Annals of Mathematics and Artificial Intelligence 10(1994)85–124

A probabilistic matching algorithm for computer vision

Octavia I. Camps
Department of Electrical Engineering and Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, USA

Linda G. Shapiro
Department of Computer Science, FR-35, University of Washington, Seattle, WA 98195, USA

Robert M. Haralick
Electrical Engineering Department, FT-10, University of Washington, Seattle, WA 98195, USA

A model-based vision system attempts to find correspondences between features of an object model and features detected in an image for purposes of recognition, localization, or inspection. In this paper we pose the relational matching problem as a special case of the pattern complex recognition problem and propose a probabilistic model to describe the images of an object. This Bayesian approach allows us to make explicit statements of how an image is formed from a model, and hence define a natural matching cost that can be used to guide a heuristic search in finding the best observation mapping. Furthermore, we show that even though the nature of the feature matching problem is exponential, the use of the proposed algorithm keeps the size of the problem under control, by efficiently reducing the search space.

1. Introduction

Computer vision is the area of research concerned with the analysis of sensed images of scenes. An image is input to a program in the form of a matrix of numeric values representing light intensities. Image processing and feature extraction operators are used to convert the matrix to a symbolic description in terms of features such as line segments. Figure 1 illustrates an image and a set of extracted line segments features.

A model-based vision system attempts to find correspondences between the features of an object model and the features detected in the image for purposes of recognition, localization, or inspection. Examples of model-based systems are given in [1, 3–6, 8, 13, 14].

The matching algorithm described in this paper can be thought of in two ways, as a heuristic search and as a relational matching algorithm. Although

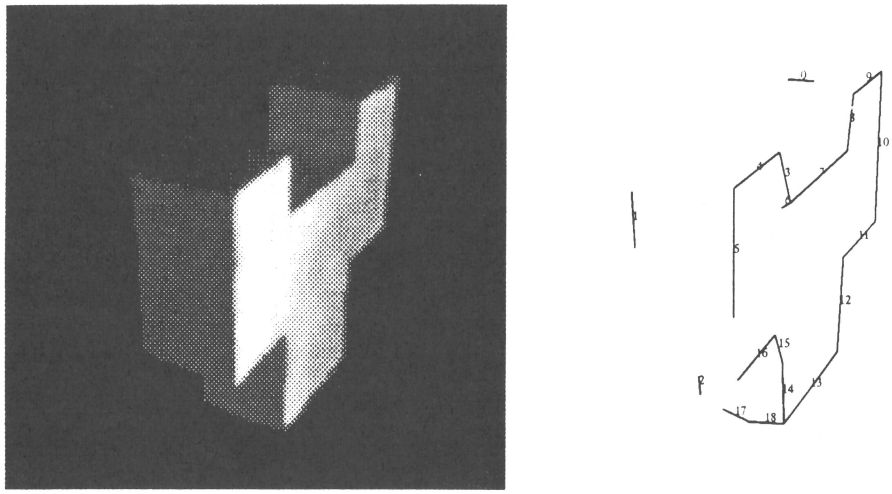


Fig. 1. (a) Image of an object. (b) Extracted segments.

heuristic search is easily implemented and the supporting theory is well known [15, 23], it has been shown that the number of nodes expanded during a depth-first search of an interpretation tree in the presence of spurious data is exponential, due to the combinatorics of the problem [7]. Relational matching has been expressed in several different formalisms. Early papers concentrated on graph or subgraph isomorphisms [22]. This led to many algorithms for discrete relaxation and the introduction of probabilistic relaxation [18]. The exact matching problem was generalized to the consistent labeling problem [11] and to the inexact matching problem [20]. This was extended further to the problem of determining the relational distance between two structural descriptions [19, 21]. Some recent related work includes structural stereopsis using information theory [2]. The present algorithm differs from all of these in its use of a Bayesian theoretical framework for the matching problem and the reduction of the search space.

2. Definitions and notation

Models and images are represented by their features, the relationships among them, and the measurements associated with them. In order to describe the matching algorithm, we must first provide some notation concerning features, relations, images and models.

2.1. FEATURES

A *feature* is an entity that describes a part of an object or an image. It is an abstract concept: it can be a point, an edge, a hole, a junction, or a higher level

combination of any of these. A feature has a *type* that identifies it, and a vector of *attributes* that represent its global properties. Formally,

DEFINITION 2.1

A *feature* f is a triple $f = (id, t, a)$ where id is a unique identifier, t is the feature type, and a is a vector of N attributes. The dimension of the attribute vector, N, is the *order* of the feature, and it only depends on the type of feature.

Examples of types of features are *edge* and *fork*; examples of attributes are *length* and *angles* between edges of a fork.

Let Tf be the set of all possible types of features, A be the set of all possible feature attribute vectors, and F a set of features. We define the *type mapping* $\mathrm{T}_F : F \rightarrow Tf$ associated with F to be the mapping that when applied to a feature returns its type, the *order mapping* $\mathrm{O}_F : F \rightarrow \mathbf{Z}^+$ associated with F to be the mapping that when applied to a feature returns the order of the feature, and the *attribute mapping* $\mathrm{A}_F : F \rightarrow A$ associated with F to be the mapping that when applied to a feature, returns the vector of attributes associated with its type.

Associated with every feature f there is a vector of measurements, the *attribute-value vector* $v = \mathrm{AV}_F(f)$ of dimension $\mathrm{O}_F(f)$. Let V be the set of all possible attribute-value vectors. The mapping $\mathrm{AV}_F : F \rightarrow V$ is called the *attribute-value* mapping associated with F.

Table 1
Feature sets and associated operators.

Symbol	Description
$f = (id, t, a)$	Feature id of type t and attribute vector a
Tf	Set of all possible type of features
A	Set of all possible attribute vectors
V	Set of all possible attribute-value vectors
F	Set of features
$\mathrm{T}_F(f)$	Returns the type of f
$\mathrm{O}_F(f)$	Returns the order of f
$\mathrm{A}_F(f)$	Returns the attribute-vector of f
$\mathrm{AV}_F(f)$	Returns the attribute value of f

2.2. RELATIONS

A feature participates in spatial relationships with other features. Each such relationship is represented by a *relational tuple*, which consists of a *type* specifying the relationship and a vector of *related features* that participate in that relationship. Associated with every relational tuple of features there is a real number between

0 and 1 called the *strength* of the relational tuple. The strength is a measurement of the confidence of the feature vector satisfying the specified relationship. Formally,

DEFINITION 2.2

A *relational tuple of features* r is a triple, $r = (id, t, v)$ where *id* is an unique identifier, t is the type of a relation over a set of features, and v is a vector of features satisfying the relation t. The *order* of the relational tuple of features r is defined as the dimension of the vector v.

Consider the parallelogram shown in fig. 2. It can be described in terms of its four sides and the relationships among them. In this case, the features are the four sides of the parallelogram, l_1, l_2, l_3, and l_4. Each side has an associated *length* attribute and is related to the other three features by the relationships *adjacent* and *parallel*.

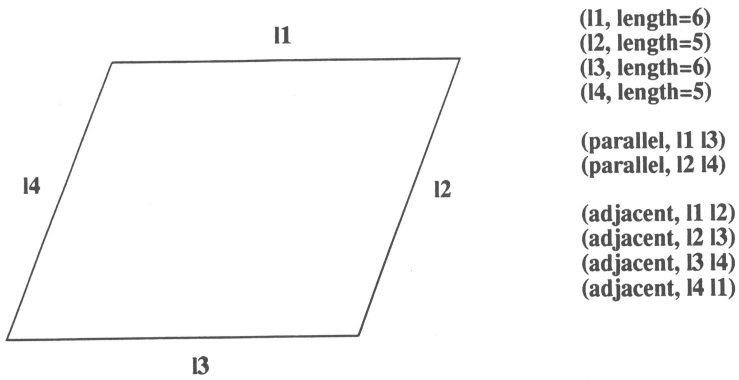


Fig. 2. Feature and relationships example.

Let Tr be the set of all possible types of relations among features in F, T be the set of all possible vectors of features in F, and R be a set of relational tuples of features in F. We define the *type mapping* $\mathrm{T}_R : R \rightarrow Tr$ associated with R to be the mapping that when applied to a relational tuple returns its type, the *vector mapping* $\mathrm{V}_R : R \rightarrow T$ associated with R to be the mapping that when applied to a relational tuple returns its feature vector, the *feature mapping* $\mathrm{F}_R : R \times Z \rightarrow F$ associated with R to be the mapping that when applied to a relational tuple returns the ith element of its feature vector, and the *order mapping* $\mathrm{O}_R : R \rightarrow \mathbf{Z}^+$ associated with R to be the mapping that when applied to a relational tuple returns its order.

Associated with every relational tuple of features r, there is a real number s between 0 and 1, called the *strength* of the relational tuple. The strength is a

measurement of the confidence that the feature vector $V_R(r)$ satisfies the relation of type $T_R(r)$. We define the *strength mapping* $S_R : R \rightarrow [0, 1]$ associated with R to be the mapping that when applied to a relational tuple returns its strength.

2.3. MODELS AND IMAGES

Models and images are represented by their features, the relationships among them, and the measurements associated with them. As in the consistent labeling formalism [11], we will call the image features *units* and the model features *labels*. Formally,

DEFINITION 2.3

A *model* M is a quadruple $M = (L, R, f_L, g_R)$ where L is the set of model features or *labels*, R is a set of relational tuples of labels, f_L is the attribute-value mapping associated with L, and g_S is the strength mapping associated with R.

Table 2
Relational tuple sets and associated operators.

Symbol	Description
$r = (id, t, v)$	Relational tuple *id* of type t and vector of features v
F	Set of features
Tr	Set of all possible type of relational tuples
T	Set of all possible vectors of features
R	Set of relational tuples
$T_R(r)$	Returns the type of r
$V_R(r)$	Returns the vector of features of r
$F_R(r, i)$	Returns the ith element of the feature vector of r
$O_R(r)$	Returns the order of r
$S_R(r)$	Returns the strength of r

DEFINITION 2.4

An *image* I is a quadruple $I = (U, S, f_U, g_S)$ where U is the set of image features or *units*, S is a set of relational tuples of units, f_U is the attribute-value mapping associated with U, and g_S is the strength mapping associated with S.

3. Relational matching

Although it is possible to identify simple objects by finding correspondences between some of the labels and units, recognizing complex objects having different

parts requires the use of the spatial relationships among the labels and units being matched.

The relational matching problem is a special case of the "pattern complex recognition problem" [9]. An image is an observation of a particular model. Let $M = (L, R, f_L, g_R)$ be the model, and $I = (U, S, f_U, g_S)$ be the observed image. Not all the labels in L participate in the observation, only a subset of labels $L^\circ \subseteq L$ is actually observed. Furthermore, only the relational tuples of labels representing relationships among labels in L° can be observed, and only a subset of them are actually observed. Let $R^\circ \subseteq R$ be the subset of the relational tuples of labels that are observed:

$$R^\circ \subseteq \{r | r \in R, n = \mathrm{O}_R(r), \mathrm{V}_R(r) \in (L^\circ)^n\}.$$

The set U consists of the unrecognized units. Some of the units observed in U come from labels in L°; others are unrelated and can be thought of as clutter objects. The set S is a set of observed relational tuples of units in U.

Table 3
Model and image symbols.

Symbol	Description
L	Set of model features or labels
R	Set of relational tuples of labels
f_L	Attribute-value mapping associated with L
g_R	Strength mapping associated with R
$M = (L, R, f_L, g_R)$	A model
U	Set of image features or units
S	Set of relational tuples of units
f_U	Attribute-value mapping associated with U
g_S	Strength mapping associated with S
$I = (U, S, f_U, g_S)$	An image

The relational matching problem is to find an unknown one-to-one correspondence $h : H \rightarrow U$ between a subset $H \subseteq L$ and a subset of U, associating some labels of L with some units of U. The mapping h is called the *observation mapping*. Notice that the matching process not only consists of finding the model M, but also of finding the correspondence h, which is the explanation of why the model M is the most likely model. In general we seek to maximize the *a posteriori* probability

$$P(M, h | I).$$

That is, we want to maximize the probability of the model being M, and the observation mapping being h, given that the image I is observed.

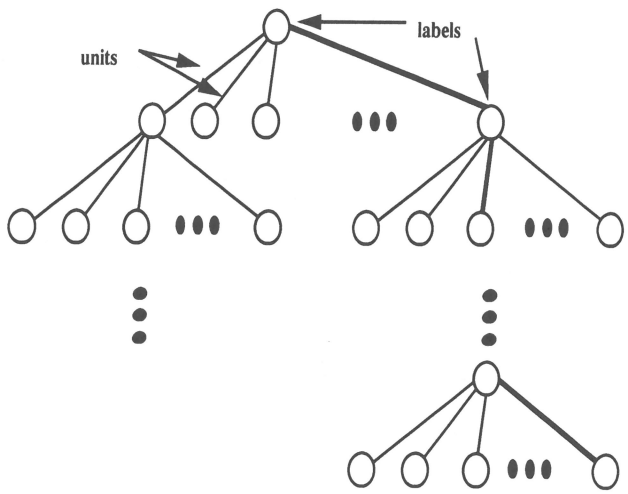
4. Matching by tree search

The matching process can be thought of as a state space search through the space of all possible interpretations Σ. The state space Σ is called the *matching space* and it is defined as follows:

DEFINITION 4.1

The *matching space*, Σ, is the state space of all possible interpretations, in which each state σ is defined by an observation mapping h_σ with degree of match $k_\sigma = \#\mathrm{Dom}(h_\sigma)$, where $\#$ indicates cardinality.

The search through the state space Σ can be achieved by doing an ordered search on a interpretation tree $\mathcal{T}$ such as the one shown in fig. 3. Each node in $\mathcal{T}$ represents a label, and each of its branches represents an assignment of the label to a unit. A search state σ in Σ is represented by a path $\mathcal{P}$ from the root to a leaf in the tree $\mathcal{T}$. In the following, the terms "path" and "partial mapping" will be used interchangeably.



Search Space: All possible interpretations
Search State: A path in the tree.

Fig. 3. Search tree $\mathcal{T}$.

The main difficulty in solving the matching problem by a tree search is the high combinatorics involved in the problem [7]. The number of possible interpretations in the tree grows exponentially with the number of labels and units. The

number of interpretations could be reduced by stopping the search before having a complete mapping. The problem, of course, is to determine when to stop.

The usual approach towards solving this problem is as follows: A path $\mathscr{P}$ in the interpretation tree $\mathscr{T}$ defines an observation mapping $m_{\mathscr{P}}$ with an associated cost $C_{\mathscr{P}} = C(m_{\mathscr{P}}, M, I)$ that measures the correctness of the mapping. Then, the matching process consists of finding the path $\mathscr{P}^*$ such that its associated observation mapping $m_{\mathscr{P}^*}$ has the least cost. In this way, the problem of selecting the correct interpretation has been shifted to the problem of defining an adequate cost function such that the interpretation having the least cost is indeed the correct one.

In the following sections we will introduce a theoretical probabilistic framework for the matching problem. The proposed framework allows us to define the cost of a mapping in a rigorous way, with a strong physical meaning. Furthermore, we will show how to find lower bounds of the defined cost, so that it can be used in guiding a heuristic search.

5. Solving the relational matching problem

The relational matching problem is a special case of the "pattern complex recognition problem" [9]. In the pattern complex recognition paradigm, the relational matching problem can be stated as follows:

> Given a model $M = (L, R, f_L, g_R)$ and an image $I = (U, S, f_U, g_S)$, find the observation mapping (h, H) such that the *a posteriori* probability $P(M, h|I)$ is maximum.

That is, we want to maximize the probability of the model being M and the observation mapping being h, given that the image I is observed. Hence, solving the relational matching problem requires a search procedure that can identify the model M and the mapping h such that the probability $P(M, h|I)$ is maximized. In order to define such a procedure, this probability must be further broken down and related to a cost function to be used in the search.

5.1. PROBABILITY OF AN OBSERVATION MAPPING

An observation mapping h consists of a domain contained in the set of labels L, and of the correspondences of these labels to some units belonging to the set of units U. Let $H \subseteq L$ be the domain of the mapping h. In the following, whenever we want to make explicit the need to consider both the correspondence h and its domain H, we will denote them as the pair (h, H). By the definition of conditional probability,

$$P(M, (h, H)|I) = \frac{P(M, (h, H), I)}{P(I)}.$$

Since M, and (h, H) do not appear in the denominator, maximizing the conditional probability $P(M, (h, H)|I)$ is equivalent to maximizing $P(M, (h, H), I)$. Replacing I by its components, and using the definition of conditional probability, we have

$$P(M, (h, H), I) = P(S, g_S|U, f_U, M, (h, H)) \cdot P(U, f_U, M, (h, H)). \quad (1)$$

Assuming that measurements and relationships are conditionally independent given M, U, f_U, and (h, H), and further that the relationships are independent of the feature measurements, we have

$$P(S, g_S|U, f_U, M, (h, H)) = P(S|U, M, (h, H)) \cdot P(g_S|U, f_U, M, (h, H)). \quad (2)$$

By definition of conditional probability,

$$P(U, f_U, M, (h, H)) = P(f_U|U, M, (h, H)) \cdot P(U, (h, H)|M) \cdot P(M). \quad (3)$$

Substituting eqs. (2) and (3) in eq. (1), and reordering we obtain

$$P(M, (h, H), I) = P_M \cdot P_U \cdot P_{f_U} \cdot P_S \cdot P_{g_S}, \quad (4)$$

where

$$\begin{aligned} P_M &= P(M), \\ P_U &= P(U, (h, H)|M), \\ P_{f_U} &= P(f_U|U, M, (h, H)), \\ P_S &= P(S|U, M, (h, H)), \\ P_{g_S} &= P(g_S|U, f_U, M, (h, H)). \end{aligned}$$

Equation (4) breaks down the joint probability of observing the model M through the mapping h as the image I into five terms. The first term is the prior probability of the model M. The other four terms are such that each one of them can be directly related to one of the four elements that describe the model and the image.

5.2. RELATIONAL MATCHING COST

In the context of the previous section, it is natural to define the *relational matching cost* of the observation mapping h as the *information* content in the probabilistic event that h is the observation mapping between the model M and the image I:

DEFINITION 5.1

Let $h : L \rightarrow U$, with $\mathrm{Dom}(h) = H$, be an observation mapping. The *relational matching cost* of h, $C(M,(h,H),I)$ is defined by

$$C(M,(h,H),I) = -\ln P(M,(h,H),I). \tag{5}$$

Taking the logarithm on both sides of eq. (4) and changing the sign we have that maximizing $P(M,(h,H),I)$ is equivalent to minimizing

$$C(M,(h,H),I) = C_M + C_U + C_S + C_{f_U} + C_{g_S}, \tag{6}$$

where

$$\begin{aligned} C_M &= -\ln P_M, \\ C_U &= -\ln P_U, \\ C_{f_U} &= -\ln P_{f_U}, \\ C_S &= -\ln P_S, \\ C_{g_S} &= -\ln P_{g_S}. \end{aligned}$$

Equation (6) shows that the cost C depends on the model, the label-unit assignments, the relational structures, and their measurements.

5.3. A PROBABILISTIC MODEL

To compute the relational matching cost defined in the previous section we need the corresponding probabilities. In this section we propose a model to compute these probabilities based on their physical meaning and the *Central Limit Theorem* [17].

• Model cost

The probability $P(M)$ is the *prior* probability for the model M to be observed, and it is available from the prediction system. The cost C_M is the cost associated with the model being considered, and it penalizes the selection of models whose prior probability of occurring is low.

• Label-unit assignment cost

The probability $P_U = P(U,(h,H)|M)$ evaluates the likelihood of the number of labels in H being matched through the mapping h to a subset of the observed units U. Since for the model M, the set L designates the set of possible labels, it is natural for $P(U,(h,H)|M)$ to depend on the difference between the size of the

set L and the size of the domain of h, as well as on the difference between the size of the set U and the size of the range of h. This probability should be high for observation maps that assign corresponding labels and units, and lower for those mappings that either miss assignments or assign labels to spurious units. Defining the *feature error* of mapping h, $E_f(h)$ as

$$E_f(h) = \#L - \#H + \#U - \#H = \#L + \#U - 2\#H, \tag{7}$$

it is reasonable to model

$$P_U = P(U,(h,H)|M) = \frac{1}{\sqrt{2\pi}\sigma_f} \exp\left[-\frac{1}{2}\left(\frac{E_f(h)-\mu_f}{\sigma_f}\right)^2\right], \tag{8}$$

where μ_f and σ_f are the mean and standard deviation of E_f. In the sequel, P_U will be referred as the *feature error distribution*. Figure 4 shows a Venn diagram representing the set of labels L, the set of units U, and the mapping $h : H \rightarrow U$, with $H \subseteq L$. The probability $P(U,(h,H)|M)$ as defined in eq. 4 penalizes those mappings h such that the hatched area is large or very small.

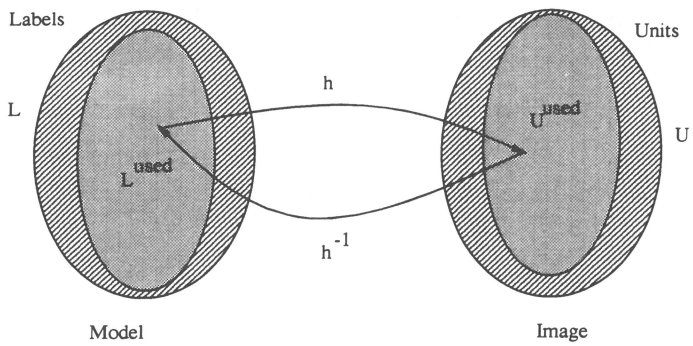


Fig. 4. $P(U,(h,H)|M)$ Venn Diagram. This probability penalizes those mappings h such that the hatched area is large or very small.

Using eqs. (6) and (8), the label-unit assignment cost or *feature cost* is given by

$$C_U = -\ln P_U = \ln\sqrt{2\pi}\sigma_f + \frac{1}{2}\left(\frac{E_f(h)-\mu_f}{\sigma_f}\right)^2, \tag{9}$$

$$= \ln\sqrt{2\pi}\sigma_f + \tfrac{1}{2}||E_f(h)-\mu_f||^2_{\sigma_f}, \tag{10}$$

where $||E_f(h)-\mu_f||_{\sigma_f}$ is the Mahalonabis distance between $E_f(h)$ and μ_f. Thus, the term C_U is the part of the cost that penalizes for the differences of sizes between the set of observed features U and the set of features of the model L.

• **Relational structural cost**

The probability $P_S = P(S|U, M, (h, H))$ evaluates how well the relationships among the labels are preserved by the mapping h. We will take this probability to be dependent on the number of relational tuples that are not preserved by the mapping.

Let Tr be the set of all possible types of relational tuples of labels. We define the *composition* of a relational tuple of labels of order N, $r \in (Tr \times H^N) \subseteq R$, with the mapping h, as a relational tuple of units of the same type as r such that each unit of its feature vector corresponds to a label in r through the mapping h:

DEFINITION 5.2

Given the one-to-one mapping $h : H \rightarrow U$, and the relational tuple of labels $r \in (Tr \times H^N) \subseteq R$, the *composition* of r with h is denoted as $h \circ r$, and is defined as the relational tuple of units given by,

$$h \circ r = (t, (u_1, u_2, \ldots, u_N)),$$

where t is the type of tuple r, N is the number of labels participating in tuple r, and $u_i = h(\mathrm{F}_R(r, i))$ for $0 < i \leq N$, where $\mathrm{F}_R(r, i)$ denotes the ith element of the feature vector of r.

The *composition* of the set of relational tuples of labels R with the mapping h is defined as the set of the relational tuples of units resulting from composing each of the relational tuples $r \in (Tr \times H^N) \subseteq R$ with h.

DEFINITION 5.3

Given the set of relational tuples of labels R, and the one-to-one mapping $h : H \rightarrow U$, the *composition* of R with h is denoted as $h \circ R$, and is defined as the set of relational tuples of units given by

$$h \circ R = \{s = h \circ r | r \in (Tr \times H^N) \subseteq R\}.$$

The compositions of a relational tuple of units s and of the set of relational tuples of units S with the inverse mapping h^{-1} are denoted by $h^{-1} \circ s$ and $h^{-1} \circ S$, respectively and are defined in a similar way.

DEFINITION 5.4

Given the one-to-one mapping $h : H \rightarrow U$, and the relational tuple of units $s \in (Tr \times \mathrm{Rge}(h)^N) \subseteq S$, where $\mathrm{Rge}(h)$ denotes the range of the mapping h, the

composition of s with h^{-1} is denoted as $h^{-1} \circ s$, and is defined as the relational tuple of labels given by

$$h^{-1} \circ s = (t, (l_1, l_2, \ldots, l_N)),$$

where t is the type of s, N is the number of units participating in s, and $l_i = h^{-1}(\mathrm{F}_S(s, i))$ for $0 < i \leq N$, where $\mathrm{F}_S(s, i)$ denotes the ith element of the feature vector of s.

DEFINITION 5.5

Given the set of relational tuples of units S, and the one-to-one mapping $h : H \rightarrow U$, the *composition* of S with h^{-1} is denoted as $h^{-1} \circ S$, and is defined as the set of relational tuples of labels given by

$$h^{-1} \circ S = \{r | \text{there exists } s \in (Tr \times \mathrm{Rge}(h)^N) \subseteq S, N = \mathrm{O}_S(s), \text{such that } r = h^{-1} \circ s\}.$$

The Venn diagram given in fig. 5 shows the sets of relational tuples R and S and their composition with h and h^{-1}, respectively. We will model the probability $P(S|U, M, (h, H))$ to penalize the number of relational tuples not preserved in the match as well as the number of relational tuples matched to spurious relational tuples in the image, i.e. to penalize those mappings h such that the hatched area in fig. 5 is large or very small. Defining the *relational error* of mapping h, $E_r(h)$ as

$$E_r(h) = \#(R - h^{-1} \circ S) + \#(S - h \circ R), \tag{11}$$

it is natural to use the concepts introduced above to model:

$$P_S = P(S|U, M, (h, H)) = \frac{1}{\sqrt{2\pi}\sigma_r} \exp\left[-\frac{1}{2}\left(\frac{E_r(h) - \mu_r}{\sigma_r}\right)^2\right]. \tag{12}$$

P_S will be referred as the *relational error distribution*. Using eqs. (6) and (12), the relational structural cost or *relational cost* is given by

$$C_S = -\ln P_S = \ln \sqrt{2\pi}\sigma_r + \frac{1}{2}\left(\frac{E_r(h) - \mu_r}{\sigma_r}\right)^2 \tag{13}$$

$$= \ln \sqrt{2\pi}\sigma_r + \tfrac{1}{2}||E_r(h) - \mu_r||^2_{\sigma_r}, \tag{14}$$

where $||E_r(h) - \mu_r||_{\sigma_r}$ is the Mahalonabis distance between $E_r(h)$ and μ_r. Thus, the cost C_S is the part of the cost that accounts for the differences between the set of observed relationships S and the set of relationships of the model R.

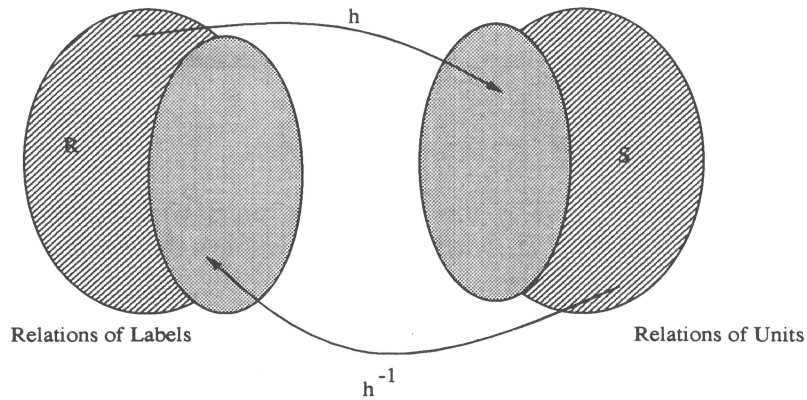


Fig. 5. $P(S|U, M, (h, H))$ Venn Diagram. This probability penalizes those mappings h such that the hatched area is large or very small.

• Metric costs

Since the probabilities $P_{f_U} = P(f_U|U, M, (h, H))$ and $P_{g_S} = P(g_S|U, f_U, M, (h, H))$ are both probabilities of mappings that associate values to elements of a set, their treatment is similar. Consider $P(f_U \circ h|U, M, H)$, where $f_U \circ h$ is the composition of f_U with h defined by

$$(f_U \circ h)(l) = f_U(h(l)), \quad l \in H.$$

Let ρ be a suitable metric and $f_{L|_H}$ represent the attribute-value mapping f_L restricted to the labels in the domain H. Then, we define the *feature metric error* of mapping h, $E_{f_U}(h)$ as

$$E_{f_U}(h) = \rho(f_U \circ h, f_{L|_H}). \tag{15}$$

Since f_L is the attribute-value mapping associated with L, we can model P_{f_U} by

$$P_{f_U} = P(\rho(f_U \circ h, f_{L|_H})|U, M, (h, H)) = \frac{1}{\sqrt{2\pi}\sigma_{f_U}} \exp\left[-\frac{1}{2}\left(\frac{E_{f_U}(h) - \mu_{f_U}}{\sigma_{f_U}}\right)^2\right]. \tag{16}$$

P_{f_U} will be referred as the *feature metric error distribution*. Using eqs. (6) and (16), the *feature metric cost* C_{f_U} is given by

$$C_{f_U} = -\ln P_{f_U} = \ln\sqrt{2\pi}\sigma_{f_U} + \frac{1}{2}\left(\frac{E_{f_U} - \mu_{f_U}}{\sigma_{f_U}}\right)^2 \tag{17}$$

$$= \ln\sqrt{2\pi}\sigma_{f_U} + \tfrac{1}{2}||E_{f_U} - \mu_{f_U}||^2_{\sigma_{f_U}}, \tag{18}$$

where $||E_{f_U} - \mu_{f_U}||_{\sigma_{f_U}}$ is the Mahalonabis distance between E_{f_U} and μ_{f_U}. Hence, the

farther the feature metric error is from its expected value, the larger the cost term C_{f_U}. Figure 6 illustrates this concept. The Venn diagram shows a label l with its associated attribute value $f_L(l) = v_l$, and its corresponding unit $h(l) = u$ with its associated attribute value $f_U(u) = v_u$. The probability defined in eq. (16) assigns a high probability to those mappings such that the values v_l and v_u are at the "right" distance apart with respect to the metric ρ.

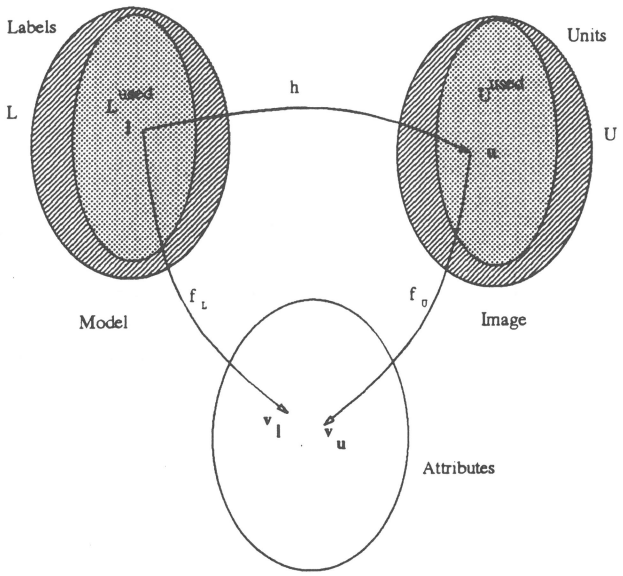


Fig. 6. $P(f_U|U, M, (h, H))$ Venn Diagram. This probability assigns a high probability to those mappings such that the corresponding attribute values $f_L(l)$ and $f_U(h(l))$ are at the "right" distance.

Reasoning in an analogous way, the probability $P_{g_S} = P(g_S|U, f_U, M, (h, H))$ can be rewritten as $P(g_S \circ h|U, f_U, M, H)$, where $g_S \circ h$ is the composition of g_s with h defined by

$$(g_S \circ h)(r) = g_S(h \circ r), \quad r \in R, \quad \text{and} \quad (h \circ r) \in S.$$

Let ρ be a suitable metric. Then, we define the *relational strength error* of mapping h, $E_{g_S}(h)$ as

$$E_{g_S}(h) = \rho(g_S \circ h, g_{R|_{h \circ R}}), \tag{19}$$

where $g_{R|_{h \circ R}}$ represents the strength mapping g_S restricted to the set $h \circ R$.

Since g_R is the strength value mapping associated with R, we can model P_{g_S} by

$$P_{g_S} = P(\rho(g_S \circ h, g_R)|f_U, U, M, (h, H)) = \frac{1}{\sqrt{2\pi}\sigma_{g_S}} \exp\left[-\frac{1}{2}\left(\frac{E_{g_S}(h) - \mu_{g_S}}{\sigma_{g_S}}\right)^2\right]. \tag{20}$$

P_{g_S} will be referred as the *relational strength error distribution*. Using eqs. (6) and (20), the *relational strength cost* C_{g_S} is given by

$$C_{g_S} = -\ln P_{g_S} = \ln\sqrt{2\pi}\sigma_{g_S} + \frac{1}{2}\left(\frac{E_{g_S} - \mu_{g_S}}{\sigma_{g_S}}\right)^2 \tag{21}$$

$$= \ln\sqrt{2\pi}\sigma_{g_S} + \tfrac{1}{2}||E_{g_S} - \mu_{g_S}||^2_{\sigma_{g_S}}, \tag{22}$$

where $||E_{g_S} - \mu_{g_S}||_{\sigma_{g_S}}$ is the Mahalonabis distance between E_{g_S} and μ_{g_S}.

Having modeled the probabilities involved, we can now compute the relational matching cost. Substituting eqs. (10), (14), (18), and (22) in eq. (6), we have

$$C(M, (h, H), I) = A + \tfrac{1}{2}||E_f(h) - \mu_f||^2_{\sigma_f} + \tfrac{1}{2}||E_r(h) - \mu_r||^2_{\sigma_r} + \tfrac{1}{2}||E_{f_U}(h) - \mu_{f_U}||^2_{\sigma_{f_U}} + \tfrac{1}{2}||E_{g_S}(h) - \mu_{g_S}||^2_{\sigma_{g_S}}, \tag{23}$$

where $A = -\ln P(M) + 4\ln\sqrt{2\pi} + \ln(\sigma_f \sigma_{f_U} \sigma_r \sigma_{g_S})$, is a constant for a given model M.

Hence, the relational matching cost consists of four quadratic terms plus a term that is constant for a given model. Each one of the quadratic terms depends on a different type of *error*; i.e., there is a term for the feature error, a term for the relational error, a term for the feature attributes error, and a term for the relational strength error. Equation (23) clearly shows that there is an optimum size for the mapping h, between the null mapping (a mapping with no correspondences) and a mapping that would match everything (including all spurious data).

6. Iterative-deepening-A* matching

A match can be found by using the relational matching cost defined in section 5.2, and the well known *branch-and-bound* tree search technique. In the standard branch and bound approach during search there are many incomplete paths contending for further consideration. The one with the least cost is extended one level, creating as many new incomplete paths as there are branches. This procedure is repeated until the tree is exhausted.

The branch and bound search can be improved greatly if the path to be extended is selected such that an estimate of the total cost using that sub-path is minimal. This search technique is usually known as A^*. An important and well known result is that if the estimate of the total cost is always less than the actual cost, the path found by A^* is optimal. The time complexity of the A^* search, is in general, an exponential function of the error of the estimate. If the error is constant, then the time complexity is constant. If the error is a linear function of the depth of the search (i.e. constant relative error), then the time complexity is exponential. The advantage of A^* over brute-force search, which also has exponential time complexity, is that the base of the exponent in the A^* case is significantly smaller if the estimate is accurate. The drawback of the A^* algorithm is the same as that of breadth-first search, namely its memory requirement. The algorithm must maintain a list of all contending paths. In each cycle, the number of contending paths is increased by $b - 1$, where b is the branching factor of the node being extended. Thus, the space complexity of A^* is $O(b^d)$, where d is the solution depth level.

Korf [15] presented a new search algorithm, called iterative-deepening-A^* (ID A^*) that gets around the memory problem of A^* without sacrificing optimality or time complexity. The algorithm consists of a sequence of depth-first searches. ID A^* starts with an initial threshold value equal to the estimate total cost for the root of the tree. In each iteration, the algorithm is a pure depth-first search, cutting off any branch that has an estimated total cost larger than the current threshold value. If a solution is expanded, the algorithm is finished. Otherwise, a new threshold value is set to the minimum estimated cost that exceeded the previous threshold, and another depth-first search is begun from scratch.

As in the case of A^*, if the estimated total cost is an underestimate of the real total cost, ID A^* finds the optimal solution. The advantage of ID A^* over A^* is that since each iteration of the algorithm is a depth-first search, the memory complexity is $O(d)$, instead of being exponential. The number of nodes opened by ID A^* is asymptotically the number of nodes opened by A^*, provided that the tree grows exponentially. In practice, ID A^* runs faster than A^*, since its overhead per node is less than the overhead for A^*.

6.1. RELATIONAL MATCHING COST UNDERESTIMATE

The ID A^* algorithm requires an underestimate of the relational matching cost of an observation mapping that is an *extension* of the current partial mapping. In this section we formally define the extension of a partial mapping and use this concept to find a lower bound of the relational matching cost.

DEFINITION 6.1

Given two one-to-one mappings h and m, such that $\mathrm{Dom}(m) \subseteq \mathrm{Dom}(h)$, and $m(l) = h(l)$ for all $l \in \mathrm{Dom}(m)$, we say that the function h is an *extension* of the

function m, and that the function m is a *restriction of the function h. The order* of the extension h with respect to m is the difference between the cardinalities of the sets Dom(h) and Dom(m).

To find a lower bound of the relational matching cost of an extension of a partial mapping, we start by noticing that a partial mapping partitions the sets of features into disjoint sets. Figure 7 gives a diagram of the sets L and U showing the partitions induced by a partial match m.

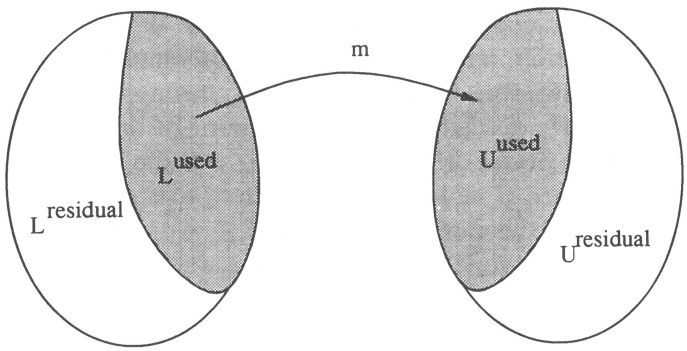


Fig. 7. Partition of the sets of features induced by a partial match.

Let $m : L \rightarrow U$ be a partial mapping assigning some labels to some units. The mapping m partitions the sets of features L and U into the sets of *used* features in the match and the sets of *residual* features; i.e., those not used in the match.

DEFINITION 6.2

The *set of used labels*, $L^u(m)$, is the subset of labels in L that have been assigned a unit in $U^u(m)$ by the mapping $m : L^u(m) = \text{Dom}(m)$.

DEFINITION 6.3

The *set of used units*, $U^u(m)$, is the subset of units in U such that there is some label in L^u that have been assigned a unit in U through the mapping $m : U^u(m) = \text{Range}(m)$.

DEFINITION 6.4

The *set of residual labels*, $L^r(m)$, is the subset of labels in L that have not been assigned a unit in U by the mapping $m : L^r(m) = L - L^u(m)$.

DEFINITION 6.5

The *set of residual units*, $U^r(m)$, is the subset of units in U such that no label in L has been assigned to them by the mapping m : $U^r(m) = U - U^u(m)$.

Let m_j be an extension of order j of m. Then, the feature error for the mapping m_j, $E_f(m_j)$, is given by

$$E_f(m_j) = \#L^r(m_j) + \#U^r(m_j) - 2\#\mathrm{Dom}(m_j) = E_f(m) - 2j, \tag{24}$$

and the feature error cost is given by

$$C_U(m_j) = \ln \sqrt{2\pi}\sigma_f + \tfrac{1}{2}||E_f(m) - 2j - \mu_f||^2_{\sigma_f}. \tag{25}$$

Figure 8 shows a plot of the feature error cost versus the feature error. The feature error cost is quadratic on the feature error $E_f(h) = \#L + \#U - 2\#H$. As the order of the extension j increases, the feature error decreases, and the cost moves on a parabola towards the left. The maximum possible order of an extension of m is given by $J = \min\{\#L^r(m), \#U^r(m)\}$. Hence,

$$E_f(m_j) \geq E_f(m_J) = E_f(m) - 2J \tag{26}$$

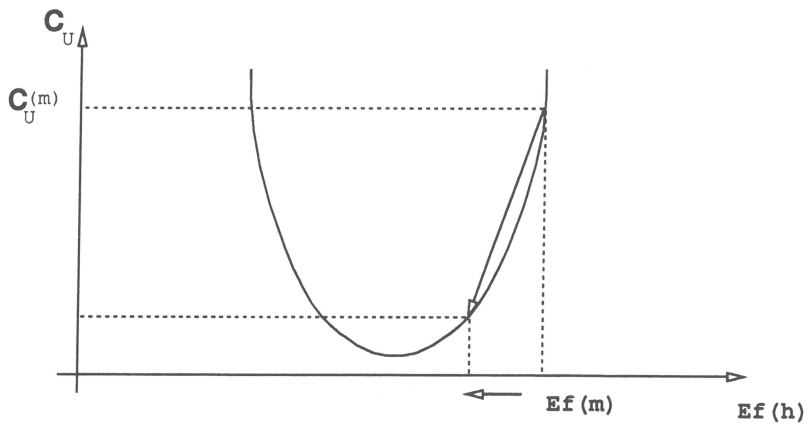


Fig. 8. Feature Error Cost. The feature error cost is quadratic on the feature error $E_f(h) = \#L + \#U - 2\#H$. As the order of the mapping increases, the feature error decreases and the feature error cost moves on the parabola towards the left.

and a lower bound of the feature error cost for the extensions of m is given by

$$C_U(m_j) \geq \begin{cases} C_U(m_J) & \text{if } \mu_f \leq E_f(m_J) \leq E_f(m), \\ \ln(\sqrt{2\pi}\sigma_f) & \text{if } E_f(m_J) \leq \mu_f \leq E_f(m), \\ C_U(m) & \text{otherwise,} \end{cases} \tag{27}$$

as it is illustrated in fig. 9.

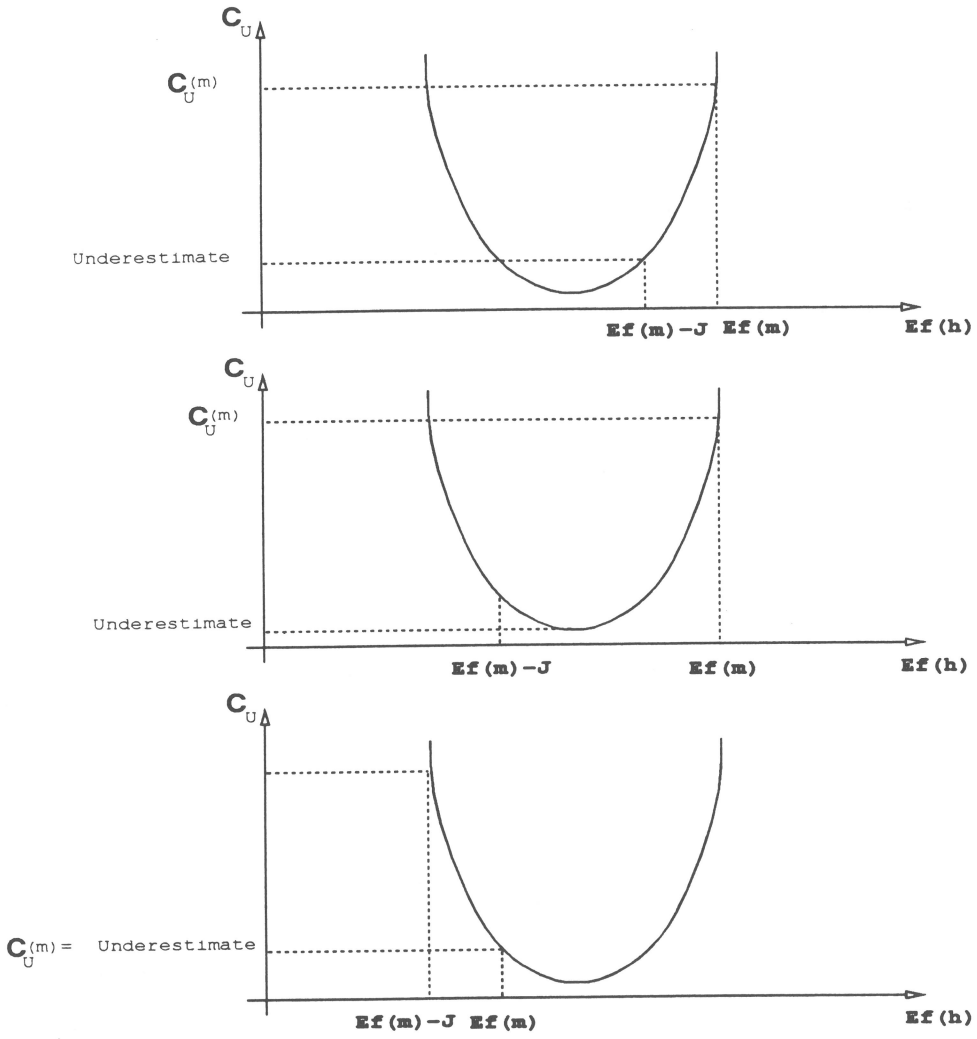


Fig. 9. Feature Error Cost Underestimate. (a) $\mu_f \leq E_f(m_J) \leq E_f(m)$. (b) $E_f(m_J) \leq \mu_f \leq E_f(m)$. (c) $E_f(m_J) \leq E_f(m) \leq \mu_f$.

Similarly, a partial mapping m partitions the sets of relational tuples into disjoint subsets. Figure 10 shows a diagram of the sets S and R, and the partition induced on them by the partial match m.

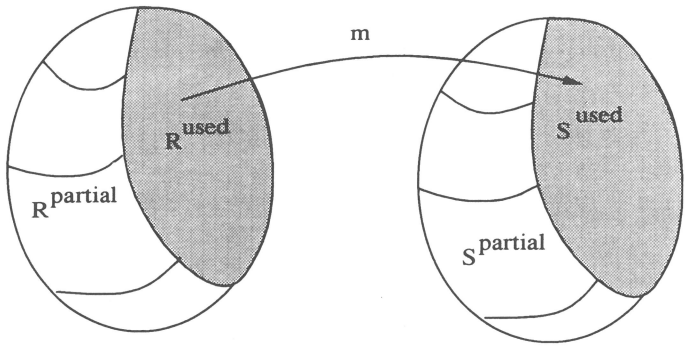


Fig. 10. Partition of the sets of relational tuples of features induced by a partial match.

DEFINITION 6.6

The *set of used relational tuples of labels*, $R^u(m)$, is the subset of relational tuples of labels in R such that all the labels in their feature vectors have been associated with a corresponding unit in U through the mapping m.

DEFINITION 6.7

The *set of i-partially free relational tuples of labels*, $R_i^p(m)$, is the subset of relational tuples of labels in S such that all but $i > 0$ of the labels in their feature vectors have been associated with a corresponding unit in U through the mapping m.

DEFINITION 6.8

The *set of used relational tuples of units*, $S^u(m)$, is the subset of relational tuples of units in S such that all the units in their feature vectors have been associated with a corresponding label in L through the mapping m.

DEFINITION 6.9

The *set of i-partially free relational tuples of units*, $S_i^p(m)$, is the subset of relational tuples of units in S such that all but $i > 0$ of the units in their feature vectors have been associated with a corresponding label in L through the mapping m.

DEFINITION 6.10

A partially free relational tuple of labels $r \in R_i^p(m)$ and a partially free relational tuple of units $s \in S_i^p(m)$ are *compatible* if they agree on the features that have been already matched.

The relational error for the mapping m_j, $E_r(m_j)$, is given by

$$E_r(m_j) = \#(R - m_j^{-1} \circ S) + \#(S - m_j \circ R), \tag{28}$$

and the relational cost is given by

$$C_S(m_j) = \tfrac{1}{2}||E_r(m_j) - \mu_r||^2_{\sigma_r} + \ln \sqrt{2\pi}\sigma_r. \tag{29}$$

Figure 11 shows a plot of the relational error cost versus the relational error. The relational error cost is quadratic on the relational error, $E_r(h) = \#(R - h^{-1} \circ S) + \#(S - h \circ R)$. As the order of extension j increases, the cardinal of the domain of m_j increases and some partially free relational tuples become used and the cardinal of the sets $(R - m_j^{-1} \circ S)$ and $(S - m_j \circ R)$ cannot increase. Hence, as j increases, the relational error $E_r(m_j)$ decreases, and the relational error cost C_S moves on a parabola towards the left:

$$E_r(m) \geq E_r(m_j). \tag{30}$$

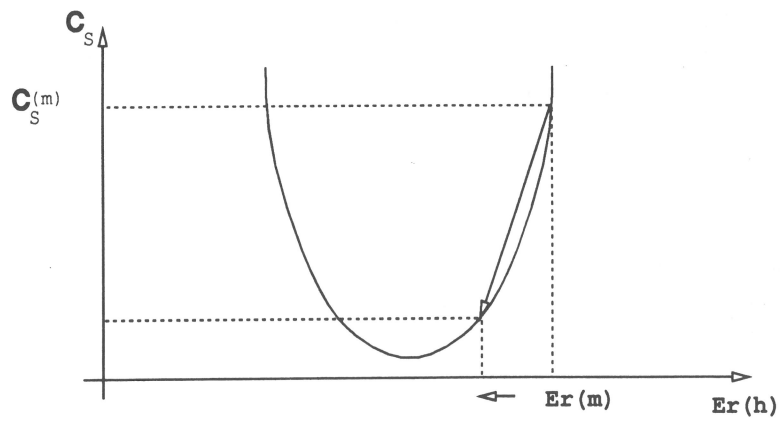


Fig. 11. Relational Error Cost. The relational error cost is quadratic on the relational error $E_r(h) = \#(R - h^{-1} \circ S) + \#(S - h \circ R)$. As the order of the mapping increases, the relational error decreases and the relational error cost moves on a parabola towards the left.

Let $c_i(m)$ be the minimum between the number of relational tuples of labels i-partially free that have compatibles, and the number of relational tuples of units i-partially free that have compatibles through the mapping m. That is, $c_i(m)$ is the maximum number of partially free relational tuples that could be correctly matched by extending the mapping m with i correspondences. Then, the relational error of the mapping m_j, $E_r(m_j)$, is bounded by

$$E_r(m_j) \geq E_r(m) - 2\sum_{i \leq j} c_i(m) = E_r^{\min}(m, j). \tag{31}$$

Thus, using eqs. (30) and (31) the relational error cost of an extended map is bounded by

$$C_S(m_j) \geq \begin{cases} \ln(\sqrt{2\pi}\sigma_r) + \frac{1}{2}||E_r^{\min}(m, j) - \mu_r||^2_{\sigma_r}, & \text{if } \mu_r \leq E_r^{\min}(m, j) \leq E_r(m), \\ \ln(\sqrt{2\pi}\sigma_r) & \text{if } E_r^{\min}(m, j) \leq \mu_r \leq E_r(m), \\ C_S(m) & \text{otherwise} \end{cases} \tag{32}$$

as it is illustrated in fig. 12.

Inequalities (27) and (32) provide an easy to compute underestimate of the total cost of a partial match. This lower bound of the total cost can be used to quickly guide an ID A^* algorithm to the correct mapping reducing dramatically the number of nodes to be opened during the search. The complete matching algorithm using ID A^* is given in fig. 13.

7. Model parameters estimation

In order to estimate the model parameters, we use the PREMIO system [4]. PREMIO is a CAD-based computer vision system that employs CAD models of 3D objects and knowledge of surface reflectance properties, light sources, sensors characteristics, and the performance of feature detectors to estimate the probability of features being detected in various views of the object.

The model

PREMIO's prediction module predicts which 2D features should be detectable for a given configuration of light and sensor and a given image processing sequence. Each of the detectable 2D features correspond to their originating 3D feature, and have associated attribute values.

Given a set of n predictions for a set of n sensor and lighting configurations corresponding to a view aspect[1] of the object, PREMIO approximates the

[1] A view aspect is defined as a set of views with similar properties. In this paper, an aspect corresponds to a set of views that have the same visible faces.

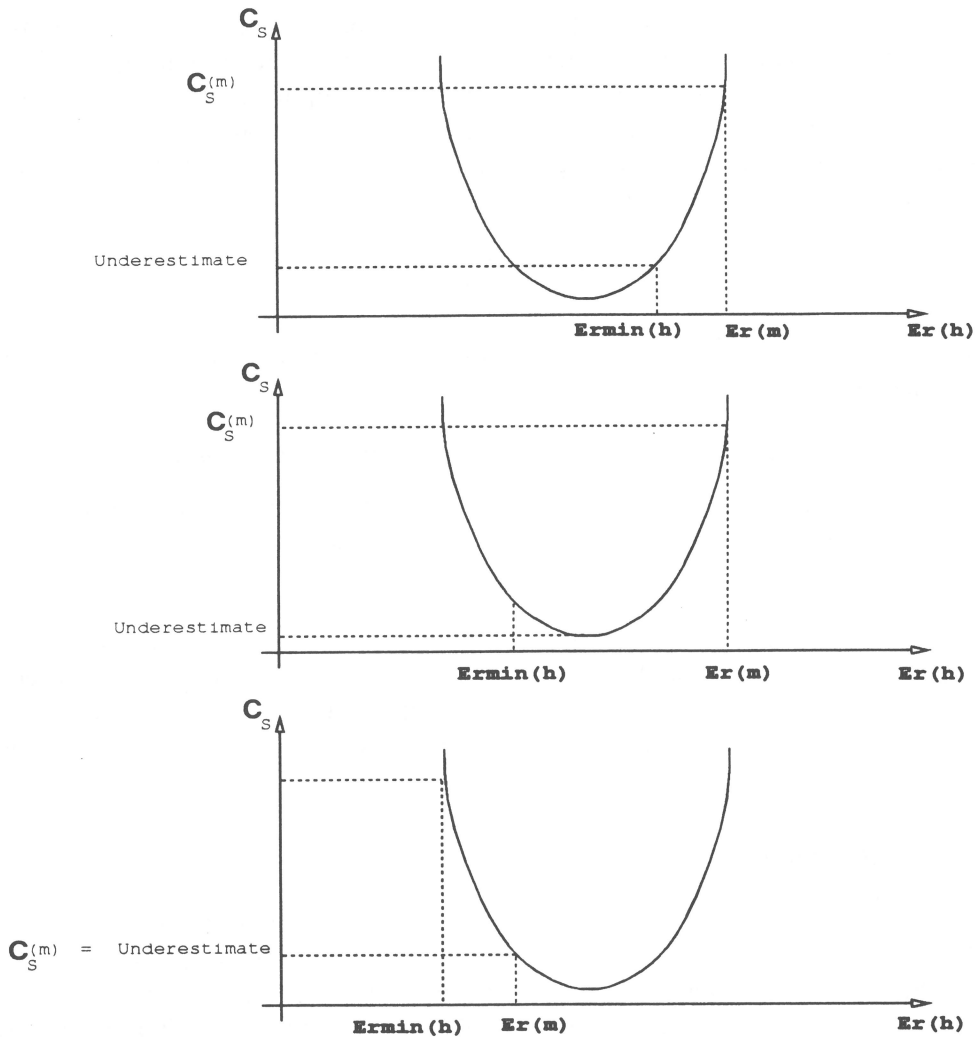


Fig. 12. Relational Error Cost Underestimate. (a) $\mu_r \leq E_r(m) - 2\sum_{i \geq j} c_i(m) \leq E_r(m)$. (b) $E_r(m) - 2\sum_{i \leq j} c_i(m) \leq \mu_r \leq E_r(m)$. (c) $E_r(m) - 2\sum_{i \leq j} c_i(m) \leq E_r(m) \leq \mu_r$.

detectability of a 2D feature by the frequency rate of its appearance. Two 2D features appearing in two different images are considered to be the same feature if they have a common 3D originating feature. The set of labels L is formed by those 2D features that have high enough probability of being detected (above threshold t_f), as a whole or in pieces for the given set of sensors and light sources. Furthermore, each feature in L has associated attributes which are given by the mean and the standard deviation of the attribute values of the feature for the n predictions.

Similarly, PREMIO's prediction module predicts which relationships among features would be detected and their strength, for a given configuration

```
Step 0: Initialization.
Set Threshold Th = EC(root).
Set ε to the desired matching cost.
If Th ≥ ε
  Begin
    The model and the image do not match.
    Go to step 4.
  End if.
Until the solution is found or the tree is exhausted, do :
Begin
  Step 1: Start Depth First.
  Form a stack Q_P of partial matches, and let P_0 be the initial partial match.
  Set MinPruned := highest value.
  Step 2: Iterate over current paths.
  Until Q_P is empty, do
  Begin
    P := FIRST(Q_P), m := partial mapping associated with P
    C_m := relational cost of m
    Step 2.1: Test if P can be extended.
    If the path P can be extended,
    Begin
        Step 2.1.1: Select a label to extend the path.
        Look for two tuples, one from R and one from S whose components
        are not all matched, that are compatible. Two relational tuples are
        compatible if they have the same number of features and they agree on
        the features that have been already matched. The relational tuples that
        are partially matched should be checked before than those that are not.
        Step 2.1.2 Extend the path
        For each u ∈ U^r, do
        Begin
          h_1 := path m extended with the pair (l, u).
          P' := path associated with the mapping h_1.
          C_{h_1} := relational cost of h_1.
          EC(h_1) := underestimate of the cost of the extensions of h_1.
          Step 2.1.2.1 Compare to the upper bound.
          If EC(h_1) ≤ ε
          Begin
            Step 2.1.2.1.1 Check if the goal has been achieved.
            If C_{h_1} ≤ ε
            Begin
              P' is a satisfactory match.
              Exit .
            End if.
            Step 2.1.2.1.2 Add the path to the stack.
            If EC(h_1) ≤ Th
            Begin
              FIRST(Q_P) := P'.
            Else
              MinPruned := min{MinPruned, EC(h_1)}
            End if.
          End if.
        End for.
    End if.
  End until.
  Step 3: Update Threshold
  Th = MinPruned
  End until.
Step 4: End of Algorithm
Announce failure.
```

Fig. 13. Matching algorithm.

of light and camera and a given image processing sequence. Given the n predictions we approximate the probability of a relation among a set of features being detected by the frequency rate of its appearance. The set of relational tuples R is formed for those relations among features in L that have high enough probability of holding (above threshold t_R). As with feature attributes, the relationship strength values of the tuples in R are represented by the mean and standard deviation of the relational tuples for the n predictions.

The model $M = (L, R, f_U, g_S)$ obtained in this way, is a *probabilistic model* of the object for the given set of configurations of sensors and lights. Note that neither all the features in L, nor all the relational tuples in R need to be present in a single prediction. Neither do all the features of a particular prediction need to be in L. The model M combines a group of predictions into a single model, which is a sort of "average" model. The differences between the model M and the individual predictions that were used to build the model are summarized in the statistics P_U, P_S, P_{f_U}, and P_{g_S}.

The statistics

Once the model M is obtained, the individual predictions can be used to generate samples for the four error distributions P_U, P_S, P_{f_U}, and P_{g_S}. Let $I_1, \ldots, I_n$ be a set of n predictions. Each prediction can be represented by a four tuple $I_i = (U_i, S_i, f_{U_i}, g_{S_i})$ where U_i is the set of units, S_i is the set of relational tuples of units, f_{U_i} is the attribute mapping for the units in U_i, and g_{S_i} is the relationship strength mapping for the relational set S_i. By construction we know

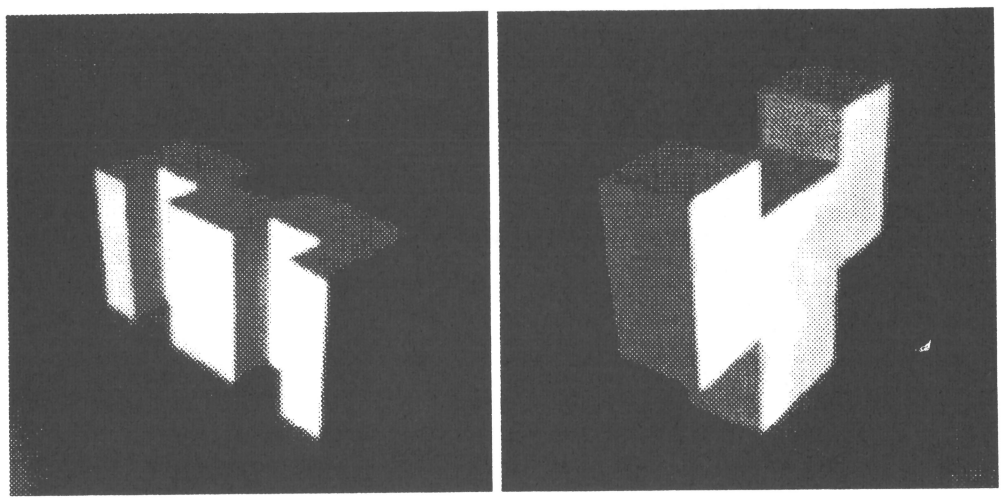


Fig. 14. (a) Cube3Cut. (b) Fork.

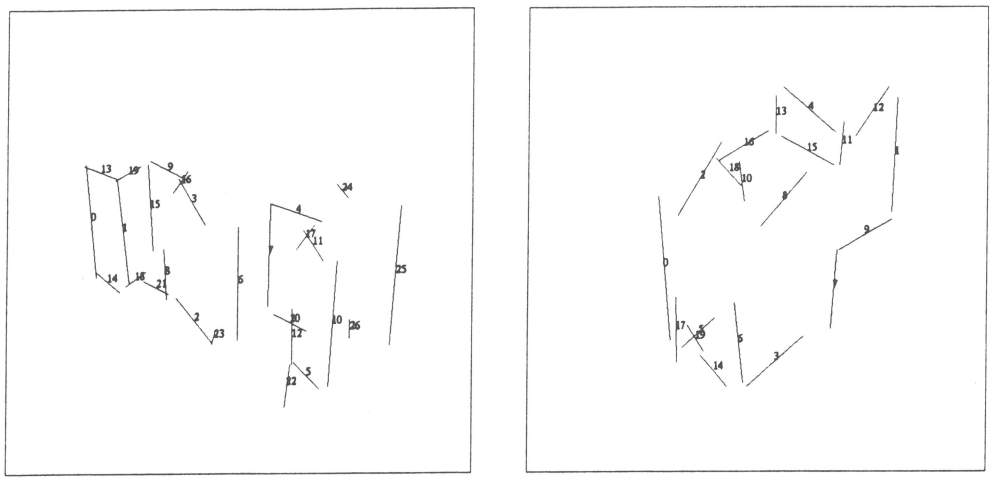


Fig. 15. Cube3Cut and Fork Models (segments). The line drawings are a visualization of the set of labels forming the models. The labels are drawn using their mean attribute values. The numbers by the segment are the labels Ids., and indicate their relative detectability, with the lower the number, the higher the detectability.

the true observation mapping for each of the predictions. Let $h_i : H_i \rightarrow U_i$, with $H_i \subseteq L$ be the true observation mapping for a prediction i. Then, we can compute the quantities $\#L + \#U_i - 2\#H_i$, $\#(R - S_i \circ h_i^{-1}) + \#(S_i - R \circ h_i)$, $\rho(f_{U_i} \circ h_i, f_{L|H_i})$, and $\rho(g_{S_i} \circ h_i, g_{R|H_i})$ for $i = 1, \ldots, n$.

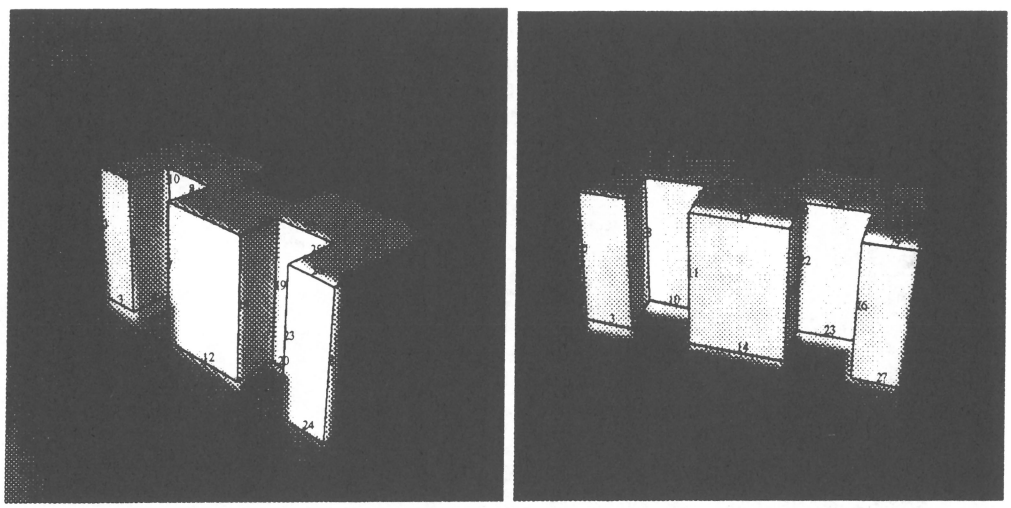


Fig. 16. Estimated Pose for Cube3Cut (Real Images). The figure shows the estimated wireframes found after matching 11 segments, superimposed on the corresponding image.

Now, the problem of finding the statistics P_U, P_S, P_{f_U}, and P_{g_S} reduces to the well-known problem of estimating the parameters of normal distributions given sets of n samples. A detailed treatment of this topic can be found in statistics textbooks [16].

8. Experiments

In our experiments we used a CCD camera with focal length 4.8 mm and a resolution of 1.25901 mm/pixel × 1.18758 mm/pixel. The light is a point source of unpolarized light, of intensity 1 cd, located at a fixed position.

The set of features L is made up of 2D-segments, projections of the 3D-segments forming the objects. The feature attribute mapping f_L assigns with each label l, four attributes: its midpoint image coordinates, x_m^l and, y_m^l, its length, λ^l, and its orientation α^l. Each label attribute value is given by a mean and a standard deviation representing the variations of the attribute among the different predictions used to obtain the model. The set of units U is the set of 2D-segments forming the image to be matched. The feature attribute mapping f_U associates to each unit u four attributes: its midpoint image coordinates, x_m^u and, y_m^u, its length, λ^u, and its orientation α^u. The set of relational tuples of segments R and S are formed by three different types of relationships: *junctions* of two segments, *junctions* of three segments, and *triples* of segments. A *junction* of two/three segments is an ordered set of two/three lines which meet at a common endpoint. The segments are ordered such that the angles between the segments are less than 180 degrees when the lines are traced clockwise. A *triple* of

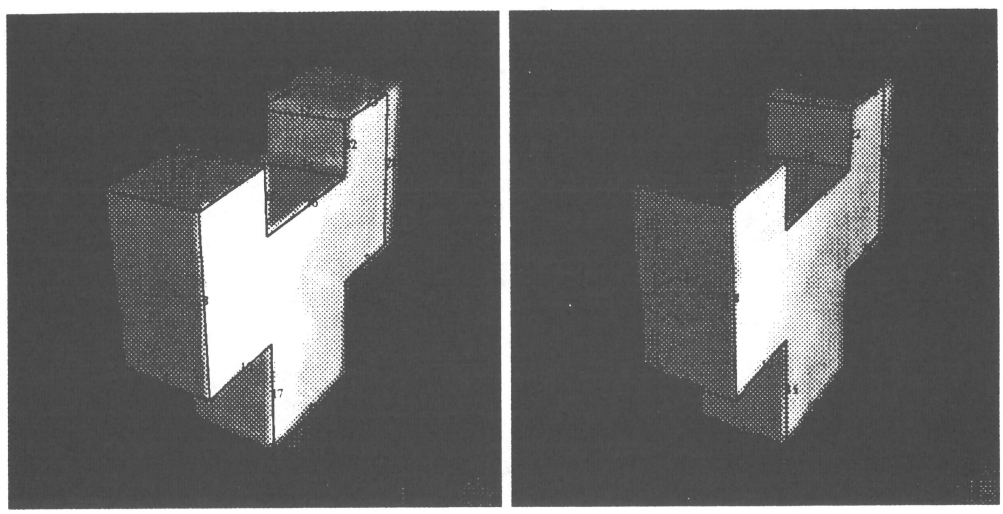


Fig. 17. Estimated Pose for Fork (Real Images). The figure shows the estimated wireframes found after matching 10 segments, superimposed on the corresponding image.

segments [12] is defined as an ordered set of three lines, two pairs of which meet at a junction. The angles at the two junctions must both be less than 180 degrees when the lines are traced clockwise, so the triple has a well defined "inside". For this set of experiments, the relationship strength mappings g_R and g_S were not used. Conceptually, this amounts to having constant relationship strength mappings.

In order to compare the attributes of the labels in the set L and those of the corresponding units in the set U, for a given observation mapping h, the following feature metric error was used:

$$E_{f_U}(h) = \rho(f_U \circ h, f_L|H) = \sum_{l \in H} \rho_{lu}(l, h(l)), \tag{33}$$

with

$$\rho_{lu}(l, h(l)) = \sqrt{\rho_x^2(l, h(l)) + \rho_y^2(l, h(l)) + \rho_\lambda^2(l, h(l)) + \rho_\alpha^2(l, h(l))}, \tag{34}$$

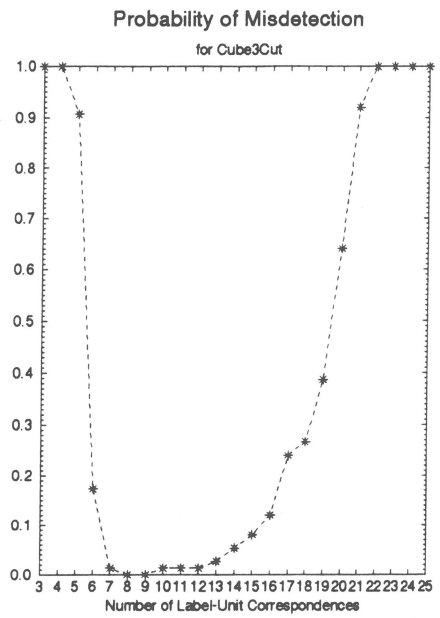


Fig. 18. Misdetection Probability for Cube3Cut. The misdetection probability is plotted against the number of correspondences sought.

where

$$\rho_x(l,u) = \frac{|\overline{x_m^l} - x_m^u|}{\sigma_{x_m}^l},$$

$$\rho_y(l,u) = \frac{|\overline{y_m^l} - y_m^u|}{\sigma_{y_m}^l},$$

$$\rho_\lambda(l,u) = \frac{|\overline{\lambda^l} - \lambda^u|}{\sigma_\lambda^l},$$

$$\rho_\alpha(l,u) = \frac{|\overline{\alpha^l} - \alpha^u|}{\sigma_\alpha^l},$$

$((\overline{x_m^l}, \sigma_{x_m}^l)$, $(\overline{y_m^l}, \sigma_{y_m}^l)$, $(\overline{\lambda^l}, \sigma_\lambda^l)$, $(\overline{\alpha^l}, \sigma_\alpha^l))$ are the attribute values of label l, $(x_m^u, y_m^u, \lambda^u, \alpha^u)$ are the attribute values of unit $h(l)$, and H is the domain of the mapping h.

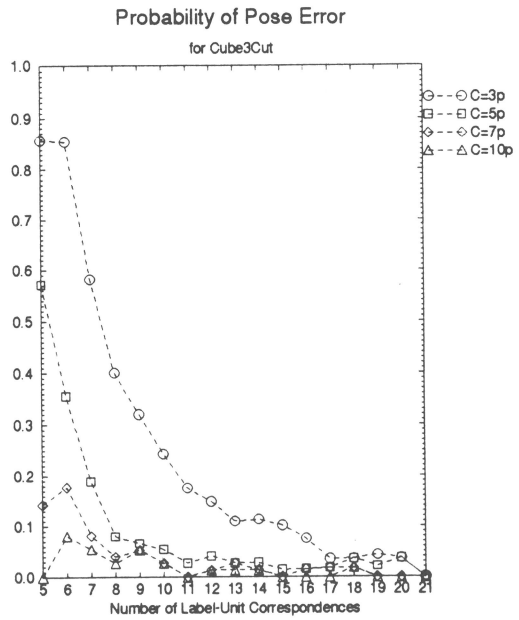


Fig. 19. Pose Error Probability for the Cube3Cut Model. The plots show the pose error conditional probability, given that a camera position was found, for the different numbers of correspondences sought, parameterized on the accuracy criterion C.

Figure 14 shows images of *Cube3Cut* and *Fork*, two of the objects modeled in PREMIO. In what follows we describe the results of a series of experiments with probabilistic models of Cube3Cut and Fork combining over a hundred of predictions for each. The predictions for Cube3Cut were generated with the object located at the origin, the light fixed at (−3.0 cm, −2.0 cm, 60.0 cm), and the camera moving on a sphere of radius $R = 35.3857$ cm, with longitude $20° \leq \Phi_v \leq 70°$ and latitude $20° \leq \theta_v \leq 70°$. The predictions for Fork were generated with the object located at the origin, the light fixed at (62.5 cm, −8.0 cm, 6.0 cm), and the camera moving on a sphere of radius $R = 46.1071$ cm, with longitude $290° \leq \Phi_v \leq 340°$, and latitude $30° \leq \theta_v \leq 90°$. The minimum feature detectability was set to $t_f = 0.0$, and the minimum relational detectability was set to $t_R = 0.15$ for both objects. Figure 15 shows line drawings of the model M for Cube3Cut and Fork. The segments are shown with their mean attributes, labeled in descending order of detectability. The parameters for the error distributions are: $\mu_f = 9.6875$, $\sigma_f = 2.5042$, $\mu_R = 10.4107$, $\sigma_R = 3.3867$ (junctions of two segments), $\mu_R = 7.9375$, $\sigma_R = 1.5024$ (junctions of three segments), and $\mu_R = 17.9107$, $\sigma_R = 5.6226$ (triples), and $\mu_{f_U} = 32.5366$, $\sigma_{f_U} = 8.9240$ for

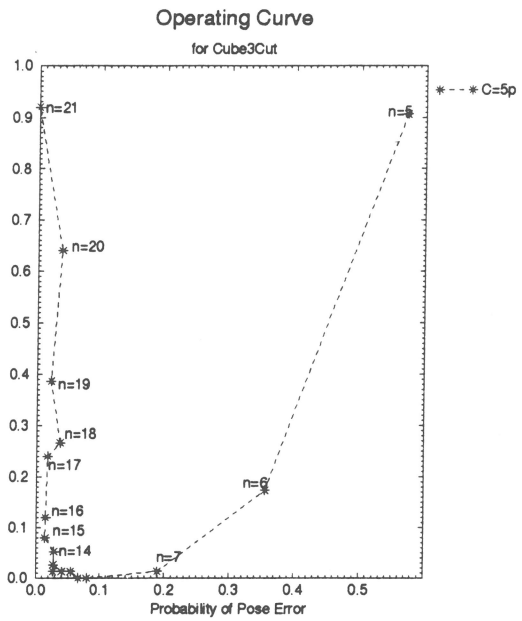


Fig. 20. Matching Operating Curve for Cube3Cut for $C = 5$ pixels. The plots shows the misdetection probability versus the pose error probability, given that a camera position was found, parameterized by the number of correspondences sought, for an accuracy criterion of $C = 5$ pixels.

Cube3Cut and $\mu_f = 5.9773$, $\sigma_f = 2.0582$, $\mu_R = 7.4091$, $\sigma_R = 1.7818$ (junctions of two segments), $\mu_R = 8.2803$, $\sigma_R = 1.1278$ (junctions of three segments), and $\mu_R = 17.0530$, $\sigma_R = 3.8706$ (triples), and $\mu_{f_U} = 27.7983$, $\sigma_{f_U} = 7.3375$ for Fork.

8.1. PERFORMANCE EVALUATION

The obtained models were matched against artificial and real images, varying the number of correspondences sought, producing an observation mapping for each experiment. The observation mappings found by the matching algorithm were used to estimate the camera location with a robust iterative linearized least-squares algorithm [10]. Figures 16 and 17 show estimated wireframes for 10 and 11 correspondences superimposed on the corresponding real images for Cube3Cut and Fork, respectively.

The pose estimation algorithm requires at least four correspondences between 2D *points* in the image and 3D *points* on the object. Thus, the correspondences

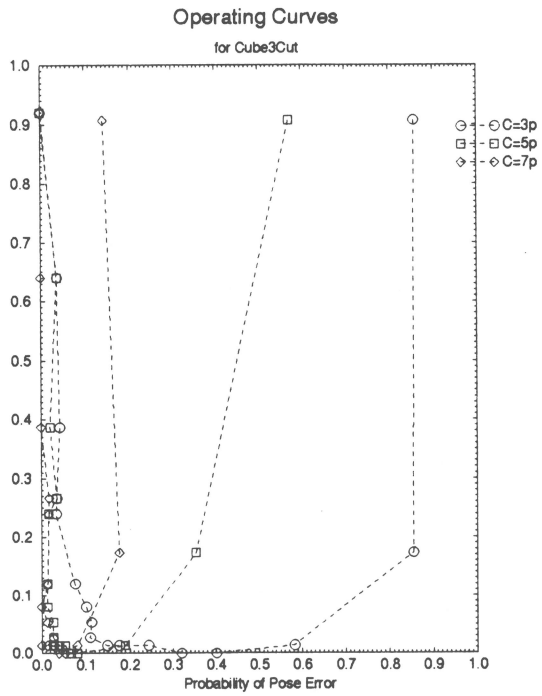


Fig. 21. Matching Operating Curves for Cube3Cut for $C = 3$, $C = 5$ and $C = 7$. The plots show the probability of misdetection versus the probability of pose error, given that a camera position was found, parameterized by the number of correspondences sought, for accuracy values of $C = 3$, $C = 5$ and $C = 7$.

found between labels (2D segments) and units (2D segments) had to be used to determine correspondences between 2D points in the image and 3D points in the object. Points in the image are determined by junctions of 2 and 3 segments. Points in the object are vertices of the object. Thus, a 2D point determined by a junction of units corresponds to a 3D point in the object if at least two of the units forming the junction correspond to two labels that have associated 3D segments having the 3D point in common.

If the number of point correspondences found is less than or equal to three, the pose estimation algorithm cannot determine the camera location, and the experiment is referred to as a *misdetection* error. Figure 18 shows a plot of the misdetection probability versus the number of correspondences sought, for the artificial test images of Cube3Cut. In the beginning, as the number of correspondences between labels and units increases, the number of junctions of 2 and 3 segments that are matched increases and the probability of misdetection, that is the probability of finding three or less point correspondences, decreases. However, for $n \geq 9$, the matching algorithm fails, for some images, to find n or more correspondences between labels and units, and therefore the probability of misdetection starts to increase again.

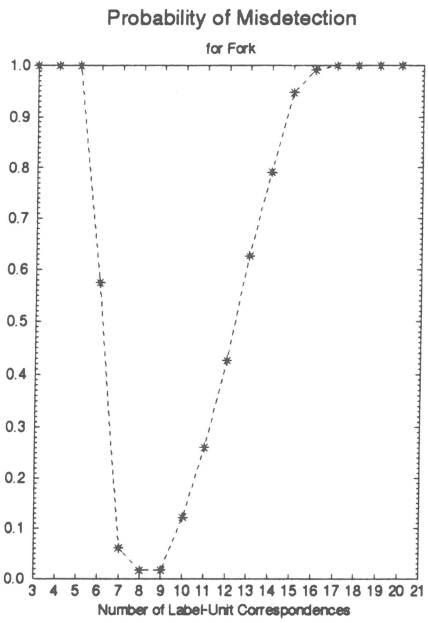


Fig. 22. Misdetection Probability for Fork. The misdetection probability is plotted against the number of correspondences sought.

If the number of point correspondences found is greater than or equal to four, the pose estimation algorithm computes the camera location and orientation. The position error is defined as the average distance, measured in pixels, between the vertices of the wireframe resulting of projecting the object with the *true camera position* and the corresponding vertices of the wireframe obtained by projecting the object with the *estimated camera position*:

$$\text{Position Error} = \sum_{v} \frac{||\text{True Vertex} - \text{Computed Vertex}||_2}{\text{Number of Vertices}}.$$

If the position error of an image is larger than an accuracy criterion C, the observation mapping found is declared incorrect and the experiment is referred to as a *pose error*. Figure 19 shows plots of the pose error conditional probability, given that a camera position was found, versus the number of correspondences sought, parameterized on the accuracy criterion C.

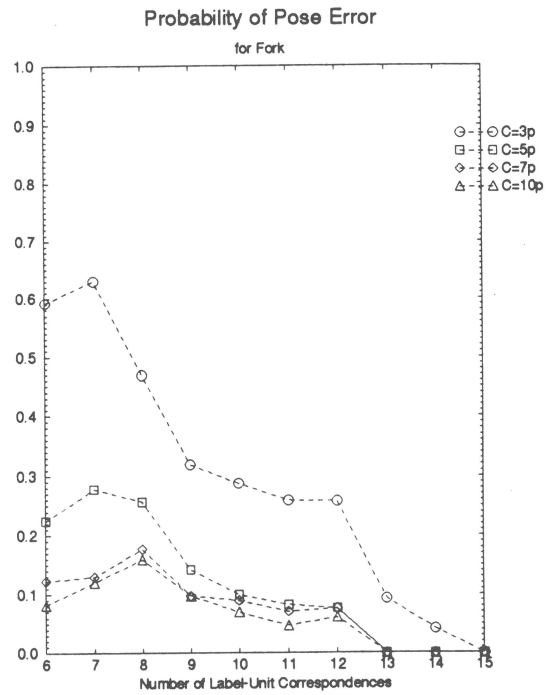


Fig. 23. Pose Error Probability for the Fork Model. The plots show the pose error conditional probability, given that a camera position was found, for the different numbers of correspondences sought, parameterized on the accuracy criterion C.

As the number of correspondences between labels and units increases, the number of junctions of 2 and 3 segments matched increases, and therefore we can expect the pose estimation algorithm to be more accurate and the probability of pose error to decrease.

The plots shown in figs. 18 and 19 are combined into a single plot, for each value of C, by plotting the probability of misdetection versus the pose error probability. The curves obtained in this way are traditionally called the "receiver operating curves (ROC)" or simply the operating curves. Each point in a curve corresponds to the number of correspondences sought during the matching, and it has associated a probability of misdetection and a pose error probability.

The operating curve for the experiments performed and an accuracy value of $C = 5$ pixels is shown in fig. 20. Let n be the number of correspondences sought. For $n = 5$, the probability of misdetection is high. If too few labels and units are matched, few point correspondences are found, resulting in a high misdetection rate. Furthermore, for those images where the camera position is found, the pose error is high due to the small number of points used in the estimation. For

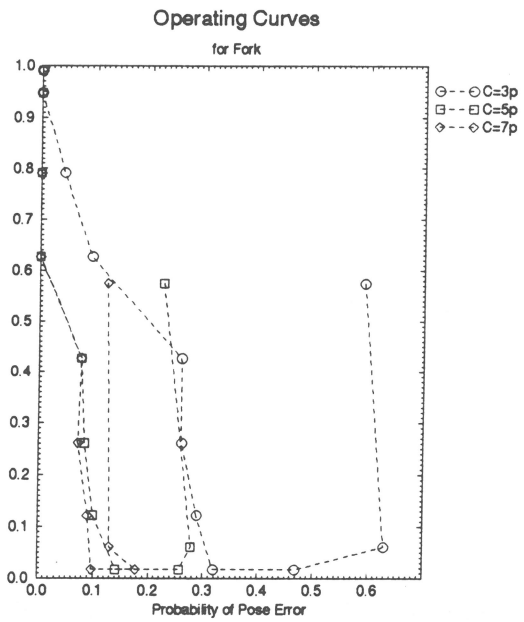


Fig. 24. Matching Operating Curves for Fork. The plots show the probability of misdetection versus the probability of pose error, given that a camera position was found, parameterized by the number of correspondences sought, for accuracy values of $C = 3$, $C = 5$ and $C = 7$.

$5 \leq n \leq 8$, the number of point correspondences found increases, and both probabilities decrease. For $n = 8$ and $n = 9$, there are always at least four point correspondences found, and therefore the rate of misdetection is zero. The more correspondences found, the more accurate the computed camera position is, and the smaller the pose error conditional probability is. For $n > 9$, there are images for which the matching routine cannot find n correspondences, and hence the misdetection rate increases. However, for those images where a set of correspondences is found, the accuracy of the computed camera increases, since more point correspondences are available, and therefore the pose error probability decreases. For $n > 14$, the probability of misdetection increases, while the probability of pose error remains approximately constant.

In general, we would like the system to have both low probability of misdetection, *and* probability of pose error, that is we would like the system to operate on the point of the operating curve that is the closest to the origin. For example, for an accuracy of $C = 5$ pixels the optimal operating point is for $n = 12$. At this point the probability of misdetection is equal to 0.013 and the probability of a pose error larger than 5 pixels is 0.04. Figure 21 shows the operating curves for Cube3Cut for the accuracy values $C = 3$, $C = 5$ and $C = 7$ pixels.

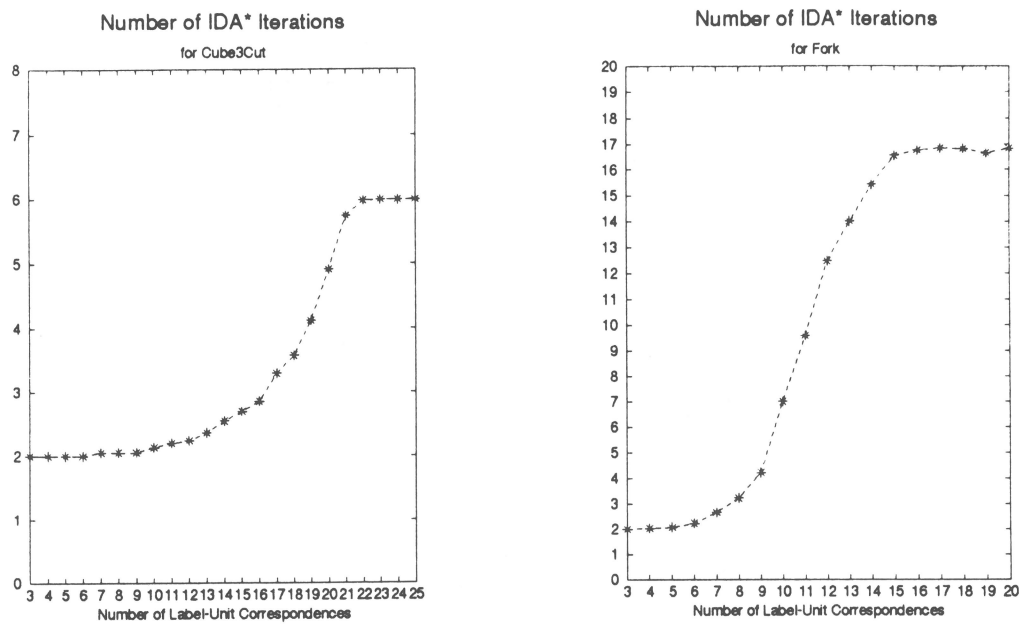


Fig. 25. ID A^* Number of Iterations. The figure shows the average number of iterations during the execution of the iterative-deepening-A^* algorithm for Cube3Cut and Fork. The number of iterations grows exponentially with the number of correspondences sought. The leveling off after $n = 21$ for Cube3Cut and $n = 15$ for Fork is caused by forced termination of the matching program after exceeding a given execution time limit of 2 minutes.

The plots for the probability of misdetection, probability of pose error, and the operating curves for Fork are given in figs. 22 to 24.

8.2. TREE SEARCH EFFICIENCY EVALUATION

In this section we will examine the efficiency of the ID A^* search algorithm for matching, in terms of the number of iterations, the number of paths opened and pruned, and the execution time.

Figure 25 shows plots of the average number of iterations during the execution of the algorithm for Cube3Cut and Fork. The number of iterations grows exponentially with the number of correspondences sought, n. However, the rate of growth is slow, for n below 17 and 10 for Cube3Cut and Fork, respectively.

Figure 26 shows plots of the average number of opened and pruned paths for Cube3Cut and Fork. The number of opened paths grows exponentially with the number of correspondences sought. However, the rate of growth is slow for a number of correspondences less than 17 for Cube3Cut and 10 for Fork. Furthermore, the number of pruned paths also grows exponentially, at a slower rate.

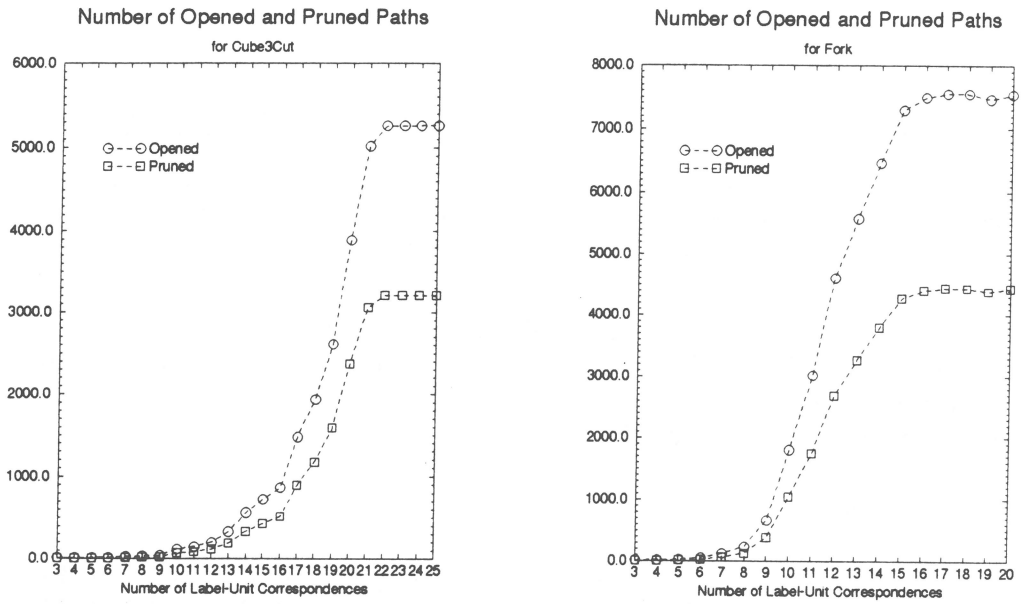


Fig. 26. ID A^* Number of Opened and Pruned Paths. The figure shows the average number of opened and pruned paths in the tree during the execution of the iterative-deepening-A^* algorithm for Cube3Cut and Fork. The number of opened and pruned paths grows exponentially with the number of correspondences sought. However, the rate of growth is slow for a number of correspondences less than 17 for Cube3Cut and 10 for Fork. The leveling off after $n = 21$ for Cube3Cut and $n = 15$ for Fork is caused by forced termination of the matching program after exceeding a given execution time limit of 2 minutes.

Figure 27 shows the average percentage of pruned paths. The pruning ratio is defined as the percentage of pruned paths relative to the total number of opened paths. The pruning ratio for Cube3Cut is between 40% and 60% and for Fork is approximately 50%.

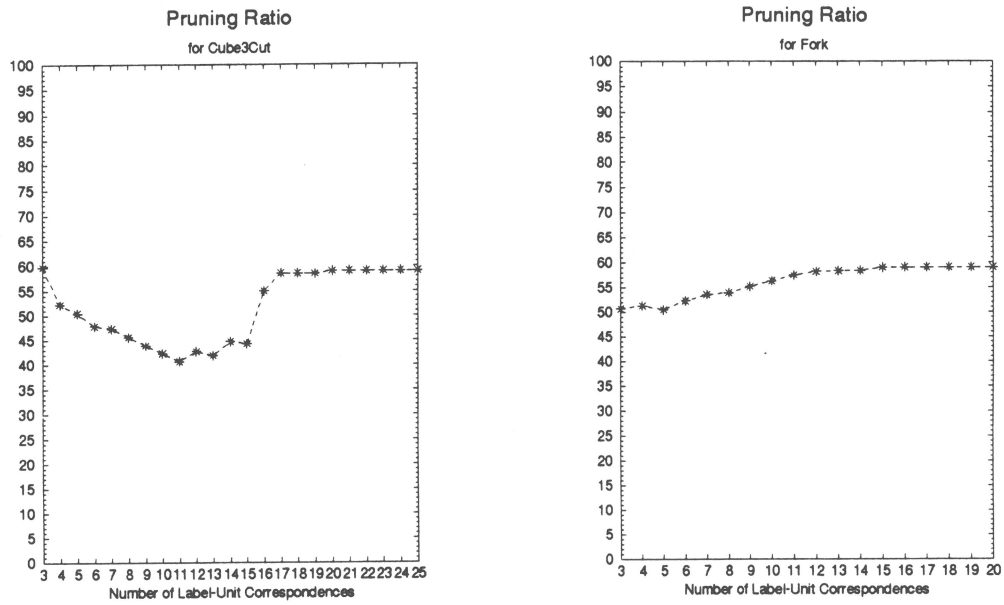


Fig. 27. ID A^* Path Pruning Ratio. The figure shows that the average percent of opened paths that are pruned by the iterative-deepening-A^* algorithm is always between 40% and 60% for Cube3Cut and it is approximately 50% for Fork.

The average execution time for the matching algorithm for Cube3Cut and Fork is shown in fig. 28. The execution time grows exponentially with the number of correspondences. The average CPU time is less than 1 second for a number of correspondences sought under 7 for Cube3Cut and 6 for Fork.

9 Conclusion

In this paper we have posed the relational matching problem as a special case of the pattern complex recognition problem. This probabilistic approach allowed us to make explicit statements about how an image is formed from a model, and hence to define a natural matching cost that can be used to find the best observation mapping. Furthermore, we have showed how to find an underestimate of this cost. The relational matching cost and its underestimate are used to guide an iterative-deepening-A^* (ID A^*) heuristic search procedure in finding a set of correspondences between the model and the image.

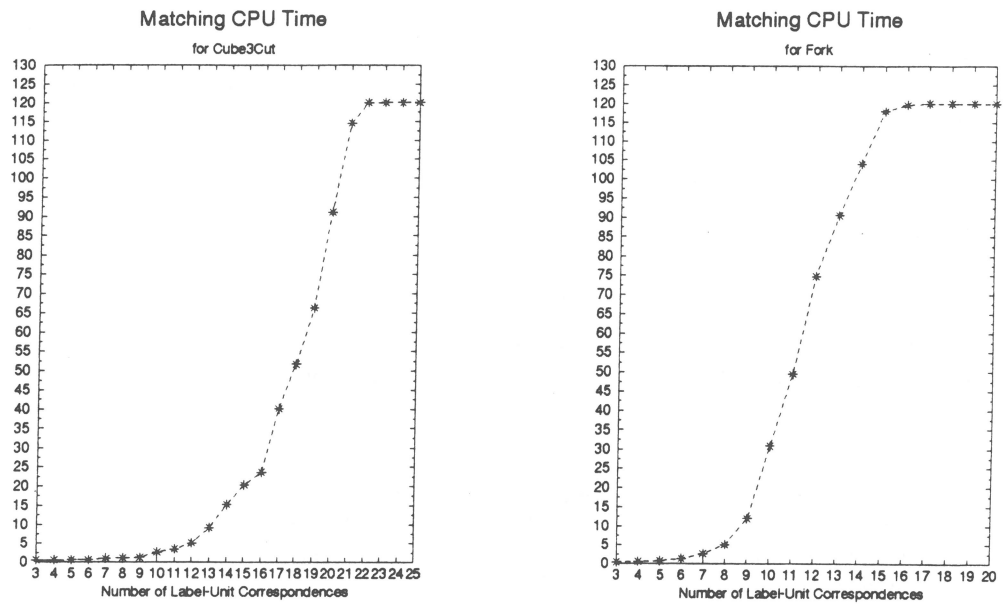


Fig. 28. ID A^* CPU Time. The figure shows the average CPU time for the execution of the iterative-deepening-A^* algorithm for Cube3Cut and Fork. The time grows exponentially with the number of correspondences sought. However, the rate of growth is slow with the time of execution being under a second when the number of correspondences sought is less than 7 for Cube3Cut and 6 for Fork. The leveling off after $n = 21$ for Cube3Cut and $n = 15$ for Fork is caused by forced termination of the matching program after exceeding a given execution time limit of 2 minutes.

The matching algorithm was tested on real and simulated data. The experiments with real images indicate that the system performs well in a real environment. The simulated data was used to obtain the operating curves of the algorithm that characterize its performance. The experiments showed that even though the nature of the feature matching problem is exponential, the use of the ID A^* matching algorithm keeps the size of the problem under control, by opening fewer nodes and pruning between 40% and 60% of them. Specifically, for Cube3Cut and the optimal number of correspondences ($n = 12$), the number of paths opened by the matching algorithm was, on the average, 202. Moreover, 116 of these paths were pruned (on the average). This should be compared with the 27^{12} possible paths to be explored by an exhaustive search. Similar results were obtained for Fork (optimal $n = 9$), with 668 paths opened (out of a possible 20^9) and 387 pruned.

References

[1] R. Bolles and R. Cain, Recognizing and locating partially visible objects: The local-feature focus method, Int. J. Robot Res. 1(3) (1982) 57–82.

[2] K.L. Boyer and A.C. Kak, Structural stereopsis for 3-d vision, IEEE Trans. Syst. Man Cybern. SCM-110(2) (1988) 144–166.
[3] R. Brooks, Symbolic reasoning among 3-D models and 2-D images, Artificial Intelligence 17 (1981) 285–348.
[4] O.I. Camps, PREMIO: The use of prediction in a CAD-model-based vision system, Ph.D. Thesis, Department of Electrical Engineering, University of Washington, Seattle, Washington (1992).
[5] P.J. Flynn, CAD-based computer vision: Modeling and recognition strategies, Ph.D. Thesis, Michigan State University (1990).
[6] C. Goad, Special purpose automatic programming for 3D model-based vision, in: *Proc. of the Image Understanding Workshop*, June 1983, pp. 94–104.
[7] W.E.L. Grimson, The combinatorics of object recognition in cluttered environments using constrained search, in: *Proc. Int. Conf. on Computer Vision*, 1988, pp. 218–227.
[8] C.D. Hansen, CAGD-based computer vision: The automatic generation of recognition strategies, Ph.D. Thesis, The University of Utah (1988).
[9] R. Haralick, The pattern complex, in: *Structural Pattern Analysis*, eds. R. Mohr, T. Pavlidis and A. Sanfeliu (World Scientific, Singapore, 1989) pp. 57–66.
[10] R. Haralick and L. Shapiro, *Computer and Robot Vision* (Addison-Wesley, New York, 1992).
[11] R. Haralick and L.G. Shapiro, The consistent labeling problem: part i, IEEE Trans. Pattern Anal. Machine Intellig., PAMI-1(2) (1979) 173–184.
[12] J. Henikoff and L. Shapiro, Interesting patterns for model-based matching, in: *ICCV*, 1990.
[13] P. Horaud and R. Bolles, 3DPO: A system for matching 3-D objects in range data, in: *From Pixels to Predicates*, ed. A. Pentland (Ablex, Norwood, New Jersey, 1986) pp. 359–370.
[14] K. Ikeuchi, Generating an interpretation tree from a CAD model for 3D-Object recognition in bin-picking tasks, Int. J. Comp. Vision 1(2) (1987) 145–165.
[15] R.E. Korf, Search: A survey of recent results, in: *Exploring Artificial Intelligence* eds. H.E. Shrobe and T.A.A. for Artificial Intelligence (Morgan Kaufmann, 1988) chap. 6, pp. 197–237.
[16] L. Ott, *An Introduction to Statistical Methods and Data Analysis*, 2nd Ed. (Duxbury Press, Boston, MA, 1984).
[17] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd Ed. (McGraw-Hill, New York, 1991).
[18] A. Rosenfeld, R.A. Hummel and S.W. Zucker, Scene labeling by relaxation operations, IEEE Trans. Syst. Man Cybern. SMC-06 (June 1976).
[19] A. Sanfeliu and K.S. Fu, A distance measure between attributed relational graphs for pattern recognition, IEEE Trans. Syst. Man and Cybern. SMC-13(13) (1983) 353–362.
[20] L. Shapiro and R. Haralick, Structural descriptions and inexact matching, IEEE Trans. Pattern Anal. Machine Intellig. PAMI-3(5) (1981) 504–519.
[21] L. Shapiro and R. Haralick, A metric for comparing relational descriptions, IEEE Trans. Pattern Anal. Machine Intellig. PAMI-7 (1985).
[22] J.R. Ullman, An algorithm for subgraph homomorphisms, J. Assoc. Comput. Mach. 23 (1976) 31–42.
[23] P. Winston, *Artificial Intelligence*, 2nd Ed. (Addison-Wesley, New York, 1984).