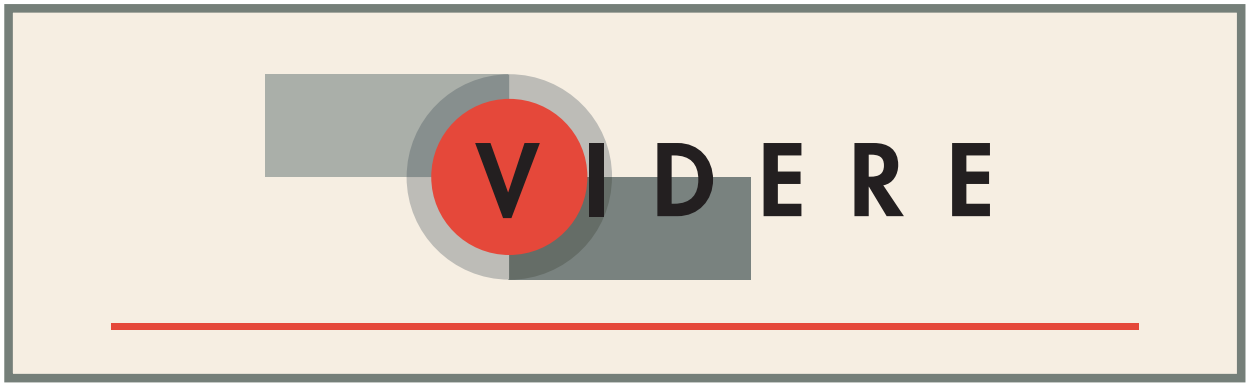


Videre: Journal of Computer Vision Research

Quarterly Journal
Winter 1998, Volume 1, Number 2
The MIT Press

Article 4

Learning Image Annotation: The CITE System

Craig Dillon
Terry Caelli

Videre: Journal of Computer Vision Research (ISSN 1089-2788) is a quarterly journal published electronically on the Internet by The MIT Press, Cambridge, Massachusetts, 02142. Subscriptions and address changes should be addressed to MIT Press Journals, Five Cambridge Center, Cambridge, MA 02142; phone: (617) 253-2889; fax: (617) 577-1545; e-mail: journals-orders@mit.edu. Subscription rates are: Individuals $30.00, Institutions $125.00. Canadians add additional 7% GST. Prices subject to change without notice.

Learning Image Annotation: The CITE System

Craig Dillon,[1] Terry Caelli[2]

In this paper we consider how more-recent machine learning techniques can be used in building adaptive and trainable image annotation systems that can mimic human performance. In this case the annotation is defined with respect to domain-specific labels for image regions, hierarchies, and, in general, cliques that are of specific importance to the application domain. To illustrate these issues, we consider one of our more recent annotation systems, the CITE system, where the domain knowledge consists of hierarchies of labeled and attributed graphs, and how it has been used to annotate outdoor, airport, and office scenes consisting of different types of scenarios and information.

Keywords: image labeling, image animation, learning, knowledge-directed computer vision, image segmentation

1. Department of Computer Science, Curtin University of Technology, Perth, Western Australia, 6019. cdillon@cs.curtin.edu.au

2. Deputy Director, Center for Mapping, Professor of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, 1216 Kinnear Rd., Columbus, Ohio 43212. Phone: 614 292-3396 (W), 614 292-8062 (H). caelli@cfm.ohio-state.edu
Current address: Curtin University of Technology, Department of Computer Science, Perth, Western Australia, 6019

1 Introduction

An image annotation system (IAS) is a methodology, usually embodied in a set of algorithms, that can take as input thematic image data (in forms including color, intensity, and/or range information) and produce some form of labeling or description of the image being viewed with respect to domain knowledge. Perhaps the two more-recent and well-known IASs are SCHEMA and SIGMA. In SCHEMA [1] both knowledge and computation are partitioned at a course-grained semantic level. The knowledge base contains a set of schemas, each of which is designed to recognize one particular class of object. This differs from most machine vision approaches which usually focus on one representation and one matching strategy.

Each schema has access to a global blackboard that contains the current interpretation of the scene, and can post and receive messages to this global space asynchronously. An instance of the appropriate schema is generated for each hypothesized instance of the object in the scene. Each schema instance has access to what are termed *knowledge sources*. The knowledge sources, which operate at a number of different levels, provide analysis of the image and resultant image structures. The knowledge base in SCHEMA is hand coded by a knowledge engineer in a declarative language that covers the *endorsement space*, *confidence function*, and *control strategies* for each schema.

The SIGMA system [2] is an image-understanding system designed for aerial image analysis. It contains three modules for low-level vision, model selection, and geometric reasoning, as well as a query module through which the user interacts. These modules are interconnected such that top-down and bottom-up vision processes are closely integrated. The interpretation within SIGMA is in the form of a *part-of* hierarchy with image features at the leaf nodes and spatial relations describing the higher-level structure. SIGMA's knowledge base is object oriented in the sense that instances of an object are created (instantiated) dynamically from base object classes. Each object has three main knowledge components: unary properties of the object, relationships with other objects, and control information to guide the analysis process. The control structures are triplets of the form (CONDITION, HYPOTHESIS, ACTION).

SIGMA contains a sophisticated low-level vision module that performs goal-directed image segmentation. A *goal* in this system is expressed as a set of constraints and properties that the solution must satisfy, some of which specify the environment in which the solution is to be found. One strength of this system is its ability to search the image at a very low level for expected image features. It has been demonstrated on aerial images of new housing developments, and robustness to shadows and poor image contrast has been shown. However, as with SCHEMA, the knowledge base for SIGMA is hand-coded by an expert, and the system

as a whole has not been demonstrated to recognize more than a few different object classes and is restricted to a specified domain.

2 Proposed Theory and System

CITE differs from SIGMA and SCHEMA in a number of fundamental ways. Primarily, it uses an incremental learning paradigm with the aim of *tracking* the types of labeling that experts utilize when using images for interpreting what is sensed. This incremental learning is used to build and improve the knowledge base after each scene has been fully analyzed as shown in Figure 1, where the numbers beside each function block represent the approximate order of operation for each operator. This learning and adaptive perspective is also incorporated in low-level processes in order to produce image features or regions that correspond to what experts would deem appropriate to describe with specific domain knowledge. That is, traditional feed-forward segmentation is augmented with knowledge-driven resegmentation that closes the control loop on low-level vision processes to match the human annotation process as close as possible. CITE also extends the conventional knowledge bases (including relational database models) to a fully hierarchical

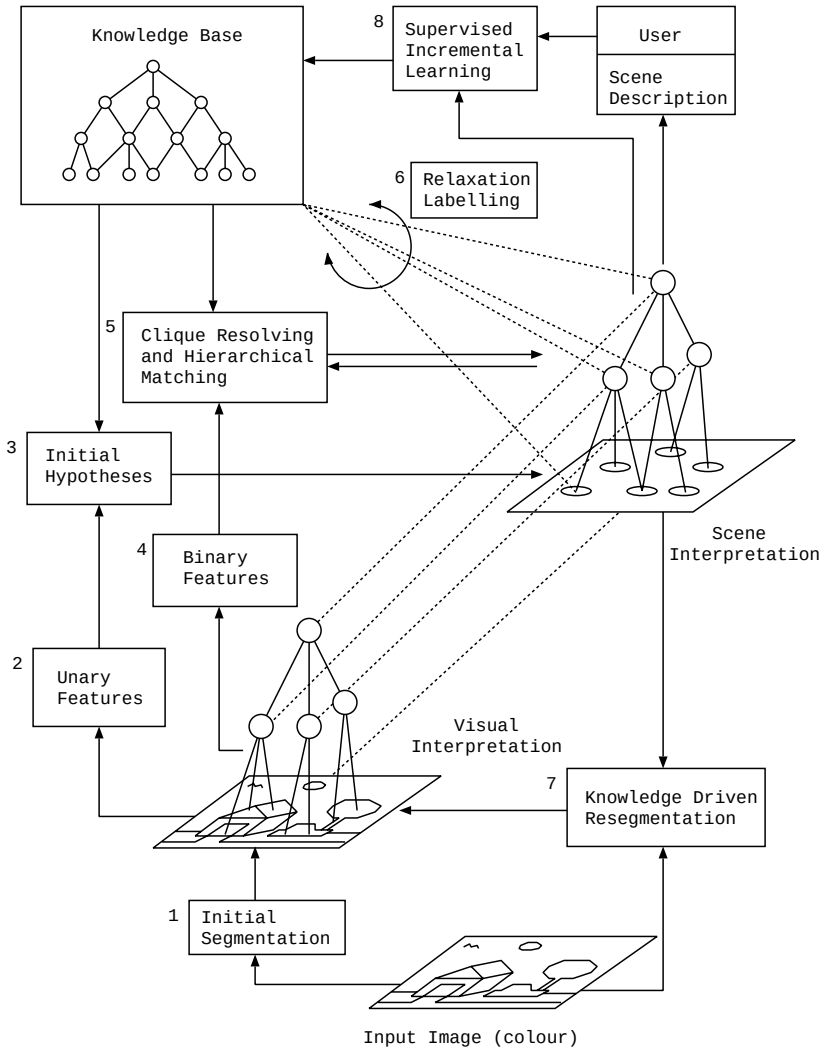


Figure 1. Overview of CITE architecture.

one with, in principal, no depth limitation. Multiple hypotheses are generated for each scene element, and these are resolved using a hierarchical extension to relaxation labeling.

An initial segmentation (1) of the image causes the unary (part) feature calculator to compute features for each of the low-level regions (2). These features are then matched with the knowledge base (3) to provide initial labeling hypotheses that are represented in the indexing structure called the *scene interpretation*. Clique resolving and hierarchical binary matching then occur on these initial hypotheses (5) using binary features calculated from the hierarchical segmentation (4). The higher-level scene hypotheses are added into the scene interpretation structure, and hierarchical relaxation labeling begins to resolve the multiple ambiguous labels for each object (6). As the labels begin to resolve, nodes are individually resegmented (7) using parameters stored in the knowledge base. These resegmentations replace the initial segmentations in the visual interpretation structure, resulting in a repeat of the unary and binary feature extraction and matching (stages (2) through (6)). This cycle continues a number of times until the interpretation becomes stable. If CITE's final interpretation is incorrect, the user may chose to incrementally learn the correct object labeling (8) by selecting the incorrectly labeled nodes and the desired knowledge base node. The updated knowledge base is then available as the next scene is viewed.

The relaxation labeling, knowledge-driven resegmentation, and hierarchical clique resolving and matching provide very tight closed-loop feedback within CITE. The hierarchical knowledge base provides a rich scene description that can include contextual, taxonomic, and deep decomposition information. The use of incremental supervised learning provides CITE with the ability to increase the descriptive power and accuracy of its analyses, as well as to add new world knowledge as it becomes available, rather than requiring the full set of scene objects to be present during an initial learning phase.

2.1 Knowledge Representation

CITE represents world knowledge as a semi-restricted graph in which each node represents an object or visual concept that is either a *part-of*, a *view-of*, or a *type-of* its parent or parents. Figure 2 is a simple example of such a knowledge base. Within each node is stored information about the optimal segmentation, feature extraction, and matching algorithms that are used to recognize this object in an image. CITE represents taxonomies using the type-of node. For complex hierarchical world knowledge, taxonomies represent valuable reductions in data complexity.

Deep taxonomies can be built using many levels of type-of node, and each level offers a different degree of semantic generalization. Because the knowledge base is built using supervised learning, the taxonomies

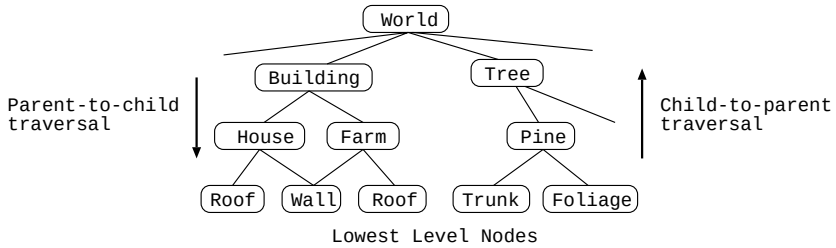


Figure 2. Knowledge base graph example.

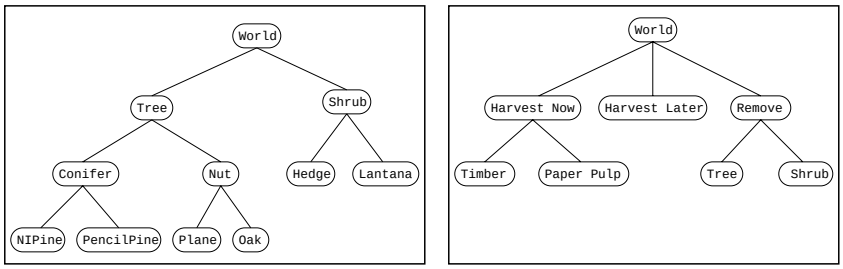


Figure 3. Two alternative tree taxonomies.

embedded in the knowledge base are context sensitive. Figure 3 illustrates two possible "tree" taxonomies—one that could have been constructed by a botanist, the other by a forester. The variations in these contexts result in differing interactions between the unary and binary learning and the hierarchical relaxation labeling. The taxonomic structure of the knowledge base is built incrementally during the supervised learning process from the simple description given for each object.

2.2 Interpretation Modules

CITE contains hierarchically segmented image data in the form of a semi-restricted (acyclic) graph called the *visual interpretation*. Multiple labeling hypotheses can be generated for each node in the visual interpretation, and these are stored in an intermediate structure called the *scene interpretation*.

Visual interpretation CITE uses hierarchical segmentation in which regions are grouped to form larger regions that are grouped to form even larger regions, and so on. There is only one type of node in the visual interpretation, called the VI node, which may contain multiple parents and multiple children. Multiple children represent the grouping of smaller regions with common properties or labels into a larger region with a single label. Multiple parents represent the notion of ambiguous set membership, which is important in solving the *clique (grouping) problem*. CITE also uses a process of parallel hypothesis generation to generate multiple likely solutions (local constraints) and then relaxation labeling to propagate these hypotheses to maximize global consistency with respect to the current knowledge base. (See below for details.)

Scene interpretation The scene interpretation (SI) structure is an indexing structure that connects the visual interpretation to the knowledge base and represents the current scene belief state. Each VI node contains a weighted list of hypothesis links to SI nodes, and each hypothesis link is interpreted as being the visual support for the given scene interpretation node. Multiple SI hypotheses may be connected to a VI node during top-down hypothesis of the existence of missing or occluded parts. Ambiguous clique memberships may also be represented by multiple SI hypotheses on the given VI node.

Each VI node also contains a list of pixels that the node represents in the original input image. There is no ordering or adjacency limitation placed on these pixels, which means that a VI node can be used to represent multiple non-connected regions. This is particularly useful for representing occluded objects and complex high-level scene elements whose constituent parts are not necessarily connected in the image. The final important component of the VI node is that this is a data structure

where image features are stored when they have been calculated. Unary (part) features are stored in the VI node and binary (relational) features are stored in the common parent of the multiple VI nodes between which they are calculated.

There is only one node type in the scene interpretation (SI) graph, called an *SI node*. Like the knowledge base, there must be at least one child-to-parent and one parent-to-child relationship in every cycle. The SI graph is unbounded with the exception that there is one node with no parents as this represents the entire scene. Leaf nodes with no children represent the lowest level (finest detail) of scene analysis.

Each SI node contains an unweighted list of parents and a weighted list of children, similar to VI nodes. Each SI node also contains a weighted list of VI nodes that are interpreted as being the visual support for that particular scene element. Multiple VI nodes may be connected to an SI node during top-down hypothesis of the existence of missing or occluded parts.

Each SI node contains a weighted list of knowledge base (KB) nodes that represent possible object labelings for that SI node. Each weight in the KB node list can be seen as a probability, and as a result of the normalization process these will add to 1.0. Multiple SI-KB labelings represent an ambiguous classification, and these are resolved through a process of relaxation labeling.

2.3 Operational Overview: A Simple Example

Consider the example of recognizing a Norfolk Island Pine using the Botanist knowledge base. A sample image and knowledge base is shown in Figure 4. When CITE first loads an image, a default top-level VI node is created. The segmentation initialization process detects that this VI node has not yet been segmented and begins this process. As no recognition has occurred yet, CITE uses the top-level KB node to determine the initial segmenter and its parameterization.

When the segmentation operator has completed, one leaf VI node is generated for each image region. Figure 5 shows this initial segmentation and the VI graph at this point. (See Section 2.7 for details of segmentation algorithms.) Note that this initial segmentation has resulted in an over-segmentation of the image. The objective is to end up with just two parts, the trunk and the foliage, as described in the knowledge base. This simple example does not contain ambiguous clique membership; hence, we have not shown the SI graph, which is identical in structure to the VI graph for this case.

After the initial segmentation, the unary feature extraction operator detects unattributed VI nodes and begins calculating the unary features for these nodes. At the same time, the connected components grouper constructs a parent VI node representing the collection of VI leaf nodes. The grouped VI nodes are then processed by the SI-from-VI matching

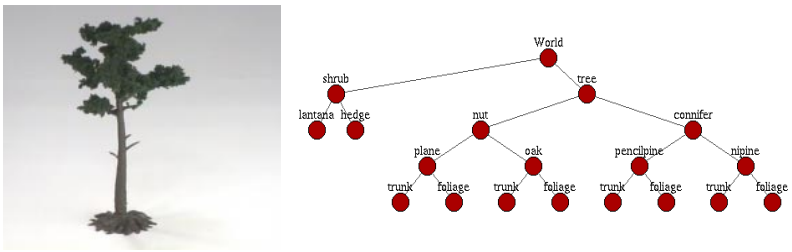


Figure 4. Norfolk Island Pine image and Botanist knowledge base.

Figure 5. Initial segmentation and corresponding VI graph.

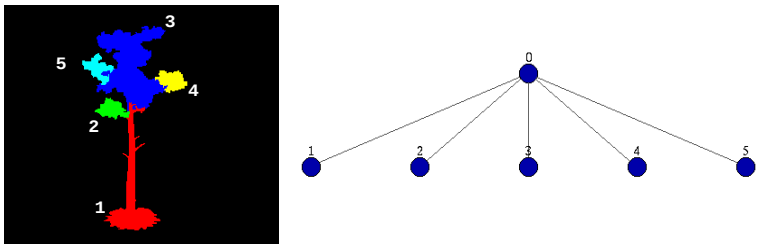


Figure 6. Final labeled segmentation of the example image.

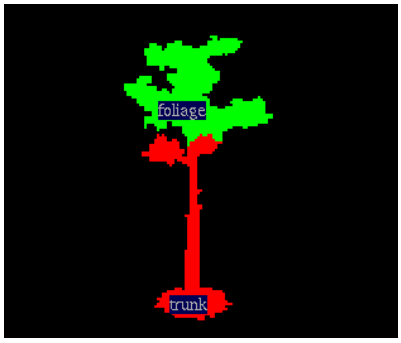


Figure 7. Text description of example scene.

```
World[0] (1.000) Consisting of:
└─ tree[11] (1.000) Of Type:
   └─ conifer[12] (1.000) Of Type:
      └─ nipine[6] (1.000) Constructed from:
         ├─ trunk[9] (0.919)
         └─ foliage[10] (0.971)
```

operator, and the SI graph is started. The basic unary matching then occurs, matching the VI nodes with the knowledge base and generating SI hypotheses linking the SI nodes to the knowledge base. When this is complete the hierarchical knowledge-based matching operator detects missing levels in the SI graph and fills these in. The VI-from-SI matching operator then propagates these down to the VI graph.

In conjunction with the SI and VI node construction operators, the knowledge-driven resegmentation operator scans the SI graph for nodes that may require resegmentation. This process is discussed in more detail in Section 2.7; suffice to say at this point that, although CITE believes the object to be an oak tree, it clearly has too many children and can be resegmented using the segmentation parameterization stored in the "Oak" KB node. This resegmentation results in the dismantling of most of the VI and SI nodes, leaving just the parent object node which it knows is a "Tree." Incompatibilities are resolved by the hierarchical relaxation labeling process, which has not been discussed in this example, but is dealt with in Section 2.6. As a result of resegmenting with more tree-specific segmentation parameters, CITE now correctly labels the new segmentation as the two parts of a "Norfolk Island Pine." The final labeled segmentation is shown in Figure 6. CITE can also produce a text description of the scene, which is listed in Figure 7 and includes the likelihood or certainty of the derived labels. This example shows the basic methods by which CITE operates. The most important aspects are the three interconnected data structures and the range of operators that create and modify these data structures.

2.4 Learning Domain Knowledge

CITE builds its knowledge base using incremental forms of supervised learning and essentially involves the system endeavoring to "track" human labeling over different images as they are viewed by the "expert." In this sense, the system recognizes or labels new data with respect to what is known at any given time. Data that is not consistent with past experience must be assumed to be a new type of concept or class, or used to update past knowledge of known classes. For this reason we have termed the actual learning procedure "explanation-based" involving both induction (generalization) from data and deduction (via relaxation-based consistency checking) from past hierarchical knowledge about the relationships between indexed image features and the knowledge base.

Learning and recognizing objects (and more general image structures) in scenes differs from more-common interpretation problems in so far as structures do not necessarily occur in fixed positions, orientations, and size, and evidence from partial information is required. For this reason we have adopted the "recognition by parts" approach where structures are defined by indexed parts and their relations, and novel learning/recognition strategies are required in such cases. One simple solution is to store a graph representation of each structure such that at recognition time the best possible cross-indexing of parts can be determined (usually a costly process using graph matching). More-sophisticated approaches have been proposed to reduce this graph subisomorphism complexity (such as the CLARET algorithm [3]), for recognizing objects more as a probabilistic mapping on parts (such as CRG and its variants [4]), or by connecting common subgraphs as used recently by Messmer and Bunke [5]. In CITE, learning of both object parts and part relations are incrementally learned according to an attribute (part: unary, relation: binary) splitting technique called Explanatory Least Generalization (ELG) which results in a decision tree that determines which attribute bounds best evidence different models. A brief description follows.

Unary (part) and binary (relational) learning algorithms First, a matching strategy is stored in each node of the knowledge base. It may be different at each node and must be able to perform three operations: to determine the degree of match or mismatch to a candidate part or relation, to learn from a new positive example, and to learn from a new negative example. Such procedures are necessary to determine, in conjunction with the user, if the current data is novel or not.

The rules are incrementally built from example positive and negative points and defined by bounds of unary or binary attributes extracted from the data up to a given event. These bounds form regions of least generalization in the feature space. The ELG algorithm builds a decision tree from these least generalizations, and this is used to determine if a test point (image part) is given a specific object label. Weightings for these labels are based on a simple distance metric to cluster centers within these ELG regions. This algorithm has been specifically designed for efficient incremental learning and rebuilding of a decision tree-based classifier, as shown in Figure 8.

The use of the decision tree built from the ELG regions (rather than using the ELG regions directly as rules) reduces this algorithm's sensitivity to noise because the decision boundaries are placed in between ELG regions. The ELG regions themselves are more sensitive to noise because

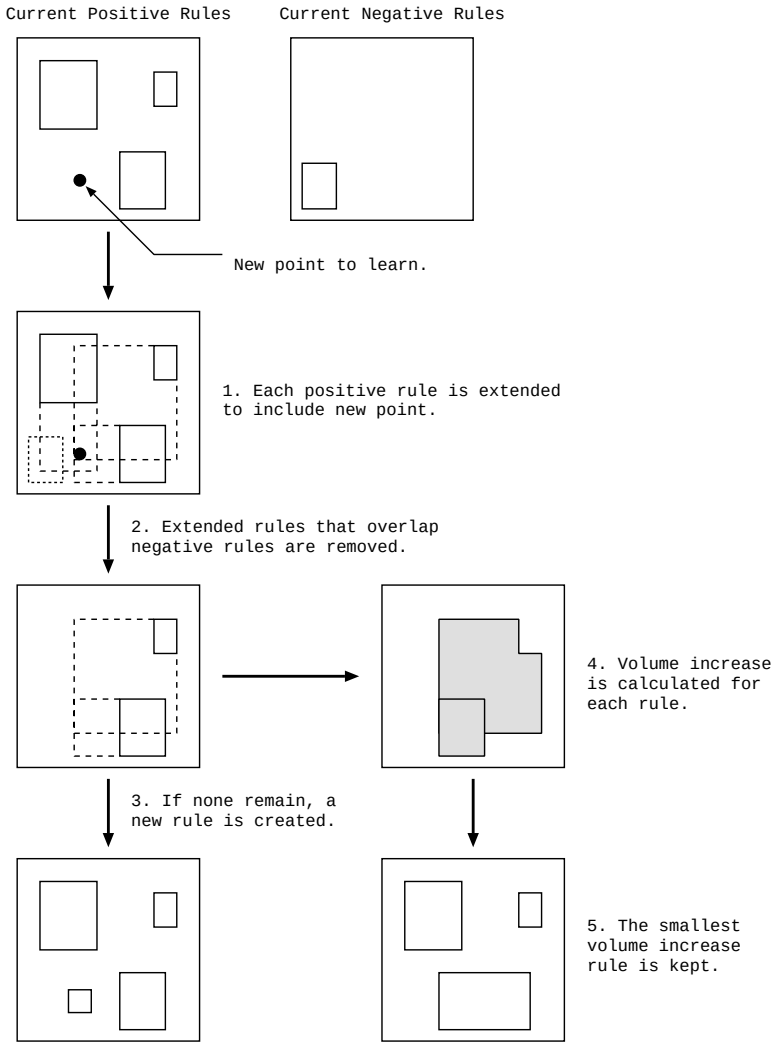


Figure 8. Learning algorithm flow chart.

they represent least generalizations of the data points they represent. Perturbations in the ELG regions will result in smaller perturbations in the decision tree, and data points that fall slightly outside the ELG region due to noise will most likely still fall within the correct decision tree region defined by the ELG region.

Matching In general, matching occurs via rule instantiation. The binary matching algorithm is run similarly to the unary matching algorithm except that the test point is a binary feature table rather than a single unary feature point. The binary (relational) matching process occurs upon identifying an unmatched parent with multiple children each of which has at least one initial label hypothesis. Binary matching can also occur as a result of the clique-resolving algorithm generating possible groupings of parts.

CITE's recognition process operates at a number of levels. At the local level, recognition is distributed between four operator classes that cover unary and binary matching, group hypothesis generation, and relaxation labeling. At a wider scale, other operators such as the knowledge-driven resegmentation and hierarchical matching also contribute importantly to the recognition outcome.

2.5 Hypothesis Generation

CITE generates classification labels and grouping hypotheses in the visual and scene interpretation structures based on segmentation results, unary and binary feature matching, and examination of knowledge base data structures. Hypotheses exist at many different levels within CITE and are resolved by a process of hierarchical relaxation labeling and constraint propagation. By extending traditional relaxation labeling to operate over a hierarchy, top-down and bottom-up recognition processes are seamlessly integrated. There are two basic types of hypothesis-generation procedures. The generation of an SI or a VI node is achieved by *unary* hypothesis procedures, and the generation of VI-VI, VI-SI, SI-SI, and SI-KB hypotheses is performed by *binary* hypothesis testing procedures. There will be some situations where multiple binary hypotheses can be generated as a result of a single unary hypothesis. For example, in clique resolving by connected components a parent VI node is generated with binary hypotheses to the image regions that are its children. This can only be sensibly achieved in a single procedure, which can generate both unary and binary hypotheses.

Generating VI nodes from image data VI nodes generated by the first segmentation will always end up as leaf nodes added under the top-level VI node. Nodes generated by subsequent resegmentations will be attached as leaf nodes at the appropriate point in the hierarchy, which is typically under the parent VI node from which the resegmentation occurred.

Generating VI nodes from the VI graph Intermediate parent VI nodes can be generated directly from the image data and VI graphs when a connected components operator is activated. Connected components grouping is used only during early training when the system is shown single objects and has not yet learned enough to reliably run the clique-resolving process. The decision to switch from connected components grouping to the clique-resolving grouping is controlled from the user interface. The algorithm for the connected components grouping is straightforward and moderately efficient.[1]

Generating VI nodes from the SI graph There are two processes that build VI nodes by examining the SI graph structure. The first is the resegmentation process that will take a VI node and its leaf node children and apply a new segmentation process to this based on the SI node knowledge base hypothesis. The second is the top-down process of ensuring that higher-level labeling structures (such as taxonomic expansions) are reflected in the VI graph. Both of these processes are top-down, one representing knowledge-driven resegmentation, the other knowledge-driven image analysis.

Generating SI nodes from the VI graph When grouping occurs in the VI graph, this structure must also be translated into the SI graph. Region grouping will occur in the VI graph as a result of connected components grouping, which is only used during early training, and the more sophisticated region grouping by clique resolving which is used a majority of the time.

1. A full discussion of linear time connected components labeling is given in [6].

Generating SI nodes from the KB graph Here, the knowledge base is examined looking for missing nodes to insert into the SI graph. This is a top-down process that expands the SI graph to include taxonomy and view information relating to already hypothesized objects.

Binary hypothesis generation by basic unary matching The basic unary matching operator generates SI-KB hypotheses for leaf VI nodes by matching them to KB nodes. Matches are made based on a ranking and threshold system, and candidates are taken from a pruned set of KB nodes based on higher-level matching. Although this algorithm runs on the VI graph, it creates SI-KB hypotheses that are stored in the scene interpretation graph.

Resolving clique membership One important factor in categorizing object recognition and scene understanding systems is their ability to isolate and identify multiple touching or overlapping objects. The process of grouping parts or image features into objects or scene elements is called the *clique membership* or *parts clique* problem.

Connectivity is a common constraint used to reduce the complexity of the clique membership problem in conventional object recognition systems. In this situation, the connectivity constraint simply states that any object or scene element in the image will be constructed from connected regions. This constraint is useful for physically constructed objects, but it is too strong for higher-level scene elements, such as those found in the airport scenes we have explored. Note that object recognition systems that are restricted to single isolated objects do not address the clique membership problem. CITE uses an image-based connected components algorithm on the presentation of the initial image. After this, the knowledge-driven clique resolving is used.

The clique membership problem is solved in CITE using interaction among a number of different processes. Operating on the VI graph, the SEGMENTGROUPCLIQUE process generates sets of possible VI parent nodes, each of which represents a possible clique. Intermediate SI nodes are then generated by the MATCHSIFROMVI process operating on the SI graph. These generated nodes then interact with the normal relaxation labeling update processes until stable hypothesis weights are achieved.

The clique membership process SEGMENTGROUPCLIQUE operates on groups of VI nodes that have SI hypotheses, which cover KB nodes with different parents. This situation is better understood by examining Figure 9. The VI node on which the algorithm is being run is indicated as the "current vi." This VI node has a number of children, each of which has an SI hypothesis. The SI nodes each have hypotheses linking them to the knowledge base. One can see in Figure 9 that children of objects labeled "A," "B," and "C" have each been hypothesized.

The objective of the clique membership process is to produce possible groupings of VI nodes based on their common KB parent hypotheses. This is done by indexing the VI nodes against the children of each node in the parent KB node list (denoted "kbplist" in Figure 9). A full set of possible matching child VI nodes is constructed by enumerating these index sets. The typical result is a small number of possible cliques that match each KB parent node. Following this, the clique membership process generates all of what now becomes an intermediate layer of VI nodes, each one representing part of a possible decomposition of the original set of VI nodes (the children of the "current vi" node).

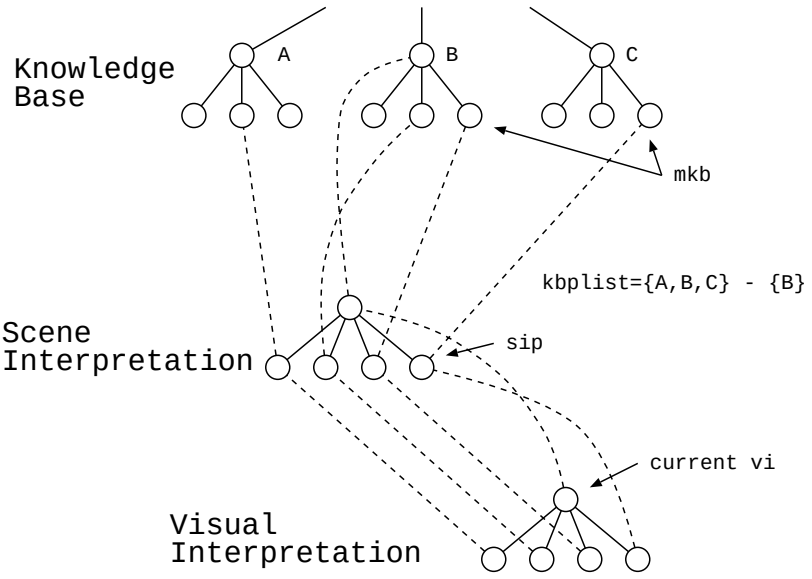


Figure 9. Illustration of variables for clique membership algorithm.

An important consequence of this approach to solving the clique membership problem is that it is achieved though local computations only. Once created, each clique is treated independently of the others. Multiple parent hypotheses in the VI graph are resolved through relaxation labeling processes. Note that the local nature of this clique membership resolving algorithm is local *with respect to the knowledge base*, not with respect to the image. The relaxation labeling for clique resolution occurs only on knowledge-based structures, not on image-based structures. This is an important knowledge-based constraint that is based on the structure of the knowledge base, and not on the structure of the viewed image.

Hypothesis generation by KB matching When groups exist in the SI graph, they need to be matched against the knowledge base to determine the most likely object labeling. This involves examining the unary matching strengths in addition to executing the binary matching strategy. CITE uses unary (part) matching to prune the possible parent object labelings. In this, possible parent labelings are pruned by the child matches, and the degree of match that is finally loaded into the parent SI node hypothesis combines unary and binary matching strengths.

2.6 Relaxation Labeling with Hierarchical Constraints

There are typically multiple labeling and grouping hypotheses generated by CITE for any image region or set of image regions. These multiple hypotheses are resolved by a process of relaxation labeling and constraint propagation. Relaxation labeling is a method for resolving multiple hypothesis labelings with respect to a compatibility function describing the compatibility between pairwise labelings. This method is essentially gradient descent and consequently cannot guarantee a globally optimal solution [7]. However, the iterative nature of the relaxation labeling process and its ability to propagate local constraints through the interaction of compatible or incompatible labels are ideal for the purposes of CITE. The following is one formulation for traditional relaxation labeling.

Let $\mathbf{B}$ be a set of objects $\{b_1, \ldots, b_n\}$, and $\mathbf{\Lambda}$ be a set of labels $\{1, \ldots, m\}$. For each object b_i we can obtain an initial label probability vector $\mathbf{P}_i^0 = (p_{i1}^0, \ldots, p_{im}^0)$, where $0 \leq p_{ij}^0 \leq 1$, for $i = 1 \ldots n$ and $j = 1 \ldots m$, and $\sum_j p_{ij}^0 = 1$, for $i = 1 \ldots n$.

Each initial probability vector is interpreted as the prior probability distribution of labels for that object, and can be computed from the initial unary and binary matching. The compatibility constraints can be described as a $n \times n$ block matrix $\mathbf{R}$, where each R_{ij} is a $m \times m$ matrix of non-negative, real-valued compatibility coefficients, denoted $r_{ij}(1..m, 1..m)$. The coefficient $r_{ij}(\lambda, \mu)$ is a measure of the compatibility between object b_i being labeled λ and object b_j being labeled μ. These coefficients are typically determined heuristically from analysis of object and part relationships, and may be restricted to the values 0 and 1 in the simplest cases. The relaxation labeling algorithm iteratively updates the probability vectors $\mathbf{P}$ using a normalized weighted sum equation:

$$p_{i\lambda}^{t+1} = \frac{p_{i\lambda}^t q_{i\lambda}^t}{\sum_{\mu=1}^{m} p_{i\mu}^t q_{i\mu}^t} \tag{1}$$

where the denominator is the normalization factor and

$$q_{i\lambda}^t = \sum_{j=1}^{n} \sum_{\mu=1}^{m} r_{ij}(\lambda, \mu) p_{j\mu}^t. \tag{2}$$

The objective is to end up with a unique label for each object. Depending on the compatibility coefficients $r_{ij}(\lambda, \mu)$, it is not possible to guarantee convergence. In this form of relaxation labeling, the number of objects is constant and the number of iterations, t, is the same for each object. However, CITE contains hierarchical knowledge and can generate and remove image regions dynamically. Static hierarchical relaxation labeling has been discussed briefly in the literature [8], but not considered further. The relaxation labeling in CITE is different for each of the categories of relationships that require labeling resolution, and is formulated in a more condensed form that is robust to the dynamic state of the hierarchical labelings. For example, the SI-KB labelings are updated according to the level of support given by the SI children *and* SI parents, as follows:

$$\begin{aligned} \widehat{P}_{i,j}^{SK} = (1 - \alpha_{pc}) & \sum_{\lambda \in S_i^C} P_\lambda^S P_{\lambda,i}^S \sum_{\xi \in K_j^C} P_{\lambda,\xi}^{SK} \\ & + \alpha_{pc} \sum_{\lambda \in S_i^P} P_\lambda^S P_{i,\lambda}^S \sum_{\xi \in K_j^P} P_{\lambda,\xi}^{SK} . \end{aligned} \tag{3}$$

The initial hypothesis value is set by the unary and/or binary matching from the operator which created the given SI-KB hypothesis. The update ratio of parent and child support (α_{pc}) reveals some asymmetry in the update procedure in terms of the relative importance of children and parents. This is necessary because, in general, there are fewer parents of an SI node than children.

In Equation 3, the double summations represent the summing over what are termed *compatibility cycles* between the scene interpretation and knowledge base graphs. The compatibility cycle is a cycle comprising four hypotheses and is computed through the parent and child chain

(right half of Equation 3). There are only three terms, rather than four, in each half of the update equation because the parent-child knowledge base link has a set hypothesis weight of 1.0. An extension to the knowledge base facilitating partial belief in the knowledge structure could be achieved by including a non-unity term into this equation.

2.7 The Segmentation Cycle

In CITE, segmentation is a controlled process in which the current expectation of scene content is used to determine which segmentation algorithm and parameterization is to be used. This is achieved by storing in each knowledge base node a list of those segmentation procedures and parameterizations that have been the most effective at segmenting the object or scene element represented by that node. Image segmentation may occur on any arbitrary subset of the image, and it is not uncommon for CITE to be executing half a dozen different segmentations at once. By closing the loop on segmentation, CITE is able to apply multiple segmentation algorithms and parametrizations to different regions of the image based on ongoing hypotheses of scene content. This provides the overall system with greater robustness to noise and distortion, and permits a much broader range of objects and scene elements to be recognized using the one methodology.

Segmentation and resegmentation occur continuously in CITE. Every node in the visual interpretation graph has a *segmentation stack* that contains the segmenter currently being executed on that part of the image represented by that node. Segmenters are added to the segmentation stack by bottom-up and top-down processes. The segmenters are executed effectively at random and will take varying amounts of time to compute depending on the algorithm being run and the size and complexity of the region.

Once an initial segmentation has been completed and hypotheses created, CITE tests these hypotheses to determine if subregion resegmentation is possible and appropriate. If the SI node currently being considered for resegmentation has multiple KB hypotheses and the best of these is significantly better than the others, then this node is considered for resegmentation according to the dictates of the KB node hypothesized. CITE accesses the knowledge base to determine the best segmenter to run on the region, and this is then loaded into the segmenter stack of the region's VI node. An SI node with a single KB hypothesis does not require resegmentation.

To determine the best segmenter to use at each resegmentation stage, CITE accesses the most likely KB parent node. If this KB node has a segmenter, this is used. If it does not, the system recursively looks back up through the knowledge base to obtain the next closest node that does contain a segmenter. Once located, the segmenter is compared to those already run on the region by examining the *segmenter history stack* in the VI node for that region. If the new segmenter has not been previously executed on this region, the resegmentation occurs.

An important graph maintenance function is undertaken when resegmentation occurs. The SI nodes and their supporting VI nodes below the SI node to be resegmented are deleted. All SI nodes with only one parent and one child above the SI node to be resegmented are also deleted. This second phase of deletion is important in situations where deep instantiations of taxonomic decomposition have been created before the resegmentation process became possible.

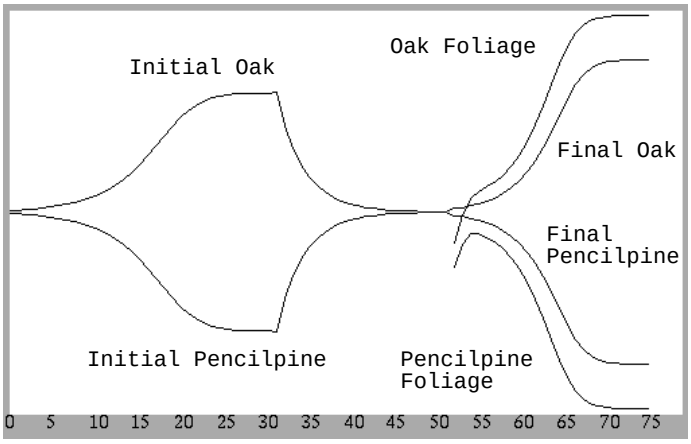


Figure 10. Hypothesis graph window illustrating selected hypothesis weights (ordinate axis is unit interval: 0-1).

Interaction of relaxation labeling and resegmentation The relaxation labeling and resegmentation processes in CITE interact quite closely in a dynamic way, which is illustrated by a simple example of a two-stage segmentation and recognition of an oak tree. Figure 10 illustrates four hypotheses as a function of time as the relaxation labeling and resegmentation processes interact. In this example, the object in question is a single tree that matches well to the "oak" and to the "pencilpine" after the initial segmentation. The object has been imperfectly segmented (in this case under-segmented) and only one part is available for matching. The initial segmentation occurs at iteration 0, followed by relaxation to iteration 30, at which point the region is resegmented. During the resegmentation the hypothesis weights move back towards equal values because the bottom-up support VI nodes have been removed. At iteration 52, the new segmentation creates the new child nodes, and the hypotheses are relaxed to their final levels. Without resegmentation, a weakly matched, under-segmented result would have been the best that could have been obtained.

Evidence of the power of resegmentation The Botanist knowledge base classification results are given as an illustration of the power of knowledge-driven resegmentation. The Botanist knowledge base used is shown in Figure 3, and a sampling of the images covered by this knowledge base is shown in Figure 11.

The classification results are shown in Table 1. Each column in this table represents results obtained with a different default segmenter (the segmenter stored in the top-level knowledge base node). The system achieves an average classification rate of 99% when operating normally. Each instance where a resegmentation has occurred is noted; for example, $(7 \rightarrow 3 \rightarrow 2)$ means that the initial segmentation gave seven parts, the first resegmentation gave three parts and the second resegmentation gave two parts. If the knowledge-driven resegmentation is deactivated, the average classification rate drops to 59%.

Segmentation algorithms All of the segmenters in CITE have a number of common properties. Each segmenter must be able to operate on an arbitrary subset of any of the images currently being processed. The segmenter must be able to operate on normalized color images and run in parallel with other segmenters running on the same or different regions. When a segmenter is loaded into a VI node for execution, it is initialized with the parameters stored in the best-hypothesized

Figure 11. Sample of images used in Botanist knowledge base.

Table 1. Botanist knowledge base classification results.

Image	NIPine Seg	PencilPine Seg	Oak Seg	Plane Seg
nipine1	Correct	Correct (2→3)	Correct	Correct (5→2)
nipine2	Correct	Correct	Correct (3→2)	Correct (7→3→2)
nipine3	Correct	Correct	Correct	Correct (4→2)
nipine4	Correct	Correct	Correct	Correct (3→2)
pencilpine5	Correct (1→2)	Correct	Correct (3→2)	Correct (7→3→2)
pencilpine6	Correct (1→2)	Correct	Correct	Correct (3→2)
pencilpine7	Correct (1→2)	Correct	Correct (1→2)	Correct (11→2)
pencilpine8	Correct (1→2)	Correct	Correct	Correct (5→2)
oak5	Correct (1→2)	Correct	Correct	Correct (4→2)
oak6	Correct (1→2)	Correct	Correct	Correct (5→2)
oak7	Correct (1→2)	Correct	Correct	Correct (4→2)
oak8	Correct (1→2)	Correct	Correct	Correct (9→2)
plane3	Correct	Correct	Correct	Correct
plane4	Correct (1→2)	Correct (1→2)	Correct (1→2)	Correct
plane5	Correct (1→2)	Correct (1→2)	Correct (1→2)	Correct
hedge1	Correct	Correct	Correct	Correct
hedge2	Correct	Correct	Correct	Correct
lantana1	Correct	Correct	Correct	Correct
lantana2	Correct	Incorrect	Correct	Correct

knowledge base node. The three segmentation algorithms available in CITE are described briefly below.

Segmentation by clustering and growing The SEGMENTSEED algorithm initially builds a localized (windowed) 3-D, colored (RGB) histogram to find seed points for region growing. The most common color is chosen as a seed point and regions are then grown from all pixels of this color. The regions may incorporate pixels of slightly different colors according to a color distance threshold. This process is repeated until all pixels in the area of the image to be segmented have been labeled.

Each region in the labeled image is then relabeled so that the labels are unique. Regions are then grown to remove small regions. A small region is one whose size is less than a specifiable ratio of the total cover. In the region-growing algorithm, regions are relabeled to the nearest neighbor with the highest-matched color. The region-growing process continues until there are no small regions.

Segmentation by clustering and ratio growing The SEGMENTERSEED algorithm was found to work well for a large number of objects, but failed on objects with low-contrast parts. This was mainly because color distance is calculated as an absolute measure, which performs poorly on badly illuminated objects. A modified version of this segmenter was constructed that computed all color differences as ratios rather than absolute differences.

Segmentation by edge extraction and merging The SEGMENTEDGE algorithm initially performs a standard edge detection (in this case, the Sobel operator was used (see Ballard and Brown [9])) based on a small local window and added across each of the three color planes. The set of edge pixels E is created by thresholding all pixels in the edge map against an upper threshold. From these edge pixels, the edges are grown in an eight-connected manner against a lower threshold. This edge growing is iterated until there are no more pixels being labeled as edges.

Once edge growing is complete, uniquely labeled regions are created by a standard recursive paint-filling algorithm. The edge pixels are then eroded away to leave a full covering of the region of the image being segmented. Regions below a specified fraction of the overall region size are then merged in a two-phase operation. The first phase considers small regions only connected to other small regions, and then propagates their labels into other small regions. The second phase looks at all remaining small regions and relabels them as their closest large neighbor, for which there will always be at least one. This region merging is ideal for complex textured regions that yield a large number of small regions placed close together. In cases where small regions may be generated not only from large textured surfaces, a different segmenter can be chosen.

2.8 Feature Extraction

The problem of determining how to adequately represent segmented regions based upon pixel statistics is difficult and is best decided via expert or other types of domain constraints. In this case, such features were calculated on single regions (*unary* features) typically including color, texture, size, and shape descriptions of the region. Relational (*binary*) features were calculated on pairs of regions and typically concentrate on geometric properties such as the distance between the regions, length

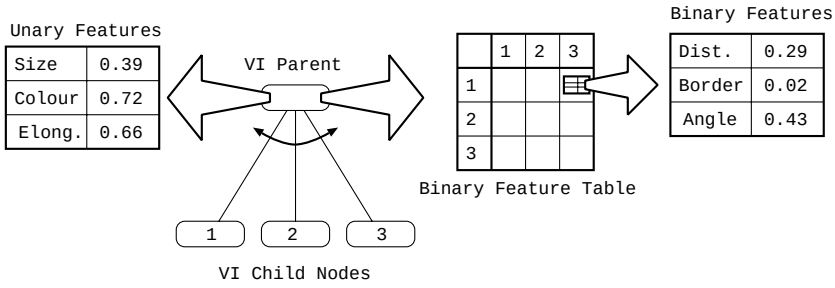


Figure 12. Storage of unary and binary features in VI nodes.

Table 2. Summary of unary features.

Feature	Type	Description
Illumination	Color	Average region illumination
Red	Color	Average region red component
Green	Color	Average region green component
Blue	Color	Average region blue component
VarRed	Texture	Variance in region red component
VarGreen	Texture	Variance in region green component
VarBlue	Texture	Variance in region blue component
Size	Geometric	Ratio of region to parent
Elongation	Geometric	Ratio of diameter to area
HeightWidth	Geometric	Height to width ratio
Straightness	Geometric	Vertical straightness measure
AOnPSqr	Geometric	Area to Perimeter ratio
Extent	Geometric	Whether region has infinite extent

of common border, and so on. It is conceivable (although not common) that binary features could describe unusual relational features such as how similarly colored or textured the two parts were.

CITE uses both unary and binary features, which can be calculated across the visual interpretation hierarchy. This represents a broadly quantized contextual scale invariance in the feature-extraction process that goes a small way to address this issue of such large information reduction.

Feature storage and calculation When calculated, features are stored in the visual interpretation nodes. Binary features stored at a node describe the relationships among the children of the node, whereas the unary features stored at a node describe the node itself, as illustrated in Figure 12.

For the purposes of presenting the learning and classification systems with uniform data, all unary and binary features within CITE are scaled to the range 0.0 to 1.0. In the case of features that have no upper bound, a reasonable scaling is applied and then the feature is clipped to 1.0.

Unary features The choice of unary features calculated for each region is based on knowledge about the current scene interpretation and learning results, as governed by the matching algorithms in the knowledge base. Table 2 lists the unary features.

The unary size feature of a region is given as the square root of the ratio of its size (in pixels) to its parent's size. A normalized height-to-width ratio of a region is calculated as the ratio between the height of the region and the sum of its height and width. The *elongation* of a region

Table 3. Summary of binary features.

Feature	Description	Directed
Size Ratio	Ratio of image sizes	Yes
Distance	Image distance to size ratio	No
Boundary	Ratio of shared boundary	Yes
Above/Below	Proportion above/below region	Yes

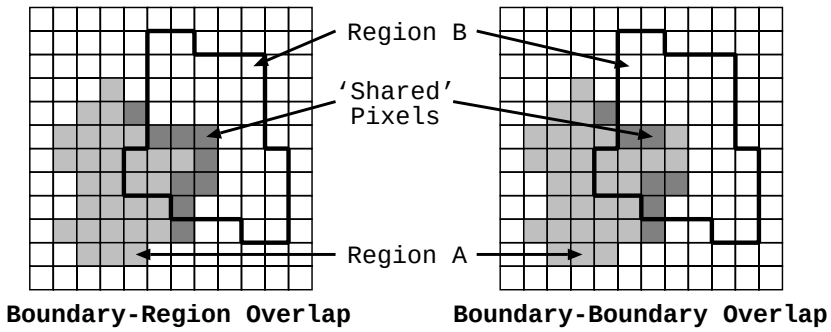


Figure 13. Two methods for computing shared boundary ratio.

is defined as twice the ratio of the maximum diameter to the perimeter. A common region shape descriptor is the ratio of area to perimeter squared, called "A on P squared," or "compactness." This feature is unitless and bounded by $\frac{1}{4\pi}$ corresponding to a circle. An estimate of the vertical straightness of a region can be obtained by fitting a quadratic line to the midpoint of each scan line in the region. The straightness of the region is then inversely proportional to the coefficient of the second-order term. The unary extent feature gives CITE an indication of whether a region is bounded. For example, sky and ground regions can be considered to have "infinite" extent.

Binary features Table 3 lists the binary features available in CITE. A binary feature is said to be *directed* if F(a,b) $\neq$ F(b,a). Most learning and binary matching systems are designed to use directed features, and they incorporate undirected features simply by computing the feature using both region orderings. CITE adopts this strategy.

The binary size ratio feature provides an indication of how large or small one region is compared to another. This is computed as a nonlinear ratio of region sizes bounded such that a region eight times larger (or greater) gives a size ratio of 1.0, and a region of identical size will give a size ratio of 0.5. A common binary feature is to compute the distance between the centroids of the two regions, scaled by their average area. Another common binary feature implemented in CITE is the fraction of shared boundary between two regions. Given the hierarchical nature of CITE's segmentation structure, and the fact that pixels may belong to a number of regions at the same level in the hierarchy, there are two ways to compute the binary boundary feature, as illustrated in Figure 13.

The *boundary-region* method counts all boundary pixels of region A that are within one (four-connected) pixel of region B. The *boundary-boundary* method counts all boundary pixels of region A that are within one (four-connected) pixel of the boundary pixels of region B. Both methods give identical results when regions are not permitted to overlap, but because this restriction is lifted in CITE the boundary-region method is used. The actual binary boundary feature is the ratio of the

overlapping boundary pixels in region A to the total boundary length of region A.

The above/below feature computes the number of pixels of region A that are above region B, n_{above}, and the number that are below region B, n_{below}, and computes the above/below feature as $R_{abovebelow} = \frac{1}{2}\left(1 + \frac{n_{above} - n_{below}}{N_a}\right)$. N_a is the total number of pixels in region A, which means that $R_{abovebelow}$ is correctly bounded between 0.0 and 1.0.

Feature invariances An important characteristic of both unary and binary features is their invariance. Of most importance to CITE is invariance to two-dimensional translation, rotation, scale, and illumination. Translation invariance is typically the easiest to obtain; however, for reasons of global positioning of objects, sometimes features are used that are not invariant to translation. The color and texture unary features are invariant to translation, rotation, and scale, but not to illumination. The geometric unary features are invariant to all four types of perturbation, although the height-width and straightness measures are deliberately variant under rotation. The binary features in CITE are all invariant to illumination, rotation, scale, and translation with the exception of the above-below feature which varies significantly with rotation. The image distance feature is the only binary feature in CITE that is symmetric (undirected).

3 System Performance and Results

CITE is demonstrated on four scenarios: views of street scenes, views of office objects, aerial views of airports, and the context-sensitive tree taxonomies. The results summary for these scenarios is listed in Table 4. In total, 253 scene elements were detected from 38 images of the four scenarios. The knowledge bases for the scenarios were constructed from a collection of 85 images. The knowledge bases themselves contain a total of 155 scene elements at different levels of abstraction. Of the 253 detected scene elements, an average classification success of 96% was achieved.

3.1 Views of Street Scenes

The street scene scenarios provide a simple demonstration of CITE in a general environment. The knowledge base was constructed from isolated and in situ examples of objects. Objects of a finite extent such as the buildings and vehicles were learned from isolated examples. The objects of unlimited extent such as the various ground covers and the sky were learned from landscapes containing just those scene elements.

In the first town image example, CITE correctly recognizes the objects present on the first pass and then uses the knowledge-driven resegmen-

Table 4. Summary of results presented.

Scenario	KB Elements	Elements Tested	% Correct
Street Scenes	24	38	95
Office Scenes	64	73	95
Airport Scenes	19	70	100
Tree Taxonomies	48	72	93
Total	155	253	96

Figure 14. Images and analysis graph of simple street scene.

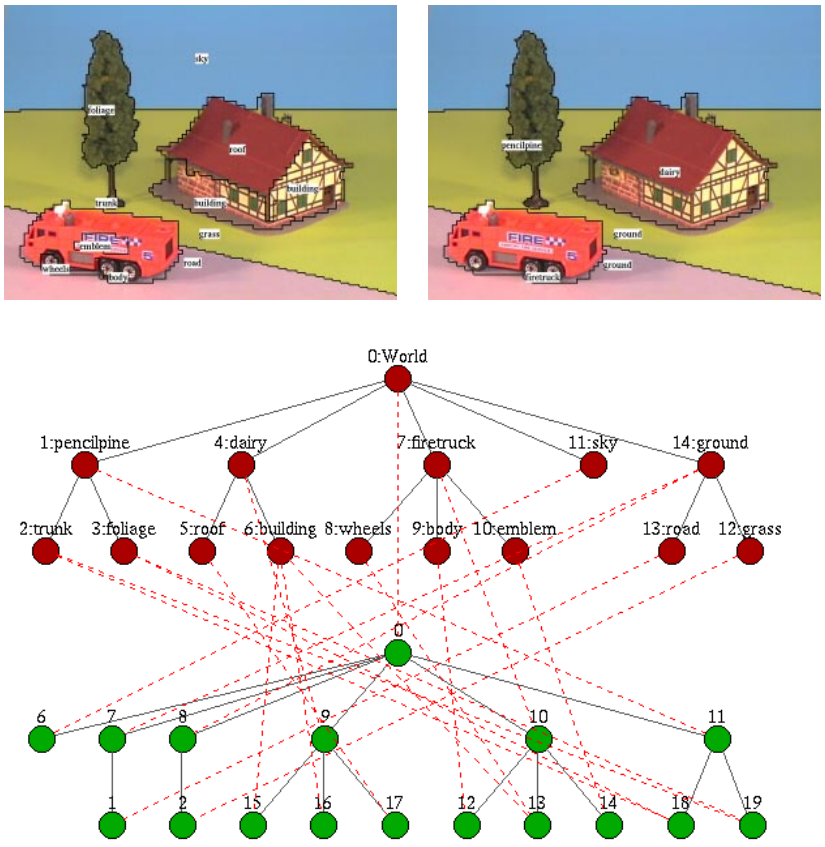


Figure 15. Text description of street scene.

World[0] (1.000) Consisting of:
- sky[6] (1.000)
- ground[7] (1.000) Of Type:
  - road[1] (1.000)
- ground[8] (1.000) Of Type:
  - grass[2] (1.000)
- dairy[9] (1.000) Constructed from:
  - building[15] (1.000)
  - building[16] (1.000)
  - roof[17] (1.000)
- firetruck[10] (1.000) Constructed from:
  - body[12] (1.000)
  - wheels[13] (1.000)
  - emblem[14] (1.000)
- pencilpine[11] (1.000) Constructed from:
  - trunk[18] (0.981)
  - foliage[19] (0.981)

tation to obtain a finer-resolution description. The resulting object labels and segment boundaries are shown overlaying the original image in Figure 14. Their interpretation is shown in Figure 15, with the results at both the leaf node description (most detailed) and at the next level up.

Figure 16. Images and analysis graph of complex street scene.

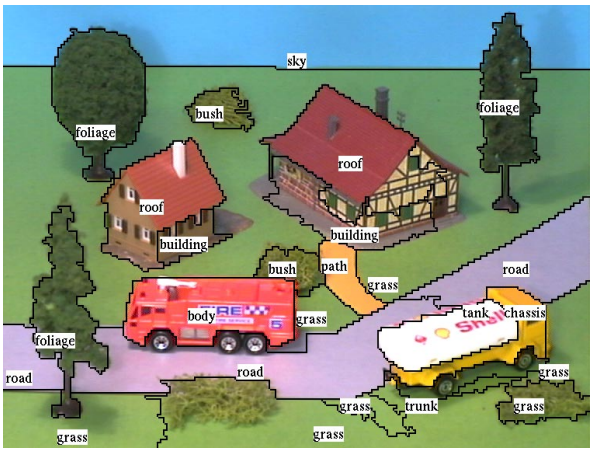


Figure 17. Text description of street scene.

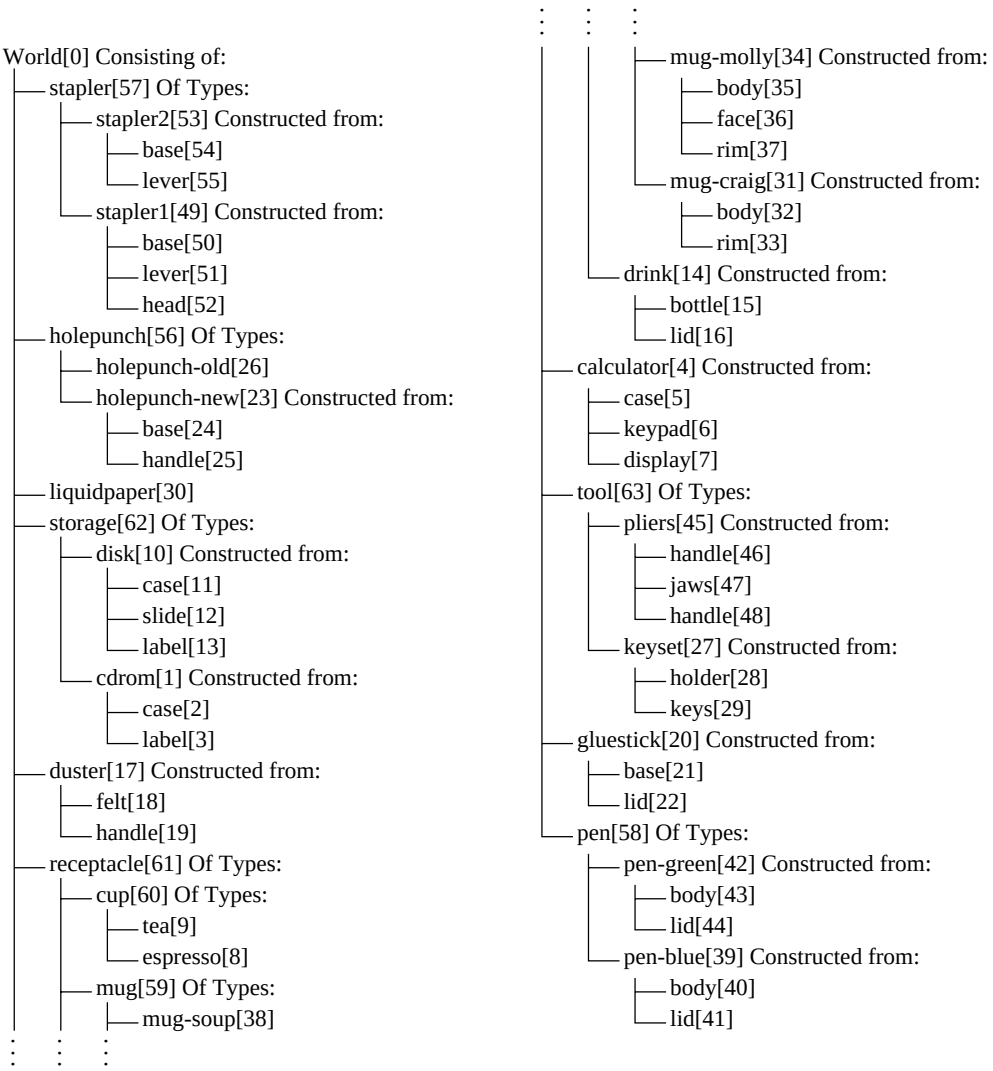
```
World[0] (1.000) Consisting of:
  sky[25] (0.876)
  ground[26] (1.000) Of Type:
    grass[1] (0.825)
  pencilpine[27] (1.000) Constructed from:
    foliage[9] (0.932)
    trunk[5] (0.878)
  pencilpine[29] (1.000) Constructed from:
    foliage[22] (0.922)
  pencilpine[31] (1.000) Constructed from:
    foliage[23] (0.939)
  house[36] (0.666) Constructed from:
    building[19] (1.000)
    roof[18] (1.000)
  fueltruck[44] (1.000) Constructed from:
    chassis[11] (1.000)
    tank[10] (0.920)
  dairy[45] (1.000) Constructed from:
    roof[21] (0.934)
    building[20] (1.000)
  ground[46] (1.000) Of Type:
    grass[2] (0.868)
  ground[47] (1.000) Of Type:
    grass[3] (0.848)
  ground[48] (1.000) Of Type:
    grass[4] (0.563)
  ground[49] (1.000) Of Type:
    road[6] (1.000)
  ground[50] (1.000) Of Type:
    grass[7] (0.687)
  ground[51] (1.000) Of Type:
    road[8] (0.830)
  firetruck[52] (1.000) Constructed from:
    body[12] (0.842)
  ground[53] (1.000) Of Type:
    grass[13] (1.000)
  ground[54] (1.000) Of Type:
    grass[14] (0.716)
  ground[55] (1.000) Of Type:
    path[15] (1.000)
  ground[56] (1.000) Of Type:
    road[16] (1.000)
```

In the more complex street scene shown in Figure 16, a total of 24 scene elements have been detected. Of these, one is a misclassification and three are the result of spurious segmentations. One of the spurious segmentations, the shadow under the fuel truck, generates an incorrect "trunk" object. The other two are under and over segmentations, but are correctly labeled as "grass." Figure 17 shows the full text description generated by CITE, including the groupings of the low-level parts into constituent objects.

3.2 Views of Office Objects

This section illustrates CITE in operation recognizing small collections of objects from a database of twenty objects arranged in a hierarchy containing 64 elements. The full knowledge base is listed in an expanded text form in Figure 18. Following this are four figures showing the results

Figure 18. Text description of office knowledge base.



for the system labeling and interpreting small collections of partially overlapping objects, with results perfectly consistent with the knowledge base.

Four of five objects in Figure 22 are correctly recognized, with the teacup being mismatched for part of a type of mug. This small error can be corrected with subsequent incremental supervised learning, but has been included to demonstrate one possible failure mode of CITE. In total, five previously unseen images were tested using this knowledge base, with an average correct labeling rate of 95%.

3.3 Aerial Views of Airports

The objective of this section is to demonstrate the operation of the clique-resolving process and the hierarchical relaxation labeling as a constraint-propagation mechanism. To demonstrate these features of CITE, seven aerial images of an airport are analyzed to determine not only the labeling of the constituent parts, but the local configurations that they are in. These local configurations have been chosen to heavily overlap so that the system must rely on relational information to resolve ambiguities.

Figure 19. CD, keyset, gluestick, and drink scene.

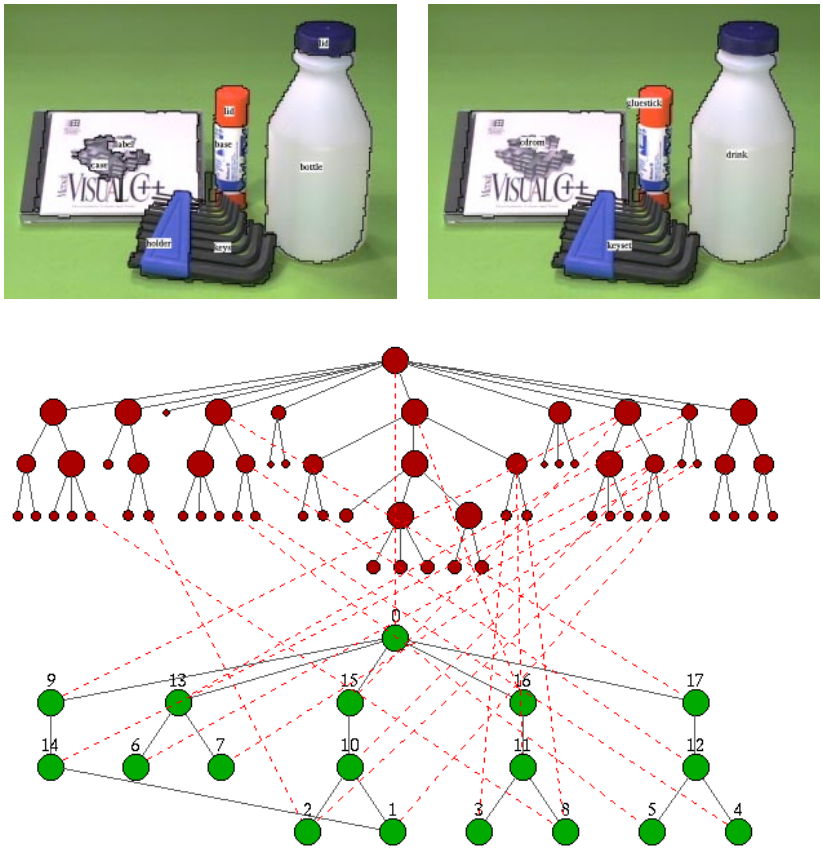


Figure 20. Text description of CD, keyset, gluestick, and drink scene.

World[0] (1.000) Consisting of:
- gluestick[13] (1.000) Constructed from:
  - base[6] (1.000)
  - lid[7] (0.831)
- tool[15] (1.000) Of Type:
  - keyset[10] (1.000) Constructed from:
    - holder[2] (1.000)
    - keys[1] (1.000)
- receptacle[16] (1.000) Of Type:
  - drink[11] (0.896) Constructed from:
    - bottle[3] (0.906)
    - lid[8] (0.891)
- storage[17] (1.000) Of Type:
  - cdrom[12] (0.545) Constructed from:
    - case[5] (0.725)
    - label[4] (0.706)

Figure 21. Two pens, two cups, and holepunch scene.

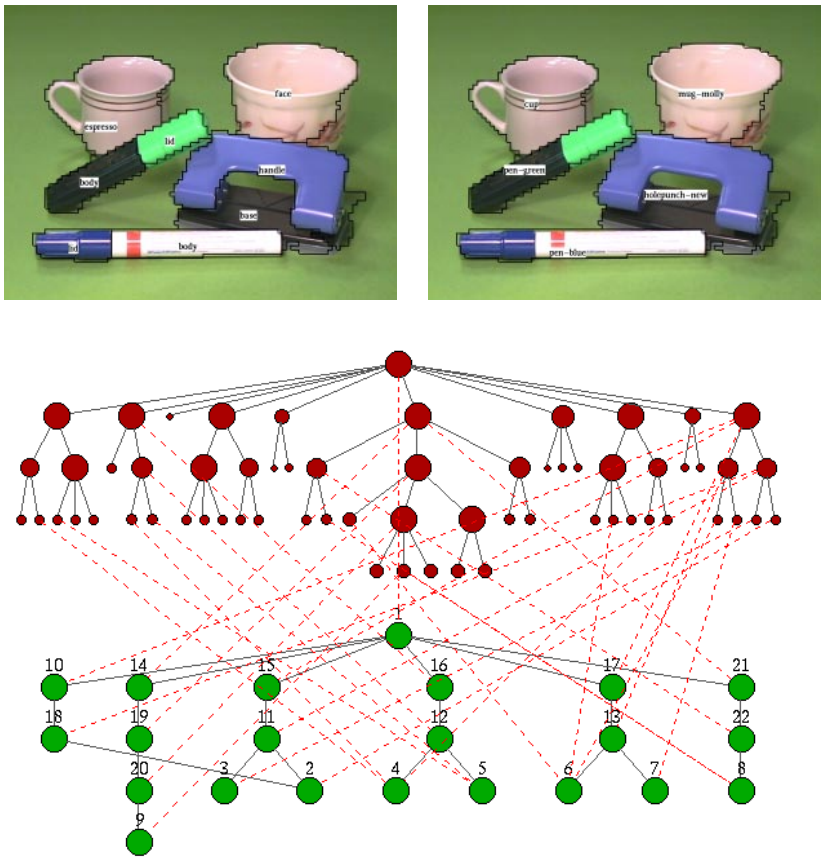
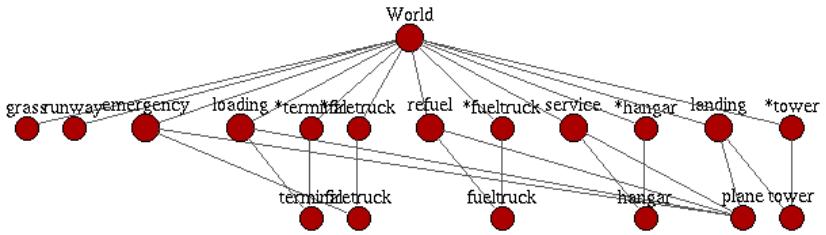


Figure 22. Text description of pens, cups, and holepunch scene.

World[1] (1.000) Consisting of:
- receptacle[14] (1.000) Of Type:
  - mug[19] (1.000) Of Type:
    - mug-molly[20] (1.000) Constructed from:
      - face[9] (1.000)
- pen[15] (1.000) Of Type:
  - pen-blue[11] (1.000) Constructed from:
    - body[3] (1.000)
    - lid[2] (1.000)
- holepunch[16] (1.000) Of Type:
  - holepunch-new[12] (1.000) Constructed from:
    - base[4] (1.000)
    - handle[5] (1.000)
- pen[17] (1.000) Of Type:
  - pen-green[13] (1.000) Constructed from:
    - body[6] (1.000)
    - lid[7] (1.000)
- receptacle[21] (1.000) Of Type:
  - cup[22] (1.000) Of Type:
    - espresso[8] (1.000)

Figure 23. Full knowledge base used in airport analysis.



The full knowledge base in shown in Figure 23. This was constructed by learning each object from a single image, followed by learning each higher-level configuration from full-scene examples. As can be seen, the "plane" object plays a central role, being involved in five higher-level objects: the "loading," the "emergency," the "service," the "landing," and the "refueling" objects. When a plane is detected, the system must use relational information to determine which of the five possible parent objects the plane is most likely to belong to, and then propagate this through the scene interpretation graph using relaxation labeling. The clique-resolving operator generates the most likely set of possibilities, and relaxation labeling resolves these to determine a subset of high compatibility.

A further complication in these scenes is that a number of objects in the database can appear in isolation, or as part of a higher-level object. The determination for this in each case rests solely on the strength of the binary matching, and the propagation of the resulting hypotheses through the scene interpretation structure. These nodes are those whose names are preceded by an asterisk are described as "Not-Always-Partof."

The first result (Figures 24 and 25) shows one single instance situation, and the second result (Figures 26 and 27) shows a situation in which two planes are involved. In this second test, CITE must determine not only if a particular situation is occuring, but which objects are involved in that situation. In all cases this is determined solely by the propagation of relational matching through the hierarchical scene description. Five other images were tested, and CITE correctly recognized the various objects and the higher-level situations in which they were involved.

4 System Operation

CITE is implemented as a single X-Windows Unix program. The graphical user interface uses the Motif widget set and was developed on a Silicon Graphics Indy using GNU C++. The system dynamically determines whether the live video capture hardware is present, and if so allows the user to grab images directly. However, most of the testing of CITE was done using pre-saved images that can be loaded into CITE at any point. Each of the main windows of the CITE system is described in the following paragraphs. However, for more details and downloading instructions, consult `www.cs.curtin.edu.au/~cdillon/`.

The Command Window

The command window is the main control window in CITE. It lets the user enter commands from pulldown menus or from a one-line text

Figure 24. Refueling image and analysis graph.

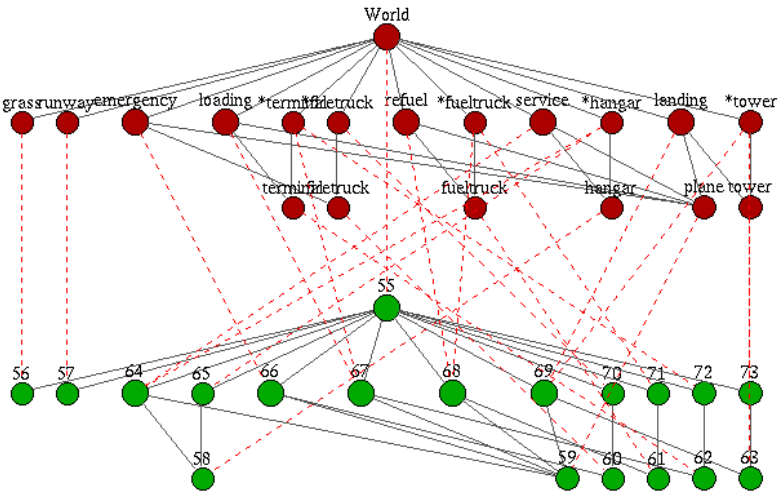
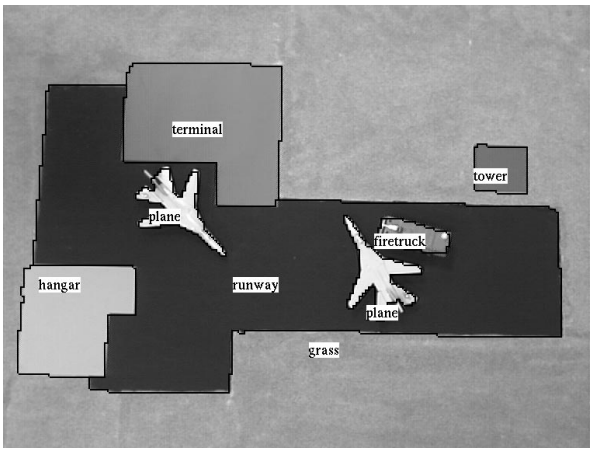


Figure 25. Text description of refueling scene.

World[55] (1.000) Consisting of:
- grass[56] (1.000)
- runway[57] (1.000)
- *hangar[65] (1.000)
- refuel[68] (0.946) Constructed from:
  - fueltruck[61] (1.000)
  - plane[59] (1.000)
- *firetruck[70] (1.000)
- *terminal[72] (1.000)
- *tower[73] (1.000)

Figure 26. Emergency and load image and analysis graph.



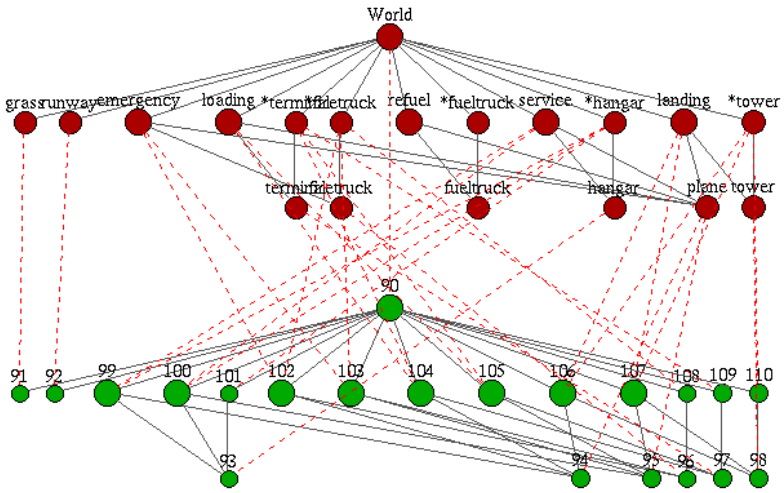


Figure 27. Text description of emergency and load scene.

World[90] (1.000) Consisting of:
- grass[91] (1.000)
- runway[92] (1.000)
- *hangar[101] (1.000)
- emergency[102] (0.946) Constructed from:
  - plane[94] (1.000)
  - firetruck[96] (1.000)
- loading[105] (0.946) Constructed from:
  - plane[95] (1.000)
  - terminal[97] (1.000)
- *tower[110] (1.000)

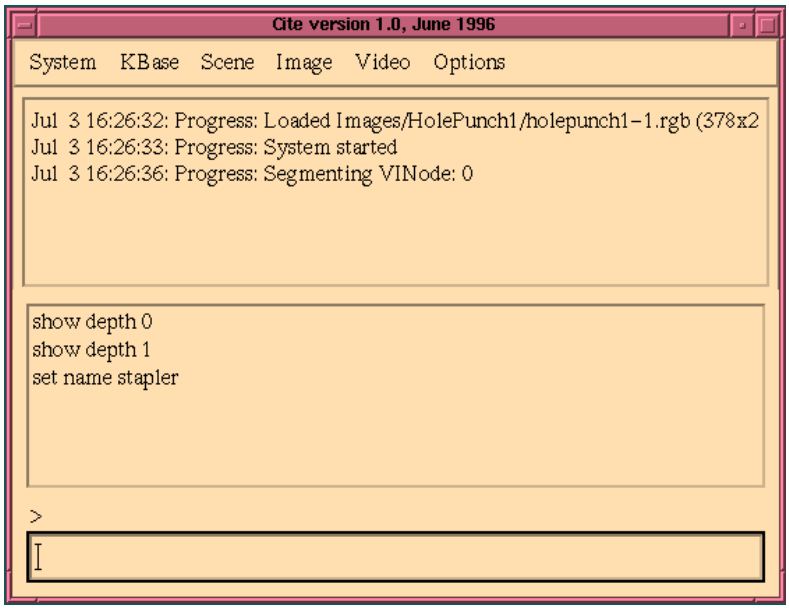


Figure 28. The main command window in CITE.

command prompt. Figure 28 shows the command window as it normally appears. The top text window contains the status and progress-reporting messages from CITE. The middle text window is a history of the previous entered commands, and the lower text window is the command prompt.

Under the main menu in the command window are menu options that duplicate the commands that can be entered at the command prompt. In addition, there are menu options for displaying a live video window and for grabbing the live video input and saving it to a file (Silicon Graphics machines only). There is an additional window under the "Options" window which contains a slidebar for each of the real-valued options under the "set" command.

The Image Window

The image window contains the images of the current scene and their segmentations. When labeling is complete the labels are placed over the segmented image. The user can choose to display the segmentation at any level, such as at leaf nodes, at parents of leaf nodes, and so on. (See Figure 29.)

The Data Graph Window

The data graph window is the most important visual display in CITE. It contains three panels arranged vertically. The top panel contains the knowledge base, the middle panel contains the scene interpretation, and the bottom panel contains the visual interpretation for each image.

There are many display options that control what is displayed on the screen. This includes displaying hypotheses from the visual interpretations to the scene interpretation, and from the scene interpretation to the knowledge base. The user can also select nodes for editing, grouping, or deleting with the mouse. The data graph window is shown in Figure 30.

The menu options in this window enable various levels of display detail of hypotheses. There is a considerable amount of information in the three graphs represented in this window, and to display all this at once would be virtually impossible. Nodes may be selected by clicking

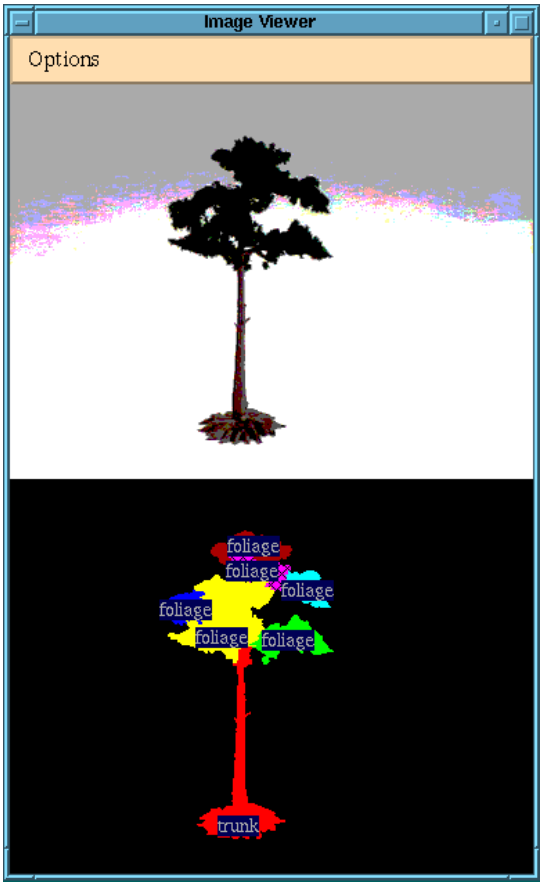


Figure 29. The image and segmentation window.

on them with the left mouse button. Clicking on them with the right button brings up the display window for that node. There are three basic types of node display window: one for VI nodes, one for SI nodes, and one for KB nodes.

The Hypothesis Weight Window

A more detailed analysis of the hypothesis weights can be obtained via the hypothesis weight window. This displays a graph of hypothesis weights as a function of time. The weights that are displayed can be individually selected from the node display windows by clicking on the text display of the particular hypothesis of interest. Figure 31 shows the hypothesis weight window with three SI-KB hypotheses selected. Every hypothesis has a unique integer identifier, and this is displayed on this window as well as the node display window.

The horizontal scale in the hypothesis weight window is the iteration number where each iteration represents one complete pass through the process stack when a change occurred in at least one hypothesis. This prevents the system running to infinity with subsequent loss of the interesting detail of the hypothesis curves. The horizontal axis is autoscaling, and the vertical axis ranges between 0.0 and 1.0.

The profile window CITE has a cyclic process scheduler that runs each loaded operator for a short period before going to the next. The total amount of time spent in each operator is stored in the operator description class and displayed in the profile window shown in Figure 32.

Figure 30. The data graph window.

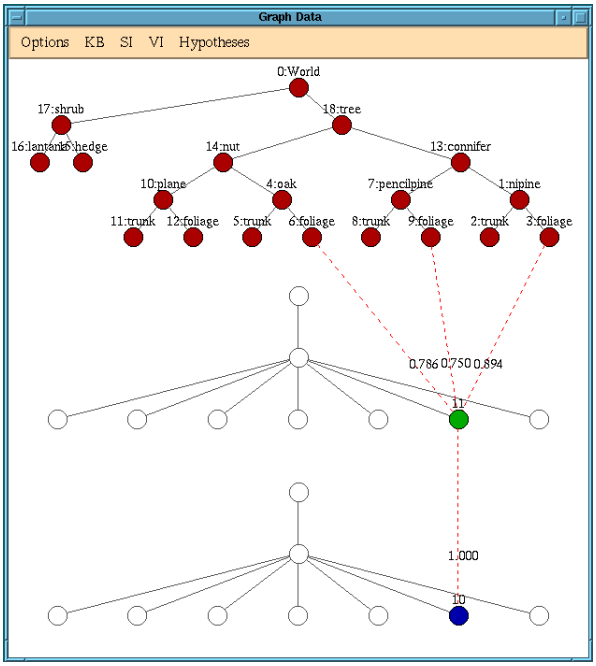
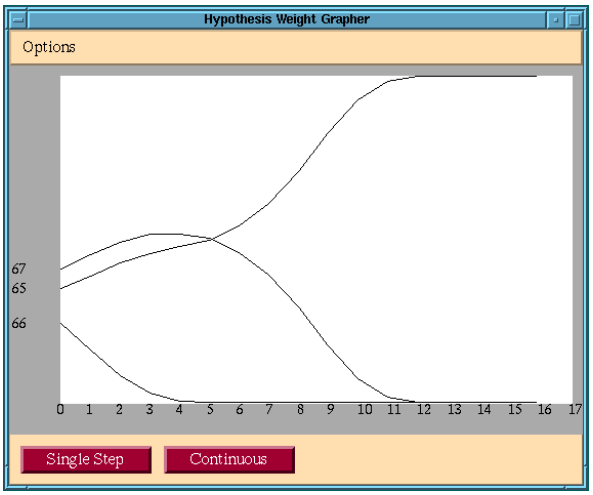


Figure 31. The hypothesis graphing window.



The columns in the profile window show the total time spent executing each operator, the overall percentage, a short-term percentage, a long-term percentage, and the absolute time spent in the most previous execution. In addition there is a performance measure expressed relative to the original development platform of a 150 MHz R4400 processor. The profiler gives useful information as to the relative computation required for each subtask within CITE.

Results Window

In order to produce the highest quality results, there is an additional results window that shows the image in the original size with black-bordered regions and the region labels in a more printable font. The results window is shown in Figure 33 and is displayed from a menu entry under the main system menu in the command window.

Figure 32. The operator profile window.

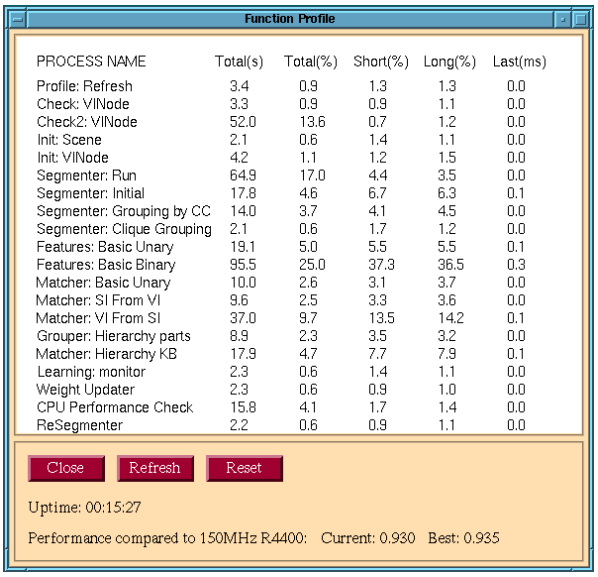
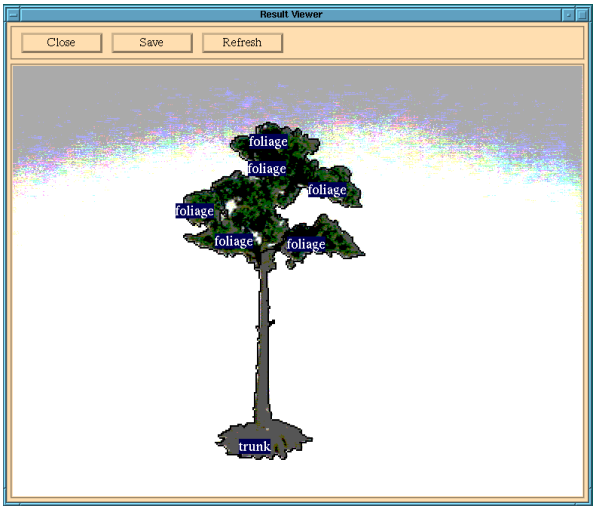


Figure 33. The results window.



5 Conclusion

In this paper we have presented a new system for domain-specific image annotation and interpretation that integrates bottom-up and top-down processing to integrate object recognition and scene-understanding capabilities within the same framework. The more novel aspects of this paper are in the use of hierarchical structures to describe world knowledge in addition to scene and visual decompositions, knowledge-driven resegmentation, incremental supervised learning methods, and hierarchical relaxation labeling.

The explanatory least generalization on decision trees is presented as one method of converting non-incremental learning algorithms into an incremental form that obeys the identified desirable qualities of sub-linear growth in memory and computational complexity. Results are presented illustrating the performance improvement gained by closing the loop on segmentation with the knowledge-driven resegmentation algorithm.

Again, CITE demonstates that the procedures required to build systems that accomplish understanding image components, as they reference past knowledge, as expressed by domain-specific annotation, is complex and requires combining vision, learning, and knowledge representation. Although all these processes have been isolated in human cognition and, to some extent, in the cerebral cortex, the types of processes and their operations used in CITE were necessary for us to implement a trainable and robust system without making any claims that this is exactly how humans solve such tasks.

6 References

[1] Draper, B., Collins, T., Brolio, J., Hanson, A., and Riseman, E. The schema system. *International Journal of Computer Vision*, 2:209–250, 1989.

[2] Matsuyama, T. and Hwang, V. *SIGMA: A Knowledge-Based Aerial Image Understanding System*. Plenum Press, 1990.

[3] Pearce, A. R., Caelli, T., and Bischof, W. F. Rulegraphs for graph matching in pattern recognition. *Pattern Recognition*, 27(9):1231–47, 1994.

[4] Bischof, W. F. and Caelli, T.. Learning structural descriptions of patterns: A new technique for conditional clustering and rule generation. *Pattern Recognition*, 27(5):689–97, 1994.

[5] Messmer, B. T. and Bunke, H. Subgraph isomorphism detection in polynomial time on preprocessed model graphs. In S. Li, D. Mital, E. Teoh, and H. Wan, editors, *Recent Developments in Computer Vision. Second Asian Conference on Computer Vision, ACCV '95, Singapore, Invited Session Papers*, pages 373–82, Berlin, Germany, Springer-Verlag, December 1995.

[6] Dillencourt, M., Samet, H., and Tamminen, M. A general approach to connected-component labeling for arbitrary image representations. *Journal of the ACM*, 39(2):253–280, April 1992.

[7] Rosenfeld, A., Hummel, R., and Zucker, S. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):420–433, June 1976.

[8] Davis, L. and Rosenfeld, A. Cooperating processes for low-level vision: A survey. *Artificial Intelligence*, 17:245–263, 1981.

[9] Ballard, D. and Brown, C. *Computer Vision*, chapter 9, pages 291–311. Prentice-Hall, Inc., 1982.