A Pattern Recognition System Driven By Discriminability

Mayez Al-Mouhamed

Computer Engineering Department
CCSE, King Fahd University of
Petroleum and Minerals
Dhahran 31261, Saudi Arabia.
mayez@ccse.kfupm.edu.sa

Abstract

This paper presents 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. 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 avoids 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 model size is relatively small as a result of selective processing and optimization of global model effects.

1 Introduction

An effective model-based recognition system [1, 2] must be capable of retrieving the best matched objects as well as carrying out massive pruning of inconsistent models. Modelling objects by their local geometric features [1] 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.

The efficiency of the matching depends to a large extent on the scalability [1, 3, 4] of the recognition operator which is the ability to recognize whole contours as well as fragments of contours. For this, the extracted features [5] must be local and small enough to match wherever they are present but must also be stable and discriminative. The features can be used as searching keys in indexing/hashing schemes [6].

Model organization was studied by Califano and Mohan [4] which proposed the use of larger indices (multidimensional indexing) to keep a relatively coarse bucket quantization without scarifying selectivity. The synergy of the indexing scheme must be small enough because all the models are potentially involved in the initial search [7, 6]. Kalving et al. [8] used a hashing descriptor that is derived from the relationships between lengths and relative orientation of contour segments. Knoll and Jain [9] 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. Grimson [3, 10] 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.

Our objective is to optimize the model and the search so that the recognition time would mainly depend on the scene complexity. Our model and algorithm are designed so that a small fraction of time is spent in global model processing. The aim is to avoid early processing of fragments having poor discriminative information. We selectively process the most robust hypotheses which are validated further through *spatial* and *shape matching*. Shape matching enables comparing whole and fragment of contours. Selective recognition provides efficient pruning of large inconsistent hypotheses. Thus the processing scheme is driven by discriminability. 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 object modeling. Section 3 presents the recognition system. Section 4 presents the evaluation. In Section 5 we conclude about this work.

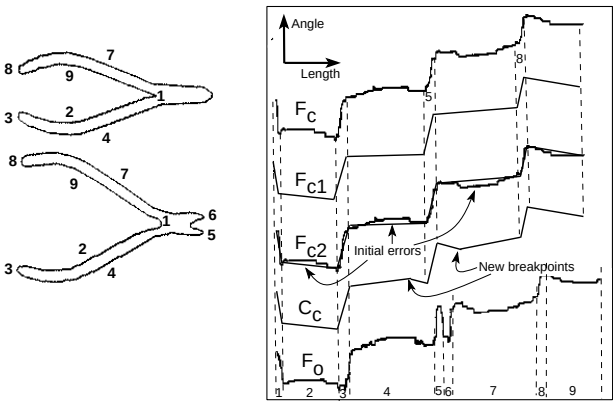


Figure 1: A cutter and its fine and coarse models

2 Object modeling

Our approach uses the well known coarse-to-fine matching concept. We use an angle-length model that provides scale, rotation, and translation invariant properties.

Edge detection allows extracting the shape of object by detecting the presence of an edge at some pixel. For this an approximate of the gradient magnitude is evaluated and used with the sign of the *Laplacian* to determine whether a given edge pixel is located on the background or on the object side of the edge. To reduce the sensitivity to noise the *sobel operator* is used for averaging the gradient over larger pixel neighborhood. An edge is detected when the gradient magnitude exceeds some threshold that is experimentally determined.

The contour is encoded by using the *pixel direction coding* [11] which consists of encoding each border pixel by its direction with respect to previous border pixel. For this a conventional 8-direction convolution mask is used. Each chain of connected contour pixels is represented by a chain of directions with one reference point at starting pixel.

A segmentation algorithm is used to build fine angle-length model of contour by breaking down the contour into a sequence of straight segments. The algorithm repeatedly picks a group of contour pixels, located next to current segment, and compare their average direction to that of current segment. The size of each group of pixels is 2^k, where k is a small integer. A threshold is used to detect the presence of a coarse breakpoint whenever the difference between the above directions exceeds some threshold. The fine breakpoint is searched within a neigborhood formed by the last $2^k - 1$ pixels of current segment and the coarse breakpoint pixel. We carry out binary splitting of the above neigborhood and the direction of each split pixel (fine breakpoint) is matched to the updated direction of current segment. As validation, the average direction of the 2^k pixels following the current fine breakpoint must exceed that of current segment by the same threshold.

Each segment is associated its length and its angle with respect to the previous segment. Formally, the kth segment D_k is formed by a pair of breakpoints $b_k = (x_k, y_k)$ and $b_{k+1} = (x_{k+1}, y_{k+1})$. The length s_k of D_k is $s_k = (\Delta x_k^2 + \Delta y_k^2)^{1/2}$, where $\Delta x_k = x_{k+1} - x_k$ and $\Delta y_k = y_{k+1} - y_k$. 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}.

Segmenting of the contour enables building a fine angle-length model of contour, denoted by $F = \{(\theta(s_i), s_i)\}$, which consists of an ordered set of segment lengths s_i and their geometric angles $\theta(s_i)$. Figure 1 shows the correspondence between the contours of a cutter (left part) and its fine angle-length 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. Changing the initial orientation produces fine models that differ in their starting segment. Note that long straight segment of contours are associated horizontal straight segments in the angle-length graph. A sequence of segments that corresponds to a constantly curved contour can then be associated one coarse segment with constant slop.

2.1 Coarse segmentation

The fine angle-length model is inefficient to directly extract geometric features out of fine-grain segments. Too simple features may occur in many models which make the search inefficient. Too complex features have two drawbacks: 1) cannot be observed from partial contours, and 2) lead to linear search across the model. The features should contain enough discriminatory information to order to provide efficient and accurate indexing of candidate models.

Thus we need to build a sketch of the contour or *coarse model* (C) by clustering fine segments having constantly curved contours into *coarse segments* linked by inflection points. In the angle-length plan, non-horizontal segments represent constantly curved contours and horizontal segments correspond to straight contours. We present a method to build stable local shape features.

A fragment of contour that is constantly curved is represented in the fine model by a sequence of small segments $\{\theta(s_i), s_i\}$, where s_i is the length of ith straight segment and $\theta(s_i)$ is the exterior angle between segment s_i and its neighbor s_{i-1}. Segment s_i is a linear approximation of a small contour region, thus the ratio $h_i = \theta(s_i)/s_i$ can be considered as an approximation of the curving for the corresponding contour for small s_i. Segmenting of the fine model consists of clustering all neigboring segments for which the signed ratios $\theta(s_i)/s_i$ are nearly constant along a given sequence of segments.

The coarse segmentation algorithm is based on the following two steps. The first step consists of selecting breakpoints among the fine segments corresponding to strong change in the direction which is equivalent to thresholding the derivative of the gradient. The above breakpoints are temporarily linked by straight *coarse segments* in the angle-length plane. This is shown in the transformation from F_c to F_{c1} of Figure 1. The second step consists of creating

additional breakpoints when the maximum distance from a new coarse segment to fine contour exceeds some threshold as shown on the plot of F_{c2} of Figure 1. In this case, the segment edge of the fine contour that is the fartest from current coarse segment is added as a new breakpoint. This results in the coarse model C_c having 10 segments that is shown on Figure 1. The stability of breakpoints needs to be improved because of the effects of digitization, rotation, shadows, and lighting.

We attempts to correct a situation where a coarse polygonal segment may be subject to some fragmentation during the initial segmentation because of noise in the fine model. In this case, neigboring segments having close orientation angles are fused into one single coarse segment that must have bounded orientation error compared to the originally fragmented fine segments (validation). On the other hand, a fragmented contour formed by three initial neigboring segments S_{i-1}, S_i, and S_{i+1} can be represented by a *corner* if the following three conditions are met. First, we have to make sure that S_{i-1} and S_{i+1} are not nearly parallel which causes some loss of accuracy if the above segments are modified by extending S_{i-1} and S_{i+1} until their intersection. Second, the length of S_{i-1} and S_{i+1} must largely exceeds that of S_i in order to avoid confusing a corner with a true fragmented contour. Third, the relative orientations of S_{i-1} and S_{i+1} with respect to S_i must be of oposite sign. This is needed to make sure that the corner does not form an inflection point. When all three conditions are met the polygonal approximation of S_{i-1}, S_i, and S_{i+1} is modified by cancelling S_i and extending S_{i-1} and S_{i+1} to their intersection.

The *coarse model* C is defined by the resulting collection of segments in which each segment S_k is characterized by three parameters that are: 1) the initial angle $\theta_{init}(S_k)$, 2) the total angular change $\Delta\theta_e(S_k)$, and 3) the segment length S_k. Note that S_k denotes a coarse segment and its length. The initial angle $\theta_{init}(S_k)$ is the exterior angle between segments S_{k-1} and S_k that is the turning angle from S_{k-1} or its tangent if S_{k-1} is curved and S_k or its tangent if S_k is curved. The total angular change $\Delta\theta_e(S_k)$ is the turning angle from the tangent to S_k at its start point to the tangent to S_k at its end point. Formally, the coarse model C is an approximation of the original contour by means of an ordered set of constantly curved segments, i.e. $C = \{S_k = (\theta_{init}(s_k), \Delta\theta_e(s_k), s_k)\}$.

2.2 Feature extraction

A recognition system must exploit the *local geometric features* carried by the contour fragments in order to classify these fragments and link up sub-set of segments in an attempt to find a complete scene interpretation. Features must be simple enough to be locally present and completely observed on relatively short contours. 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.

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 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 angle-length plan in front of each type. A feature discriminates the curving factors of connected segments as well as the actual angular shift between them.

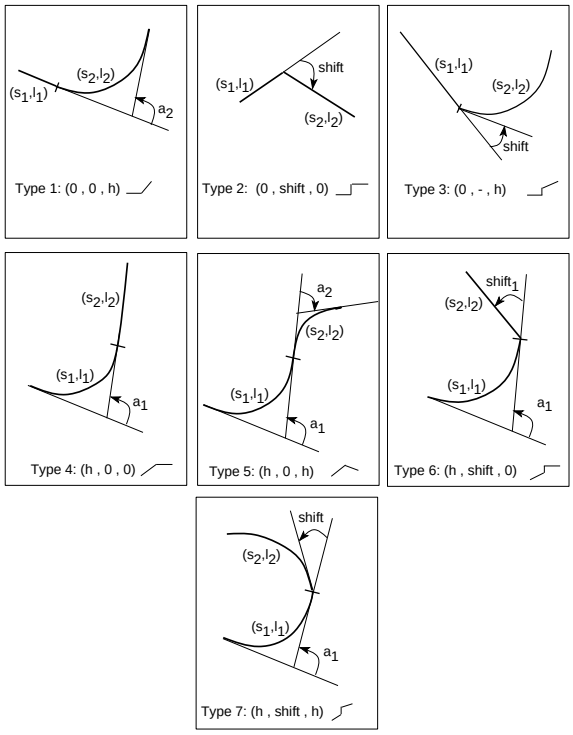


Figure 2: Features and their associated types from 1 to 7

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.

The main effect of breakpoint selection is the introduction of noise in the values of the $shift$ parameter. However, the noise due to selection of beakpoints has much less effect on types 1, 4, and 5 which have no shift parameter. For types 2, 3, 6, and 7 we strogly reduce the effect of breakpoint selection by creating features only when there are enough *evidences* and *confidence* in the presence of distinct semgents. In other words, a feature is created only when the two adjacent segments have distinctive curving factors. Another alternative is to eliminate the $shift$ parameter from the above types and then reduce the number

of types to 4. Unfortunately this approach also reduces the selectivity of the recognition because of the implied increase in the number of entries in the resulting buckets. The selectivity is improved by using the $shift$ parameter especially when the creation of a feature (types 2, 3, 6, and 7) is conditioned by a $shift$ value that exceeds some noise dependent threshold which is experimentally determined for each type.

2.3 Model organization

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. 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 its 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 upper bound on tolerance for each parameter is experimentally found. 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.

3 The recognition system

Grimson [10, 12] 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.

Our approach consists of initially selecting a sub-set of scene contours among those having the largest number of features among all scene contours. In other terms, poor contours are not processed in the early stages of our recognition approach but used latter. Pruning and verification of the initially generated hypotheses is done through application of a low low cost *spatial matching*. Further refinement of the previously verified hypotheses consists of carring out accurate *shape distance matching*. At this level, the retained hypotheses on the fragments represent a small fraction of the original hypotheses. Clearly in our approach the matching complexity increases but the problem size significantly decreases as we move further in the recognition. In the following we present the detail of this approach.

3.1 Initialization

In our representation, a vertex is the intersection point of contours or an end point of open contour. A contour that links up a pair of vertices is called a *line*. At least three lines intersect at each vertex in the case of connected 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.

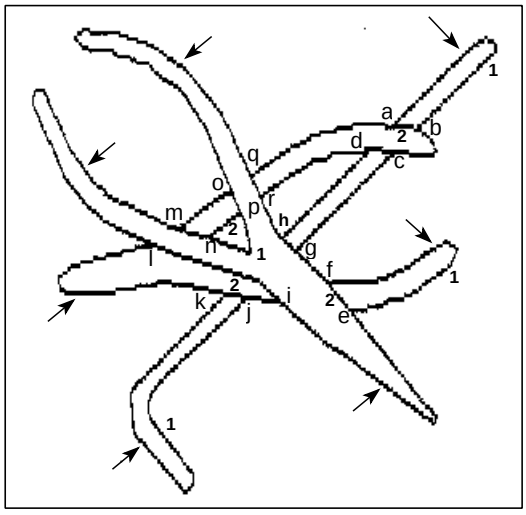


Figure 3: Partial occluding among 3 objects

We start by sorting the lines in the decreasing order of the number of segments according to their coarse model. The lines having the largest number of segments have richer discriminative information than the others and used in indexing the models to generate hypotheses on possible matching. We retrieve a sufficient number of candidate lines from a list sorted according to the principle of *largest number of features first*. This represents some percentage of the total number of lines.

Figure 3 shows an example of partial overlapping between three objects. There are 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 $ab2$.

Initially all vertices are inactive. 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. This allows lines be directly matched to sub-sets of the model. Indirect matching of lines will be described later. Matched lines are called *fragments*. In the example, the set of fragments (27%) found following the initialization step is $\{ab1, ef1, ei, jk1, kl, lm, oq\}$ for which each fragment has 3 features or more. These are indicated by arrows in Figure 3.

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 having poor information.

In the next Section we show how one can find robust

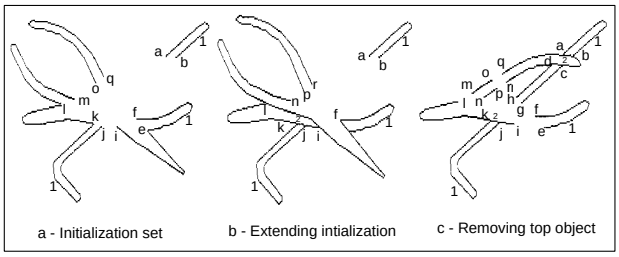


Figure 4: Steps in extending the initialization set to top object and fragments

initial matching hypotheses which result from carrying out gross-to-fine matching for the initial set of fragments only.

3.2 Spatial and shape matching

Each matching ($<f_x, f_m>$) from a scene feature f_x to a model feature f_m represents a hypothesis that must verified. 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 the model we carry out *spatial matching* which 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 that of matched 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 attached to previous feature $f_{x,i}$. The above position and orientation are compared to those of the matched features ($f_{m,i+1}$ with respect to $f_{m,i}$) with the objective of validating or invalidating the ordered matching $<f_{x,i}, f_{x,i+1}, O_m>$ based on matching pair $<f_{x,i}, O_m>$ and $<f_{x,i+1}, O_m>$.

The *shape matching* compares contour shapes after refering to spatially matched features in scene and model. This enables further pruning and consolidation of the subset of intial hypotheses. Shape matching consists of evaluating the minimum possible area difference between scene fragment $A_x(s)$ and its matched fragment $A_m(s)$ versus all possible vertical shift operations. The evaluation is carried out in the corresponding fine angle-length model. The distance function is defined by $d_a(m,x) = \Sigma_{i=1}^{N}(A_x(i) - A_m(i) + a)^2 s_i$, where N is the least number of length intervals in which both $A_x(s)$ and $A_m(s)$ are constant. $d_a(m,x)$ is a convex function [13] of the vertical shift parameter a that would vertically translate $A_x(s)$ in order to yield the least value of $d_a(m,x)$. The total length is being $S = \Sigma_{i=1}^{N} s_i$, the minimum value of $d(m,x)$ that minimizes a quadratic error is then:

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

In summary, spatial and shape matching enable powerful pruning of hypothesized models. The result is a set of *robust hypotheses* that will be extended in next Sub-Section through inter-fragment matching and predictive matching which enables moving forward and reaching a global interpretation.

3.3 Selective processing

An active vertex has at least one fragment and a number of lines. 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 angle-length plan and comparing (g, l) to the models that match 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. The pairing $<ei, ef2, il>$, $<lk, kj2>$, $<lm, il, mn>$, 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 like (n, p, r). 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). 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*. 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.

3.4 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. For example, fragment ef_1 (Figure 4-c) could not be matched to any neighboring contours at vertices e and f.

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. We 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. 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, respec-

tively. 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 matching the distance between every pair of points (x, y) in the scene to the corresponding distance in the model.

3.5 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.

Examining 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. The question is how to efficiently search for a scene line that can be present in many matched models. For this we use: 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 is the nearest unvisited vertex to u. The vector is reported with respect to edge u in each of model 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.

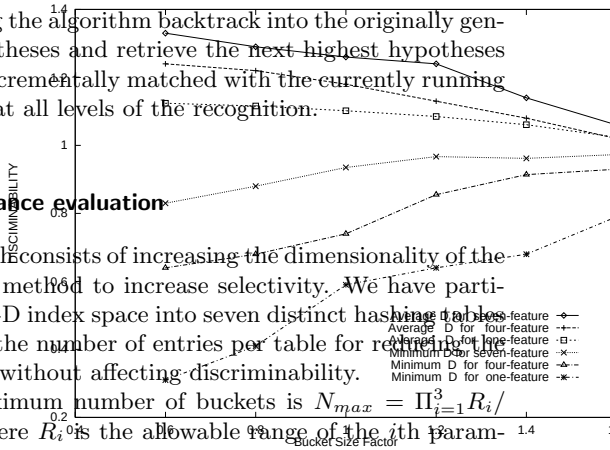
Predictive matching (figure 4-c) allows fragments kl to be matched to line mo, mo is matched to qa, and so on. This results in matching the chains $(kl, mo, qa, ab2, bc, cd, dr, pn)$, $(jk1, hd)$, and $(jk1, gc)$ that each contains at least one initialization fragment.

3.6 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.

In some cases the originally generated hypotheses of some fragment are not 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. To avoid this blocking the algorithm backtrack into the originally generated hypotheses and retrieve the next highest hypotheses which are incrementally matched with the currently running hypotheses at all levels of the recognition.

Figure 5: Storage size and recognition time versus bucket size



4 Performance evaluation

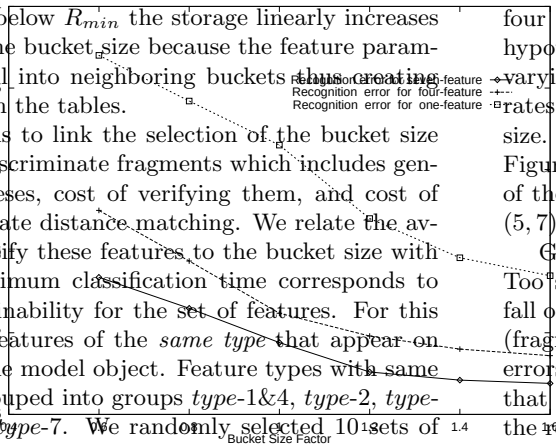
Our approach consists of increasing the dimensionality of the indices as a method to increase selectivity. We have partitioned the 3-D index space into seven distinct hashing tables to decrease the number of entries per table for retrieving the storage size without affecting discriminability.

The maximum number of buckets is $N_{max} = \Pi_{i=1}^{3} R_i / \Pi_{i=1}^{3} 2\epsilon_i$, where R_i is the allowable range of the ith parameter and ϵ_i is the overall variance. 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.

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 6 shows the plot of the storage size and the normalized recognition time versus the bucket size. The storage size decreases versus increase in the bucket size around reference R_{min} in the case of 100-object model. The storage is likely to be constant when bucket size exceeds R_{min} because of non-uniform parameter distribution. When

Figure 6: Percentage of errors in recognizing sets of 4 features



the bucket size is below R_{min} the storage linearly increases with decrease in the bucket size because the feature parameters randomly fall into neighboring buckets thus creating duplicate entries in the tables.

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. We relate the average time to classify these features to the bucket size with the idea that minimum classification time corresponds to maximum discriminability for the set of features. For this we consider four features of the *same type* that appear on the contour of some model object. Feature types with same complexity are grouped into groups *type*-1&4, *type*-2, *type*-3&6, *type*-5, and *type*-7. We randomly selected 10 sets of these features, each set is taken from one model, and plotted on Figure 6 their average recognition time versus the storage factor $st(\alpha)$. The increase in the recognition time (fine-level) for $\alpha > 1.4$ is due to increase in the number of hypotheses that result from selecting bucket size larger than the allowable range of its parameters. On the other hand, selecting excessively small backets (left part of Figure 6) may also increase the recognition time due to bucket fragmentation and redundent processing.

Type-2 has two straight segments separated by some shift used as searching key which explain why the recognition time is linear function of model size (Figure 6). *type*-1&4 and *type*-3&6 have one straight and one curved segments and differ only by the presence or not of the shift separator. Their recognition time slightly favors a specific bucket size. Finally, *type*-5 and *type*-7 have two curving factors and differ only by the shift separator. Here, the recognition time is very sensitive to the bucket size. Feature-based search involves a mixing of feature types, therefore, we need to find a bucket size that is globally suitable. After examining the average recognition time for feature types we decided that the best discrimination corresponds to $1 \leq \alpha \leq 1.75$. Our final setting was $S(\alpha) = 1.4S_{min}$.

The discrimination associated to recognition of whole objects used in [4] 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. In our approach we avoid excessively reducing the bucket size to gain selectivity but instead the bucket size is selected to minimize the recognition time of a sufficiently large number of features. This exposes the recognition system to a finer-level recognition which is more relevant for partially occluded scene.

4.1 Effects of recognition errors

Now we evaluate the recognition errors for each type of features as function of bucket size. For this the storage factor α was given five values starting with 0.75 with a step of 0.5. For each value of α we re-build our hashing tables using data collected from 130 model objects. To study the error rate for each feature type, we select model objects so that each has at least one set of four features of the same type. Each selected object is set with random orientation and its four features (same type) are used to generate the initial hypotheses needed to carry out the recognition. Thus, by varying the type of selected features we can study the error rates of each type as well as relating the error to the bucket size. The error rates for each type of features are plotted on Figure 7. Errors were different depending on the complexity of the features. For this we gouped the features into types $(5, 7)$, $(1, 3, 4, 6)$, and (2).

Generally the errors decrease with increasing bucket size. Too small buckets increase the errors because indices can fall outside the useful range which requires duplicate entries (fragmentation) to reduce errors. Large buckets reduce the errors but also increase the number of initial hypotheses that must be processed. This increases the dependency of the recognition time over the size of the models. Therefore, the bucket size must be engineered with respect to error rates, storage size, and recognition time. Intuitively one is to adopt a coarse bucket quantization but traditional 1-D hashing schemes do not behave well in the presence of uncertainties, digitization noise, and saturation. By categorizing our features into seven types we expanded the indexing mechanism beyond the 1-D table to 2-D and 3-D hashing schemes. This approach lead to buckets having *less denser population* than 1-D hashing which enables the use of coarse buckets to reduce errors while keeping reasonable the number of initial hypotheses.

4.2 Effects of the model size

Here we study the recognition time as function of model size. A model for n-object is denoted by M_n and the studied instances of n are 10, 40, 70, 100, and 130. The bucket size was set as $S(\alpha) = 1.4S_{min}$. The recognition algorithm is run under each of the model settings for recognizing the scene shown on Figure 3.

Table 1 shows the total number of hypotheses generated and the ranking for all the three scene objects versus the size of the model. For example, object-2 received a ranking

Hypotheses pruning versus model (M) size

M	hypo.	Rank-1	Rank-2	Rank-3	Time	% Δ
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

of 6.2% under M_{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 model size, the algorithm spends a small fraction of the recognition time on processing of these hypotheses. 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 M_{10}. The recognition time is likely to be independent from the number of hypotheses originally generated.

5 Conclusion

We presented a coarse-to-fine recognition algorithm that selectively processes robust initial hypotheses which are expanded in the recognition course through predictive matching and other neighborhood relational operators. This avoids processing of a large number of initial hypotheses and allows pruning large portion of inconsistent hypotheses. Traditionally, coarse buckets lacks selectivity due to their dense population and saturation effects. To maintain high selectivity we decided to split our indexing scheme into seven types (features) and expanded its dimentionality to 3. Evaluation shows that the recognition time favors a specific bucket size. However, selecting large buckets was found to lower the recognition errors but at the cost of increasing the recognition time and dependence over the size of the model. Evaluation of 100-object model shows that the recognition time is nearly independent from the number of hypotheses originally generated.

6 Acknowledgments

The authors would like to acknowledge support for attending this Conference from the Research Committee at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia.

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] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] H. Freeman and L. Davis. A coner-finding algorithm for chain-coded curves. *IEEE Trans. on Computers*, pages 297–303, 1977.

[12] 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.

[13] 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.