A Robust Gross-to-Fine Pattern Recognition System

Mayez Al-Mouhamed [*]

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 from scenes with partial occluding. In order to carry out an early pruning of a large portion of the models, hypotheses are only generated for a sub-set of contours with enough discriminative information. Poor scene contours are used later in validating or invalidating a relatively small set of hypotheses. Since hypotheses are selectively verified, blocking is avoided by extending current matching through pairing of hypotheses, predictive matching, and retrieving the next weighted hypotheses. This avoids the processing of a large number of initial hypotheses. Our evaluation shows that a high recognition error results from the use of too small a bucket size because the indices may fall at random, producing non-repeatable results. We use a multi-dimensional hashing scheme with space separation between dense parameter areas to create additional hashing tables. The robustness of the recognition is based on engineering a coarse bucket size to the best tolerance with respect to various sources of noise. Partially occluded scenes having 3 objects can be recognized with a success rate of 84%. The results are reproducible against changes in scale, rotation, and translation. Due to the selection of robust initial hypotheses and the structure of the selective matching system, the processing time essentially depends on scene complexity with a marginal dependence on database size.

Keywords: Hashing, heuristic-search, partial occluding, robust segmentation, shape matching

1 Introduction

Model-based recognition systems [1, 2, 3] are capable of detecting similarities between a scene object and a few model objects without scanning the entire database model. Only a minimal portion of the database is examined while finding a small set of potentially matched objects. The strategy is to use an intermediate representation for capturing the structural and functional similarities of the contours by using a set of robust descriptors or features. The recognition algorithm uses the features as indexing keys in an efficient

[*]Computer Engineering Department, College of Computer Science and Engineering, King Fahd University of Petroleum and Minerals, Box 787, Dhahran 31261, Saudi Arabia. Fax 966-38603059, Email: mayez@ccse.kfupm.edu.sa

hashing look-up process [4, 5] in an attempt to reduce the complexity of the search space. Finally, histogramming of the partial matching enables global pattern matching and classification.

The features [2, 1, 6, 7] must be discriminative, stable, local, and small enough to fit on contour fragments. Global features are inadequate for partially occluded scenes. The features [4, 8] are used as searching keys in some quick indexing and hashing schemes. To find stable partitioning criteria, curved contours suggest the use of sharp convexities, deep concavities, or straight segments as reference points [9] in finding the boundaries between different geometric features. Curvature maxima have also been used [10] as segmentation points and straight lines as primitives. Decomposing smoothed contours at extremes of negative curvature has also been investigated in [11]. The detection of significant changes in curvature [12] has been applied for encoding geometrical signatures such as smooth joins, corner cranks, etc. Decomposing contours by using a constant curvature criterion was proposed by Wuescher et al. [13] and used in other studies [14].

Kalving et al. [15] used a hashing descriptor derived from the relationships between lengths and the relative orientation of contour segments. Knoll and Jain [16] proposed a model organization based on common features to index into the model by recognizing features and to carry out a further search to narrow the object class down to the correct interpretation. Califano and Mohan [7] proposed the use of larger indices as well as redundancy in the generation of indices to maintain a relatively coarse bucket quantization without degrading selectivity. Grimson [6] treats all the available features equally in generating hypotheses on possible matches. This results in a tree-matching structure that is scanned by using a depth-first search. The search over the current sub-tree is abandoned when sufficient inconsistent evidence is 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. Indexing by using a tree grid data structure [17] reduces the storage size and preserves the spatial ordering which enables efficient retrieval of nearby entries. The perspective alignment [18] is one method of solving the back projection which enables computing the 3-D position and orientation based on 2-D image features. Indexing of 3-D objects from single 2-D images may also be implemented by using the kd-trees as data structure in a standard hash table [19].

Our objective is to engineer the model representation and search procedure so that the recognition time would mainly depend on the scene complexity without explicit dependence on the database size. The generation of hypotheses follows a different approach 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 words, the model and the algorithm are to be designed so that the recognition spends a small fraction of time in processing the global database while keeping the rate of correct classification as high as possible.

To provide efficient pruning of inconsistent hypotheses, we propose a shape matching metric for comparing whole contours as well as fragments of contours. Our approach is intended to avoid the brute force processing of a large number of hypotheses 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 discriminative power of the recognition is carried out by relating the bucket size of the model parameters to a metric of discriminability.

The proposed 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 avoid early processing of poor contours because the early generation of a large number of hypotheses would necessarily increase the dependency over the size of the model. Poor contours are used in subsequent phases to validate or invalidate a relatively small set of robust hypotheses. We selectively process the most discriminative hypotheses which are further validated through *spatial* and *shape matching*. Selective recognition provides efficient pruning of a large number of inconsistent hypotheses. Predictive matching is used to extend the recognition in the neighborhood of previously matched contours. Thus the processing scheme is capable of effectively carrying out model pruning.

We experimentally study the recognition error by tuning the bucket size, which is critical to the efficiency and robustness of the hashing scheme and the whole recognition system. Scenes are studied against changes in scale, rotation, and translation. To improve the robusteness of the recognition, we adjust the bucket size to a level that constrains the indices from falling at random. We propose a multidimensional indexing scheme with a space separation between dense parameter areas as one solution to enable the adoption of a coarse bucket quantization. We study the performance of partially occluded scenes and determine some statistical evaluation linking the ability to discriminate shapes to the probability of classification of errors. We also study the dependency between the recognition time, the scene complexity, and the model size.

This paper is organized as follows. Section 2 presents the low-level contour modeling, feature extraction, and the proposed hashing scheme. Section 3 presents each component of the recognition system and explains its operations by using a complete example. Section 4 presents the evaluation of the proposed scheme. In Section 5 we conclude this work.

2 Building an intermediate model

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 the extraction of the shape of object by detecting the presence of an edge at some pixel. For this an approximation 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 sensitivity to noise, the *sobel operator* is used for averaging the gradient over a larger pixel neighborhood. The gradient magnitude is used as the basis of edge detection [20].

The contour is encoded by using the *pixel direction coding* [21] which consists of encoding each border pixel by its direction with respect to the previous border pixel. For this purpose 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 its starting pixel.

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

Each segment is represented by its length and angle with respect to the previous segment. Formally, the kth segment t_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 of t_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 t_{k-1} and t_k is evaluated as the exterior angle which is defined by:

$$\theta(s_k) = \cos^{-1}((\Delta x_{k-1} \times \Delta x_k + \Delta y_{k-1} \times \Delta y_k)/(s_{k-1} \times s_k)) \quad (1)$$

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 the building of a fine angle-length model of contour, denoted by $F = \{(\theta(s_k), s_k)\}$, which consists of an ordered set of segment lengths s_k and their geometric angles $\theta(s_k)$. 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 angle axis is used to represent the accumulation of exterior angles (Eq. 1). 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 segments. Note that a long straight segment of contour is represented by a horizontal straight segment in the angle-length graph. A sequence of small segments that corresponds to a constantly curved contour can then be represented by one coarse segment with a constant slope.

2.1 Coarse segmentation

The fine angle-length model is not efficient enough to enable direct extraction of geometric features from fine-grain segments. Simple features may occur in many models which make the search inefficient. Overly complex features have two drawbacks: 1) they cannot be observed from partial contours, and 2) they lead to a linear search across the model. The features should contain enough discriminatory information to provide efficient and accurate indexing of candidate models. 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 a 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 t_i and its neighbor t_{i-1}. Segment t_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 value of s_i.

Segmenting of the fine model consists of clustering all neighboring 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 two phases which are: (1) the primary segmentation and (2) the contour improvement rules which operate on the fine model in the angle-length plan as shown on Figure 2. The first step consists of selecting breakpoints among the fine segments corresponding to a strong change in direction. The above breakpoints are temporarily linked by straight *coarse segments* in the angle-length plan. 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 the 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 has the largest distance from the current coarse segment is added as a new breakpoint when the above distance is sufficiently large compared to segment lengths as shown in step 2 on Figure 2-(a). This results in the coarse model C_c having 10 segments, as shown in Figure 1. The stability of the breakpoints needs to be improved because of the effects of digitization, rotation, shadows, and lighting.

We use a set of contour improvement rules (Figure 2-(b)) to attempt correcting those situations where a coarse polygonal segment may have been fragmented during initial segmentation because of noise in the fine model. In this case, neighboring segments having almost the same orientation angles are fused into one single coarse segment that must have a bounded orientation error compared to the originally fragmented fine segments. On the other hand, a fragmented corner contour formed by three initial neighboring segments t_{k-1}, t_k, and t_{k+1} can be represented by a *two-segment corner* if the following three conditions are met. First, we have to make sure that t_{k-1} and t_{k+1} are not co-linear which causes some loss of accuracy if the above segments are modified by extending t_{k-1} and t_{k+1} up to their intersection. Second, the length of t_{k-1} and t_{k+1} must greatly exceed that of t_k in order to avoid confusing a corner with a true fragmented contour. Third, the relative orientations of t_{k-1} and t_{k+1} with respect to t_k must be of the same 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 t_{k-1}, t_k, and t_{k+1} is modified by cancelling t_k and extending t_{k-1} and t_{k+1} to their intersection.

The *coarse model* C is defined by the resulting collection of segments in which each segment t_k is characterized by three parameters which are: 1) the exterior angle $\theta_{ext}(t_k)$, 2) the total angular change $\theta_{chg}(t_k)$, and 3) the segment length s_k. The exterior angle $\theta_{ext}(t_k)$ is defined as the angle between the tangent to t_{k-1} and the tangent to t_k at their intersection point. It also denotes the turning angle from t_{k-1} or its tangent if t_{k-1} is curved and t_k or its tangent if t_k is curved. The total angular change $\theta_{chg}(t_k)$ is the turning angle from the tangent to t_k, at its start point, to the tangent to t_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, which is:

$$C = \{t_k = (\theta_{ext}(t_k), \theta_{chg}(t_k), s_k)\} \tag{2}$$

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 the 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 between models and be able to limit potential matching to a sub-set of the model.

Figure 3 shows seven possible configurations of two successive segments from the coarse model (Eq. 2) that are linked with each other. Each vertex v of the coarse model which links up two successive segments t_1 and t_2 can be represented by the tuple:

$$(s_1/(s_1+s_2), \theta_{ext}, \theta_{chg1}, \theta_{chg2}) \tag{3}$$

where $s_1/(s_1+s_2)$ is the relative length of segment t_1 with respect to its neighbor t_2, θ_{ext} is the exterior angle between t_1 and t_2, and θ_{chg1} and θ_{chg2} are the total angular changes of segments t_1 and t_2, respectively. The tuple defined in Eq. 3 can be considered as a *geometric feature* that is invariant against scale, rotation, and translation. A feature can be constructed based on the knowledge of two joining coarse segments which can be formed by all possible combinations of straight (s) or curved (c) coarse segments. In addition we may distinguish features having an exterior angle between their two segments and those which do not. This has the effect of isolating the source of noise introduced by the measurement of θ_{ext} which may improve the robustness of the recognition. Thus we may increase the discriminability of a recognition system if each feature can be further classified by using seven data types (1 to 7) as shown in Figure 3. For example, feature $(sc0, s_1/s_1+s_2, \theta_{chg})$ is formed by a straight segment t_1 (s_1 as length), a curved segment t_2 (s_2 as length), a nil θ_{ext}, and an arbitrary angular change θ_{chg} for t_2.

Figure 3 shows the retained seven features, their indexing parameters, and their representations in the angle-length plan below each type. The main effect of breakpoint selection is the introduction of noise in the values of the exterior angle. However, the noise caused by the detection of beakpoints has less effect on types $sc0$, $cs0$, and $cc0$ for which the exterior angle is nil. For the other types, we strongly reduce the effect of breakpoint selection by creating features only when there is enough *evidence* and *confidence* in the presence of distinct segments. In other words, a feature is created only when the two adjacent segments have distinctive curving factors or there is a noticeable exterior angle. Keeping as many different features as possible improves the selectivity of the model and recognition system because of the implied decrease in the number of entries in the corresponding buckets. Compared to previous proposals [20] this organization is expected to improve the discriminability and tolerance to noise as well as improving the robustness of the indexing of the features.

2.3 Model organization

There are seven distinct types of feature and each type is represented by an 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 represented by a pointer value (f_v) that results from concatenation of its parameters $\rho = s_1/s_1+s_2$, θ_{ext}, and θ_{chg}. The generated discrete index addresses a bucket in a look-up table ($Inx\text{-}type(f_v)$) to which we add the entry $f_v = (\rho, \theta_{ext}, \theta_{chg})$, whose dimensionality depends on the feature type, with an object label O from which the feature was extracted. 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 ρ, θ_{ext}, and θ_{chg}. The upper bound on tolerance for each parameter is experimentally found. Indexing allows definitive mapping from an input feature into a group of model objects that are associated with the corresponding range. Each model object that falls in the range of a given feature type has at least one feature of that type whose parameters fall within that range. Therefore the selection of the bucket size has a fundamental effect on the performance and robustness of the recognition system.

3 The recognition system

Our approach consists of initially selecting a sub-set of scene contours from those having the largest number of features among all scene contours. In other words, poor contours are not processed in the early stages of our recognition approach but used later. Pruning and verification of the initially generated hypotheses is done through the application of a low cost *spatial matching* which compares the relative positioning of features in the scene to that of their matched features in the model. Further refinement of the previously verified hypotheses consists of carrying 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 details 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 *v-link.* At least three v-links intersect at each vertex in the case of connected contours. A collection of v-links that link up an arbitrary number of vertices may or may not belong to the contour of the same object. The problem is to cluster the v-links into the subsets that are individually matched with enough confidence to some model objects.

We start by sorting the v-links in a decreasing order of the number of segments according to their coarse model. The v-links having the largest number of segments have richer discriminative information than the others and are used in indexing the models to generate hypotheses on possible matching. In other words, we retrieve a sufficient number of candidate v-links from a list sorted according to the principle of the *largest number of features first.* This represents a significant fraction of the total number of scene v-links and available features. The idea of selecting rich contours at the start of the recognition algorithm is summarized in Figure 4.

Figure 5 shows an example of partial overlapping between three objects. There are 18 vertices labeled as $(a, b, \ldots, q)$ and 26 v-links labeled by the pair of vertices they connect.

For example, vertices a and b directly connect two contours that are labeled $ab1$ and $ab2$.

Initially all vertices are set into a state called "inactive". The features that belong to the initially selected v-links are then used in the indexed search which allows the finding of one or more matches for each selected v-link. This allows v-links to be directly matched to sub-sets of the model. Indirect matching of v-links will be described later. Matched v-links are called *fragments*. Each v-link is matched to a sub-set of models that are sorted in order of decreasing number of matched features. In the example, the set of fragments (27%) is 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 5.

The vertices connected to fragments become active as the fragments may be used to extend the matching to some of their neighboring v-links which are connected to active vertices. In the example, the active vertices are $(a, b, e, f, i, j, k, l, m, o, q)$ (Figure 6-a). These vertices are obtained after removing all inactive vertices and v-links having poor information.

In the next section we show how robust initial matching hypotheses can be found which result from carrying out gross-to-fine matching for the initial set of fragments only.

3.2 Spatial and shape matching

We denote by $(<f_x, f_m>)$ the operation of matching a scene feature f_x to a model feature f_m. Thus the *primary matching* $<f_x, f_m>$ represents a hypothesis that might need to be verified. Assume that 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 O_m 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 purpose the position and orientation of each $f_{x,i+1}$ is evaluated with respect to some frame of reference attached to a 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 the matching pair $<f_{x,i}, f_{m,i}>$ and $<f_{x,i+1}, f_{m,i+1}>$. Operation $<f_{x,i}, f_{x,i+1}, O_m>$ consists of matching the relative position and orientation of $(f_{x,i}, f_{x,i+1})$ in the scene with respect to that of $(f_{m,i}, f_{m,i+1})$ in the model.

The idea of using spatial matching for the pairs $(f_{x,i}, f_{x,i+1})$ and $(f_{m,i}, f_{m,i+1})$ is summarized in the phase of primary matching described in Figure 4. 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 the vector $f_{m,i+1}$ that is observed with respect to $f_{m,i}$. Spatial matching is 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 the relative positioning of the scene features differ from their corresponding model features. A global measure of the relative positioning error between $f_{x,1}, \ldots, f_{x,n}$ and $f_{m,1}, \ldots, f_{m,n}$ can be defined by adding up the squares of the error vectors:

$$\epsilon_{x,m} = \Sigma_{i=1}^{i=n-1} \epsilon(i)^2 \tag{4}$$

In other words, $\epsilon_{x,m}$ (Eq. 4) is the spatial matching error for $<A_x, O_m>$. The error is normalized and used in evaluating the matching confidence.

The advantage of this approach lies in 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 spatial matching. First, only those contours having sufficient discriminative information participate in the original spatial 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 those contours having sufficient connected features. The process of spatial matching allows a low cost verification of the most probable hypotheses that are directly produced by the indexed search. The low cost of spatial matching and the organization of the storage to this effect is one important issue for reducing the dependency of the recognition time over the size of the database. Verifying the most probable hypotheses by using low cost spatial matching is an essential refinement step prior to applying the more costly shape matching which is described below.

The *shape matching* compares contour shapes after referring to spatially matched features in scene and model. This enables further pruning and consolidation of the subset of initial hypotheses. Shape matching consists of evaluating the minimum possible area difference between scene fragment $A_x(s)$ and its matched fragment $A_m(s)$ against all possible vertical shift operations.

Functions $A_x(s)$ and $A_m(s)$ are the polar angles as a function of the contour length s. Figure 7-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 L and $A_m(s)$ is of equal length but with N segments. In Figure 7-a $k = 6$ and $N = 5$. The interval $[0, L]$ is divided into a number of sub-intervals so that in each sub-interval i functions $A_x(s)$ and $A_m(s)$ are common over the length $L(s_i)$ of that sub-interval. The total length satisfies:

$$L = \Sigma_{i=1}^{N} L(s_i) \tag{5}$$

The effect of the scene instance on $A_x(s)$ appears as a vertical shift in the angle-length plan when compared to the model instance. The shift is due to the original orientation of the objects with respect to the horizontal.

The shape matching is a distance that measures the minimum possible area difference between $A_x(s)$ and $A_m(s)$ against all possible vertical shift operations. To increase the accuracy, the evaluation is carried out in the corresponding fine angle-length model. The distance function is defined by:

$$d_a(m, x) = \Sigma_{i=1}^{N} (A_x(s_i) - A_m(s_i) + a)^2 L(s_i) \tag{6}$$

where N is the least number of length intervals in which both $A_x(s)$ and $A_m(s)$ are common and constant and a is the value of a constant that vertically shifts $A_x(s)$ with respect to $A_m(s)$. The distance $d_a(m, x)$ (eq. 6) is a convex function [22] 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 (Eq. 5) is $L = \Sigma_{i=1}^{N} L(s_i)$; the minimum value of $d(m,x)$ that minimizes a quadratic error is then:

$$d(m,x) = \Sigma_{i=1}^{N} [A_x(s_i) - A_m(s_i)]^2 L(s_i) - \frac{1}{L} [\Sigma_{i=1}^{N} (A_x(s_i) - A_m(s_i)) L(s_i)]^2 \quad (7)$$

To find the least possible value of $d(m,x)$ (Eq. 7) the matched features of $A_x(s)$ and $A_m(s)$ are aligned (horizontal shift) prior to finding the appropriate horizontal shift value of a given setting. The idea of evaluating the shape matching for a scene contour with its features and a matched model contour with its matched features is summarized in Figure 4. The normalized distance $d(m,x)/L$ allows the smallest area difference (shape distance) between two fragments A_x and A_m of equal length s to be found that is determined by the length of the scene fragment A_x.

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 the next subsection via inter-fragment matching and predictive matching which facilitates progress in reaching a global interpretation.

3.3 Selective processing

An active vertex has at least one fragment and a number of v-links. Each fragment A of some active vertex is paired with a v-link l for possible matching. This consists of appending the v-link to A in the angle-length plan and comparing the shape (A, l) to the models that match A. In the examples of Figures 5 and 6-a, the fragments ab_1, ef_1, and jk_1 could not be matched to their neighboring v-links. The pairing $<ei, ef2, il>$, $<lk, kj2>$, $<lm, il, mn>$, and $<oq, op, qr>$ were successful and the newly matched v-links became fragments and their connected vertices were then considered as newly active vertices. By transitivity, the matched chain of fragments and v-links are extended such as in the case of the chain (fe, ei, il, lm, mn) as more vertices become newly active such as (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 6-b where most of the contours that belong to the top object are discovered. The newly active vertices (h, g) enable 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). Removal of the top object leaves all the v-links and fragments that are shown in Figure 6-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 v-links 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 in Figure 6-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 v-link.

3.4 Pairing of hypotheses

Pairing of hypotheses applies to active vertices that have fragments cannot be matched to other contours of the same vertices. Pairing of hypotheses evaluates the potential matching of distant fragment of contours that have been highly hypothesized to the same models. For example, fragment ef_1 (Figure 6-c) cannot be matched to any neighboring contours at vertices e and f.

Assume two scene fragments A_1 and A_2 that are matched to some contours denoted by A_1^* and A_2^* of the same model. We compare the position and orientation of each pair of features from A_1 and A_2 to those of the matched features from the model. For this purpose we choose two points (x_1, x_2) on A_1 and (y_1, y_2) on A_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 A_1^* and A_2^* that correspond to x_1, x_2, y_1, y_2, respectively. Now the position and orientation of A_2 with respect to A_1 can be matched to that of A_2^* with respect to A_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 recognition because all contours having significant discriminative information have already been hypothesized and there are still some contours (v-links) that must be included in the global interpretation.

Examining the *geometric relationships* between a fragment A at vertex u ($<A, O_m>$) and a v-link l at vertex v allows an extension of the matching process to l, i.e. whether $<l, O_m>$ holds or not. The question is how to search efficiently for a scene v-link that can be present in many matched models. To do this we use: 1) vector matching, 2) orientation matching, and 3) shape matching. It is possible to backtrack at each step and terminate 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 model to which A is matched. If the reported vector points to a contour point w of the model, then the relative orientation (tangential) of l with respect to A in the scene is compared to the relative orientation of w with respect to A in the model. The shape distance matching is attempted only when vector and orientation matching succeed. Otherwise, the next model to which A is matched to is taken and the previous steps are repeated.

Predictive matching (figure 6-c) allows the fragments kl to be matched to the v-link 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 $(jk1, gc)$ each of which contains at least one initialization fragment.

3.6 Interpretation

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

In some cases the originally generated hypotheses of a 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 a relatively very low vote. To avoid this blocking, the algorithm backtracks into the originally generated hypotheses and retrieves the next highest hypotheses which are incrementally matched with the currently running hypotheses at all levels of the recognition.

4 Performance evaluation

Our approach consists of increasing the dimensionality of the indices as a way of increasing selectivity. We use 2, 3, and 4 dimensional indices with a space partitioning of the index space into seven distinct hashing tables to decrease the number of entries per table and reduce the storage size without affecting discriminability. Our objective is to avoid the traditional problems of 1-D hashing processes such as limited selectivity, excessive accumulation of vote in each bucket, limited number of useful buckets, and extreme sensitivity to all sources of digitization noise. Unlike other approaches [7], our higher-level geometric features allowed us to avoid starting the recognition with a large number of redundant features in an attempt to increase selectivity. Given a set of features and their hashing scheme, the main issue is how to select a coarse enough bucket size to guarantee robustness to noise without scarifying the discriminating power of the system. In the following we present the performance of the proposed recognition versus the bucket size.

For a given geometric feature, the maximum number of buckets is given by:

$$N_{max} = \Pi_{i=1}^{\eta} R_i / \Pi_{i=1}^{\eta} 2\epsilon_i \tag{8}$$

where η is the dimensionality of the feature parameters, R_i is the allowable range of its ith parameter, and ϵ_i is the overall variance. For the features shown on Figure 3, the values of η are 2 , 3, and 4 for the feature types (ss,sc0, and cs0), (sc, cs, and cc0), and (cc), respectively. As an example, the maximum number of buckets of feature $(sc, s_1/s_1+s_2, \theta_{ext}, \theta_{chg})$ is $N_{max} = R_\rho R_{\theta_{ext}} R_{\theta_{chg}} / 8\epsilon_\rho \epsilon_{\theta_{ext}} \epsilon_{\theta_{chg}}$ (Eq. 8), where $\rho = s_1/s_1+s_2$.

Since parameters ϵ_ρ, $\epsilon_{\theta_{ext}}$, and $\epsilon_{\theta_{chg}}$ are global variances it is important to optimize the bucket size by directly relating it to the probability of correct classification. For this we heuristically searched for the best possible bucket size by stepping the size around the value of $R_{min} = \Pi_{i=1}^{\eta} 2\epsilon_i$ which is the smallest bucket size, after estimating the values of ϵ_i based on known thresholding and repetitive acquisitions of the features by varying their position and orientation. The current bucket size is given by:

$$R_{\rho,\theta_{ext},\theta_{chg}} = \alpha R_{min} \tag{9}$$

where α is the bucket size factor that is studied here in the range $0.6 \leq \alpha \leq 1.6$ by using steps of 0.2.

The discrimination function ($D(A)$) for a scene shape A is defined as the ratio of the number of votes received for the correct matching of A over the maximum number of votes received for any incorrect shape instance. Our objective is to link the bucket size (eq. 9) to the discrimination function as well as the recognition error and processing time which includes the generation of hypotheses, the cost of verifying them, the cost of carrying out accurate distance matching, etc. For this we selected 25 scenes each consisting of three objects with partial occluding. Each object belongs to a library of 130 formed by mechanical tools and 2-D plastic shapes with different sizes and different shapes. Each object contour has between 20 to 140 fine segments or between 10 to 50 coarse segments. Each of the above 3-object scenes was rotated by using four different angles, digitized (see Section 2), and the resulting 100 scenes were stored into memory. The scene rotation is useful in studying the robustness of the recognition in the presence of various sources of digitization noise.

We examined three hashing schemes denoted by one-feature, four-feature [20], and our proposed seven-feature. For the one-feature hashing scheme there is one single feature $(s_1/s_1 + s_2, \theta_{ext}, \theta_{chg1}, \theta_{chg2})$. In this case, a feature instance formed by two straight segments generates $(s_1/s_1+s_2, \theta_{ext}, 0, 0)$. For the four-feature there are four hashing schemes generated for the four types ss, sc, cs, and cc with their appropriate attributes, where s and c denote a straight segment and a curved segment, respectively. For the seven-feature there are seven hashing schemes generated for the seven types ss, sc, $sc0$, cs, $cs0$, cc, and $cc0$ as defined in the model organization of Section 2. For each of the above three hashing schemes our recognition algorithm was run over the stored 100 scenes. The results are displayed on Figures 8, 9, and 10. Figure 8 shows the average object discriminability D and the minimum value of D as a function of the bucket size for each of the above three hashing schemes. The recognition failed for every $D < 1$.

Increasing selectivity through the selection of a small bucket size, as shown on Figure 8, gives higher values for D $(1.15 \leq D \leq 1.35)$ but this is done to the detriment of the levels of noise that the algorithm can handle which is evidenced by the high probability of recognition error as shown on Figure 9. A small bucket size also corresponds to a low processing time because of the reduced number of hypotheses that fall into the allowable range of its parameters. Furthermore, a robust recognition system must avoid the use of a small bucket size to prevent the indices from falling at random which produces non repeatable results.

Intuitively one adopts a coarse bucket quantization but traditional 1-D hashing schemes do not behave well in the presence of uncertainties, digitization noise, and saturation. The use of a large bucket size for the one-feature hashing process produced the lowest discriminability (Figure 8) with relatively high recognition errors (Figure 9). Furthermore, a coarse bucket quantization with the one-feature hashing is responsible for limiting the index selectivity due to a relatively high degree of storage sharing which produces very low discriminability values. In addition, the high recognition errors are indicators of the excessive sensitivity to various sources of digitization that noise produces. The increase in the processing time for excessively large buckets is due to an increase in the number of hypotheses that result from selecting a bucket size larger than the allowable range of its parameters. Many unrelated models come into the picture and those have to be processed.

Our recognition system relies on expanding the dimensionality of the indexing scheme in space and parameters which enables the use of a coarse bucket quantization to guarantee a good level of robustness to various sources of noise. The selection of a coarse bucket $(1 < \alpha \leq 1.6)$ for the four-feature and seven-feature hashing was rewarded by a sufficiently high level of discrimination and a greater noise tolerance.

The main difference between the seven and four-feature hashing is the space separa-

Hypotheses pruning versus model (M) size

M size	hypotheses	Ranking 1	Ranking 2	Ranking 3	Recog. time	% increase
10	61	16%	14%	25%	24	–
40	258	7.6%	6.2%	10.4%	25.5	6.25%
70	497	4.3%	3.8%	6.7%	26.5	10.4%
100	796	2.8%	2.5%	4.5%	27.5	14.5%
130	1172	2.1%	1.8%	3.25%	29	20.8%

Table 1: Hypothesis generation, ranking, and recognition time

tion between dense parameters areas. By distinguishing between features having a zero and non-zero exterior angle, a significant index expansion was made through the space separation between two dense parameters areas. The increase in the selectivity in the case of the seven-feature hashing was rewarded by a noticeable increase in discriminability, an increase in the rate of correct recognition, and a shorter processing time. Based on the above the bucket size factor was set to $\alpha = 1.4$.

By categorizing our features into seven types we expanded the indexing mechanism beyond the 1-D table to 2-D, 3-D, and 4-D hashing schemes together with the parameter space separation to gain selectivity. This approach leads to buckets having a *less dense population* than 1-D hashing which enables the use of coarse buckets to reduce errors while keeping a reasonable number of initial hypotheses. Thus our approach is based on *extended indexing* and *bucket size engineering.*

4.1 Effects of model size

Here we study the recognition time as a function of model size. A model for n-objects is denoted by M_n and the studied instances of n are 10, 40, 70, 100, and 130. The bucket size was set as $R_{\rho,\theta_{ext},\theta_{chg}} = 1.4 \times R_{min}$. The recognition algorithm (seven-feature) was run for each of the previous 3-object scenes which are similar to the scene shown on Figure 5.

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 for the 3-objects of the typical scene shown on Figure 5. 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 pose a major problem if we do not retain all these hypotheses but only a very small subset which will be further processed. For example, object-2 received a ranking of 6.2% under DB_{40} which indicates that, on average, the number of votes received by each of its features for the correct matching was in the top 6.2% of all the highly voted matches. 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 the processing of these hypotheses because only a small fraction

Paper	Type of features used	Invariance to similarity transformation	Dimention of index space	Type of shape used	Search Linearity $H(M_{100})/H(M_{50})$	Time Linearity $T(M_{100})/T(M_{50})$	Drop in discriminability $1-D(M_{50})/D(M_{10})$	Drop in discriminabilty $1-D(M_{100})$ $D(M_{10})$
Califano & Mohan	Correlation of geometric triangle	Yes	4-D & 7-D	Leaf shapes	-	-	0.4	0.5
Stein & Medioni	Super segment, 6 attributes	Yes	1-D to 6-D	Animal shapes	1.76	1.63	-	-
Al-Mouhamed & Ismail	Curving, length, & corners	No	1-D & 2-D	Polygonal shapes	2.1	1.18	0.58	0.66
Al-Mouhamed	Constant curvature parameters	Yes	1-D & 2-D	Polygonal shapes	2.36	1.07	0.26	0.42

Table 2: Main features of some recognition systems based on multi-dimensional indexing

of these hypotheses are to be verified and the remainder is pruned.

The number 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 pairs of features has a relatively larger overhead than hypothesis generation but its global overhead is moderate because the set of potential matches is small after the hypotheses pruning step. Spatially unmatched hypotheses are further pruned prior to carrying out the accurate distance matching having the highest overhead. The overall recognition time of the three objects is shown on Table 1 together with the percentage increases in recognition time over that obtained for DB_{10}. The recognition time is likely to be independent the number of hypotheses originally generated. The time increases are 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 scene complexity without explicit dependence on database size.

4.2 Comparison to other recognition schemes

In Tucker et al. [23], all scene features are allowed to participate in the generation of hypotheses which are ranked by mutual support. A massively parallel machine was used in the verification of hypotheses. The hypothesis with the highest confidence level is retained. Due to the parallelism used, the recognition time depends mainly on scene complexity with only a minor dependence on model size.

Table 2 compares the main features of our approach to those of three other recognition systems [7, 24, 25] which are based on a multi-dimensional indexing scheme. The first five columns compare the type of features used, the invariance to similarity, the dimension of the index space, and the shapes of objects used. In columns 6 and 7, we list the number of hypotheses generated ($H(M_n)$) and the recognition time ($T(M_n)$) against a change in model complexity (M_n), respectively. Although ($H(M_{100})/H(M_{50})$) is relatively large in our case, due to bucket sharing, the pruning process significantly reduced its effect on the recognition time ($T(M_{100})/T(M_{50})$). In the last two columns we show the drop of the discrimination function $D(M_n)$ (see Section 4) against an increase in model complexity. The larger drop in the discrimination indicates that the recognition decision has a greater dependence on the complexity of the model.

We reduced the dependence of the recognition time on the size of the model by using the following strategy. First, we avoided increasing the selectivity by reducing the granule size of the bucket [13] which has the effect of degrading the discrimination of the whole scheme. The reason for this is that such a reduction occurs at the expense of the levels of noise that the system can handle. This effect has been extensively studied by Califano and Mohan [7] who proposed the adoption of *larger indices* to keep a relatively coarse bucket quantization without reducing the selectivity. Unlike [7, 24, 25], we used non-redundant geometric features.

Second, we avoided generating a large number of hypotheses by selecting initial contours that have rich discriminatory information. Poor fragments are tested later on the validity of current hypotheses.

Third, to prune a large fraction of the models, we used an in-depth first-hypothesis verification approach that progresses from coarse spatial matching of 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 of the scene are rapidly recognized. This facilitates the predictive matching of contour fragments that intersect with top objects.

Fourth, we used a global interpretation approach that leaves the least percentage of uncovered contours. Though a serial processing is used, the fraction of the recognition time that is database-dependent is only a small percentage of the overall processing time (Table 1).

5 Conclusion

We have presented a coarse-to-fine recognition algorithm that selectively processes discriminative initial hypotheses which are expanded through predictive matching and other neighborhood relational operators. This avoids the processing of a large number of initial hypotheses and allows pruning of large portions of inconsistent hypotheses. We have avoided the problems of limited selectivity, excessive accumulation of vote, and extreme sensitivity to noise by selecting a set of high-level multidimensional features. Our evaluation shows that a high recognition error results from the use of an excessively small bucket size because, in this case, the indices may fall randomly, producing non-repeatable results. A robust recognition requires the use of a coarse bucket quantization which is one feasible solution if we adopt a multidimensional hashing scheme, with space separation between dense parameter areas, to create additional hashing tables. The robustness of the recognition is based on engineering a bucket size with the best tolerance to various sources of noise. Partially occluded scenes consisting of 3 objects can be recognized with a probability of 0.84. These results are reproducible against changes in scale, rotation, and translation. Due to the selection of robust initial hypotheses and the structure of the selective matching system, the processing time depends mostly on scene complexity with only a marginal dependence on database size.

6 Acknowledgment

The author acknowledges computing support and conference attendence support from the King Fahd University of Petroleum and Minerals, Dhahran 31261, 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] 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.

[3] S. Tsugawa. Vision-based vehicles in Japan: machine vision systems and driving control systems. *IEEE Trans. on Industrial Electronics*, 41, No 4:389–405, Aug 1994.

[4] 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.

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

[6] 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.

[7] 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.

[8] A. Naiberg and J. Little. A unified recognition and stereo vision system for size assesment of fish. *Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision*, pages 2–9, 1994.

[9] 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.

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

[11] 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.

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

[13] 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.

[14] H. Zghal, D. Strong, and H. ElMaraghy. Matching topographic features in 2-d images for model-based recognitionin manufacturing applications. *The IEEE Candian Conf. on Electrical and Computer Enginrring (CCECE'96)*, pages 262–565, 1996.

[15] 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.

[16] 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.

[17] A. Wallack and J. Canny. Object recognition and localization from scanning beam sensors. *Proceedings of the IEEE Inter. Conference on Computer Vision and Pattern Recognition*, pages 259–266, June 1994.

[18] G. Verghese and J. Tsotsos. The role of feature visibility constraints in perspective alignment. *Proc. of the IEEE Inter. Conf. on Image Processing*, 1:386–389, 1995.

[19] J. Beis and D. Lowe. Indexing without invariants in 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, No 10, October 1999.

[20] M. Al-Mouhamed. An efficient indexing scheme for image storage and recognition. *IEEE Trans. on Industrial Electronics*, 46, No 2:429–439, April 1999.

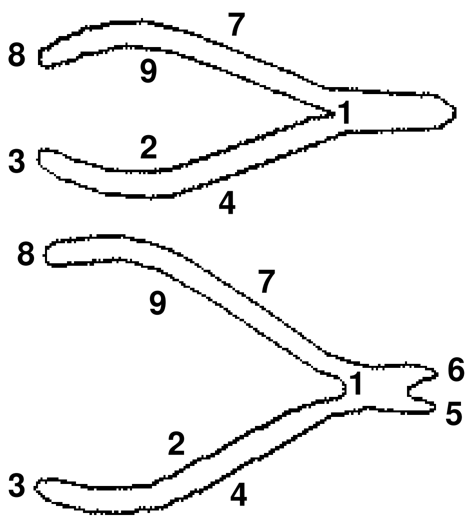
[21] H. Freeman and L. Davis. A coner-finding algorithm for chain-coded curves. *IEEE Trans. on Computers*, pages 297–303, 1977.

[22] 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.

[23] 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.

[24] F. Stein and G. Medioni. Structural indexing: efficient 3-D object recognition. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 14, No 2:125–145, Feb 1992.

[25] Al-Mouhamed M. and Ismail L. A database appraoch to hierarchcal image storage and recognition. *Journal of Engineeing Applications of Artificial Intelligence*, 10, No 2:189–204, 1997.



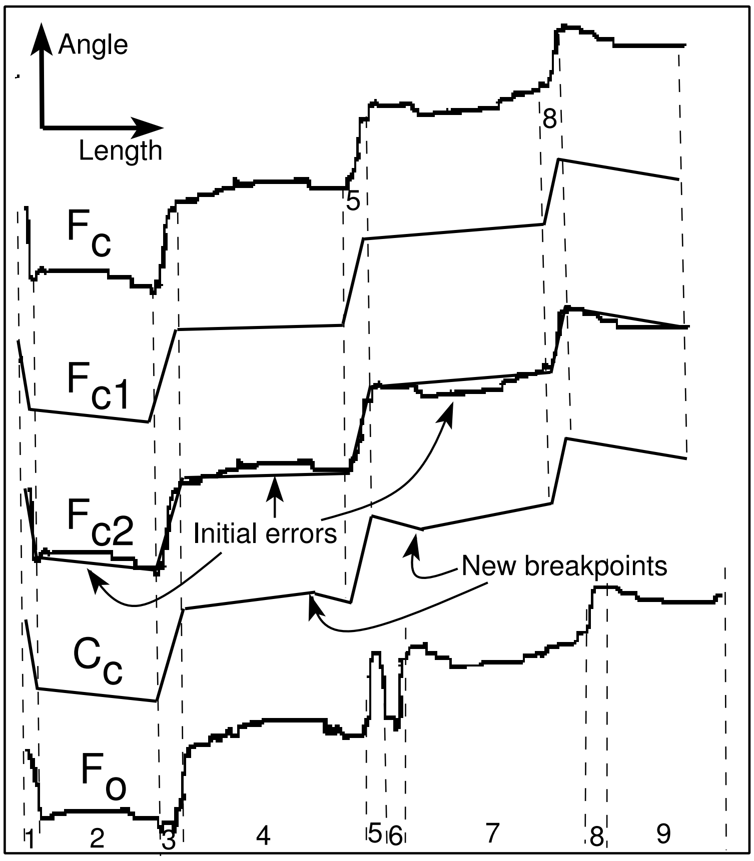


Figure 1: A cutter and its fine and coarse models

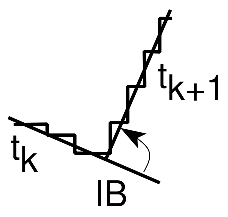
(a)

COARSE SEGMENTATION

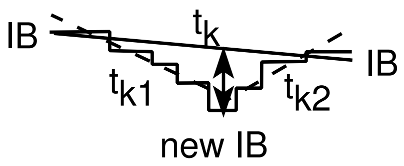
Phase 1: Primary Breakpoints (PBs)

Input: fine contour in the length-angle plan
$F = \{ (O (s_i) , s_i) \}$.

Step 1: Create initial breakpoints (IBs).
Detect strong change in the direction.



Step 2: Create corrective breakpoints.
Detect maximum diatance from a chain of IBs and fine contour.

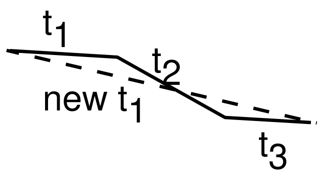


(b)

Phase 2: Improvement Rules

Input: chain of primary beakpoints in the angle-length plan

Rule 1: correcting a fragmented straight segement



Rule 2:

condition 2.1: Segements t_{k-1} and t_{k+1} are not co-linear.

condition 2.2: Each of t_{k-1} and t_{k+1} is longer than t_k.

condition 2.3: Orientation of t_{k-1} and that of t_{k+1} are of the same sign.

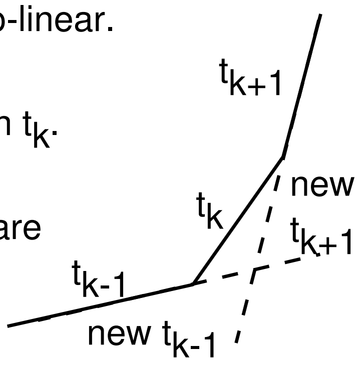


Figure 2: Primary segmentation (a) and improvement rules

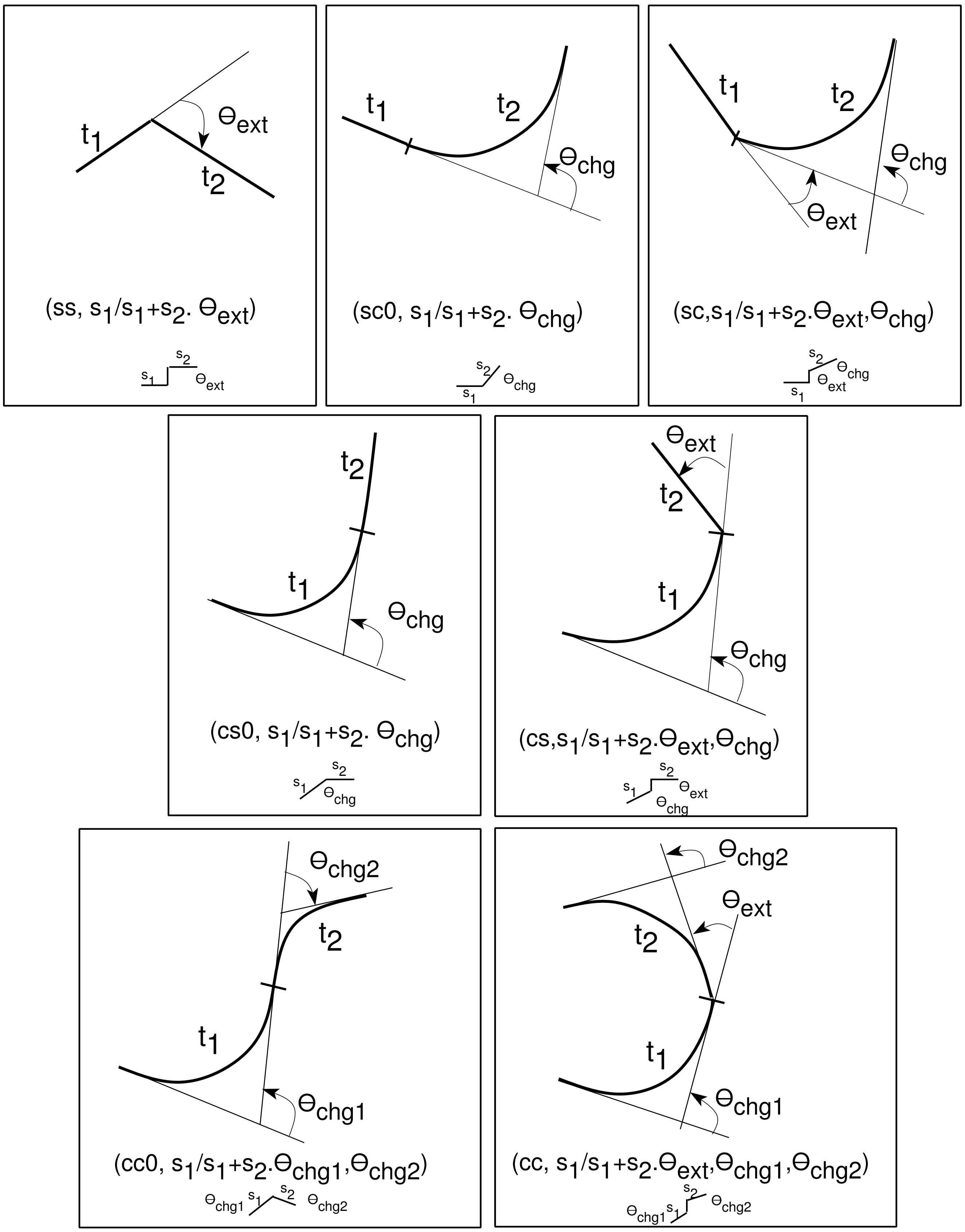


Figure 3: Features and their associated types from 1 to 7

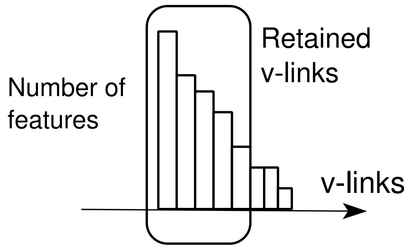
CONTOUR MATCHING

Input:

set of v-links sorted in the order of decreasing number of their scene eatures

Selection of rich contours:

Select a subset of v-links { A_X } such that their aggregate number of features is a farge fraction of the number of scene features.



Primary matching:

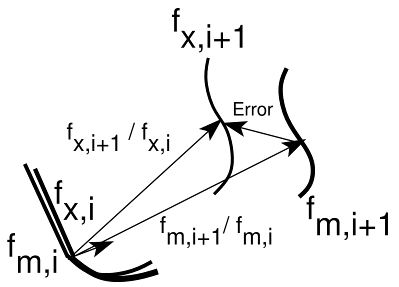
+ Index the features of selected v-links into the model and find all matching {< A_X , O_m>}.
+ For each v-link, find a set M ={O_m} of matched models.
+ For each <A_X,M>, sort M in the order of decreasing number of matched features of A_X.

	$f_{m,i}$	$f_{m,i+1}$	$f_{m,i+2}$
$f_{x,i}$	0.9	0	0.7
$f_{x,i+1}$	0.1	0.6	0.2
$f_{x,i+2}$	0.8	0.1	0.7

Spatial matching:

For every pair of features $f_{x,i}$ and $f_{x,j}$ of some v-link A_X that are matched to features $f_{m,i}$ and $f_{m,j}$ of model O_m, do:

+ evaluate the confidence of matching the relative position $f_{x,i} / f_{x,j}$ in the scene to $f_{m,i} / f_{m,j}$ in the model O_m;
+ sort M in the order of decreasing matching confidence.



Shape matching:

For each matching < A_X, O_m>, do:

+ find best horizontal shift of A_X w.r.t. model O_m that aligns the largest number of features;
+ evaluate the minimum value of the shape distance d(A_X m) by optimizing the vertical shift;
+ for each <A_X, M>, sort M in the order of decreasing shape distance.

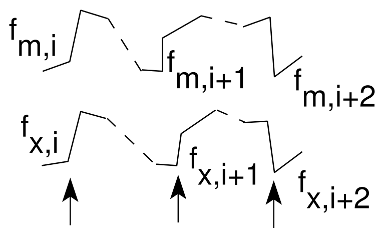


Figure 4: Contour matching functions

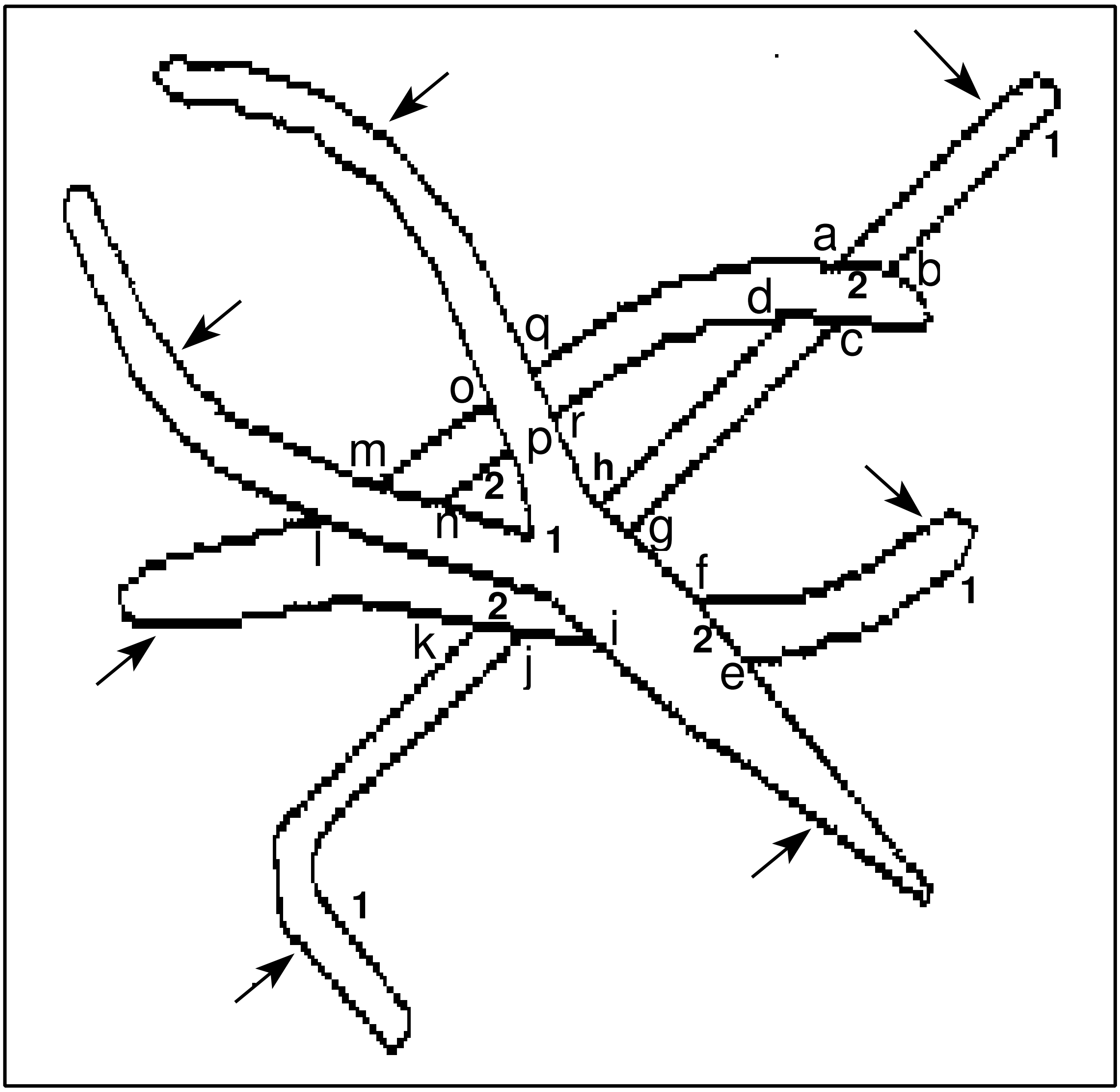


Figure 5: Partial occluding among 3 objects

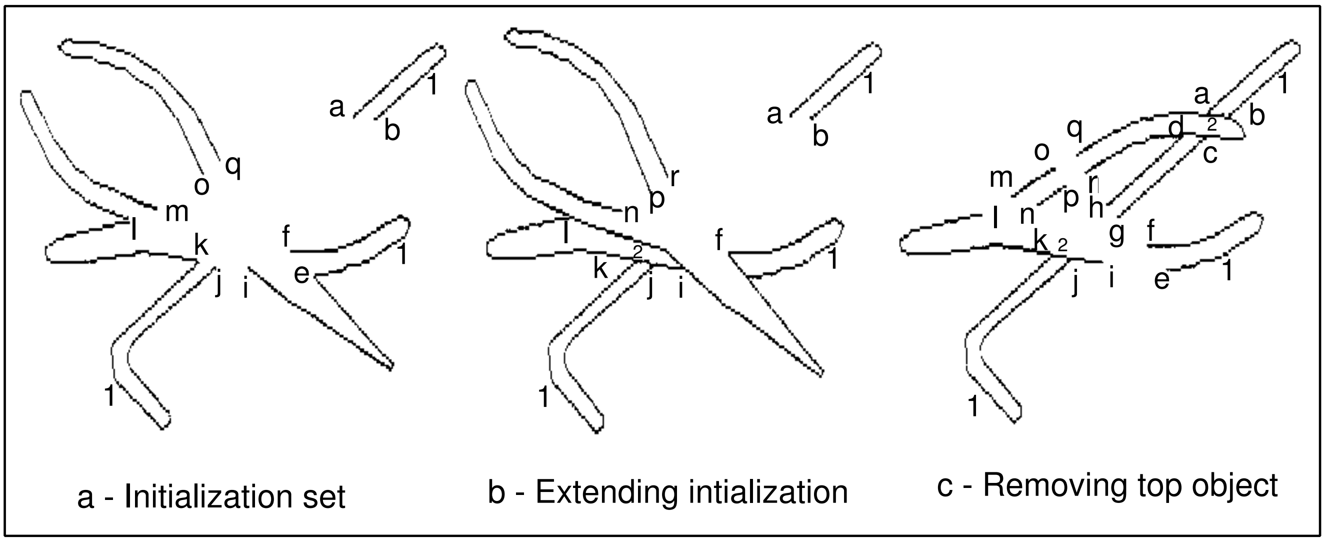
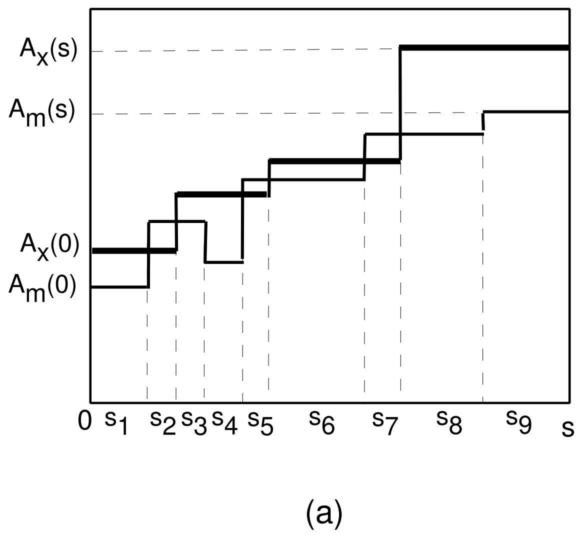


Figure 6: Steps in extending the initialization set to top object and fragments



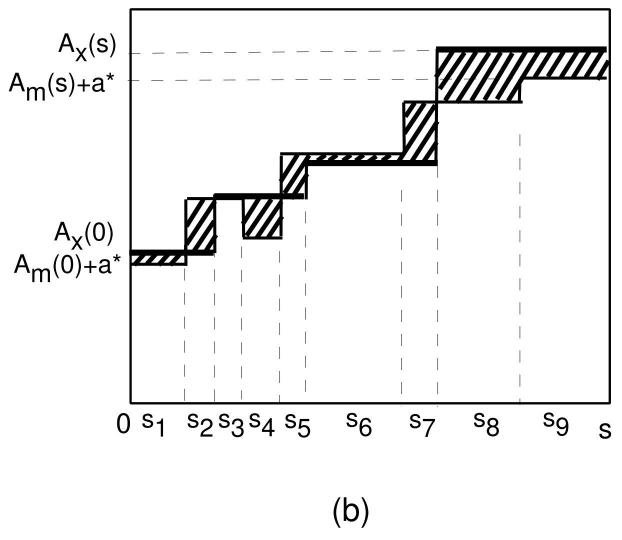


Figure 7: Initial (a) and least (b) distances between $A_x(s)$ and $A_m(s)$

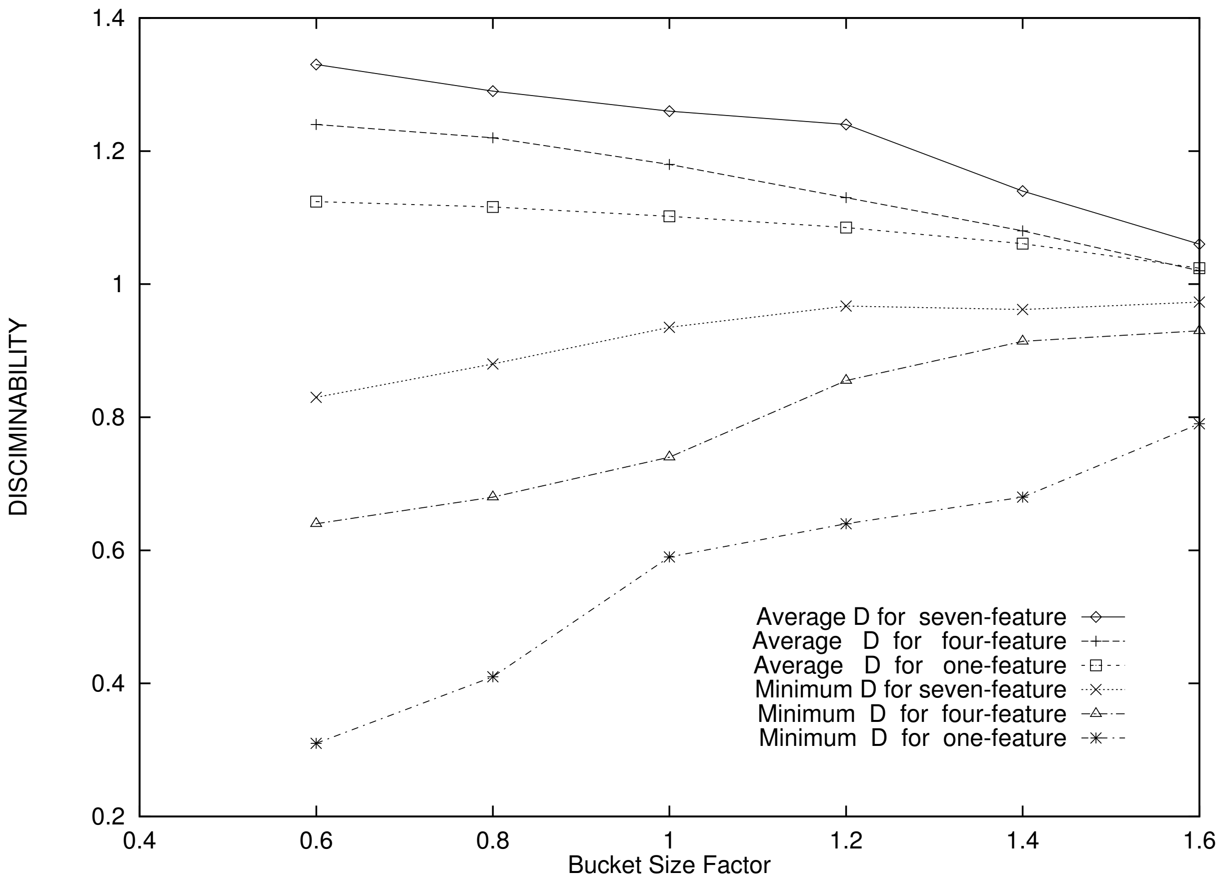


Figure 8: Discriminability function D versus bucket size

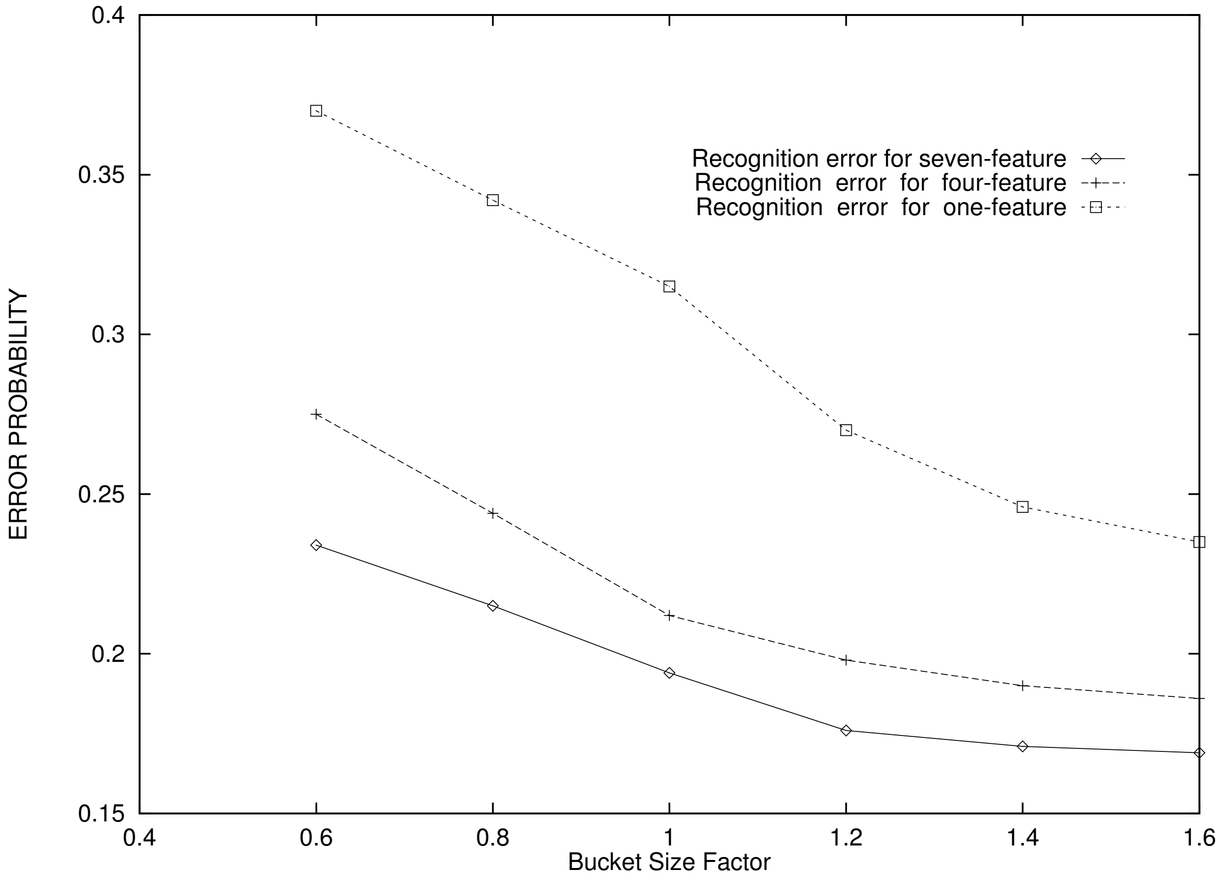


Figure 9: Recognition error for One-feature, four-feature, and seven-feature

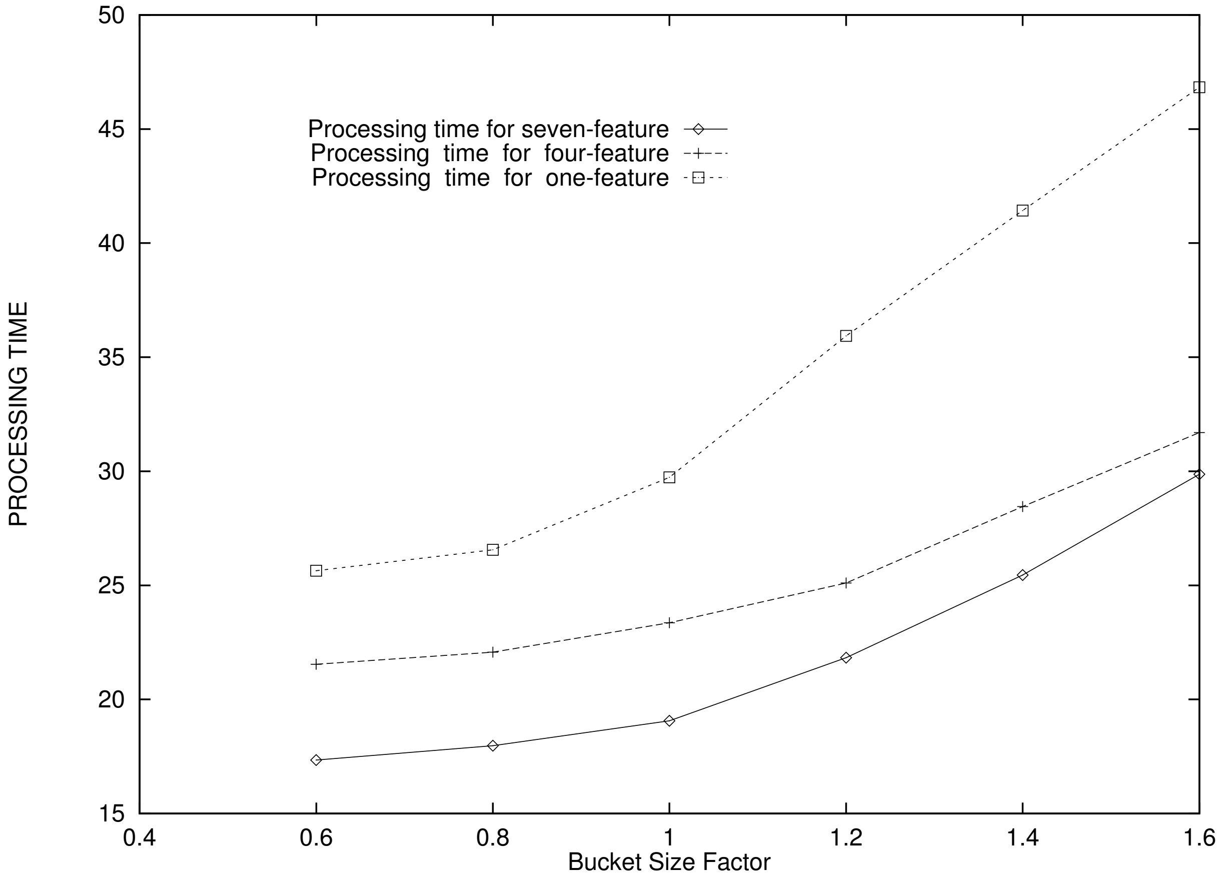


Figure 10: Processing time for one-feature, four-feature, and seven-feature