A Pattern Recognition System Driven By Discriminability

Mayez Al-Mouhamed *

Abstract

This paper presents the design of a model-based vision recognition engine for planar contours that are scale invariant of known models. Features are obtained by using a constant-curvature criterion and used to carry out efficient coarse-to-fine recognition. A robust shape matching is proposed for comparing contour fragments out of scenes with partial occluding. Model organization and indexing schemes are optimized in a manner that reduces the dependency of the recognition time over the model size. To carry out early pruning of large portion of the models, hypotheses are only generated for a sub-set of contours with enough discriminative information. Poor scene contours are used latter in validating or invalidating a relatively small set of hypotheses. Recognition takes advantage of robust hypotheses by categorizing the contour intersection points so that hypothesis processing is driven by discriminability. Since hypotheses are selectively verified, blocking is avoided by extending current matching by pairing of hypotheses, predictive matching, and fetching next weighted hypotheses. This has the advantage of avoiding brute force processing of a large number of hypotheses. Storage is optimized by experimentally relating the bucket size to a metric of discriminability for typical patterns. Evaluation shows that the recognition time is nearly independent from the number of hypotheses originally generated. The time increasing due to increase in the database size is relatively small as a result of selective processing and optimization of global database effects.

Keywords: Database, indexed search, matching, pattern recognition, segmentation

1 Introduction

An effective model-based recognition system [1, 2, 3] must be capable of retrieving the best matched objects as well as carrying out massive pruning of inconsistent models. Modeling objects by their local geometric features [4] takes advantage of the coarse shape and enables quick indexing of object features into the models in an attempt to reduce the complexity of the search space before carrying out finer pattern matching. Neural

*Computer Engineering Department, College of Computer Science and Engineering, King Fahd University of Petroleum and Minerals, Box 787, Dhahran 31261, Saudi Arabia. mayez@ccse.kfupm.edu.sa

networks were also used for finding the final interpretation by using the features as inputs to a *neural network* [5] that is trained to classify objects based on subset of features. Hierarchical object modeling partitions the object contour in a collection of fragments so that each fragment is a set of features which are selected as invariant under translation and rotation [6, 7].

The efficiency of the matching depends to a large extent on the scalability [1, 8, 9] of the recognition operator which is the ability to recognize whole contours as well as fragments of contours while reducing the combinatorics of the search. For this, the extracted features [10, 11] must be local and small enough to match wherever they are present but must also be stable and discriminative. Global features are inadequate when contours are partially observed. Methods for partitioning and representing contours can be found in many research papers such as [4, 12] where in most cases the features are used as searching keys in some quick indexing/hashing schemes. To find stable partitioning criterion, curved contours suggest the use of sharp convexities, deep concavities, or straight segments as reference points [2, 11, 13] in finding the boundary between different geometric features. Curvature maxima has also been used as segmentation points and straight lines as primitives in [14]. However, the above approach cannot be applied to regularly curved shapes because of boundary ambiguity. Decomposing smoothed contours at extreme of negative curvature has been also investigated in [15]. Detection of significant changes in curvature [16] has been applied for encoding geometrical signatures like smooth join, corner crank, etc. Decomposing contours by using constant curvature criterion was proposed by Wuescher et al. [17] and used in other proposals. This provides means for partitioning contours of planar objects by using particular features that can be linked together for building the model.

Model organization was studied by Califano and Mohan [9] which proposed the use of larger indices (multidimensional indexing) to keep a relatively coarse bucket quantization without scarifying selectivity. To gain selectivity, intuitive reduction of the bucket size may cause loss in overall discriminability. For this efficient object modeling requires establishing some theoretical and experimental relationships that tells how discriminability is affected when adjusting the bucket size.

The synergy of the indexing scheme must be small enough because all the models are potentially involved in the initial search [4, 12, 18, 19]. Kalving et al. [20] used a hashing descriptor that is derived from the relationships between lengths and relative orientation of contour segments. Knoll and Jain [21] proposed a model organization based on common features so that to index into the model by recognizing features and further iterate to narrow the object class down to the correct interpretation. Turney, Mudge, and Voltz [18] used a model organization based on identification of salient model features which are use as keys to index into the model. Grimson [13, 22] equally treats all the available features in generating hypotheses on possible matches. This results in tree-matching structure that is scanned by using depth-first search. The search over the current sub-tree is abandoned when enough inconsistent evidences are accumulated and the next sub-tree is started. Though this organization allows pruning many inconsistent sub-tree interpretations, the number of visited sub-trees is large even for simple scenes.

Our objective is to optimize the model and the search so that the recognition time

would mainly depend on the scene complexity without explicit dependence on the database size. For this, the generation of hypotheses follows a different approach compared to previously proposed approaches. An approach to the generation of robust hypotheses is proposed for efficiently reducing the dependency of the recognition time over the size of the model. In other term, the model and the algorithm are to be designed so that the recognition spends a small fraction of time in global database processing while keeping the rate of correct classification as high as possible. Our aim is to avoid early processing of contours having poor discriminative information. The early generation of a large number of hypotheses would necessarily increase dependency over the size of the models. Such poor contours will be used in latter stages in validating or invalidating a relatively small set of robust hypotheses.

To keep on pruning of inconsistent hypotheses we propose an efficient shape matching for comparing whole contours as well as fragments. Our approach is intended to avoid brute force processing of a large number of hypotheses that must be processed regardless of discriminability. For this we propose a processing scheme that is driven by discriminability in which blocking is avoided by extending the matched contours through predictive matching and pairing of hypotheses. Reducing the storage size without affecting the discrimination power is carried out by relating the bucket size to a metric of discriminability.

This paper is organized as follows. Section 2 presents the low-level contour modeling and the generation of detailed and coarse object descriptions. Section 3 defines the features and their extraction from the model. Section 4 presents the database organization and the associated indexing. Section 5 presents a demand-driven recognition algorithm and its associated tools such as shape matching, pairing of hypotheses, and predictive matching. In Section 6 we evaluate the proposed scheme with respect to storage requirements, effect of database size, and compare to other approaches. In Section 7 we conclude about this work.

2 Object modeling

Our approach uses the well known coarse-to-fine matching concept. For this we propose modeling objects by using a coarse and a fine polar representations which are used for implementing our coarse-to-fine recognition system. The polar model is used because it provides scale, rotation, and translation invariant means. In this Section we study the salient aspects in the design of these models, their relationships, and their characteristics. describes the method used to obtain

The binary image is processed by using the gradient operator and contours are obtained by using the *direction coding* that links up the border pixels on each contour. Contours associated to partially occluded scenes produce multiple intersecting segments which requires that all directions are scanned and encoded as chains with reference to intersecting pixels. Segmentation of the chains produce a set of linked polygon-based contours which are represented by using the length-angle (polar) representation that is one choice to obtain geometrical features which are invariant with respect to translation and rotation.

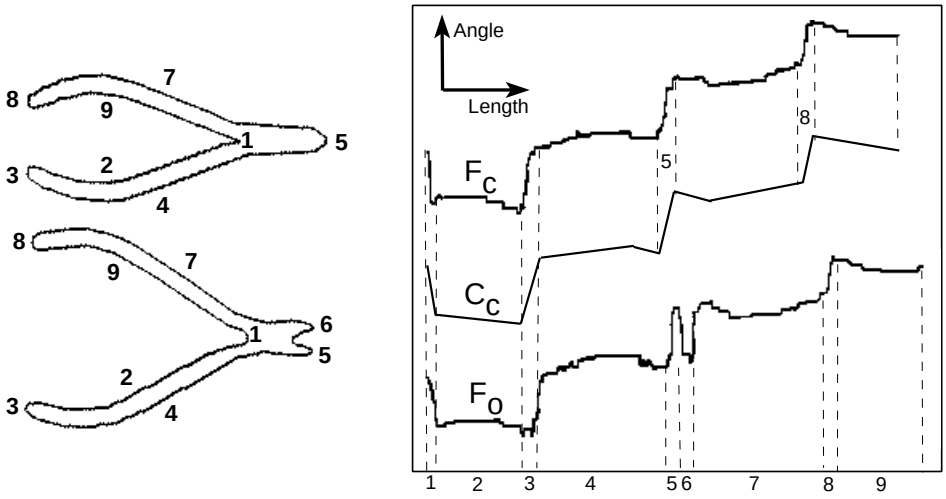

Figure 1: A cutter and its fine and coarse models

Every segment D_k is formed by a pair of points b_k and b_{k+1}, where a point b_k is defined by its coordinate b_k=(x_k,y_k). The length s_k of D_k is defined by $s_k = (\Delta x_k^2 + \Delta y_k^2)^{1/2}$, where $b_{k+1} - b_k = (\Delta x_k, \Delta y_k)$. The angle $\theta(s_k)$ between segments D_{k-1} and D_k is evaluated as the exterior angle which is defined by $\theta(s_k) = \cos^{-1}((\Delta x_{k-1}.\Delta x_k + \Delta y_{k-1}.\Delta y_k)/(s_{k-1}.s_k))$. The correct sign of $\theta(s_k)$ can be found by examining the coordinates of b_{k-1}, b_k, and b_{k+1}.

A fine angle-length model of contour is represented by means of an ordered set of segment lengths s_i and their geometric angles $\theta(s_i)$ that is $F = \{(\theta(s_i), s_i)\}$. Different positions and orientations of the contour shape leads to different segmentations but the shape characteristics of the corresponding length-angle plots are fundamentally preserved. Due to the effects of noise and digitization one can obtain some level of stability but overall effects of these variations depends on how this representation is used in the recognition process. Figure 1 shows the correspondence between the contours of a cutter (left part) and its fine polar models F_c and F_o which are associated to a closed and open cutter, respectively. The mapping from contours to the plots F_c and F_o are marked by numbers. The effect of noise and digitization on fragments (1, 2, 3, and 4) of the cutter are fundamentally identical in F_c and F_o by differ in their details. Corner 1 has sharp angular change on F_c but smoothly represented on F_o. Corner 5 in F_c is broken into two corners (5 and 6) in F_o. The total contour length of F_o is slightly longer than that of F_c due to the opening at 5 and 6 which indicates that global contours cannot be directly compared. Note that the sequence of fragments (2, 3, and 4) in F_c and F_o can be easily matched regardless of the opening. The same remark is also true for sequence (7, 8, and 9). Therefore a shape recognition scheme should only match contours that are not affected by the non-rigid shape of objects. The polar plot gives information on the relative orientation of the above sequences that depends on the opening angle. The remaining contours have variable-geometry and must be associated some lower and upper bounds based on their polar plots. The relative orientation between rigid sequences and the constraints on variable-geometry contours can then be used as complementary constraints to shape matching.

Changing the object orientation produces different fine models F and F' but one can be obtained from the other after a number of horizontal (length) shift operations over the ordered set $F = \{(\theta(s_i), s_i)\}$. The original angle or $\theta(0)$ is the reference angle of the first segment with respect to some fixed orientation such as the horizontal axis. When matching the polar models F and F', the effect of the original angle is that even with proper horizontal shifting the values of $\theta(s_i)$ still differ regardless of i The reason is that the effect of the starting segment angle produces a horizontal shift between models F and F'. The minimum area difference between two models F and F' provides a mean for finding how similar two object contours are. This provides a metric that is useful for shape matching. Arkin [23] et al. proposed a method for the efficient computation of the above metric for entirely observed objects. However, the use of this metric in a recognition system faces the problem of linearly searching all the models that any recognition system is to avoid. In Section 5.2 we extend the above results to fragment of contours and show how the minimum area difference can be computed regardless of the horizontal shift. We will show that computing the minimum area difference for fragment of contours requires horizontally shifting one of the polar models by some constant angle.

In the angle-length plan, long straight segment of contours are associated horizontal straight segments regardless of the orientation of the corresponding object. A regular circular shape is associated a sequence of small horizontal segments (stair) in the fine model that appear as a single segment with some constant slop depending on the curving factor of the shape. In the polar plan, a sequence of segments that corresponds to a constantly curved contour can then be associated one single segment with constant slop which is basis for the coarse model.

The coarse model C results from clustering the segments of the fine polar F as a method to obtain a sketch of the original object. This operation is based on grouping constantly curved segments into longer segments with constant curving and linking the resulting segments by a number of inflection points. For example, the fine polar model F_c of the cutter is associated a coarse model C_c having 10 segments as shown in Figure 1. In the polar plan, the non-horizontal segments of C_c represent constantly curved contours and horizontal segments correspond to straight contours.

A fragment of contour that is constantly curved is represented in the fine model by a sequence of small segments $\{\theta(s_i), s_i\}$. Segmenting of the fine polar model consists of merging of these small segments into one single long segment which can be achieved when the signed ratios $(\theta(s_i) - \theta(s_{i-1}))/s_i$ are nearly constant along the sequence. Such a sequence is associated a curving factor $h = \sum \theta(s_i) / \sum s_i$, where $\theta(s_i)$ and s_i are the angle and length of the ith segment of the above sequence, respectively. The curving factor h is the ratio of the total angular change $(\sum \theta(s_i))$ that the fragment of contour undergoes over the curvilinear length of the contour $(\sum s_i)$. The sequence is maximal when its curving factor significantly changes with respect to any extension of it to the left, to the right, or both. A *coarse polar model* C can be defined as the collection of all the maximal sequences which correspond to some fine model F. Formally, $C = \{(\theta_b(s), \theta_e(s), s)\}$, where $\theta_b(s)$ and $\theta_t(s)$ are the angles at the beginning and end of the segment and s is the total length the constantly curved segment.

This is equivalent to breaking up long contours into a number of fragments with

constant curvatures from a finite set of curving factors. Straight contours are represented in C by means of horizontal segments. Contours with constant concavities or constant convexities are associated single segments with slops that represent their curving factors. Mainly, C_c is a polygonal approximation in the polar plan of some ideal polar model of the cutter as shown in Figure 1. One may observe that sequence of fragments $(2, 3, 4)$ can be affected by noise and digitization but its corresponding coarse segments are much more stable. The coarse model C_c provides a stable sketch of the object regardless of the cutter scale or the original position and orientation. The geometric relationships between the segments of C_c are invariant candidates and could then be used for recognizing objects under partial occluding or in the case of open contours.

The benefit of approximating the original contour by using its coarse polar representation (sketch) is to use its parameters as searching keys over the library models which is a critical ingredient to reduce the complexity of the recognition process. Conceptually, similar methods have been proposed in the literature based on some level of abstraction in modeling object contours by means of a set of local features. The latter are used in pruning inconsistent matches prior to carrying out fine matching over a small number of potentially matched objects. This approach uses the sketch of the object (coarse model) for extracting local shape features which will be used in gross-to-fine matching of entire contours, partially observed contours, and open contours. In the next section we present our method to build stable local shape features out of the coarse polar description.

3 Feature extraction

The objective of the recognition system is to map parameterized instances of objects into the model. Contour instances can be closed contours, fragment of contours, or intersecting contours which result from partial occluding in the presence of noise. Open contours may result from imperfect lighting, noise effects, object crossing the image boundary, or a combination of the above. Partial occluding refers to situations where two object or more overlap within the scene which lead to some intersection points. Under the above conditions the *global characteristics* of the contour are no more observed and some fragments of contour that are either open or intersecting become the only available information in the scene.

A recognition system is to exploit the *local geometric features* that the contour fragments carry on in order to classify these fragments and link up sub-set of segments in an attempt to find a complete scene interpretation. For this the features must be carefully chosen so that they can be locally observed and be stable enough to carry on some discriminative information. Features must be simple enough to be locally present and completely observed on relatively short contours. In the same time, they must also be coarse enough to discriminate models and be able to limit potential matching to a sub-set of the model where they can be present. Under these conditions the features can considerably contribute in reducing the combinatorics of the search and further matching can then operate on few objects that have similar features to those present in the scene.

Our objective is to use a coarse sketch of contour fragments from the scene in a primary search in order to avoid linearly searching across the models which would considerably

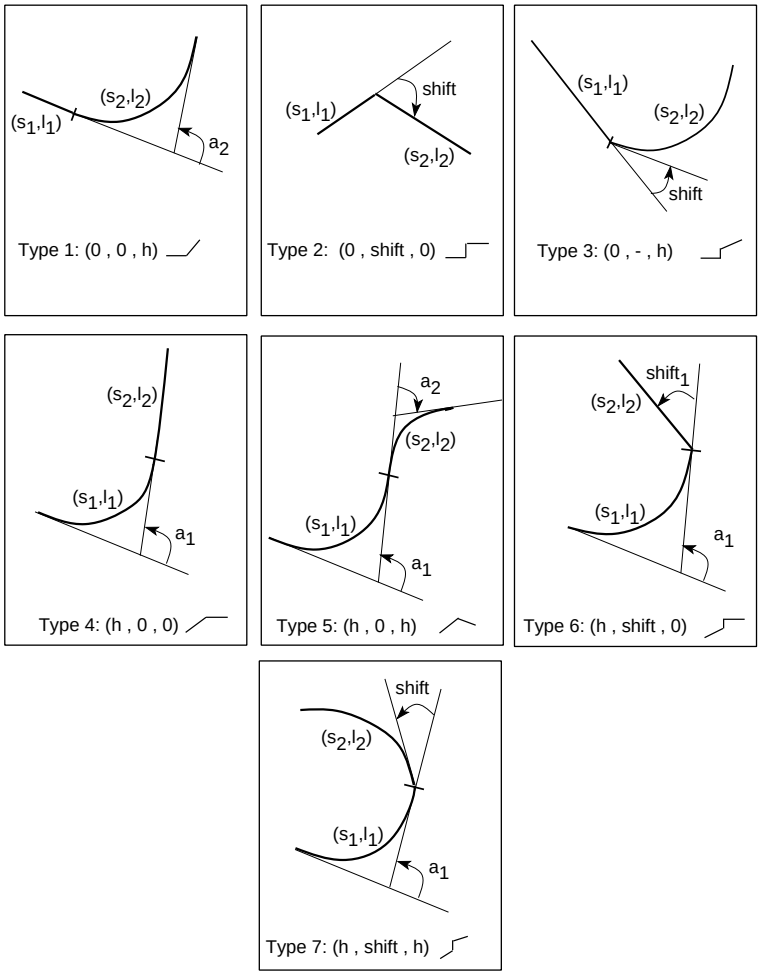

Figure 2: Features and their associated types from 1 to 7

slow down the recognition process. For this we need to define a set of features that should be extracted from the coarse representation and be used later as searching keys. Each feature f_x from the scene will be matched to some features f_m from the database model so that each match ($<f_x, f_m>$) generates a hypothesis stating that the object scene of f_x and the model object of f_m could be identical. Hypotheses need to be verified later for the geometrical relationships among each set of scene features that are matched to one common model which allows finding whether these hypotheses are consistent or not. In the other hand, identification of the matched model is also useful in finding the location of each occurrence of f in the model and therefore enables finer matching such as comparing the inter-relationships between scene features and model features.

The features should contain enough discriminatory information to order to provide efficient and accurate indexing of candidate models from the database. Too simple features may occur in many models which make the search inefficient because all database models will be hypothesized. Too complex features have two drawbacks: 1) cannot be observed from partial contours, and 2) lead to linear search across the database. We therefore need local features that contain enough discriminatory information.

Figure 2 shows seven possible configurations of two successive segments from the coarse model that are linked with each other. Each vertex v of the coarse model which links up two successive segments s_1 and s_2 can be associated a tuple $(h_1, shift, h_2)$ where h_1 and h_2 are the curving factors of the left and right segments s_1 and s_2 and $shift$ denotes the angle between the tangent to s_1 and the tangent to s_2 at the intersection point v. The tuple $(h_1, shift, h_2)$ is a simple feature of the coarse polar model because h_1 and h_2 are simply the curving rate of segments s_1 and s_2 and $shift$ is the angular shift between them. Each tuple $(h_1, shift, h_2)$ from the scene contour defines a *feature* which can be classified by its type (1 to 7) as shown in Figure 2. This Figure also shows the corresponding seg-shift-seg representation in the polar plan in front of each type. A feature discriminates the curving factors of connected segments as well as the actual angular shift between them. The feature encode geometric information in the neighborhood of *generalized inflection points* that link up the coarse model. Figure 2 shows the contour which corresponds to each type of features. Note that feature $(h_1, shift, h_2)$ is independent from the length of segment s_1 and s_2. This is useful because of two reasons: 1) the segment may not be entirely observed in the scene which poses a problem if indexing require full knowledge of lengths, and 2) the curving factors and the shift angle are fundamentally local information.

The proposed features are stable enough if we account for the digitization noise, the variance on the contour due to change in position and orientation, and other possible distortions. For example in Figure 1 the fine model F_c will be associated different sequences of small segments versus changes in position or orientation in the scene but the fundamental shape of F_c remains preserved. The coarse model C_c tolerates change in the detail of F_c and capture only the general shape of F_c so that change in C_c due to the above effects are generally small. Effect of distortion of parametrized features will be studied will be studied in the evaluation.

The use of these features will enable the implementation of some structural recognition scheme that is capable of propagating constraints in a coarse-to-fine fashion. The coaresness of the above features and their stability is one critical aspect of this approach especially during the early stages (coarse matching) of the recognition.

4 Database organization

The objective of building a database model of objects is to provide structural mapping of scene features into object models at the coarse level as well as carrying out finer verification and validation. There are seven distinct types of features and each type is associated one common indexing scheme that results from hashing the object models based on the value taken by each of their features. In order to carry on primary optimization of the storage, features are classified by type and each set of features that belong to a common type must share one single indexed storage. Indexed schemes allow scene features to be used as keys in searching candidate objects from the database for the generation of all consistent model matching. Each feature f with some type is associated a pointer value (f_v) that results from concatenation of non-zero values (by type) of its parameters h_1, $shift$, and h_2.

Indexing consists of a search procedure ($Inx\text{-}type(f_v)$) that takes a feature f with type ($type$) and generates all the model objects which contain at least one occurrence of f.

The degree of sharing within each hashing scheme $Inx\text{-}type(f_v)$ depends on the tolerance allocated to f_v which results from the variance on the values of parameters h_1, $shift$, and h_2. The variance result from the noise, digitization, and lighting condition which have different effects depending on the position and orientation of the object during the initialization phase.

To find the tolerance on the range of each parameter the object's contour is acquired several times with different positions and orientations and the feature parameters are evaluated for each setting which allows experimentally finding the tolerance for each parameter. Based on the allocated tolerance, each searched feature value f_v whose parameters falls within some range is considered to occur within each of the objects associated to that range. Formally, a feature f whose parameters h_1, $shift$, and h_2 fall within range $R = \{(h_l - \Delta h, h_l + \Delta h), (s - \Delta s, s + \Delta s), (h_r - \Delta h, h_r + \Delta h)\}$ is matched to all model objects that are associated with R, where h_l, s, h_r, Δh, and Δs are the curving factor of left segment, the shift angle, the curving of right segment, tolerance on curving, and tolerance on the shift angle, respectively. A model object that is associated with range R has at least one feature whose parameters fall within the range of R. Therefore, searching a matching between a scene feature f of some type consists of identifying a range of the model in which the parameters of f fall into. Tuning of the size (or bucket size) for each range is important to increase selectivity. Our approach to fine tuning of the bucket size will be presented in the evaluation because of its dependence on selectivity and overall discrimination power of the recognition system.

Indexing allows establishing a mapping from an input feature into a group of model objects that are associated to the corresponding range. Each model object associated to the range of a given type has at least one feature of that type whose parameters fall within that range. For example searching a matching for feature f consists of finding a cluster of objects so that $Inx\text{-}type(f_v) = \{O_k : f \in O_k\}$. To access finer information, each matched model object O_k has pointers to each occurrence of the matched features which identify the location of these features within its coarse model. The coarse model is represented by a set of ordered coarse segments and pointers are also used for each coarse segment to indicate the start and end of the set of fine segments associated to it within the fine model. The benefit of these pointers is that different scene features that are matched through indexing to some object can be further processed by checking their inter-relationships thus consolidating some interpretations and invalidating others.

This structured database organization has the advantage of searching the coarse shape (features) to derive rough matching and then use finer scene information to refine the interpretation over a small set of candidate models. The scheme avoids linearly searching the models because all the features do not have to be considered at once in early stage of the recognition. Pruning of incompatible interpretations is one important result because at the time the finer features are used large portions of the search space would have already been avoided through the indexed search.

5 A recognition strategy driven by discriminability

Grimson [13, 22] equally treats all the available features in generating hypotheses on possible matches which results in tree-matching structure that is scanned by using depth-first search. The search over the current sub-tree is abandoned when enough inconsistent evidences are accumulated and the next sub-tree is started. Efficient tree searching and updating transformations for consistency was first proposed by Faugeras and Hebert [24]. The inconsistency is detected when the verification of the inter-relationships between two features is negative with respect to some hypothesized model. Though this organization allows pruning many inconsistent sub-tree interpretations, the number of visited sub-trees is large even for simple scenes.

Our approach follows a different method which consists of initially selecting a subset of scene contours among those having the largest number of features among all scene contours. The reason for this is that this choice allows pruning large portion of the models and provide robust generation of hypotheses because it avoids handling contours with poor discriminative information. Relatively large number of hypotheses would originally be generated if all scene contours were hypothesized from the beginning. In other terms, poor contours are not processed in the early stages of our recognition approach but used latter in validating or invalidating a small set of hypotheses.

The initially generated hypotheses for the features are immediately refined with respect to contours that carry these features. The refinement consists of carrying out low cost *spatial matching* of the features of a given contour. Further refinement of the previously verified hypotheses consists of accurate *shape distance matching*. At this level, the retained hypotheses of fragments are only a very small fraction of the originally generated feature hypotheses. Clearly in our approach as we move further in the recognition the matching complexity increases but the problem size significantly decreases.

Now the task is to: 1) carry out pairwise matching of the retained hypotheses, and 2) extend current hypotheses to contours that have not been originally hypothesized. For the first case, the low cost spatial matching among hypothesized contours is involved only. For the second case, contours that have not been originally hypothesized are selectively verified for their membership of the reduced number of hypothesized models by first using the spatial matching and second the shape distance matching. The natural behind this method is to concentrate on rich contour shapes in generating the original interpretations rather than considering all contours and wasting precious time wondering about poor contour shapes and their vague hypotheses. In the following we present the detail of the proposed approach.

5.1 Generality and initialization

In our representation, a vertex is generally the intersection point of contours of two objects or more but can also be a simple end point of an open contour. An open contour may result from the effect of lighting or noise. A contour that links up a pair of vertices is called a *line*. The geometric shape of a line is arbitrary including curved shapes, polygonal shapes, or a mixing of them. At least three lines intersect at each vertex in the case of connected contours (non open). A continuous line that links up two vertices necessarily

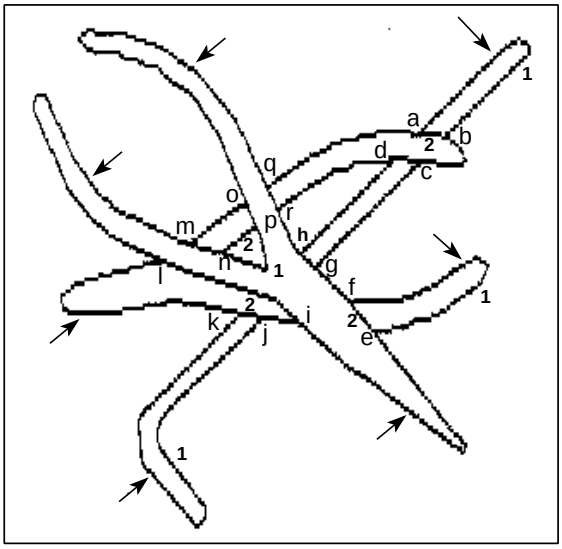

Figure 3: Partial occluding among 3 objects

belongs to the contour of one object. In other terms, vertices are the only way to sense the overlapping of object contours or the case of open contours. A collection of lines that link up an arbitrary number of vertices may or may not belong to the contour of the same object. Answering the question of whether a pair of lines, linked by some vertex, belong or not to the same object requires complex investigation involving the object library models.

The matching algorithm starts by evaluating the fine and coarse polar representations of each line that appear in the scene. It sorts the lines in the decreasing order of the number of segments according to their coarse model. The lines having the largest number of segments necessarily possess richer discriminative information (inflection points) than the others and must then be used in indexing the models to generate hypotheses on possible matching. Indexing with lines having too few number of segments leads to generate large number of potential matches as many models may include varieties of simple features. This has the effect of slowing down the whole recognition process due to the combinatorics generated when comparing many possible matches among the lines. A better method is to retrieve a sufficient number of candidate lines from a list sorted according to the principle of *largest number of features first*. This may represent some percentage of the total number of lines.

Figure 3 shows an example of partial overlapping between three objects. In this example, we have 18 vertices labeled as $(a, b, \ldots, q)$ and 26 lines labeled by the pair of vertices that directly connect. For example, vertices a and b directly connect two contours that are labeled $ab1$ and $ab2$ which are shown on the above Figure by using only their numeric identifier (1 and 2).

Initially all vertices are inactive which means that they do not require any processing until they change their state and become active. The features that belong to the initially selected lines are then used in the indexed search which enables finding one or more matches for each selected line following spatial matching and shape distance matching of the features that will be described in the next Section. This allows lines be directly

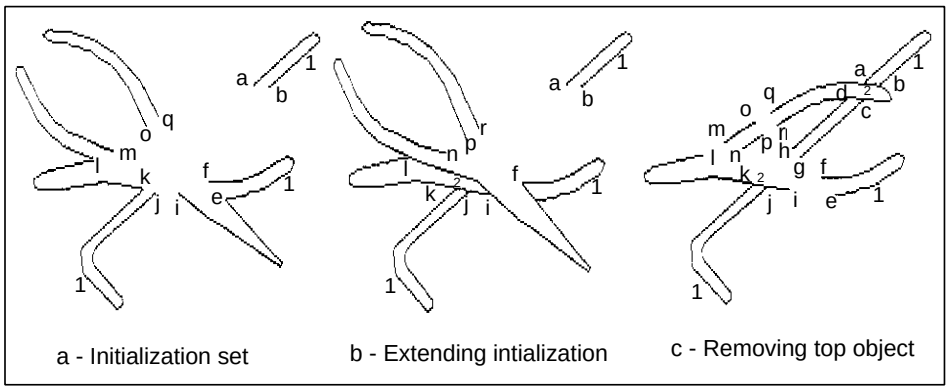

Figure 4: Steps in extending the initialization set to top object and fragments

matched to sub-sets of the library model. Indirect matching of lines will be described later. Directly or indirectly matched lines are called *fragments*. In the example, the set of fragments (27%) found following the initialization step is $\{ab1, ef1, ei, jk1, kl, lm, oq\}$ are those having 3 features or more and are marked with arrows on Figure 3.

The proposed pattern matching approach is vertex-driven. The vertices connected to fragments become active as fragments may be used to extend the matching to some of their neighboring lines which are connected to active vertices. In the example, the active vertices are $(a, b, e, f, i, j, k, l, m, o, q)$ which are shown on Figure 4-a that is obtained after removing all inactive vertices and lines which are contours having poor information.

In the next Section we show how one can find robust initial matching hypotheses which result from carrying out gross to fine matching for the initial set of fragments only.

5.2 Spatial and shape matching

The features associated to each scene contour (A_x) are generally individually matched to features that belong to a relatively large number of models. Each matching ($< f_x, f_m >$) from a scene contour feature f_x to a model feature f_m represents a hypothesis that must verified. Since the number of hypotheses is relatively large, it becomes critical to prune the maximum number of inconsistent hypotheses by using the least costly checking. The reason is that sophisticated verification of all the originally generated hypotheses would dramatically increases the recognition time making it linear in the model size.

Assume a fragment of scene contour A_x has a set of n features $f_{x,1}, \ldots, f_{x,n}$ which have been one-to-one matched to features $f_{m,1}, \ldots, f_{m,n}$ of some model O_m. The ordering of $f_{x,1}, \ldots, f_{x,n}$ corresponds to their order on contour A_x according to a given direction. To consolidate the matching of A_x to model we first check the *geometric matching* between these features which is, in turn, consolidated by the accurate *distance matching*.

The geometric matching consists of comparing the relative position and orientation of features $f_{x,1}, \ldots, f_{x,n}$ according to their setting in the scene to those of features $f_{m,1}, \ldots, f_{m,n}$ according to their setting in the model. For this the position and orientation of each $f_{x,i+1}$ is evaluated with respect to some frame of reference that is attached to previous feature $f_{x,i}$. The above position and orientation are compared to those of the matched features

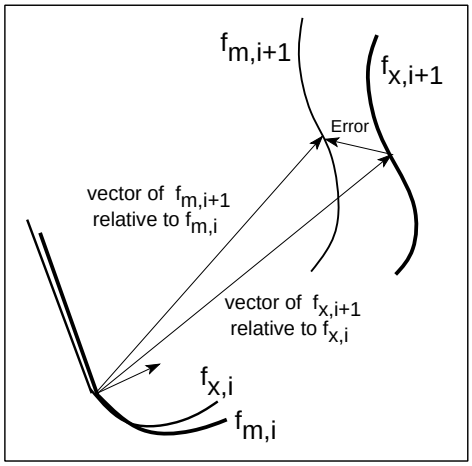

Figure 5: Geometric matching of features

($f_{m,i+1}$ with respect to $f_{m,i}$) with the objective to validate or invalidate the ordered matching $<f_{x,i}, f_{x,i+1}, O_m>$ based on pair $<f_{x,i}, O_m>$ and $<f_{x,i+1}, O_m>$.

Figure 5 shows two scene features $f_{x,i}$ and $f_{x,i+1}$ which are matched to two model features $f_{m,i}$ and $f_{m,i+1}$, respectively. The relative position and orientation vector of $f_{x,i+1}$ is evaluated with respect to $f_{x,i}$ for the scene features and compared to vector $f_{m,i+1}$ that is observed with respect to $f_{m,i}$. Geometric matching consists of a low cost operator that evaluates the error vector $\epsilon(i)$ which is simply the difference between the above vectors. The error vector measures how different are the relative positioning of the scene features from their corresponding model features. A global measure of relative positioning can be defined by adding up the squares of the error vector $\epsilon_{x,m} = \Sigma_{i=1}^{i=n-1} \epsilon(i)^2$ that is associated to the error between $f_{x,1}, \ldots, f_{x,n}$ and $f_{m,1}, \ldots, f_{m,n}$. In other term, $\epsilon_{x,m}$ is the spatial matching error for $<A_x, O_m>$.

The advantage of this approach is the ability to carry out additional model pruning at low cost processing because the number of originally generated hypotheses is still relatively large. Three important characteristics contribute in the efficiency of geometric matching. First, only those contours that with enough discriminative information participate in the original geometric matching. Second, the relative position and orientation of features within the model are pre-computed and need no further processing. Third, the relative position and orientation of the scene features are evaluated once and used in pruning all inconsistent hypotheses among contours having enough connected features. The process of geometric matching allows carrying out low cost verification of the most probable hypotheses that are directly produced by the indexed search. The low cost of geometric matching and the organization of the storage to this effect is one important issue in reducing the dependency between recognition time and size of the database. Verifying the most probable hypotheses by using low cost geometric matching is an essential refinement step prior to applying the more costly shape matching which is described below.

The finest step in verifying the matching $<A_x, O_m>$ consists of evaluating the polar distance between A_x and the portion of contour (A_m) of O_m that is matched to A_x. Geometric matching consolidate the positioning of the sequence of features from scene

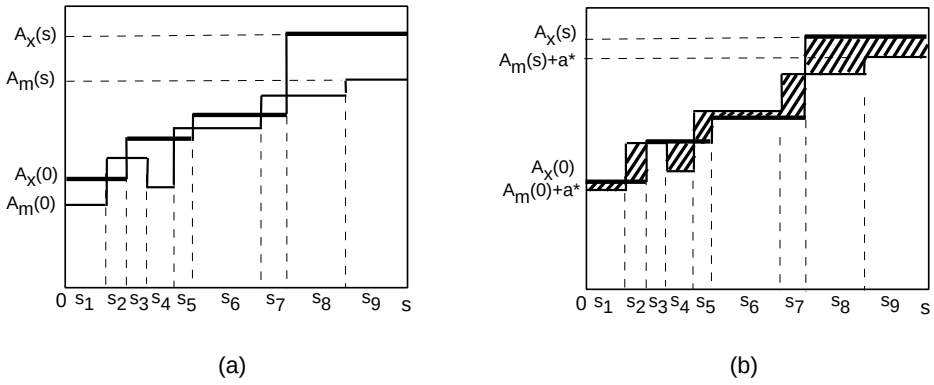

Figure 6: Initial (a) and least (b) Distances between $A_x(s)$ and $A_m(s)$

to model but does not tell how similar are the contour details. The functions $A_x(s)$ and $A_m(s)$ denote the polar models as function of the contour length s. Figure 6-a shows the fine polar models of $A_x(s)$ and $A_m(s)$ that will be used here to evaluate a shape similarity function.

$A_x(s)$ is a set of k polygonal segments with a total length S and $A_m(s)$ has equal length but with L segments. In Figure 6-a $k = 6$ and $L = 5$. The interval $[0, S]$ is divided into N intervals so that in every interval i functions $A_x(s)$ and $A_m(s)$ are constant over the length s_i of that interval. The total length satisfies $S = \Sigma_{i=1}^{N} s_i$. Since the effect of the scene instance on $A_x(s)$ appear as a vertical shift in the polar plan when compared to the model instance. The shift is due to the original orientation of objects with respect to horizontal. The polar distance is meant to be the minimum possible area difference between $A_x(s)$ and $A_m(s)$ versus all possible vertical shift operations. In other terms, the distance function is defined by $d_a(m, x) = \Sigma_{i=1}^{N}(A_x(i) - A_m(i) + a)^2 s_i$ which is a convex function [23] of the vertical shift parameter a that would vertically translate $A_x(s)$ in order to yield the least value of $d_a(m, x)$.

To find the minimum value of $d_a(m, x)$ one needs to find a vertical shift value a^* that minimizes $d_a(m, x)$ which must be the least possible value of $d_{a^*}(m, x) = d(m, x)$ among all possible vertical shifts. For this we differentiate $d_a(m, x)$ with respect to a which gives $\delta d_a(m, x)/\delta a = 2\Sigma_{i=1}^{N}(A_x(i) - A_m(i) + a)s_i$. Since $d_a(m, x)$ is convex function of a the optimum value of a corresponds to $\delta d_a(m, x)/\delta a = 0$ which gives $a^* S = \Sigma_{i=1}^{N}(A_m(i) - A_x(i))s_i$. By substituting a^* into $d_a(m, x) = \Sigma_{i=1}^{N}(A_x(i) - A_m(i) + a)^2 s_i$ we obtain

$$d(m, x) = \Sigma_{i=1}^{N}(A_x(i) - A_m(i))^2 s_i - \frac{1}{S}(\Sigma_{i=1}^{N}(A_x(i) - A_m(i))s_i)^2$$

The normalized distance $d(m, x)/S$ allows finding the smallest area difference (shape distance) between two fragments A_x and A_m of equal length S that is determined by the length of the scene fragment A_x. Figure 7 shows one scene contour (*shape* 1) and two spatially matched model contour (*shape* 2) and (*shape* 3) whose horizontal shifts were set based on feature matching over the length of *shape* 1. Though *shape* 1, *shape* 2, and *shape* 3 have a number segments and corners, as shown on Figure 7-b, function $Min\{d(shape\,1, shape\,3), d(shape\,2,\ shape\,3)\}$ is several fold $d(shape\,1, shape\,3)$.

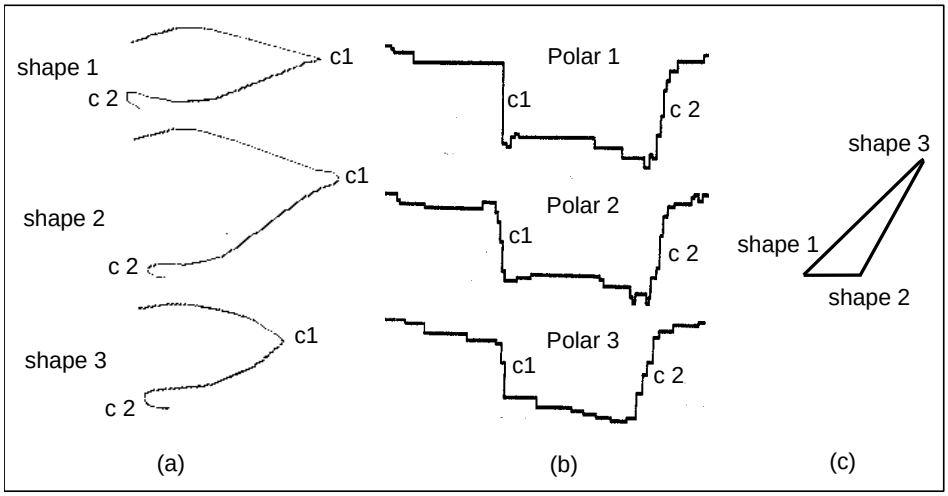

Figure 7: Shape matching: shapes (a), polar models (b), and distance (c)

In summary, the objective of geometric and shape matching is to operate on a subset of hypotheses that are among those having most of the discriminative information so that large pruning of hypothesized models is performed by progressively applying low-cost gross matching to more costly fine matching. The result is a set of *robust hypotheses* for a subset of scene contours that will be used in subsequent steps as the seed for the generation of correct interpretations. Extending the current hypotheses through inter-fragment matching and predictive matching is the next step to come up with a global interpretation. This will be described in the next two sections.

5.3 Selective processing

An active vertex has at least one fragment and a number of lines which means that a vertex becomes active only when at least one of its lines is matched to some model and that line is called fragment (g). Processing of an active vertex consists of attempting the matching of some of its lines to a subset of the models which have already been matched to fragments connected to this vertex. Each fragment g of some active vertex is paired with a line l for possible matching. This consists of appending the line to g in the polar plan (g, l) and comparing it to the models that match the fragment g. In the example of Figures 3 and 4-a, the fragments ab_1, ef_1, and jk_1 could not be matched to their neighboring lines. For example, the pairing $<ab_1, ab_2>$ which are connected at vertex a failed as well as the pairing with line aq because the models to which fragment ab_1 is matched to do not contain any of these combined contours.

Referring to Figure 3 the pairing $<ei, ef, il>$, $<lk, kj>$, $<lm, il, mn>$, and $<oq, op, qr>$ succeeded and the newly matched lines become fragments and their connected vertices are then considered as newly active vertices. By transitivity, the matched chain of fragments and lines is extended such as in the case of chain (fe, ei, il, lm, mn) as more vertices become newly active such as (n, p, r). As a result of the previous matching some lines become fragments and consequently these new fragments fetch inactive vertices to which

they are connected (if any) which become active as some processing can now be done to extend the matching process further. Repeating the above matching process enables extending the previous matching to new chains that are $(gf, fe, ei, il, lm, mn, np_1)$ which can now be combined with chain (np_1, po, oq, qr, rh). An intermediate step of combining matched chains is shown in Figure 4-b where most of the contours that belong to the top object are discovered. The newly active vertices (h, g) enables matching hg to the previous chain thus identifying the top object. Other combined chains can also be matched at this level such as (ij, jk, kl). Clearly, the evolution of vertex-driven matching allows early recognition of top intersecting contours including whole objects and long portion of top contours. Removing of the top object leaves all the lines and fragments that are shown on Figure 4-c.

At this point, we note that the active vertices can be classified into two categories: 1) the vertices that connect only fragments which we call *completed vertices*, and 2) the vertices that connect fragments and lines which we call *blocking vertices*. A completed vertex whose fragments were successfully matched pairwise does not provide any additional information and can then be removed. None of the completed vertices of the example can be removed. By keeping activation of a blocking vertex, poor contours can be fetched in subsequent matching in order to examine potential pairing with other contour having stronger hypotheses.

A completed vertex with at least one fragment that could not be matched across this vertex to other fragments or lines must remain active or the algorithm blocks as no further move can be done. In the example completed vertices that must remain active are $(e, f, k, g, i, l, o, p, m, n)$ which appear on Figure 4-c. On the other hand, blocking vertices are $(a, b, g, h, m, n, o, p, q, r)$ as each of these vertices still have at least one line.

The above blocking can be removed by extending the search through pairing of hypotheses and predictive matching which will be described in the next subsections.

5.4 Pairing of hypotheses

Pairing of hypotheses applies to active vertices that have fragments which could not be matched to other contours of the same vertices. This situation is frequent in partial occluded scenes. For example, fragment ef_1 (Figure 4-c) could not be matched to any neighboring contours at vertices e and f.

To extend further the matching, searching for fragment-fragment matching can give fast result because strongly hypothesized contours of the same object have identical matched model among those that received the highest vote following the indexed search. Pairing of hypotheses can done by simply checking whether the currently unconnected chains have some common model matching. For example, chains (lk, kj, ji) and (ef_1) for which the correct model to which they belong to is likely to be among the current hypotheses. For this the algorithm picks up a matched model that is common for two hypothesized chains and compares the relative position and orientation of these fragments in the scene to that present in their hypothesized model. This task has low overhead because each of the fragments has previously been matched to a sub-set of features of the same model and the costly shape matching has already been successful for each fragments as previously described in Section 5.2.

Assume two scene fragments g_1 and g_2 that are matched to some contours denoted by g_1^* and g_2^* of the same model. For this, one may compare the position and orientation of each pair of features from g_1 and g_2 to those of the matched features from the model. Since each fragment can be formed by numerous features the cost of such a comparison can be relatively high. A less costly approach is to consider the fragments as two arbitrary contours and carry out the comparison by using a distance that constrain their relative position and orientation. For this we choose two points (x_1, x_2) on g_1 and (y_1, y_2) on g_2 so that any combination of three points out of (x_1, x_2, y_1, y_2) is not co-linear. Based on previous feature matching with the models, choose x_1^*, x_2^*, y_1^*, y_2^* as the points of g_1^* and g_2^* that correspond to x_1, x_2, y_1, y_2, respectively. Now the position and orientation of g_2 with respect to g_1 can be matched to that of g_2^* with respect to g_1^* by comparing the distance between every pair of points (x, y) in the scene to the corresponding distance in the model. In other terms, we evaluate a distance error that characterizes the position and orientation of these fragments in the scene and in the model. The distance error $d(g_2/g_1, g_2^*/g_1^*)$ is given by

$$d(g_2/g_1, g_2^*/g_1^*) = \sum_{k}^{2} \sum_{l}^{2} \frac{|d(x_k, y_l) - d(x_k^*, y_l^*)|}{Min\{d(x_k, y_l), d(x_k^*, y_l^*)\}}$$

where $Min(.,.)$ is used to obtain a relative distance error which guarantees that $d(g_2/g_1, g_2^*/g_1^*)$ is a small percentage in case of correct matching. As each fragment is generally hypothesized with few models the above testing can easily identify the model that provide the best matching or more than one matching should be kept if the corresponding distance errors are small. At this level, the presence of multiple possible matching implies that each of these hypotheses must be checked in subsequent matching extension.

For the example shown in Figure 4-c the fragments (ij, jk, kl) and (ef_1) is one example of successful pairing of hypotheses. Pairing of fragments (jk_1) and (ab_1) is another example. Successful pairing results in appending sparse chains together and we assume that pairing is transitive operation for subsequent matching.

5.5 Predictive matching

Predictive matching is an advanced step in the recognition because all contours having significant discriminative information have already been hypothesized and there is still some contours (lines) that must participate in making the global interpretation. When lines and fragments do not match at some vertices, there is need to attempt matching distant fragments and lines because this situation is typical of partial occluding scene and contours that are altered by noise and lighting. At this level of the recognition, all the remaining vertices become active in order to attempt matching lines with previously hypothesized chains.

Examining of the *geometric relationships* between a fragment g at vertex u ($<g, O_m>$) and a line l at vertex v enables extending the matching process to l, i.e. whether $<l, O_m>$ holds or not. Since a line has poor discriminative information our approach narrow its matching to small number of models $\{O_m\}$ for each neighboring chain. The question is how to efficiently search for a line from the scene that is can also be present in one or more of the matched models of some neighboring fragment g.

One method is to report l into each model O_m that is matched to g and evaluate the shape matching distance for the new postulated chain $<g, l, O_m>$ and repeat this operation for all the scene lines that are likely to fall onto the model. This requires repeatedly evaluating the shape matching distance for each scene line by shifting it in the vicinity of g within model O_m. Since the number of active vertices is still large, this approach is computationally inefficient and there is need for a low cost matching operator that must backtrack as soon as some inconsistency is detected. In other terms the cost of verifying $<g, l, O_m>$ must initially be as low as possible and can be allowed to increase in cost only when enough confidence is accumulated.

The method used consists of three ordered steps: 1) vector matching, 2) orientation matching, and 3) shape matching. It is possible to backtrack at each step to abandon the current search when any mismatching occurs. For vector matching, we evaluate the vector uv by selecting a vertex v that the nearest unvisited vertex to u. The vector is reported with respect to edge u in each of the models $\{O_m\}$ to which g is matched to. If the reported vector points to some contour point w of the model then the relative orientation (tangential) of l with respect to g in the scene is compared to the relative orientation of w with respect to g in the model. The shape distance matching is attempted only when vector and orientation matching succeed. Otherwise, the next model to which g is matched to is taken and the previous steps are repeated.

The cost of vector matching is the lowest and involves simple access of the model and evaluation of a distance because the model contains a sorted list for the coordinate of the contour points. The cost of the second operator is also very low because the polar representation directly gives the relative orientation (difference) between two polar segments. Therefore, pruning many inconsistent matching ($<g, l, O_m>$) can be done at low cost.

Referring again to figure ref3over-c. Predictive matching allows fragments kl to be matched to line mo, mo is matched to qa, and so on. This results in matching the chains $(kl, mo, qa, ab2, bc, cd, dr, pn)$, $(jk1, hd)$, and $(jk1, gc)$ that each necessarily contains at least one initialization fragment.

5.6 Interpretation

During recognition, each fragment of contour g retains a number of valid hypotheses that are processed every time g is involved in some matching extension like the pairing of hypotheses or predictive matching. Now each matched model of fragment g accumulates some vote that is the length of all the fragments and lines which have been successfully matched to g. The retained models are taken as those having the highest vote when all possible matching have been completed for the retained hypotheses. Each fragment is hypothesized to one of its matched models that received the highest vote which generally allow complete clustering of the scene.

Note that in some cases the originally generated hypotheses of some fragment may not be matched to any other scene fragment because the hypothesized contour is present in many models and the selected hypotheses are incorrect. In this case these hypotheses accumulate relatively very low vote (sporadic). To avoid this blocking the algorithm backtrack into the originally generated hypotheses and retrieve the next highest hypotheses.

This backtracking has local effect because the newly fetched hypotheses are then incrementally matched with the currently running hypotheses at all levels of the recognition.

6 Performance evaluation

In this Section we present analysis of the proposed modeling and recognition by evaluating the proposed scheme with respect to storage requirements and effect of database size. We finally compare our approach to other proposed approaches.

6.1 Storage requirements

Decomposing contours by using constant curvature criterion was proposed first By Wuescher et al. [17] and latter used in other proposals. However our work differ from that of [17] which directly use the curvature values to index into the model. In our approach we used a 3-dimensional features that include at most two constant curving factors and a shift angle which represents the basis for our multi-dimensional indexing. The problem with [17] is that increasing selectivity by reducing the granule size of the bucket has the effect of degrading the discriminability of the whole scheme because this reduction occurs at the expense of the levels of noise that the system can handle. This effect has been extensively studied by Cafifano and Mohan in [9] which proposed the adoption of larger indices to keep a relatively coarse bucket quantization without scarifying selectivity.

Our approach consists of increasing the dimensionality of the indices (feature parameters) as a method to increase selectivity. On the other hand, we have partitioned our 3-D index space into seven distinct hashing tables in an attempt to decrease the number of entries per table which has the effect of reducing the storage size of these tables without affecting the discrimination power of the indexing scheme.

The maximum number of buckets in this approach is $N_{max} = \Pi_{i=1}^{3} R_i / \Pi_{i=1}^{3} 2\epsilon_i$, where R_i is allowable range of the ith parameter and ϵ_i is the overall variance on that parameter due to various effects such as noise, digitization, and theresholding. In this case, $N_{max} = R_h^2 R_\theta / (8\epsilon_h^2 \epsilon_\theta)$, where R_h, ϵ_h, R_θ, ϵ_θ are the range of the curving factor, its variance, the range of the angular shift, and its variance, respectively. Since parameters ϵ_h and ϵ_θ are global variances, therefore, it is important to optimize to bucket size by directly relating the bucket size to discriminability.

For this we heuristically searched for the best possible bucket size by stepping the size around the value of $R_{min} = 8\epsilon_h^2 \epsilon_\theta$, which is the smallest bucket size, after estimating the values of ϵ_h and ϵ_θ based on known thresholding and repetitive acquisitions of features by varying their position and orientation. We first study the storage required for the indexing tables versus the bucket size for a 100-object database. To construct the indexed tables, each object is scanned several time by randomly setting the position and orientation. Table entries are created for the resulting parameters of the features which include variances due to noise and digitization. This process affects the storage volume as the same feature may create multiple entries due to: 1) current bucket size, and 2) the variance on the feature parameters.

Figure 8: Storage size and recognition time versus bucket size

The current bucket size is $R_{h,\theta} = R_h^2 R_\theta = \alpha R_{min}$, where α is the bucket size factor that is studied here in the range $0.25 \leq \alpha \leq 2.25$ by using steps of 0.25. The *Storage Size* ($S(\alpha)$) is defined as $S(\alpha) = st(\alpha)S_{min}$ where S_{min} is the minimum storage over the studied range of α and $st(\alpha)$ is the storage factor. Figure 8 is used to plot the storage size and the normalized recognition time versus the bucket size. Figure 8-*storage size* shows how the storage size grows versus increasing or decreasing the bucket size around reference R_{min} (Bucket size factor of 1 in the figure) in the case of 100-object database.

Because of the large number of features and the variety of object shapes the storage is likely to be constant when bucket size ($\alpha \in [1, 2.25]$) exceeds R_{min} because of non-uniform parameter distribution. When the bucket size ($\alpha \in [0.25, 1]$) is below R_{min} the storage linearly increases when the bucket size decreases because the feature parameters randomly fall into neighboring buckets thus creating duplicate entries in the tables.

The value of R_{min} is just an indicator and the optimum bucket size results from many superimposing effects that are difficult to model. One needs to link the bucket size with the discrimination power in order to find an appropriate bucket size. One way to handle the discriminability versus the bucket size is to consider the recognition time for a small set of features, having identical type, which includes the time to generate hypotheses as well as the verification time. For each experiment we consider a group of four features that appear on the contour of some database object such that all four features have either of the following types: $type - 1\&4$, $type - 2$, $type - 3\&6$, $type - 5$, and $type - 7$. The grouping by type is meant to have equal feature complexity within each group. We relate the average time to classify four features of a given type to the bucket size with the idea that minimum classification time corresponds to maximum discriminability for the set of features. Clearly, our discrimination function is inversely proportional to the recognition time measured under different instances of bucket size. Features considered are similar to those found in Figure 3.

Our objective is to link the selection of the bucket size to the ability to discriminate fragments which includes generation of hypotheses, cost of verifying them, and cost of carrying out accurate distance matching. This is likely to involve most of the recognition system. However, there are different ways the recognition time of some features can be affected by the bucket size. Due to their shape, features may have strong dependence on bucket size when there is some potential similarities among group of features. Other isolated features have recognition times that are much less sensitive to change in bucket size. Therefore, we randomly selected a 10 set of 4 features, each set of 4 is taken from one model, and averaged their overall recognition time which is shown on Figure 8 for each type.

Type 2 has a two straight segments separated by some shift which the searching key which explain why the recognition time is linear function of database size (Figure 8-$type-2$). Types $3-6$ and $1-4$ have one straight and one curved segments and differ only by the presence or not of the shift separator. Their recognition time slightly indicates (the minimum) the need for a specific bucket size. Finally, types 6 and 7 have two curving factors and differ only the shift separator. Here there is strong dependence of the recognition time over the bucket size. Since generally, searching by using a set of features and classification involve a mixing of all the feature types, therefore, we need to find a bucket size that is suitable (minimum recognition time) for most of the above cases. After examining the average recognition time for different sets of features we decided that the best discrimination for this approach corresponds to $1 \leq \alpha \leq 1.75$. Our final setting was $S(\alpha) = 1.4S_{min}$.

By this approach we avoid excessively reducing the bucket size to gain selectivity but instead we preferred setting the bucket size to some level so that to minimize the recognition time for a sufficiently large number of random features. The discrimination associated to recognition of whole object that is used by Califano and Mohan [9] is defined as V_c/V_w, where V_c is the votes for the correct shape instance and V_w is the maximum votes received for the incorrect shape instance. This function applies to whole object recognition which incorporate the accumulation of evidences from many features. In this work, the discrimination is applied to fragments of contours which exposes the recognition system to a finer-level recognition that must be frequently carried out when recognizing partially occluding situations. Another difference is that the discrimination used here incorporates the generation of hypotheses, the cost of verification, and the distance matching. This process definitely classifies the fragments, thus strongly reduces the remaining task of linking the fragments to find globally correct interpretation.

6.2 Effects of the database size

One important issue in evaluating performance of recognition systems is the study of the dependence of the recognition process on the number of database objects. Scenes of 4 objects with partial occluding are studied. The objects are mechanical items (tools, keys, etc.) and flat plastic shapes. The number of features for these objects range from 8 to 20 per object. Thresholds used throughout the modeling were experimentally evaluated by repeatedly tracing contours under different position and orientation which allowed setting estimated tolerance parameters at each level.

Hypotheses pruning versus database (DB) size

DB size	hypotheses	Ranking 1	Ranking 2	Ranking 3	Recog. time	% increase
10	61	16%	14%	25%	48	–
40	258	7.6%	6.2%	10.4%	51	6.25%
70	497	4.3%	3.8%	6.7%	53	10.4%
100	796	2.8%	2.5%	4.5%	55	14.5%
130	1172	2.1%	1.8%	3.25%	58	20.8%

Table 1: Hypothesis generation, ranking, and recognition time

Database setting for n-object is denoted by DB_n and the studied instances of n are 10, 40, 70, 100, and 130. We started by setting up the model for database $n = 10$ by using the 3 objects of the scene with other randomly selected objects. To build the database with $n = 30$ we randomly selected 20 more objects and added them to DB_{10}, and so on. The recognition algorithm is run under each of the database settings for recognizing the scene shown on Figure 3.

The indexed search provides hypotheses of the mapping from scene features to features of the models. Note that a feature may be hypothesized more than once within the same model because of possible parameter matching with distinct features. Each feature of some scene object accumulates votes for matching with features of the models. The ranking of a feature is taken as the minimum percentage of hypotheses generated which contains the correct matching. The ranking of an object is the average of the ranking of all its features. Table 1 shows the total number of hypotheses generated and the ranking for all the three scene objects versus the size of the database. The reason for the large number of hypotheses generated is that most individual features are found in many object feature instances due to noise, digitization, thresholding, and bucket tolerance. This does not poses a major problem as far as we do not retain all these hypotheses but only a very small subset will be further processed. For example, object-2 received a ranking of 6.2% under DB_{40} which indicates that, on the average, the number of votes received by each of its features for the correct matching was among the top 6.2% among all the highly voted matching. Though the number of total hypotheses generated is at least quadratic in the database size, the algorithm spends a small fraction of the recognition time on processing of these hypotheses because only a small fraction of these hypotheses are to be verified and the remainder is pruned.

The fraction of hypotheses that are checked for their spatial relationships is slightly larger than the percentages shown on Table 1 which depend on the database size. Checking for the spatial relationships between pair of features has relatively larger overhead than hypothesis generation but its global overhead is moderate because the set of potential matching is small after the hypotheses pruning step. Spatially unmatched hypotheses are further pruned prior to carrying out the accurate distance matching which has the highest overhead. Overall recognition time of the three objects is shown on Table 1 together with the percent increases in the recognition time over that obtained for DB_{10}. The recognition time is likely to be independent from the number of hypotheses originally generated. The time increases is relatively small as the algorithm must spend only 20% extra time when the database size becomes 13 fold that of DB_{10}. By experimentally choosing the ranking

percentages of hypotheses, the *number of retained hypotheses* becomes nearly constant regardless of the database size. Under this condition, the recognition algorithm would likely depend only on the scene complexity without explicit dependence on the database size.

6.3 Comparison to others

In the following we compare our approach to other model-based recognition that are hierarchical in structure and scale citeart:ettinger.

In [6] all scene features participate in the generation of hypotheses that are ranked by mutual support. This consists of reporting the matched models into the scene and collecting supporting evidences whenever they map to similar locations. Due to the large number of initially generated hypotheses, a massively parallel machine (CM5) is used to parallelize the complexity of scene and each processor carry out verification of one hypothesis. A confidence degree that is the percentage of total matched line segment weighted by the line segment length is evaluated for each interpretation which allow selecting the instance with the highest confidence. Results show that recognition time mainly depends on scene complexity but with only secondary dependence on the model size. As a result of parallelism the recognition time of few objects is about that of recognizing one object. These results are partially due to the use of huge parallelism which hide the cost of verifying many improbable hypotheses.

Our approach avoids generating large number of hypotheses by selecting contours that have rich discriminatory information to generate initial hypotheses. Poor fragments are questioned latter on the validity of current hypotheses. To prune large fraction of the models a depth first hypothesis verification is carried on from gross spatial matching among features to fine shape matching. To extend the current hypothesized matching the scene vertices (intersecting fragments) are progressively activated to exploit contour continuity between strongly hypothesized fragments and poor fragments. Objects on top in the scene are rapidly recognized which facilitate predictive matching of contour fragments that intersect with top objects. A global interpretation that leaves the least percentage of uncovered contours is selected. Though a serial processing is used, the fraction of the recognition time that is database dependent is a small percentage of overall processing time.

7 Conclusion

In this paper we presented the design and implementation of a model-based pattern recognition system based on coarse-to-fine matching. Shapes are modeled by using the polar representation of contours for which we developed a metric for efficient shape matching that can be used for comparing whole object contours as well as fragments of contours.

The aim of optimizing the model structure and the searching algorithm is to reduce significantly the explicit dependence of the recognition time over the database size. To enable early pruning of large portion of the models, only scene contours among those having most of the discriminative information are selected for the generation of initial

hypotheses. This allows avoiding early processing of contours having poor discriminative information which otherwise would increase the dependency over the model size. Such poor contours are used latter in validating or invalidating a relatively small set of robust hypotheses.

The generation of robust hypotheses enabled us categorizing the contour intersection points with respect to discriminability. The result is some processing that is driven by discriminability which can be extended to whole scene by using predictive matching and pairing of hypotheses. The approach was intended to avoid brute force processing of a large number of hypotheses that must be processed regardless of discriminability. Reducing the storage size without affecting discrimination power was carried out by relating the bucket size to typical recognition time. To increase selectivity, we used a 3-D index that is further partitioned into several optimized indexed schemes in an attempt to decrease the number of entries without loss of discriminability.

Evaluation shows that the recognition time is nearly independent from the number of hypotheses originally generated. The time increase due to increase in the database size is relatively small because the model and the algorithm are designed so that the algorithm spends a small fraction of time in global database processing. This approach proved to be efficient for a 100-object database.

References

[1] G. J. Ettinger. Large hierarchical object recognition using libraries of parametrized model sub-parts. *Proc of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 32–41, 1988.

[2] T. Pavlidis. Algorithm for graphics and image processing. *Comp. Science Press, Rockville, Md.*, 1982.

[3] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. Robotics: control, sening, vision, and intelligence. *McGraw Hill Inter. Eds.*, 1987.

[4] X. Chin and C. R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18, No. 1:67–108, 1986.

[5] P. W. M. Tsang and P. C. Yuen. Recognizing of partially occluded objects. *IEEE Trans. on Systems, Man, and Cybernatics*, 23, No 1:228–236, Jan-Feb 1993.

[6] L. W. Tucker, C. R. Feynman, and D. M. Fritzsche. Object recognition using the Connection Machine. *Proc of the IEEE Conf. on Computer Vision and Pattern Recognitiopn*, pages 871–878, Jun 1988.

[7] N. R. Corby. Machine vision for robotics. *IEEE Trans. on Industrial Electronics*, 30, No 3:282–291, Aug 1983.

[8] W. E. L. Grimson and D. P. Huttenlocher. On the verification of hypothesized matches in model-based recognition. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 13, No 12:1201–1213, Dec 1991.

[9] A. Califano and R. Mohan. Multidimensional indexing for recognizing visual shapes. *Proc of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 28–34, Jun 1991.

[10] N. Ayache and O. D. Faugeras. HYPER: A new approach for recognition and positioning of two dimensional objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 1:44–54, Jan 1986.

[11] B. K. P. Horn. Robot vision. *The MIT Press*, 1986.

[12] X. Bolles and R. A. Cain. Recognizing and locating partially visible objects: the local-features-focus method. *Inter. J. of Robotics Research*, 1, No. 3:57–82, 1982.

[13] W. E. L. Grimson. On the recognition of curved objects. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 11, No 6:632–642, Jun 1989.

[14] A. Rosenfeld and J. S. Weszka. An improved method of angle detection on digital curves. *IEEE Trans. on Computers*, C-24:940–941, 1975.

[15] W. Richards, B. Dawson, and D. Whittington. Encoding contour shape by curvature extrema. *Journal of Optimization America A*, 9, No 3:1483–1491, Sep 1986.

[16] M. Brady and H. Assada. Smoothed local symmetries and their implementation. *Inter. J. Robotics Research*, 3, No. 3:36–61, 1984.

[17] D. M. Wuescher and K. Boyer. Robust contour decomposition using a constant curvature criterion. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 13, No 1:41–51, Jan 1991.

[18] J. L. Turney, T. N. Mudge, and R. A. Volz. Recognizing partially occluded parts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7, No. 4:410–421, Jul 1985.

[19] Y. Lamdan, T. T. Schwartz, and H. J. Wolfson. On recognition of 3-D objects from 2-D images. *IEEE lnter. Conf. on Robotics and Automation*, Vol. 3:1407–1413, 1988.

[20] A. Kalving, E. Schonberg, J. T. Schwartz, and M. Sharir. Two dimensional model based boundary matching using footprints. *International Journal of Robotics Research*, 5, No 4, 1986.

[21] T. F. Knoll and R. Jain. Using features indexed hypotheses. *Technical Report RSD-TR-10-85, University of Michigan, Robot Systems Division, Center for Research on Integrated Manufactiring*, 1985.

[22] W. E. L. Grimson. The combinatorics of heuristic search termination for object recognition in cluttered environments. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 13, No 9:920–935, Sep 1991.

[23] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. An efficient computable metric for comparing polygonal shapes. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 13, No 3:209–216, Mar 1991.

[24] O. D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-D objects. *Int. Journal of Robotics Research*, 5, No 3:27–52, 1986.