Generic Model Abstraction from Examples

Yakov Keselman[1] and Sven Dickinson[2]

[1] Department of Computer Science
Rutgers University
New Brunswick, NJ 08903, USA
[2] Department of Computer Science
University of Toronto
Toronto, Ontario M5S 3G4, Canada

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 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, and demonstrate the approach on real imagery.

1 Introduction

1.1 Motivation

The goal of generic object recognition is to recognize a novel exemplar from a known set of object classes. For example, given a generic model of a coffee cup, a generic object recognition system should be able to recognize "never before seen" coffee cups whose local appearance and local geometry vary significantly. Under such circumstances, traditional CAD-based recognition approaches (e.g., [25, 29, 22]) and the recently popular appearance-based recognition approaches (e.g., [44, 31, 27]) will fail, since they require a priori knowledge of an imaged object's exact geometry and appearance, respectively. Unfortunately, progress in generic object recognition has been slow, as two enormous challenges face the designers of generic object recognition systems: 1) creating a suitable generic model for a class of objects; and 2) recovering from an image a set of features that reflects the coarse structure of the object. The actual matching of a set of salient, coarse image features to a generic model composed of similarly-defined features is a much less challenging problem.

The first challenge, which we will call generic model acquisition, has been traditionally performed manually. Beginning with generalized cylinders (e.g., [4, 1, 33, 6]), and later through superquadrics (e.g., [34, 20, 42, 24, 43, 28]) and geons (e.g., [3, 12, 14, 13, 2, 37, 47, 18, 10, 5]), 3-D generic model acquisition required the designer to not only identify what features were common to a set of object exemplars belonging to a class, but to construct a model, i.e., class prototype, in terms of those features. The task seems quite intuitive: most cups, for example, have some kind of handle part attached to the side of a larger container-like part, so choose some parameterized part vocabulary that can accommodate the within-class part deformations, and put the pieces together. Although such models are generic (and easily recognizable[1]), such intuitive, high-level representations are extremely difficult (under the best of conditions) to recover from a real image.

The generic object recognition community has long been plagued by this representational gap between features that can be robustly segmented from an image and the features that make up a generic model. Although progress in segmentation, perceptual grouping, and scale-space methods have narrowed this gap somewhat, generic recognition is as elusive now as it was in its prime in the 1970's. Back then, those interested in generic object recognition eliminated the gap by bringing the objects they imaged closer to their models, by removing surface markings and structural detail, controlling lighting conditions, and reducing scene clutter. Since then, the recognition community has eliminated the gap by steadily bringing the models closer to the imaged objects, first resulting in models that were exact 3-D reproductions (CAD-based templates) of the imaged objects, followed by today's 2-D appearance-based templates.

Interestingly enough, both approaches to eliminating this gap 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) and a feature in the model. This assumption is fundamentally flawed, for saliency in the image does not equal saliency in the model. Under this assumption, object recognition will continue to be exemplar-based, and generic recognition will continue to be contrived.

Returning to our two challenges, we first seek a (compile-time) method for automatically acquiring a generic model that bridges the representational gap between the output of an image segmentation module and the "parts" of a generic model. 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 address the first challenge – that of generic model acquisition.

1.2 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

[1] Take a look at the object models of Marr and Nishihara [30], Nevatia and Binford [33], Brooks [6], Pentland [34], or Biederman [3], and you will easily recognize the classes represented by these models.

the viewpoint with respect to the exemplar in each image is similar. Fig. 1(a) illustrates a simple example in which three different images, each containing a block in a similar orientation, are presented to the system. Our task is to find the common structure in these images, under the assumption that structure that is common across many exemplars of a known class must be definitive of that class. Fig. 1(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, the approach might be expected to include vertical stripes in that view of the abstracted model.

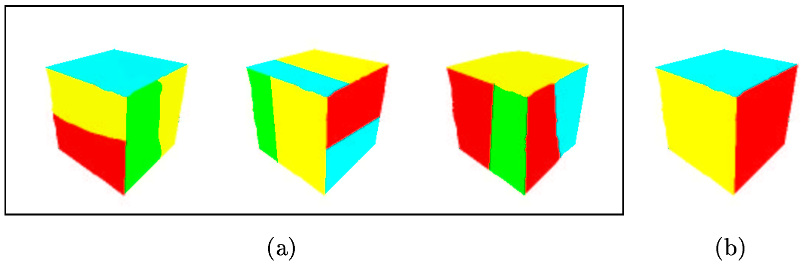


Fig. 1. Illustrative Example of Generic Model Acquisition: (a) input exemplars belonging to a single known class; (b) generic model abstracted from examples.

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. 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. 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 (e.g., see [12, 14, 13]) is beyond the scope of this paper. For now, the result will be a collection of 2-D views that describe a generic 3-D object. This collection would then be added to the view-based object database used at recognition time.

1.3 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., [44, 31, 27, 7] is that no segmentation, 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., [36, 38, 32, 45]. Still, the resulting models are very exemplar-specific due to the extreme locality at which they extract and match features (e.g., one pixel or at best, a small neighborhood around one pixel). The resulting models 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 [35, 42, 24, 21, 43, 11]. Provided the range data can be segmented into parts or surfaces, these generic primitives can be used to approximate structural detail not belonging to the class. 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 generating generic models from 2-D data, there has been much less work. The seminal work of Winston [46] 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 specializations of others, the resulting descriptions could be organized into 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 [17, 16]. The technique was successfully applied to traffic sign recognition and remains one of the more elegant examples of feature abstraction and generic model acquisition.

1.4 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 a known class, and conclude with a discussion of the method's strengths and weaknesses, along with a glimpse of where we're heading.

2 Problem Formulation

Returning to Fig. 1, 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 (later on, we will discuss the problem of undersegmentation, and how our approach can accommodate it). Under this assumption, the space of all possible region adjacency graphs formed by any sequence of merges of adjacent regions will form a lattice, as shown in Fig. 2. The lattice size is exponential in the number of regions obtained after initial oversegmentation.[2]

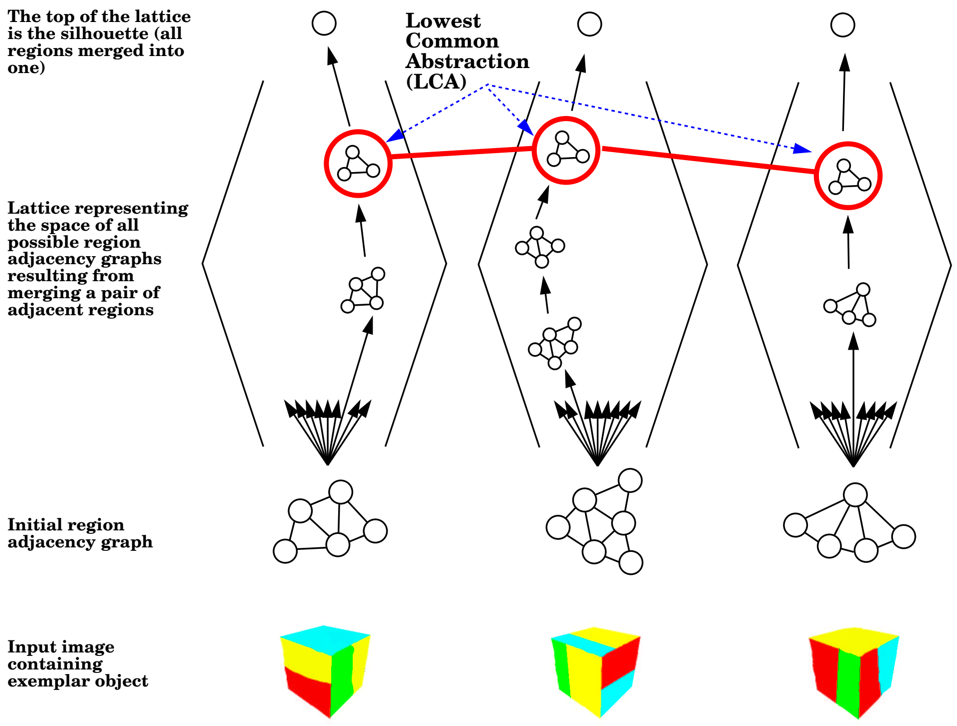

The top of the lattice is the silhouette (all regions merged into one)
Lowest Common Abstraction (LCA)
Lattice representing the space of all possible region adjacency graphs resulting from merging a pair of adjacent regions
Initial region adjacency graph
Input image containing exemplar object

Fig. 2. The Lowest Common Abstraction of a Set of Input Exemplars

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

[2] Indeed, considering the simplest case of a long rectangular strip subdivided into $n + 1$ adjacent rectangles, the first pair of adjacent regions that can be merged can be selected in n ways, the second in $n - 1$, and so on, giving a lattice size of $n!$. The lattice is even larger for more complex arrangements of regions.

a silhouette of the object (where all regions have been merged into one region), carries little information about the salient surfaces of the object. Following some preliminary definitions, our problem can be stated as follows:

Definitions: Given N input image 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 (silhouette) 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). Note that there may be more than one LCA.

Problem Definition: For N input image exemplars, find the LCA.

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. Unfortunately, the presence of a single, heavily undersegmented exemplar (a single-node silhouette in the extreme case) will drive the LCA towards the trivial silhouette CA. In a later section, we will relax our LCA definition to make it less sensitive to region undersegmentation.

3 The Lowest Common Abstraction of Two Examples

3.1 Overview

For the moment, we will focus our attention on finding the LCA of two lattices; 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. Our approach will be to restrict the search for the LCA to the *intersection* of the lattices, which is much smaller than either lattice, 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 set – the root node. If one or both of the roots have no children in the lattice, i.e., the original region segmented image was already a silhouette, then the process stops and the LCA is simply the root. However, in most cases, each root (silhouette) has many possible decompositions, or *specializations*. We will restrict ourselves to the space of specializations of each root into two component regions, and attempt to find a specialization that is common to both lattices. Again, there may be multiple 2-region specializations that are common to both lattices; each is a member of the intersection set.

Assuming that we have some means for ranking the matching *common specializations* (if more than one exists), we pick the best one (the remainder constituting a set of backtracking points), and recursively apply the process to each pair of isomorphic subregions. The process continues in this fashion, "pushing" its way down the intersection lattice, until no further common specializations 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.

3.2 The Common Specialization of Two Abstraction Graphs

One of the major components of our algorithm for finding the LCA of two examples is finding the common specialization of two abstraction graphs. In this subsection, we begin by formulating the problem as a search for a pair of corresponding cuts through the two abstraction graphs. Next, we reformulate the problem as the search for a pair of corresponding paths in the dual graph representations of the two abstraction graphs. Finally, high search complexity motivates our transformation of the problem into the search for a shortest path in a product graph of the two dual graphs. In the following subsections, we elaborate on each of these steps.

Problem Definition Our *specialization* problem can be formulated as follows: Given a pair of isomorphic graphs G_1 and G_2 in L_1 and L_2, find a pair of isomorphic specializations of G_1 and G_2, denoted by $H_1 \in L_1$ and $H_2 \in L_2$, if such a pair exists. 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 work in generic 2-D object recognition [40, 39, 41], in which distance is a weighted function of a region's part structure and part geometry. 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 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. If the constraints are satisfied, the distance will be small.

The Search for Corresponding Graph Cuts 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. To reduce the computational complexity of finding a pair of corresponding cuts, we will restrict ourselves to cuts that generate regions that are simply connected in the topological sense, i.e., they have no internal "holes". Despite this restriction, the complexity is still prohibitive, and we need to take further measures to simplify our formulation.

The Search for Corresponding Paths in a Dual Graph To find a pair of corresponding cuts, we first define a *dual graph* to be any graph with the property that a cut in the original graph can be generated by a path in its dual graph.[3] Thus, finding a pair of corresponding cuts in the original graphs reduces to finding a pair of corresponding paths in their dual graphs. Moreover, our restriction of open cuts (preventing holes) will result in a corresponding restriction (preventing cycles) on the paths in the dual graphs. Before we discuss how to generate a dual graph, which we visit in a later section, we will assume that the dual graphs exist and proceed to find a pair of corresponding paths in the dual graphs.

The Product Graph of Two Dual Graphs Our transformation to the dual graph has not affected the complexity of our problem, as there could be an exponential number of paths in each dual graph, leading to an even larger number of possible pairs of paths (recall our checkerboard example). Rather than enumerating the paths in each dual graph and then enumerating all pairs, we will generate the pairs directly using a heuristic that will generate more promising pairs first. To accomplish this, we define the *product graph* of two dual graphs, such that each path in the product graph corresponds to a pair of paths in the dual graphs. Moreover, with suitably defined edge weights in the product graph, we can ensure that paths resulting in nearly optimal values of an appropriately chosen objective function will correspond to more promising pairs of paths in the 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:

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

Hence, 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.

Algorithm for Finding the Common Specialization Notice that a path that is optimal with respect to an objective function defined in terms of edge weights of the product graph may result in unacceptable partitions. Therefore, we will evaluate several near-optimal paths in terms of similarity of regions resulting

[3] While our definition agrees in principle with the usual notion of the dual graph for planar graphs [23], we have chosen a more flexible definition which does not prescribe its exact form.

from the partitions. Our generic algorithm for finding the common specialization of two abstraction nodes is shown in Algorithm 1. In the following sections, we will elaborate on a number of components of the algorithm, including the choice of a dual graph, edge weights, and the objective function.

Algorithm 1 A Generic Algorithm for Finding a Common Specialization

```
1: Let $A_1$, $A_2$ be subgraphs of the original region adjacency graphs that correspond
   to isomorphic vertices of the abstraction graphs.
2: Let $G_1$, $G_2$ be dual graphs of $A_1$, $A_2$.
3: Form the product graph $G = G_1 \times G_2$, as described above.
4: Choose an objective function $f$, compute edge weights $w_i$, and select a threshold
   $\varepsilon > 0$.
5: Let $P_f$ be the optimal path with respect to $(f, \{w_i\})$ with value $F(P_f)$.
6: Let $P = P_f$
7: while $|f(P) - f(P_f)| < \varepsilon$ do
8:    Let $P_1$ and $P_2$ be the paths in $G_1$, $G_2$ corresponding to $P$.
9:    Let $(V_1, W_1)$ and $(V_2, W_2)$ be the resulting cuts in $A_1$, $A_2$
10:   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
11:      output decompositions $(V_1, W_1)$ and $(V_2, W_2)$.
12:      return
13:   end if
14:   Let $P$ be the next optimal path with respect to $(f, \{w_i\})$.
15: end while
16: output "no non-trivial specialization is found".
```

Choosing a Dual Graph We now turn to the problem of how to define a dual graph of an abstraction graph. In this section, we will present two alternatives. However, before describing these alternatives, we must first establish some important definitions, in conjunction with Fig. 3.

- A region pixel is a *boundary pixel* if its 8-connected neighborhood contains pixels belonging to one or more other regions.
- At a given step, our algorithm for computing the LCA of two examples will focus on a connected subgraph of the input region adjacency graph. A region belonging to this subgraph is called a *foreground region*, while any region not belonging to this subgraph is called a *background region.*
- A boundary pixel of a region is *exterior* if it is adjacent to a background region; it is *interior* otherwise.
- A *boundary segment* of a region is a contiguous set of its boundary pixels, and is *interior* if all its points (except possibly for the endpoints) are such.

– A *junction* is a point whose neighborhood of a fixed radius r includes pixels from at least 3 different regions.[4] If one or more of the regions are background, the junction is *exterior*; otherwise, it is *interior*.

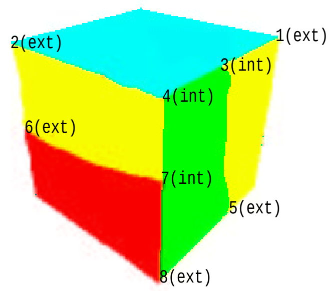


Fig. 3. Illustration of Basic Definitions. All the regions belonging to the cube are foreground. Junctions are labeled as either interior or exterior. The boundary segments between junctions 2 and 4, and 4 and 7 are interior, while those between 1 and 5, and 5 and 8 are exterior.

Our initial choice for a dual graph representation was a junction graph:

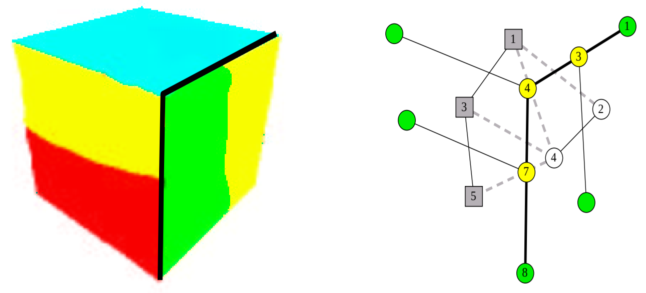
– The *junction graph* of a region adjacency graph is a graph whose nodes represent region junctions and whose arcs are those internal boundary segments that connect junctions.[5]

Fig. 4 illustrates a junction graph. Since a path between two exterior junctions along internal boundary segments separates regions into two disjoint groups, a path in the junction graph between nodes corresponding to external junctions generates a cut in the region adjacency graph. Thus, the junction graph is dual to the region adjacency graph.

Despite the simplicity and intuitive appeal of the junction graph, it poses some limitations. For example, consider the different types of edges that arise in the product graph. Looking back at its definition, we notice that some edges are of type "edge-edge", corresponding to the $((u_1, u_2), (v_1, v_2))$ terms, while others are of type "node-edge", corresponding to the $((v_1, u_2), (v_1, v_2))$ and $((u_1, v_2), (v_1, v_2))$ terms. Since edge weights in the product graph will be based

[4] The radius r is a parameter of the definition and should be chosen so as to counteract the effects of image noise. In the ideal case, the immediate neighbors of the junction point will belong to 3 different regions, which may not be true if the boundaries of the regions are noisy.

[5] There can be multiple arcs between two nodes corresponding to different internal boundary segments connecting the two junctions.



(a) (b)

Fig. 4. The Dual Graph of a Region Adjacency Graph. (a) Segmented image with a path between external junctions shown in black. (b) Junction graph overlaid on the region adjacency graph, with the corresponding path in the junction graph shown in black. The path edges cut the region adjacency graph (at the dashed edges) into two parts, one whose nodes are square and shaded and one whose nodes are circular and unshaded.

on node and edge data in the dual graphs, a definition of an edge weight in the junction graph that will result in acceptable paths will require sufficiently precise geometric alignment of the object silhouettes, such that junctions and internal boundary segments are closely aligned.[6] However, such a close alignment is unlikely, with both the shapes and locations of the boundary segments varying across exemplars. As boundary segments possess not only positional but also local shape information, below we define a graph whose edges *and nodes* carry information about internal boundary segments.

If we define a graph on undirected boundary segments, i.e., take boundary segments as nodes and define node adjacency according to the adjacency of the corresponding boundary segments, then many paths in the dual graph will not result in cuts in the region adjacency graph due to the presence of 3-cycles occurring at junction points. Although these unacceptable paths can be modified to produce cuts in the region adjacency graph, they may still be generated among near-optimal paths, thus decreasing the overall efficiency. Orienting boundary segments, i.e., splitting each undirected boundary segment into two directed copies, eliminates the problem, leading to the following alternative dual graph:

– The *boundary segment graph* of a region adjacency graph has directed internal boundary segments of the regions as its node set (two nodes per undirected internal boundary segment), and an edge from boundary segment b_1 to b_2 if the ending point of b_1 coincides with the starting point of b_2, unless b_1 and b_2 are directed versions of the same undirected boundary segment, in

[6] A simpler, watershed-like method, where blurred dark region boundaries are overlaid on top of each other, and darkest curves are found, might also be applicable.

which case they are not adjacent. Nodes of the graph are attributed with the corresponding boundary segments, while an edge is attributed with the union of the adjacent boundary segments corresponding to the nodes it spans.

The directed boundary segment graph possesses the interesting property that for each path starting at a node v_1 and ending at a node v_2, there is a "reverse directed" path starting at the duplicate of v_2 and ending at the duplicate of v_1, which is characteristic of undirected graphs. An example of a boundary segment graph is given in Fig. 5.

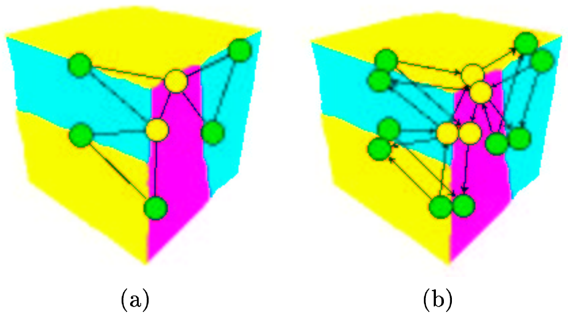


Fig. 5. Two Possible Boundary Segment Graphs. Nodes representing boundary segments are placed on or near the segments. A path between any two green nodes in both graphs corresponds to a cut in the region adjacency graph. Notice that in the undirected version, (a), cycles have no meaningful interpretation in terms of curves in the original image. The directed acyclic version of the same graph, (b), in which each boundary segment is duplicated and directed, eliminates this problem.

Assigning Edge Weights to the Product Graph Consider now the task of defining edge weights so that optimal paths will result in regions of similar shape. The minimal requirement that shape similarity imposes on the paths is that their shapes are similar and that they connect "corresponding points" on the object's silhouettes. Despite the fact that corresponding contours are unlikely to be closely aligned, their proximity in both shape and position can be taken into account. We therefore define edge weights as a combination of metrics reflecting the amount of rigid displacement and the amount of non-rigid transformation required to align one boundary segment with another. Due to the special type of the initial boundary segment graphs, whose nodes and edges are both attributed with boundary segments, it is sufficient to define distances on pairs of boundary segments from different images. We employ a simple Hausdorff-like distance between the two boundary segments, yielding a local approximation to the global similarity of the regions.

Choosing an Objective Function 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 [9]. Fast algorithms for generating successive shortest and simple shortest paths are given in [15, 26]. 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.[7]

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[8] of Dijkstra's algorithm [9] 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 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 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 [15], 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 preliminary empirical testing.

3.3 Algorithm

Now that we have fully specified our algorithm for finding a common specialization of two abstraction graphs, we will embed it in our solution to the problem of finding the LCA of two examples. Recall that our solution to finding the LCA of two examples computes the intersection of the 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 specialization of these nodes, and add it to the intersection set. The process is recursively applied to each common specialization (i.e., member of the intersection set) until no further common specializations are found. The resulting set of "lowest" common specializations represents the LCA of the two lattices. The procedure is formalized in Algorithm 2.

4 The LCA of Multiple Examples

So far, we've 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?

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

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

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

```
1: Let $A_1$, $A_2$ be the initial region adjacency graphs.
2: Let $G_1$, $G_2$ denote abstraction graphs belonging to abstraction lattices, $L_1$ and $L_2$
   respectively.
3: Let $G_1^0$, $G_2^0$ be the topmost nodes of the lattices.
4: Let $G_1 = G_1^0$, $G_2 = G_2^0$.
5: while there are unexplored non-trivial isomorphic nodes $u_1 \in G_1$, $u_2 \in G_2$ do
6:   Let $U_1$ and $U_2$ be the corresponding subgraphs of $A_1$, $A_2$.
7:   if there is a common specialization $U_1 = V_1 \cup W_1$ and $U_2 = V_2 \cup W_2$ then
8:     Split the nodes $u_1 \in G_1$, $u_2 \in G_2$ by forming the specialization 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 $A_1$, $A_2$.
9:     Let $G_1 = H_1$, $G_2 = H_2$, and goto 5.
10:  end if
11: end while
12: output $G_1$, $G_2$.
```

Furthermore, when moving towards 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 Fig. 6. If the definition of the pairwise LCA is directly generalized, thus requiring the search for 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 (to outliers), 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. Our solution begins by relaxing the first property. We will define the LCA of a set of 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. As finding the desired abstraction according to this definition would still involve the construction of many abstraction lattices, whose complexity is intractable, we will pursue an approximation method.

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

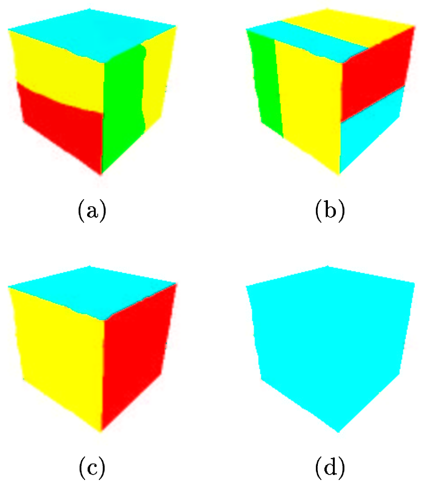


Fig. 6. The straightforward computation of the Lowest Common Abstraction of exemplars (a)-(d) gives the exemplar in (d). However, (c) is the Lowest Common Abstraction of exemplars (a)-(c), and therefore is more representative.

arcs go from nodes corresponding to G_1, G_2 to the node corresponding to H in the closure graph.

Although a graph may not be directly linked to all of its abstractions, if H is an abstraction of G, there is a directed path between the nodes corresponding to G and H. Thus, any abstraction is reachable from any of its specializations by a directed path. Each edge in the closure graph is assigned a weight equal to the merge edit distance that takes the specialization to the abstraction. The edit distance is simply the difference between the numbers of nodes in the specialization 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 Fig. 7.

Given such a graph, the robust LCA of *all* inputs will be that node that minimizes the sum of shortest path distances from the initial input region adjacency graphs. In other words, we are looking for the "median" of the graph. Note that the resulting solution is bound to lie in the constructed graph, and therefore may be only an approximation to the true answer. To find a possibly better solution, one must consider a supergraph of the closure graph. Algorithm 3 computes the LCA for a set of input examples.

To prove correctness of the algorithm, we must prove that the distance sum for the output node is minimal. Adopting the convention that edges are directed towards the current sink node, we denote $R_1, \ldots, R_k$ to be those nodes whose outgoing edges point directly to the sink, and denote $L_1, \ldots, L_k$ to be subgraphs with nodes $R_1, \ldots, R_k$ as their roots (L_i will consist of all nodes having R_i as

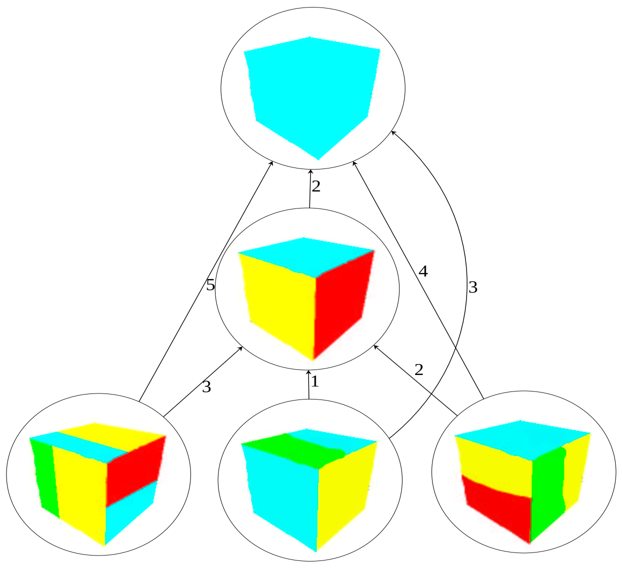


Fig. 7. Embedding Region Adjacency Graphs and their Pairwise LCA's in a Weighted Directed Acyclic Graph. The center node is the median, as its distance sum value is $3+1+2+2=8$, while the sum is $5+3+4+0=12$ for the topmost node.

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**
```

their abstraction). It suffices to prove that if $DS(\text{sink}) > DS(R_i)$ and $DS(R_i) \leq DS(R_j)$ for all j, then a solution lies in L_i.

Consider the original problem restricted to L_i's and call the restricted objective function of the i-th problem DS_i. The values of the objective functions approximately satisfy the following relations:

$$DS(R_i) = \sum_{j=1}^{k} DS_j(R_j) + 2(k-1) - F(i) \tag{1}$$

Here $2(k-1)$ comes from the fact that to get from an element of L_j, that is not in L_i, to R_i, we must necessarily pass through the two edges connecting the sink node to L_i, L_j. $F(i)$ is the sum of shortest path distances from the original nodes to R_i taken over the nodes that belong to both L_i and L_j (for some $i \neq j$). Under the assumption $DS(R_i) \leq DS(R_j)$, we have that $F(i) \geq F(j)$, which means that L_i wholly contains at least as many shortest paths as any other L_j. This, in turn, implies that a node minimizing the original objective function will lie in L_i.

To analyze the complexity of the algorithm, notice that the first step, i.e., finding the distance sum to 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 [9]. 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 complexity can be somewhat reduced by using the relations (1). 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.

The possibility that the size of the generated graph is exponential in the size of the initial region adjacency graphs cannot be ruled out. This could happen, for example, when the images are segmented too finely, and different pairs of region adjacency graphs are abstracted to similar but unequal graphs. We hope to address this issue in the future. Alternatives include resegmenting the images when the size of the generated closure graph exceeds a threshold value, and subsampling the graph in a randomized fashion.

5 Experiments

In this section, we begin by showing two results of our approach applied to synthetic image data, and follow that with two results using real image data. As mentioned before, images were region segmented using the approach of Felzenzwalb and Huttenlocher [19]. For the experimental results shown below, considerable parameter tuning in the region segmentation algorithm was performed to avoid region oversegmentation.

5.1 Synthetic Imagery

The first input exemplar set to our program consisted of four synthetic images of books, whose segmentations are shown as rectangular nodes with black, bold outline in the closure graph, shown in Fig. 8. The other two nodes are the LCA's of each pair of nodes just below them. Edges of the graph are labeled with the merge edit distance between the nodes, while nodes are labeled with the distance sum value. Although the three upper nodes are labeled with the same minimal value, 9, the upper node is optimal according to the algorithm. Our computed LCA agrees with our intuition given the input set.

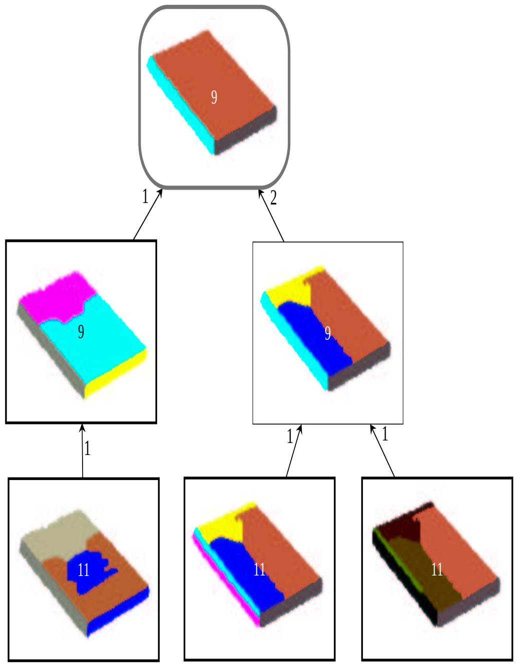


Fig. 8. The LCA of Multiple Books. The upper node is optimal. See text for details.

The next input set to our program consists of four images of bottles, whose segmented versions are shown as rectangular nodes with black, bold outline in the closure graph, shown in Fig. 9. The fifth, upper node is the LCA of the two nodes just below it. Again, edges of the graph are labeled with the merge edit distance between the nodes, while nodes are labeled with the distance sum value. The center node, with rounded corners and boldest outline, is optimal according to the algorithm.

On the positive side, the computed LCA preserved all features present in the majority of the inputs. For example, the four rectangular stripes, into which the region segmentation algorithm segmented the label of the bottle, are preserved, as is the region corresponding to the cork. However, on the negative side, the subdivision of the label into four stripes is undesirable, as it does not correspond to a partition of the object into meaningful structure based on coarse shape. This is more a limitation of the original exemplar set. As was pointed out earlier, our algorithm finds the finest-level common structure present in the input region adjacency graphs, which may not correspond to the desired shape-based structure. If additional examples of bottles were available, in which the label was not segmented into several pieces or was segmented very differently, a more appealing answer would likely have resulted.

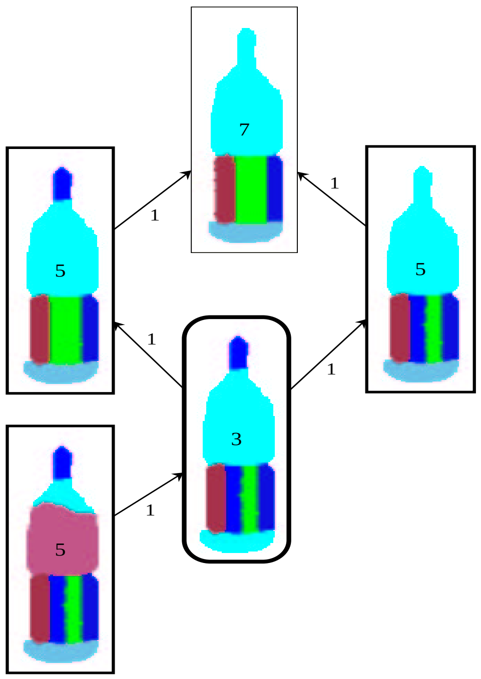


Fig. 9. The LCA of Multiple Bottles. The node in the center, with rounded corners, is optimal. See text for details.

5.2 Real Imagery

We now turn to two examples of our approach applied to real imagery. In the figures below, each of the two rows corresponds to one entry, the first column containing the original intensity image, the second column containing its region segmentation, and the third column containing the LCA of the two region segmentations (although each lattice's LCA is shown for completeness, either one could be chosen for inclusion in a database).

In the first example, we compute the LCA of two input image exemplars, as shown in Fig. 10. The computed LCA captures the coarse structure with one exception. The ideal LCA would assign the handle two regions rather than one. The reason the "hole" was left out of the handle is that the shape matching module found that the "handle silhouette" regions (i.e., the union of the handles with their holes) were sufficiently similar in shape, while the two-region (handle,hole) representation was found to exceed the shape dissimilarity threshold. This is a deficiency of the current implementation of the shape similarity module, and it will be eliminated in the near future.

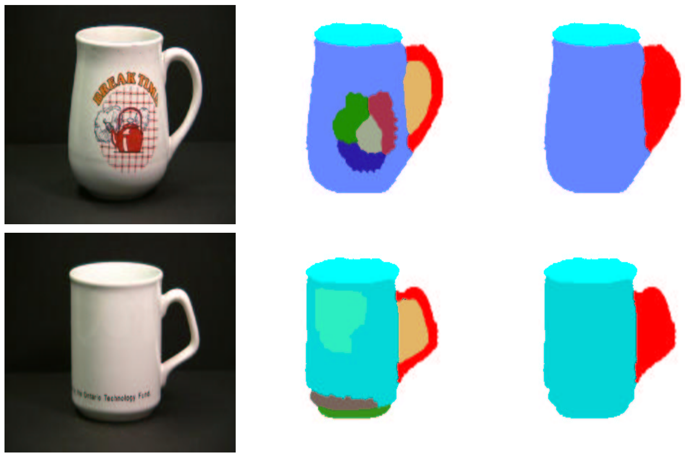


Fig. 10. The LCA of Two Cups.

Our second example, again computing the LCA of two exemplars, is shown in Fig. 11. In this case, the results deviate more significantly from our expectation. For example, the handle of the second cup was not segmented from the body of the cup. This illustrates the need for a region splitting operation, which we discuss in the next section. Notice, however, that even if the second cup was better segmented, the segmentation of the first cup would have prevented the algorithm from obtaining the correct abstraction, since the handle of the first cup is directly attached to the top portion, while in the second cup, it is attached to the body portion. In this case, a split would be required on the first cup. The other undesirable feature computed in the LCA is the stripe at the bottom of the LCA. As has been already mentioned, this stripe would disappear if there were other examples, most of whom lacked the stripe.

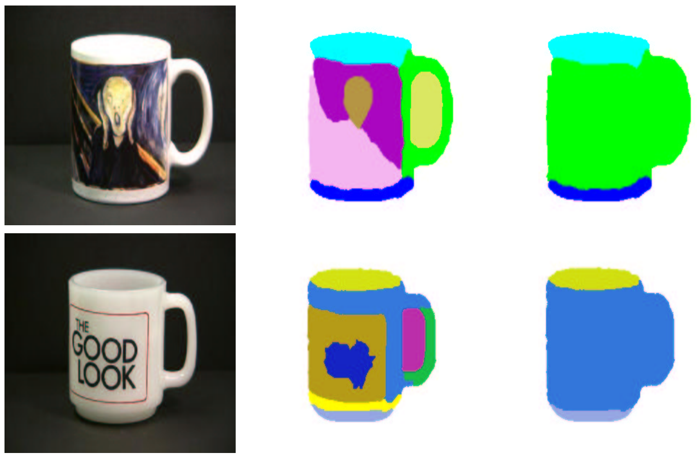


Fig. 11. The LCA of Two Cups.

6 Limitations

There are a number of limitations of our proposed approach that deserve mentioning. As we discussed earlier in the paper, we have assumed that a lattice can be formed from an input region adjacency graph through a sequence of adjacent region merge operations. However, this assumes that the input region adjacency graph is *oversegmented*. This is a limiting assumption, for lighting effects and segmentation errors can lead to region *undersegmentation*[9]. Granted, our algorithm for computing the LCA of multiple examples, through its search for the median of the closure graph, can avoid the influence of an undersegmented example, such as a silhouette. Nevertheless, a more direct approach to recovering from region undersegmentation would be appropriate. Such an approach might decompose selected regions in each input exemplar into a set of subregions which could remain intact, merge with each other, or merge with other neighbors. Although discretion must be exercised in selecting which regions should be split (undue oversegmentation would result in a much more complex and possibly ineffective input graph), we are fortunate in that the shock graph representation of regions specifies a finite number of splits (branches in the shock graph, which can be severed).

A second limitation involves the method by which a common specialization of two graphs is found. Recall that this consists of finding corresponding paths through two graphs that yield matching subregions. In order to reduce the complexity of examining a possibly exponential number of corresponding paths (not to mention application of the shape similarity function), we reformulated the problem as a search for the shortest path in a product graph. The limitation

[9] Frequent undersegmentation of our input data was the main reason we did not use the segmentation algorithm of Comaniciu and Meer [8]

arises from the fact that in order to make the search tractable, our edge weights capture local information. Thus, although the algorithm may find a global minimum path, such a path may not represent the best cut. We are investigating the incorporation of additional information into the edge weights, including more global shape information.

Finally, more experimentation is needed to better understand the performance of the framework as a function of the number and nature of the input exemplars. For example, although the algorithm for finding the LCA of multiple examples (i.e., the median of the closure graph) can theoretically handle any number of input exemplars, our experiments to date have included only a few input exemplars. Experiments using larger numbers of input exemplars are necessary to establish the performance of the algorithm. In addition, although our experiments to date have included some undersegmentation and oversegmentation, we have not evaluated the performance of the entire framework as a function of degree of segmentation error. A more thorough set of experiments, parameterized in terms of number of exemplars and degree of segmentation error, is essential before the approach can be fully evaluated.

7 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 runtime, for the recognition of objects, but 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 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. We have shown some very preliminary results that compute a generic model from a set of example images belonging to a known class. Although these results are encouraging, further experimentation is necessary and a number of limitations need to be addressed.

Our next major step is the actual recognition of the derived models from a novel exemplar. 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), we are 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.

8 Acknowledgements

The authors gratefully acknowledge the support of the National Science Foundation, the Army Research Office, and the National Sciences and Engineering Research Council of Canada.

References

1. G. Agin and T. Binford. Computer description of curved objects. *IEEE Transactions on Computers*, C-25(4):439–449, 1976.
2. R. Bergevin and M. D. Levine. Part decomposition of objects from single view line drawings. *CVGIP: Image Understanding*, 55(1):73–83, January 1992.
3. I. Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 32:29–73, 1985.
4. T. Binford. Visual perception by computer. In *Proceedings, IEEE Conference on Systems and Control*, Miami, FL, 1971.
5. D. Borges and R. Fisher. Class-based recognition of 3d objects represented by volumetric primitives. *Image and Vision Computing*, 15(8):655–664, 1997.
6. R. Brooks. Model-based 3-D interpretations of 2-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–150, 1983.
7. O. Camps, C. Huang, and T Kanungo. Hierarchical organization of appearance based parts and relations for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–691, Santa Barbara, CA, 1998.
8. 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.
9. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, chapter 25. The MIT Press, 1993.
10. S. Dickinson, R. Bergevin, I. Biederman, J.-O. Eklundh, A. Jain, R. Munck-Fairwood, and A. Pentland. Panel report: The potential of geons for generic 3-D object recognition. *Image and Vision Computing*, 15(4):277–292, April 1997.
11. S. Dickinson, D. Metaxas, and A. Pentland. The role of model-based segmentation in the recovery of volumetric parts from range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):259–267, March 1997.
12. 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.
13. 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.
14. 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.
15. David Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2):652–673, 1999.
16. 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.

17. G.J. Ettinger. Hierarchical object recognition using libraries of parameterized model sub-parts. Technical Report 963, MIT Artificial Intelligence Laboratory, 1987.
18. R. Fairwood. Recognition of generic components using logic-program relations of image contours. *Image and Vision Computing*, 9(2):113–122, 1991.
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. F. Ferrie, J. Lagarde, and P. Whaite. Darboux frames, snakes, and super-quadrics: Geometry from the bottom up. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(8):771–784, 1993.
22. 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.
23. R. Gould. *Graph Theory*, pages 170–172. The Benjamin/Cummings Publishing Company, Inc., 1988.
24. 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.
25. D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.
26. N Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for k shortest simple paths. *Networks*, 12:411–427, 1982.
27. 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.
28. 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.
29. D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, MA, 1985.
30. D. Marr and H. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Royal Society of London*, B 200:269–294, 1978.
31. H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.
32. R. Nelson and A. Selinger. A cubist approach to object recognition. In *Proceedings, IEEE International Conference on Computer Vision*, Bombay, January 1998.
33. R. Nevatia and T. Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8:77–98, 1977.
34. A. Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28:293–331, 1986.
35. A. Pentland. Automatic extraction of deformable part models. *International Journal of Computer Vision*, 4:107–126, 1990.
36. 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.

37. N. Raja and A. Jain. Recognizing geons from superquadrics fitted to range data. *Image and Vision Computing*, 10(3):179–190, April 1992.
38. 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.
39. A. Shokoufandeh and S. Dickinson. Applications of bipartite matching to problems in object recognition. In *Proceedings, ICCV Workshop on Graph Algorithms and Computer Vision (web proceedings: http://www.cs.cornell.edu/iccv-graph-workshop/papers.htm)*, September 1999.
40. A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker. Indexing using a spectral encoding of topological structure. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 491–497, Fort Collins, CO, June 1999.
41. K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.
42. 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.
43. 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.
44. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
45. Markus Weber, M. Welling, and Pietro Perona. Unsupervised learning of models for recognition. In *European Conference on Computer Vision*, volume 1, pages 18–32, 2000.
46. P.H. Winston. Learning structural descriptions from examples. In *The Psychology of Computer Vision*, chapter 5, pages 157–209. McGraw-Hill, 1975.
47. K. Wu and M. Levine. Recovering parametric geons from multiview range data. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pages 159–166, Seattle, WA, June 1994.