ELSEVIER

Artificial Intelligence 78 (1995) 289-326

Artificial Intelligence

Recognition of object classes from range data

I.D. Reid*, J.M. Brady

Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, UK

Received September 1993; revised October 1994

Abstract

We develop techniques for recognizing instances of 3D object classes (which may consist of multiple and/or repeated sub-parts with internal degrees of freedom, linked by parameterized transformations), from sets of 3D feature observations. Recognition of a class instance is structured as a search of an interpretation tree in which geometric constraints on pairs of sensed features not only prune the tree, but are used to determine upper and lower bounds on the model parameter values of the instance. A real-valued constraint propagation network unifies the representations of the model parameters, model constraints and feature constraints, and provides a simple and effective mechanism for accessing and updating parameter values.

Recognition of objects with multiple internal degrees of freedom, including non-uniform scaling and stretching, articulations, and sub-part repetitions, is demonstrated and analysed for two different types of real range data: 3D edge fragments from a stereo vision system, and position/surface normal data derived from planar patches extracted from a range image.

Keywords: Object recognition; Parametric objects; Range data; Stereo

1. Introduction

Consider a robot performing a task in an unknown or partially known environment. If it is to react to that environment in other than a haphazard "trial-and-error" fashion, the robot must learn about its surroundings using sensed data. Perhaps the simplest requirement is to avoid obstacles; for this task the robot need know nothing more about obstacles than roughly their positions and extents. A considerably more complicated task involves finding and manipulating a specific object in the environment. In this case the robot must not only locate an object, but also recognize it. Recognition demands the use of both sensed data and prior knowledge about the object in order to detect its

* Telephone: +44-1865 273168. Fax: +44-1865 273908. E-mail: ian@uk.ac.ox.robots.

SSDI 0004-3702(95)00062-3

Fig. 1. Two typical pallets. The one the left is considerably larger and has six slats and four struts. The smaller pallet has five slats and three struts.

presence, and to determine its pose (position and orientation) relative to the sensor. The model-based vision paradigm addresses just this problem.

Model-based vision embraces a class of techniques which achieve object recognition by encoding high-level prior knowledge about objects in models which describe the objects' observable properties, and comparing the models with sensed data to find a "best interpretation" of a scene, given the sensed data. A common approach formulates the problem as one of finding low-level features in sensory data (such as surfaces or edges), determining correspondences (subject to certain matching constraints) between these features and object features represented in a model, and then computing a transformation from a model-centred reference frame to the reference frame of the sensor(s). Such a formulation fits into a class known as constraint satisfaction problems. The paradigm involves some of the major research problems in artificial intelligence; those of representation, search strategies, interpretation of sensory data, and handling uncertainty.

The vast majority of recognition systems built to date—whether they use single intensity views of objects, or three-dimensional range data—have been designed to operate with geometrically fixed objects (e.g. [8, 15, 22, 24]) or objects with only a few internal degrees of freedom [14, 19, 40]. While such systems have been successfully applied to a number of useful robotic and automation tasks, many other applications involve the need to recognize members of object classes for which a fixed geometry cannot be defined *a priori*. A good example of such, and which has motivated the research we report, is the use of a mobile vehicle to locate and acquire industrial pallets. Fig. 1 depicts two such items, clearly showing that although they belong to the same

object class, they vary considerably in size, shape, and in the number of slats and struts they have. Such object classes are common, particularly in man-made environments. Most of the systems mentioned are unsuitable since each variation within a class would necessitate a separate model.

To address this deficiency we develop a 3D recognition system for recognizing polyhedral objects which may have multiple and repeated sub-parts which may move relative to one another, each of which may have multiple internal free parameters. Our system takes as input a set of primitive features—currently either 3D line segment data (from, for example, a stereo vision system) or surface patch data from a range finder (for example, a laser range triangulation system)—and determines the pose of an instance of a model class, at the same time placing upper and lower bounds on the parameter values of the instance, despite the presence of significant noise and occlusion. Furthermore, an additional level of competence allows the system to distinguish between different instances of the same class.

The system is based on three techniques, well established in their own right in the literature:

- interpretation tree search [17];
- binary geometric constraints [20];
- a continuous-valued constraint propagation network [14].

The use of the first two items in recognition systems has been well documented, particularly in [20]. Interpretations—branches of the tree—are grown by adding matches (or assignments) of observed features to model features which are pairwise consistent with those already in the interpretation. Pairwise consistency is defined using a set of viewpoint-invariant measurements (such as angles and distances) on pairs of features, and bounds on these measurements precomputed from a model.

Our system is designed to use either surface patch data (position/surface normal points) or straight 3D edge fragments. For either a pair of surface patches or a pair of edge fragments, there exist four independent invariants, consisting of one angle and three distance measurements. The sets of invariants used in our system are given in Appendix A.

The novelty of our system comes through the combination of the first two items above with the third item, which is a powerful way of representing and solving sets of inequality constraints. So-called SUP/INF networks were developed by Fisher and Orr [14] based on the work of Brooks [6], who, in the ACRONYM system for recognition of 3D models from 2D images, was the first to introduce the idea of symbolic constraint satisfaction for object recognition. A set of inequality constraints on the model and on the viewing parameters was solved symbolically to give the pose and internal parameters of an instance of a parametric object class. Fisher and Orr's development was to show how much of the work involved in such symbolic manipulation can be performed off-line by compiling the symbolic constraints into a real-valued constraint propagation network. They used the resulting networks to solve for object pose and some (limited) internal transformations such as articulations, gaining an improvement in performance over the purely symbolic approach of ACRONYM.

For linear constraints, the SUP/INF method is guaranteed to give a correct solution. However the major weakness of both the Brooks and the Fisher/Orr approaches is that

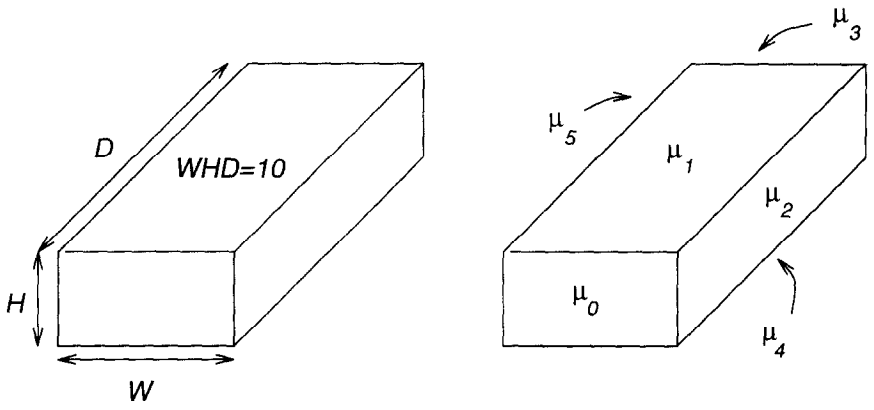


Fig. 2. A model class of equal volume boxes: (a) a parameterization and constraints; (b) model surfaces.

constraints on object pose result in complex nonlinear inequalities which often lead to poor (sometimes useless) bounds on the pose transformation parameters. Indeed in later work Orr et al. have argued for a probabilistic rather than interval representation of uncertainty in order to obtain more realistic pose computations [30].

In our method, the pose computation and internal parameter estimation are separated. The constraint network is used to store internal model parameter values (upper and lower bounds) and define relations between parameters, which are updated and propagated as an interpretation grows. This parameter representation is ideally suited to the application, as we show in later sections. However the network is *not* used for pose computation, which is performed using a two-stage least-squares method by first finding a best-fit rotation [11] followed by a best-fit translation [15]. This trade-off makes full use of the benefits of the SUP/INF network without suffering from its major drawback.

Prior to detailed description, in order to give the reader an intuitive feel for the system's operation, we illustrate the general principles by example. Consider the class of equal volume boxes depicted in Fig. 2. This class is characterized by its three free parameters, H, W and D, the height, width and depth of the box. Here, the angles between the surfaces are constant values, independent of the model parameters. However, for example, the distance d of a point on the top surface from the plane defined by the front surface (invariant f_2 for surfaces, surf-dist-1, in Appendix A) is constrained by $0 \leqslant d \leqslant D$, where D is the depth of the box, initially unknown.

We draw two major insights from this, which form the basis for the remainder of this work. These are:

- $d \in [0, D]$ is a *symbolic* bound. Algorithms exist for solving symbolic sets of equations, albeit rather less efficiently than numeric ones; we exploit one such method in our algorithm.
- The measurement actually gives information about the unknown parameter value, namely that $D \geqslant d$; i.e. the measurement d provides a lower bound on the true value of D.

Suppose then, that the system is exploring the branch of the interpretation tree shown in Fig. 3. At Level 2 of the tree the assignment $s_1 \rightarrow \mu_0$ is considered. Initially there is

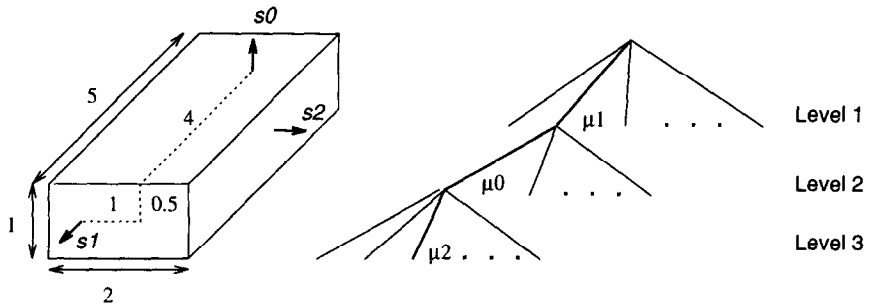


Fig. 3. Constraining the interpretation tree search and parameter values: (a) observations from a box instance; (b) exploration of the interpretation tree.

no knowledge of the parameter values, i.e.:

Initially: $W \in [0,\infty], \quad H \in [0,\infty], \quad D \in [0,\infty].$

Three distance invariants with values 1, 0.5 and 4 may be computed from the pair of surface patch observations s_0 and s_1 (see Fig. 3 and Appendix A). Therefore the pairwise consistency of the matches $s_0 \rightarrow \mu_1, s_1 \rightarrow \mu_0$ is determined by the three expressions:

Test: $1 \in [0,\infty], \quad 0.5 \in [0,\infty], \quad 4 \in [0,\infty],$

which are all clearly satisfied. The match $s_1 \rightarrow \mu_0$ is therefore accepted, but we now note that this match implies that $W \geqslant 1$, $H \geqslant 0.5$ and $D \geqslant 4$, thus the lower bounds can be updated:

Update: $W \in [1,\infty], \quad H \in [0.5,\infty], \quad D \in [4,\infty].$

Furthermore, once this update from the observations is complete, enforcement of the model constraint $WHD = 10$ (i.e. the equal volume condition, which is equivalent to the two conditions $WHD \geqslant 10$ and $WHD \leqslant 10$) leads to upper bound estimates for the parameters:

Propagate:
$$W \leqslant 10/HD \Rightarrow W \leqslant 5,$$
$$H \leqslant 10/WD \Rightarrow H \leqslant 2.5,$$
$$D \leqslant 10/WH \Rightarrow D \leqslant 20.$$

So

$$W \in [1,5], \quad H \in [0.5,2.5], \quad D \in [4,20].$$

Subsequent assignments (e.g. $s_2 \rightarrow \mu_2$) provide further evidence of the parameter values, gradually improving the bounds.

Thus our system is based around an observe–test–update cycle, in which we test an observation for consistency with the current size/shape estimate of the model, and if consistent, use the observation to update the estimates of size/shape. In the following sections these ideas are formalized and generalized, and used to build a general-purpose parametric object recognition system.

Initially we consider the case of finding an instance of a single object class consisting of one part. However in later sections we extend the system in a number of ways. Firstly, in Section 4, we show how different instances of the same object class may be distinguished and identified. Secondly, in Section 5 we show how the system copes with multiple sub-parts and even variable numbers of repetitions of sub-parts. Finally, we give a performance analysis of the system (Section 6) and show that the performance compares favourably with systems which enforce strict model rigidity. We place the work in the context of previous efforts in Section 7 and conclude in Section 8.

2. Principles

2.1. Interpretation tree search

At the highest level, the basis of our system is the well known hypothesize/test paradigm common to most recognition systems (e.g. [5,11,21,23,24]). A search of an interpretation tree, using geometric constraints for pruning, generates pairwise consistent matches of sensed features to model features and computes a pose hypothesis from these matches. Previously, Grimson [20] has shown that for geometrically fixed objects, binary geometric constraints between pairs of data to model feature matches are a surprisingly powerful tool for pruning the exponential search space of the interpretation tree. Usually the number of hypotheses which must be tested is reduced to a handful; often only one. A major contribution of our work is to show that this is still the case even for parameterized objects (see Section 6).

Formally, an object is modelled by a set M of m model features, (e.g. the object's surfaces or edges):

$$M = \bigcup_{i=1}^{m} \mu_i.$$

When observed in the world an object gives rise to a set S of n sensed features:

$$S = \bigcup_{j=1}^{n} s_j.$$

Only rarely will a full model feature be observed. More typically, due to occlusion and noise, only a portion of the feature will be observed; e.g. a small surface patch or an edge fragment. In our work we consider either type of input data: either surface data from our in-house (Oxford/NEL) laser range finder [33,35], or 3D edge fragments from a stereo vision system [31]. Surface data are represented by a 3D position and unit surface normal, $s = (\boldsymbol{p}, \boldsymbol{n})$. This representation is rather conservative since the dense range maps supplied by the Oxford/NEL sensor give information about surface extent and connectivity. We discuss this further in Section 3.5. Edge data are represented by a set of edge fragments, $s = (\boldsymbol{e}, \boldsymbol{m}, l)$; $\boldsymbol{e}$ is a unit vector in the direction of the fragment, $\boldsymbol{m}$ is its midpoint, and l is the fragment length.

Then, we define:

Definition 1. An *image measurement* or *measurable invariant* is a function f which maps a set of observable features to a scalar, independent of the viewpoint. In our work we make use of *binary* measurements (i.e. invariants from a pair of features), and in general several such independent measurements are available, with this set denoted by $\{f_k\}$. For pairs of either surface patches $(\boldsymbol{p}, \boldsymbol{n})$ or edge fragments $(\boldsymbol{e}, \boldsymbol{m}, l)$, there exist four independent invariants. We list those used in our system in Appendix A.

Consider applying the kth invariant to a pair of observations on a pair of model features. As the locations of the observations vary over the model features, so will the value of the invariant. Thus each pair of model features generates an interval giving the valid range of the kth invariant for that pair of model features. If we use the notation $v \in \mu$ to denote an observation on a model feature (for example, a position/surface normal point on a model surface) we can formalize the definition of this valid range:

Definition 2. The *feature constraint interval* for the kth invariant and the model features μ_i and μ_j is given by

$$Z_k(\mu_i, \mu_j) = \left[\min_{v \in \mu_i, w \in \mu_j} f_k(v, w), \max_{v \in \mu_i, w \in \mu_j} f_k(v, w) \right].$$

This interval is a function of the model. For geometrically fixed objects, these intervals can be precomputed as part of the object model as was done in [21]. In the case of parameterized models, they will often be known functions of the unknown model parameters.

Definition 3. A *feature constraint table* (or FCT) is a convenient storage method for feature constraint intervals in which entry (i, j) of table k refers to the interval $Z_k(\mu_i, \mu_j)$.

Definition 4. A *feature constraint* is a constraint on a potential match of a pair of sensed features to a pair of model features, $s_1 \rightarrow \mu_i$, $s_2 \rightarrow \mu_j$:

$$f_k(s_1, s_2) \in Z_k(\mu_i, \mu_j).$$

The left-hand side of the expression is a measurement derived from observations. The right-hand side is the feature constraint interval. The expression is true if the measured invariant lies within the valid range for the pair of model features μ_i and μ_j and if true for all invariants, then the match is *consistent.*

In practice sensing errors mean that a sensed feature s is a corruption of a true value $\hat{s}$. A consistency test which accounts for this must test the likelihood of $f_k(\hat{s}_1, \hat{s}_2) \in Z_k(\mu_i, \mu_j)$ given an uncertain (but hopefully nearby) measurement $f_k(s_1, s_2)$. In our system we use the method of [21] to model errors, by constructing an interval $F_k(s_1, s_2)$ around $f_k(s_1, s_2)$ which is known to contain the true value. Then, the kth feature constraint for the potential match $s_1 \rightarrow \mu_i$, $s_2 \rightarrow \mu_j$ becomes the intersection test:

$$F_k(s_1, s_2) \cap Z_k(\mu_i, \mu_j) \neq \emptyset.$$

```
interpret(I, j)
if j = n
    verify(I);
    return;
i ← 0;
while i < m
    if consistent(I, {s_j, μ_i})
        I ← I ∪ {s_j, μ_i}
        interpret(I, j + 1)
    i ← i + 1
return;
```

Fig. 4. Pseudo-code for a recursive version of the interpretation tree search. I is a list of data feature to model feature matches representing the current interpretation, and j is the level in the tree. The search begins with the call `interpret({},0)`.

Given these definitions, the interpretation tree search for geometrically fixed objects can be described by the pseudo-code in Fig. 4. The function `consistent` applies the rule in the equation above. The reader will note that there is no mechanism for dealing with outlying data in this search. Grimson [20] shows how this may be overcome by adding a so-called "null" feature to the search which always matches, however at the expense of greatly increasing the size of the search. In Section 7 and in the conclusions we discuss ways in which the outlier and clutter problem may be overcome. However the problem is not addressed in the work we report, this being the subject of on-going research.

2.2. *Model representation*

The previous section has established the familiar framework for recognition from either 3D edge fragments or surface patches which uses binary geometric constraints to control an interpretation tree search. It should be clear, however, that as it stands, this framework is insufficient for recognizing generic objects. In this section we give a specific definition of an object class suited to many parametric object recognition tasks, and then discuss the limitations of the framework described above with respect to this definition.

2.2.1. *Object classes*

We model an object class using:

- a set of sub-parts, each with a REV-graph boundary representation [2], i.e. the faces, edges and vertices of the sub-part, and the topological relationships between them; each sub-part is defined relative to a local coordinate system, and has an associated transformation (possibly parameterized) which defines its position relative to the model base-frame;

- a set of internal model parameters;
- a set of constraints on the parameters, called *model constraints*;
- a set of feature constraint tables which store information about the geometrical relationships between pairs of model features (including those which do not come from the same sub-part);
- a specialization hierarchy, with each sub-level of the hierarchy corresponding to a specialization of the parent through the addition of extra model constraints, active for that sub-model and all its children.

Multiple and repeated sub-parts are allowed, including parameterized numbers of repetitions. We discuss recognition of multiple sub-part objects with parameterized transformations between sub-parts and those with parameterized part repetitions in Section 5. We now discuss the salient features of the representation and elaborate on its various aspects.

The REV-graph of the object is used to define the underlying shape of the object. The planar equations of the surfaces, the directions of the edges, and the positions of the vertices, stored with the REV-graph, are all functions of the model parameters. For example in the box example of Fig. 2 the vertices would be described by the coordinates $\{(0,0,0), (0,0,D), (W,0,D), (W,0,0), (0,H,0), (0,H,D), (W,H,D), (W,H,0)\}$.

If the model is geometrically fixed then we have numeric values for W, H and D and it is straightforward to compute numeric upper and lower bounds on the binary invariants for storage in feature constraint tables (see Section 2.1). However, if the boundary features are functions of unknown parameters this computation must be performed symbolically, and entries in the feature constraint tables must be uninstantiated symbols rather than numbers. We discuss this in Section 3.1.

Model constraints are inequalities involving one or more of the model parameters (note that equalities may be represented by two inequalities). They are used for two separate purposes:

- to enforce dependencies between parameters;
- to impose size/shape constraints on the model.

Consider the square pyramid class shown in Fig. 5. A possible parameterization of the class is the set $\{b, h, \theta\}$. This parameterization is not independent, containing the implicit constraint $\tan\theta = 2h/b$. This is an example of the former use of model constraints. An example of the latter is the constraint $3h \geqslant b$, which restricts the shape of the class somewhat.

It should be clear that the mechanisms described in the preceding sub-sections (based on the work of Grimson and others) are incapable of recognizing instances of such a model class. Models can no longer be represented by constant feature constraint tables; the bounds are now functions of the uninstantiated model parameters. For example, if we define a class which is fixed but for a uniform scale factor, σ (the simplest possible parameterization), then each distance constraint now involves the unknown value of the parameter σ.

However, as we observed in the introduction, the invariant measurements are functions of the parameters of the instance being observed and, in principle, we should be able to recover the model parameters from observed data during the interpretation tree search. Grimson [20,22], in extensions to the RAF system [21], describes a method for coping

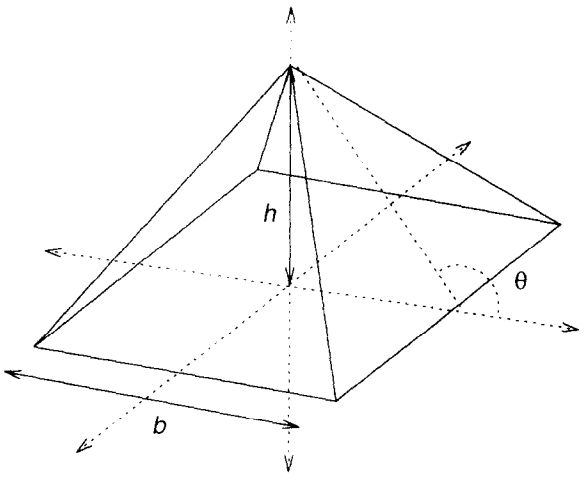


Fig. 5. A square pyramid class, parameterized by the set $\{h, b, \theta\}$.

with simple parameterizations—uniform scale and stretching and simple articulations—of 2D models. The method involves solving for parameter values explicitly from image measurements and modifying the search algorithm accordingly. The generalization to arbitrary parameterizations and 3D models was never achieved. Grimson states [20, p. 423] that,

> The main difficulty ... is finding a clean way of representing the parameterized constraints, especially in a manner that will easily allow the computing and updating of feasible ranges for each of the parameters.

In summary, the extensions required in order to cope with arbitrary and compound parameterizations are:

- a representation for model parameters;
- a way of defining and representing constraints on model parameters;
- a way of computing *symbolic* feature constraint tables off-line and then instantiating these at run-time;
- a way of updating parameter estimates based on sensed data (specifically, from invariant values);
- a way of determining if the constraints have been violated.

To this end we next present a representation for parameters and model constraints based upon Fisher's network implementation [12–14] of the SUP/INF method [4,6]. In addition to handling all the cases Grimson describes, the representation is general enough to specify arbitrary scaling (not only uniform scaling) in full 3D, articulations and other linear and nonlinear constraints on model parameters.

2.3. SUP/INF algorithm

The SUP/INF algorithm, devised by Bledsoe [4] and refined by Brooks [6,7], is a method for solving symbolically a set of inequality constraints. It is based on recursive

application of two functions, sup (Supremum) and inf (Infimum), which find upper and lower bounds, respectively, on the free variables in the constraints. The goal of the algorithm is to determine whether or not a solution can be achieved, and if so, over what ranges of the variables the constraints can be satisfied.

Brooks first used this algorithm in the ACRONYM system [6] for recognition of 3D objects from 2D data. In ACRONYM the recognition problem was formulated as a constraint satisfaction problem involving constraints on the model and on the viewing parameters. A constraint manipulation system (CMS) evaluated the constraints symbolically at run-time using ribbon cues extracted from an image.

The principle behind the operation of the CMS is as follows. An inequality of the form $x \leqslant \langle \text{expr} \rangle$ means that $\langle \text{expr} \rangle$ is an upper bound for x, or equivalently, $\sup x = \sup \langle \text{expr} \rangle$. The SUP/INF method recursively applies sup and inf to $\langle \text{expr} \rangle$ until it consists of atoms and sups and infs of atoms; e.g.

$$\begin{aligned} & x \leqslant -y \\ \Rightarrow\ & \sup x = \sup -y \\ \Rightarrow\ & \sup x = -\inf y. \end{aligned}$$

An inconsistency is encountered if the system determines that $\sup x < \inf x$ for any x, in which case there is no solution to the problem. For sets of linear inequalities the algorithm produces both necessary and sufficient conditions for a solution, however the introduction of nonlinearity to the inequalities means that the result of the algorithm is no longer a sufficient (though still necessary) condition.

The nature of the measurable invariants in our domain makes the use of this algorithm attractive, since the knowledge of upper and lower bounds on variables is made explicit. We have developed a CMS based on Brooks' rule-base [6] with extensions for trigonometric functions, using the symbolic manipulation tool Mathematica [42]. The system takes a set of model parameters, and model and feature constraints and applies strategic substitutions to obtain simplifications where possible. The type of parameterizations laid out in Section 2.2 may be implemented using the operator set described in Table 1 (the symbol NaN denotes "undefined").

Unfortunately, as with most automatic symbolic computation, evaluation of the expressions is slow—this, indeed, was a major drawback of ACRONYM. Fisher and Orr [14] showed how to transfer the much of the cost to an off-line procedure; constraints are reduced off-line by a CMS as much as possible without knowing variable values and then compiled into a network whose topology derives from the SUP/INF constraint expressions. A SUP/INF network is built from nodes and directed arcs, where each node is one of:

- a constant;
- a variable, V, with associated interval $[\inf V, \sup V]$;
- an operator which performs a given operation on its input(s).

The allowable operator set in a network is as above, i.e.: $+$, $-$, $*$, reciprocal, signed reciprocal, min, max, sqrt, the three main trigonometric functions and their inverses, integer rounding functions (used to ensure that integer variables remain integral) and the constants $\pm\infty$ and NaN.

Table 1
Operator set

Operation	Domain	Sup/Inf
Minimum		$\sup \min\{x, y\} = \min\{\sup x, \sup y\}$ $\inf \min\{x, y\} = \min\{\inf x, \inf y\}$
Maximum		$\sup \max\{x, y\} = \max\{\sup x, \sup y\}$ $\inf \max\{x, y\} = \max\{\inf x, \inf y\}$
Unary minus		$\sup -x = -\inf x$ $\inf -x = -\sup x$
Addition		$\sup(x + y) = \sup x + \sup y$ $\inf(x + y) = \inf x + \inf y$
Subtraction		$\sup(x - y) = \sup x - \inf y$ $\inf(x - y) = \inf x - \sup y$
Multiplication		$\sup xy = \max\{\inf x \inf y, \inf x \sup y, \sup x \inf y, \sup x \sup y\}$ $\inf xy = \min\{\inf x \inf y, \inf x \sup y, \sup x \inf y, \sup x \sup y\}$
Square root		If $\inf x < 0$ Then NaN Else $\sup \sqrt{x} = \sqrt{\sup x}$ $\inf \sqrt{x} = \sqrt{\inf x}$
Reciprocal	$x \geqslant 0$	$\sup 1/x = 1/\inf x$ $\inf 1/x = 1/\sup x$
Signed reciprocal		If $0 \in [\inf x, \sup x]$ Then NaN Else $\inf 1/x = 1/\sup x$ $\sup 1/x = 1/\inf x$
Sine	$x \in [-\pi, \pi]$	If $\pi/2 \in [\inf x, \sup x]$ Then $\sup \sin x = 1$ Else $\sup \sin x = \max\{\sin \inf x, \sin \sup x\}$ If $-\pi/2 \in [\inf x, \sup x]$ Then $\inf \sin x = -1$ Else $\inf \sin x = \min\{\sin \inf x, \sin \sup x\}$
Cosine	$x \in [-\pi, \pi]$	If $0 \in [\inf x, \sup x]$ Then $\sup \cos x = 1$ Else $\sup \cos x = \max\{\cos \inf x, \cos \sup x\}$ $\inf \cos x = \min\{\cos \inf x, \cos \sup x\}$
Tangent	$x \in [-\pi, \pi]$	If $\pm\pi/2 \in [\inf x, \sup x]$ Then NaN Else $\sup \tan x = \tan \sup x$ $\inf \tan x = \tan \inf x$

If a variable's bounds change, the change is propagated via the network to other variables. Convergence requires that bounds on each variable $y = f(x)$ be evaluated as $\inf y = \max\{\inf y, \inf f(x)\}$ and $\sup y = \min\{\sup y, \sup f(x)\}$. This ensures that bounds only ever improve. A further requirement is that there be a small, but nonzero threshold, to interrupt asymptotic convergence.

Table 1
Continued

Operation	Domain	Sup/Inf
Arcsine	$[-\pi, \pi]$ (Range)	If $0 \in [\inf x, \sup x]$ Then $\sup \arcsin x = \pi - \arcsin \sup x$ $\inf \arcsin x = -\pi + \arcsin \inf x$ If $\inf x > 0$ Then $\sup \arcsin x = \pi - \arcsin \inf x$ $\inf \arcsin x = \arcsin \inf x$ If $\sup x < 0$ Then $\sup \arcsin x = \arcsin \sup x$ $\inf \arcsin x = -\pi + \arcsin \sup x$
Arccosine	$[-\pi, \pi]$ (Range)	$\sup \arccos x = \arccos \inf x$ $\inf \arccos x = -\arccos \inf x$
Arctangent	$[-\pi, \pi]$ (Range)	If $0 \in [\inf x, \sup x]$ Then $\sup \arctan x = \pi + \arctan \inf x$ $\inf \arctan x = \arctan \inf x$ If $\inf x > 0$ Then $\sup \arctan x = \arctan \sup x$ $\inf \arctan x = -\pi + \arctan \inf x$ If $\sup x < 0$ Then $\sup \arctan x = \pi + \arctan \sup x$ $\inf \arctan x = \arctan \inf x$
Integer		$\sup \text{int } x = \lfloor \sup x \rfloor$ $\inf \text{int } x = \lceil \inf x \rceil$

3. Coping with parameterizations

We now show how the SUP/INF algorithm and SUP/INF networks may be used to satisfy the five requirements laid out in Section 2.2.

3.1. Feature constraint tables

The generation of FCTs from geometrically fixed models is a simple matter. It is significantly more difficult for parameterized models because the feature constraints are functions of the parameters which are not necessarily quantifiable until run-time. Our feature constraint tables therefore consist of both numeric (where possible) and *symbolic* entries. For each feature constraint f_k and for each pair of features μ_i and μ_j we must compute the interval

$$Z_k(\mu_i, \mu_j).$$

For example, consider the feature constraint surf-dist-2 (see invariant f_3 in Appendix A) applied to a pair of surfaces μ_1 and μ_2 shown in Fig. 6. The inf and sup for this

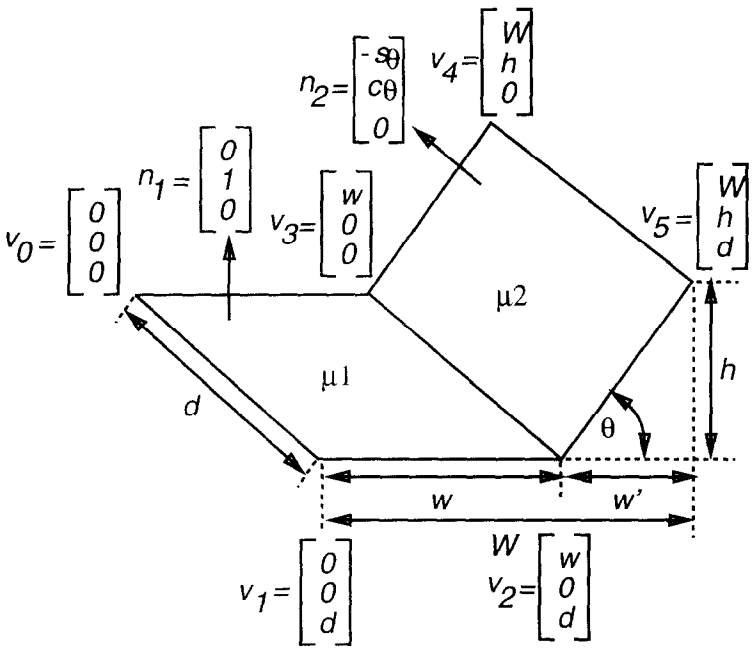


Fig. 6. Two parametric model surfaces.

measurement—i.e. the upper and lower bounds on the interval $Z_3(\mu_1, \mu_2)$—are given by

$$\begin{matrix} \min \\ \max \end{matrix} \{(\boldsymbol{v}_0 - \boldsymbol{v}_5) \cdot \boldsymbol{n}_2, (\boldsymbol{v}_1 - \boldsymbol{v}_5) \cdot \boldsymbol{n}_2, (\boldsymbol{v}_2 - \boldsymbol{v}_5) \cdot \boldsymbol{n}_2, (\boldsymbol{v}_3 - \boldsymbol{v}_5) \cdot \boldsymbol{n}_2\}.$$

Upon substitution for vertex coordinates and face equations, they are simplified symbolically (off-line) by the CMS using the constraints:

$$w, w', W \geqslant 0,$$
$$0 \leqslant \theta \leqslant \pi/2,$$
$$W = w + w',$$
$$h/w' = \tan\theta$$

to the expressions

$$0, \qquad w \sin\theta.$$

If an expression is not atomic (as is the case for $w\sin\theta$) then a new variable is created, and a new model constraint added to the model. For example, we would create a variable U and add the constraint $U = w\sin\theta$. The entries in the feature constraint table for surf-dist-2 are 0 and U, indicating that

$$\inf Z_3(\mu_1, \mu_2) = 0, \qquad \sup Z_3(\mu_1, \mu_2) = \sup U.$$

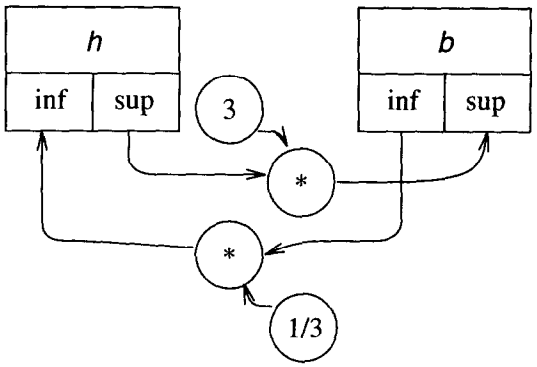


Fig. 7. A sample network, generated from the constraint $3h \geqslant b$.

3.2. Model parameters and constraints

Parameter values at run-time, and the constraints between them, are represented using a SUP/INF network. Model parameters are variables in the network, as are the variables created during FCT construction. Network paths represent:

- the functions that relate the FCT entries to the model parameters (see the example in the previous sub-section, $U = w \sin\theta$);
- implicit constraints between dependent model parameters (see the example in Section 2.2, $\tan\theta = 2h/b$);
- other model constraints for model restrictions (see the example in Section 2.2, $3h \geqslant b$).

This latter constraint would be parsed and simplified to the two equations

$$h \geqslant \tfrac{1}{3}b \Rightarrow \inf h = \tfrac{1}{3}\inf b,$$
$$b \leqslant 3h \Rightarrow \sup b = 3 \sup h.$$

and compiled to the network shown in Fig. 7.

3.3. Algorithm

The interpretation tree is a dynamic structure, growing as consistent assignments are added and shrinking if they are found to be inconsistent. The variables in the (static topology) network hold the current estimates of the parameter values and feature constraints for the current state of the interpretation tree. Fig. 8 shows a recursive version of the modified interpretation tree search algorithm, explained in further detail below.

Consistency of assignments $s_1 \rightarrow \mu_i$, $s_2 \rightarrow \mu_j$ is checked (by function `consistent`) by enforcing each of the feature constraints

$$F_k(s_1, s_2) \cap [\inf L, \sup U] \neq \emptyset,$$

where the L and U variables are the appropriate FCT entries, and therefore their sup and inf are readily available. Note that L and U need not refer to the same variable; a measurement may be bounded below by one parameter and above by another.

```
interpret(I, P, j)
[ if j = n
  [ verify(I, P);
  [ return;
  i ← 0;
  while i < m
  [ if consistent(I, P, {s_j, μ_i})
    [ P' ← update(I, {s_j, μ_i}, P)
    [ if P' ≠ ∅
      [ I ← I ∪ {s_j, μ_i}
      [ interpret(I, P', j + 1)
    [ i ← i + 1
  return;
```

Fig. 8. Pseudo-code for the modified interpretation tree search: the basic structure of the search is the same as in Fig. 4. The major difference is the addition of P, the current network state, which describes a multi-dimensional rectangular space. The function `consistent` returns true if each feature constraint satisfies the consistency test described in the text. The new function `update` applies the update rules given in the text to the current network state and then allows the network to converge. The condition $P' = \emptyset$ represents that the upper and lower bounds of a network variable crossed during network iteration.

If an assignment is consistent then the image measurements derived from it are used to update the network. The lower bound L must be less than the measured value, so a correct update for L is:

$$\sup L = \min\{\sup L, \sup F_k(s_1, s_2)\}$$

and the upper bound must be greater than the measured value, so a correct update for U is:

$$\inf U = \max\{\inf U, \inf F_k(s_1, s_2)\}.$$

Including the previous values in the update rule ensures that bounds can only ever remain stable or improve. Thus progressive refinements of the legal bounds of the image measurements are performed. The function `update` performs the updates above for each invariant measurement, and then propagates the effects to all other variables in the network.

If, at any stage, a variable's bounds cross, then an inconsistency has been found and the interpretation tree search backtracks. If not, once the network has converged, the search proceeds downwards with the new improved model parameter and feature constraint estimates.

Thus we have satisfied the requirements that we be able to determine the feature constraints at run-time, that we be able to determine when constraints have been violated, and that we provide a mechanism for updating parameter estimates based on the image measurements.

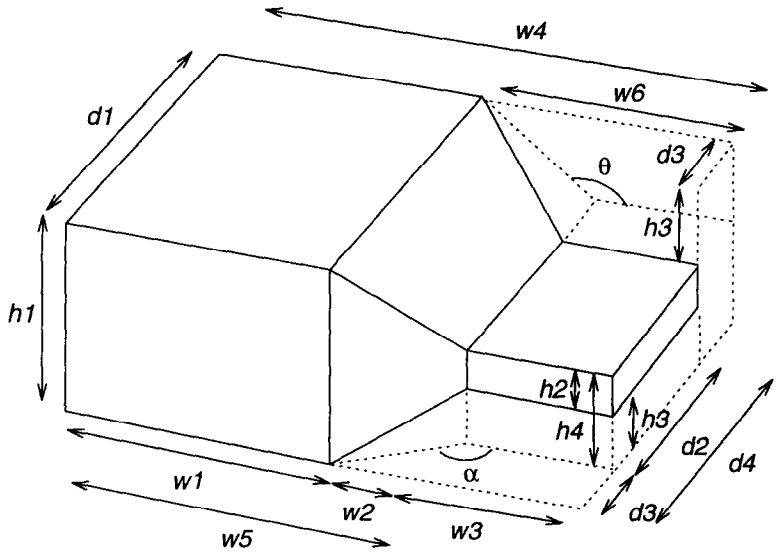


Fig. 9. A parameterization for a chimney object.

After the interpretation tree search, each complete branch of the tree constitutes a feasible hypothesis. The pose is then computed using a two-stage least-squares approach by first finding a best-fit rotation using the quaternion method of [11], followed by a best-fit translation [15,28]. Finding the pose via least-squares avoids the problems encountered by ACRONYM and by Fisher's system IMAGINE which both attempt to solve for pose using the SUP/INF method. The complex nonlinear constraints imposed by the pose computation often result in poor or even useless bounds being placed on the pose parameters. Our method provides a compromise in which the SUP/INF method recovers internal parameters, to which it is well suited, while the least-squares pose computation provides a stable, accurate pose estimate which is crucial for any subsequent tasks making use of the results of recognition. Further details, and descriptions of the hypothesis verification procedures used may be found in [34].

3.4. System operation

We now present an example of the system in operation. Fig. 9 depicts a class of chimney objects. The class is parameterized by the set:

$$\{\theta, \alpha, w_1, w_2, w_3, w_4, w_5, w_6, h_1, h_2, h_3, h_4, d_1, d_2, d_3, d_4\}$$

two angle and fourteen size parameters. A synthetic range image of an instance of the class is shown in Fig. 10(a), and six position/surface normal data estimated from the range data are indicated by small vectors and a point number, i, for observation $(\boldsymbol{p}_i, \boldsymbol{n}_i)$. Uncertainty in these points is due only to slight aliasing effects; thus uncertainty in the computed parameter ranges is attributable almost entirely to a lack of evidence rather than noise.

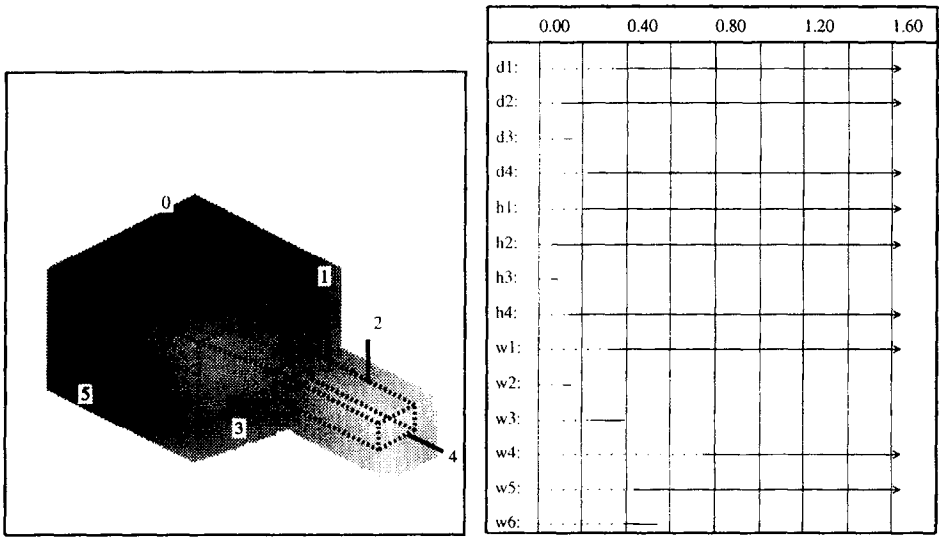


Fig. 10. Chimney recognition: (a) a range image with position/surface normal data extracted, and the computed pose/size superimposed; (b) the computed parameter ranges.

Legal interpretations for these data, and valid parameter ranges were derived using the constrained search and network propagation algorithm described in previous sections. Four legal interpretations were found corresponding to the symmetries around the principal axis, thus they are all equivalent (i.e. there is only one truly unique interpretation).

A wireframe for the class instance is shown superimposed on the range image in the computed pose (Fig. 10(a)). The parameter ranges for the distance parameters are shown graphically in Fig. 10(b). Intervals for the angle parameters, θ and α, were computed to be:

$$\theta \in [2.585, 2.650] \quad (\text{true value} = 5\pi/6 \approx 2.618),$$
$$\alpha \in [2.255, 2.461] \quad (\text{true value} = 3\pi/4 \approx 2.356).$$

In some cases very good bounds have been computed; e.g. d_3, h_3, w_2. We can see why this is so by considering, for example, the surf-dist-2 feature constraint (invariant f_3) applied to the matches $s_0 \to \mu_0, s_2 \to \mu_8$, which results in the constraints

$$[\inf h_3, \sup h_3] \cap F_3(s_0, s_2) \neq \emptyset,$$

where $F_3(s_0, s_2)$ is the error interval around the measurement $\boldsymbol{n}_2 \cdot (\boldsymbol{p}_0 - \boldsymbol{p}_2)$, and hence the updates

$$\inf h_3 = \max\{\inf h_3, \inf F_3(s_0, s_2)\}, \qquad \sup h_3 = \min\{\sup h_3, \sup F_3(s_0, s_2)\}.$$

Thus h_3 can be determined up to some small error. The final computed range was

$$h_3 \in [0.068, 0.088] \quad \text{(true value} = 0.075).$$

However for many other parameters the lower bound is a poor estimate, and no upper bound has been found. Consider the parameter d_1. Here, surf-dist-1 (invariant f_2) applied to the matches $s_0 \rightarrow \mu_0, s_5 \rightarrow \mu_1$ gives the constraint

$$[\inf Md_1, 0] \cap F_2(s_5, s_0) \neq \emptyset,$$

where Md_1 is a variable defined such that $Md_1 = -d_1$ and $F_2(s_5, s_0)$ is the error interval around the measurement $\boldsymbol{n}_5 \cdot (\boldsymbol{p}_0 - \boldsymbol{p}_5)$, hence the subsequent update is:

$$\sup Md_1 = \min\{\sup Md_1, supF_2(s_5, s_0)\}.$$

The final range for Md_1 was $[-\infty, -0.35]$ and hence for d_1:

$$d_1 \in [0.35, \infty] \quad \text{(true value} = 0.45).$$

3.5. *Improving parameter bounds*

Grimson and Lozano-Pérez [21] first advocated the use of sparse point-based features for recognition in order to avoid the problems caused by noise, occlusion and deficiencies in segmentation algorithms, leading to over-segmentation. Because it is a minimalist representation and because the IT search is data-driven (i.e. multiple observations may match to the same model surface) it copes with data fragmentation gracefully. A second major advantage of this data representation is that it leads to convenient, simple and elegant invariants and feature constraints. However a drawback of the representation is that it leads in many cases to the computation of overly conservative bounds. For example, the computed range for d_1 is $[0.35, \infty]$, while inspection of the range image in Fig. 10 clearly shows that much tighter bounds could potentially be computed. The reason for the conservative bounds stems from the fact that a position/surface normal is not a particularly good representation of a large surface. Fortunately, minor extensions to the search strategy can compensate for the representational deficiencies.

Firstly, for points which lie on the same observed surface patch, a connectivity constraint is enforced during the search, simply by requiring that they all match to the same model surface. Usually only a few key points delimit the patch entirely, and these will provide the best possible lower bound size estimates. As well as improving parameter estimates this has the obvious virtue of providing better constraint during the remainder of the search, leading to improved performance.

Secondly, if a point is known to lie on an occluding boundary (this can be determined easily from the range data since it corresponds to the point lying on the near side of a depth discontinuity), then the point must lie on *two* model surfaces, albeit one of them invisible. Having established a match for the point in the current interpretation a *secondary match* is sought from amongst the adjacent model surfaces. Because the surface is invisible, no surface normal is observed, meaning

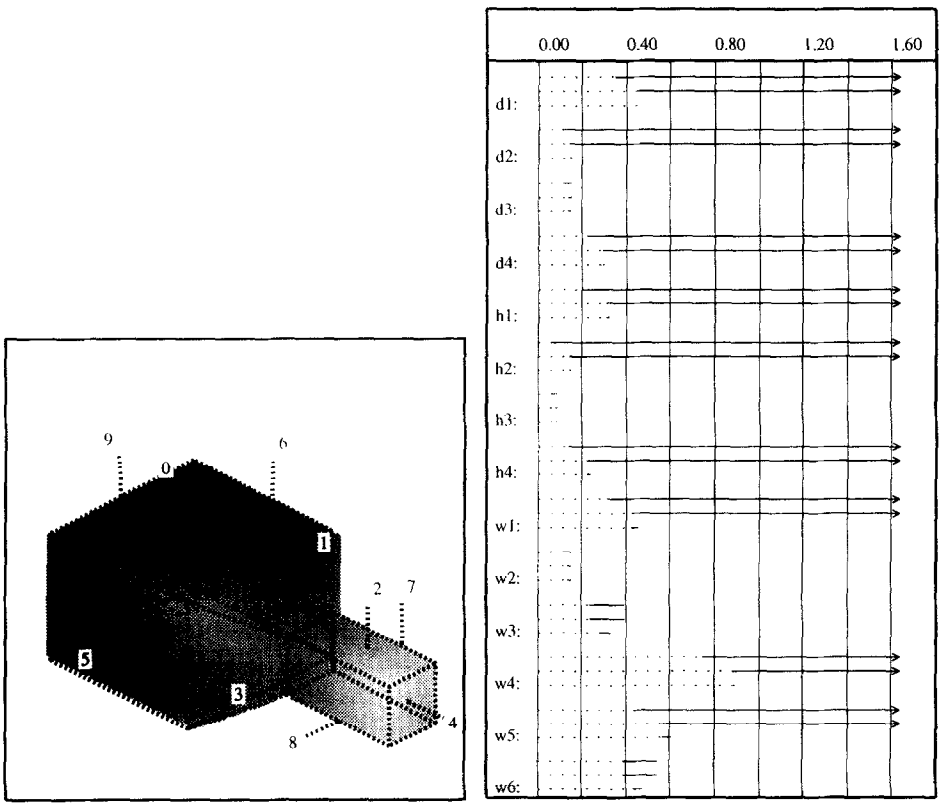


Fig. 11. Improved chimney recognition: (a) range image with position/surface normal data extracted, and the computed pose/size superimposed; (b) the computed parameter ranges (see text for details).

three of the four invariants cannot be computed. The fourth however (one of the surf-dist constraints) can still be computed, and it is this which provides an upper bound estimate. A further advantage of secondary matching is that a point matched on an occluding boundary constrains two of the three translational degrees of freedom of the object's pose (as opposed to only one for a point on the interior of a surface).

Fig. 11(a) shows the range image of Fig. 10 with a number of additional surface points extracted from the near side of steps in the image. The improved wireframe estimate is superimposed on the image in the computed pose. The real improvement is apparent from examination of the parameter ranges, shown in part (b) of the figure, which gives from top to bottom: the original ranges from Fig. 10(b); ranges from boundary observations—note the improvement in lower bounds; ranges from boundary observations with secondary matching—note the significant improvement in both upper and lower bounds.

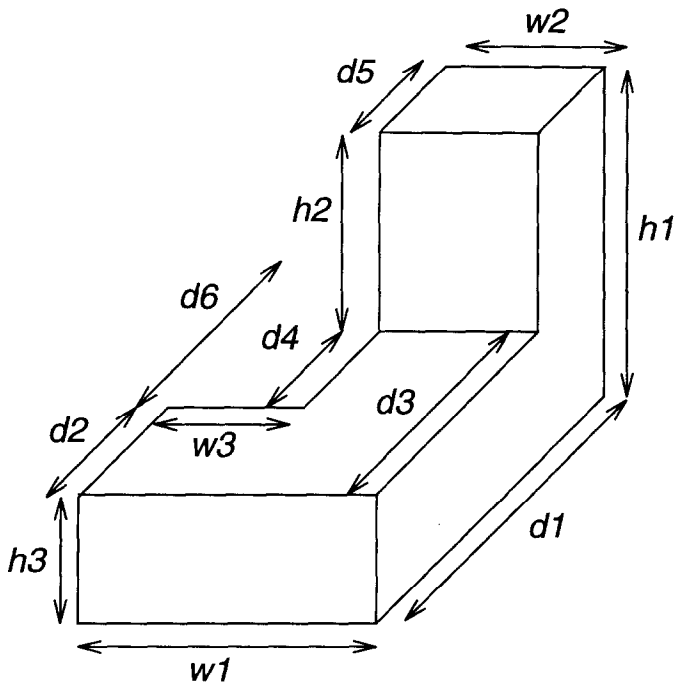


Fig. 12. A parameterization of a "widget" class.

4. Discrimination and identification

Part of the model definition given in Section 2.2 was a specialization hierarchy. Each sub-level of the hierarchy corresponds to a specialization of the parent through the addition of extra model constraints, active for that sub-class and all its children. [1]

Consider the class of "widgets" shown in Fig. 12 parameterized by the set:

$$\{w_1, w_2, w_3, h_1, h_2, h_3, d_1, d_2, d_3, d_4, d_5, d_6, s\}.$$

An exemplary specialization hierarchy is given in Fig. 13. At each level of the tree extra model constraints specialize the previous level, until, at the leaf nodes, three different widgets, A, B and C are completely geometrically determined.

Having established a set of legal interpretations using a high (general) level of the hierarchy, it is then a simple matter to check lower levels for sub-class membership. This can be performed as a straightforward depth-first traversal of the hierarchy. However a more informative search tests each child of a parent node before descent to consistent children; a useful feature of this method is that we obtain directly the most specific level of the tree with which the observations are consistent. Note that neither of these methods is the same as finding an interpretation at each level of the model hierarchy; checking

[1] Brooks [6] discussed a similar representation with respect to ACRONYM, although this was never adequately demonstrated experimentally.

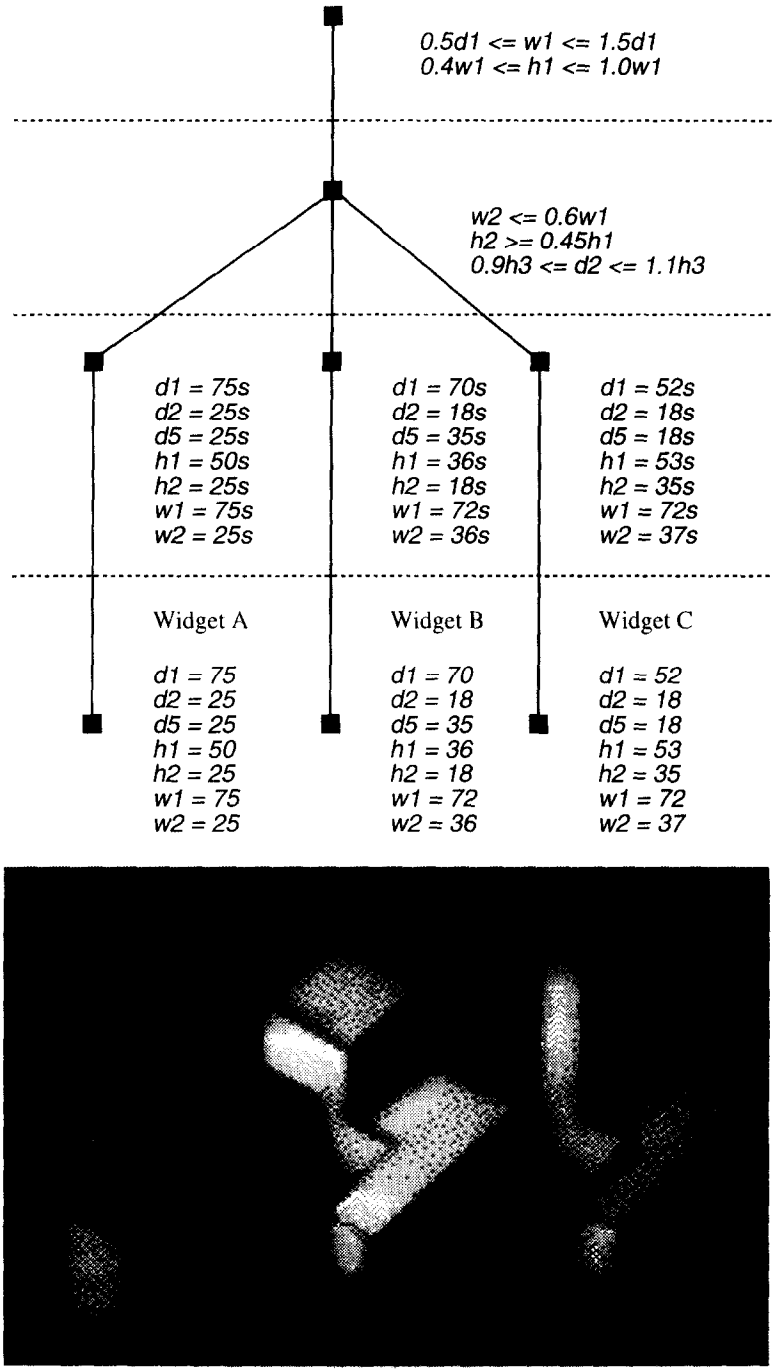


Fig. 13. A model hierarchy for the widget class, and three instances of the class, consistent with the leaf nodes of the tree.

a specialization is a cheap operation. The specialized constraints are precompiled along with the most general ones, but only invoked during the specialization process. Only a handful of network iterations are then required to find a subset of the rectangular parameter space, indicating either the new parameter ranges, or if the space is empty, that no interpretation of the observations exists for the current node in the model hierarchy. Even if an exact identification or sub-class membership cannot be established, the parameter ranges can be used to distinguish between instances simply by testing for non-intersecting ranges of the same parameter in the different instances.

Fig. 14(a) shows the left camera view from a stereo pair of the three widgets jumbled together, and the 3D line segments found by the TINA stereo vision system [31]. Fig. 14(b) shows the computed pose and size of the three widgets superimposed on the left-hand view (alternative views of each are shown below). Each has been correctly identified using the parameter ranges shown in Fig. 14(d).

5. Objects with multiple and repeated sub-parts

Sub-parts are defined in a model relative to a coordinate frame local to the sub-part, and by a REV-graph, a set of sub-part model parameters, and a set of model constraints. In addition, each sub-part has an associated transformation, possibly parameterized, which gives the location of the sub-part relative to the global model frame. The transformation is given by terms (either numbers or parameters) specifying:

- an axis of rotation;
- an angle of rotation about the axis;
- a translation.

In contrast to other recognition systems which search separately for different sub-parts, then test the consistency of the poses of the sub-parts with an overall model [10, 14, 19], our system attempts to match over the whole model immediately. This has the advantage that in instances where there is insufficient evidence to hypothesize the presence of individual sub-parts, it may still be possible to hypothesize the presence of the model as a whole.

Each sub-part also has an associated number of repetitions. Usually this will be a constant (frequently equal to one), but on occasions it is useful to define an object in terms of a parameterized number of repetitions; the pallet, which we use as an exemplar here, is one such object. This is achieved by defining the number of repetitions to be a parameter constrained to take on integer values, and defining the transformation between local and global coordinate systems to be a function of the sub-part number.

Although the model is defined in terms of sub-parts, the search algorithm operates on a "flattened" version of the model. This is created when the system first reads in the model. For each part encountered in the model definition, r copies of the part are created, where r is the maximum number of repetitions of the sub-part allowed (this is specified in the model). The copies are identical except for a *sub-part number* (from 1 to r) and the local to global transformation for each (which is a function of the sub-part number). In addition r copies of each feature in the part are created, identical except that each maintains a record of which sub-part it belongs to.

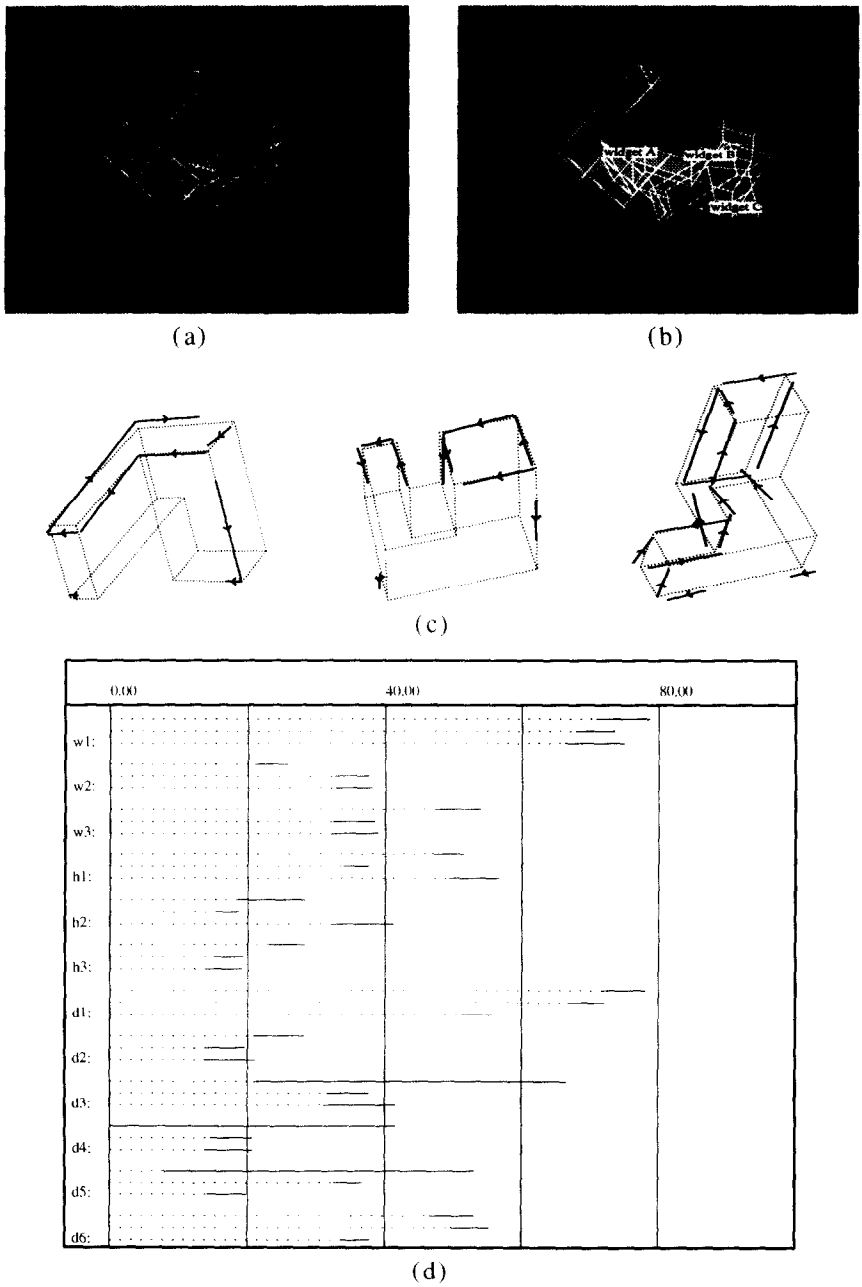


Fig. 14. Pose and identification from stereo edge fragments: (a) the left view from a stereo pair of images of widgets A, B and C with stereo edge fragments superimposed; (b) the computed pose, size and identity of each widget superimposed; (c) alternative views of each widget and the edge fragments used; (d) the computed parameter ranges in millimetres—the solid bars indicate the ranges for each parameter for the three objects (from top to bottom: widget A, B and C).

```
interpret(I, P, j)
[ if j = n
  [ verify(I, P);
  [ return;
  i ← 0;
  while i < m
  [ if pnum(μ_i) < sup r(μ_i)
    [ t ← inf r(μ_i)
    | inf r(μ_i) ← max(t, pnum(μ_i))
    | if consistent(I, P, {s_j, μ_i})
    | [ P' ← update(I, {s_j, μ_i}, P)
    | | if P' ≠ ∅
    | | [ I ← I ∪ {s_j, μ_i}
    | | [ interpret(I, P', j + 1)
    | [ inf r(μ_i) ← t
    [ i ← i + 1
  [ return;
```

Fig. 15. Pseudo-code for the extended search: the basic structure of the search is the same as those in Figs. 4 and 8. The new condition involves a function pnum(μ) which returns the sub-part number of the part to which feature μ belongs, and r(μ) which returns the current upper and lower bounds on the number of repetitions of the sub-part to which feature μ belongs. The variable t is temporary storage for the value of inf r(μ_i).

By insisting that the maximum number of repetitions is known in advance, we can create a list of all features which are in the *maximal set* of model features and guarantee that an instance of the model is a *subset* of the maximal set. Feature constraint tables are computed and stored for the maximal model.

An extended version of the search is given in Fig. 15. The additions in this algorithm correspond to the following tests and associated actions:

- If the current sub-part number of the sub-part to which the feature belongs is greater than the maximum allowed (which may be dependent on the current interpretation) then the match is invalid: i.e. if it has already been established through various constraints that the model is a *proper* subset of the maximal model, then there is no need to test matches to features not in the subset.
- If the current sub-part number of the sub-part to which the feature belongs is greater than the minimum allowed (in the current interpretation) then update the current minimum: i.e. if we match to a feature from a sub-part not in the current minimal set, but in the current maximal set, then henceforth in this interpretation the minimal set must be expanded to include the extra sub-part.

In order to illustrate the use of these extensions, we now return to the specific example used to motivate our work, that of recognizing industrial pallets in NEL range data (for other examples, including recognition of articulated objects, refer to [34]). A

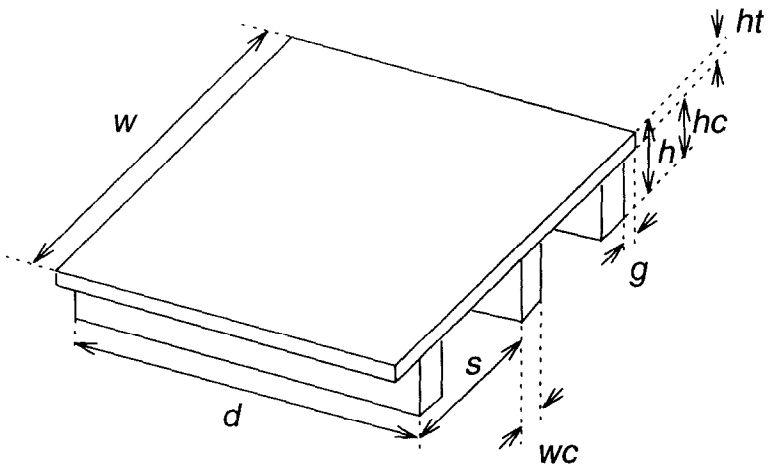


Fig. 16. The simplified pallet model: the model contains a top surface, parameterized by $\{w, h_t, d\}$, and a set of struts parameterized by $\{w_c, h_c, d\}$. The strut positions relative to the top surface are governed by the overlap at each end, g, and the strut separation, s. The figure shows only three struts, however in the model a fourth is possible; the number of struts, n, being an integer parameter of the model.

pallet consists of several parallel, equal-sized top and bottom slats, which are long, flat rectangular prisms, and several equal-sized struts of similar dimension to the slats but running perpendicular to the slats. Typical examples were shown in the introduction in Fig. 1.

In the recognition examples in this section we consider a class of pallets which have an arbitrary number of slats (all slats are modelled as a single solid board instead of individual slats), and which have either three or four struts (a parameterized quantity). The bottom slats are omitted from the model. A further simplification comes from the observation that in most poses of the pallet many of the surfaces defined by the REV-graph cannot be reliably detected in an NEL range image. These surfaces, which include the ends and sides of the slats and the strut ends are marked in the model as *secondary model features*, meaning they are not considered during primary matching, but are eligible as secondary matches.

Two sub-parts are defined in the model: a top, parameterized by $\{w, h_t, d\}$, and a strut parameterized by $\{w_c, h_c, d\}$. The number of struts is parameterized by an integer parameter, $n \in [3,4]$, and the transformation of each strut is the translation $[l_i, 0, 0]^T$, where l_i is defined by the constraints:

$$l_i = g + is, \quad \forall i = 0, \ldots, n-1$$

(g is the overhang of a slat at the edge of the pallet and s is the strut separation).

The simplified model, parameterized by the set

$$\{n, w, h, d, s, w_c, h_c, h_t, g\}$$

is shown in Fig. 16 (an annotated version of the model definition appears as an appendix in [34]).

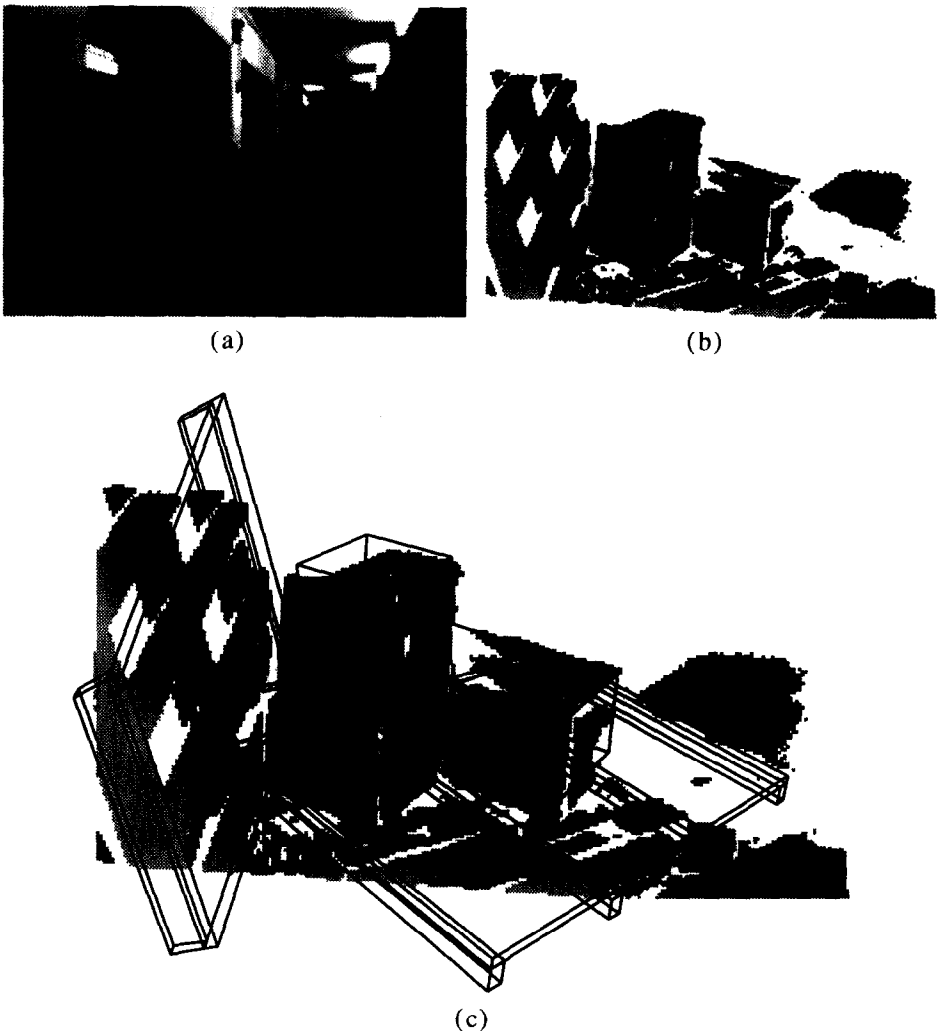


Fig. 17. A scene containing two pallets of different sizes and a cardboard widget: (a) an intensity image of the scene with a small pallet on the left and a large pallet on the ground; (b) an NEL range image of a similar scene in which lighter represents closer, and white means no valid range data;(c) pose and size results for both pallets and the widget.

A complex scene involving two pallets, one substantially occluded, and a large widget, is shown in Fig. 17. The pallet on the left has three struts and the one on the right has four, but the furthest strut has not been detected by the range sensor.

The recognition algorithm was run for each of three sets of surfaces, corresponding to the two pallets and the widget. These surfaces were extracted automatically from the range image but the selection of which surfaces to use was made manually. The searches for the two pallets each generated four legal hypotheses; two symmetric interpretations

with three struts, and two symmetric interpretations with four struts. The widget generated two legal interpretations corresponding to the symmetric interpretation of its width and depth. The correct interpretations are shown superimposed on the range image in Fig. 17(c).

In the case of the smaller (upright) pallet, the three strut interpretations are the correct ones. However the system makes no use of negative evidence which might rule out the four strut interpretation. Likewise, one of the two widget interpretations could be ruled out by considering global evidence about the scene since in this interpretation the size computed requires the base of the widget to extend below the floor. The correct interpretations for the larger pallet are the four strut ones, but the lack of evidence (no fourth strut was detected by the sensor) makes it impossible for the system to rule out the three strut interpretation. These results present a strong case for both the use of negative and global evidence, and intelligent sensing strategies to follow up initial guesses in order to resolve ambiguities, which will form the basis for future research.

6. Performance

For the case of parameterized models, the ability of geometric constraints to prune the search space of the interpretation tree is hampered by the lack of knowledge of parameter values, particularly early in the search. It is therefore crucial in assessing the practicality of the algorithm we have presented to study the effectiveness of the constraints and how the search performance is affected.

Our experiments have confirmed the intuition that a data driven search such as the one we use, is better suited to surface data since the breadth of the tree is determined by the number of model features, which is considerably smaller for surfaces than edges. Furthermore, the edge invariants (see Appendix A) tend to result in more complex networks. Therefore as a performance gauge, we consider the experiment shown in Section 4 of widget recognition from stereo data which represents close to a "worst-case" scenario.

In order to provide a meaningful set of comparisons between the various levels of model specialization (the limit being geometrically fixed models) we invoked the search for each of the levels separately. Fig. 18 shows three graphs, one for each of the interpretations of the segmented data, plotting the number of consistent assignments against the level of the interpretation tree. Each graph contains four plots, each one corresponding to a complete search using one level from the model hierarchy. Although the performance of the parameterized models is considerably worse than for fixed models early in the search, the most striking feature of the graphs is that they show that the search has been brought under control (i.e. the number of tests is no longer increasing) after 2 or 3 matches, and has comparable performance to the fixed models after 4 or 5 matches. Also note that despite the combinatorial increase in the search over the first few levels, the effort is still much less than the worst case for this model of $O(24^{n+1})$ (the model has 24 edges and n is the level in the tree search).

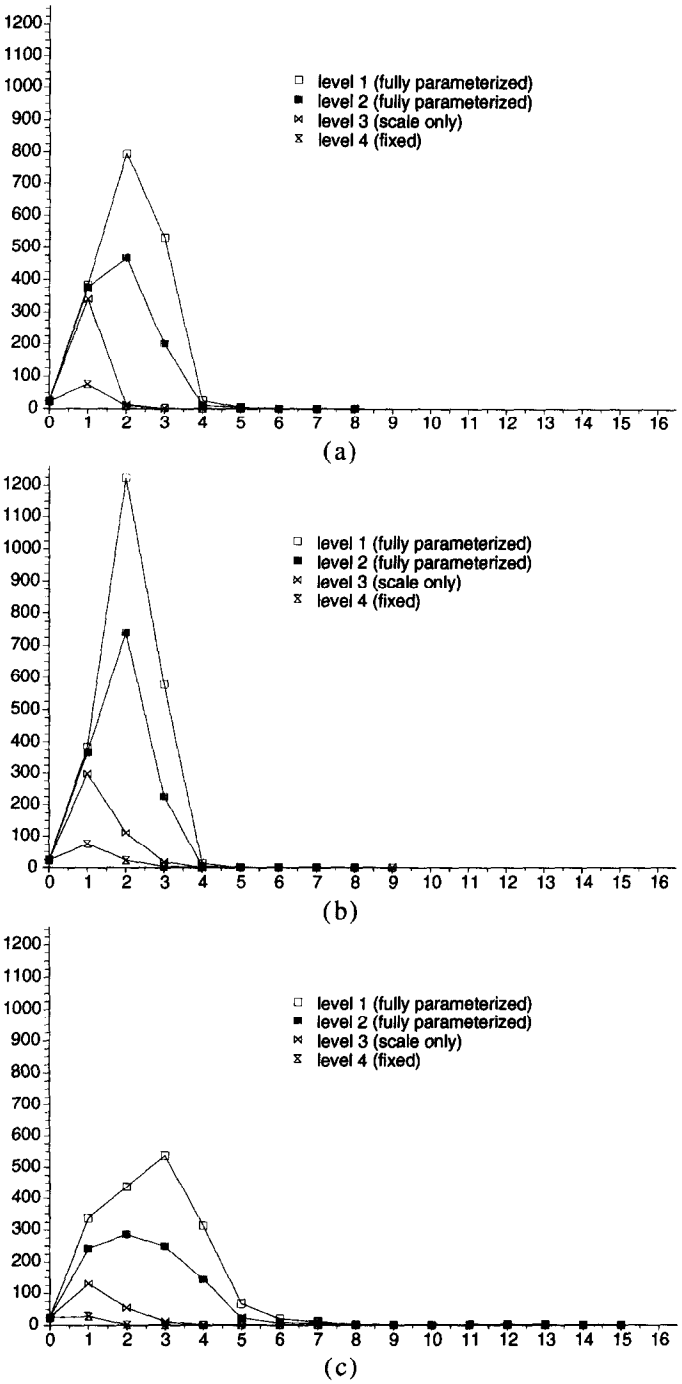


Fig. 18. Number of assignments made during the tree search for each widget in the scene. The horizontal axis corresponds to tree level, and the vertical axis to the number of consistent assignments made: (a) widget A; (b) widget B; (c) widget C (see text for details).

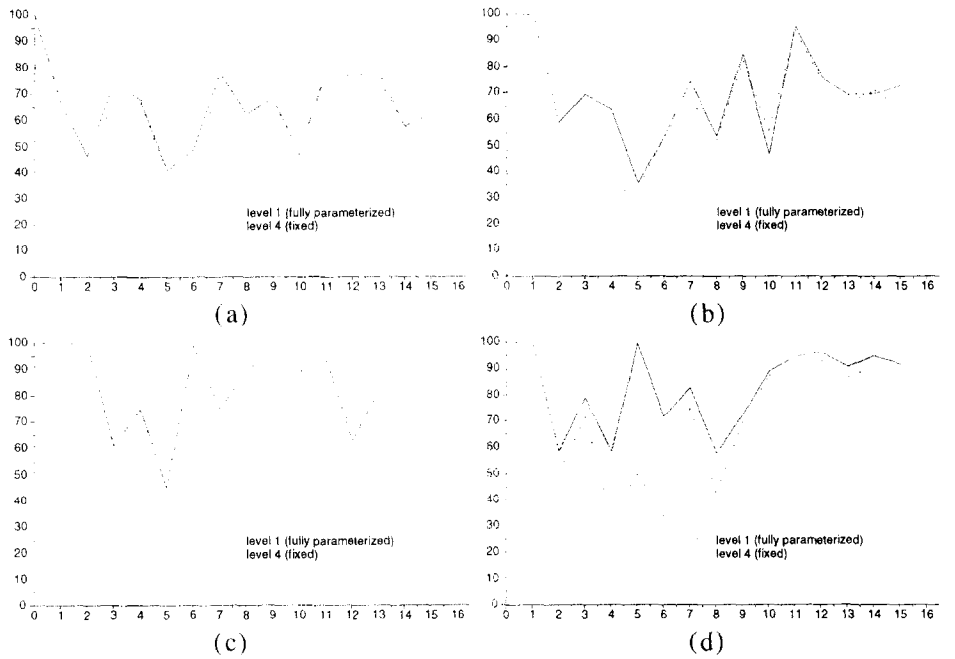


Fig. 19. Percentage of false positives allowed by each feature constraint plotted against at each level of the tree search for each of the four edge constraints: (a) edge-angle, (b) edge-dist, (c) edge-proj-1, (d) edge-proj-2 (see text for details).

We have measured the pruning power of each of the feature constraints by considering the percentage of false positives accepted by each feature constraint at a given tree level, calculated as

$$100\frac{c-t}{c}$$

(where, for a given level of the tree, c is the number of times the constraint was tested and t number of correct matches). This percentage has been plotted against the level of the tree in Fig. 19 (for the widget C example). These graphs show the expected effect: for the edge–angle constraint there is little difference between the fixed and the parameterized cases (since in this experiment the model angles are fixed even in the parameterized model), and the performance of the distance constraints, edge-dist, edge-proj-1 and edge-proj-2, is poorer for parameterized models over the first few tree levels, but improves to give approximately the same performance as for fixed models thereafter.

Performance statistics for all the results given in this paper are summarized in Table 2, giving the network size, size of the tree search required, number of interpretations, network statistics (number of iterations and total network flops), and time for recognition. These latter times reflect the total time for recognition for the algorithm running on a SunSparc-2 workstation including all algorithm steps except network compilation (which, in any case, is negligible).

Table 2
System performance summary

Experiment	Network size (nodes)	Search size (IT nodes)	Interpret-ations	Total network iterations	Total network operations	Time (CPU secs)
widget A (Fig. 14)	699	693	1	1984	357516	8.38
widget B (Fig. 14)	699	1443	1	3783	616380	14.10
widget C (Fig. 14)	699	2553	1	7202	1177030	28.37
chimney (Fig. 10)	4094	67	4	237	350378	2.86
chimney (Fig. 11)	4094	99	4	326	383412	3.43
pallet3 (Fig. 17)	2328	115	4	277	333425	3.49
pallet4 (Fig. 17)	2328	145	4	529	709792	4.69
widget (Fig. 17)	699	980	2	2573	196978	4.03

7. Relationship to previous work

The extensive literature on three-dimensional object recognition has concentrated, for the most part, on geometrically fixed objects, since it was realised that such objects led to powerful constraints which could be exploited in numerous ways to achieve recognition and localization. We begin this section with a brief tour of the seminal work in the area of fixed 3D object recognition, particularly from 3D data, paying particular attention to work which has influenced our own. A full survey of these techniques is beyond the scope of the paper (an excellent, though dated, survey can be found in [3]), and we therefore concentrate on the considerably less common systems for parametric object recognition.

7.1. Fixed 3D object recognition

A popular approach, and one adopted in our work described in the previous sections is that of hypothesis generation, through a search of the object-model feature space seeking appropriate matches followed by a global verification stage.

Grimson and Lozano-Pérez developed the RAF system [20,21] based on a data-driven search of an interpretation tree which we have already discussed in detail in earlier sections. Developments from their work, using the same principles of geometric constraints and interpretation tree search were made by Murray and Cook [28] using scale invariant constraints on 3D edge fragments and by Flynn and Jain in the Bonsai system [15] using unary and binary constraints on planar and quadric surfaces from range data.

A useful refinement of the IT is based around the concept of *alignment*. In this approach an object pose (i.e. a transformation which maps a model into the sensor coordinate frame) is computed as soon as possible, and then a search initiated to provide support for the hypothesized pose. Faugeras and Hebert developed such an approach using model-driven rather than data-driven tree search [11]. A major disadvantage of the model driven approach is that the nature of the search restricts each model feature

to match to at most one data feature. In practice, occlusions and poor segmentation often lead to fragmented data features which ideally all match to one model feature. The work of Bolles and Horaud, who developed the 3DPO system [5] replaced the first few levels of the interpretation tree search with so-called *feature foci*—features which when matched would determine, or greatly restrict the pose of the object. The IT search was replaced by an equivalent algorithm, that of finding maximal cliques in a graph, for obtaining hypothesis support subsequent to pose estimation. Generalizations of the feature focus idea have appeared in 3D recognition systems developed by Chen and Kak [8] using range data, and Pollard et al. [32] who combined ideas from 3DPO and RAF systems in which feature foci were groups of 3D edge fragments such as three concurrent 3D line segments. Lowe's SCERPO system [24] was based on similar principles but used 2D intensity images. Matching was constrained by the *viewpoint consistency constraint*, and proceeded once an initial pose had been estimated from a *perceptual group* (another term for feature focus) such as a parallelogram formed by edge segments in the scene. Other notable work on recognition of 3D objects from 2D images using object–feature match search techniques subsequent to alignment includes that by Goad [18] and Huttenlocher and Ullman [23].

An alternative to search through feature-match space was proposed by Thompson and Mundy [38] who used vertex–edge pairs (two edges with a coincident vertex and a further vertex) to estimate a transformation from model to scene under affine projection. Each vertex–edge pair casts a vote in transformation space for a particular transformation with the true one(s) estimated as peaks in this space.

Finally, we briefly mention the relatively recent developments of recognition through the use of projective invariants, pioneered by Weiss [41], and best illustrated in the system developed by Rothwell et al. [16,37]. To date most of this work has concentrated on planar objects. Only recently [36] have extensions to three-dimensional objects been achieved (for objects which can be "caged" by polyhedral point sets, and objects with bilateral symmetries), and it is too early to tell how robust and amenable these methods will prove to full automation.

7.2. *Parametric object recognition*

The systems above were all designed to operate with fixed, rather than parametric objects. Some of these have been extended more recently to cope with internal degrees of freedom. We discuss these, and other systems designed for parametric objects below.

The best known of all parameterized object recognition schemes is ACRONYM [6], designed to find instances of 3D models in single 2D views (i.e. intensity images). ACRONYM consisted of three distinct parts massaged together:

- a powerful representational scheme which used generalized cones (volumetric primitives each defined by an axis, a cross-section and a sweeping rule which determines how the cross-section changes as it sweeps along the axis) as the basic building blocks and a frame-based, semantic network [27] of objects, parts and sub-parts and a hierarchy of models and sub-models;
- a symbolic inference engine, CMS, which was based on the SUP/INF algorithm of Bledsoe [4], for the propagation of symbolic and numeric constraints;

- a prediction mechanism which used *ribbons* (the two-dimensional equivalent of generalized cones) observed in an image to hypothesize the presence of generalized cones and set up constraints appropriate to the observed data.

Although there is a multitude of good ideas within ACRONYM, it was slow (owing to the nature of symbolic computation). More seriously, it was only ever demonstrated on a handful of images meaning that many of its potential uses were never tested (including objects with repeated sub-parts, as demonstrated in this paper). This was a result of the difficulty in extracting data from intensity images useful for constraining the size and positions of the generalized cones representation.

Grimson [19,20] described how to extend the RAF system for simple parameterizations of 2D objects, dealing with the cases of uniform scale, uniform stretching along one axis, and rotation about a common axis. The former two were incorporated by treating each as a special case hard-coded into the search algorithm itself, and the latter by recognizing sub-parts and then finding parts consistent with a rotation about a common axis. The system thus lacked generality, a limitation overcome in the work we have described in this paper.

Ettinger [10] generalized the latter part of Grimson's strategy, building a system based on a bottom-up hierarchical search, in which primitive features (in this case components of the *curvature primal sketch* [1] located in 2D images of 2D models such as road-signs) are built into sub-parts which ultimately are linked to recognize objects. A limited amount of parameterization was allowed.

Fisher's IMAGINE system [12,14] also allows the definition of objects in terms of sub-parts, and parameterizations are introduced in the form of rigid motions between sub-parts—although sub-parts may not have internal degrees of freedom such as scale or stretch. Like Ettinger's system, IMAGINE recognition is based on a bottom up approach using feature clusters, similar to the feature foci mentioned above, formed using topological constraints and matched to sub-parts. A SUP/INF network is used to solve for the transformations between sub-parts and the overall model to sensor transformation.

Vayda and Kak [40] have demonstrated a system for recognition of generic postal objects from range data. Individual items are recognized in scenes of piles of postal objects. The system's geometric reasoning algorithms consider not only information about single objects but about the entire scene (for example, intersection tests are performed between hypothesized objects in the scene). However the success of the algorithms used depends largely on the limited scope of the representation; objects are modelled as either a box with three scaling degrees of freedom or cylinders with two scaling degrees of freedom.

The major research on 3D parametric object recognition from 2D data has been conducted by Lowe [25] (an extension of the SCERPO system) and Nguyen et al. [29]. Both methods use gradient descent from an *a priori* estimate of parameter values (including pose parameters) to solve simultaneously for pose and parameters. Such systems are extremely useful for tracking articulated objects from a dense sequence of frames where the previous frame gives a good guess for the current frame, as demonstrated in [26], or in human–computer interfaces where a good initial guess can be provided by a user, as in [29]. However the problems with gradient descent in non-

convex spaces is well documented, and the major problem with these methods is their reliance on prior estimates combined with inability to self-bootstrap unknown parameter values.

Recent work by Umeyama [39] bears closest resemblance to the work we have presented. In common with our work he represents parameters by intervals, and bases recognition around depth-first search of an interpretation tree with simultaneous solution for parameter values. General parameterizations such as combinations of articulations and arbitrary scaling and stretching are allowed, however a major limiting factor is that it operates only with 2D models from points found on silhouettes (formed under parallel projection).

8. Discussion

As we have mentioned earlier, most previous recognition systems have been designed to work with fixed objects. This is a clear limitation in many environments and to address this deficiency we have developed techniques for recognizing instances of 3D object classes (which may consist of multiple and/or repeated sub-parts with internal degrees of freedom, linked by parameterized transformations), from sets of 3D feature observations. Recognition of a class instance is structured as a search of an interpretation tree in which geometric constraints on pairs of sensed features not only prune the tree, but are used to determine upper and lower bounds on the model parameter values of the instance. A real-valued constraint propagation network unifies the representations of the model parameters, model constraints and feature constraints, and provides a simple and effective mechanism for accessing and updating parameter values.

We have examined a number of different cases using real 3D data of two different types—3D edge fragments from a stereo vision system, and position/surface normal data derived from planar patches extracted from a range image—and shown that the system is effective in determining object pose and parameters. Although the cost associated with recognition of parametric objects is greater than for fixed objects, our experiments demonstrate that in many cases this is a price worth the flexibility gained.

As it stands, the system is based around a very general interpretation tree search, as used in, for example, [21,28]. However currently the system is limited to determining the pose and parameters of an object in a non-cluttered scene. One way around this problem is to introduce the notion of a "null feature" which always successfully matches, as suggested by Grimson and Lozano-Pérez in [22]. The price paid when using this technique, however, is unacceptable, as it results in combinatorial explosion in the number of interpretations [20]. We noted in Section 7 that a great increase in efficiency can be gained through a more strategic search, seeded by a focus feature or perceptual group [5,8,32]. The effect of this is to direct the search to specific areas of interest, avoiding many problems associated with excessive scene clutter, currently a limitation of our system. A major target of our future research will therefore be the incorporation of these tools into the parametric framework we have described. One point of particular interest raised by such an extension is that the search could be controlled *strategically* to minimize uncertainty about parameter values.

A second obvious way of improving the performance of the system is to exploit the potentially parallel nature of many aspects of the method. To this end we have explored distributing both the tree search and the network evaluation throughout a MIMD processor network [9].

Acknowledgements

We are indebted to David Murray for many useful discussions. We are also grateful to Steve Pollard of Sheffield University for the use of the TINA stereo vision system, and to an anonymous referee for some insightful comments. Ian Reid was supported by a Rhodes scholarship.

Appendix A. Binary geometry constraints

This appendix specifies the feature constraints (i.e. the f_k) used in our system. Two types of data may be input: either surface data from our in-house (Oxford/NEL) laser range-finding system [33,35], or three-dimensional edge fragments from a stereo system [31]. Below we give sets of feature constraints for each of these data types. Demonstrations of the system working with both types of data are given in Sections 4 to 5.

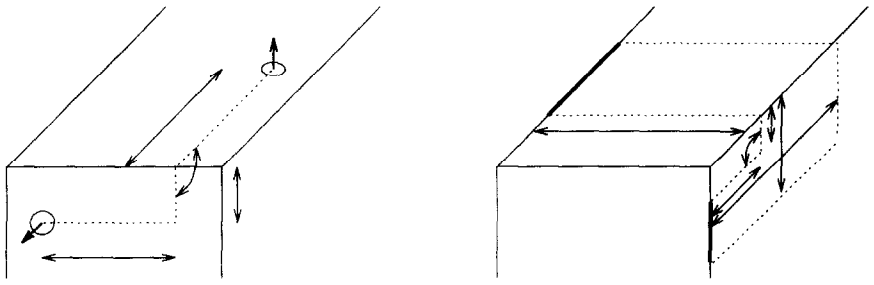
Surface data are represented by a 3D position and a unit surface normal, $s = (\boldsymbol{p}, \boldsymbol{n})$. This representation is rather conservative, since dense range maps supplied by the Oxford/NEL range finder give information about surface extent and connectivity. However, range-image segmentation remains a difficult problem, and a conservative data representation obviates the problems of *over-segmentation* where noise (or other factors) cause a single surface to be broken into multiple facets, and those problems created by occlusions. Furthermore, it is a simple matter to transfer topological constraints such as connectivity (when available) to the interpretation to the tree search itself, by, for example, enforcing the constraint that data from the same scene surface must match to the same model surface, or that data from adjacent surfaces in the scene must match to adjacent model surfaces.

We use the same set of surface constraints as in the original 3D version of RAF. This set of image measurements (and feature constraints) is given by:

(1) surf-angle: $\boldsymbol{n}_1 \cdot \boldsymbol{n}_2$.
(2) surf-dist-1: $\boldsymbol{n}_1 \cdot (\boldsymbol{p}_1 - \boldsymbol{p}_2)$.
(3) surf-dist-2: $\boldsymbol{n}_2 \cdot (\boldsymbol{p}_2 - \boldsymbol{p}_1)$.
(4) surf-cross-dist:

$$\frac{\boldsymbol{n}_1 \wedge \boldsymbol{n}_2}{|\boldsymbol{n}_1 \wedge \boldsymbol{n}_2|} \cdot (\boldsymbol{p}_1 - \boldsymbol{p}_2).$$

These measurements embody all the possible constraint on position and orientation available from two position/surface normal data points. They are depicted in Fig. A.1(a). If, instead of $\boldsymbol{n}_1 \cdot \boldsymbol{n}_2$, we use $\arccos \boldsymbol{n}_1 \cdot \boldsymbol{n}_2$, with the sign adjusted so that surfaces which



a b

Fig. A.1. Feature constraints used by our system: (a) for planar surface patches; (b) for straight edge fragments.

face each other have positive angles and surfaces which point away from each other have negative angles (and the same convention is adopted in the model), then sign ambiguity is avoided. Henceforth surf-angle refers to this revised measurement.

Edge data are represented by a set of edge fragments, $s = (\boldsymbol{e}, \boldsymbol{m}, l)$; $\boldsymbol{e}$ is an unsigned unit vector in the direction of the fragment, $\boldsymbol{m}$ is the midpoint of the fragment and l is the fragment's length. From these it is straightforward to define the endpoints of the fragment, $\boldsymbol{p}_1 = \boldsymbol{m} + \frac{1}{2}l\boldsymbol{e}$ and $\boldsymbol{p}_2 = \boldsymbol{m} - \frac{1}{2}l\boldsymbol{e}$. Murray and Cook [28] proposed a set of feature constraints based on edge directions. The recognition problem they considered—edge fragments sensed from moving images—required scale independent constraints because of the depth/speed scaling ambiguity inherent in visual motion processing under perspective projection. For the case where scale independence is not an issue their constraints can be extended to include distances as well. Instead of these, we propose a new, but equivalent, set of feature constraints, listed below:[2]

(1) edge-angle: $\arccos \boldsymbol{e}_i \cdot \boldsymbol{e}_j$.

(2) edge-dist:

$$\text{if } (s_i \text{ and } s_j \text{ parallel}) \ \operatorname{sgn}(\boldsymbol{e}_i \cdot \boldsymbol{e}_j)(\boldsymbol{v}\cdot\boldsymbol{v} - (\boldsymbol{v}\cdot\boldsymbol{e}_j)^2)^{1/2} \text{ else } \boldsymbol{v} \cdot \frac{\boldsymbol{e}_i \wedge \boldsymbol{e}_j}{|\boldsymbol{e}_i \wedge \boldsymbol{e}_j|},$$

where $\boldsymbol{v} = \boldsymbol{m}_i - \boldsymbol{m}_j$ is a vector between the edges.

(3) edge-proj-1:

$$(\boldsymbol{p}_{i_1} - \boldsymbol{m}_j) \cdot \left[\boldsymbol{e}_j \wedge \frac{\boldsymbol{e}_i \wedge \boldsymbol{e}_j}{|\boldsymbol{e}_i \wedge \boldsymbol{e}_j|}\right] \quad \text{and} \quad (\boldsymbol{p}_{i_2} - \boldsymbol{m}_j) \cdot \left[\boldsymbol{e}_j \wedge \frac{\boldsymbol{e}_i \wedge \boldsymbol{e}_j}{|\boldsymbol{e}_i \wedge \boldsymbol{e}_j|}\right].$$

(4) edge-proj-2:

$$(\boldsymbol{p}_{j_1} - \boldsymbol{m}_i) \cdot \left[\boldsymbol{e}_i \wedge \frac{\boldsymbol{e}_j \wedge \boldsymbol{e}_i}{|\boldsymbol{e}_j \wedge \boldsymbol{e}_i|}\right] \quad \text{and} \quad (\boldsymbol{p}_{j_2} - \boldsymbol{m}_i) \cdot \left[\boldsymbol{e}_i \wedge \frac{\boldsymbol{e}_j \wedge \boldsymbol{e}_i}{|\boldsymbol{e}_j \wedge \boldsymbol{e}_i|}\right].$$

Image measurement edge-dist is the perpendicular distance between the lines on which the fragments lie and edge-proj gives upper and lower bounds on the distance of edge

[2] This constraint set more closely resembles those used by Pollard [32].

s_i to the plane containing s_j with surface normal $e_j \wedge e_i \wedge e_j$. These measurements are depicted in Fig. A.1(b).

References

[1] H. Asada and J.M. Brady, The curvature primal sketch, *IEEE Trans. Pattern Anal. Mach. Intell.* **8** (1) (1986) 2–14.

[2] B.G. Baumgart, Winged-edge polyhedron representation, Technical Report AIM-179, Stanford University (1972).

[3] P.J. Besl and R.C. Jain, Three-dimensional object recognition, *Comput. Surv.* **17** (1) (1985) 75–145.

[4] W.W. Bledsoe, The sup-inf method in presburger arithmetic, Technical Report Memo ATP 18, Department of Mathematics and Computer Science, University of Texas (1974).

[5] R.C. Bolles and P. Horaud, 3DPO: a three-dimensional part orientation system, *Int. J. Robotics Research* **5** (3) (1986) 3–26.

[6] R.A. Brooks, Symbolic reasoning among 3D models and 2D images, *Artif. Intell.* **17** (1981) 285–348.

[7] R.A. Brooks, Symbolic error analysis and robot planning, *Int. J. Robotics Research* **1** (4) (1982) 29–68.

[8] C.H. Chen and A.C. Kak, A robot vision system for recognizing 3D objects in low-order polynomial time, *IEEE Trans. Syst. Man Cybern.* **19** (6) (1989) 1535–1563.

[9] F. Chenavier, I.D. Reid and J.M. Brady, Recognition of parameterized objects in range images: a parallel implementation, *Image Vision Comput.* **12** (9) (1994) 573–582.

[10] G.J. Ettinger, Large hierarchical object recognition using libraries of parameterized model sub-parts, in: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition* (1988) 32–41.

[11] O.D. Faugeras and M. Hebert, The representation, recognition and locating of 3D objects, *Int. J. Robotics Research* **5** (3) (1986) 27–52.

[12] R.B. Fisher, *From Surfaces to Objects: Computer Vision and Three Dimensional Scene Analysis* (Wiley, New York, 1989).

[13] R.B. Fisher and M.J.L. Orr, Solving geometric constraints in a parallel network, in: *Proceedings 3rd Alvey Vision Conference* (1987) 87–95.

[14] R.B. Fisher and M.J.L. Orr, Geometric reasoning in a parallel network, *Int. J. Robotics Research* **10** (2) (1991) 103–122.

[15] P.J. Flynn and A.K. Jain, BONSAI: 3D object recognition using constrained search, in: *Proceedings 3rd International Conference on Computer Vision* (1990) 263–267.

[16] D.A. Forsyth, J.L. Mundy, A.P. Zisserman, C. Coelho, A. Heller and C.A. Rothwell, Invariant descriptors for 3D object recognition and pose, *IEEE Trans. Pattern Anal. Mach. Intell.* **13** (10) (1991) 971–991.

[17] P.C. Gaston and T. Lozano-Pérez, Tactile recognition and localization using object models: the case of polyhedra on a plane, *IEEE Trans. Pattern Anal. Mach. Intell.* **6** (3) (1984) 257–265.

[18] C. Goad, Special purpose automatic programming for 3d model-based vision, in: *Proceedings Image Understanding Workshop* (1983) 371–381.

[19] W.E.L. Grimson, Recognition of object families using parametrized models, in: *Proceedings 1st International Conference on Computer Vision*, London (1987) 93–101.

[20] W.E.L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints* (MIT Press, Cambridge, MA, 1991).

[21] W.E.L. Grimson and T. Lozano-Pérez, Model-based recognition and localization from sparse range or tactile data, *Int. J. Robotics Research* **3** (3) (1984) 3–35.

[22] W.E.L. Grimson and T. Lozano-Pérez, Localizing overlapping parts by searching the interpretation tree, *IEEE Trans. Pattern Anal. Mach. Intell.* **9** (4) (1987) 469–482.

[23] D.P. Huttenlocher and S. Ullman, Recognizing solid objects by alignment, *Int. J. Comput. Vision* **5** (2) (1990) 255–274.

[24] D.G. Lowe, The viewpoint consistency constraint, *Int. J. Comput. Vision* **1** (1) (1987) 57–72.

[25] D.G. Lowe, Fitting parameterized 3D models to images, Technical Report TR 89-26, Computer Science Department, University of British Columbia, Vancouver, BC (1989).

[26] D.G. Lowe, Robust model-based motion tracking through the integration of search and estimation, *Int. J. Comput. Vision* **8** (2) (1992) 113–122.

[27] M. Minsky, A framework for representing knowledge, in: P.H. Winston, ed., *The Psychology of Computer Vision* (McGraw-Hill, New York, 1975).

[28] D.W. Murray and D.B. Cook, Using the orientation of fragmentary 3D edge segments for polyhedral object recognition, *Int. J. Comput. Vision* **1** (2) (1988) 153–169.

[29] V.D. Nguyen, J.L. Mundy and D. Kapur, Modelling generic polyhedral objects with constraints, in: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition* (1991) 479–485.

[30] M.J.L. Orr, R.B. Fisher and J. Hallam, Computing with uncertainty: intervals versus probability, in: P. Mowforth, ed., *Proceedings of the British Machine Vision Conference* (Springer-Verlag, Berlin, 1991) 351–354.

[31] S.B. Pollard, J.E.W. Mayhew and J.P. Frisby, PMF: a stereo correspondence algorithm using a disparity gradient limit, *Perception* **14** (1985) 449–470.

[32] S.B. Pollard, J. Porrill, J.E.W. Mayhew and J.P. Frisby, Matching geometrical descriptions in three-space, *Image and Vision Computing* **5** (2) (1987) 73–78.

[33] G.T. Reid, S.J. Marshall, R.C. Rixon and H. Stewart, A laser scanning camera for range data acquisition, *J. Phys. D Appl. Phys.* **21** (1988) 1–3.

[34] I.D. Reid, Recognizing parameterized objects from range data, Ph.D. Thesis, OUEL TR 1918/92, University of Oxford (1991).

[35] I.D. Reid and J.M. Brady, Model-based recognition and range imaging for a guided vehicle, *Image Vision Comput.* **10** (3) (1992); Special Issue on Range Image Understanding.

[36] C.A. Rothwell, Extracting projective information from single views of 3d point sets, in: *Proceedings 4th International Conference on Computer Vision* (1993) 573–582.

[37] C.A. Rothwell, Recognition using projective invariance, Ph.D. Thesis, University of Oxford (1993).

[38] D.W. Thompson and J.L. Mundy, Three-dimensional model matching from an unconstrained viewpoint, in: *Proceedings IEEE Conference on Robotics and Automation*, Raleigh, NC (1987).

[39] S. Umeyama, Parameterized point pattern matching and its application to recognition of object families, *IEEE Trans. Pattern Anal. Mach. Intell.* **15** (2) (1993) 136–144.

[40] A.J. Vayda and A.C. Kak, A robot vision system for recognizing generic shaped objects, *Comput. Vision Graph. Image Process Image Understanding* **54** (1) (1991) 1–46.

[41] I. Weiss, Geometric invariants and object recognition, *Int. J. Comput. Vision* **10** (3) (1993) 207–232.

[42] S. Wolfram, *Mathematica: A System for Doing Mathematics by Computer* (Addison-Wesley, Reading, MA, 1988).