Recognition and Learning with Polymorphic Structural Components

Mark Burge, Wilhelm Burger and Wolfgang Mayr

Johannes Kepler University, Dept. of Systems Science, Computer Vision Laboratory, Linz, Austria

We address the problem of describing, recognizing, and learning *generic*, free-form objects in real-world scenes. For this purpose, we have developed a hybrid appearance-based approach where objects are encoded as loose collections of parts and relations between neighboring parts. The key features of this approach are: part decomposition based on local structure segmentation derived from multi-scale wavelet filters, flexible and efficient recognition by combining weak structural constraints, and learning and generalization of *generic* object categories (with possibly large intra-class variability) from real examples.

Keywords: recognition, learning, polymorphic structural components, Gabor probe structural description.

Recognizing and Classifying 3D Objects

Recognizing three-dimensional objects under different viewing and lighting conditions is a traditional problem in computer vision. The difficulty of the problem depends upon many factors including: types of objects, number of classes, inter- and intra-class variability, number of objects in a scene, background complexity, ammount of occlusion, etc. We focus on recognizing the class of single objects scenes (e.g., *chair*, *table*, *bench*, etc.) which often exhibit high intra-class variability.

Generally, all such recognition approaches assume a noiseless, pre-segmented image with all parts of the object visible in every view, i.e. no self-occlusion, and that models of all objects have been given *a priori*. Even with these restrictions there has been only limited success for small datasets e.g. Bergevin and Levine's PARVO system[BL93] which uses volumetric geons to successfully discriminate among 23 objects. The subgraph isomorphism problem forces them to compare only those models which contain the same geons as the view, preventing its use in cases of views with missing parts.

Techniques from machine learning are increasingly being incorporated into object recognition methods. Using the ability of these techniques to generalize from the training data previously unseen views can be recognized with increased accuracy. One class of these new methods uses the relative frequency of a class in a region as an evidence-based approach[JH88] to generalization while others use neural networks[Ede93], Eigenvalue decompositions[MN95], or decision trees[BC94]. The decision tree method forms the basis of our approach.

An Appearance and Structural Approach

A number of different recognition approaches has been devised which can be roughly divided into those using a 3-D model and those based on one or more 2-D views of each object, i.e., based on object *appearance*. Our approach is appearance-based and structural, i.e., typical views of an objects are described in terms of the configurations of their parts [Bie87, Low87, Bro81, Fu92].

Structural representation and recognition approaches are attractive because they emphasize: shape, spatial arrangement, and topology of a pattern, thereby ensuring a high tolerance against: variations in lighting conditions, object deformation, articulation, and occluded, missing, and extraneous parts. This makes them well suited to describe objects in terms of characteristic views, because the visible parts and the

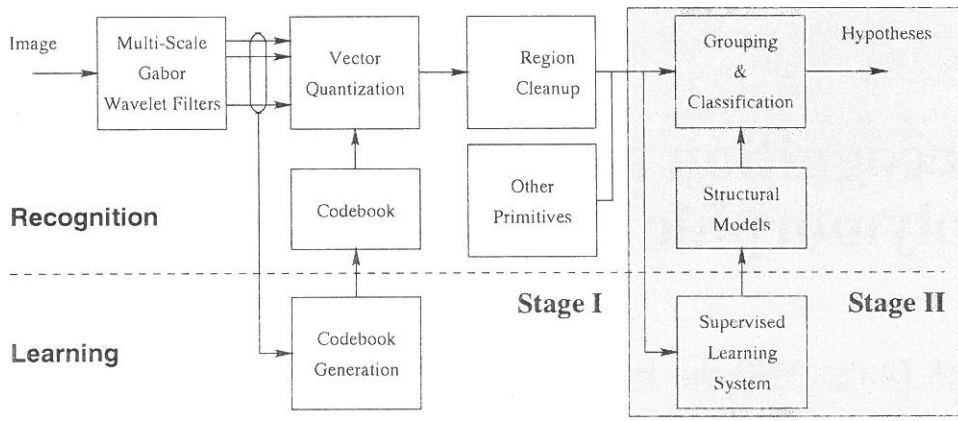

Fig. 1. System Overview.

structure of a view remain largely unchanged over large regions of the view sphere.

Structural recognition methods are based on both the availability of primitive structural elements like: straight line segments, corners, blobs, arcs, and other parametric strokes, and on the assumption that these elements can be extracted from the image data with sufficient reliability. Regardless of what the primitives are, the performance of the recognition process depends critically on how reliably they can be extracted, which is difficult even under ideal viewing conditions. When images are noisy and cluttered, the extraction of suitable primitives using only local information may not be possible. In addition, practically all structural feature extraction schemes work in a rather myopic fashion, trying to assemble larger meaningful structures from scattered pieces using weak local evidence and therefore often missing the dominant global structures of the resulting image description (segmentation). In practice, most current structure extraction schemes are fragile and no single method exists that can reliably deliver a good part decomposition, unless the scene domain is highly restricted. Table 1 gives a comparision of recent approaches to structural pattern recognition.

We attempt to overcome the problem of fragile segmentation by a three pronged method. Firstly, instead of relying upon a single type of primitive, we combine *multiple* classes of primitives into a single, polymorphic representation and recognition scheme, introducing additional redundancy. Secondly,we do not require structural primitives to be precisely delineated but only their approximate spatial position and shape properties are needed, as in Lindeberg's[Lin93] blob features or Perona's[LBP95] and others[Bur88] specific local patterns. In particular, parts may be overlapping and ambiguous. Finally,structural primitives do not need to correspond to parts such as would be meaningful to a human observer, it is only necessary that they can be recovered reliably and repeatably from an image.

Gabor Probe Structural Description

The main steps in the recognition (Figure 1) are preprocessing, parts extraction and preliminary classification, followed by combined grouping and object classification. In one of our feature extraction methods preprocessing of the image data consists of applying a set of Gabor quadrature filters with N_ω log-spaced center frequencies ω_k and N_ϕ regularly spaced orientations ϕ_l [BB94].

For each pixel location $\mathbf{x}_i$, we compute the values

$$\begin{aligned} J^+_{k,l}[i] &= (I * G^+_{\omega_k,\phi_l})[\mathbf{x}_i] \\ J^-_{k,l}[i] &= (I * G^-_{\omega_k,\phi_l})[\mathbf{x}_i] \end{aligned}$$

where I denotes the original image, $(G^+_{\omega_k,\phi_l}, G^-_{\omega_k,\phi_l})$ is a Gabor quadrature filter pair with center frequency ω_k and modulation orientation ϕ_l, and $*$ is the convolution operator. $G^+_{\omega_k,\phi_l}$ and $G^-_{\omega_k,\phi_l}$ are the cosine and the sine Gabor filter kernels, respectively. The spacing in frequency is Δ_ω, such that $\omega_k = \omega_0 \cdot \Delta^k_\omega$ for $1 \leq k \leq N_\omega$, and the spacing in orientation is $\Delta_\phi = \frac{\pi}{N_\phi}$, such

that $\phi_l = \phi_0 + l \cdot \Delta_\phi$ for $1 \leq l \leq N_\phi$, for given $\omega_0, N_\omega, \phi_0$, and N_ϕ.

For typical values of $\omega_0 = 0.05\pi$, $N_\phi = 4$, $\Delta_\omega = \sqrt{2}$, and $N_\omega = 4$, we obtain a 32-element *Gabor probe* $\mathbf{G}[i]$ for each image location $\mathbf{x}_i$.

Fig. 2. Initial structural description e.g. Women's Face.

The output of this filter bank is a 32-element vector $D_{\omega,\phi}(\mathbf{x})$ (called a *Gabor probe*) that describes the structure around the image location $\mathbf{x}$ at multiple scales. Subsequently, the Gabor probes are classified into N (we use $N = 64$) structural event types, using vector quantization with a codebook derived from a representative set of training images. Contiguous areas in the resulting label image $L(\mathbf{x}) \in [0, N-1]$ form a set $\mathcal{R}$ of regions with approximately homogeneous image *structure* (Figure 2). Each region $R_i \in \mathcal{R}$ is characterized by *(a)* its *shape* and *(b)* its *codebook index* which describes the corresponding image structure in terms of the Gabor filter response. Both combined supply a strong set of unary region features that are useful for efficient indexing and recognition.

Polymorphic Primitives

Most structural recognition methods (and representation formalisms) are based on a single type of primitive, e.g., straight line segments. Some approaches combine different primitive types of the same *class* of primitives, e.g., straight line segments are sometimes used in combination with circular arcs, both being contour-type primitives. This results in potentially very powerful pattern descriptions (considering that these are sufficient for cartoon-type illustrations which are highly expressive for a large variety of real-world scenes). For some reason it is difficult, however, to achieve similar recognition performance with stroke-type features alone. The combination of different primitive *classes*, e.g., stroke-based and area-based primitives, adds another level of descriptive power. We refer to collections of structural primitives from different primitive classes as *polymorphic*.

Why are polymorphic primitives more powerful? First, because they increase the available alphabet and thus the number of unary features available for single-stage decisions, thereby reducing the combinatorial requirements and making recognition more efficient. Another advantage is that polymorphic primitives originate from separate and (largely) independent operators, such that they complement each other. In particular, this reduces the probability that crucial parts of an objects are not detected at the feature level and also makes the recognition process more general since the features are less biased towards a particular type of scene. For example, contour-type primitives may be quite successful on indoor scenes containing man-made objects but fail completely on outdoor scenes, where area-based features may perform well. In general, each feature type exhibits certain strengths and weaknesses not only on different scene types but also on different objects in a scene and different areas of an object.

Why are polymorphic descriptions not more popular? One can think of several reasons:

- The appeal of generating a clean segmentation from an image, in which segments correspond to (semantically) meaningful parts of an objects. This is generally not possible in a polymorphic approach. One can segment an image into polymorphic parts that are mutually disjoint (such as done in [FF95]), but this only works for non-overlapping components and does not provide for multiple and competing hypotheses.
- One usually has enough problems with a single feature detector or segmentation algorithm, such as unreliable results, difficulties in combining primitives into larger assemblies.

- How to merge different kinds of primitives into a combined description is not straightforward, as attribute sets differ among feature classes. This is mainly a problem with binary and higher order relations between parts. While it is easy to relate two straight line segments in 2-D space, for example, relations between parts of arbitrary classes are more difficult to specify.

Learning and Recognition

Bischof and Caelli's[BC94] decision-tree based *Conditional Rule Generation* (CRG), improves upon the previously mentioned PARVO system in its dealing with missing parts. During training and classification it exhaustively generates neighboring part paths so that, if during classification a part is missing, a path without that part might still be found in the classification tree and the object may still be correctly classified. Another improvement on the PARVO system is their method of automatic model acquisition. The CRG method encodes the models into the rules of the decision tree. This method forms the basis of our recognition method which we have extended in several directions.

In the CRG method an attributed graph $G(\mathcal{P}, \mathcal{E}, \mathcal{U}, \mathcal{B})$ is built from a pre-segmented image. The image has been pre-segmented into regions, each of which is considered a "part" of the object. As not all important "parts" in an object are region based, we have extended the CRG method to use polymorphic components, or other types of parts, e.g. stroke-based features. Figure 3 gives an overview of the learning and recognition stages.

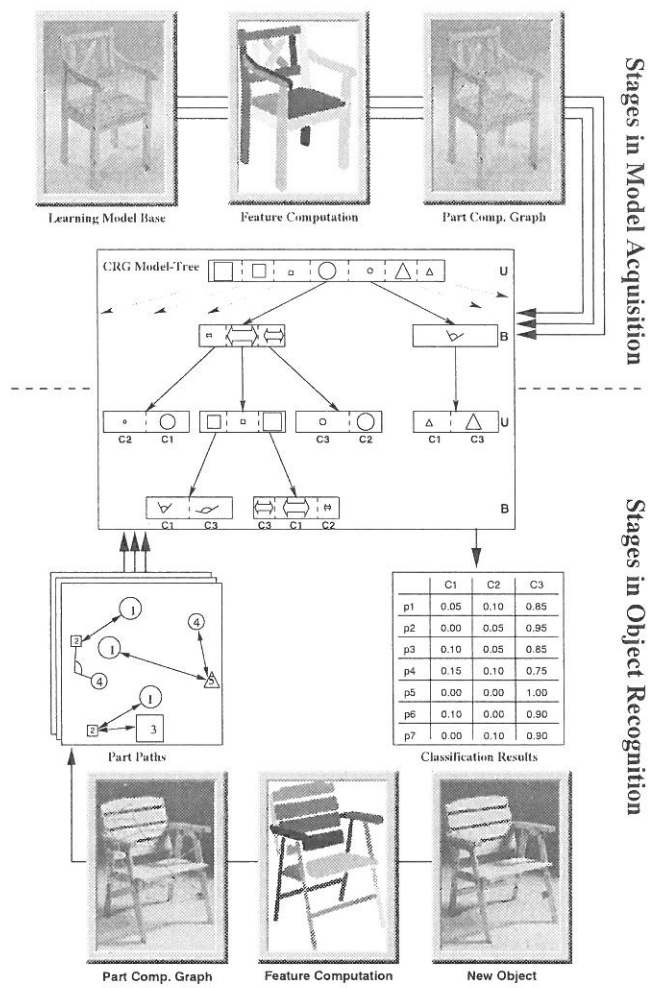

Fig. 3. Flowchart of the Learning and Recognition process: (a) *Learning path: computation of the features from all images in the database; building of a part compatiblity graph (PCG); construction of a classification tree for part paths. (b) Recognition process: presentation of a new image; feature computation; PCG; extraction of part paths; classification of each part path using the classification tree; combining evidence.*

The PAG graph G consists of a set of nodes which represent the set of parts $\mathcal{P}$ of the image. These parts are connected by a set of edges $\mathcal{E}$ which represent relations between the parts. To avoid the exponential number of edges in a fully connected graph only edges between neighboring parts or parts within a certain distance are included. We present an improvement for the neighboring relation, in this case, based upon part similarity.

Each part $p_i \in \mathcal{P}$ is attributed by a unary feature vector $u_i \in \mathcal{U}$ with a predetermined number of features, each edge $e_{i,j} \in \mathcal{E}$ is attributed by a binary feature vector $b_{i,j} \in \mathcal{B}$ whose features are computed from the parts p_i and p_j $(i \neq j)$. The feature vectors u_i and $b_{i,j}$ form the unary and binary feature spaces $\mathcal{U}$ and $\mathcal{B}$ respectively. In the *learning* phase the problem is learning how to classify a part p_i into the correct class $C(p_i)$. The training examples are presented separately and sequentially in a supervised batch learning fashion. In Figure 4 the root shows the unary feature space U and below it the binary feature spaces UB resulting from the unresolved clusters, the bottom row contains the unary feature spaces UBU computed from the non-unique clusters of the UB level. In Figure 4 large rectangles represent feature spaces, e.g. u_1, b_2, dashed lines determine split boundaries, and hatched clusters denote uniqueness.

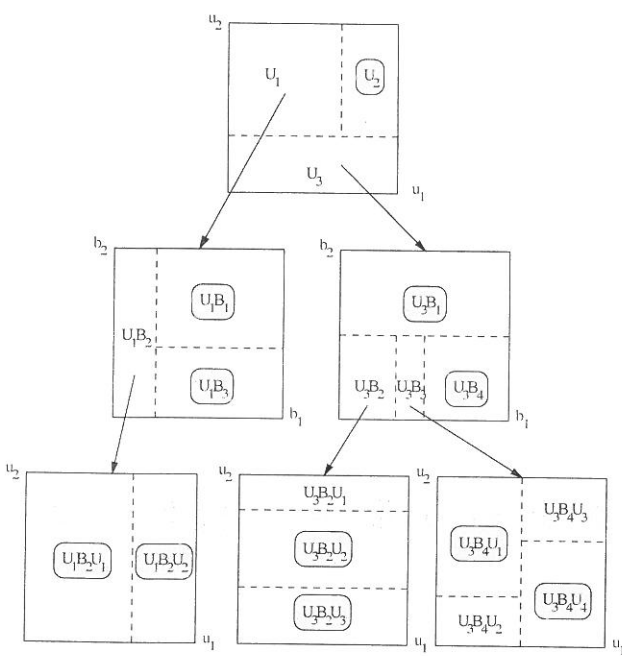

Fig. 4. An example of a cluster tree.

The CRG method works by first classifying all unary features from all parts, of all views and all object classes into a unary feature space U. This feature space is then clustered into a number of clusters U_n, some of which may already be unique with respect to the class membership, i.e. all parts in a cluster belong to the same class, $\exists_c \forall_{p \in U_n} C(p) = c$, while others do not. Binary features are then calculated between the parts of the non-unique clusters and its neighbors in the graph. The binary feature spaces resulting from the unresolved clusters are clustered again, each forming feature spaces of type $U_n B_m$. For the non-unique clusters the unary feature spaces $U_n B_m U_o$ of the parts of the previous binary relations are determined and clustered. This continues until all clusters are unique or a predetermined maximum depth has been reached.

Once the graph model of the object to be recognized has been constructed it must be "matched" against those in the database to determine the class in which it should be classified. One method of matching graph based models would be to determine the relational distance metric between the models. Even with new algorithms [CYS+96] developed for parallel computers, unlabeled attributed graph matching is a computationally expensive process, with the general problem being NP complete. Instead of attempting to match the object graphs against the model graphs, i.e. finding subgraph isomorphisms, short part paths through the graph are matched. An evidence accumulation technique is used where a large number (dependent upon the object but typically in the range of $50-100$) of small (typically 3 to 5 nodes in length) paths is extracted starting from each node in the object's graph representation and then matched against the generic model.

In *recognition* the goal is to correctly classify, using the tree, the previously unseen objects with possibly occluded and missing parts. First a PAG graph similar to that used in the learning phase is constructed. In this graph all non-cyclic paths up to a maximum length of the depth of the decision tree are generated and then classified using the decision tree. For each path an evidence vector with the probability that this path belongs to a particular class is computed. During learning this evidence vector is computed using the relative class frequencies for this branch of the decision tree. The evidence

vectors of all paths starting at some part p_i determine its classification.

Part Paths – *Where to look next?*

The problem of label compatibility arises when an object has the same parts and features as a different object. When only unary features are used the problem is more readily apparent as many objects may have the same parts and unary features, for example objects consisting of different arrangements of similar circles and squares. The addition of relations between parts, that is binary features, adds structural context. In recognition without labeled parts one must exhaustively attempt all matchings between unary and binary relations. If the relations are encoded as in a PAG, then it becomes a problem of subgraph isomorphism between unlabeled graphs. It is possible to avoid this prohibitively expensive matching problem by encoding the binary relations between the parts during learning in such a way that *only* label compatible part paths can be generated during both the learning and recognition stages.

Generating only label compatible part paths can be done by calculating the binary features only between *neighboring* parts of the same object while building the tree. The tree constructed in this way contains only paths from the root, where all parts are represented in some cluster, to the final classification leaves which correspond to the paths in the object from a part to its neighbor. The use of the neighboring relation as a constraint during both the training and recognition stages implicitly solves the label compatibility problem by assuring that any matched sequence must have arisen from a label compatible sequence. This solution is not sufficient since representative and therefore important sequences for recognition occur among *non-neighboring* parts scattered across the image. It is necessary to have a relation which still provides the constraints for solving the label compatibility problem and allows the non-neighboring representative parts to be combined into the same part paths.

The number of paths which can be considered in a view is $\binom{n}{l}$, where n is the number of parts in the view and l is the length of the path. For any given length, l, it is desirable to choose a subset $\mathcal{Q}_l$ of all n choose l paths, which is minimal but representative of the object. Without *a priori* knowledge this is not possible, so we must select a function which produces a set with cardinality somewhere between that of the ideal set and $\binom{n}{l}$.

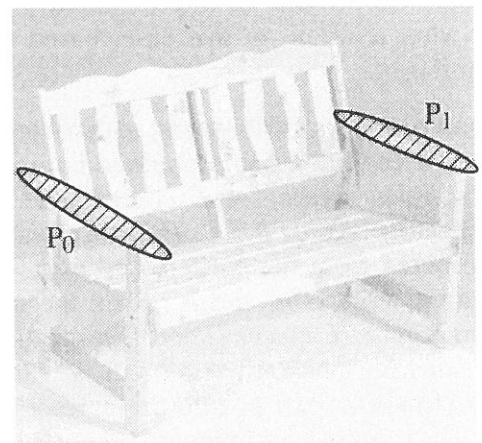

Fig. 5. Two significant parts, p_0 and p_1, which will not occur together in a path extracted from a PAG.

The neighborhood relation serves as one function to effectively limit the cardinality of $\mathcal{Q}_l$, but it does not achieve the other goal of selecting sufficent representative paths. If representative parts are separated by more than l neighbors then they will never be considered in a part path when using the neighborhood relation. As an example illustrating the need for the PCG, figure 5 shows that the marked parts, i.e. arm rests, are significant for identifying the class of bench, but are never considered when using a neighborhood function with a reasonable value of l. The problem still remains in how to formulate a function which produces a path set $\mathcal{Q}_l$ of reasonable cardinality and representative paths without *a priori* knowledge. We develop this function as a consequence of our algorithm for constructing the attributed PCG below.

The Part Compatibility Graph

In a PAG graph G an edge $e_{i,j}$ is constructed between the parts p_i and p_j if the parts are neighbors. Neighborhoods are defined through either physical adjacency or, more generally, spatial proximity, e.g., parts within 10 pixels from each

other. Other neighborhood definitions, for example Area Voronoi neighborhoods, are possible but all suffer from the drawback that simply being a neighbor of another part is not necessarily a significant relation for recognition. In a PCG graph G, however, a graph edge $e_{i,j}$ denotes a high *similarity measure* between parts p_i and p_j, that is, the similarity measure replaces distance as used in the PAG.

To construct the PCG graph G, features whish consisti of a tuple containing the feature value, v, and a confidence value, c, $\langle v, c \rangle$ are computed for each part p_i. Initially a seed part p_s is selected based upon the high confidence value of its feature vector and the feature base is queried for other parts with similar values within a given radius of p_s, these parts are then termed the *selection set* of part p_s, $\mathcal{S}_{p_s}$. The similarity between p_s and each member of the set $\mathcal{S}_{p_s}$ is then calculated and those parts having high similarity values are inserted into the *conditional expansion set* of part p_s, $\mathcal{E}_{p_s}$. This process is repeated recursively for each member of the expansion set $\mathcal{E}_{p_s}$ until a preselected depth level is reached. At this point paths are pruned from the tree T_{p_s} rooted at part p_s and progressing through its expansion sets so that only those with the highest similarity values remain. The pruned tree is computed for each node of the PCG graph and the resulting forest of trees is combined in a straightforward manner to create the PCG graph.

The similarity measure used depends upon the nature and type of features used. In our examples we have used moment based features of regions with a simple similarity measure. In Figure 6 a PAG and two PCG with different similarity functions can be examined. The cardinality of the edge set $\mathcal{E}$ of the PCG graph is approximately $1/3$ greater than that of a region adjacency graph. This increase in size is significant, growing from an approximately $1:2$ ratio of nodes to edges in the region adjacency graph to $1:3$ in the PCG graph.

Polymorphic Primitives in the CRG Framework

The CRG approach is based on a decision tree, in which decisions are based alternately on unary and binary relations along each path originating from the root of the tree. These relations between parts are pre-calculated for the object to be recognized. Assume we have generated a non-cyclic part path $S = \langle p_1, p_2, \ldots, p_k \rangle$, where all the parts are from the same class. In order to classify S, a sequence of decisions:

$$\langle U_1(p_1) \rightarrow B_1(p_1, p_2) \rightarrow U_2(p_2) \rightarrow B_2(p_2, p_3) \rightarrow \ldots \rightarrow B_{k-1}(p_{k-1}, p_k) \rangle$$

is performed. Since all the parts are of the same class, the attributes and thus the decision domains are the same for all unary and binary relations, respectively:

$$\begin{aligned} \mathcal{D}(U_1) &= \mathcal{D}(U_2) = \ldots = \mathcal{D}(U_{k-1}) \\ \mathcal{D}(B_2) &= \mathcal{D}(B_3) = \ldots = \mathcal{D}(B_{k-1}) \end{aligned}$$

At each decision node v_i the corresponding domain $\mathcal{D}(U_i)$ or $\mathcal{D}(B_i)$ and each cluster is assigned to one decision branch leaving node v_i. We now consider the polymorphic case, where $Class(p_i) \neq Class(p_j)$ in general, and there are no constraints on the sequence of part classes.

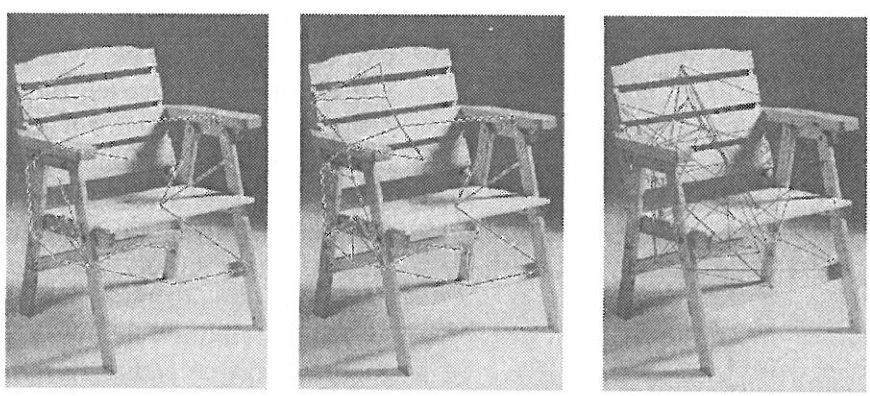

Fig. 6. An example of PAG, followed by two PCG with different similarity functions.

Each primitive class c_j $(j = 1 \ldots N)$ is characterized by an ordered set of unary features

$$U_j = (u_{j,1}, u_{j,2}, \ldots, u_{j,M_j})$$

called its unary feature vector. For each pair of primitive classes (c_j, c_k) we define a binary feature vector

$$B_{jk} = B_{kj} = (b_{jk,1}, b_{jk,2}, \ldots, b_{jk,M_j})$$

and a corresponding mapping function $f_{jk} : U_j \times E_k \mapsto B_{jk}$. We also define a generic mapping $f()$, such that $f(p_a, p_b) = f_{A,B}(U_a, U_b)$, where $a = Class(p_a)$ and $b = Class(p_b)$, respectively. The mapping f is not symmetric to preserve the directedness of the part path, i.e., $f(p_a, p_b) \neq f(p_b, p_a)$ in general.

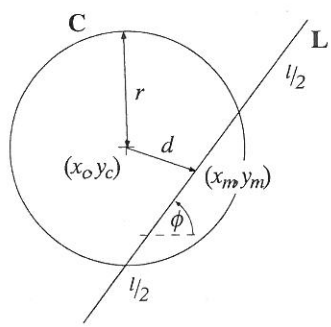

Fig. 7. Parameterization of lines and circles to describe binary relations.

For example, given a circle **C** and a straight line segment **L** the elements of the binary relation vectors $B_{Circle,Line} = f(\mathbf{C}, \mathbf{L})$ and $B_{Line,Circle} = f(\mathbf{L}, \mathbf{C})$ are different (although their structure may be the same). Given unary feature vectors of lines and circles (Figure 7),

$$\begin{aligned} U_{Line} &= (x_m, y_m, l, \phi) \\ U_{Circle} &= (x_c, y_c, r)\,, \end{aligned}$$

then a simple mapping functions for the binary feature vectors are:

$$\begin{aligned} f_{Line,Circle}(\mathbf{L}, \mathbf{C}) &= (d_x, d_y, \tfrac{d}{l}, \tfrac{r}{l}) \\ f_{Circle,Line}(\mathbf{C}, \mathbf{L}) &= (-d_x, -d_y, \tfrac{d}{r}, \tfrac{l}{r}) \end{aligned}$$

with:

$$\begin{aligned} d_x &= x_m(\mathbf{L}) - x_c(\mathbf{C} \\ d_y &= y_m(\mathbf{L}) - y_c(\mathbf{C}) \\ d &= \sqrt{d_x^2 + d_y^2} \\ l &= l(\mathbf{L}) > 0 \\ r &= r(\mathbf{C}) > 0. \end{aligned}$$

Since one of the involved parts (the circle) has no orientation, the angle of the line (ϕ) is not used in the binary feature vector.

At each level of the CRG decision tree, clusters in the unary or binary attribute space are associated with tree branches. At the root node of the tree, the initial decision is based on the unary attributes of all parts. When different classes of parts are involved, we have no homogeneous attribute space that can be clustered in the conventional way, but the class attributes of the parts provide for a natural partitioning.

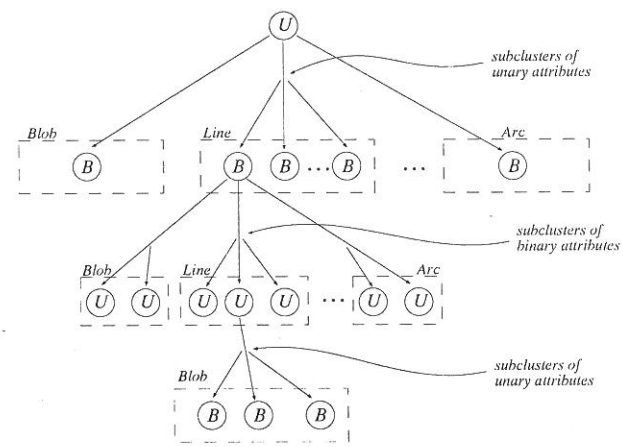

Fig. 8. Sample classification tree for polymorphic part paths.

The basic decision structure we use is similar to the CRG tree with decisions alternating between unary (U) and binary (B) attributes. At each level, *base clusters* are formed implicitly by classes of the parts involved. Classification of a part path $\langle p_1, p_2, \ldots, p_k \rangle$ is initiated by examining the unary attributes of the first part p_1, which can be of any class $c_1, \ldots, c_N$. For example (Figure 8), the part classes *Blob*, *Line*, *Arc*, etc., can form the base clusters of a decision tree. Within each of the base clusters, sub-clusters can develop during the training phase. For this we can use any conventional clustering technique, because we operate in a homogeneous attribute space. Decision nodes for *binary* relations are formed in much the same way. When we need to classify a link from one part to the next $(p_i \rightarrow p_{i+1})$, the class membership of the first part, $c_i = Class(p_i)$, is known,

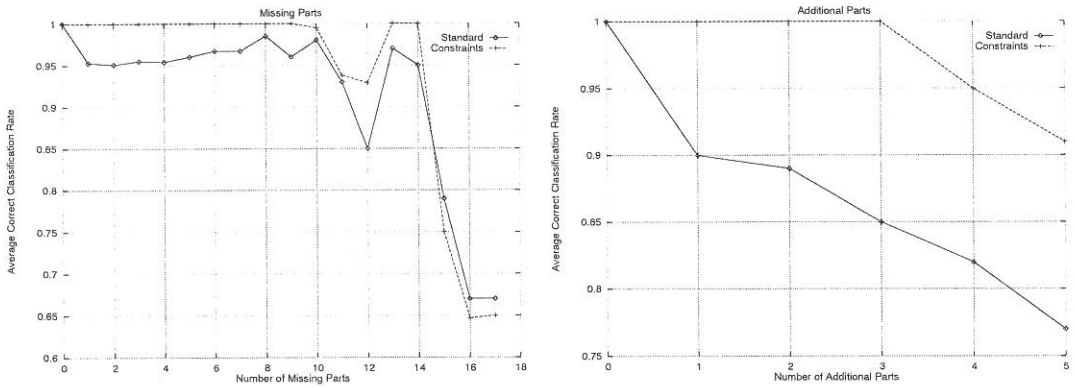

Fig. 9. An Example of behavior with additional and missing parts.

while p_{i+1} can be any class. Thus, for each (observed) pair of classes (c_i, c_j), $j = 1, \ldots, N$, a binary decision branch is inserted, each representing a base cluster for a specific binary relation B_{ij}. During the learning phase, additional discrimination rules are introduced by creating sub-clusters within the (homogeneous) domain B_{ij}.

Experiments and Results

A small database of images showing several instances of generic classes in typical poses, was used in an experiment to correctly recognize previously unseen instances and to reject unknown objects. The set exhibits real-world imaging problems such as shadowing and self occlusion, and the content, e.g. chairs, benches, and tables, was purposely selected to provide for a high degree of structural similarity between classes.

The systems performance with missing and additional parts was examined. The experiment was carried out using a previously learned object and parts were randomly selected (Figure 9). In each case, two lines are shown, one using the results from the system without any preprocessing and the other using constraint rules for inter- and intra-path compatibility[BBM96]. The system performs well, even when there are considerable missing parts, this is due to the way the part paths are selected and the PCG. Performance drops when additional parts occurs. The performance with occluded parts is a combination of missing and additional parts, as an occluded part generally causes a missing part and a number of additional parts to occur in the image.

The chart shown in Figure 10 compares the original PAG neighborhood definition and the proposed PCG method. In order to better analyze the contribution of the new part path selection method, deliberately weakened features were used for the comparison. Nine objects, three

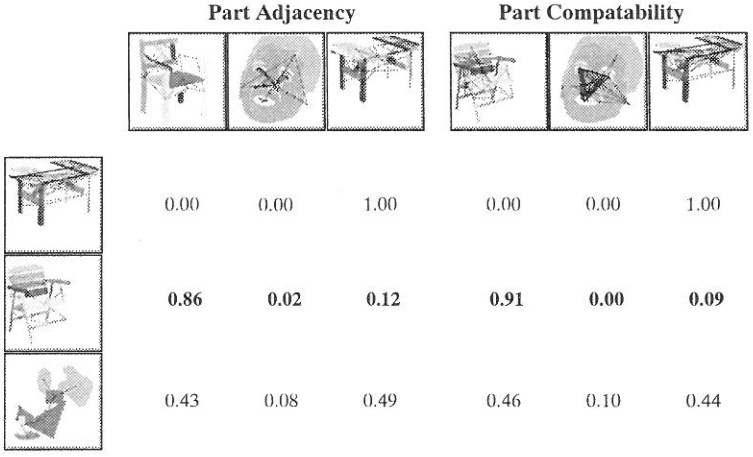

Fig. 10. Recognition results.

from each class, were used for training and a previously unseen object of type chair was presented for recognition. The first row of Figure 10 gives the results for recognizing a previously learned object, both the PAG and the PCG classify it correctly with 100 percent confidence. The last row of the table shows the classification result for a type of object which does not match any previously learned class, here both methods are uncertain how to classify the object. The algorithm works as expected in these two cases, identifying a previously learned object and rejecting an object which is unlike any one previously learned. The middle row shows the reaction of the system to a previously unseen object of the class which it has learned. Using the PAG the system is able to classify it correctly with 86 percent confidence, and using the new PCG with 91 percent confidence.

The main reason for the performance improvement is that the part paths extracted by the proposed PCG method extend over larger regions of the object and are therefore more likely to contain non-neighboring significant parts.

Implemention of the System

Different components of the system were developed using a number of different class libraries and languages including C, C++, Lisp, TK-TCL, Expect. We would have liked to develop the entire system within a single platform, but our evaluations of the large vision platforms like Khoros an KBVision found them wanting. The new generation of platforms, such as Alphelion and the IUE and its preliminary TargetJr are much more attractive. We have begun to port our system to TargetJr and as the IUE matures we will incorportate it into it. In figures 11 and 12 examples of the various modules are given.

Conclusion

A new PCG based method for creating part paths in the CRG method for structural object recognition was presented and evaluated. Initial experiments show it to be promising with an slight improvement in recognition performance for a small increase (from 1:2 to 1:3) in the node/edge ratio of the graph representation. A formalism for combining evidence from polymorphic feature sets was elaborated. We are currently working on a new learning methodology allowing for incremental learning from a small number of examples to replace the current batch learning. In addition a much larger model base is being compiled.

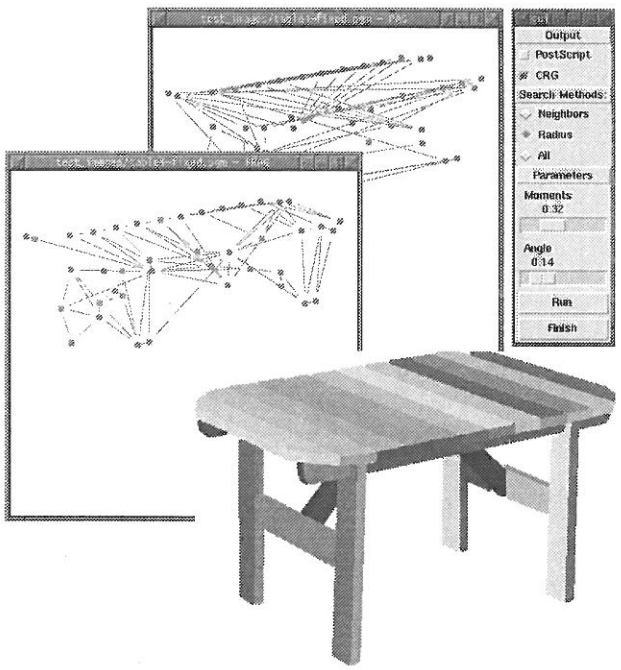

Fig. 11. The system as implemented using: a TK-TCL GUI, Expect process control. The actual learning, classification, and feature extraction code was written in C++.

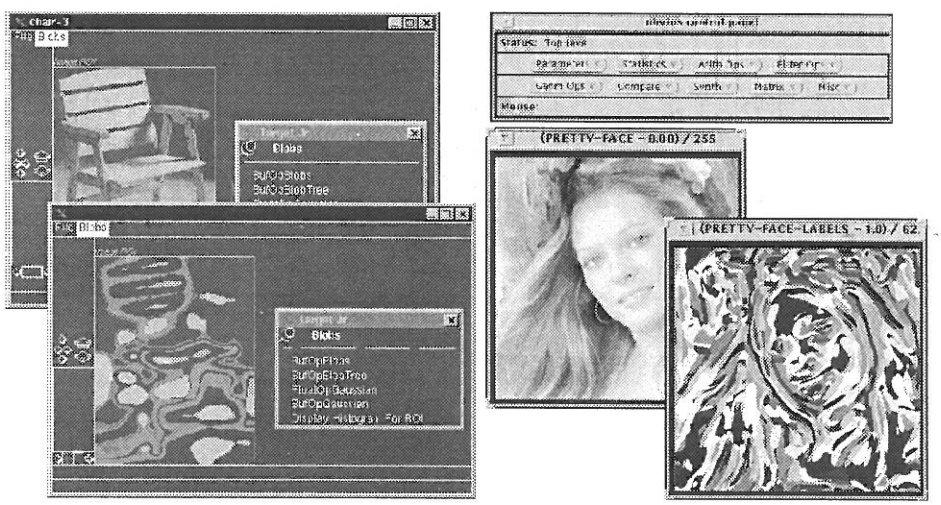

Fig. 12. (a) The blobs system uses Interviews for GUI and the class library TargetJr, the code is written in C++. (b) The local structure segmentation as implemented using: a LispView GUI, Obvious process control. The actual segmentation code was written in Common Lisp.

Reference	2-D 3-D	# of Objects	Scale Invar.	Rotat. Invar.	Occlu-sion	Image Features	Representation & Matching	Training
Ansari, Delp 90	2-D aff.	1	yes	yes	yes	ordered sequence of interest points ('landmarks') along the object contour; 'sphericity' used as a local shape measure	'hopping' dynamic programming and least-squares fitting	
Ayache, Faugeras 86	2-D	few	yes	yes	yes	polygonal approximation of boundary	HYPER, recursive estimation of model transformation	
Baird 88	2-D	many	no	no	yes	stroke, hole, arc, crossing, endpoint.	parameter-space clusters, approximated by square neighborhoods (OCR application)	by clustering
Bhanu, Faugeras 84	2-D	few	yes	yes	yes	ordered sequences of straight line segments	polygonal approximation of object boundaries; hierarchical and probabilistic relaxation labeling	
Bhanu, Ming 87	2-D 3-D	1	some	yes	yes	ordered sequences of straight line segments	polygonal approximation of object boundaries; sequence clustering and maximum-distance selection	
Bolles, Cain 82	2-D	1	no	yes	yes	corners, holes, and boundary lists	local-feature-focus method: redundant object description for each focus feature; clustering of mutually compatible tokens using maximal clique algorithm	yes
Burt 88	2-D	1	no	no	yes	hierarchy of iconic templates at multiple resolutions, arranged as a pattern tree	template matching at multiple resolutions	interactive
Chaudhuri et al. 90	2-D	few	yes	yes	yes	polygonal approximation of boundary	matching by heuristic search with segment merging	
Ettinger 88	2-D	many	yes	yes	prob-ably	local shape of boundary segments: corners, joins, inflections; compound features: ends, cranks, bumps. features are organized in two hierarchies: the structure hierarchy and the scale hierarchy.	initial hypotheses are made at the subpart level. recognition and verification uses the scale hierarchy of the subpart features. hierarchical organization of subpart library lets effort grow sub-linearly with library size.	representation hierarchies are built automatically
Fischler, Elschlager 73	2-D	1	no	no		iconic picture segments	binary elastic relations, spring templates; linear sequential embedding algorithm, dynamic programming	
Gorman et al. 88	2-D	1	no	yes	yes	contour segments encoded with normalized Fourier descriptors	dynamic programming, relations to string matching and dynamic time warping	
Griffin, Deuermeyer 90	2-D	few	no	yes	yes	point patterns of objects with movable subparts	objects are represented by a graph structure of subparts (described by point patterns).	
Han, Jang 90	2-D	few	no	yes	yes	boundary segments, points of local maximum curvature arranged in a graph	max. clique, edges between image points indicate a high likelihood that they belong to the same object	
Hernandez et al. 89	2-D	few	no	yes	yes	curvature of contour segments is evaluated to extract straight and curved segments. scene primitives are assigned a set of possible labels and corresponding certainty factors.	iterative, heuristic graph matching (hill climbing)	
Hwang 89	2-D	1	yes	yes	yes	oriented linear edge segments (not only boundary), corners (from line segments).	Image corners are matched to model corners and used to create the initial matching hypotheses (pos., orientation, scale). Some matches can be rejected from local intensity characteristics.	yes (for single model)
Kalvin et al. 86	2-D	many	no	yes	yes	Fourier-descriptors of local boundary segments (footprints)	hashing	-
Knoll, Jain 86	2-D	1	no	yes	yes	boundary segment prototypes (60 pixels long)	Set of significant features for each object. Exhaustive cross matching between image and all model features.	Automatic selection of significant model features.
Kundu et al. 89	2-D	50	ltd	no	no	moments, loops, T-joints, X-joints, semi-circles	handwritten word recognition on letter basis. vector quantization of letters (90 optimal symbols from 2500 sample letters). 1st and 2nd order HMM for word recognition.	yes
Neveu, Dyer, Chin 86	2-D	1		yes	yes	boundary segments at multiple resolution, obtained through a Laplacian pyramid	generalized Hough transform to match subshapes + graph matching	interactive
Segen 89	2-D	1+	no	yes	yes	local curvature extrema of boundaries, structures are grouped to form more complex structures	H-graph matching	yes
Shepherd et al. 92	2-D	few	yes	yes	yes	curved boundary segments	geometrical congruency graph matching	yes
Stein, Medioni 90	2-D 3-D	many		yes	yes	several polygonal boundary approximations with different line-fitting tolerances	Gray-coding of super-segments and hashing for indexing.	
Tsai, Yu 85	2-D	1	no	yes	no	shape boundary primitives, segments between local max. curvature points	attributed string matching with merging	
Wallace 87	2-D	few		yes	yes	straight lines, arcs + spatial relations between pairs of them: bilink, parallel, concentric, connect	heuristically guided tree search	

Tab. 1. Comparision between different structural approaches

References

[BB94] W. BURGER AND B. BHANU, Signal-to-Symbol Conversion for Structural Object Recognition Using Hidden Markov Models, In *Proc. ARPA Image Understanding Workshop*, pages 1287–1291, Monterey, CA, Nov. 1994.

[BBM96] M. BURGE, W. BURGER AND W. MAYR, Generic Object Recognition Using Weak Structural Representations, In *3rd Slovenian Speech and Image Understanding Workshop*, Ljubljana, Slovenia, April 24–26 1996, SDRV, to appear.

[BC94] W. BISCHOF AND T. CAELLI, Learning structural descriptions: a new technique for conditional clustering and rule generation, *Pattern Recognition*, pages 689–697, May 1994.

[Bie87] I. BIEDERMANN, Recognition-by-Components: A Theory of Human Image Understanding, *Psychological Review*, 94(2), 115–147, 1993.

[BL93] R. BERGEVIN AND M. D. LEVINE, Generic object recognition: building and matching coarse descriptions from line drawings, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1(15), 19–36, 1993.

[Bro81] R. A. BROOKS, Symbolic Reasoning Among 3-D Models and 2-D Images, *Artificial Intelligence*, 17, 285–348, 1981.

[Bur88] P. J. BURT, Smart Sensing within a Pyramid Vision Machine, *Proceedings of the IEEE* pages 1006–1015, 1988.

[CYS+96] CINQUE, L., YASUDA, D., SHAPIRO, L. G., TANIMOTO, S. AND ALLEN, B., An Improved Algorithm for Relational Distance Graph Matching, *PR*, 29(2), 349–359, February 1996.

[Ede93] S. EDELMAN, On Learning to Recognize 3-D Objects from Examples, *PAMI*, 15(8), 833–837, August 1993.

[FF95] C. FUCHS AND W. FÖRSTNER, Polymorphic Grouping for Image Segmentation, In *Proc. ICCV95*, pages 175–182, 1995.

[Fu92] K.-S. FU, *Syntactic Pattern recognition and Applications*, Englewood Cliffs, 1992.

[JH88] A. K. JAIN AND R. HOFFMAN, Evidence-Based Recognition of 3D Objects, *PAMI*, 10(6), 783–802, November 1988.

[LBP95] T. LEUNG AND M. BURL AND P. PERONA, Finding Faces in Cluttered Scenes Using Labelled Random Graph Matching, In *Proc. ICCV95*, pages 637–644, 1995.

[Lin93] T. LINDEBERG, Detecting Salient Blob-Like Image Structures and Their Scales with a Scale-Space Primal Sketch: A Method for Focus-of-Attention, *IJCV*, 11(3), 283–318, 1993.

[Low87] DAVID G. LOWE, Three-dimensional Object Recognition from Single Two-dimensional Images, *Artificial Intelligence*, 31, 355-395, 1987.

[MN95] H. MURASE AND S. K. NAYAR, Visual Learning And Recognition Of 3-D Objects From Appearance, *IJCV*, 14(1), 5–24, January 1995.

Contact address:
Mark Burge, Wilhelm Burger, Wolfgang Mayr
Johannes Kepler University, Dept. of Systems Science
Computer Vision Laboratory
Altenberger Straße 69
A-4040 Linz, Austria
burge@cast.uni-linz.ac.at
Tel. ++43-732-2468/9824, Fax. /893

MARK BURGE received the BSc. degree from Ohio Wesleyan University in 1990 and the MSc. degree from The Ohio State University in 1993, both in computer science. In 1993 he joined the computer science department of The Swiss Federal Institute of Technology (ETH) in Zürich and later the computer vision group within the Dept. of Systems Science at Johannes Kepler University.

WILHELM BURGER received the MSc. degree from the University of Utah in 1985 and the PhD. from Johannes Kepler University in Linz, Austria, in 1992. He is the head of the computer vision group within the Dept. of Systems Science at Johannes Kepler University.

WOLFGANG MAYR received the degree of a Dipl.-Ing. from the Johannes Kepler University in Linz, Austria, in 1995. After that he joined the vision group within the Dept. of Systems Science at Johannes Kepler University.