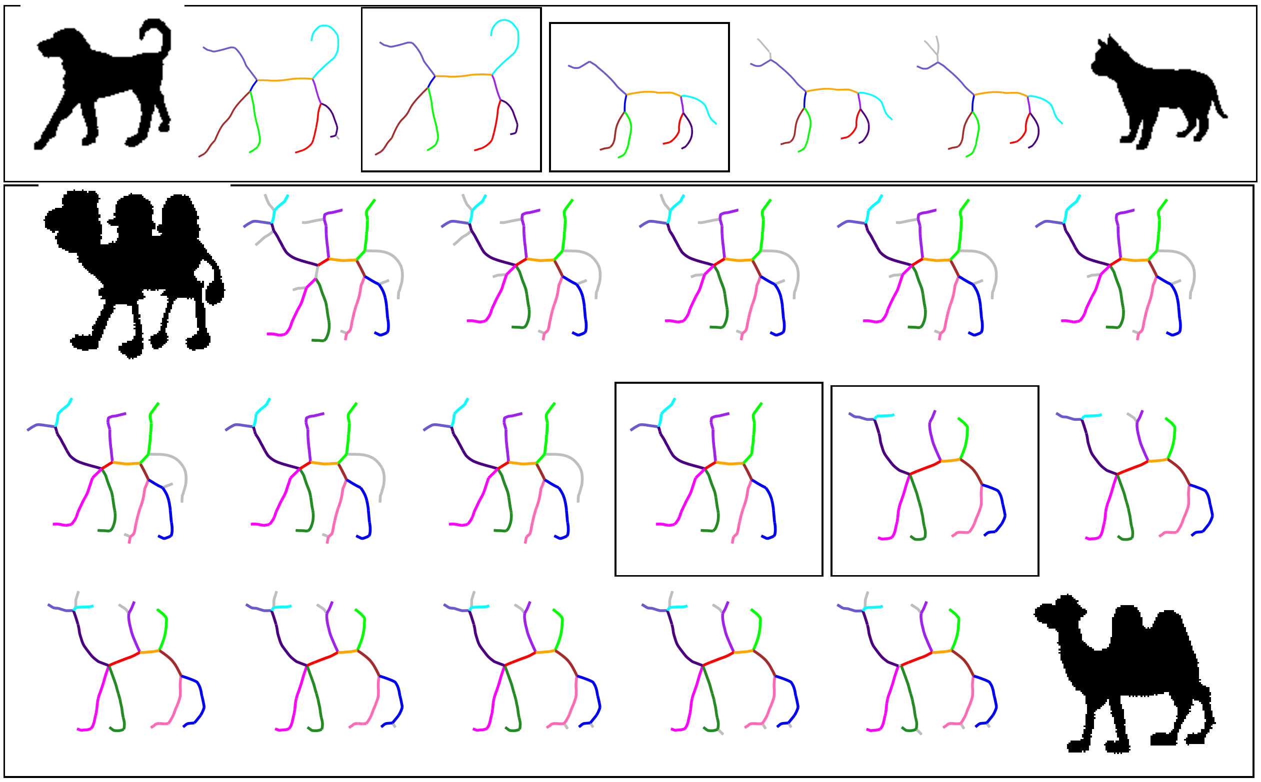


Figure 23: The intermediate shock graphs resulting from applying the edits in the optimal edit sequence for matching a cat and dog (top) and two camels (bottom). The boxed shock graphs have the same topology. The distance between the shapes is the sum of all edit costs.

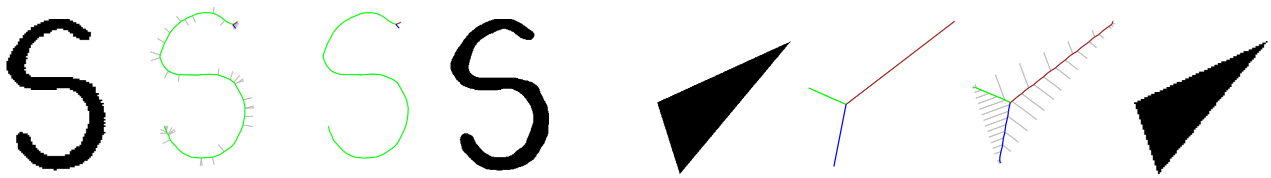


Figure 24: **Boundary perturbations:** Shock matching in the presence of boundary noise where the same shape is represented by a coarser discretization. Note that the shock graphs are matched intuitively: all the edges corresponding to the boundary noise are pruned, while matching salient parts.

structural changes.

Second, we examine robustness of the match in the presence of **articulation and deformation of parts**. The shock graph of a shape inherently segments the shape into parts, and captures the hierarchical relationship between those parts. Thus, shock-graph matching implicitly involves matching the global hierarchy of parts in addition to matching the individual parts, making it robust to changes which may occur in some of the parts, Figure 25.

Third, since 2D shapes are typically obtained from projections of 3D objects, robustness to **viewpoint variations** is critical to a 2D shape matching technique. When the viewpoint is varied gradually, the spatial location and the shape of parts typically changes gradually, and such changes are handled by the

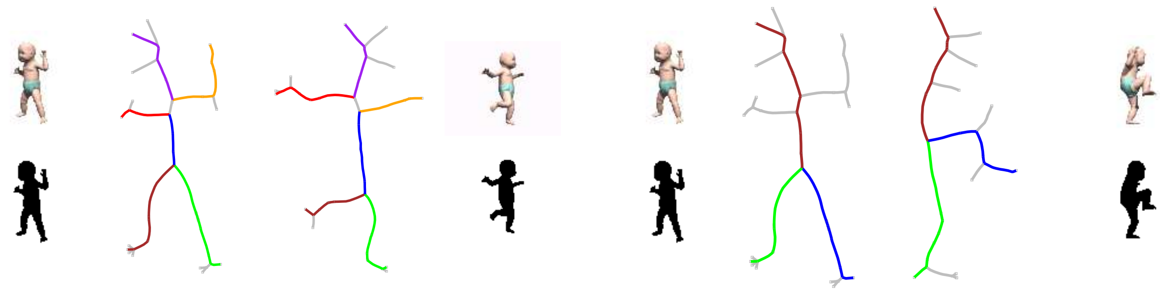


Figure 25: **Articulation and deformation of parts:** Matching of the two poses of the "baby" is intuitive. Observe, in particular, how the legs are matched correctly in spite of the self occlusion that merges part of the legs in the shape on the right (top row) and how the arms on the left are pruned (bottom row).

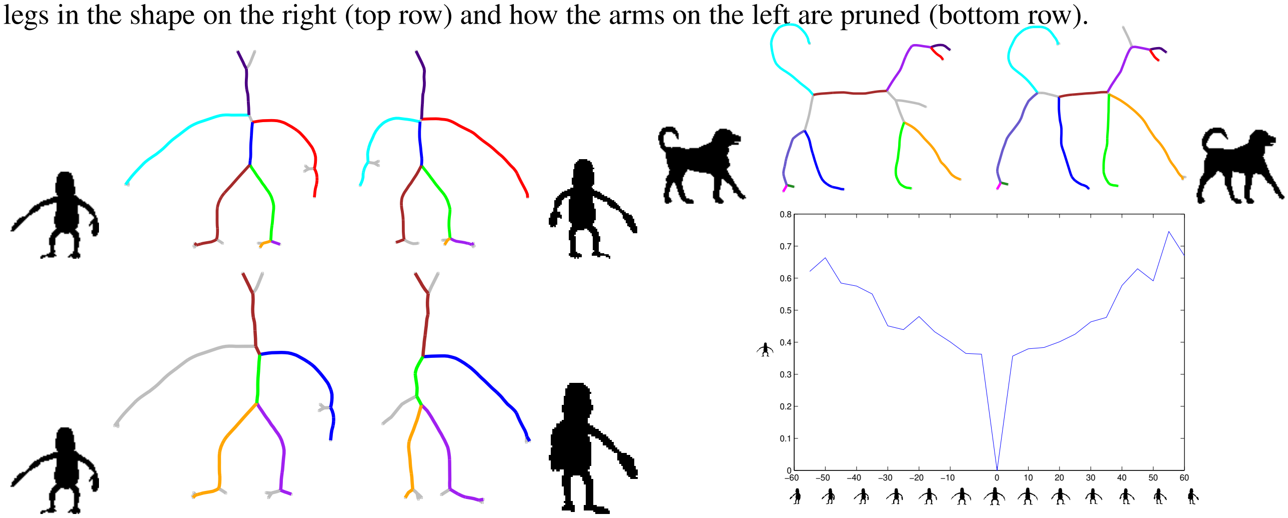


Figure 26: **Viewpoint variation:** Shock graph matching gives the intuitive correspondence when a change in viewpoint results in (i) disappearance of a part (left arm of model in middle row) and (ii) a change in aspect (hind legs of retriever in top row). Right: The plot shows that the normalized edit cost (y-axis) increases monotonically with viewpoint variation for a local neighborhood. This property has been used for 3D object recognition from 2D views [11].

deform edit. Exceptionally, however, at certain views, there are sudden changes that are handled by the remaining edits, $e.g.$, appearance of a part is handled by the splice edit; a change in aspect is handled by the contract edit, etc, Figure 26. In general, a change in viewpoint constitutes a one-parameter family of deformations on shape, for which a well-defined neighborhood locally exists through the explicit embedding of transitions.

Fourth, the presence of **illumination variation** like shadows and highlights often leads to **segmentation errors** in figure-ground segregation. On the one hand, changes in the boundary caused by highlights tend to be small, but typically affect the shock-graph topology. On the other hand, shadows are often more global in nature, and tend not to affect the shock-graph topology. Figure 27 shows that shock-graph matching is effective in the presence of a limited extent of segmentation errors.

Fifth, we examine the robustness of the proposed technique to **scale variations**. Observe that the

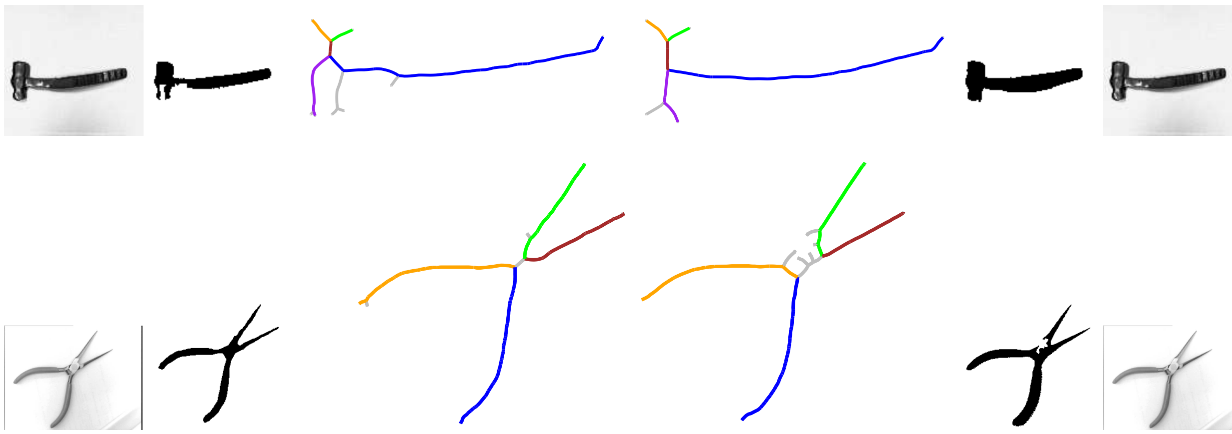


Figure 27: **Illumination variation:** Shock graph matching gives intuitive correspondence in the presence of segmentation errors due to highlights (hammer in top left and pliers in bottom left) and shadows (hammer in top right). The original images are from the database of Stan Sclaroff, Boston University, and segmented using a region growing technique with manually selected thresholds.

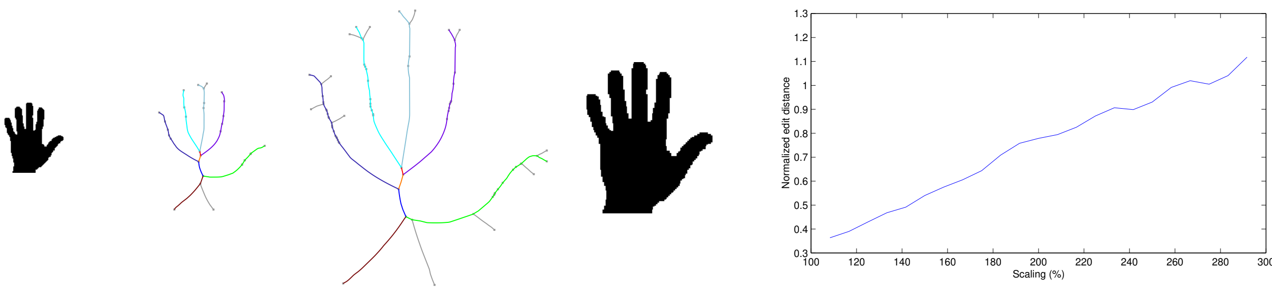


Figure 28: **Scale variations:** Left: Shock graph matching of the hands gives intuitive correspondence in the presence of a scaling transform. The hand on the right was obtained by scaling the original hand by 225%. Right: The plot shows that the edit cost (y-axis) increases linearly as scaling is increased (x-axis).

shock-graph topology does not change when a global scaling transformation is applied, but the shock edge comparison metric is not scale-invariant (as it should not be because scale-invariance requires consideration of the entire shape). However, the behavior of the metric with scale is theoretically well-understood: outside a limited range of scales, the metric is linear in scale changes based on Equation 5. Figure 28 illustrates that shock-graph matching gives the intuitive correspondence in the presence of modest amounts of scaling. However, in the presence of large amounts of scaling, in this case more than three times, unintuitive correspondence *can* result if parts of unequal size are present, as the cost of matching equal-size but non-corresponding parts may become dominant.

Sixth, **partial occlusion** is a serious challenge for any recognition framework. We consider two types of occlusion, one where the occluder blends with the shape, and one where the occluder blends with the background. In both types shock-graph matching gives the intuitive correspondence by pruning out the occluded part, Figure 29. However, it may fail if a large part of the shape is occluded, in which case the splice cost may dominate the total cost.

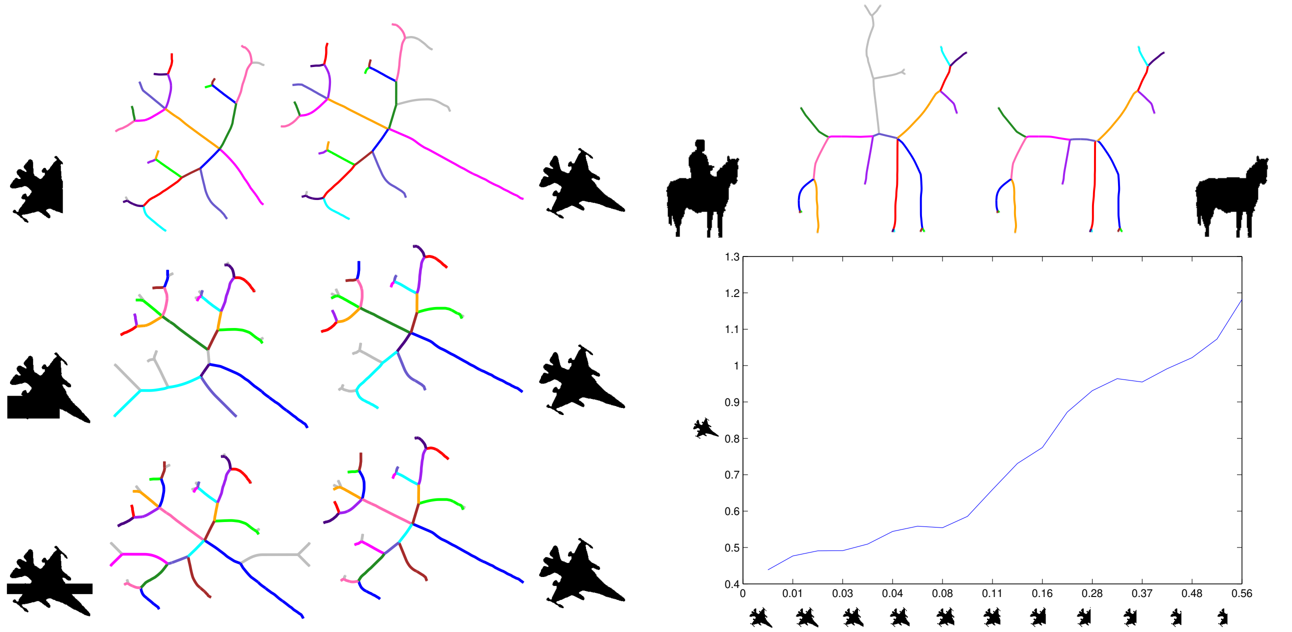
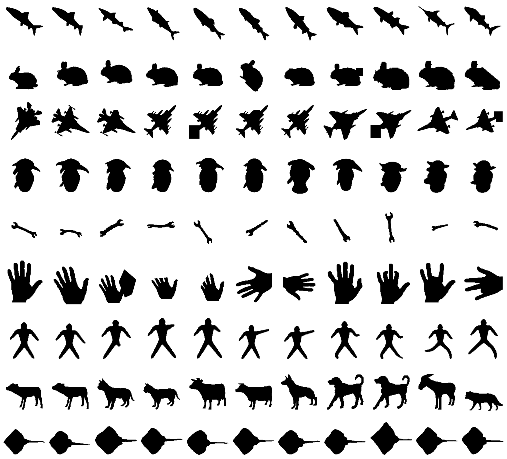


Figure 29: **Partial occlusion:** This figure illustrates that shock graph matching gives intuitive correspondence in the presence of partial occlusion, both when the occluder blends with the background (top row) and with the object (middle and bottom left). Observe in particular how the optimal edit sequence between a horse and a rider with the horse by itself involves pruning the shock edges corresponding to the rider. The plot shows that normalized edit cost (y-axis) increases as the fraction of the plane occluded (x-axis) increases.

6.2 Indexing Results

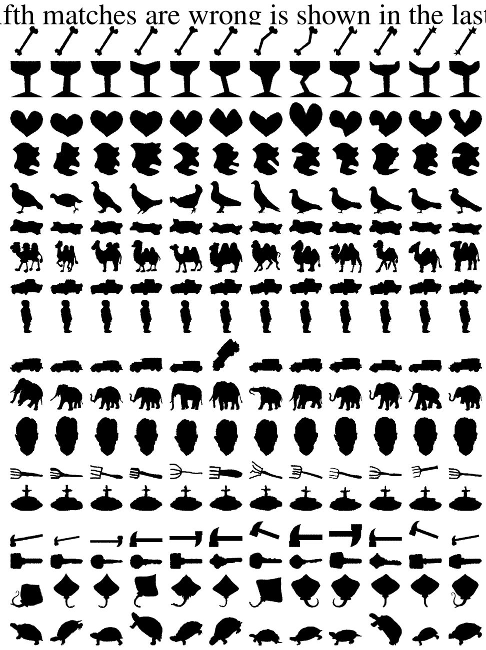
We now examine recognition rates for indexing into two databases. The first database was created by assembling sample shapes from a wide spectrum of sources[2], to form nine categories: fish, rabbit, airplane, "greeble", tool, hand, doll, four-legged animal, and sea-animal. We include eleven shapes in each category to allow for variations in form, as well as for occlusion, articulation, missing parts, $etc.$, for a total of 99 shapes. Each shape is matched against all others, and results are ordered by edit-distance, Table 1. Recognition rates in $\%$ by this measure are $(100, 100, 100, 99, 99, 99, 97, 96, 95, 87)$; in other words the top three choices are always correct for this database with a fairly slow dropoff in performance for the other choices.

We constructed a second database from samples of a very large database of shapes created for testing the compression rates for MPEG7, kindly provided by Latecki and Lakamper [25]. Our selection from this database consists of 18 categories with 12 shapes in each category, Table 2. Recognition rates in $\%$ are $(100, 100, 100, 99, 97, 99, 96, 96, 95, 91, 80)$; in other words the top three choices are always correct for this database.



	550	551	560	567	572	589	593	613	616	678	809	812	828	836	838
	350	573	581	600	616	618	646	655	720	770	793	824	860	860	869
	739	748	753	756	756	777	788	811	812	836	932	932	933	937	946
	300	306	352	379	381	426	452	462	464	488	1036	1057	1092	1127	1132
	322	507	572	574	578	589	649	649	704	911	939	942	955	956	957
	209	255	265	268	268	273	276	289	299	334	650	679	697	714	714
	535	556	558	614	628	637	646	652	685	693	702	702	714	720	738
	300	600	607	617	622	628	634	637	641	642	643	643	649	650	654

Table 1: Left: A database of 99 shapes. Right: The 15 nearest neighbors for a few representative shapes ordered by their normalized edit distance are shown. Note that the edit distances are multiplied by 1000 for clarity of presentation. As there are 11 shapes in each category, up to ten nearest neighbors can be from the same category. The next five matches are shown for completeness. Observe that in most cases the top 10 matches are from the same category. In fact, the top three matches are always correct, and the only instance where the fourth and the fifth matches are wrong is shown in the last row.



	426	427	464	472	473	510	511	522	527	544	553	612	616	625	631
	398	485	520	527	528	545	558	574	575	591	616	768	768	777	778
	449	457	477	517	519	521	531	536	562	587	600	619	647	658	658
	705	706	715	721	725	752	760	766	772	800	803	814	814	820	824
	584	614	652	653	661	663	696	699	713	736	768	771	808	817	823
	303	319	320	323	354	360	363	369	377	380	382	604	609	615	620
	186	188	213	277	313	335	338	406	456	477	492	528	557	557	560
	519	530	554	599	610	630	651	665	682	692	702	732	745	760	766
	537	546	558	565	568	592	622	623	635	647	656	667	676	678	693

Table 2: Left: The database of 216 shapes selected from the MPEG-7 test database [25]. Right: The 15 nearest neighbors for a few representative shapes are shown. As there are 11 other shapes in each category, up to eleven nearest neighbors can be from the same category. Observe that in most cases the top 11 matches are from the same category.

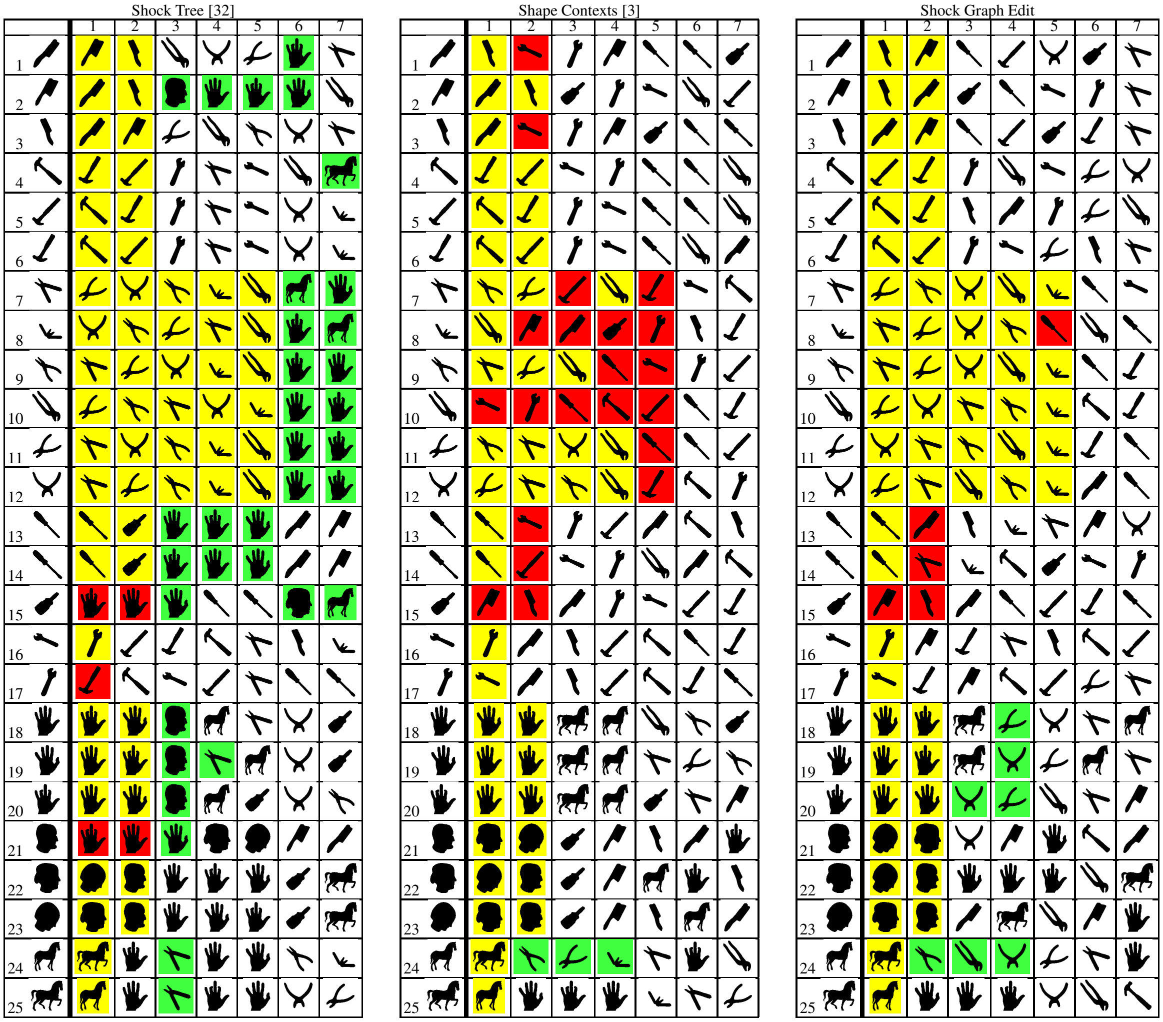


Table 3: Comparison of shock tree matching using association graphs [32], shape-contexts matching [3], and shock graph edit distance using the database used in [32]. The correct within-category matches are highlighted using yellow surround, erroneous within-category matches using red, and unintuitive between-category matches using green. Note the sensitivity of shape contexts to articulation in matching the pliers (rows 7-12, middle table).

6.3 Comparison to other approaches

In this section, we compare our approach to the approaches based on matching Siddiqi et al.'s rooted shock tree [32] and shape context [3]. We have repeated the indexing experiment reported in [32] using our approach and shape-context matching [3]. The database used in [32] contains twenty five shapes, which can be organized into eight categories, Table 3: brush (1,2,3), hammer (4,5,6), pliers (7,8,9,10,11,12), screwdriver (13,14,15), wrench (16,17), hand (18,19,20), profile (21,22,23), horse (24,25). Four categories, namely, hammer, plier, screwdriver and wrench group into a more abstract superordinate category of tools. In addition, just based on shape comparison, note the similarities of the brush and screwdriver categories. The remaining categories are quite distinct; the only possible

[2]"Greebles" are obtained from Mike Tarr's collection, the sea-animals from Farzin Mokhtarian's database, and tools from Stan Sclaroff's collection.

connection is that both hands and horses have four elongated sequentially placed "limbs".

With this in mind, two types of performance measures are used to compare the results. First, we ask how many of the within category shapes are correctly recalled among the top matches. We highlight correct within-category matches using a yellow surround, and erroneous within-category matches using a red surround. In Siddiqi et al.'s rooted shock tree matching shapes 15, 17 and 21 give rise to a total of 5 errors. In shape-context matching shapes 1, 3, 7, 8, 9, 10, 11, 12, 13, 14 and 15 give rise to a total of 21 errors. In the edit-distance approach shapes 8, 13, 14, 15 give rise to a total of 5 errors. Clearly, both shock-based approaches perform significantly better than shape-context matching.

A more careful examination of the differences between the two shock-based methods, however, reveals that in the five errors of Siddiqi et al.'s rooted shock-tree matching, the erroneously recalled shapes are infact perceptually very dissimilar: fat screwdriver (shape 15) is matched to the hands, wrench (shape 17) with the hammer, and profile (shape 21) with hands. In contrast, in three out of five errors in using edit-distance shock-graph matching the top-ranking shapes match the query shape very well, $i.e.$, fat screwdriver (shape 15) with brush and screwdriver (shape 13) with brush. In the fourth error, the match between shape 14 with shape 8, the screwdriver is considered a part of the shape 8 (a single ply). The other remaining error (shape 14), however, has a dissimilar shape as a match. Thus, the within-category errors in Siddiqi et al.'s rooted shock-tree matching have in general dissimilar shapes, while in edit-distance shock graph matching the erroneous recalls are in fact correct in picking similar shapes, which are nevertheless cognitively categorized differently.

A second type of performance measure involves counting the number of unintuitive matches using the superordinate categorization described above. We have highlighted such matches using a green surround. In Siddiqi et al.'s rooted shock tree matching shapes 1, 2, 4, 7-15, 18, 19, 20, 21, 24, 25 give rise to a total of 33 unintuitive matches. For example, the top matches of a brush (shape 2) includes a profile (3rd match), and hands (matches 4-6) where more similar shapes are available. Similarly, for the screwdrivers (shapes 13, 14, 15) the top matches include the hands. On the other hand, there are 3 unintuitive matches using shape-context matching for shape 24, and 7 unintuitive matches for shapes 18, 19, 20 and 24 using edit-distance shock-graph matching. This second performance measure, while somewhat subjective (in determining the category structure), is a clear qualitative indicator that the edit-distance shock-graph matching recalls similar shapes better than both Siddiqi et al.'s rooted shock tree matching and shape-context matching for this database.

We have also compared our approach to shape-context matching [3] using our databases of 99 and 216 shapes[3]. The results are summarized using precision-recall diagrams, Figure 30. Note that precision and recall have been used to evaluate the performance of a shape retrieval scheme [27]. Precision

[3]We have not applied Siddiqi et al.'s rooted shock-tree matching [43, 32] to these databases, because currently we do not have access to their code.

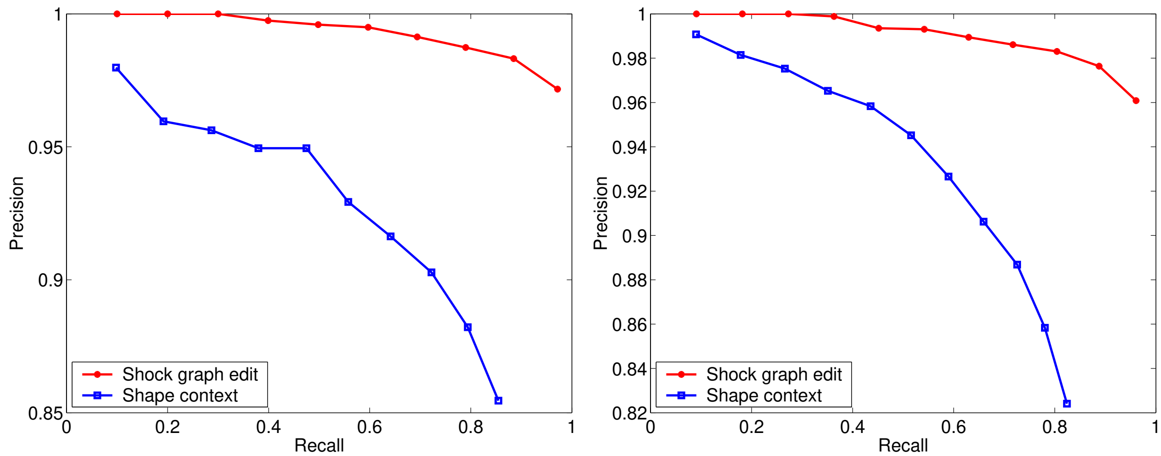


Figure 30: The precision-recall diagrams for indexing into the database of 99 shapes (left) and 216 shapes (right). Note that both methods perform well with more that 80% precision.

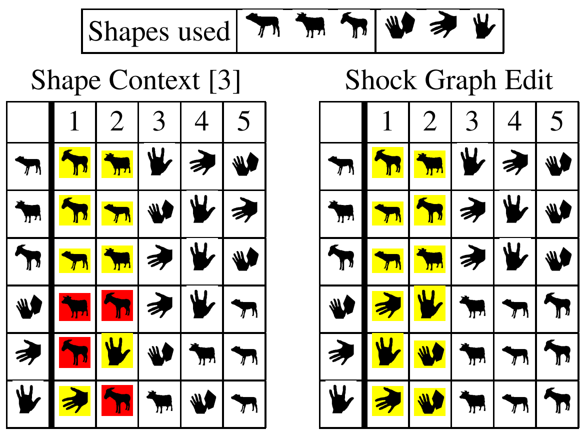


Table 4: This table shows the sensitivity of shape contexts to partial occlusion. In particular, observe how the occlusion of the thumb results in the hand being matched to an animal (rows 4-6, left table).

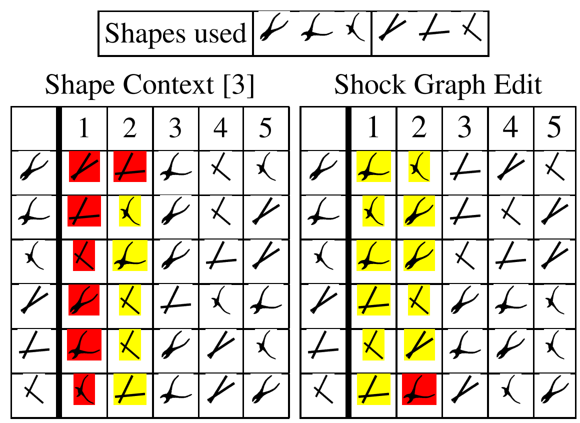


Table 5: This table illustrates the sensitivity of shape contexts to articulation. Observe how shape context matches the pliers to sticks in the same relative pose, and not to other pliers.

is the ratio of the number of similar shapes retrieved to the total number of shapes retrieved, while recall is the ratio of the number of similar shapes retrieved to the total number of similar shapes in the database. It is clear from these diagrams that shock graph edit-distance outperforms shape-context matching for these databases.

Finally, we examine the performance of our approach and shape-contexts matching in the presence of partial occlusion and part-based changes like articulation. We have selected a small dataset of six shapes (three hands and three animals) to examine the performance in the presence of partial occlusion, Table 4. Occlusion affects a portion of a shape, and in the shock graph representation, the changes are limited to the affected portions. On the other hand, shape contexts is global in that all points are used to compute a summary signature for each point. This difference results in shock graph matching outperforming the shape contexts matching, Table 4. Shock graphs inherently partitions the shape and

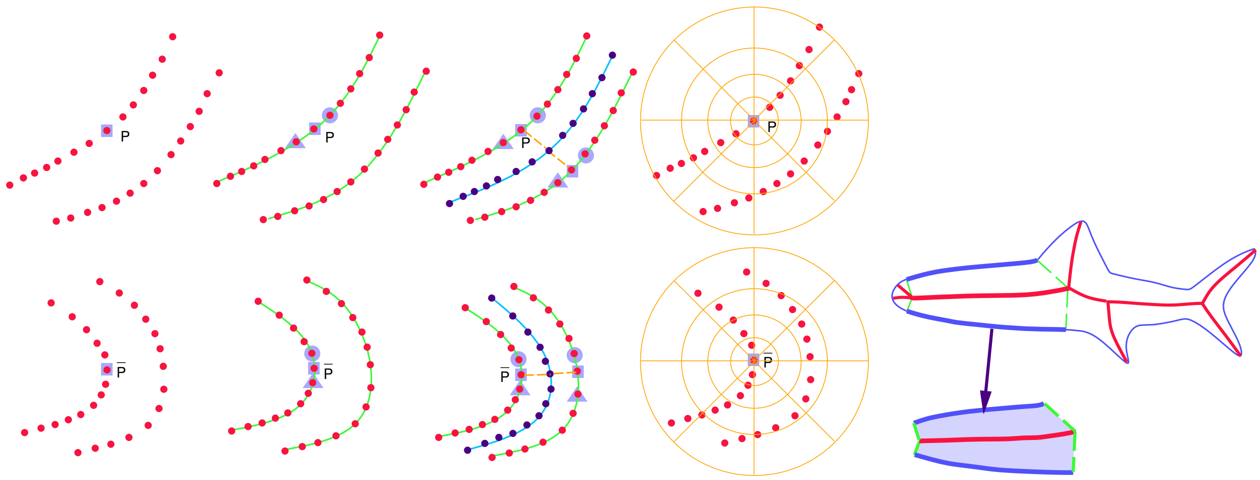


Figure 31: Left: A schematic comparison of shape representation using unorganized points, outline curve, and shock graph. The point-based representation uses a local or global similarity measure. The curve representation induces an ordering of points. The shock graph representation introduces an additional constraint by the pairing two boundary segments (shown by dotted lines) to represent the interior of the shape. The main point is that in matching P to $\bar{P}$ the curve-based representation also needs to match neighboring points in the boundary, while the shock-based representation in addition forces a matching of the corresponding points as another constraint. Right: Each shock segment (red) corresponds to two boundary segments (blue) and represents a shape fragment (shaded).

captures the hierarchical relationship among the parts. This allows it to be robust to part-based changes like articulation, unlike the shape context representation, Table 5.

7 Discussion and Conclusions

We have presented a novel recognition framework based on considering different deformation paths of shock graphs of 2D shapes which brings them into correspondence, and finding the optimal one. Our approach to finding the optimal deformation path relies on a formal classification of shock transitions which is used to discretize the shape space and deformation paths. We employ a graph edit distance algorithm to find the optimal deformation sequence. The cost of the optimal sequence gives a metric of dissimilarity which is then used for indexing into databases based on shape. The effectiveness of the approach in the context of a variety of visual transformations is demonstrated and indexing into two separate databases of 99 and 216 shapes results in 100% accuracy for the top three matches. Note that this approach measures distance between shapes in a *dynamic* fashion in that intermediate representations are explicitly considered [24, 26, 56] during the process of finding the least-cost deformation path. This is in contrast to the *static* approach where two shape representations are directly compared [13, 34, 2, 3, 32, 40, 43] to determine the distance.

A question which naturally arises is whether the introduction of an additional "medial" machinery can be justified as compared to point-based and curve-based approaches. First, observe that in the medial/shock graph representation, there is an increased level of "organization" as compared to curve or point-based representations. In point-based representations, a point is identified by its local or global context. In curves, the point's identity is established through the ordered 1D neighborhood. In shock-graph (or medial) representations, the point's identity is established through a *pair* of joint 1D ordered neighborhoods that are captured via the medial segment. This additional level of organization in going from points to shocks can greatly constrain a match as highlighted by matching icons in Figure 31, and thus avoid potential mismatches and preserve the coherence of shapes in the match process.

Another difference between point-based [3, 9] and medial-based approaches is that the arrangement and spatial layout of local features is captured via embedding it in an *extrinsic* framework for the former, while it is captured *intrinsically* in the latter. That a point's identity is measured in terms of its signature in an external coordinate system with respect to other points is precisely what makes it sensitive to shape deformations which are small in "intrinsic" deformation energy, but which are large in extrinsic deformation energy, $e.g.$, bending, articulation, $etc.$ In other words, instead of measuring the *relative* spatial arrangements of local features, as is done in medial representation, extrinsic methods capture the absolute spatial arrangement. This difference is what makes the point-based methods sensitive to object-based changes like articulation, occlusion and to global rotations, as illustrated in Section 6.3.

There are a few areas for future improvement. The current approach to computing the edit costs is not scale-invariant. The individual edit costs themselves should not be scale-invariant since the determination of scale requires the consideration of the entire shape. However, the overall metric needs to be scale-invariant. We plan to examine the match at several scales and determine the global scale by selecting the one that minimizes the match cost. Since the edit costs are linear outside a limited range of scales, only a few scales need to be examined [34]. A second direction involves modifying how the matching algorithm penalizes the deletion of a portion of a shape that consists of several parts. Currently, such a delete is penalized as the sum of the costs of the individual parts being deleted, which can lead to unintuitive matches in cases where a large portion of the shape is occluded. In future, we plan to incorporate a more explicit notion of parts to address this issue.

Our initial shape comparison results are rather promising and support further exploration of this framework. We aim to explore recognition rates on a much larger database, as we believe that it is in the context of a much larger space of shapes that the effectiveness of this algorithm can truly be studied. However, two developments are needed. First, the algorithm for matching shapes typically takes about 5 minutes on an SGI Indigo II (195 MHz) for the examples presented here, which limits the number of shapes that can be practically matched. We plan to improve the computational efficiency by revising and optimizing the "development" code and also by revising the algorithm itself. We are also developing a coarse-to-fine scale matching strategy, notion of exemplars to represent a shape category [39] and a nearest neighbor search method [36], which can be integrated in the indexing

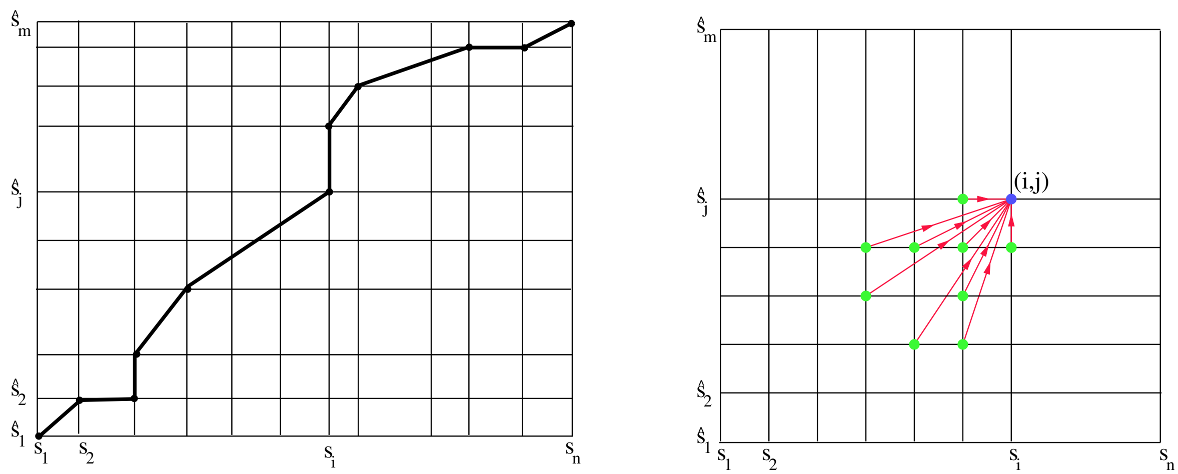


Figure 32: (a) This figure illustrates the grid used by the dynamic-programming method to compute the optimal alignment curve. Discrete samples along the curves specify the axes. The sample alignment curve is shown using bold line segments. (b) This figure illustrates the template which is used to limit the choices when updating the cost at (i, j) in the dynamic-programming cost table. The template ensures that only nine candidate alignment curves at different orientations are considered at each point.

scheme to avoid unnecessary matches, thus reducing the computational burden.

Appendix A Algorithm to Compute the Alignment Curve

This section briefly reviews how the optimal alignment curve is computed by minimizing the cost functional in Equation 4 using dynamic-programming. For details see [35, 37]. Let $\mathcal{C}$ and $\hat{\mathcal{C}}$ be the curves to be matched. The alignment curve of $\mathcal{C}$ and $\hat{\mathcal{C}}$ is the a curve in the 2D plane whose axes are specified by $\mathcal{C}$ and $\hat{\mathcal{C}}$. We discretize $\mathcal{C}$ and $\hat{\mathcal{C}}$ by samples $s_1, s_2, \ldots, s_n$ and $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_m$, respectively. Let $\mathcal{C}|_{[s_i,s_j]}$ be the subsegment of curve $\mathcal{C}$ with s_i and s_j as endpoints. Let $d(i, j)$ denote the cost of matching the segments $\mathcal{C}|_{[s_1,s_i]}$ and $\hat{\mathcal{C}}|_{[\hat{s}_1,\hat{s}_j]}$, and let $\delta([k, i], [l, j])$ be the cost of matching the segments $\mathcal{C}|_{[s_k,s_i]}$ and $\hat{\mathcal{C}}|_{[\hat{s}_l,\hat{s}_j]}$. The definition of the distance in terms of a functional ensures that it satisfies the following rule

$$d(i, j) = \min_{k,l} \left[d(i-k, j-l) + \delta([i-k, i], [j-l, j]) \right] \tag{9}$$

which gives a recipe for computing $d(\mathcal{C}, \hat{\mathcal{C}})$ via dynamic programming. The dynamic-programming cost table is sequentially updated, and the optimal alignment curve is found by tracing through the cost table. We limit the choices of the k and l, as a first approximation, to nine values achieved by using the template shown in Figure 32.

Appendix B Edit-Distance Algorithm and Computational Complexity

This section reviews the algorithm for finding the edit-distance between two unrooted trees, which is used to find the globally optimal sequence of transitions between two shock graphs in polynomial time [22, 23]. We first describe the case of two *rooted* trees T_1 and T_2; the case of unrooted trees is discussed later. Given a rooted tree T, each edge $\{x, y\}$ of T connecting nodes x and y can be represented by two oppositely directed arcs (x, y) and (y, x), called *darts*. The *Euler tour* $E(T)$ traverses the tree, and is a cycle consisting of all the darts. An *Euler string* is a consecutive subsequence of the dart-sequence which represents the Euler tour. Observe that a portion of a tree T_i can be specified by the Euler string s_i, and is denoted by the pair $H_i = (T_i, s_i)$. A *sub-problem* is specified by the pair $(H_1, H_2) = ((T_1, s_1), (T_2, s_2))$. Dynamic programming is used to solve all the sub-problems, $i.e.$, compute the tree edit-distance between the portions of the trees specified by the Euler strings s_1 and s_2. Note that the edit-distance for each sub-problem is in turn computed from the edit distance for a sub-problem having shorter strings ($i.e.$ the sum of the lengths of the two strings is smaller). Finally, the edit-distance between two trees is computed as a sub-problem where the Euler strings encompass the trees.

First we sketch how a sub-problem is solved in the absence of *splice* and *merge* operations. This method is due to Zhang and Shasha [55]. To compute the edit distance for $((T_1, s_1), (T_2, s_2))$, consider the rightmost darts of s_1 and s_2. In the optimal edit sequence, there are only three possibilities concerning these darts: (i) the edge corresponding to the rightmost dart in s_1 of T_1 gets contracted, (ii) the edge corresponding to rightmost dart in s_2 is similarly contracted, (iii) the two edges are matched together. Using this observation, the edit-distance for $((T_1, s_1), (T_2, s_2))$ can be expressed as the minimum of three quantities, each involving the edit distance for a sub-problem with smaller substrings, one for each of the three cases *(i)*, *(ii)*, and *(iii)*. The quantity for case *(i)* is the cost of contracting the edge corresponding to the rightmost dart in s_1 plus the edit-distance for $((T_1, s_1'), (T_2, s_2))$, where s_1' is the prefix of s_1 consisting of all darts but the rightmost. The quantity for case *(ii)* is the cost of contracting the edge corresponding to the rightmost dart in s_2 plus the edit-distance for $((T_1, s_1), (T_2, s_2'))$, where s_2' is the prefix of s_2 consisting of all darts but the rightmost. The quantity for case *(iii)* is the cost of matching the edge of T_1 corresponding to the rightmost dart of s_1 against the edge of T_2 corresponding to the rightmost dart of T_2.

Next, we discuss how the above approach can be adapted to handle *merge* and *splice*. Note that repeated use of the *merge* or *splice* operations can combine a path of edges into a single edge. Hence we assume here (and discuss in Section 5) a method for assigning a match cost not to a pair of edges but to a pair of paths.

In computing the edit distance for $((T_1, s_1), (T_2, s_2))$, there are now many additional cases: for each dart a_1 in s_1 that is a descendent in T_1 of the last dart in s_1, and for each dart a_2 in s_2 that is a descendent in T_2 of the last dart in s_2, there is a quantity: the cost of matching the path in T_1 from a_1 to the last dart in s_1 against the path in T_2 from a_2 to the last dart in s_2, plus the edit-distance of $((T_1, a_1), (T_2, a_2))$. In this sketch of the algorithm we have omitted some technical details arising in handling *merge* and *splice*.

Thus far, we have discussed how to compute the edit distance between to two rooted trees. This algorithm finds not the best match but the best match subject to the constraint that the two roots are matched to each other (and hence that these roots are not removed by splice operations). To remove this constraint, we must modify the above algorithm. A straightforward way of doing this is to run the rooted algorithm once for each possible choice of roots, then take the best match among all of those obtained. This approach is guaranteed to find the optimal solution.

A heuristic that yields a faster algorithm but one that is not guaranteed to find the optimal solution is as follows. Choose a small set of candidate roots for T_2, and run the rooted algorithm for every possible root for T_1 and only the candidate roots for T_2. This approach finds the best match subject to the constraint that not all of the candidate roots is eliminated by splices. In order to make it likely that the optimal solution satisfies this constraint, the set of candidate roots should be chosen to contain nodes whose elimination is expensive. In the experiments described in this article, we chose a singleton set consisting of that degree-three node r with outgoing darts g_1, g_2, g_3 so as to maximize $\hat{c} = \min_j \text{splicecost}(g_i)$. This guarantees that the prevented splices will cost at least $\hat{c}$. When using the heuristic, we can obtain an even faster algorithm by applying a technique due to Zhang and Shasha. The details of this technique are beyond the scope of this paper.

Now we turn to time complexity analysis. We sketch a straightforward analysis of the non-heuristic algorithm ($i.e.$ all roots of T_1 and T_2 are considered). For $i = 1, 2$, let n_i be the number of nodes in T_i. Then the number of subproblems is $O(n_1^2 n_2^2)$. For each subproblem $((T_1, s_1), (T_2, s_2))$, the number of quantities involved is the number of darts in s_1 that are descendents of the rightmost dart of s_1, times the number of darts in s_2 that are descendents of the rightmost dart of s_2. This product is $O(n_1 n_2)$, so the total time is $O(n_1^3 n_2^3)$.

The heuristic method (using one candidate root for T_2) can be shown to require time $O(n_1^2 \text{ diameter}(T_1) n^2 \text{ collapsedDepth}(T_2) \text{ depth}(T_2))$, where the diameter diameter(T_1) is the maximum number of edges on a path in T_1, the depth depth(T_2) is the maximum number of edges on a path in T_2 that starts from the root, and collapsedDepth(T_2) is a quantity defined by Zhang and Shasha, and shown to be no more than the depth of T_2. If each of the two trees has no more than n nodes and has diameter no more than k, the time is $O(n^3 k^3)$. The analysis of the heuristic method is beyond the scope of this paper.

References

[1] N. Ayache and O. Faugeras. HYPER: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):44–54, January 1986.

[2] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38:2365–2385, 1998.

[3] S. Belongie and J. Malik. Matching with shape contexts. *IEEE Workshop on Content-based Access of Image and Video Libraries*, 2000.

[4] H. Blum. Biological shape and visual science. *J. Theor. Biol.*, 38:205–287, 1973.

[5] J. Bruce, P. Giblin, and C. Gibson. Symmetry sets. *Proceedings of the Royal Society of Edinburgh*, 101A:163–186, 1985.

[6] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, August 1997.

[7] H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(9):917–922, September 1999.

[8] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, March 1998.

[9] H. Chui and A. Rangarajan. A new algorithm for non-rigid point matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages II:44–51, 2000.

[10] I. Cohen, N. Ayache, and P. Sulger. Tracking points on deformable objects using curvature information. *European Conference on Computer Vision*, pages 458–466, 1992.

[11] C. M. Cyr and B. B. Kimia. 3D object recognition using shape similarity-based aspect graph. *International Conference on Computer Vision*, pages 254–261, 2001.

[12] W. Dewaard. An optimized minimal edit distance for hand-written word recognition. *Pattern Recognition Letters*, 16(10):1091–1096, October 1995.

[13] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(12):1312–1328, December 1999.

[14] P. J. Giblin and B. B. Kimia. On the intrinsic reconstruction of shape from its symmetries. *IEEE Conf. Computer Vision and Pattern Recognition*, pages 79–84, 1999.

[15] P. J. Giblin and B. B. Kimia. On the local form and transitions of symmetry sets, and medial axes, and shocks in 2D. *International Conference on Computer Vision*, pages 385–391, 1999

[16] P. J. Giblin and B. B. Kimia. Transitions of the 3D medial axis under a one-parameter family of deformations. *European Conference on Computer Vision*, pages 718–724, 2002

[17] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(4):377–388, 1996.

[18] D. G. Kendall, D. Barden, T. K. Carne, and H. Le. *Shape and Shape Theory*. John Wiley & Sons, Inc, Chichester, UK, 1999.

[19] B. B. Kimia. Conservation laws and a theory of shape. Ph.D. dissertation, McGill Center for Intelligent Machines, McGill University, Montreal, Canada, 1990.

[20] B. B. Kimia, J. Chan, D. Bertrand, S. Coe, Z. Roadhouse, and H. Tek. A shock-based approach for indexing of image databases using shape. *SPIE: Multimedia Storage and Archiving Systems II*, volume 3229, pages 288–302, 1997.

[21] P. Klein. Computing the edit distance between unrooted ordered trees. *European Symposium on Algorithms*, pages 91–102, 1998.

[22] P. Klein, T. Sebastian, and B. Kimia. Shape matching using edit-distance: an implementation. *ACM-SIAM Symposium on Discrete Algorithms*, pages 781–790, 2001.

[23] P. Klein, S. Tirthapura, D. Sharvit, and B. Kimia. A tree-edit distance algorithm for comparing simple, closed shapes. *ACM-SIAM Symposium on Discrete Algorithms*, pages 696–704, 2000.

[24] K. Kupeev and H. Wolfson. On shape similarity. In *International Conference on Pattern Recognition*, pages 227–231, 1994.

[25] L. J. Latecki, R. Lakamper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. *IEEE Conf. Computer Vision and Pattern Recognition*, pages 424–429, 2000.

[26] T. Liu and D. Geiger. Approximate tree matching and shape similarity. *International Conference on Computer Vision*, pages 456–462, 1999.

[27] E. Milios and E. Petrakis. Shape retrieval based on dynamic programming. *IEEE Trans. Image Processing*, 9(1):141–146, 2000.

[28] R. Myers, R. Wilson, and E. Hancock. Bayesian graph edit distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(6):628–635, June 2000.

[29] R. L. Ogniewicz. *Discrete Voronoi Skeletons*. Hartung-Gorre, 1993.

[30] S. Pae and J. Ponce. On computing structural changes in evolving surfaces and their appearance. *Intl. J. of Computer Vision*, 43(2):113–131, July 2001.

[31] M. Pelillo. Matching free trees, maximal cliques, and monotone game dynamics. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 423–437, 2001.

[32] M. Pelillo, K. Siddiqi, and S. Zucker. Matching hierarchical structures using association graphs. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(11):1105–1120, November 1999.

[33] M. Pelillo, K. Siddiqi, and S. W. Zucker. Many-to-many matching of attributed trees using association graphs and game dynamics. *International Workshop of Visual Form*, pages 583–593, 2001.

[34] T. Sebastian, P. Klein, and B. Kimia. On aligning curves. *IEEE Trans. Pattern Analysis and Machine Intelligence*, page In Press, 2002.

[35] T. B. Sebastian, J. J. Crisco, P. N. Klein, and B. B. Kimia. Constructing 2D curve atlases. *Mathematical Methods in Biomedical Image Analysis*, pages 70–77, 2000.

[36] T. B. Sebastian and B. B. Kimia. Metric-based shape retrieval in large databases. *International Conference on Pattern Recognition*, Accepted – To Appear, 2002.

[37] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Alignment-based recognition of shape outlines. *International Workshop on Visual Form*, pages 606–618, 2001.

[38] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing their shock graphs. *International Conference on Computer Vision*, pages 755–762, 2001.

[39] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Shock-based indexing into large shape databases. *European Conference on Computer Vision*, pages 731 – 746, 2002

[40] D. Sharvit, J. Chan, H. Tek, and B. B. Kimia. Symmetry-based indexing of image databases. *Journal of Visual Communication and Image Representation*, 9(4):366–380, December 1998.

[41] K. Siddiqi, B. Kimia, A. Tannenbaum, and S. Zucker. Shocks, shapes, and wiggles. *Image and Vision Computing*, 17(5-6):365–373, 1999.

[42] K. Siddiqi and B. B. Kimia. A shock grammar for recognition. *IEEE Conf. Computer Vision and Pattern Recognition*, pages 507–513, 1996.

[43] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *Intl. J. of Computer Vision*, 35(1):13–32, November 1999.

[44] H. D. Tagare. Shape-based non-rigid correspondence with application to heart motion analysis. *IEEE Trans. Medical Imaging*, 18(7):570–578, July 1999.

[45] K.-C. Tai. The tree-to-tree correction problem. *Journal of the Association for Computing Machinery*, 26:422–433, 1979.

[46] H. Tek. The role of symmetry maps in representating objects in images. Ph.D. dissertation, Division Of Engineering, Brown University, Providence, RI, 02912, July 1999.

[47] H. Tek and B. B. Kimia. Symmetry maps of free-form curve segments via wave propagation. *International Conference on Computer Vision*, pages 362–369, 1999.

[48] H. Tek, F. Leymarie, and B. B. Kimia. Multiple generation shock detection and labeling using CEDT. *International Workshop on Visual Form*, pages 582–593, 1997.

[49] A. Torsello and E. R. Hancock. Efficiently computing weighted tree edit distance using relaxation labeling. *Workshop of Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 438–453, 2001.

[50] A. Torsello and E. R. Hancock. A skeletal measure of 2D shape similarity. *International Workshop of Visual Form*, pages 260–271, 2001.

[51] S. Vidal, E. Bardinet, G. Malandain, S. Damas, and N. de la Blanca Capilla. Object representation and comparison inferred from its medial axis. *International Conference on Pattern Recognition*, pages Vol I: 712–715, 2000.

[52] R. Wagner and M. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.

[53] W. Wallis, P. Shoubridge, M. Kraetz, and D. Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22(6-7):701–704, May 2001.

[54] L. Younes. Computable elastic distance between shapes. *SIAM Journal of Applied Mathematics*, 58:565–586, 1998.

[55] K. Zhang and D. Sasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18:1245–1262, 1989.

[56] S. C. Zhu and A. L. Yuille. FORMS: A flexible object recognition and modeling system. *Intl. J. of Computer Vision*, 20(3):187–212, 1996.