Generic Model Abstraction from Examples

Yakov Keselman
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903, USA

Sven Dickinson*
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 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, and demonstrate the approach on real imagery.

1. Introduction

1.1 Motivation

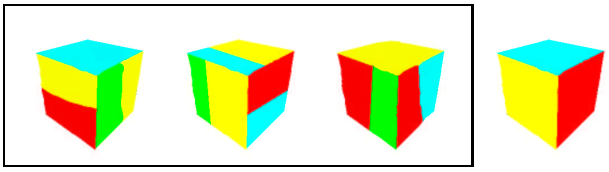
The goal of generic object recognition is to recognize a novel exemplar from a known set of object classes. For example, given a generic model of a coffee cup, a generic object recognition system should be able to recognize "never before seen" coffee cups whose local appearance and local geometry vary significantly. Under such circumstances, traditional CAD-based recognition approaches (e.g., [8, 10, 7]) and the recently popular appearance-based recognition approaches (e.g., [11]) 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 traditionally been performed manually. Beginning with generalized cylinders (e.g., [2]), and later through superquadrics (e.g., [13]) and geons (e.g., [1]), 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. The problem has always been that 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 has narrowed this gap somewhat, generic recognition is as elusive now as it was in its prime in the 1970's. Unfortunately, rather than bridging this gap, the recognition community has artificially removed it. In the 1970's, 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.

*The authors would like to acknowledge the support of the Army Research Office, the National Science Foundation, and the National Sciences and Enginnering Research Council of Canada. The authors would also like to thank Ali Shokoufandeh (Drexel) for providing the shock graph matching code, and to Xuhui Ao and Wei Wang (Rutgers) for providing the shock graph generation code.

[1]Take a look at the object models of Brooks [2], Pentland [13], or Biederman [1], and you will easily recognize the classes represented by these models.



(a) (b)

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

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 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 (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 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 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. 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, 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., [11], 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., [14, 15, 12, 18]. 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 [13, 17]. 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 generating generic models from 2-D data, there has been much less work. The seminal work of Winston [19] 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

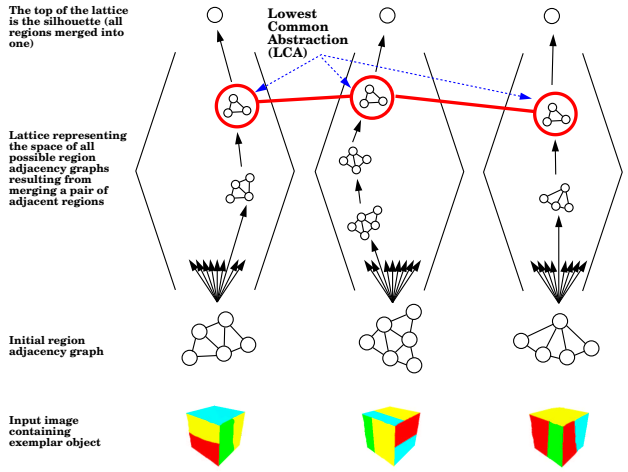


Figure 2: The Lowest Common Abstraction of a Set of Input Exemplars

primal sketch features [5]. The technique was successfully applied to traffic sign recognition and remains one of the more elegant examples of feature abstraction and generic model acquisition.

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 [6]). 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 6, 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 Fig. 2. The lattice size is exponential in the number of regions obtained after initial oversegmentation.[2]

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, $E_1, E_2, \ldots, E_N$, 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. 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 such outliers.

3. 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. 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). How-

[2] Indeed, considering the simple 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 $n!$.

ever, 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.

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 the process to each pair of isomorphic component subregions.[3] 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.

3.1 Problem Definition

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 (each a node in its own lattice) 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 decompositions of G_1 and G_2, respectively, if they are isomorphic. H_1 and H_2 are formed by starting with G_1 and G_2, identifying a pair of corresponding nodes (regions), G_{1_i} and G_{2_j}, and replacing each of these nodes with two nodes (subregions). 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, find a pair of isomorphic 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.

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 [16], 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.

3.2 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.[4] 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." Despite this restriction, the complexity is still prohibitive, and we need to take further measures to simplify our formulation.

We begin by transforming our two region adjacency graphs into their dual *boundary segment graphs*. 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. This 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, 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 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:

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

[3] 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.

[4] 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.

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

A 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.[5]

Each path in our product graph corresponds to a pair of possible cuts in two regions. Consequently, our goal is to find that path yielding the best matching subregions and relations. Here's where we face a problem. This objective function can only be evaluated once a complete path is found, since a 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're comparing regions, while locally, we're 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's begin with the edge weights. If we align the two regions (from which we seek corresponding cuts) through our region matching algorithm [16], 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. We employ a simple Hausdorff-like distance between the two boundary segments, yielding a local approximation to the global similarity of the regions. Defining the objective function to be the sum of path edge weights or the maximum path edge weight brings us to the computationally tractable domains of shortest and minmax paths, respectively.

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 [3]. Fast algorithms for generating successive shortest and simple shortest paths are given in [4, 9]. 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.[6]

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[7] of Dijkstra's algorithm [3] 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 [4], 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

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 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.

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? Furthermore, when moving towards multiple examples, how do we prevent a "noisy" example, such as a single, heavily undersegmented silhouette, from driving the solution towards the trivial silhouette? To extend our two-exemplar LCA solution to a robust, multi-exemplar solution, we begin with two

[5] Although the node set of the product graph is the product of the node sets of the initial graphs, 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 ensure that the product graph will be connected.

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

[7] 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".

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 non-trivial 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 $G_1 = H_1, G_2 = H_2$, and **goto** 5.
  end if
end while
output G_1, G_2.

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.

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 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 input exemplars will be large, eliminating that node from contention.

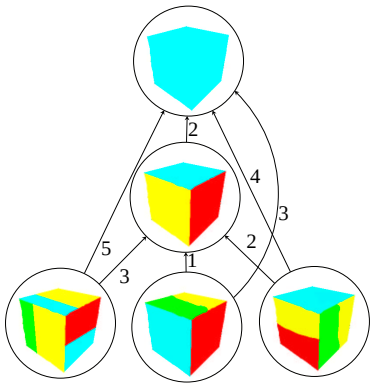


Figure 3: Embedding Region Adjacency Graphs and their Pairwise LCA's in a Weighted Directed Acyclic Graph. The three nodes at the bottom, as well as the undersegmented "outlier" at the top are the four input exemplars. Although the true LCA is the top node, the center node is the median, as its distance sum value is $3 + 1 + 2 + 2 = 8$, while the sum is $5 + 3 + 4 + 0 = 12$ for the topmost node.

Our solution begins by constructing an approximation to the intersection lattice of multiple examples. 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, G_2 to the node corresponding to H in the closure graph.

A graph may not be directly linked to all of its abstractions. 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. 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 Fig. 3.

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 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 approxima-

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**

tion 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. Although the closure graph will contain the true LCA, our median formulation may lower the LCA fringe below the true LCA.

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 [3]. 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.

5. Experiments

In Figures 4 and 5, 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 at the top. In each case, the closure graph consists of only four members, with the same pairwise LCA emerging from all input pairs. While in Fig 4, the solution captures our intuitive notion of the cup's surfaces, the solution in Fig 5 is less intuitive. A strip along the bottom is present in each exemplar, and understandably becomes part of the solution. However, due to region segmentation errors, the top region in the body of 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 6, we again present three exemplars to the system. In this case, the closure graph has many nodes. Unlike Figures 4 and 5, 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 shown with an elliptical border, and represents an effective model for the cup.

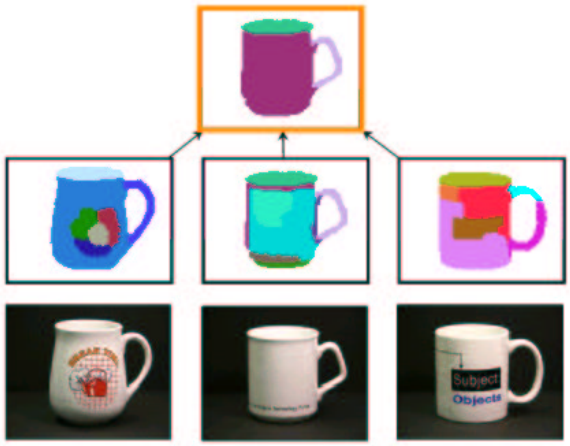


Figure 4: Computed LCA (top node) of 3 Examples. Note that color figures are available on the ps file provided on the conference CD.

6. 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 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. For example, we currently assume an oversegmented image, thereby requiring only region merge operations. However, our region representation explicitly encodes a finite number of region split points [16], allowing us to accommodate region splitting within our framework.

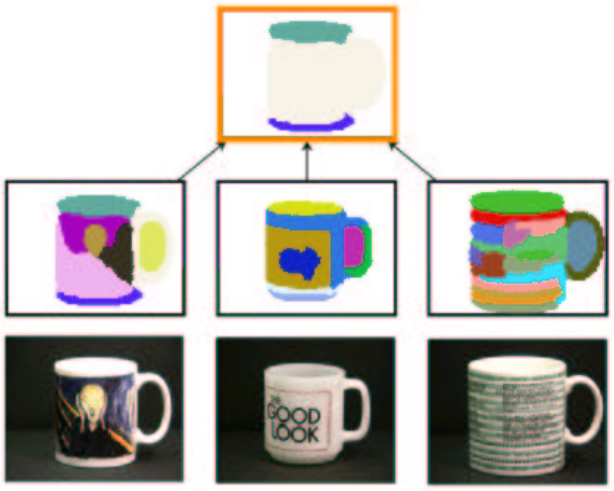


Figure 5: Computed LCA (top node) of 3 Examples

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.

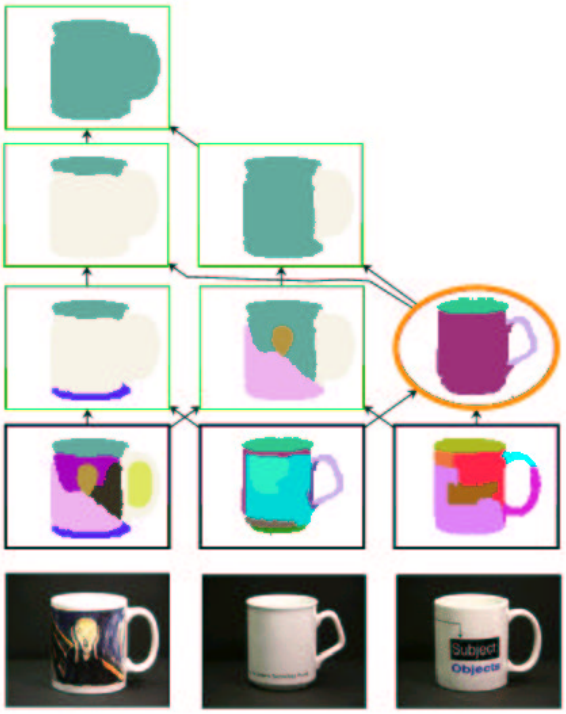


Figure 6: Computed LCA (elliptical border) of 3 Examples

References

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

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

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

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

[5] 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.

[6] 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.

[7] D. Forsyth, J. 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.

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

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

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

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

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

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

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