Generic Model Abstraction from Examples

Yakov Keselman, Sven Dickinson

Abstract

The recognition community has long avoided bridging the representational gap between traditional, low-level image features and generic models. Instead, the gap has been artificially eliminated by either bringing the image closer to the models, using simple scenes containing idealized, textureless objects, or by bringing the models closer to the images, using 3-D CAD model templates or 2-D appearance model templates. In this paper, we attempt to bridge the representational gap for the domain of model acquisition. Specifically, we address the problem of automatically acquiring a generic 2-D view-based class model from a set of images, each containing an exemplar object belonging to that class. We introduce a novel graph-theoretical formulation of the problem, in which we search for the *lowest common abstraction* among a set of lattices, each representing the space of all possible region groupings in a region adjacency graph representation of an input image. The problem is intractable, and we present a shortest path-based approximation algorithm to yield an efficient solution. We demonstrate the approach on real imagery.

I. Introduction

A. Motivation

In the object recognition community, object representations have spanned a continuum ranging from prototypical models (often called class-based or generic models) to exemplar-based models (often called template-based or appearance-based models). Those advocating prototypical models address the task of recognizing novel (never before seen) exemplars from known classes, whose definitions strive to be invariant to changes in surface texture, color, part articulation, and minor deformation in shape. Those advocating exemplar models address the very different task of recognizing particular instances of objects, such as JFK's face or a can of Coke. In a completely orthogonal direction, prototypical models can be object-centered or viewer-centered (or both [14], [16], [15]), provided that the 3-D or 2-D features that comprise the model satisfy the above invariance goals. Similarly, exemplar models can be object-centered, specifying the exact 3-D geometry of an object, e.g., a rigid CAD model "template", or viewer-centered, specifying the exact appearance of an object.

Interestingly, the evolution of object recognition over the past 30 years has followed a path from prototypical models to exemplar models, as illustrated in Figure 1. Beginning in the 1970's, vision researchers aimed for prototypical vision systems, using complex volumetric parts, such as generalized cylinders (e.g., [6], [1], [41], [8]), and later superquadrics (e.g., [43], [20], [52], [22], [53], [34]) and geons (e.g., [5], [14], [16], [15], [4], [45], [7]). The main challenge to these early systems was the *representational gap* that existed between the low-level features that could be reliably extracted, and the

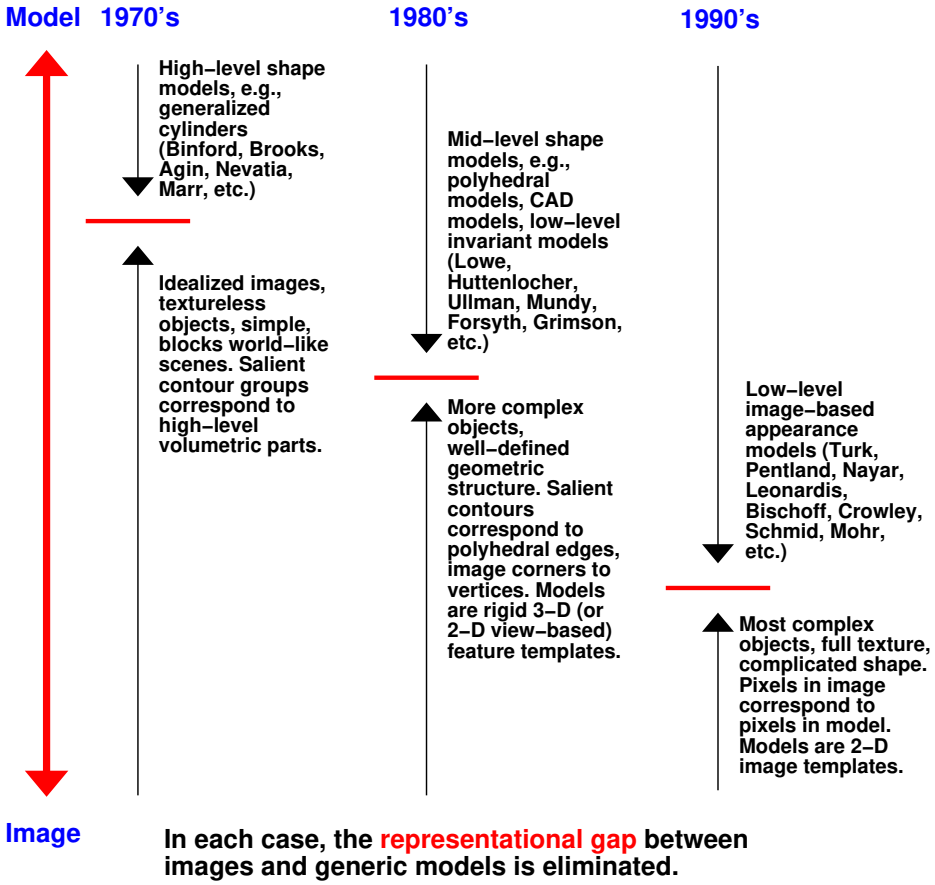


Fig. 1. Evolution of object recognition.

abstract nature of the model components. Rather than addressing this representational gap, the community effectively eliminated it by bringing the images closer to the models. This was accomplished by removing object surface markings and structural detail, controlling lighting conditions, and reducing scene clutter. Edges in the image could then be assumed to map directly to the limbs and surface discontinuities of high-order volumetric parts making up the models. The results left many unsatisfied, as the images and objects were often contrived, and the resulting systems were unable to deal with real objects imaged under real conditions.

The 1980's ushered in 3-D models that captured the exact shape of the object. Such models, often in the form of CAD models, were effectively 3-D templates, e.g., [24], [35], [21]. Provided that such a model could be acquired for a real object, the community now found that it could build object recognition systems that could begin to recognize real (albeit restricted) objects. This time, the representational gap

was eliminated by bringing the model closer to the imaged object, requiring the model to capture the exact geometry of the object. Moreover, since the presence of texture and surface markings seriously affected the computational complexity of these systems, they too selected objects which were texture-free – objects for which a salient image edge discontinuity mapped to a polyhedral edge. Again, there was dissatisfaction, as the resulting systems were unable to recognize complex objects with complex surface markings.

Recently, beginning in the 1990's, appearance models have replaced CAD models, and for the first time, recognition systems were constructed that could recognize arbitrarily complex objects, e.g., [54], [39], [33]). Storing a dense set of images of the object from all possible viewpoints, the appearance models were not limited by object geometric complexity, texture, or surface markings. In this case, the representational gap was eliminated by bringing the models all the way down to the image. The resulting systems could therefore recognize only exemplar objects – specific objects that had been seen at training time. Despite a number of serious limitations of this approach, including difficulties in dealing with background clutter, occlusion, object translation, rotation, and scaling, the approach has gained tremendous popularity.

It is important to note that in bringing the model closer to the image, the appearance-based and CAD-based approaches have altered the problem definition from generic to exemplar object recognition. The systems developed in the 70's cannot be compared to those developed today, for their target domains are different. We must acknowledge the efficacy of appearance-based recognition systems for exemplar recognition; provided that the above limitations can be overcome, this technique may emerge as the best method for recognizing exemplars. However, it is important to acknowledge that the prototypical recognition problem is an important one, and despite the vision community's movement toward appearance-based methods, the hypothesis that these (or their analogous 3-D template-based) methods can scale up to perform prototypical object recognition is dubious. What, then, has led us away from the important problem of generic object recognition?

Over the past 30 years, the three approaches to eliminating the representational gap (shown in Figure 1) are driven by the same limiting assumption: there exists a one-to-one correspondence between a "salient" feature in the image (e.g., a long, high-contrast line or curve, a well-defined homogeneous region, a corner or curvature discontinuity or, in the case of an appearance-based model, the values of a set of image pixels, possibly restricted to the vicinity of an interest point) and a feature in the model. In the case of generic object recognition, this assumption is fundamentally flawed, for saliency in the image does *not* imply saliency in the model. Under this assumption, object recognition will continue to be exemplar-based, and generic recognition will continue to be rather contrived.

To make real progress on the problem of generic object recognition, we must address the representational gap outlined in Figure 1. Not only must we continue to push the technologies of segmentation and perceptual grouping, we must be able to generate image *abstractions* that may not exist explicitly in the image, but which capture the salient, invariant shape properties of a generic model.[1] We argue that for the purpose of generic or prototypical object (or, more specifically, shape) recognition, the process of image abstraction must recover a set of "meta-regions" that map to the coarse surfaces (or volumetric primitives) on a prototypical model.

In light of this goal, the difficult challenge of image abstraction can therefore be cast as a twofold problem, as shown in Figure 2. First, we seek a (compile-time) method for automatically acquiring a generic model from a set of images (containing exemplars belonging to a known class) that bridges the representational gap between the output of an image segmentation module and the "parts" of a generic model. Although in Figure 2, this is exemplified by a generic, view-based model (a coffee cup), those advocating object-centered models could easily map the parts of this view into a set of object-centered components, e.g., [14], [16], [15], leading to an object-centered model. Although we make no claim as to the final form of the model (2-D or 3-D), we believe that a parts-based approach to viewer-centered representations can better accommodate the intractable complexity associated with "whole object" views of complex, articulating objects [16].

Next, from an image of a real exemplar, we seek a (run-time or recognition-time) method that will recover a high-level "abstraction" that contains the coarse features that make up some model. In this paper, we present an approach to the first problem, that of generic model acquisition from a set of exemplars belonging to a known class.[2] The second, and more difficult problem of generic object recognition, is work in progress and is not reported here.[3] Generic object recognition is an essential task, whose lack of solution confines object recognition to the laboratory, where such conditions as lighting, clutter, occlusion, and object domain can be tightly controlled. Granted, the generic object recognition torch has continued to burn, albeit dimly, with a number of researchers committed to the problem, e.g., [48], [57], [2], [51], [42], to name just a few. We believe the time has come for the pendulum to swing back towards solving the problem of generic, prototypical, or class-based modeling and recognition. In fact, such a trend is

[1] Although a number of approaches extract regions or perceptual groups as input to a recognition system, they typically assume that corresponding regions or groups exist on the model.

[2] Preliminary algorithms and results have been reported in [30], [29], [28]. This paper expands on these earlier versions, providing additional details on the algorithms and including many new experiments.

[3] A discussion of the recognition problem, along with our proposed approaches, appears in Section VI.

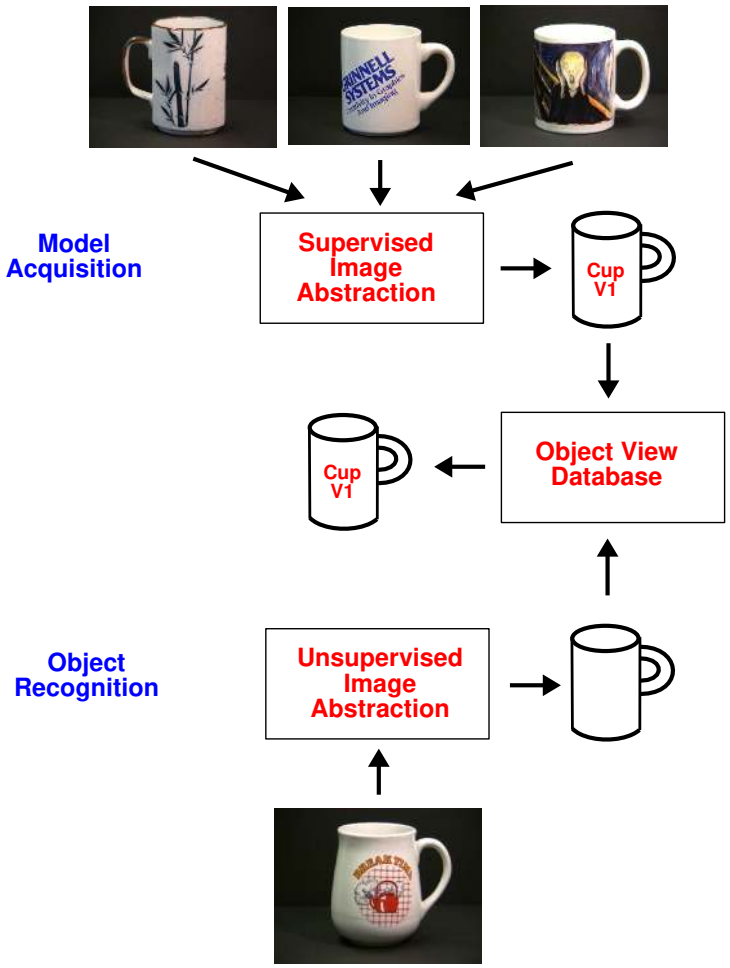
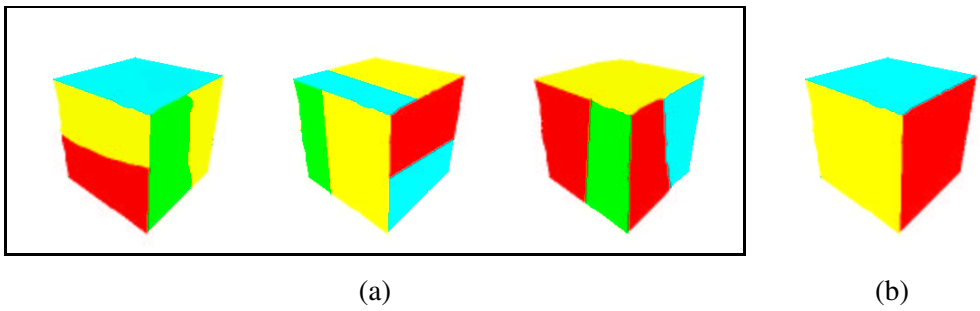


Fig. 2. The role of image abstraction in both model acquisition and object recognition.

beginning, as witnessed by the movement away from image-based appearance models to robust, interest point features, e.g., [47], [36], [9], [10], and shape contexts (point features with an even larger, albeit qualitative, neighborhood description), e.g., [3].

B. An Illustrative Example

Assume that we are presented with a collection of images, such that each image contains a single exemplar, all exemplars belong to a single known class, and that the viewpoint with respect to the exemplar in each image is similar. Figure 3(a) illustrates a simple example in which three different images, each containing a block in a similar orientation, are presented to the system (we will return to this example throughout the paper to illustrate the various steps in our algorithm). Our task is to find the common structure in these images, under the assumption that structure that is common across many



(a) (b)

Fig. 3. Illustrative example of generic model acquisition: (a) input exemplars belonging to a single known class; (b) generic model abstracted from examples.

exemplars of a known class must be definitive of that class. Figure 3(b) illustrates the class "abstraction" that is derived from the input examples. In this case, the domain of input examples is rich enough to "intersect out" irrelevant structure (or appearance) of the block. However, had many or all the exemplars had vertical stripes, for example, the approach would be expected to include vertical stripes in that view of the abstracted model.

Any discussion of model acquisition must be grounded in image features. In our case, each input image will be region-segmented to yield a region adjacency graph.[4] Similarly, the output of the model acquisition process will yield a region adjacency graph containing the "meta-regions" that define a particular view of the generic model.[5] Other views of the exemplars would similarly yield other views of the generic model. The integration of these views into an optimal partitioning of the viewing sphere, or the recovery of 3-D parts from these views, is beyond the scope of this paper. For now, the result will be a collection of 2-D views that describes a generic 3-D object. This collection would then be added to the view-based object database used at recognition time.

C. Related Work

Automatic model acquisition from images has long been associated with object recognition systems. One of the advantages of appearance-based modeling techniques, e.g., [39], is that no segmentation,

[4]We advocate region segmentation for several reasons. First, surfaces on the object naturally correspond to regions in the image. Second, region shape similarity (a critical component of our algorithm) can be effectively computed using the approach (our previous work) described in [51]. Third, a number of efficient and robust region segmentation algorithms have been recently developed [19], [50], [11].

[5]Unlike the regions in the input exemplar region adjacency graphs, the regions of the model region adjacency graphs are salient, and can be mapped into meaningful higher-level surface or volumetric primitives.

grouping, or abstraction is necessary to acquire a model. An object is simply placed on a turntable in front of a camera, the viewing sphere is sampled at an appropriate resolution, and the resulting images (or some clever representation thereof) are stored in a database. Others have sought increased illumination-, viewpoint-, or occlusion-invariance by extracting local features as opposed to using raw pixel values, e.g., [44], [36], [9], [10], [47], [40], [55]. Still, the resulting models are very exemplar-specific due to the extreme locality at which they extract and match features, and are as far from generic as one can get.

In the domain of range images, greater success has been achieved in extracting coarse models. Generic shape primitives, such as restricted generalized cylinders, quadrics, and superquadrics have few parameters and can be robustly recovered from 3-D range data [43], [52]. Provided the range data can be segmented into parts or surfaces, these generic primitives can be used to model the coarse shapes of the parts, effectively abstracting away structural detail. Unlike methods operating on 2-D data, these methods are insensitive to perceived structure in the form of surface markings or texture.

In the domain of recovering generic models from 2-D data, there has been much less work. The seminal work of Winston [56] pioneered learning descriptions of 3-D objects from structural descriptions of positively or negatively labeled examples. Nodes and edges of graph-like structures were annotated with shapes of constituent parts and their relations. As some shapes and relations were abstractions and decompositions of others, the resulting descriptions could be organized into a specificity-based hierarchy. In the 2-D shape model domain, Ettinger learned hierarchical, structural descriptions from images, based on Brady's curvature primal sketch features [18]. The technique was successfully applied to traffic sign recognition and remains one of the more elegant examples of generic model acquisition.

In the domain of graph algorithms and computer vision, there have been efforts to generate a prototypical graph from a set of examples. For example, Jiang, Munger, and Bunke introduced the concepts of the *median graph* and the *set median graph* [25]. The set median of a set of graphs is defined as that graph, drawn from the set, whose sum distance to the other members (graphs) of the set is minimized, while the median, a more general concept, is not constrained to come from the set. Choosing a prototypical graph from a set of graphs is an important problem in view-based object recognition, in which a prototypical graph must be chosen to represent a region on the viewing sphere [38], [13], [49]. Jiang et al. [25] proposed a genetic algorithm for computing the generalized median, while Luo et al. have explored the related problem of graph clustering using a spectral embedding of graphs [37]. It is important to note that these approaches assume that graphs belonging to the same class are structurally similar, and do not accommodate the many-to-many correspondences common to exemplars belonging to a single class.

D. What's Ahead

In the following sections, we begin by presenting a detailed formulation of our problem and conclude that its solution is computationally intractable. Next, we proceed to reformulate our problem by focusing on deriving abstractions from pairs of input images through a top-down procedure that draws on our previous work in generic 2-D shape matching. Given a set of pairwise abstractions, we present a novel method for combining them to form an approximation to the solution of our original formulation. We demonstrate the approach by applying it to subsets of images belonging to known classes.

II. Problem Formulation

Returning to Figure 3, let us now formulate our problem more concretely. As we stated, each input image is processed to form a region adjacency graph (we employ the region segmentation algorithm of Felzenzwalb and Huttenlocher [19]). Let us now consider the region adjacency graph corresponding to one input image. We will assume, for now, that our region adjacency graph represents an oversegmentation of the image. (In section VI, we will discuss the problem of undersegmentation, and how our approach can accommodate it.) The space of all possible region adjacency graphs formed by any sequence of merges of adjacent regions will form a lattice, as shown in Figure 4. The lattice size is exponential in the number of regions obtained after initial oversegmentation.[6] Kropatsch [32] has studied the problem of structure-preserving graph contraction, leading to techniques that can be used for generating such a lattice.

Each of the input images will yield its own lattice. The bottom node in each lattice will be the original region adjacency graph. In all likelihood, if the exemplars have different shapes (within-class deformations) and/or surface markings, the graphs forming the bottom of their corresponding lattices may bear little or no resemblance to each other. Clearly, similarity between the exemplars cannot be ascertained at this level, for there does not exist a one-to-one correspondence between the "salient" features (i.e., regions) in one graph and the salient features in another. On the other hand, the top of each exemplar's lattice, representing a silhouette of the object (where all regions have been merged into one region), carries little information about the salient surfaces of the object.

We can now formulate our problem more precisely, recalling that a lattice consists of a set of nodes, with each node corresponding to an entire region adjacency graph. Given N input image exemplars,

[6]Indeed, considering the simple case of a long rectangular strip subdivided into $n+1$ adjacent rectangles, the first pair of mergeable adjacent regions can be selected in n ways, the second in $n-1$, and so on, giving a lattice size of $n!$.

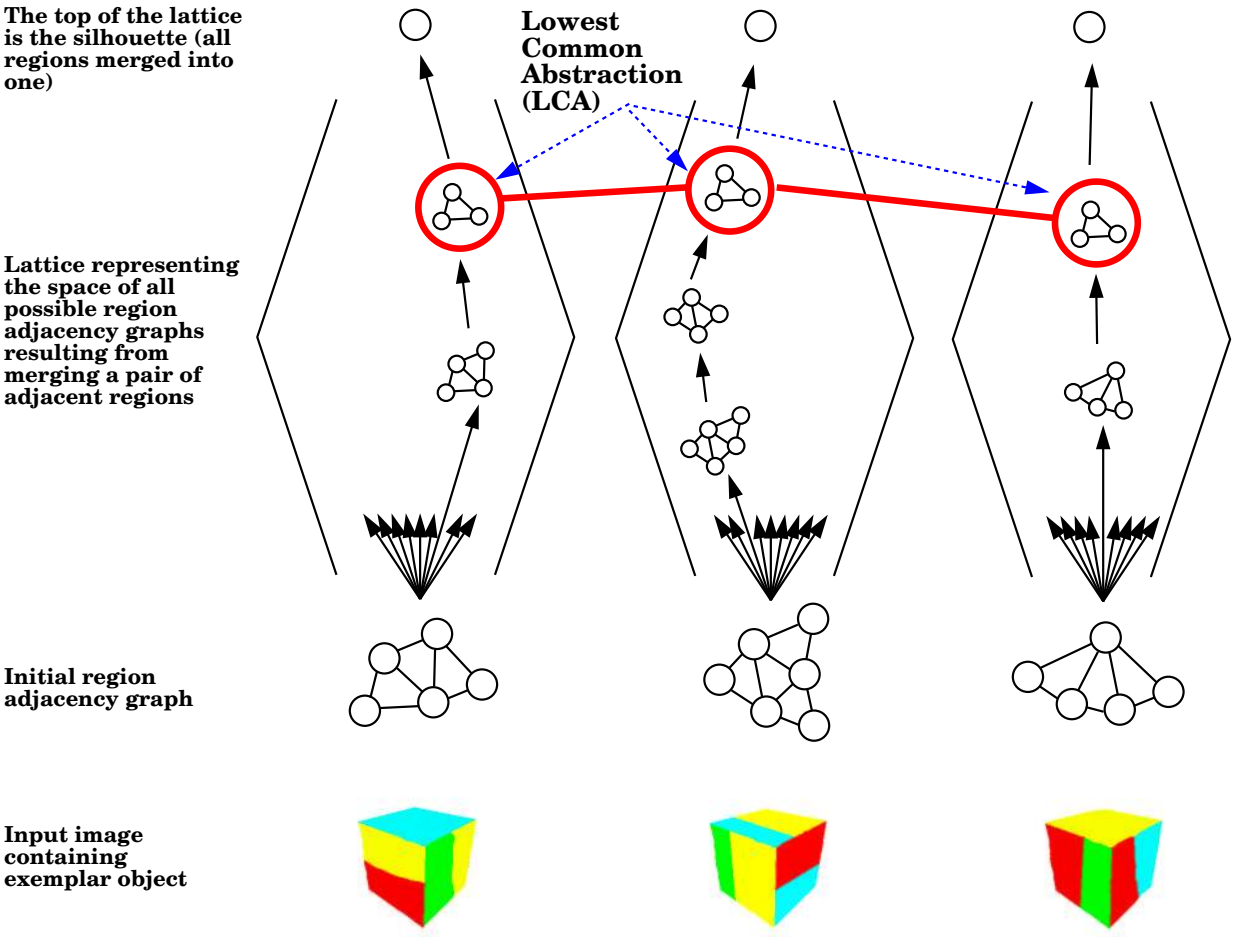


Fig. 4. The Lowest Common Abstraction (LCA) of a set of input exemplars.

$E_1, E_2, \ldots, E_N$, let $L_1, L_2, \ldots, L_N$ be their corresponding lattices, and for a given lattice, L_i, let $L_i n_j$ be its constituent nodes, each representing a region adjacency graph, G_{ij}. We define a *common abstraction*, or CA, as a set of nodes (one per lattice) $L_1 n_{j_1}, L_2 n_{j_2}, \ldots, L_N n_{j_N}$ such that for any two nodes $L_p n_{j_p}$ and $L_q n_{j_q}$, their corresponding graphs G_{pj_p} and G_{qj_q} are isomorphic. Thus, the root node (whose graph consists of one node representing the silhouette region) of each lattice is a common abstraction. We define the *lowest common abstraction*, or LCA, as the common abstraction whose underlying graph has maximal size (in terms of number of nodes). Given these definitions, our problem can be simply formulated as finding the LCA of N input image exemplars.

Intuitively, we are searching for a node (region segmentation) that is common to every input exemplar's lattice and that retains the maximum amount of structure common to all exemplars. If all the initial exemplars are oversegmented, the desired abstraction can be reached from each initial exemplar by a

path corresponding to a sequence of region merges. Unfortunately, the presence of a single, heavily undersegmented exemplar (a single-node silhouette in the extreme case) will drive the LCA toward the trivial silhouette CA. In a later section, we will relax our LCA definition to make it less sensitive to such outliers, effectively allowing an abstraction to be reached from each initial exemplar by a path corresponding to a sequence of region merges *and region splits.*

III. The LCA of Two Examples

For the moment, we will focus our attention on finding the LCA of two lattices, while in the next section, we will accommodate any number of lattices. Since the input lattices are exponential in the number of regions, actually computing the lattices is intractable.[7] Clearly, we need a means for focusing the search for the LCA that avoids significant lattice generation. Our approach will be to restrict the search for the LCA to the *intersection* of the lattices. Typically, the intersection of two lattices is much smaller than either lattice (unless the images are very similar), and leads to a tractable search space. But how do we generate this new "intersection" search space without enumerating the lattices?

Our solution is to work top-down, beginning with a node known to be in the intersection lattice – the root node, representing a single region (silhouette). If the intersection lattice contains only this one node, i.e., one or both of the region segmented images contain a single region, then the process stops and the LCA is simply the root (silhouette). However, in most cases, the root of each input lattice is derived from an input region adjacency graph containing multiple regions. So, given two silhouettes, each representing the apex of a separate, non-trivial lattice, we have the opportunity to search for a lower abstraction (than the root) common to both lattices. Our approach will be to find a decomposition of each silhouette region into two subregions, such that: 1) the shapes of the corresponding subregions are similar, and 2) the relations among the corresponding regions are similar. Since there are an infinite number of possible decompositions of a region into two component regions, we will restrict our search to the space of decompositions along region boundaries in the original region adjacency graphs. Note that there may be multiple 2-region decompositions that are common to both lattices; each is a member of the intersection set.

The process is illustrated in Figure 5. At the top are the silhouettes of two blocks with their region adjacency graphs overlaid. Shown in red is the best pair of corresponding cuts in the two region adjacency

[7]Thus, any approach that examines significant portions of the lattices, including Frequent Itemset [23], will be computationally infeasible.

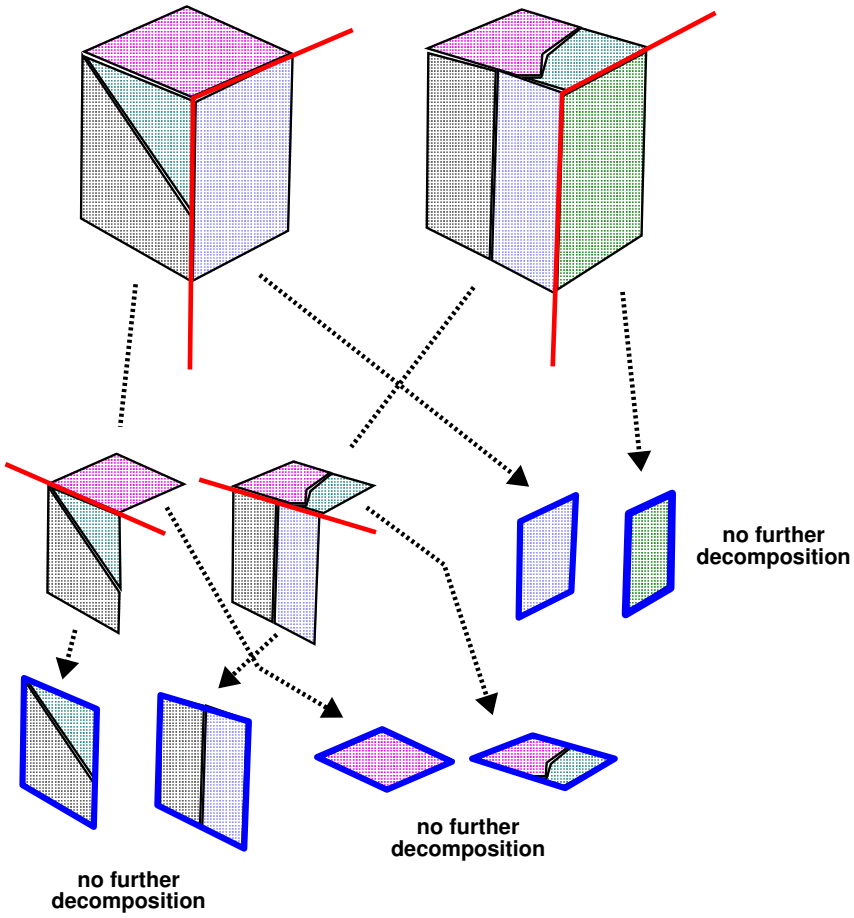


Fig. 5. Finding the lowest common abstraction between two exemplars through a coordinated, recursive decomposition of their silhouettes.

graphs, such that the two regions to the left of the cuts are similar in shape, the two regions to the right of the cuts are similar in shape, and the relations between the pairs of regions spanning the cuts are similar. The process is repeated recursively, with the left pair of regions decomposed once more, until no further corresponding cuts can be made. The two regions to the right of the top-level cuts are already primitive and cannot be further decomposed. The union of the primitive (blue) regions forms the lowest common abstraction between the two region adjacency graphs. Note that the decomposition is not necessarily unique.

Assuming that we have some means for ranking the matching decompositions (if more than one exists), we pick the best one (the remainder constituting a set of backtracking points), and recursively apply

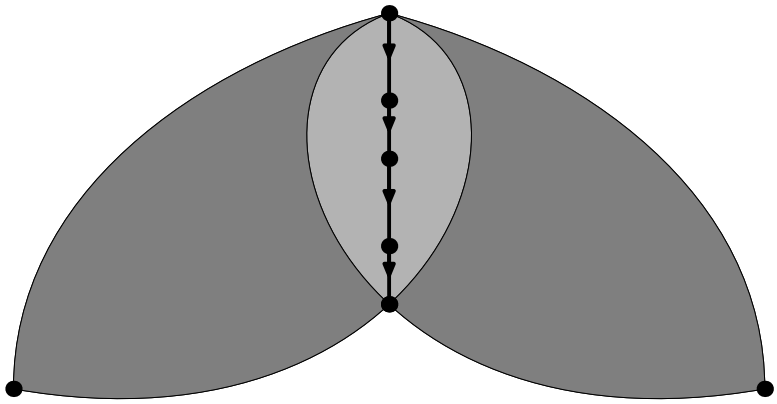


Fig. 6. Abstraction lattices and their intersection. The abstraction lattices of two region adjacency graphs are shown in dark gray. Their intersection, which is significantly smaller, is shown in light gray. To find the lowest common abstraction, we start at the top of the intersection set and work our way down until no further common decompositions are found.

the process to each pair of isomorphic component subregions.[8] The process continues in this fashion, "pushing" its way down the intersection lattice, until no further decompositions are found. This lower "fringe" of the search space represents the LCA of the original two lattices. In the following subsections, we will formalize this process and examine our approach in detail. An illustration is given in Figure 6.

A. Problem Definition

We define graph H to be an immediate decomposition of graph G if G can be obtained from H by merging two nodes. Let L_1 and L_2 be two lattices, and let $G_1 \in L_1$ and $G_2 \in L_2$ be two graphs that are isomorphic. G_1 (or G_2, since they are sufficiently similar) is therefore in the intersection of L_1 and L_2. Two graphs, $H_1 \in L_1$ and $H_2 \in L_2$, are common immediate decompositions of G_1 and G_2, respectively, if they are isomorphic immediate decompositions of the respective graphs. Thus, our problem can be formulated as follows: Given a pair of isomorphic graphs G_1 and G_2 in L_1 and L_2, respectively, find a pair of isomorphic immediate decompositions of G_1 and G_2, denoted by $H_1 \in L_1$ and $H_2 \in L_2$, if such a pair exists.

[8]Each subregion corresponds to the union of a set of regions corresponding to nodes belonging to a connected subgraph of the original region adjacency graph.

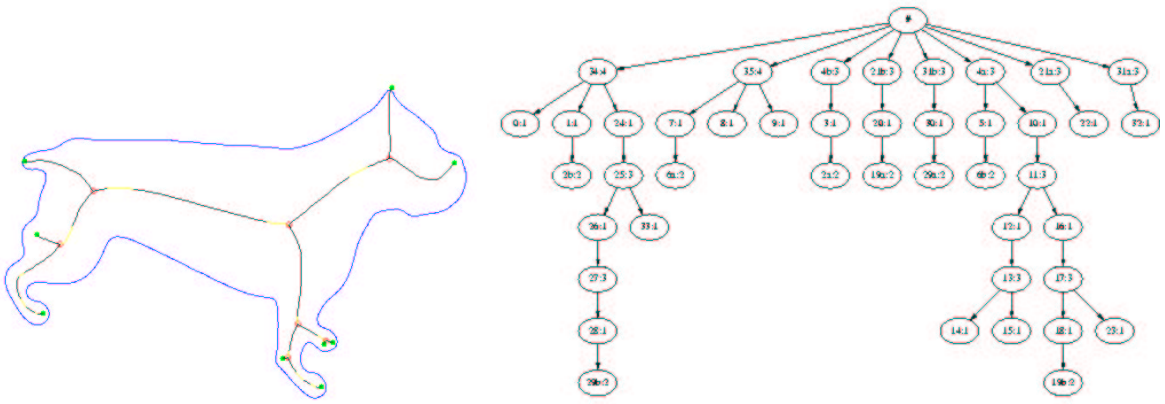


Fig. 7. An illustrative example of a silhouette (left) and its shock graph (right).

B. 2-D Shape Similarity

Two decompositions (in general, two region adjacency graphs) are isomorphic if their corresponding regions have similar shapes and similar relations. For corresponding regions, it is imperative that we define a similarity metric that accounts for coarse shape similarity. Since the exemplars are all slightly different, so too are the shapes of their abstracted regions. To compute the coarse shape distance between two regions, we draw on our previous approach to generic 2-D object recognition that is efficient and robust to occlusion and noise [51]. Specifically, a silhouette (or region boundary, in our case) is decomposed into a set of qualitative parts based on a coloring of the shocks (singularities) of a curve evolution process acting on simple closed curves in the plane [31]. Intuitively, the taxonomy of shocks consists of four distinct types: the radius function along the medial axis varies monotonically at a 1, achieves a strict local minimum at a 2, is constant at a 3 and achieves a strict local maximum at a 4. We have abstracted this system of shocks into a *shock graph* where vertices are labeled by their shock types, and the shock formation times direct the edges (see Figure 7). The space of such shock graphs is completely characterized by a small number of rules which, in turn, permits the reduction of each graph to a *unique rooted tree*. In recent work, we developed an algorithm for matching two shock trees based on both topological structure and geometric structure [51].

For relational (or arc) similarity, we must check the relational constraints imposed on pairs of corresponding regions. Such constraints can take the form of relative size, relative orientation, or degree of boundary sharing. We implicitly check the consistency of these pairwise constraints by computing the shape distance (using the same distance function referred to above) between the union of the two regions

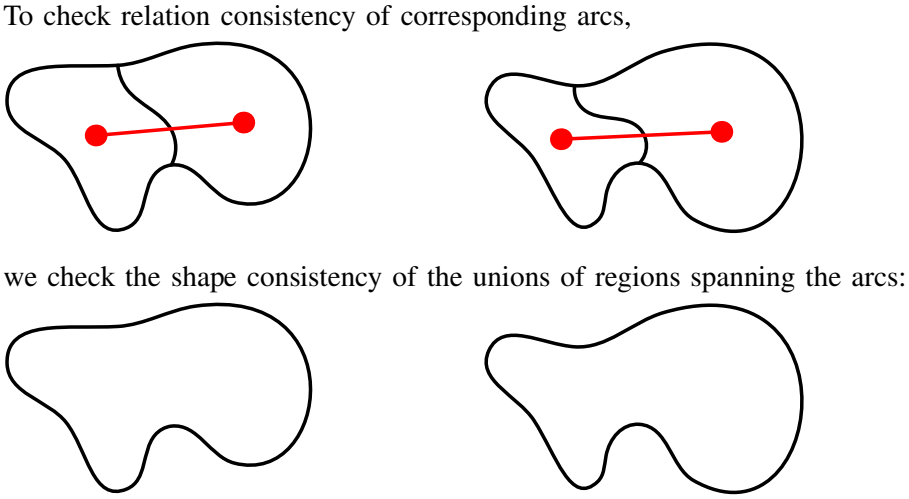


Fig. 8. Checking the relation between two regions. The relation is said to be satisfied if the corresponding regions match *and* their respective unions match.

forming one pair (i.e., the union of a region and its neighbor defined by the arc) and the union of the two regions forming the other, as shown in Figure 8. If the constraints are satisfied, the distance will be small. In our recursive decomposition framework, the relation between two regions is accounted for by the fact that the parents (i.e., the unions) have already been successfully matched. If the corresponding components match, their relational constraints are therefore satisfied, by definition.

C. A Shortest Path Formulation

The decomposition of a region into two subregions defines a cut in the original region adjacency subgraph defining the region. Unfortunately, the number of possible 2-region decompositions for a given region may be large, particularly for nodes higher in the lattice.[9] One way we can reduce the complexity is to restrict our search for cuts that span two points on the region's boundary, i.e., cuts that don't yield regions with "holes." [10] Despite this restriction, the complexity is still prohibitive, and we need to take further measures to simplify our formulation.

[9]Consider, for example, a checkerboard image and its corresponding region adjacency graph. The root will be a single square region, but there will be *many* decompositions of this square region into two component regions because there are many ways the original checkerboard image can be divided into two along region boundaries. For a checkerboard graph with $(n+1)^2$ vertices, the number of monotonic paths from the upper left corner to the lower right corner is equal to the number of binary sequences of length n, which is exponential.

[10]This assumes that a "hole" in a region does not correspond to a salient model surface.

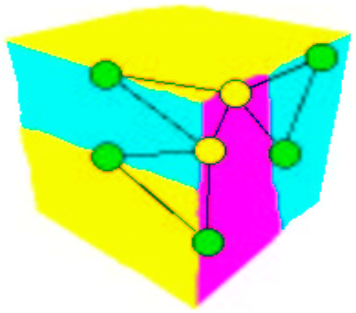


Fig. 9. A region decomposition and its dual boundary segment graph.

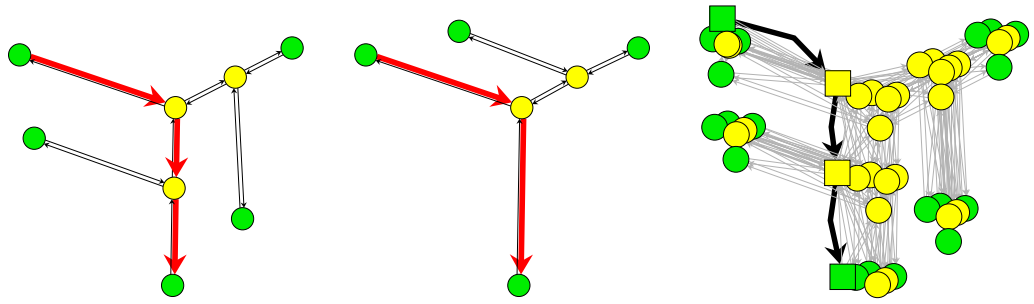
We begin by transforming our two region adjacency graphs into their dual *boundary segment graphs*, as shown in Figure 9. A boundary segment graph of a region adjacency graph has internal (i.e., common to two original regions) boundary segments as its nodes, and an edge from boundary segment b_1 to b_2 if b_1 and b_2 share an endpoint. Note that if region A lies entirely within region B, the single boundary segment that is common to A and B is not included in the dual graph. This is due to the fact that we are looking for cuts that yield regions without holes. In fact, from the true dual graph, we consider only the subgraph that is spanned by nodes that are connected to a node on the silhouette's boundary. Note also that nodes in this subgraph can potentially be connected by multiple edges, corresponding to multiple boundary fragments between adjacent regions. This case does not present any difficulties for subsequent stages of our approach.

The transformation to the dual graph allows us to reformulate the search for corresponding cuts in two region adjacency graphs as a search for corresponding paths in their boundary segment graphs. However, this has not affected the complexity of our problem, as there could be an exponential number of paths in each dual graph (recall our checkerboard example). Rather than enumerating the paths in each dual graph and then enumerating all pairs, we will cast our problem as a search for the shortest path in a *product graph* of the two dual graphs.

The product graph $G = G_1 \times G_2 = (V, E)$ of graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ is defined as follows:

$$V = \{\ (v_1, v_2) :\ v_1 \in V_1,\ v_2 \in V_2\ \} = V_1 \times V_2$$

$$\begin{aligned} E = {} & \{\ ((u_1, u_2), (v_1, v_2)) :\ (u_1, v_1) \in E_1,\ (u_2, v_2) \in E_2\ \} \ \cup \\ & \{\ ((v_1, u_2), (v_1, v_2)) :\ v_1 \in V_1,\ (u_2, v_2) \in E_2\ \} \ \cup \\ & \{\ ((u_1, v_2), (v_1, v_2)) :\ (u_1, v_1) \in E_1,\ v_2 \in V_2\ \} \end{aligned}$$



(a) (b)

Fig. 10. The product graph. The product graph of the two graphs in (a) is shown in (b). The pair of red paths shown in the original graphs corresponds to the black path shown in the product graph.

The node set of the product graph is the product of the node sets of the initial graphs. To define the edge set of the product graph correctly, notice that the simple product of the edge sets (the first term in the union) may result in a disconnected graph. The other two terms of the union (which can be viewed as the product of the node set of one graph with the edge set of the other graph) ensure that the product of two connected graphs will be connected. The essential property of the product graph that we will exploit is that a simple path $(u_1,v_1) \rightarrow (u_2,v_2) \rightarrow \cdots \rightarrow (u_n,v_n)$ in the product graph corresponds to two sequences of nodes in the initial dual graphs, $u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_n$ and $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n$ which, after the elimination of successive repeated nodes, will result in two simple paths (whose lengths may be different) in the initial dual graphs, as shown in Figure 10.

Each path in our product graph corresponds to a pair of possible cuts in two regions. Consequently, our goal is to find a path that yields the best matching subregions and relations. Here is where we face a problem. This objective function can only be evaluated once a complete path is found, since only a complete path defines a pair of closed regions whose shapes and relations can be matched. However, a given edge in the product graph represents a pair of corresponding boundary fragments from two regions. Globally, we are comparing regions, while locally, we are comparing contours. How then do we define local edge weights and a local objective function so that a shortest path algorithm will yield an approximation to the global optimum?

Let us begin with the edge weights. If we align the two regions (from which we seek corresponding cuts) through our region matching algorithm [51], then we can assume that both external and internal boundary segments are approximately aligned. Under this assumption, edge weights in the product graph can be

chosen to reflect the shape similarity of their component boundary segments. In our implementation, to compute edge weights, we employ a simple Hausdorff-like distance between the two boundary segments, yielding a local approximation to the global similarity of the regions.

In our dual graph, smaller edge weights correspond to pairs of more similar boundary segments. This leads to a number of very natural choices for an objective function, if we interpret edge weights as edge lengths. The total path length, $tl(p) = \sum_{p_i \in p} l(p_i)$, is a well-studied objective function [12]. Fast algorithms for generating successive (i.e., next optimal) shortest and simple shortest paths are given in [17], [27]. However, the above objective function tends to prefer shorter paths over longer ones, assuming path edges are of equal average length. For our problem, smaller paths will result in smaller regions being cut off, which is contrary to our goal of finding the lowest common abstraction.[11]

To overcome this problem, we turn to a different objective function that measures the maximum edge weight on a path, $ml(p) = \max_{p_i \in p} l(p_i)$. A well-known modification[12] of Dijkstra's algorithm [12] finds paths of minimal maximum edge weight (minmax paths) between a chosen node and all other graph nodes, and has the same complexity, $O(|E| + |V| \log |V|)$, as the original algorithm. However, efficient algorithms for finding successive (next optimal) minmax paths are not readily available. Leaving development of such an algorithm for the future, we will employ a mixed strategy. Namely, we find pairs of nodes providing near-optimal values of the minmax objective function and, along with the minmax path between the nodes, we also generate several successive shortest paths between them. For this, we use Eppstein's algorithm [17], which generates k successive shortest paths between a chosen pair of nodes in $O(|E| + |V| \log |V| + k \log k)$ time. The mixed strategy, whose overall complexity is $O(|V|(|E| + |V| \log |V|))$ for small k, has proven to be effective in empirical testing.

D. Algorithm

Having defined the edge weights and objective function, we can now summarize our algorithm for finding the "best" common decomposition of two abstraction nodes, as shown in Algorithm 1. This algorithm, in turn, is embedded in our solution to the problem of finding the LCA of two examples, which computes an approximation to the intersection of their respective lattices in a top-down manner. Beginning with the two root nodes (the sole member of the initialized intersection set), we recursively seek the "best" common decomposition of these nodes, and add it to the intersection set. The process is

[11] A small region is unlikely to be common to many input exemplars.

[12] Instead of summing up edge weights when determining the distance to a node, it takes their maximum.

Algorithm 1 A generic algorithm for finding a common decomposition.

Let A_1, A_2 be subgraphs of the original region adjacency graphs that correspond to isomorphic vertices of the abstraction graphs.
Let G_1, G_2 be dual graphs of A_1, A_2.
Form the product graph $G = G_1 \times G_2$, as described above.
Choose an objective function f (see text for discussion), compute edge weights w_i (see text for discussion), and select a threshold $\varepsilon > 0$.
Let P_f be the optimal path with respect to $(f, \{w_i\})$ with value $F(P_f)$.
Let $P = P_f$
while $|f(P) - f(P_f)| < \varepsilon$ **do**
    Let P_1 and P_2 be the paths in G_1, G_2 corresponding to P.
    Let (V_1, W_1) and (V_2, W_2) be the resulting cuts in A_1, A_2
    if region V_1 is similar to region V_2, and region W_1 is similar to region W_2, and arcs (V_1, U_1^i), (V_2, U_2^i) are similar for all isomorphic neighbors U_1^i, U_2^i of V_1, V_2 respectively, and arcs (W_1, U_1^i), (W_2, U_2^i) are similar for all isomorphic neighbors U_1^i, U_2^i of W_1, W_2 respectively **then**
        output decompositions (V_1, W_1) and (V_2, W_2).
        return
    end if
    Let P be the next optimal path with respect to $(f, \{w_i\})$.
end while
output "no non-trivial decomposition is found".

recursively applied to each common decomposition (i.e., member of the intersection set) until no further common decompositions are found. The resulting set of "lowest" common decompositions represents the LCA of the two lattices. The description is formalized in Algorithm 2.

IV. The LCA of Multiple Examples

So far, we have addressed only the problem of finding the LCA of two examples. How then can we extend our approach to find the LCA of multiple examples? Furthermore, when moving toward multiple examples, how do we prevent a "noisy" example, such as a single, heavily undersegmented silhouette from derailing the search for a meaningful LCA? To illustrate this effect, consider the inputs (a)–(d) shown in Figure 11. If the definition of the pairwise LCA is directly generalized, thus requiring the search for

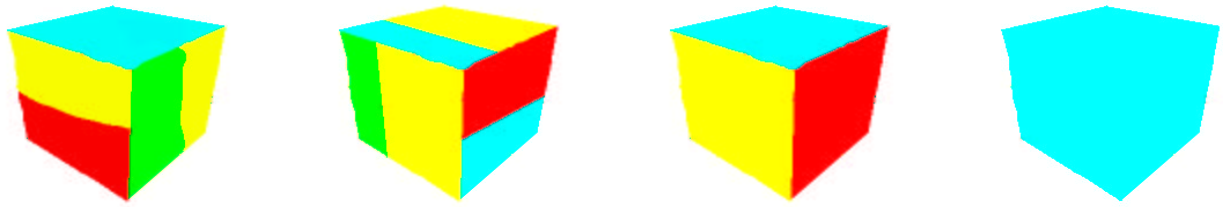
Algorithm 2 Finding the maximal common abstraction of two region adjacency graphs.

Let A_1, A_2 be the initial region adjacency graphs.
Let G_1, G_2 denote abstraction graphs belonging to abstraction lattices, L_1 and L_2 respectively.
Let G_1^0, G_2^0 be the topmost nodes of the lattices.
Let $G_1 = G_1^0$, $G_2 = G_2^0$.
while there are unexplored isomorphic nodes $u_1 \in G_1$, $u_2 \in G_2$ **do**
    Let U_1 and U_2 be the corresponding subgraphs of A_1, A_2.
    if there is a *common decomposition* $U_1 = V_1 \cup W_1$ and $U_2 = V_2 \cup W_2$ **then**
        Split the nodes $u_1 \in G_1$, $u_2 \in G_2$ by forming the *decomposition graphs* $H_1 = (G_1 - \{u_1\}) \cup \{v_1, w_1\}$, $H_2 = (G_2 - \{u_2\}) \cup \{v_2, w_2\}$ with edges established using A_1, A_2.
        Let $G_1 = H_1$, $G_2 = H_2$.
    else
        Mark u_1 and u_2 as explored.
    end if
end while
output G_1, G_2.

an element common to all abstraction lattices, the correct answer will be the input (d). However, much useful structure is apparent in inputs (a)–(c); input (d) can be considered to be an outlier.

To extend our two-exemplar LCA solution to a robust, multi-exemplar solution, we begin with two important observations. First, the LCA of two exemplars lies in the intersection of their abstraction lattices. Thus, both exemplar region adjacency graphs can be transformed into their LCA by means of sequences of region merges. Second, the total number of merges required to transform the graphs into their LCA is minimal among all elements of the intersection lattice, i.e., the LCA lies at the lower fringe of the lattice.

Our solution begins by constructing an approximation to the intersection lattice of multiple exemplars. Consider the closure of the set of the original region adjacency graphs under the operation of taking pairwise LCA's. In other words, starting with the initial region adjacency graphs, we find their pairwise LCA's, then find pairwise LCA's of the resulting abstraction graphs, and so on (note that duplicate graphs are removed). We take all graphs, original and LCA, to be nodes of a new *closure* graph. If graph H was obtained as the LCA of graphs G_1 and G_2, then directed arcs go from nodes corresponding to G_1,



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

Fig. 11. The straightforward computation of the Lowest Common Abstraction of exemplars (a)–(d) gives the exemplar in (d). However, (c) is the Lowest Common Abstraction of exemplars (a)–(c), and therefore is more representative.

G_2 to the node corresponding to H in the closure graph.

Next, we will relax the first property above to accommodate "outlier" exemplars, such as undersegmented input silhouettes. Specifically, we will not enforce that the LCA of multiple exemplars lie in the intersection set of *all* input exemplars. Rather, we will choose a node in our approximate intersection lattice that represents a "low abstraction" for many (but not necessarily all) input exemplars. More formally, we will define the LCA of a set of exemplar region adjacency graphs to be that element in the intersection of two or more abstraction lattices that minimizes the total number of edit operations (merges or splits) required to obtain the element from *all* the given exemplars. If a node in the intersection lattice lies along the lower fringe with respect to a number of input exemplars, then its sum distance to all exemplars is small. Conversely, the sum distance between the silhouette outlier (in fact, the true LCA) and the other input exemplars will be large, eliminating that node from contention.

Note that a graph may not be directly linked to all of its abstractions in the closure graph. However, if H is an abstraction of G, then there is a directed path between the nodes corresponding to G and H. Thus, any abstraction is reachable from any of its decompositions by a directed path. Such a path may move upwards from an exemplar through zero or more successive region merges to a node in the closure graph, then downwards through zero or more successive regions splits to reach the abstraction. Each edge in the closure graph is assigned a weight equal to the merge edit distance that takes the decomposition to the abstraction. The edit distance is simply the difference between the numbers of nodes in the decomposition graph and the abstraction graph. As a result, we obtain a weighted directed acyclic graph. An example of such a graph, whose edges are shown directed from region adjacency graphs to their LCA's, is given in Figure 12.

Given such a graph, the robust LCA of *all* inputs will be that node that minimizes the sum of shortest

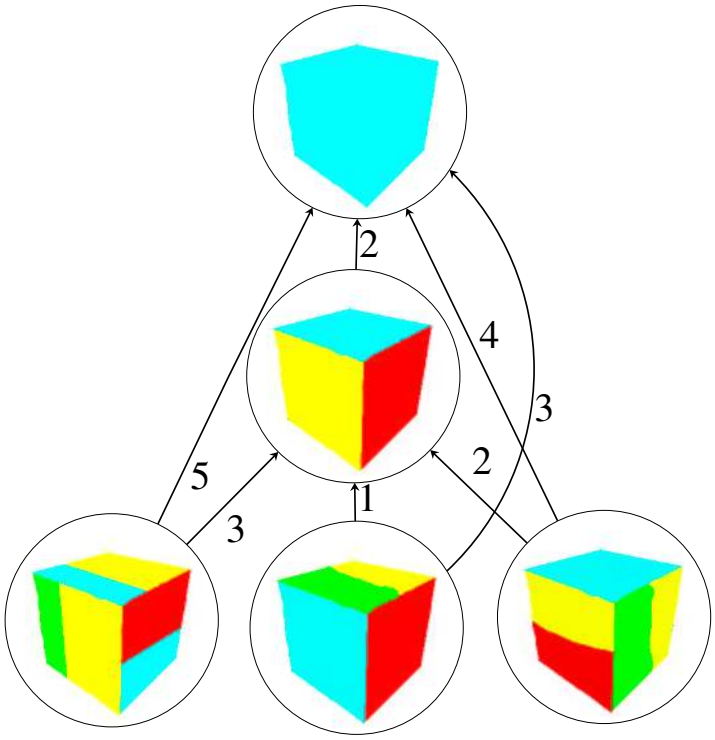


Fig. 12. Embedding region adjacency graphs and their pairwise LCA's in a weighted directed acyclic graph. The three nodes at the bottom, as well as the undersegmented "outlier" at the top are the four input exemplars. Although the true LCA is the top node, the center node is the median, as its distance sum value is $3+1+2+2=8$, while the sum is $5+3+4+0=12$ for the topmost node.

path distances from the initial adjacency graphs. In other words, we are looking for the "median" of the graph, as computed by Algorithm 3. Note that the closure graph is an approximation to the intersection lattice. On one hand, it may contain pairwise LCA's which are not contained in the intersection lattice, while on the other hand, it may not contain nodes in the intersection lattice that are not LCA's. While it will contain the true LCA, our median formulation may lower the LCA fringe below the true LCA. Note that the distance from an outlier graph to the median graph is not necessarily large. For example, in Figure 12, the distance from the top outlier node to the median is less than the distance from the bottom left node to the median. Hence, traditional statistical methods of outlier detection do not readily apply for within-class outlier detection. On the other hand, a recent adaptation of statistical techniques to the domain of strings [26] can be helpful in the case of multiple object classes.

To analyze the complexity of the algorithm, notice that the first step, i.e., finding the distance sum to

Algorithm 3 Finding the median of the closure graph.

1: Let the *sink node*, s, be the topmost node in the closure graph.
2: Solve the "many-to-one" directed shortest path problem on the graph with the source nodes being the original adjacency graphs and with the specified edge weights. Find the distance sum, $DS(s)$, for the sink node.
3: Similarly, find distance sums, $DS(s_i)$, for all unexplored $s_i \in N(s)$.
4: **if** $min_i(DS(s_i)) \geq DS(s)$ **then**
5: **return** s
6: **else**
7: Let $s = \arg\min_i DS(s_i)$.
8: **goto** 2.
9: **end if**

the topmost node, can be performed in linear time in the graph size, since the closure graph is a directed acyclic graph, and the single source shortest path problem in such graphs can be solved in $O(|V| + |E|)$ time [12]. Since the algorithm can potentially examine a constant fraction of the graph nodes (consider the case of a line graph), the total running time can be as high as $O(|V|(|V| + |E|))$. The average case complexity will depend on the particular distribution of the initial data and is beyond the scope of this paper. In practice, the algorithm stops after a few iterations.

V. Experiments

In this section, we apply our approach to three different object domains. The domains of coffee cups, books, and dispenser bottles were chosen since they can be effectively modeled as collections of surfaces in 3-D, different exemplars belonging to a class exhibit minor within-class shape deformation, and the different exemplars have significantly different low-level appearance in terms of color, texture, and surface markings. In Figures 13 and 14, we illustrate the results of our approach applied to two sets of three coffee cup images, respectively. In each case, the lower row represents the original images, the next row up represents the input region segmented images (with black borders), while the LCA is shown with an orange border.[13] In each case, the closure graph consists of only four members, with the same pairwise

[13]All region segmented images are the actual outputs of the implementation that accompanied [19]. In particular, smaller regions are sometimes subsumed by larger regions, resulting in non-intuitive segmentations.

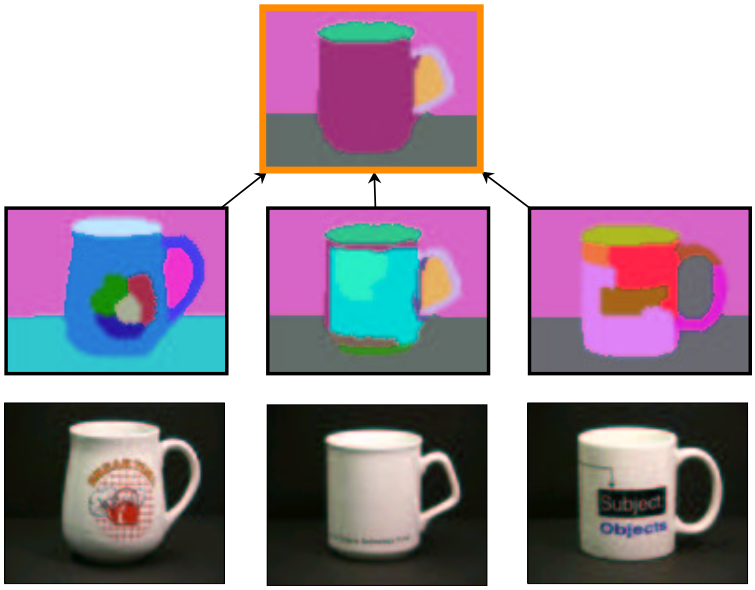


Fig. 13. Computed LCA (orange border) of 3 examples.

LCA emerging from all input pairs. While in Figure 13, the solution captures our intuitive notion of the cup's surfaces, the solution in Figure 14 merges the regions corresponding to the surfaces defined by the cup's body and handle. A strip along the bottom is present in each exemplar, and understandably becomes part of the solution. The same is true of the elliptical region at the top of each exemplar. However, due to region segmentation errors, the larger blue region in the middle cup extends into the handle. Consequently, a cut along its handle (as is possible on the other cups) is not possible for this exemplar, resulting in a "stopping short" of the recursive decomposition at the large white region in the solution (LCA).

In Figure 15, we again present three exemplars to the system. In this case, the closure graph has many nodes. Unlike Figure 13 and 14, in which all pairwise LCA's were equal (leading to a somewhat trivial solution to our search for the global LCA), each pair of input exemplars leads to a different LCA which, in turn, leads to additional LCA's. Continuing this process eventually results in the inclusion of the silhouette in the closure graph. The solution, according to our algorithm, is again shown in orange, and represents an effective model for the cup.

To test our approach more systematically, we applied it to subsets of the cup images shown in Figure 16. Specifically, we randomly selected 4 subsets, 3 to 4 images each. A typical result is shown in Figure 17. Based on the results of this and other experiments, it turned out that all but one segmentation had the

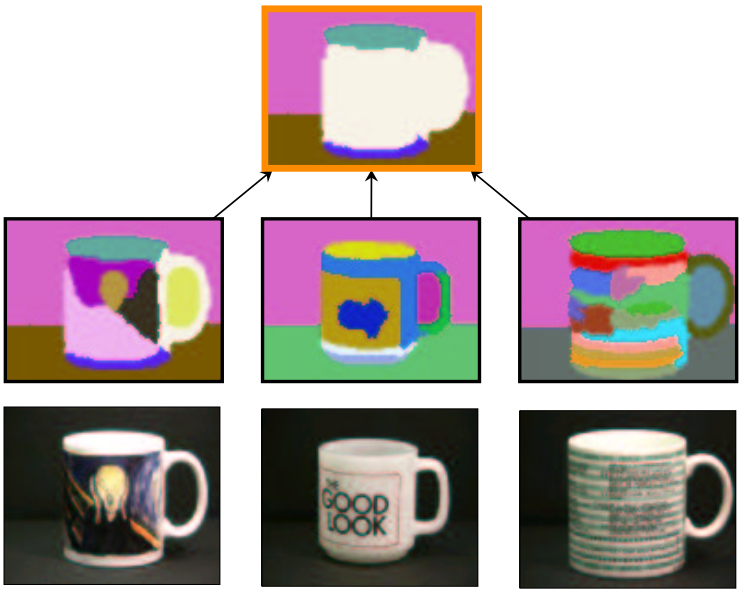


Fig. 14. Computed LCA (orange border) of 3 examples.

four-image LCA as its abstraction, while no other region adjacency graph was an abstraction of more than 2 exemplars. This means that the LCA of these four cup images is also the LCA of the whole set.

This observation suggests another approach to finding the abstraction of a large number of exemplars: generate the LCA of a small subset (2–4) of exemplars and check (by finding pairwise LCA's) whether the LCA is an abstraction of many other exemplars. It can be shown that an abstraction of more than half the initial exemplars is also an abstraction of their LCA. Moreover, the largest such "small subset" LCA (found, for example, by exhaustive search among all "small subset" LCA's) will be isomorphic to the one found with the current exhaustive approach. This alternative sample-based approach (similar to robust statistical estimation or to learning from representative samples) will not require the computation of the full closure graph, which is the main bottleneck of our current approach. Its implementation is left for future work.

To illustrate the fact that the LCA of the entire set of exemplars can be equal to the LCA of just two exemplars, we turn to the domain of books, a set of 8 exemplars of which is shown in Figure 18; as before, for each book, we include the original image and the segmented version that serves as input to our algorithm. In Figure 19, the computed LCA (orange border), which is a pairwise LCA of the left two exemplar region adjacency graphs, is in fact the LCA of the entire 8-image set. This is due to the fact that a stripe-like region at the top of the "horizontal" surface patch is present in all but 2

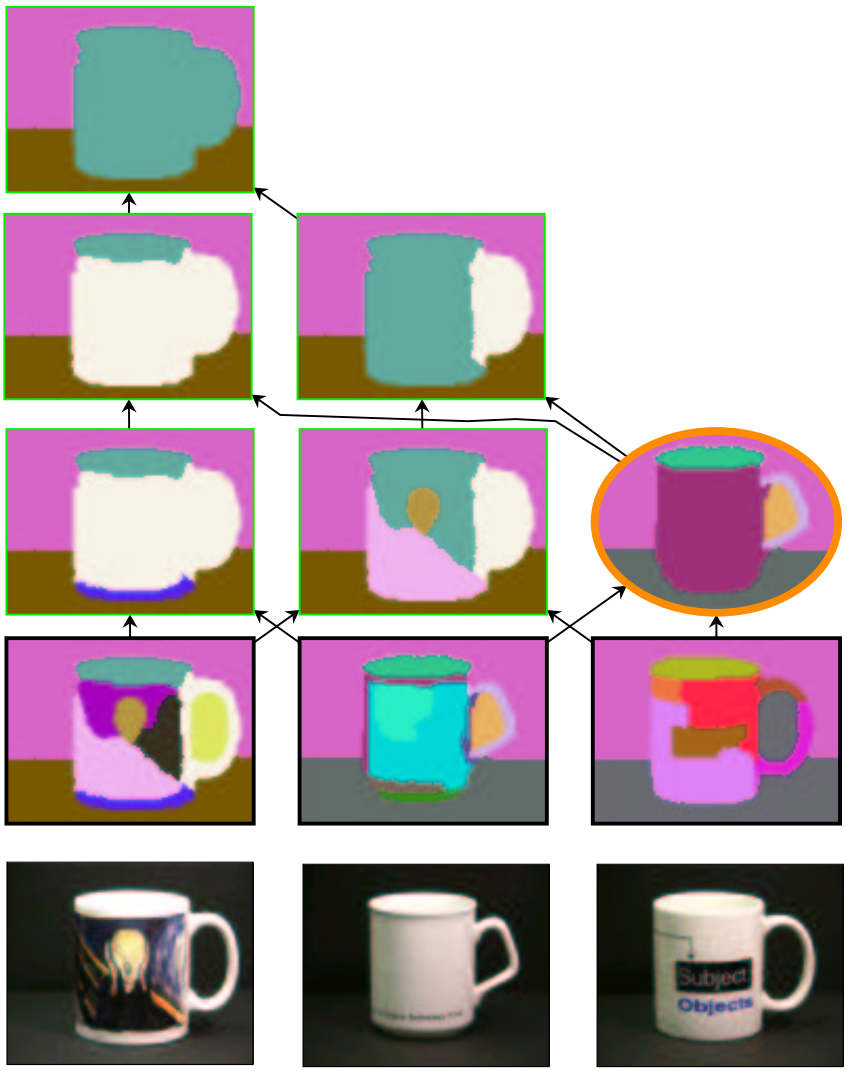


Fig. 15. Computed LCA (orange border) of 3 examples. Each non-leaf node in the closure graph is the LCA of the two nodes from which edges are directed to it.

book images. On the other hand, all other regions on the "horizontal" surface patch of the book model disappear once sufficiently many exemplars are shown to the system. The resulting LCA suggests that many such computer science texts share a horizontal stripe at the top of one of their covers.

To illustrate the fact that the LCA of a relatively large subset (in this case, 3 of 8) of input exemplars may not necessarily be the LCA of the entire set (of 8), consider the computed LCA of the three book exemplars shown in Figure 20. Due to the presence of a heavily over-segmented exemplar (the rightmost node), the resulting LCA (orange border) is skewed toward having too many regions. The apparent

Fig. 16. The set of 8 cup images (and their region segmentations) used in the experiments.

regularity of such structure renders it salient. This "skew" can be corrected by either choosing a different subset of images or by adding more exemplars to the subset. To illustrate the effect on the LCA of adding exemplars, consider the domain of dispenser bottles, as shown in Figure 21. The closure graphs of sets of 3 and 4 dispenser bottles are shown in Figure 22 and Figure 23, respectively. In Figure 22, even though the desired 3-region LCA of the three exemplars appears in the closure graph (top node), the computed LCA is the undesirable 4-region abstraction. This is due to the fact that the accidental alignment of non-salient structure in the right two exemplars renders the leftmost exemplar an "outlier". The addition of a fourth exemplar overcomes this problem, leading to the desired LCA, as shown in Figure 23.

In concluding this section, it is worth addressing the issue of background in the model acquisition process. As is evident in the above experiments, all exemplars were imaged against simple, mostly uniform backgrounds (in some cases, the table on which the cup was sitting was segmented from the background). What if each exemplar appeared against a cluttered background? If the cluttered background was consistently seen in many of the images, the algorithm would, of course, not be able to distinguish this regularly occurring structure from that of the object. If the backgrounds were inconsistent, and we

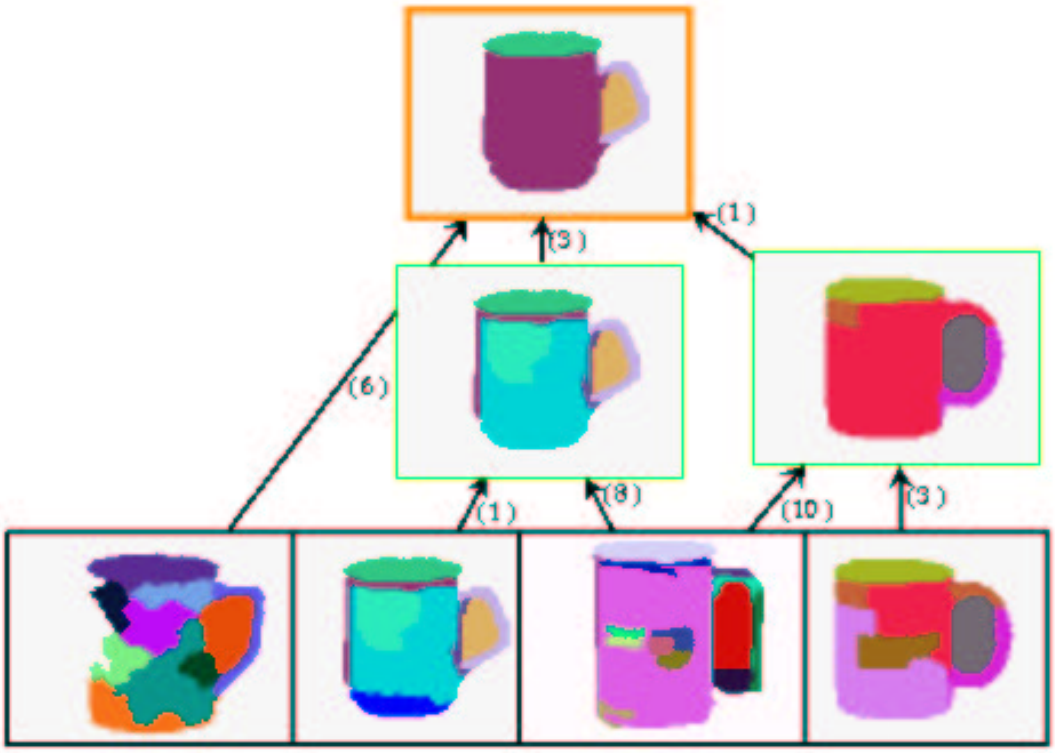


Fig. 17. Computed LCA (orange border) of 4 examples. Each non-leaf node in the closure graph is the LCA of the nodes from which edges are directed to it. Each edge label indicates the edit distance between the region adjacency graphs corresponding to the two nodes that span the edge. The computed LCA is an abstraction of 7 out of 8 initial exemplars.

could assume that the object's silhouettes were the most salient regions in the images, then a pair of large, similar regions could be extracted from a pair of images by an application of the product graph technique as follows. Recalling that a region in the image corresponds to a cycle in the dual boundary segment graph, a pair of similar regions corresponds to a pair of cycles in the dual graphs which, in turn, corresponds to a single "optimal" cycle in the product graph. To find the optimal cycle in the product graph, we can apply the minimum mean cycle algorithm [12]. Once the foreground objects have been separated from their backgrounds, our "standard" procedure can be applied.

VI. Conclusions

The quest for generic object recognition hinges on an ability to generate abstract, high-level descriptions of input data. This process is essential not only at run-time, for the recognition of objects, but also at compile time, for the automatic acquisition of generic object models. In this paper, we address the latter problem – that of generic model acquisition from examples. We have introduced a novel formulation

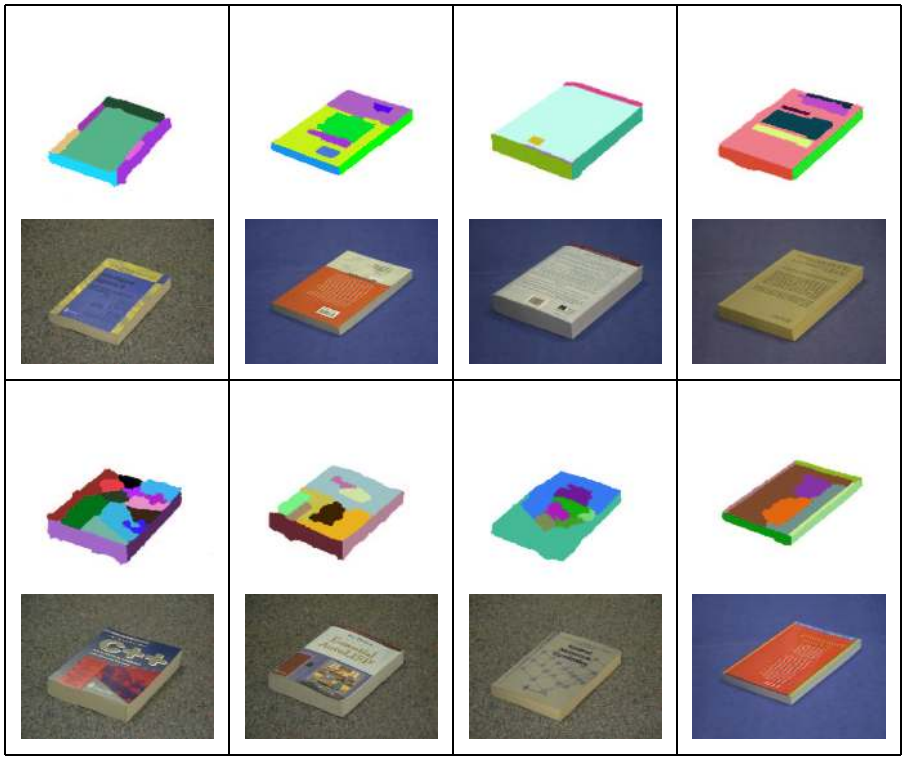


Fig. 18. The set of 8 book images (and their region segmentations) used in the experiments.

of the problem, in which the model is defined as the lowest common abstraction of a number of segmentation lattices, representing a set of input image exemplars. To manage the intractable complexity of this formulation, we focus our search on the intersection of the lattices, reducing complexity by first considering pairs of lattices, and later combining these local results to yield an approximation to the global solution.[14]

We have demonstrated the framework on three separate domains (cups, books, and dispenser bottles), in which a generic, view-based model (for the canonical view) is computed from a small set of exemplars. Although these results are encouraging, it should be noted that we have made a very important assumption that region segmentation errors exist in the form of oversegmentation. Although most region segmentation algorithms can be "pushed" toward oversegmentation, undersegmentation cannot be avoided altogether, requiring some means for splitting regions. Unfortunately, for any region, there are an infinite number of

[14] Although we solved the original intractable problem only approximately, the resulting approximate solution gives an effective object model that can be used in subsequent object recognition. Our reformulation can be compared with that used in machine learning: the search for the best classifier is often intractable, but the approximately best models are used for the actual classification with good results.

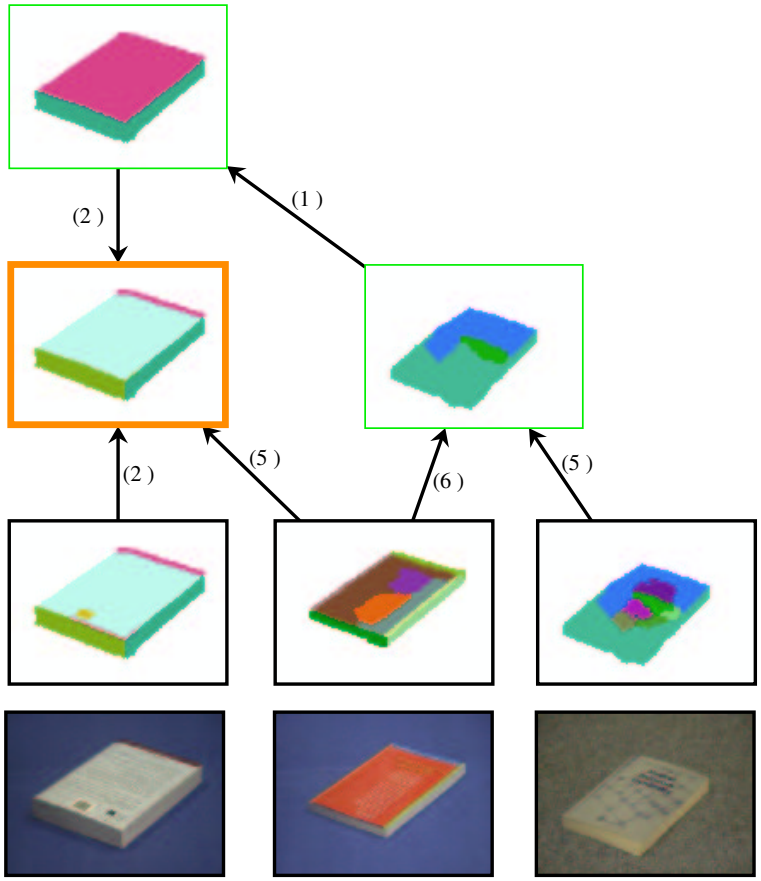


Fig. 19. Computed LCA (orange border) of 3 examples. The solution is also the LCA of the entire set of 8 book images.

ways of splitting the region. Fortunately, our region representation (a shock graph [51]) explicitly encodes a small, finite number of region split points [51], allowing us to accommodate region splitting within our framework in a tractable way. But where should such splitting occur in the process? One possibility would be to generate additional input exemplars from existing ones by splitting regions in various ways. A more efficient approach would be to incorporate the splitting process into the recursive decomposition of two input exemplars. In the case where one region is primitive and its corresponding region is not, the decomposition would normally terminate. However, if the primitive region can be split (among its finite split possibilities) so that decomposition can continue, the split is retained. We plan to explore region splitting in future work.

Very little world knowledge is currently used to constrain the abstraction process. In certain domains, particular region shapes may be prominent while others impossible. Any such knowledge, in the form of

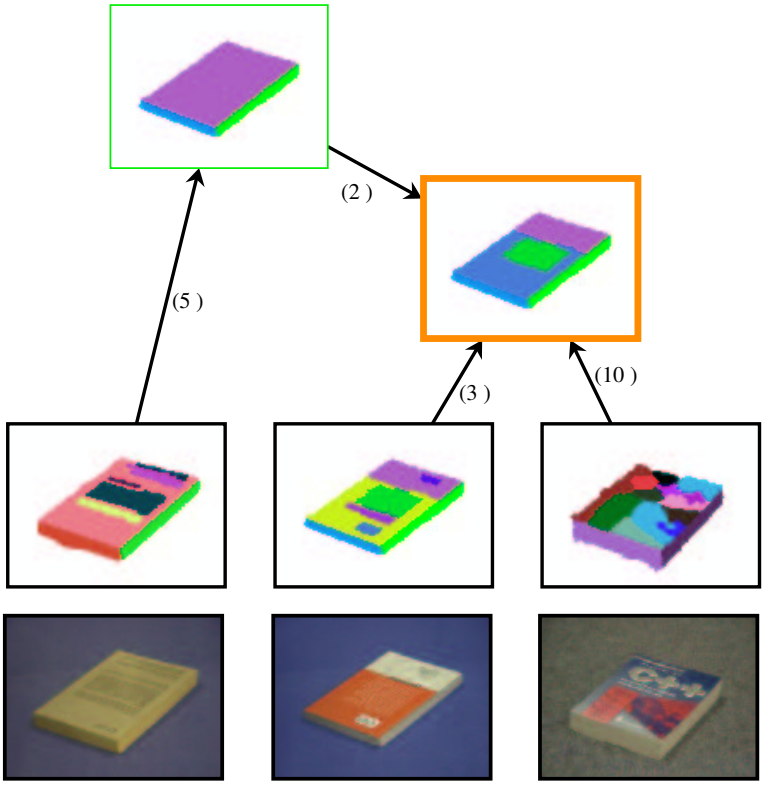


Fig. 20. Computed LCA (orange border) of 3 examples. The solution, influenced by the heavily over-segmented exemplar (rightmost node), is a decomposition of the desired LCA (top node).

known region shape or region adjacency constraints, can be used to prune the search space of possible cuts in a region adjacency graph. Perhaps even an object class's known functionality can be used to constrain the abstraction process, in terms of its mapping to underlying shape constraints [46]. Object appearance might also be exploited when *general* constraints are known on object color and/or texture.

We have focused on the generic model acquisition problem, leaving the more difficult problem of generic object recognition to future work. Toward the more restricted goal of finding a particular *target* object in an image, our approach to finding pairs of similar regions can be easily adapted as follows (assuming a region oversegmentation of the image). Given a region adjacency graph representing an object model, we first try to find in the image the object's silhouette. This can be accomplished by finding the minimum mean cycle in the product of the model's silhouette graph and the image's dual boundary segment graph. Having found the best match for the object's silhouette, we compute product graphs and minimum mean cycles at the level of individual regions. If matches are established at all levels, the model

Fig. 21. The set of 4 dispenser bottle images (and their region segmentations) used in the experiments.

is detected in the image. Thus, through an application of our model acquisition technique, target models can be detected in an image.

Our next major step is the less constrained problem of *unexpected* object recognition of a novel exemplar from our acquired object classes. Our efforts are currently focused on the analysis of the conditions under which two regions are merged. If we can derive a set of rules for the perceptual grouping of regions, we will be able to generate abstractions from images. Given a rich set of training data derived from the model acquisition process (recall that the LCA of two examples yields a path of region merges along with a set of "no-merges"), we are currently applying machine learning methods to uncover these conditions. Combined with our model acquisition procedure, we can close the loop on a system for generic object recognition which addresses a representational gap that has been long ignored in computer vision.

VII. Acknowledgements

The authors would like to thank Allan Jepson for his insightful comments and feedback on this work, as well as the three reviewers, whose comments have improved the presentation. The authors would also like to gratefully acknowledge the generous financial support of the U.S. Army Research Office, the National Science Foundation, the Natural Sciences and Engineering Research Council of Canada, and the Province of Ontario (PREA).

References

[1] G. Agin and T. Binford. Computer description of curved objects. *IEEE Transactions on Computers*, C-25(4):439–449, 1976.

[2] Ronen Basri, Luiz Costa, Davi Geiger, and David Jacobs. Determining the similarity of deformable shapes. In *Proceedings, ICCV Workshop on Physics-Based Modeling in Computer Vision*, pages 135–143, 1995.

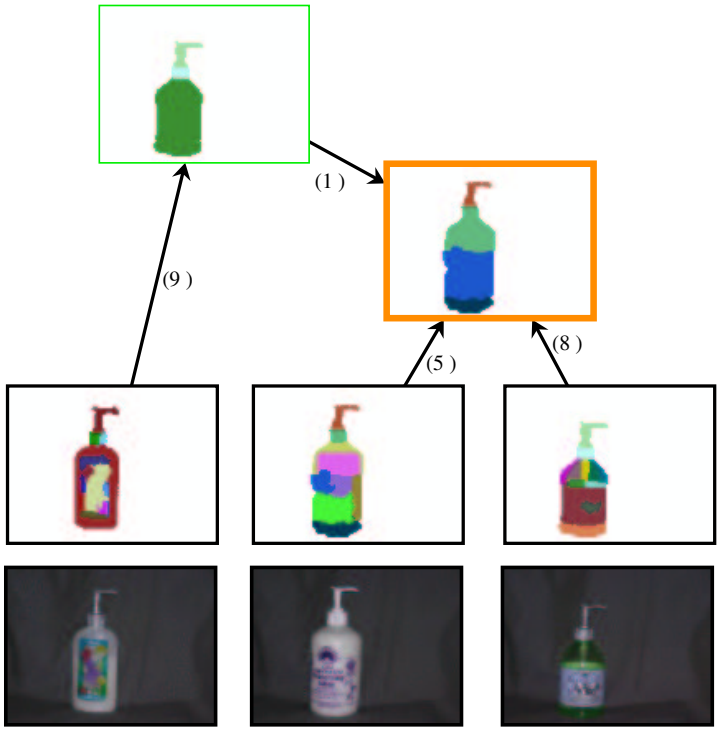


Fig. 22. Computed LCA (orange border) of 3 examples. Accidental alignment of non-salient structure in the rightmost two exemplars leads to an undesirable LCA.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[4] R. Bergevin and M. D. Levine. Part decomposition of objects from single view line drawings. *CVGIP: Image Understanding*, 55(1):73–83, January 1992.

[5] I. Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 32:29–73, 1985.

[6] T. Binford. Visual perception by computer. In *Proceedings, IEEE Conference on Systems and Control*, Miami, FL, 1971.

[7] D. Borges and R. Fisher. Class-based recognition of 3d objects represented by volumetric primitives. *Image and Vision Computing*, 15(8):655–664, 1997.

[8] R. Brooks. Model-based 3-D interpretations of 2-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–150, 1983.

[9] G. Carneiro and A. Jepson. Local phase-based features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Copenhagen, Denmark, 2002.

[10] G. Carneiro and A. Jepson. Multi-scale local phase-based features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, WI, 2003.

[11] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 750–755, 1997.

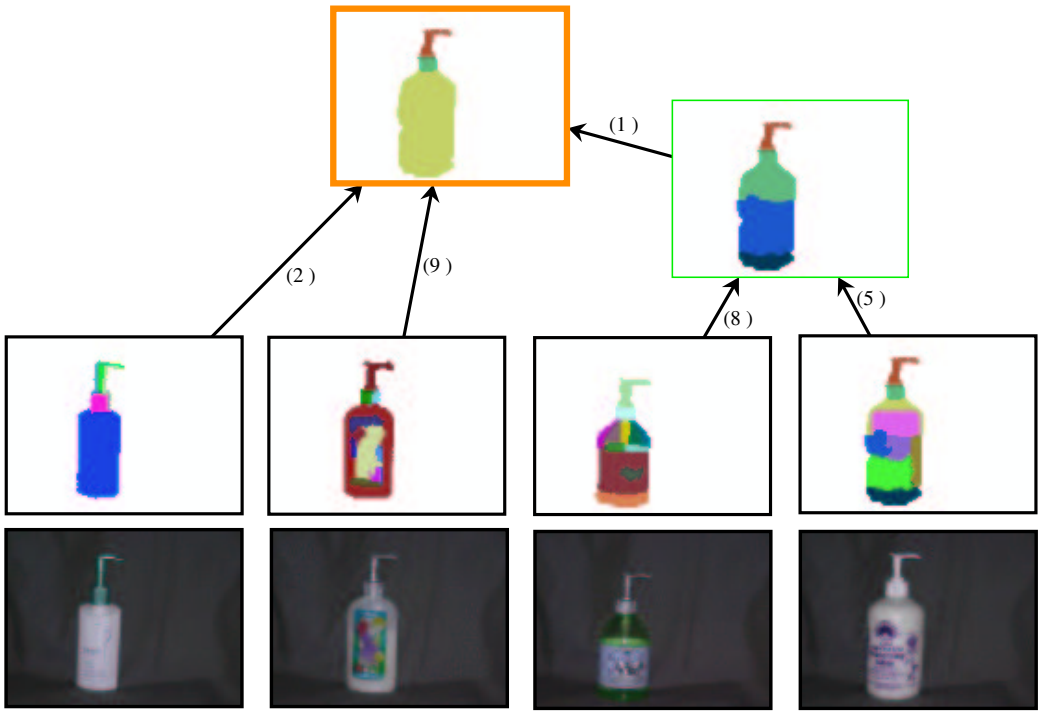


Fig. 23. Computed LCA (orange border) of 4 examples. The additional exemplar results in the desirable LCA.

[12] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, chapter 25. The MIT Press, 1993.

[13] Christopher M. Cyr and Benjamin B. Kimia. 3d object recognition using shape similarity-based aspect graph. In *IEEE International Conference on Computer Vision*, pages 254–261, 2001.

[14] S. Dickinson, A. Pentland, and A. Rosenfeld. A representation for qualitative 3-D object recognition integrating object-centered and viewer-centered models. In K. Leibovic, editor, *Vision: A Convergence of Disciplines*. Springer Verlag, New York, 1990.

[15] S. Dickinson, A. Pentland, and A. Rosenfeld. From volumes to views: An approach to 3-D object recognition. *CVGIP: Image Understanding*, 55(2):130–154, 1992.

[16] S. Dickinson, A. Pentland, and A. Rosenfeld. 3-D shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):174–198, 1992.

[17] David Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2):652–673, 1999.

[18] G. Ettinger. Large hierarchical object recognition using libraries of parameterized model sub-parts. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pages 32–41, Ann Arbor, MI, 1988.

[19] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 98–104, Santa Barbara, CA, 1998.

[20] F. Ferrie, J. Lagarde, and P. Whaite. Darboux frames, snakes, and superquadrics. In *Proceedings, IEEE Workshop on Interpretation of 3D Scenes*, pages 170–176, 1989.

[21] D. Forsyth, J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell. Invariant descriptors for 3d object recognition

and pose. *IEEE PAMI*, 13:971–992, October 1991.

[22] A. Gupta. Surface and volumetric segmentation of 3D objects using parametric shape models. Technical Report MS-CIS-91-45, GRASP LAB 128, University of Pennsylvania, Philadelphia, PA, 1991.

[23] J. Han and M Kamber. *Data Mining: Concepts and Techniques*, page 228. Morgan Kaufmann Publishers, 2001.

[24] D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[25] X. Jiang, A. Munger, and H. Bunke. On median graphs: properties, algorithms, and applications. *IEEE PAMI*, 23(10), 2001.

[26] Jean-Michel Jolion. The deviation of a set of strings. *Pattern Analysis and Applications*, 6(3):224–231, 2003.

[27] N Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for k shortest simple paths. *Networks*, 12:411–427, 1982.

[28] Yakov Keselman and Sven Dickinson. Bridging the representation gap between models and exemplars. In *Proceedings, IEEE Workshop on Models versus Exemplars in Computer Vision*, Kauai, Hawaii, December 2001.

[29] Yakov Keselman and Sven Dickinson. Generic model abstraction from examples. volume 1, pages 856–863, Kauai, Hawaii, December 2001.

[30] Yakov Keselman and Sven Dickinson. Generic model abstraction from examples. In Gregory D. Hager, Henrik I. Christensen, Horst Bunke, and Rolf Klein, editors, *Proceedings, International Workshop on Sensor-Based Intelligent Robots*, volume 2238, pages 1–24, Dagstuhl Castle, Germany, October 2002.

[31] B. B. Kimia, A. Tannenbaum, and S. W. Zucker. Shape, shocks, and deformations I: The components of two-dimensional shape and the reaction-diffusion space. *International Journal of Computer Vision*, 15:189–224, 1995.

[32] Walter G. Kropatsch. Building irregular pyramids by dual graph contraction. *IEE-Proc. Vision, Image and Signal Processing*, 142(6):366–374, 1995.

[33] A. Leonardis and H. Bischoff. Dealing with occlusions in the eigenspace approach. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pages 453–458, San Francisco, CA, June 1996.

[34] A. Leonardis, F. Solina, and A. Macerl. A direct recovery of superquadric models in range images using recover-and-select paradigm. In *Proceedings, Third European Conference on Computer Vision (Lecture Notes in Computer Science, Vol 800)*, pages 309–318, Stockholm, Sweden, May 1994. Springer-Verlag.

[35] D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, MA, 1985.

[36] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings, International Conference on Computer Vision*, pages 1150–1157, Corfu, Greece, 1999.

[37] B. Luo, R. Wilson, and E. Hancock. Spectral embedding of graphs. *Pattern Recognition*, to appear, 2004.

[38] D. Macrini, A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker. View-based 3-d object recognition using shock graphs. In *Proceedings, International Conference on Pattern Recognition*, Quebec, 2002.

[39] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[40] R. Nelson and A. Selinger. A cubist approach to object recognition. In *Proceedings, IEEE International Conference on Computer Vision*, Bombay, January 1998.

[41] R. Nevatia and T. Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8:77–98, 1977.

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

[43] A. Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28:293–331, 1986.

[44] A. Pope and D. Lowe. Learning object recognition models from images. In *Proceedings, IEEE International Conference on Computer Vision*, pages 296–301, Berlin, May 1993.

[45] N. Raja and A. Jain. Recognizing geons from superquadrics fitted to range data. *Image and Vision Computing*, 10(3):179–190, April 1992.

[46] E. Rivlin, S. Dickinson, and A. Rosenfeld. Recognition by functional parts. *Computer Vision and Image Understanding*, 62(2):164–176, 1995.

[47] C. Schmid and R. Mohr. Combining greyvalue invariants with local constraints for object recognition. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pages 872–877, San Francisco, CA, June 1996.

[48] S. Sclaroff and A. Pentland. Modal matching for correspondence and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):545–561, June 1995.

[49] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Shock-based indexing into large shape databases. In *European Conference on Computer Vision*, pages 731–746, 2002.

[50] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997.

[51] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.

[52] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–146, 1990.

[53] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991.

[54] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[55] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *European Conference on Computer Vision*, volume 1, pages 18–32, 2000.

[56] P.H. Winston. Learning structural descriptions from examples. In *The Psychology of Computer Vision*, chapter 5, pages 157–209. McGraw-Hill, 1975.

[57] S. Zhu and A. L. Yuille. Forms: a flexible object recognition and modelling system. *International Journal of Computer Vision*, 20(3):187–212, 1996.