Structural Indexing: Efficient 3-D Object Recognition

Fridtjof Stein, *Student Member, IEEE*, and Gérard Medioni, *Member, IEEE*

Abstract— **We present an approach for the recognition of multiple 3-D object models from three 3-D scene data. We work on dense data, but neither the models nor the scene data have to be complete. We are addressing the problem in a realistic environment; the viewpoint is arbitrary, the objects vary widely in complexity, and we make no assumptions about the structure of the surface. Our approach is novel in that it uses two different types of primitives for matching: small surface patches, where differential properties can be reliably computed, and lines corresponding to depth or orientation discontinuities. These are represented by *splashes* and 3-D curves, respectively. We show how both of these primitives can be encoded by a set of super segments, consisting of connected linear segments. These super segments are entered into a table and provide the essential mechanism for fast retrieval and matching. We address in detail the issues of robustness and stability of our features. The acquisition of the 3-D models is performed automatically by computing splashes in highly structured areas of the objects and by using boundary and surface edges for the generation of 3-D curves. For every model, all features are recorded in a database. The scene is screened for highly structured areas, and splashes are computed in these areas and encoded. 3-D curves, corresponding to depth or orientation discontinuities, are also encoded. These features are used to retrieve hypotheses from the database. Clusters of mutually consistent hypotheses represent instances of models. The precise pose of a model instance in the scene is found by applying a least squares match on all corresponding features. We present results with our current system (three dimensional object recognition based on super segments (TOSS)) and discuss further extensions.**

Index Terms— **Feature detection, hashing, indexing, model-based recognition, pose esimation, range-image understanding, 3-D object recognition.**

I. Introduction

WE PRESENT an object recognition system that is able to match general 3-D objects from partial 3-D data in an efficient way by using a method called *structural indexing*. By saying 3-D, we talk about models and scenes having a 3-D representation. By talking about "general objects," we make very few restrictive assumptions about their shape; therefore, we only exclude statistically defined shapes (e.g., foams) and crumpled objects (e.g., fractals). Matching and recognizing in an "efficient way" is based on a fast indexing and retrieval system that has a complexity that grows as $O(kN)$ when N is the number of models, and $k < 1$.

Manuscript received October 15, 1990; revised May 6, 1991. This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract F49620-90-C-0078. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

F. Stein and G. Medioni are with the Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, CA 90089-0273.

IEEE Log Number 9102647.

Representing a 3-D solid object is either possible by using a surface or a volumetric description. Volumetric descriptions from a single view require a difficult inference step to compensate for the unseen part; therefore, we will use descriptions based on visible surface instead. The task of object recognition involves identifying a correspondence between a part of one range image and a part of another range image with a particular view of a known object. This requires the ability to match one feature of one range image against a feature of another range image. A feature can be either a surface patch or a general 3-D curve. The question is "How can we represent such features so that they can be matched in an efficient way?"

Reviewing the existing systems, none thus far (known to the authors) is able to represent, match, and recognize *general* 3-D objects. An excellent overview regarding the problem of matching free-formed surfaces can be found in [1]. Most object recognition systems to date either rely on exact, CAD-like models or make restrictive assumptions on the possible shape of the surface patches. The following is a summary of related work:

- Grimson and Lozano Pérez [11], [12] describe a system that is able to recognize objects from sparse scene data. If there are m known objects with n_j segments each and s scene segments, there are $\sum_{j=1}^{m}(n_j)^s$ combinations of pairings between scene and model segments. The system tests these combinations using a tree search. Not all these combinations need to be tested because distance and angle constraints are used to prune almost all the combinations. Still, the number of combinations that need to be tested grows rapidly with object complexity. For those combinations not ruled out, the system tries to find an object position and orientation consistent with the segment assignments. If a consistent transformation is found, the object is recognized. In [10], Grimson shows that under some simple assumptions, the expected complexity of recognizing isolated objects is quadratic in the number of model and scene fragments but that the expected complexity of recognizing objects in cluttered environments is exponential in the size of the correct interpretation.
- Bhanu [2] presents a 3-D scene analysis system for the shape matching of real-world 3-D objects. Object models are constructed using multiple-view range images. At first, range images of each model object from different views are acquired. The relative orientation between each pair of views is assumed to be known. Then, the surface points of the object in each view are isolated from the background by removing pixels with large depth values (far away from viewer). Second, a large number of object

surface points are obtained by transforming points from each individual views into a common object-centered coordinate system. Finally, the object is represented as a set of planar faces approximated by polygons. This is accomplished by a two-step algorithm. In the first step, a three-point seed algorithm is used to group surface points into face regions, and in the second step, the face regions are approximated by 3-D planar convex polygons. Shape matching is performed by matching the face description of an unknown view with the stored model using a relaxation-based scheme called *stochastic face labeling*. The face features obtained above are used to compute the initial face-labeling probabilities for possible matching between each model face m_i and each face in the unknown view s_i. Then, the labeling probabilities are updated by checking the compatibilities in the geometric transform for each pair $<m_i, s_i>$. The compatibility between m_i and s_i is obtained by finding the transformation T_i between them and applying T_i to other pairs $<m_j, s_j>$, computing the error in feature values after the transformation. The match is less compatible if the error is large.

- Horaud and Bolles [18] and Bolles *et al.* [3] present the 3DPO system for recognizing and locating 3-D parts in range data. The model consists of two parts: an augmented CAD model and a feature classification network. The CAD model describes edges, surfaces, vertices, and their relations. The feature classification network classifies observable features by type and size. The model objects are represented by a tree-like network such that each feature contains a pointer to each instance in the CAD models. In the range images, discontinuities are detected and classified into cylindrical and linear curves. A local-feature-focus method is used for the matching process. At first, the system searches for features that match a feature for some model, for example, a cylindrical curve with a given radius. Then, objects are hypothesized by determining whether a pair of observed segments are consistent with a given model feature. The system shows good results on bin-picking tasks of industrial parts.
- Faugeras and Hebert [8] developed a system to recognize and locate rigid objects in 3-D space. Model objects are represented in terms of linear features such as points, lines, and planes. Range images are used as input. The same features such as significant points, lines, and planes are used to describe scene objects. Edges such as surface discontinuities are extracted using nonmaxima suppression on maxima of surface normals. Planar surfaces are then extracted by using a region-growing method. The system uses rigidity constraints to guide the matching process. At first, possible pairings between model and scene features are established, and the transformation is estimated using quaternions. Then, further matches are predicted and verified by the rigidity constraints.
- Ikeuchi [13] developed a method for object recognition in bin-picking tasks. Object models are generated under various viewer directions, and apparent shapes are then classified into groups. The models consist of surface inertia, surface relationship, surface shape, edge relationship, extended Gaussian image, and surface characteristic distribution. Since this system is mainly designed for the task of bin picking, only *one* type of object, which is the same one as in the model, appears in the scene. The same surface features used in models are extracted and classified by the help of the model. An interpretation is generated according to various model views. The orientation and location of the scene object are then decided by comparing their surface features and classified by the interpretation tree.
- Fan *et al.* [6], [7] use elementary surface patches as the primitives of description and recognition. The method consists of two stages. In the first stage, surfaces of 3-D objects are segmented at discontinuities. Then, these detected discontinuities are used to segment a complex surface into simpler meaningful components called surface patches. These patches can then be approximated by simple surface models. Finally, these surface patches are grouped to refer to meaningful 3-D objects, and attributed graphs are generated to describe these objects. In the second stage, the descriptions of objects are used to build multiview models. These views are arranged such that most of the significant features of the model object are contained in at least one of these views. Finally, partially occluded objects in a given scene are described using the same descriptions and identified as one of the model views. The recognition process contains three modules: the screener, the graph matcher, and the analyzer. In the first module, to avoid a search for every possible match between model views and scene objects, for every scene object, all the model views are ordered according to their similarities to the scene object. Then, in the second module, a graph matching procedure is used, starting from the first ordered model view, to find the best correspondences between this view and the scene object. Finally, in the third module, the chosen matches are refined according to their geometric similarities. They present results on a variety of complex objects in scenes with multiple objects occluding each other.
- The work conceptually closest to our approach was done by Radack and Badler [20]. Their representation of a surface patch is based on a radial decomposition. They introduce a new surface representation called *distance profile*. These profiles are used for the matching process. This method reduces the matching of 3-D surfaces to the matching of 2-D curves. They use points with high curvature to position the centers of the distance profiles. In their paper, they present results with artificial data.
- With 3-D POLY, Chen and Kak [5] developed a system in which they present a novel approach of organizing the feature data for 3-D objects. They present a data structure that they call the *feature sphere*. The matching and verification step is based on comparing spatial relationships of special feature sets. They show very fast recognition results for cluttered scenes with several industrial objects.

This brief survey is summarized in Table I. Many systems, based on different assumptions about shape, such as polygonal

TABLE I
3-D FROM 2.5-D, 3-D

Year	System	Primitives	Scene Input	CDAM	Distinct Features
1984	Grimson	face segments	sparse 3D position and orientation measurements	$O(2^n)$	
1984	Bhanu	polygonal face segments	range data	$O(2^n)$	model is multiview
1986	3DPO	significant features	range data	$O(2^n)$	in general fast (because of use of rich features)
1986	Faugeras and Hebert	points, lines, faces	range data	$O(2^n)$	
1987	Ikeuchi	faces organized in aspect graph	3D data	$O(2^n)$	CAD model
1988	Fan	faces organized in adjacency graph	range data	$O(2^n)$	model is multiview
1989	Radack	radial profile decompositions	range data	n/a	artificial data only
1989	Chen and Kak	significant feature sets organized in a feature sphere	range data	$O(n^2)$	fast

CDAM = Cost of Detection of Absence of Model
This may be considered a somewhat subjective performance measure, since many of these systems have been designed to perform in scenarios where such a situation is extremely unlikely to occur.

n = number of scene primitives

shapes, solids of revolution, or generalized cylinders, were developed. In contrast, we believe that our proposed system (three dimensional object recognition based on super segments (TOSS)) is able to recognize rigid objects whose shapes are not constrained by any simplifying assumptions. Our algorithm uses a combined representation that captures information about both smooth patches and discontinuity lines.

This paper is organized as follows: In Section II, we briefly outline our previous work on recognition of flat objects from an arbitrary viewpoint and show some results. Section III presents our approach to recognize objects in range imagery. We focus on the choice of our two features:

The 3-D Curve: For some objects, such as polyhedra, it is natural to use a representation based on edges. For that reason, we introduce in Section III-A our basic feature for the representation of surface and depth discontinuities: the *3-D curve*. We achieve a robust and stable representation by using multiple line fitting tolerances to obtain a set of polygonal approximations. The polygonal approximations are grouped in sets of connected segments. These super segments are encoded based on the angles between consecutive segments, providing invariance with respect to rotation, translation, and scale (even though scale is not needed in 3-D object recognition).

The Splash: For some objects, however, such as objects bounded by free-form surfaces, it is difficult to use edges for the representation. Therefore, in Section III-B, we present a new representation (the *splash*) based on small surface patches, where we can compute differential properties in a reliable way. A splash consists of a radial decomposition of surface normals. It is a local Gaussian map describing the distribution of surface orientation along a geodesic circle. A splash can be represented by two 2-D periodic functions, which can also be combined into one compact, 3-D curve. This allows us to use a unified representation scheme for the splash *and* the above-mentioned 3-D curve.

In Section III-C, we address in detail the issues of robustness and stability of these two features. In Section III-D, we describe how we use a table to store our features and retrieve them efficiently for hypotheses generation. We provide a brief complexity analysis with respect to the number of models in the database and with respect to the scene complexity. We present two experiments that we performed to analyze the behavior of the TOSS system for large databases (100 models).

Finally, in Section IV, we present some results on real data and provide some concluding remarks.

For clarity of presentation, we give, in the Appendices, the least squares method with which we compute transformations of matched data and the details of the geometrical constraints used for verification.

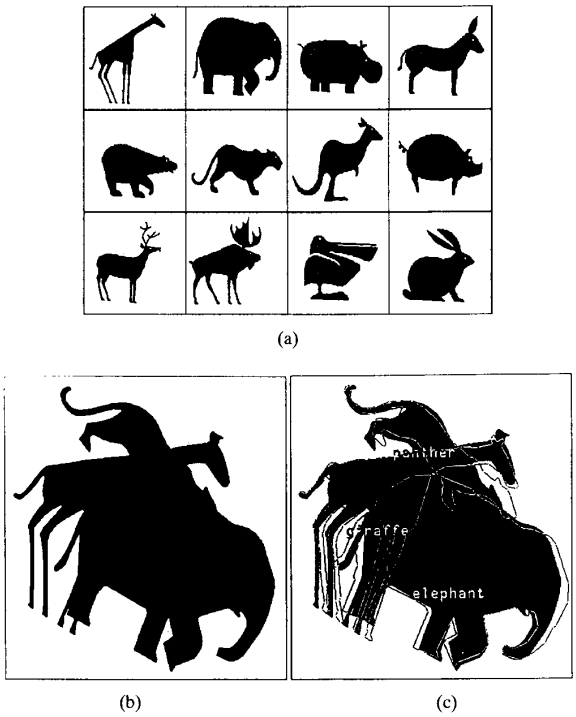
II. Our Previous Work: Structural Indexing in 2-D

Our approach is an extension of our early work [23], which addressed the problem of recognition of multiple flat objects in a cluttered environment from an arbitrary viewpoint (weak perspective). The models are acquired automatically and initially approximated by polygons with multiple line tolerances for robustness. Individual points or line segments are too local to be useful as matching primitives. Grouping a fixed number of adjacent segments provides us with our basic features: the super segments. To encode the super segments, we quantize the angles between the segments and the eccentricity of the vertices of the super segment. These quantized values serve as a key for a table, where we respectively record the super segments as entries. This provides the essential mechanism for indexing and fast retrieval. Once the database of all models is built, the recognition proceeds by segmenting the scene into a polygonal approximation; the code for each super segment retrieves model hypotheses from the table. Hypotheses are clustered if they are mutually consistent and represent the instance of a model. Finally, the estimate of the transformation is refined. This methodology allows us to recognize models in the presence of noise, occlusion, scale, rotation, and translation. We can also handle weak perspective in a certain range, which depends on the quantization intervals of the encoding mechanism. Unlike most of the other existing systems, the complexity of our algorithm grows as $O(kN)$ when N is the number of models, and $k \ll 1$.

We have tested the performance of the system with several examples. We show one of them here. We obtained the animal shapes from coarsely digitized binary images (see Fig. 1(a)). The animal scene was taken by printing the three animals, enlarging them, and cutting them out. Finally, we put the three silhouettes on a light table and took a picture with a video camera. The camera made an angle of about 20° with the normal to the light table. This procedure guaranteed scaling, occlusion, rotation, translation, weak perspective, and noise (thanks to our skills in cutting, especially around the ears of the giraffe). The recognition time (not including representation generation time) for the scene (see Fig. 1(b) and (c)) is 4.1 s on a Symbolics 3675 Lisp machine.

III. Structural Indexing in 3-D

In recent years, object recognition in 3-D has been either



(a) (b) (c)

Fig. 1. Animal shapes: (a) Database; (b) scene; (c) detected animals.

performed based on boundary and edge information (see [1], [15]) or by using surface descriptions (see, for example, [1], [6], [9], [19]). This creates problems when edges are not well defined or when the objects cannot be segmented into stable elementary patches.

We present a system that combines both approaches with the following strategy: "Use for the recognition task whatever information is available." We extract edges corresponding to depth and orientation discontinuities and use them as primitives. These are not sufficient, however, to represent smooth free-form surfaces. Therefore, we also compute differential properties called *splashes* in smooth areas. We describe both of theses primitives in the following subsections.

A. The 3-D Curve

For some objects, such as polyhedra, it is natural to use a representation based on edges. For this reason, we extract 3-D curves that are likely to correspond to depth and orientation discontinuities. Edge data consist, in general, of noisy, missing, or broken edges (see, e.g., the surface discontinuities between the fuselage and the wing in Fig. 3). We take that into account. Our effort is not to develop a system that can only deal with *perfect* edges. We want to use whatever data current state-of-the-art edge detectors can generate. When we get noninvariant edges (such as limbs, which are viewer dependent), we treat them just like all other edges. When they are matched against scene edges, they might generate wrong hypotheses, which are then discarded in the verification step. The most stable edges leading to the best matches are the edges that correspond to

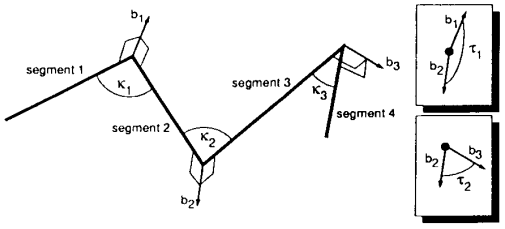


Fig. 2. Example of 3-D super segment of cardinality 4.

discontinuities of depth and surface (such as the boundary of the wings in Fig. 3).

Our representation of a general 3-D curve is based on a polygonal approximation. We do not rely on any specific feature detection algorithm, and we do not explicitly handle distinguished points such as corners or inflection points. Curvature and torsion are the most important features of a general 3-D curve. They are invariant with respect to rotation and translation. By using a polygonal approximation, we lose most of the curvature and torsion information, but we approximate it by computing the "curvature" and "torsion" angles between consecutive line segments (see Fig. 2).

Obviously, there is no unique polygonal approximation for a curve. Therefore, for the purpose of robustness, we use *several* polygonal approximations with different line fitting tolerances. Since we want to handle occlusion, we do not expect to obtain complete curves in our scenes but only portions of them. On the other hand, individual segments are too local to be useful as matching primitives. Grouping a fixed number of adjacent segments provides us with our first basic features: the 3-D super segments. The 3-D super segment is an extension of the 2-D super segment we used for recognition of flat objects in [23]. In accordance with Fig. 2, 3-D super segments are characterized by their cardinality (number of segments), curvature angles (between consecutive segments), and torsion angles (between consecutive binormals).

As mentioned before, we are mainly interested in the curvature and torsion information implicitly captured by the 3-D super segment curvature and torsion angles. This is the reason we use them to encode a 3-D super segment. The curvature (κ_i) and torsion (τ_i) angles are defined in the following way:

$$\kappa_i = \cos^{-1} \frac{s_{i+1} \cdot s_i}{|s_{i+1}||s_i|}$$

$$\tau_i = \cos^{-1}(b_i \cdot b_{i+1})$$

with the binormals

$$b_i = \frac{s_{i+1} \times s_i}{|s_{i+1}||s_i|}$$

and s_i is the ith segment of the 3-D super segment. To encode a 3-D super segment ss with cardinality n, we use a simple encoding scheme. The list of the quantized curvature and torsion angles values is the code of the 3-D super segment ss:

$$\texttt{Code}(ss) = (\texttt{Quant}(\kappa_1), \texttt{Quant}(\kappa_2), \ldots \texttt{Quant}(\kappa_{n-1}), \texttt{Quant}(\tau_1), \texttt{Quant}(\tau_2), \ldots \texttt{Quant}(\tau_{n-2})).$$

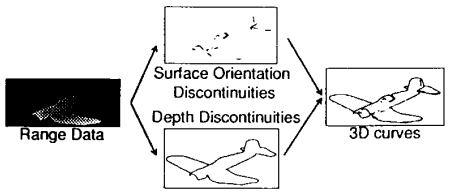


Fig. 3. Generation of 3-D curves.

All the encoded 3-D super segments serve as keys into a table (the database), where we record the corresponding 3-D super segments as entries, as explained later. We now address the following issues:

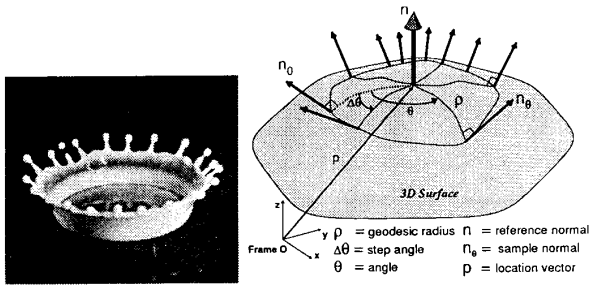
Curve Extraction: Thus far, we have not discussed the curve extraction process (see Fig. 3). We compute the edges of the range image. We use the edge detection algorithm proposed by Saint-Marc *et al.* [21]. It detects surface and depth orientation discontinuities. These features are inferred by examining the zero crossings and extremal values of the surface curvature. The method uses adaptive smoothing to smooth a range image, which preserves discontinuities and facilitates their detection. This is achieved by repeatedly convolving the image with a very small averaging filter whose weights are a function of the local gradient estimate. In order to extract curvature extrema and zero crossings, instead of smoothing the original range image R, the original derivatives $P = \frac{\partial R}{\partial x}$ and $Q = \frac{\partial R}{\partial x}$ are computed first. Then, the images P and Q are repeatedly smoothed. Finally, the curvature values are computed from the smoothed images P and Q. The curvature extrema and zero crossings are extracted using hysteresis.

Polygonal Approximation: Because a polygonal approximation with a fixed tolerance is, in general, not stable, we use multiple line fitting tolerances. A complete analysis is given later. The 3-D super segments are built by grouping adjacent segments. Because we cannot assign a specific direction to a 3-D super segment, we use both directions for our representation.

Choice of Cardinality: Which cardinalities should we use to define the link length of the 3-D super segments? To use a fixed cardinality is possible, but it reduces the flexibility of the matching process. However, when we have long 3-D super segment matches, why not use them? Therefore, we compute *all* possible cardinalities; that means that an open curve approximated by eight linear segments is represented by two 3-D super segments of cardinality 8 (one for each direction), four 3-D super segments of cardinality 7, six 3-D super segments of cardinality 6, and so on. Higher cardinalities of matched 3-D super segments increase the probability of having a *good* match.

B. The Splash

1) Basic Idea: For some objects, however, such as smooth objects, it is impossible to use edges for the representation. Therefore, we come up with a new representation based on small surface patches where we can compute differential properties in a reliable way.



(a) (b)

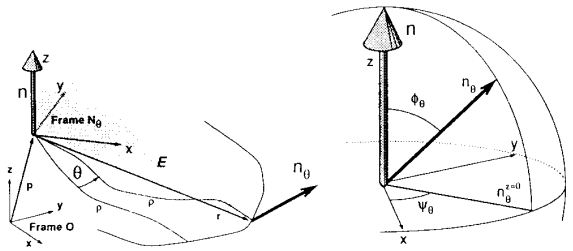
Fig. 4. Splashes: (a) Milk splash; (b) splash.

Extending the super segment idea is not straightforward. The polygonal approximation of a curve has a property that is crucial but is not extendable to higher dimensions: the well defined order of the neighborhood of a linear segment. Every segment on a polygon has two adjacent neighbor segments. Based on this fact, super segments can be generated by grouping adjacent segments together. In surface approximations, however, this ordered neighborhood property does not exist. Polygonal or other segmentations of a surface (or volume) lead, in general, to patches that can have any number and order of neighbor patches. This is a reason why we decided not to go the path of a polygonal (or higher order) surface segmentation to obtain a representation for matching and recognition. What are the requirements that a representation for general 3-D objects has to meet? We want the representation to be

1) translation invariant
2) rotation invariant
3) general, meaning that we do not have to make any assumptions about the shape of the object
4) local enough, so that we can handle occlusion
5) robust enough, so that we can handle noise.

In the following, we will use `lower case` letters to describe vectors (n, p...) and `upper case` letters to describe coordinate frames (N, O...). The basic feature for representing a general surface patch is the *splash*. The name originates from the famous picture of Prof. Edgerton (at M. I. T.) showing a milk drop falling into milk (see Fig. 4(a)). This picture bears a resemblance to the normals in our basic feature. A splash is best described by Fig. 4(b). At a given location p, we determine the surface normal n. We call this normal the *reference normal* of a splash. A circular slice around n with the geodesic radius ρ is computed. Starting at an arbitrary point on this surface circle, a surface normal is determined at every point on the circle. Practically, we walk around the reference normal with a $\Delta\theta$ angle (typically $1° \leq \Delta\theta \leq 15°$) and obtain a set of sample points on the surface circle. The normal at the angle θ is called n_θ. A *super splash* is composed of splashes with different surface radii ρ_i with $i\epsilon\{1, \ldots m\}$, where m is the number of splashes in a super splash.

In other words, the splash is the representation of a surface patch by the Gaussian map in the vicinity of the center of the patch, mapping the tangent with respect to a geodesic distance ρ. One question that is often asked is why do



(a) (b)

Fig. 5. n and n_θ: (a) Relationship between n and n_θ; (b) definition of ϕ_θ and ψ_θ.

we not use Gaussian curvature for the splash computation (which is invariant to rigid transformations)? Why do we use the tangent information? The answer is straightforward: The computation of curvature requires a higher order derivative than the tangent. This implies that the signal-to-noise ratio is lower for a curvature-based representation than for a tangent-based scheme. Therefore, from the practical viewpoint, we prefer a more reliable representation for the purpose of efficient matching.

We compute a normal in our system by approximating the environment of a normal with triangular patches of small sizes. Every triangle votes for a triangle normal. The average of the three closest triangle normals is the surface normal. This is a very rough method, but the results were always good enough for our approach.

The frame $\mathtt{N}_\theta$ (see Fig. 5(a)) is defined in the following way:

1) The surface normal $\mathtt{n}$ is the z axis.
2) At every location of $\mathtt{n}_\theta$, the location of the reference normal $\mathtt{p}$ and the tip of the reference normal $\mathtt{n}+\mathtt{p}$ describe a plane E. The x axis is defined as the vector that is perpendicular to $\mathtt{n}$ and lies in the plane E. Furthermore, the angle between the x axis and a vector $\mathtt{r}$, which is defined between the origin of Frame $\mathtt{N}_\theta$ and the location of $\mathtt{n}_\theta$, has to be in the interval $[-90°, 90°]$.
3) The y axis is perpendicular to the x and the z axes in a right-handed coordinate system.

This frame has the property that the xy plane always approximates the tangent plane of the surface in $\mathtt{p}$. We represent $\mathtt{n}_\theta$ in spherical coordinates; we compute the two angles ϕ_θ and ψ_θ:

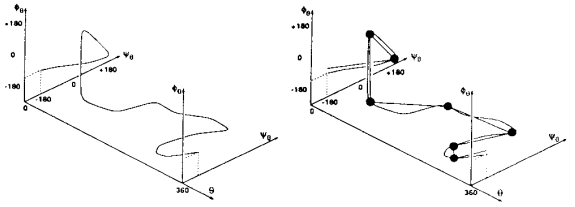
$$\phi_\theta = angle(\mathtt{n}, \mathtt{n}_\theta)$$
$$\psi_\theta = angle(\mathtt{x}, \mathtt{n}_\theta^{z=0}).$$

For every sample point of a splash, we obtain such a pair. Now, we have a 2-D mapping $\phi(\theta)$ and another one $\psi(\theta)$. In an earlier implementation (see [24]), we used these two mappings in parallel for our algorithm. However, it is in fact possible to combine these two mappings into one compact 3-D vector mapping $\vec{v}(\theta) = \begin{pmatrix} \phi(\theta) \\ \psi(\theta) \end{pmatrix}$, which describes a curve in the 3-D space (ϕ,ψ,θ). By doing so, we are able to use the representation for the general 3-D curve from Section III-A for the representation of the mappings of the splash. Drawing a mapping for ϕ and ψ with respect to θ results in a mapping illustrated in Fig. 6(a). This mapping has the following properties:

1) Dependent on where $\mathtt{n}_0$ is, the mapping is shifted along the θ axis.
2) The mapping is periodic with respect to the θ axis.
3) The variation of the curve represents the structural change in the surface environment around the reference normal $\mathtt{n}$.
   a. For a splash on a sphere or a plane, the mapping is constant.
   b. A creased surface results in a curved mapping.
4) Splashes that are located close to each other have a similarly shaped mapping. By using the word *similar*, we mean similarity in the sense that a human would classify them as "pretty much the same." That does not automatically imply that the pairwise difference results in small values (we discuss the issue of robustness further in Section III-C). To be able to compare two mappings, we therefore need a difference measure, which is introduced below.

2) Encoding: At this point, we have reduced the original question ("How do we capture the shape of a general surface patch into a representation?") into the much simpler question "How do we capture the shape of a mapping into a representation?" The solution is straightforward based on our 3-D approach for representing a general 3-D curve (see Section III-A).

1) For all splashes of a model, we compute their mappings. In Section III-B-3, we discuss the selection of the locations of the splashes.
2) For each splash, the mapping is approximated by polygonal approximations (see Fig. 6(b)). It is important to note that the mapping is periodic, and therefore, the polygon is closed. For the purpose of robustness, we use multiple line fitting tolerances. Therefore, we get a set of polygons for each mapping.
3) For every polygonal approximation, we compute a 3-D super segment. The start of the 3-D super segment is defined at the point with the maximal distance of the θ axis. This corresponds to the point at which the sample normals have the strongest tilt with respect to the reference normal. If there is more than one global maximum, we use one 3-D super segment for each of the maxima. With this 3-D super segment choice, we obtain rotational invariance in our representation. By starting all 3-D super segments at the maximum of the approximation, two shifted polygons with the same shape produce the same 3-D super segment.
4) All the obtained 3-D super segments are encoded. The encoding works as described in Section III-A. As encodable attributes, we take
   a. the curvature and the torsion angles of a 3-D super segment
   b. the maximum distance of the mapping from the θ axis
   c. the surface radius of the splash.



(a) (b)

Fig. 6. Vector mapping $\vec{v}(\theta)$: (a) Mapping of ϕ and ψ into the (ϕ, ψ, θ) space; (b) polygonal approximation of the mapping.

Incorporated in the code of the angles of the 3-D super segments is also the cardinality (number of segments) of the 3-D super segments (by the number of angles). That avoids matching 3-D super segments of different cardinality. The encoding of the maximal distance allows one to distinguish between different curved surfaces of the same shape (e.g., two spherical surfaces with different sphere radii). The encoding of the radius avoids matches between splashes with different splash radii. In summary, the code for one splash mapping is

$$\begin{aligned}&\texttt{Code}(\textit{splash-mapping}) = \\ &\quad (\texttt{Quant}(\kappa_1), \texttt{Quant}(\kappa_2), \ldots \\ &\quad \texttt{Quant}(\kappa_n), \texttt{Quant}(\tau_1), \texttt{Quant}(\tau_2), \ldots \\ &\quad \texttt{Quant}(\tau_n), \texttt{Quant}(\texttt{max}), \texttt{Quant}(\texttt{radius}))\end{aligned}$$

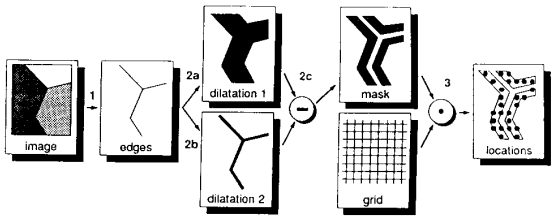
where κ_i is the i^{th} curvature angle of the 3-D super segment, τ_i is the ith torsion angle of the 3-D super segment, and n is the cardinality of the 3-D super segment.

5) All the encoded 3-D super segments serve as keys into a table (the database), where we record the corresponding splashes as entries, as will be explained later.

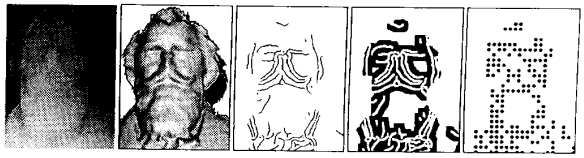
3) Interest Operator: One question remains open: At which locations of an object should we compute the splashes? The brute force answer would be the following: at every pixel (in a range image). A more realistic answer would include the observation that we will not get *structurally rich* splashes, which lead to good and unambiguous matches, at every point. Splashes in flat areas result in 3-D super segments with extremely low cardinality (e.g., a splash on a plane maps on a 3-D super segment consisting of one segment that corresponds to a cardinality of one). Super segments with such low cardinalities are less descriptive than super segments with higher cardinalities, which represent high structured surface patches. Therefore, to obtain good and unique matches, we are interested in matches of structured patches and high cardinality. These can be found at or near points of high curvature. Our simple selection method works as follows (see Fig. 7):

1) To compute the edges (surface and depth discontinuities), we use the algorithm mentioned in Section III-A.
2) We want to position the splashes in areas where we can expect structured patches on *one* object. This property is not given on a boundary. A boundary edge typically has the object as one neighborhood and other objects or background information as the other neighborhood. Therefore, we use only the "inner object edges" and throw away the boundary edges.
3) For positioning the splashes, we are interested in areas around the edges. Placing a splash on a high curvature point has the disadvantage of an unreliable reference normal. A reliable reference normal is important for a stable splash. Nevertheless, we want to capture the structure of the edges in the splash. Therefore, the best place for a splash is in the neighborhood of an edge. We get this area in three steps:
   a. We dilate the edge image by replacing every pixel on the edges by a disc of a certain radius (e.g., r_1 =8 pixels). The resulting image is called *dilatation 1*.
   b. We dilate the edge image with another radius (e.g., r_2 =3 pixels with $r_1 > r_2$). The resulting image is called *dilatation 2*.
   c. The subtraction of dilatation 1 and dilatation 2 gives us a mask. This mask describes an area with the above-described characteristics. Points in this mask are not high curvature points, but they are close to edges.
4) We compute a grid of splashes on the range image with respect to this mask.

This is obviously not the only way to choose the locations of splashes, but as we will see in the result section, this simple method works quite well. Therefore, we have not emphasized the direction of this research.



(a)

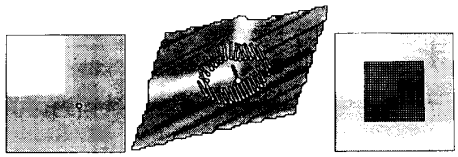


(b) (c) (d) (e) (f)

Fig. 7. Interest operator scheme on an example: (a) Scheme; (b) Brahms head range data; (c) Brahms head (artificially shaded); (d) curvature zero crossings and extrema; (e) interest mask; (f) locations.

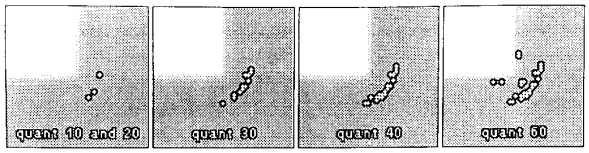
C. Robustness and Stability of the Splash Representation

In order for our representation to be useful in recognizing objects from real data, it is necessary to address the issues



(a) (b) (c)

Fig. 8. Corner surface: (a) Location of the reference splash; (b) qualitative side view; (c) 25 × 25 grid of splashes.



(a) (b) (c) (d)

Fig. 9. Matched splashes with different quantizations.

of robustness and stability of the splash representation in the presence of noise.

Location: We cannot make the assumption that two splashes on different views of the same object are at *exactly* the same location with respect to the object. Therefore, we have to examine the robustness of a splash regarding location uncertainty. We discuss the problem in Section III-C-1.

Orientation: Even when the location is correct, we cannot assume that the reference normal in both splashes is exactly the same. What are the effects on representation and matching of an error in the reference normal orientation? We address the issue in Section III-C-2.

Stability: How much noise can be added to the surface patch and still have a stable representation? We discuss the stability problem in Section III-C-3.

We have found empirically that our representation is adequate since our system performs well on real data. We would have liked to model the behavior of our scheme on arbitrary free-form surfaces, but they are too general to be of any help. Instead, we present a full analysis on a simple analytic model of a specific surface patch, as shown in Fig. 8(a) and (b), for all three issues (location, orientation, and surface stability). Furthermore, for the orientation robustness issue, we study the effects of noise in the general case of a 3-D curve (representing either one of our features).

1) Robustness in the Location: To answer the question of how robust the representation of a splash is with respect to location accuracy, we use empirical data based on an example environment as described above. This range image was generated artificially and smoothed with a Gaussian filter. As shown in Fig. 8(a), our reference splash is centered southeast of the corner. We show only the location of the splash. The corner has a height of 5 pixels. The reference splash has a radius of 30 pixels, and the linear approximation of the mapping leads to a cardinality of 3. This result was observed for the whole set of fitting tolerances (15, 20, 25, 30). This means that the key consists of six values: three curvature and three torsion values. For reasons of simplicity, we did not encode the maximal distance from the θ axis. In our test results, we got qualitatively similar results as the ones shown here. We can also ignore the encoding of the splash radius because in our example, we deal only with splashes of one fixed radius. A qualitative side view of the reference splash, its sample normals, and the shaded underlying surface is displayed in Fig. 8(b). The lines on the surface are artifacts from the rotation.

We use our reference splash and match it against a grid of 625 splashes (see Fig. 8(c)) in the vicinity of the reference splash spaced 2 pixels apart. In Fig. 9, we show the results for different quantizations of the matching key. We show in Fig. 9(a) the matched splashes between the reference splash in Fig. 8(a) and the splashes in Fig. 8(c), quantized with an interval size of 10° (the same result was observed for 20°). Fig. 9(b) shows the matches for a quantization of 30°, Fig. 9(c) for 40°, and Fig. 9(d) for 60°.

For small quantizations, the patch is recognized only if the splash is very close to the location of the original splash, whereas larger quantizations allow a much larger latitude in location uncertainty. Of course, if the quantization is too large (as in Fig. 9(d)), the splash may be confused with splashes from other locations. We therefore observe the desired robustness with respect to location uncertainty for this surface patch for the quantizations 30 and 40.

2) Robustness in the Reference Normal: **Influence of Noise on the Mapping**—In this section, we examine the influence of noise in the reference normal on the representation of a splash. In a first step, we focus on the effect on the mapping. In other words, adding an error angle ε in the direction δ to the reference normal, how is the curve $\vec{v}(\theta) = \begin{pmatrix} \phi(\theta) \\ \psi(\theta) \end{pmatrix}$ influenced? In Fig. 12, we show the frame F, which is the frame for a certain $\mathtt{n}_\theta$. The frame F consists of the axis x, y, and z. As in Section III-B-1, z is equivalent to the reference normal n. By adding an error angle ε in the direction δ to z, we get z'. Without loss of generality, we can set $\delta = \theta$. The frame F′ consists of the axis x', y', and z' with

$$z' = \begin{pmatrix} \sin(\varepsilon)\cos(\theta) \\ \sin(\varepsilon)\sin(\theta) \\ \cos(\varepsilon) \end{pmatrix}$$

$$y' = z' \times \mathtt{r} \quad x' = y' \times z' = (z' \times \mathtt{r}) \times z'$$

with r defined as in Fig. 5 in Section III-B-1. The computation of the change of ϕ with respect to θ is straightforward:

$$\begin{aligned} \Delta\phi_\theta &= \phi_\theta - \phi'_\theta \\ &= \arccos(z\mathtt{n}_\theta) - \arccos(z'\mathtt{n}_\theta) \\ &= \arccos(\mathtt{n}_\theta^z) - \arccos\big[\sin(\varepsilon)\cos(\theta)\mathtt{n}_\theta^x \\ &\quad + \sin(\varepsilon)\sin(\theta)\mathtt{n}_\theta^y + \cos(\varepsilon)\mathtt{n}_\theta^z\big]. \end{aligned}$$

The computation of $\Delta\psi_\theta$ is not as straightforward. The problem lies in the fact that the x axis is dependent on the data in the vicinity of the splash, expressed by the vector r. To overcome this problem, we have to introduce some simplifying assumptions:

1) We can assume that ε is small (smaller than 10°).
2) We can further assume that the angle between r and x is small. We observed this angle for several thousand

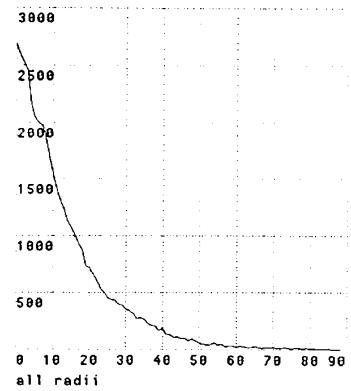
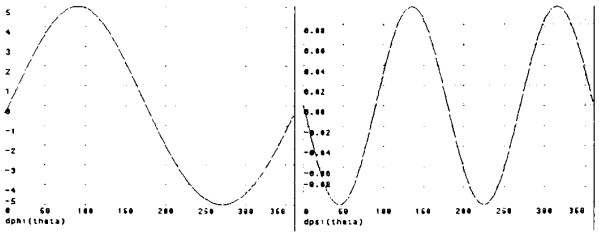


Fig. 10. Histogram of angle distribution.



(a) (b)

Fig. 11. Example for $\Delta\phi(\theta)$ and $\Delta\psi(\theta)$: (a) $\Delta\phi(\theta)$; (b) $\Delta\psi(\theta)$.

splashes on real data, and we obtained the result that in approximately 75% of the cases, the angle was smaller than 15°. In Fig. 10, we show the histogram of the distribution of the angles computed from real data (we used the data of the busts, which we discuss in the results section). The abscissa represents the angles in degrees, and the ordinate represents of the number of occurences.

This allows us to approximate $\mathbf{r} \approx x$ and $\mathbf{n}_\theta^{z'=0} \approx \mathbf{n}_\theta^{z=0}$. We therefore get

$$x' \approx (z' \times \mathbf{r}) \times z'$$

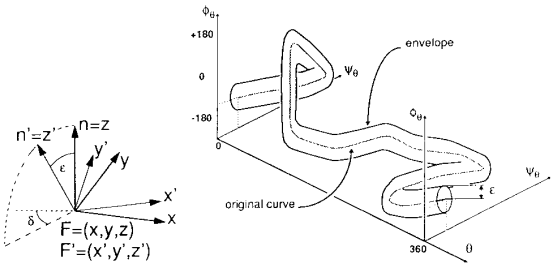
and

$$\begin{aligned}\Delta\psi_\theta &= \psi_\theta - \psi'_\theta \\ &= \arccos(x\mathbf{n}_\theta^{z=0}) - \arccos(z'\mathbf{n}_\theta^{z'=0}) \\ &= \arccos(\mathbf{n}_\theta^x) - \arccos\left[\mathbf{n}_\theta^x \cos^2(\varepsilon) + \right. \\ &\quad \left. \mathbf{n}_\theta^x \sin^2(\varepsilon)\sin^2(\theta) - \mathbf{n}_\theta^x \sin^2(\varepsilon)\sin(\theta)\cos(\theta)\right].\end{aligned}$$

The graphs of $\Delta\phi_\theta$ and $\Delta\psi_\theta$ with $\mathbf{n}_\theta = (0, \frac{1}{\sqrt{3}}, \frac{2}{\sqrt{3}})$ (assuming the sample normals are all equal) and $\varepsilon = 5°$ are displayed in Fig. 11(a) and (b), respectively. We have verified qualitatively the shape of the curves with real data.

Our second step is to look at the maximum error on the curve $\vec{v}(\theta)$. Based on the triangle equation, which is also valid for the spherical geometry, we can bound the maximal error for $\Delta\phi$ by

$$\|\Delta\phi\| = \|\phi - \phi'\| \leq \varepsilon.$$



(a) (b)

Fig. 12. (a) Frame F and frame F'; (b) envelope of curve in Fig. 7(a).

For the maximum of $\Delta\psi$, we can show under the above defined assumptions that the transformation T with $T : \mathrm{F} \rightarrow \mathrm{F}'$ is a rotation by the angle ε around an axis k, which lies in the xy plane and intersects the origin of F (and F'). This implies that the angle between x and x' can be at most ε. Therefore, we get, based on the triangle equation, a bound for the maximal error for $\Delta\psi$ as

$$\|\Delta\psi\| = \|\psi - \psi'\| \leq \varepsilon.$$

We can now address the problem of how the representation might vary. Instead of a single curve $\vec{v}(\theta)$, we have now to deal with an envelope of curves $\vec{V}(\theta, \varepsilon)$, which is a disc of radius ε swept along $\vec{v}$ (see Fig. 12(b)).

Influence of Noise on the Polygonal Approximation—What are the polygonal approximations that approximate such a set of curves? How are the angles between consecutive line segments affected? To make a general statement is very difficult. Therefore, we try to approach the problem by making some simplifying assumptions:

- We look at the 2-D problem.
- We assume constant curvature.

This means that we try to do a polygonal approximation along a circular section with a radius r_1, and we examine the changes of the polygonal approximation when we change the radius by an error ε to r_2 ($r_2 = r_1 + \varepsilon$, curvature $\kappa_i = \frac{1}{r_i}$). In Fig. 13, we start the polygonal approximation with the line fitting tolerance t at the point P. The first segment from P to Q_1 is a linear approximation for the circle C_1 with the radius r_1 for the curve from P to Q_1. The second segment from Q_1 to R_1 is a linear approximation for the circle C_1 for the curve from Q_1 to R_1. The two segments form the angle α_1. Changing the radius of the circle from r_1 to r_2 and starting the polygonal approximation at P with the same line fitting tolerance t leads to the points Q_2 and R_2. The two segments form the angle α_2. Our interest lies in the question of the difference $\Delta\alpha$ between α_1 and α_2.

From Fig. 13 and with the relationship $t = r_i(1 - \cos(\frac{\mu_i}{2}))$, we can derive geometrically the following relationship between $\Delta\alpha = \|\alpha_1 - \alpha_2\|$, r_1, r_2, and the tolerance t:

$$\Delta\alpha = \|2acos\left((1 - \frac{t}{r_1})(1 - \frac{t}{r_2}) + \sqrt{\frac{t^2}{r_1 r_2}(2 - \frac{t}{r_1})(2 - \frac{t}{r_2})}\right)\|.$$

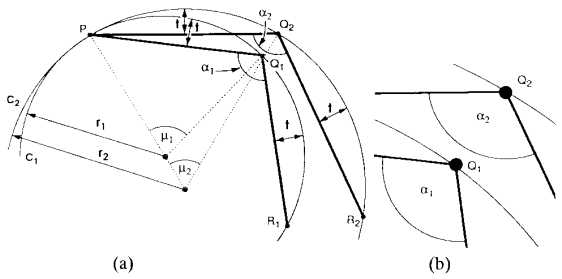


Fig. 13. Linear approximation: (a) Two different curvatures; (b) closeup.

This equation is plotted in Fig. 14 for the values $\varepsilon = 5$ and $t = 5, 10, 15, 20, 25, 30$ with respect to radius r_1. Our goal is to obtain a stable polygonal approximation. We choose a tolerance $t \gg \varepsilon$ for the following reasons (the numbers after the item correspond to the graph in Fig. 14):

Small Radius (0–20): A small radius, which corresponds to a sharp corner is a "wanted" breakpoint for the approximation. A small radius, which corresponds to noise, is ignored by an approximation with large t (see the start of the curves for the tolerances $t = 15, 20, \ldots$ in Fig. 14). A small tolerance leads to extremely large values for $\Delta\alpha$.

Large Radius (50+): The approximation of parts of the curve with a large radius are very stable (see the low values for $\Delta\alpha$ in Fig. 14). The choice of the tolerance is not crucial. All tolerances provide small values for $\Delta\alpha$.

Medium Radius (20–50): The problems of an unstable approximation start when the radius is in between the above-discussed large and small radius. We observe two effects:

1) The values for $\Delta\alpha$ are fairly large.
2) The linear approximation might "break" at different points, depending on little changes of the curve.

This is the reason why we do not use one fixed tolerance for the approximation but a whole set of tolerances.

We use the splash described in Section III-C-1 to examine its redundancy with respect to the reference normal. The results can be seen in the graphs in Fig. 15 for different quantizations (10, 20, 30, 40, and 60). The distance from the origin of the graph represents the angle α, and the direction, starting north and counting counterclockwise, represents the angle δ. Whenever a splash with the corrupted normal defined by (α, δ) matches the original splash (defined by the error $(0, 0)$), we plot a dot. Notice the different scales of the graphs.

To summarize, for small quantizations, the patch is recognized only if the reference normal is very similar to the reference normal of the original splash (see Fig. 15(a)). An uncertainty of up to 2° is bearable only for a certain δ direction. Larger quantizations allow a much larger error in the reference normal and more freedom in the δ direction. The shape of the different distributions with respect to δ is dependent on the underlying surface and the quantization.

3) Stability with Respect to Noise: We want to address the question of how corrupted can the underlying surface be to reduce the stability of the splash representation. We added zero mean Gaussian noise $a\mathcal{G}(0, \sigma)$ with the amplitude a (in pixels) and the standard deviation σ to the surface s to get the

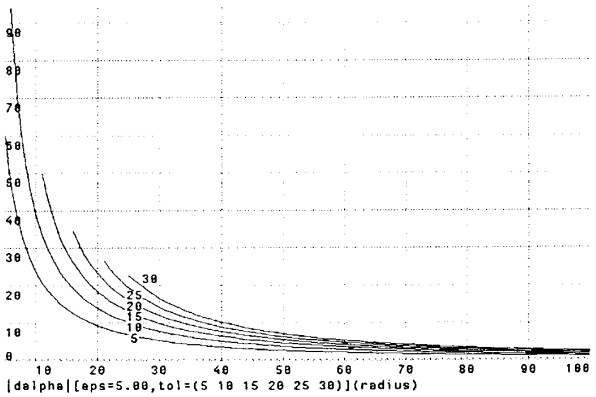


Fig. 14. $\Delta\alpha$.

corrupted surface cs:

$$cs(x, y) = s(x, y) + a\mathcal{G}(0, \sigma).$$

Then, we compute the splash at the same location as the reference splash on the corrupted surface. For a given quantization, we then match the "corrupted" splash against the reference splash. If both splashes match, we put a point in the graph for the corresponding quantization at the coordinate (σ, a). We show the scatter graph for the quantization 40 in Fig. 16(a), and the graphs of the envelopes for the quantizations 10, 20, 30, 40, and 60 are shown in Fig. 16(b), respectively.

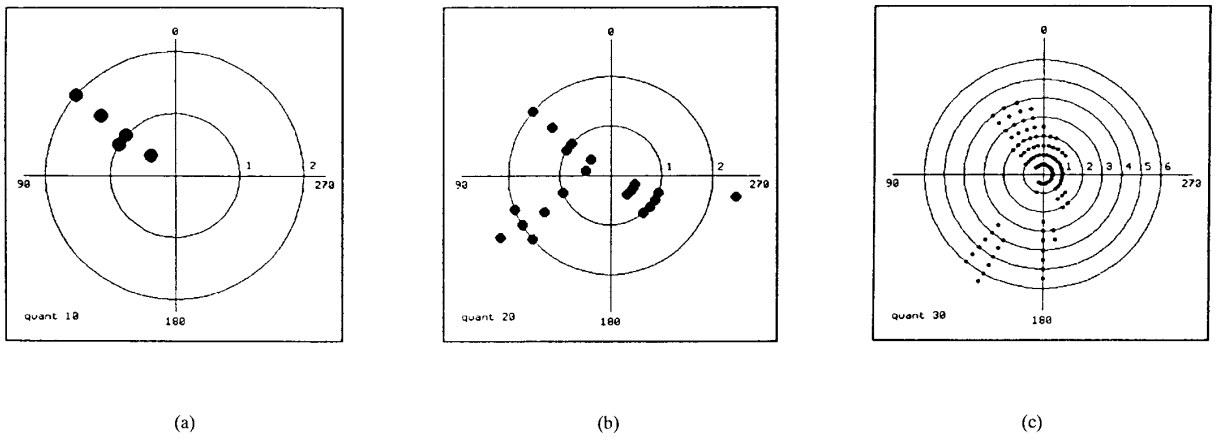
Our conclusion is that if the quantization is larger, the more stable is the splash representation, and if the quantization is smaller, the less stability we can observe. The matching behavior is not affected by either increasing the amplitude of the noise and keeping the standard deviation small or decreasing the amplitude and having a large noise deviation.

D. Recognition

1) Object Representation: As mentioned in the previous sections, we want to represent our model (or scene) with super segments for curve representation and splashes for surface patch representation. We want the representation to be compact and fast accessible, and the storage of multiple objects should be possible. We chose, for these reasons, a table that is implemented as a hash table (for more about hashing, see [25]). A hash table allows efficient storage (only pointers are recorded), the hashing scheme allows fast access, and different features with the same keys can be stored in cellar-like buckets.

The representation of an object consists of the following steps (see Fig. 17):

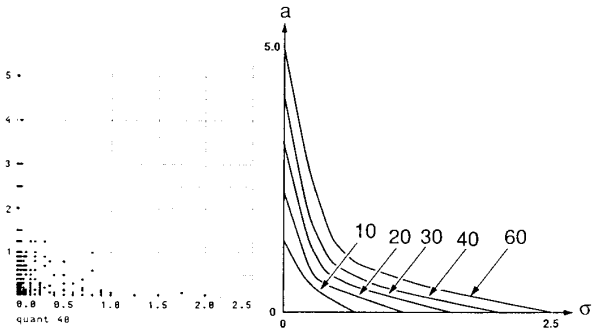
1) Compute the features ${}^{\mathtt{m}}\mathtt{F}_{\mathtt{i}}$ of model $\mathtt{m}$ with respect to the algorithms described in Section III-A for super segments and Section III-B for splashes.
2) Encode the features:
   a. Encode the curvature and torsion angles for the 3-D super segments (Section III-A).
   b. Encode the curvature angles and the other attributes of the mappings of the splashes (Section III-B-1).



(a) (b) (c)

(d) (e)

Fig. 15. Matched splashes with different quantizations.



(a) (b)

Fig. 16. Stability graph: (a) Scatter graph for quantization 40; (b) envelope of different quantization.

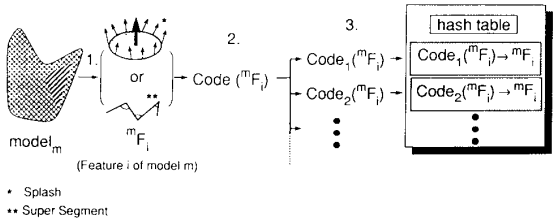


Fig. 17. Object representation.

3) For every feature $^{m}F_i$, several codes $Code_j(^{m}F_i)$ are computed. They are related to the different line fitting tolerances. Every code of every feature serves as a key for an entry in a table (the database) where we record the feature.

When we build the database for more than one object, we perform the three steps for every object. The table (data base) grows in size with the number of recorded models. This process of building the database can be done off line.

2) Hypotheses Generation: **Candidate Retrieval**—The task of hypotheses generation is the process to establish correspondences between features of stored models and features of the scene. This process results in a set of matching hypotheses that consist of *good* and *false* matches. To extract the good hypotheses from all hypotheses, we have to find consistent clusters; this process will be discussed in Section III-D-3. Our main interest for the candidate retrieval lies in the discriminative power of the hypotheses generation itself. By using indexing, we gain a lot of this power. Several

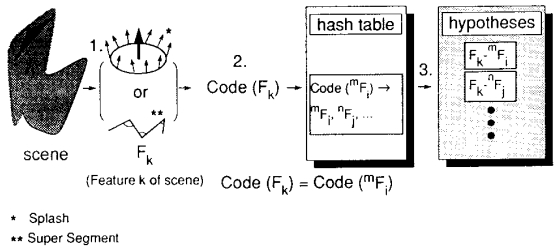


Fig. 18. Hypotheses generation.

systems of the past (see Section I) use, in this respect, very weak features, such as points or edges. This leaves all the discriminative work to the verifying step. We believe that our indexing mechanism reduces the ratio of false hypotheses to good hypotheses tremendously. One positive side effect is that in our experiments, models that were not in the scene and therefore provided only false matches had very few hypotheses and could be excluded fairly fast by the verification.

The hypotheses generation (see Fig. 18) consists of the following steps:

1) The scene is preprocessed to generate all the features F_k (splashes and super segments) as explained in Section III-D-1.
2) The features are encoded.
3) The encoded features are used to retrieve the candidate hypotheses between the features of model and scene.

Quantization and Cardinality—A crucial parameter for the hypotheses generation is the quantization size for the encoding of the features. Two features match when both keys are exactly the same. This means that all the pairwise values have to fall into the same quantization intervals. Several questions have to be raised:

- Which quantization (interval size) is the best?
- Should we use different quantizations for different cardinalities?
- Which cardinalities occur in real data and with which frequency?

To answer the first two questions, we have to look at the probability of a match, given a certain key length n. We assume equal quantization for all keys. Suppose the range is quantized into intervals of the same size q. Each value v is then assigned a key based on which interval v falls into. If the value v is corrupted by a random additive term bounded by ε, the probability that $v + \varepsilon$ is assigned the same key as v is simply

$$p(k|v) = 1 - p(k-1|v+\varepsilon) - p(k+1|v-\varepsilon) = 1 - \frac{\varepsilon}{q}.$$

When we have a vector of such values v of length n, the probability that the corrupted vector is assigned the same entry as the original one is

$$p^n(k|v) = (1 - \frac{\varepsilon}{q})^n.$$

Looking at this equation, it is obvious that the same quantization for different keys increases the probability for matches of small cardinality and decreases the probability for matches of large cardinality. In our implementation, we try to counteract this effect by using larger interval sizes for larger cardinalities. Typical values are as follows (only for the quantization of the curvature and torsion angles):

cardinality	3	4	5	6	7	8	...
number of keys	6	8	10	12	14	16	...
interval size	30	40	45	60	60	90	90

For extremely noisy data we use slightly larger values, for data with little noise we use smaller values.

What kind of splashes do occur in typical range data? Statistics for the three composers (see results section) and the combined scene gives us the table at the bottom of this page.

From our experience, we get the best matches from cardinalities 3 to 7. We believe that this is mainly due to the lower probability of matches for larger cardinalities in the equation discussed above.

3) Verification: The task of the verification is to distinguish *good* from *bad* hypotheses. Good hypotheses correspond to true matches, and bad hypotheses correspond to wrong matches. Good hypotheses have the following properties:

- They correspond to a rigid transformation.
- They can be grouped in geometrically consistent clusters.

Therefore, the verification stage consists of the following steps:

1) We compute all possible matches for the features of the scene with the model features to generate multiple hypotheses. We remove the hypotheses that do not represent a rigid transformation. This can be done for every hypothesis by computing a least squares match between the model feature and the scene feature. The determinant of the resulting rotation matrix should be approximately 1.0 to represent a rigid transformation (see Appendix A). Next, we divide the resulting n hypotheses $H = \{h_1, h_2, ...h_n\}$ according the model for which the model feature of the hypothesis votes.

cardinality	3	4	5	6	7	8	9	10	11	12	13	14	15	16
occurrence	2162	3240	3254	2923	2134	1336	715	300	114	36	8	8	1	1
percentage	13.3	19.9	20	18	13.1	8.2	4.4	1.8	0.7	0.2	0.04	0.04	≈0	≈0

We store these into a correspondence table, where we have the models m_i as keys and the i_k hypotheses $H_i = \{{}^ih_1, {}^ih_2, \ldots {}^ih_{i_k}\}$ (with $H_i \subseteq H$) as entries (see Fig. 19).

2) The next step is the formation of consistent clusters. For every model m_i, we have to check the hypotheses ${}^ih_x$ and ${}^ih_y$ with ${}^ih_x \neq {}^ih_y$ that are consistent with each other. In theory, one matched feature is enough to establish the complete transformation between model and scene. In practice, we have to consider two aspects:
   a. The splash is very local.
   b. Due to this locality, noise has a lot of influence on the transformation.

   Therefore, we view, for the geometric analysis, a hypothesis as a match equivalent to a point match. In practice, we use the locations of the splashes as the matching point pair, and because three noncollinear point matches define a unique rigid transformation in three dimensions, we adopt the criterion that three consistent hypotheses are sufficient to instantiate the model in the scene. We do not check every hypothesis against every other; instead, if we have three consistent hypotheses $C = \{{}^ih_r, {}^ih_s, {}^ih_t\}$ with $C \subseteq H_i$ for one model m_i, we examine the remaining hypotheses in $H_i \setminus C$ and collect those that are consistent with at least one of the selected three in C. When we have found one instance, represented by $I = C \cup F$, where F is the set of the additional found consistent hypotheses, we try to find more instances in the remaining hypotheses $H_i \setminus I$. What, however, is meant by consistency? We use the powerful geometrical constraints (distance, direction, and angle) introduced by Grimson and Lozano-Pérez [11], [12] to efficiently prune their interpretation trees and build our clusters. For a more detailed discussion of the geometrical constraints, see Appendix B. In the 3-D domain, these three constraints define the attitude of one feature relative to another since it specifies the five degrees of freedom (three translational for the position and two rotational for the orientation).
3) After this grouping of hypotheses into clusters, we can compute the transformation from the model coordinates to the scene coordinates by applying a least squares calculation on all the matching features (see Appendix A). Because of noise, we get, in general, a good first guess for the transformation but not an exact match. A second least squares match on corresponding corners or segments can refine the result.

E. Complexity Analysis

The whole question of the complexity focuses on the question: "What is the discriminative power of the features in the system?" The answer to this question can be split in half. The first part is the analysis of the retrieval process and the number of generated hypotheses. The second part is the discussion of the verification step and the cost of grouping them into consistent clusters. When we talk in the following about "scene features" we consider only the scene

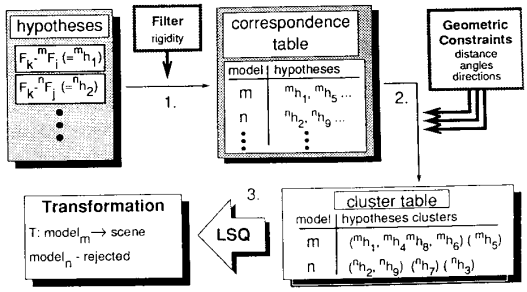


Fig. 19. Verification.

features, which lead to the generation of the minimum of one hypothesis. For the complexity analysis, we ignore the scene features whose code is disjunct from all the keys in the table (they also do not add any cost in practice). Before we start to answer the above question, we look at the parameters involved:

- n number of features in the scene
- d number of features per table entry
- m_s number of models in the scene
- m number of models in the database.

To simplify the discussion, we make the assumptions that every model has the same number of super segments and that the entries are equally distributed over the table ($d = const$). Furthermore, we assume that every model consists of f features ($f = const$).

1) Hypotheses Generation: When we have n features in the scene, it is obvious that the cost to generate all candidate hypotheses with indexing is $O(n)$. The important issue for the cost of the complete recognition is *not* the cost of retrieval, but h, which is the number of hypotheses generated, because this number has a crucial effect on the verification step. The number h is proportional to the number of features per table entry d ($h = d \cdot n$). If h is larger, the clustering into consistent clusters in the following verification is slower. Therefore, we are interested in a small h, which corresponds to a small d. This so called "ability to discriminate," is influenced by the following factors:

Noisy Data: Noisy data in the scene and in the models forces the use of larger quantization intervals for the encoding of the features. Larger quantization increases the number of retrieved hypotheses and therefore decreases the ratio of good versus bad hypotheses in the retrieval phase (see Section III-D-2).

Similar Models: Similar models consist of similar features. Obviously, similar features are less discriminative than distinct features, and therefore, d, which is the number of features per table entry, increases. As a result, a scene feature generates an increased number of candidate hypotheses. The work load of clustering can only be done based on geometrical constraints and is left to the verification step.

Grid Size: Choosing a large grid size for the interest operator, we require the system to be robust to a larger location uncertainty. This forces the user to choose a larger quantization for the encoding of the features and therefore results in a smaller ratio of good versus bad hypotheses.

To summarize, the crucial value for the retrieval stage is h, which is the number of generated hypotheses. In a best case, this number is very low. The theoretical best case is when every hypotheses votes for a different model ($d = 1$ and $n = m_s$). This results in $h = n$. The worst case corresponds to a large value of h. There is only little discriminative power in the features. This means that most features are encoded with the same code (large d). Every scene feature generates all possible candidate hypotheses, and therefore, the overall number of retrieved hypotheses candidates is $h = f \cdot m \cdot n$. This is the approach taken by many systems of the past, which use either points or lines as basic features, and must therefore perform the discrimination task during the verification step.

2) Verification: As mentioned above, the task of the verification is to cluster the h hypotheses into mutually consistent clusters. This is done for every entry in the correspondence table.

Best Case: In the best case, there is a lot of discriminative power in the features. This corresponds to a low d value. The $h = d \cdot n$ hypotheses are divided in the correspondence table according to the model for which the hypothesis votes. For m_s models in the scene, we have m_s entries in the correspondence table with $\frac{h}{m_s}$ hypotheses each. For every entry, these $\frac{h}{m_s}$ hypotheses have to be grouped with respect to the geometrical constraints. We distinguish the clustering process based on different cases:

1) *Every model in the scene occurs only once*: In this case, the hypotheses in one entry either vote for the model instance, or they are the wrong hypotheses. The grouping of the hypotheses can be done by finding three consistent hypotheses (see Section III-D-3) and then screening the remaining hypotheses in the entry for consistency with the initial three. Under the assumption that the number of wrong hypotheses is relatively small, this grouping is done in $O(\frac{h}{m_s})$. For the whole complexity, we therefore get

$$O_{best}(n) = m_s \cdot O(\frac{h}{m_s}) = O(d \cdot n) = O(n).$$

The theoretical absolute best case is when $d = 1$, and every hypothesis votes for a different model ($n = m_s$). The correspondence table has m_s entries with one hypothesis each. The cost for the clustering is zero. This results in $O_{best}(n) = O(1)$. (Note that the pose estimation computed from a single feature is likely to be very coarse.)

2) *Every model in the scene can occur more than once*: In this case, the clustering of the hypotheses in one entry cannot be done in $O(\frac{h}{m_s})$. To find the three initial hypotheses for every instance, we get the complexity of $O(\frac{h^2}{m_s^2})$. Clustering these hypotheses for all entries results in a complexity of

$$O_{best}(n) = m_s \cdot O(\frac{h^2}{m_s^2}) = O(\frac{d^2 \cdot n^2}{m_s}) = O(n^2).$$

The best case has a noteworthy side effect. Assuming the number of scene features n is fixed and examining the complexity O_m with respect to the models in the database, we find that it grows as $O_m = O(k \cdot m)$ when m is the number of stored models and $k < 1$. To show this effect, we performed two experiments:

1) *Large database using only edge information*: We created a database of 100 random polyhedra consisting of mutually intersecting random tetrahedra and parallelepipeds. The scene was composed by taking three of these polyhedra, rotating them, and overlapping them. For the recognition, we used only 3-D super segments. The cost for detecting the absence of a model is less than 10% of the cost for the detection of the occurrence. This underlines the discriminative power of super segments in three dimensions.
2) *Large database using only splashes*: We created a database of 100 random smooth range images. The scene is a composite of four of these objects including translation, rotation, and occlusion. For the recognition, we used only splashes. In our results, the cost for detecting the absence of a model is less than 50% of the cost for the detection of the occurrence. We believe that the cause for the higher relative cost of absence detection based on splashes compared with the super segments lies in the fact that splashes for surfaces are less descriptive than super segments for boundaries.

The experiment gives us an upper and lower bound for that which we have to expect for the detection of occurrence and absence of a general 3-D object using a large database in the case of discriminative features. Therefore, the complexity with respect to the number of models stored is $O_m = O(k \cdot m)$ with $0.1 < k < 0.5$.

Worst Case: In the worst case, there is little discriminative power in the features. This corresponds to a high value for d. The overall number of retrieved hypotheses candidates is $h = d \cdot n$ with $d = m \cdot f$. These h hypotheses are divided in the correspondence table according to the model for which the hypothesis votes. For m models in the database, we get, in the worst case, m entries in the correspondence table with h hypotheses each. For every entry, these h hypotheses have to be grouped with respect to the geometrical constraints. Clustering these h hypotheses results in a complexity of

$$O_{worst}(n, m) = m \cdot O(h^2) = O(f^2 \cdot n^2 \cdot m^3) = O(n^2 \cdot m^3).$$

In the worst case, the ratio of good versus bad hypotheses is very small.

3) Summary: As a conclusion, we get the result that the practical complexity of our system is

$$O(n) \leq O_{recognition} \leq O(n^2 \cdot m^3).$$

In the case of well-distinguishable models, such as the first two examples in the results section, the complexity comes close to the above discussed *best case*. An example where the system slows down is shown in the third example of the results section, which presents the results of a cluttered scene, which is composed of three composer busts. The system detects the

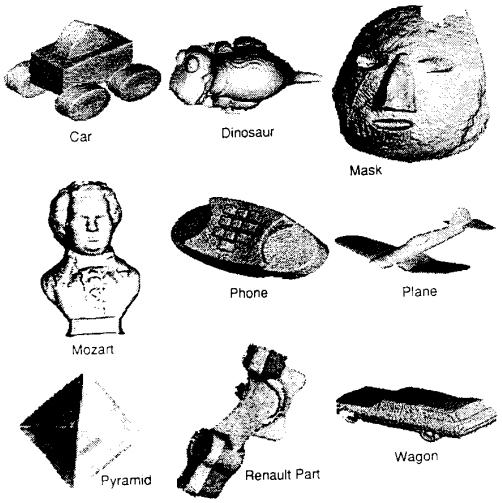


Fig. 20. Database with nine objects.

correct models and computes the correct locations, but due t noisy data and similar features, the discriminative power i smaller, and the overall recognition process is slower than i the other examples.

IV. Results

The recognition mechanism for general 3-D objects is now illustrated with real data examples. For the presentation of the range data, we always display the artificially shaded images. We choose four scenes:

1) a Mozart bust, which is not segmentable into stable patches
2) a plane and a wagon, which shows that our method works for objects that can be approximated by polygonal surfaces (this input was used by Fan in [7] and [6])
3) a very complex and cluttered scene with similar objects (busts of composers)
4) a terrain scene lacking significant features, which is an interesting application regarding navigation tasks.

The Mozart scene and the plane-and-wagon scene were recognized with a database consisting of nine objects. The contents of the database is shown in Fig. 20.

A. Mozart

Our Mozart bust is highly curved and has a partially structured surface. The range data was obtained with a laser range finder. Because of lack of data, we cannot deal with a complete 3-D model and a scene that consists of range data. Therefore, we take the original data of the Mozart bust as the model, and rotate the range data synthetically to obtain the scene. We rotate pixel by pixel and fill the holes by averaging the values of neighbor pixels. This rotation process is guaranteed to add a lot of noise! Our input data is the range image. For better visibility, we show the artificially shaded images. Fig. 21(a) shows the rotated scene of the Mozart bust, which is the model (see Fig. 20) rotated by 20° around a tilted axis. The recognition result is shown in Fig. 21(b). (We overlayed a grid of the range image of the model, which was transformed by the resulting transformation on top of Fig. 21(a).) In Fig. 21(c) and (d), we show the final hypotheses, which lead to the result in Fig. 21(b). Hypotheses 2 and 3 and 18 and 19 are overlaid.

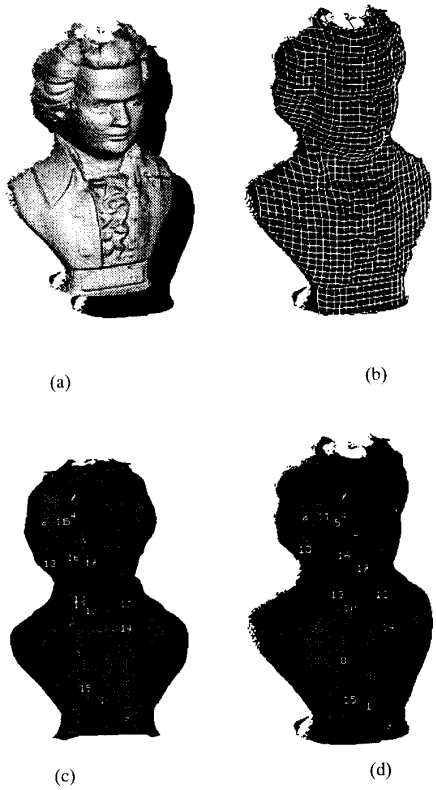


Fig. 21. Example Mozart: (a) Scene 1; (b) detected model projected on scene 1; (c) final hypothesis of Mozart; (d) final hypothesis of scene.

B. Plane and Wagon

We have four range images: two of the plane from different views and two of the wagon from different views. The range data of all four views was obtained with a laser range finder. One wagon and one plane image serve as models. The scene is composed synthetically by combining the other two range images into the scene image as displayed in Fig. 22(a). Fig. 22(b) shows the best detected solution. In Fig. 23, we display the final hypotheses. The splash hypotheses are displayed in *italic*, and the super segment hypotheses are printed in **bold**. Splash hypotheses are shown as the pointers to the corresponding splash location, and super segment pointers point to the middle vertex of the super segment (to display the super segments themselves is difficult). Whenever more than one feature pointer points to a location, we have multiple feature matches based on different linear approximations (for super segments) or different radii (for splashes). It is interesting to note that the plane was recognized based on the super segment information (only hypothesis 2 in Fig. 23(a) and (b) is a splash

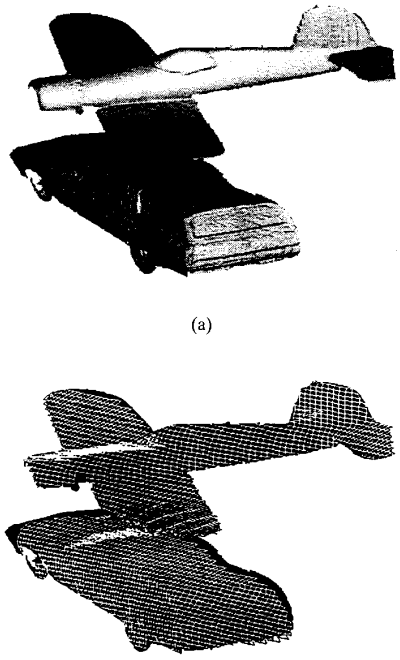


Fig. 22. Example plane and wagon: (a) Scene 2; (b) detected models projected on scene 2.

hypothesis), whereas the wagon was mainly detected based on splashes (only hypotheses (1–3) in Fig. 23(c) and (d) are a super segment hypotheses). This is a good illustration of the system's ability to use whatever information is available; for the wagon, the splashes are the best matched features, and for the plane, the 3-D curves are the best matched features.

C. Composers

We obtained three range images from the three busts shown in Fig. 24 (a)–(c) with a liquid crystal range finder [22]. These three busts serve as models and are the content of the database. The scene is composed by overlaying three different views of the busts (the range finder's field of view and depth of field are too small to acquire such an amount of data at once). The data is smoothed with the adaptive smoothing algorithm [21], and small holes are closed with bilinear interpolation. The scene is displayed in Fig. 24(d). The algorithm finds the correct correspondences and the correct positions despite the fact that the three models are locally very similar (parts of the faces, parts of the clothes). The hypotheses generation step retrieves all possible candidates, and the verification step has to unravel the consistent clusters. The projection of the detected models on the scene is shown in Fig. 24(e). This example shows the limits of our algorithm with respect to speed. The detection is by far the slowest compared with all the other scenes (see the complexity discussion in Section III-E). This is mainly due to the similarity of the objects, the noisy data (the liquid crystal range finder provides fairly noisy data due to interreflections and coarse quantization), and the complexity of the scene. It is

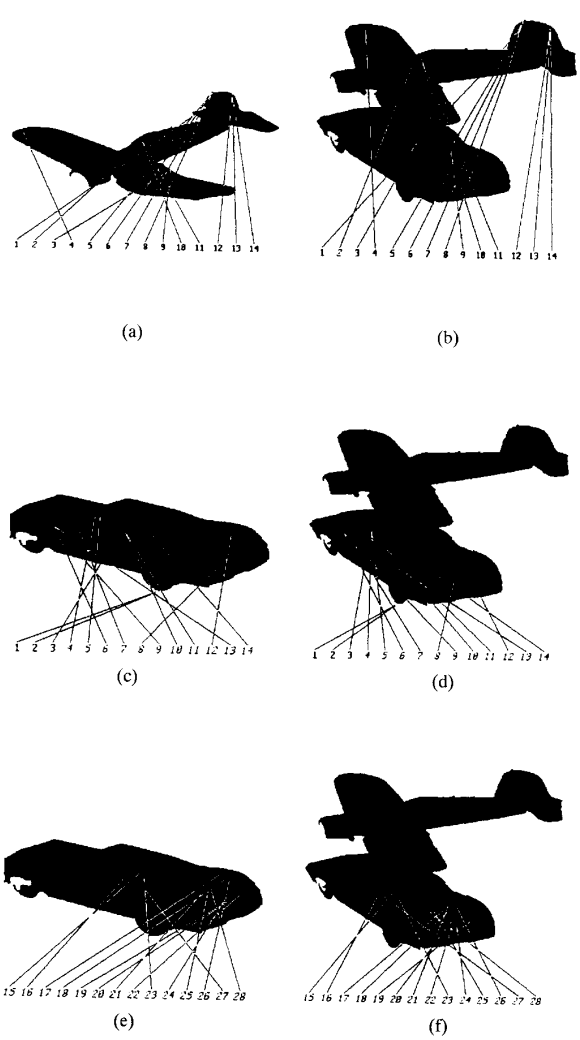


Fig. 23. Final hypotheses for scene 2 (splash hypotheses are printed in *italic*, and super segment hypotheses are printed in **bold**): (a) Plane on model; (b) plane on scene 2; (c) (1–14) for wagon on model; (d) (1–14) for wagon on scene 2; (e) (15–28) for wagon on model; (f) (15–28) for wagon on scene 2.

interesting to note that the algorithm found multiple solutions (e.g., an instance of the Chopin model was also found on Bach in the scene), but they were later rejected based on the rigidity assumption.

In Fig. 25, we show as an example the hypotheses that lead to the recognition of Chopin. Hypotheses 11 and 13 illustrate the location uncertainty very well.

D. Terrain Map

As a terrain map, we use the digital elevation model (DEM) covering the Martin Marietta ALV test area. A DEM is a 2-D array of uniformly spaced terrain elevation measurements. Our DEM map consists of 810 × 702 pixels with a pixel corresponding to a size of 5 × 5 m on the ground. For better visibility, we exaggerated the elevation data. The DEM map serves as the model in our database. We cut out a window of

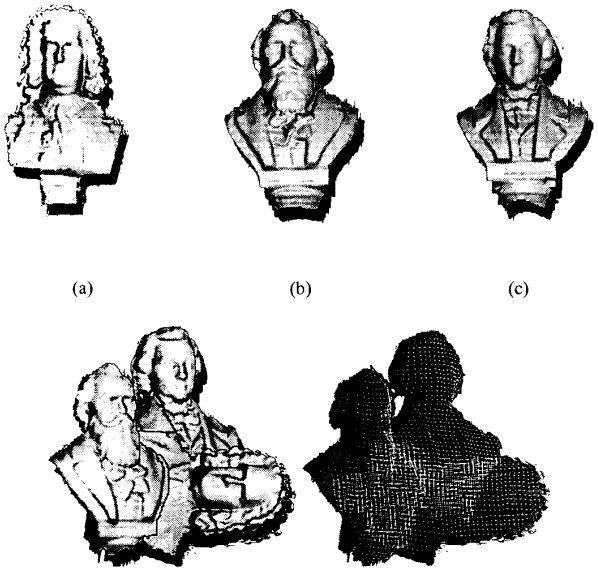


Fig. 24. Three composers: (a) Bach; (b) Brahms; (c) Chopin; (d) scene 3—three composers; (e) detected busts.

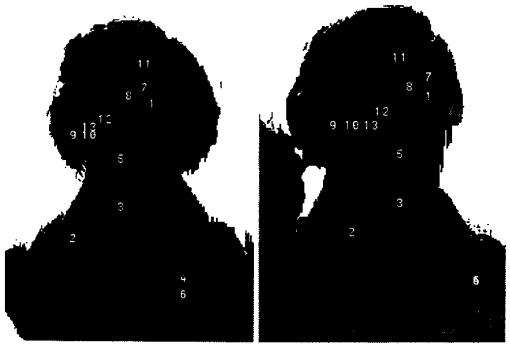


Fig. 25. Example of the hypotheses found for Chopin (hypotheses 4 and 6 are overlaid in the scene): (a) Hypotheses on Chopin; (b) hypotheses on scene 3.

80×80 pixels, rotated it by 80° around a tilted axis and used the resulting range information as the scene data. This kind of recognition can be useful in navigation or mapping tasks. Fig. 26 shows the shaded range image of the terrain map. The rectangle represents the area we picked for the window. Fig. 27(a) and (b) show the range and the shaded image of the window, which was rotated by 80° around a tilted axis. The marked arrow in Fig. 26 shows from which direction we look at this window to get the view in Fig. 27. The final hypotheses are displayed in Fig. 27(c) and (d). Fig. 28 shows the best match.

E. General Observations

We can give some rough numbers about the running time (on a serial Symbolics 3675 Lisp machine).

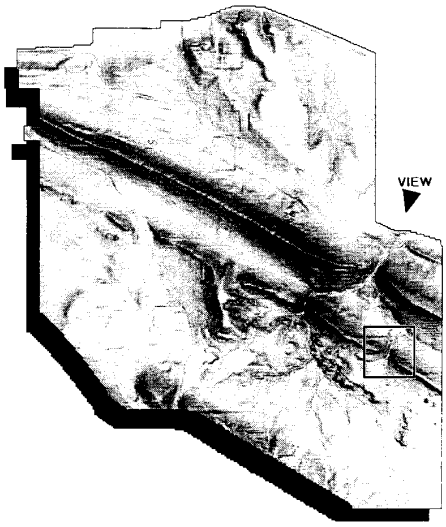


Fig. 26. Shaded DEM terrain map (= Model).

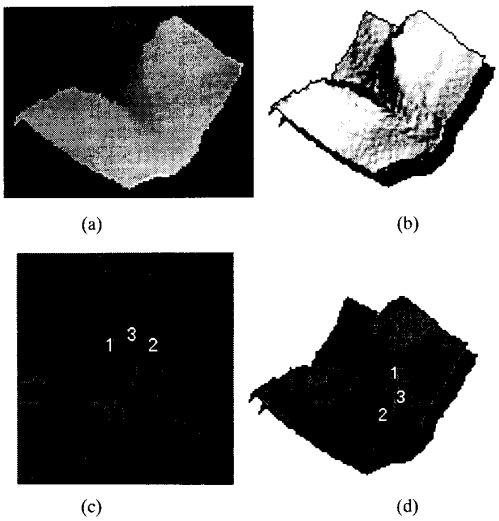


Fig. 27. Rotated DEM terrain window (= Scene) and final hypotheses: (a) Range image of window; (b) shaded image of window; (c) final hypotheses on closeup view of model; (d) final hypotheses on scene.

1) The acquisition of one super splash (consisting of, typically, three splashes with four line fitting tolerances) takes about 12 s.
2) The Mozart bust consists of about 400 super splashes; therefore, it took about 1 hr and 20 min to compute all splashes.
3) The plane consists of about 60 splashes; therefore, it took about 12 min to compute all splashes.
4) The computation of the 3-D curves takes less than 3 min.
5) The recognition process for the mozart scene takes about 50 s (retrieval: 10 s, verification: 40 s).
6) The recognition process for the plane and wagon scene

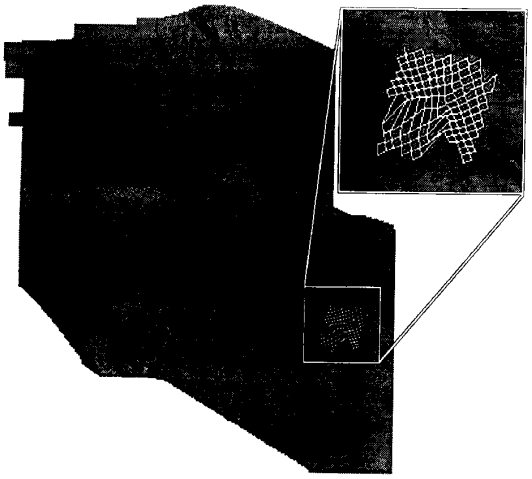


Fig. 28. Result of recognition (scene projected on model).

takes about 1 min and 30 s (retrieval: 15 s, verification: 75 s).

7) The recognition process for the bust scene takes about 30 min (retrieval: 2 min, rigidity filter: 7 min, verification: 21 min).
8) For the window of the DEM map, we compute 109 splashes (which takes approximately 20 min). The recognition itself succeeds after 35 s (retrieval: 5 s, verification: 30 s).

These numbers reflect neither the high parallelism, which is theoretically possible, nor the data redundancy with which we work at the moment. Simple improvements can significantly increase the performance. This is one goal of our future work.

V. Conclusion and Future Work

We showed with our implementation of the TOSS system that structural indexing provides a powerful mechanism for the recognition of general 3-D objects. We make very few restrictive assumptions about the shape of the objects, and we are able to acquire them automatically. By using two types of primitives, we overcome the problem of recognition in the case where either edge data or surface data does not provide enough information for a correct classification. Our encoding scheme allows us to match the primitives and verify the resulting hypotheses in a reasonable time complexity. We are able to handle large object databases. Our plan for the future is to further exploit the fact that *rich* features such as the splash or the super segment provide enough structural information to recognize objects efficiently. Our long-term goal is to build a recognition system that is able to recognize 3-D models in a 2-D gray-level image.

APPENDIX A
Least Squares Method

In this section, we introduce a least square method for finding the transformation matrix between two sets of 3-D points by minimizing the error. The correspondences are known. The method is based on the 2-D approach in [16] and [17]. The question is "What is the best match between the sequences of points $(u_j)_{j=1}^n$ and $(v_j)_{j=1}^n$?" The goal is to find the transformation $T(u) = Au + b$, which minimizes the distance between the sequences $(Tu_j)_{j=1}^n$ and $(v_j)_{j=1}^n$:

$$\delta = \min_T \sum_{j=1}^{n} | Tu_j - v_j|^2.$$

To later eliminate a second-order term, translate the set (u_j) so that

$$\sum_{j=1}^{n} u_j = 0.$$

Then

$$\begin{aligned}\delta &= \min_{A,b} \sum_{j=1}^{n} | Au_j + b - v_j|^2 \\ &= \min_{A,b} \Bigg(\sum_{j=1}^{n} | b - v_j|^2 + \sum_{j=1}^{n} | Au_j|^2 \\ &\qquad + 2\sum_{j=1}^{n} b \cdot Au_j - 2\sum_{j=1}^{n} Au_j \cdot v_j \Bigg).\end{aligned}$$

With

$$\sum_{j=1}^{n} b \cdot Au_j = b \cdot A\Big(\sum_{j=1}^{n} u_j\Big) = 0$$

TABLE II
TABLE OF OBJECTS

object (size in pixels)	# super splashes	# super segments	splash radii (pixels)	grid (pixels)	recognition time
car (387x239)	235	208	20,30,40	8	
dinosaur (141x305)	60	64	20,30,40	8	
mask (294x290)	350	276	20,30,40	8	
Mozart (264x399)	402	not computed	20,30,40	6	
phone (395x217)	262	193	20,30,40	8	
plane (507x218)	60	239	20,30,40	8	
pyramid (105x97)	27	not computed	20,30,40	5	
Renault part (315x367)	124	204	20,30,40	8	
wagon (422x178)	367	256	20,30,40	6	
Bach (230x389)	506	not computed	15,20,25	6	
Brahms (232x351)	640	not computed	15,20,25	5	
Chopin (232x359)	469	not computed	15,20,25	6	
scene 1 (240x390)	329	not computed	20,30,40	8	50 s
scene 2 (458x324)	293	350	20,30,40	8	90 s
scene 3 (446x424)	849	not computed	15,20,25	7	30 min
DEM window (80x80)	109	not computed	15,20,25	3	35 s
DEM map (811x702)	5544	not computed	15,20,25	5	

we can eliminate one second-order term, and therefore, we can minimize $\sum_{j=1}^{n} \mid b - v_j \mid^2$ and $\sum_{j=1}^{n} \mid Au_j \mid^2 - 2\sum_{j=1}^{n} Au_j \cdot v_j$ separately. The solution for b is simply

$$b = \frac{1}{n}\sum_{j=1}^{n} v_j.$$

To minimize over A, we set

$$g(A) = g(a_{11}, a_{12}, \ldots a_{33}) = \sum_{j=1}^{n} \mid Au_j \mid^2 - 2\sum_{j=1}^{n} Au_j \cdot v_j.$$

We have to find

$$\min_A g(A) = \min_A g(a_{11}, a_{12}, \ldots a_{33})$$

by solving the following system of nine equations ($i = 1, 2, 3; j = 1, 2, 3$):

$$\frac{\partial g}{\partial a_{ij}} = 0 \tag{1}$$

Since g is a quadratic function in each of its unknowns, (1) is a system of nine linear equations with nine unknowns; the nine equations are three independent sets of equations with three unknowns. The computation of the unknowns (the transformation matrix T) is straightforward.

The determinant of the rotational part A of the transformation matrix provides us with a measurement for the volume change imposed by the transformation. The transformation is rigid if $det(A) = 1$. When we examine a hypothesis on rigidity, we have to consider the effect of noise (see Section III-D-3). Therefore, in practice, we validate the rigidity assumption only if $0.5 < det(A) < 2.0$.

APPENDIX B
CONSISTENCY BETWEEN HYPOTHESES

We simplify the presentation by regarding a hypothesis in this section as a match between two splashes, whereas we actually also have to consider matches between 3-D super segments. The extension to a hypothesis based on 3-D super segments is straightforward and is addressed at the end of every subsection. It is important to note that with a match of two features, we get a set of corresponding points:

- A hypothesis of two splashes of cardinality four provides us with five corresponding point pairs. Four are in the vicinity of the patch (along the sample normals). They correspond to the vertices of the polygonal approximation in the (θ, ϕ, ψ) space. The fifth pair corresponds to the centers of the two splashes (the locations).
- A hypothesis of two 3-D super segments of cardinality four provides us with five corresponding point pairs: one for each vertex.

Suppose we have two hypotheses h_1 and h_2; every h_i consists of a match between a splash on the model m_i and a splash on the scene s_i. The reference normal of a splash s is n_s. Now, we address the question "When are the two hypotheses h_1 and h_2 mutually consistent?" We basically have to reduce the five degrees of freedom (dof's) that describe the geometrical relationship between our two features to completely constrain the attitude of one feature relative to another. We have to deal with three dof's for the relative position and two dof's for the relative orientation.

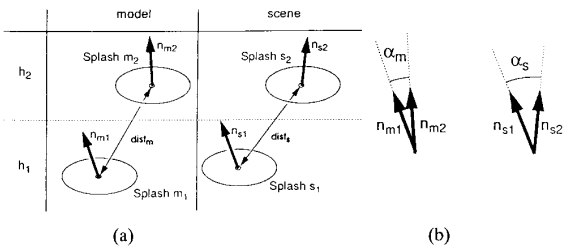


Fig. 29. (a) Distance constraint; (b) orientation constraint.

1) The Distance Constraint

- The distance between two splashes is the distance between their locations. To be consistent, the two splashes on the model should have approximately the same distance as the splashes on the scene (since it is a rigid transformation) as shown in Fig. 29(a).
- Therefore, we require

$$\| dist_m - dist_s \| < \varepsilon_d.$$

- With this constraint, we reduce the dof's for the position to two.
- For 3-D super segments, we use the middle vertex (or, in case of even cardinality, the middle point of the middle segment) as the feature position.
- In our implementation, we use $\varepsilon_d = 0.1 \cdot \max(dist_m, dist_s)$.

2) The Orientation Constraint

- For a splash, we use the reference normal as the orientation vector. The angle α_m, which is formed by the orientation vectors of the two splashes on the model, should be approximately the same as the angle α_s, which is formed by the orientation vectors of the two splashes on the scene (Fig. 29(b)).
- Therefore, we require

$$\| \alpha_m - \alpha_s \| < \varepsilon_\alpha.$$

- This reduces the orientation dof's to one.
- We use the vector from the last to the first vector as the orientation vector for a 3-D super segment.
- In our implementation, we use $\varepsilon_\alpha = 25°$.

3) The Direction Constraint

- For computing the direction angles γ_{1i} and γ_{2i} for splash s_2 in relation to s_1, we have to assign a unique frame F to the splash s_1, as shown in Fig. 30 (this frame should not be confused with the frame we defined in Section III-B-1).
  a. The frame F has its origin at the location of the splash.
  b. The z axis is the reference normal.

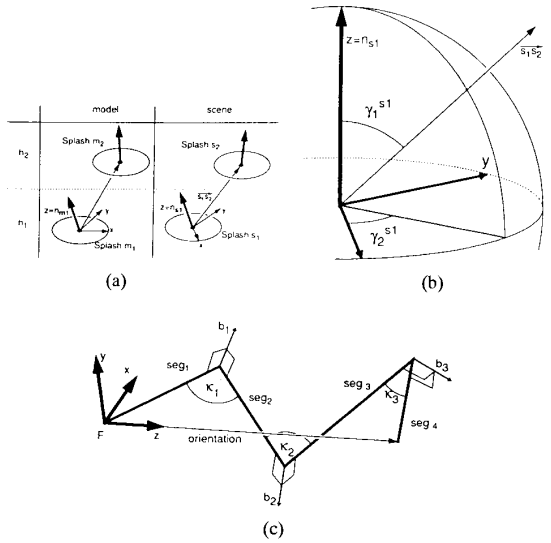


Fig. 30. Orientation constraint: (a) Direction constraint; (b) closeup of the frame F in splash s_1; (c) assigning a frame to a 3-D super segment.

c. In Section III-B-2, we use the point with the strongest tangent tilt as the start of the polygonal approximation. This point p is the point where the sample normals have the strongest tilt with respect to the reference normal, and we know that p lies on the circular slice of the splash. We know further that p has a corresponding point on the matched splash. Without loss of generality, we use p as a reference point for defining the x axis.

d. The x axis is defined as the vector that is perpendicular to z and lies in the plane defined by z and the point p.

e. The y axis is perpendicular to x and z (and x, y, z form a right-handed coordinate system).

Representing the vector $\overrightarrow{s_1s_2}$ in spherical coordinates in frame F results in an angle pair $(\gamma_1^{s_1}, \gamma_2^{s_1})$ (see Fig. 30(b)). To preserve the rigidity assumption, the values for $(\gamma_1^{s_1}, \gamma_2^{s_1})$ must be approximately the same as for $(\gamma_1^{m_1}, \gamma_2^{m_1})$. This reduces the dof's for the position by two to zero. In addition, requiring similar values for $(\gamma_1^{s_2}, \gamma_2^{s_2})$ and $(\gamma_1^{m_2}, \gamma_2^{m_2})$ reduces the dof's for the orientation to zero.

- Therefore we require, for $i, j \in \{1, 2\}$

$$\|\gamma_i^{s_j} - \gamma_i^{m_j}\| < \varepsilon_\gamma$$

- For 3-D super segments, we assign a frame F in the following way (see Fig. 30(c)):

a. The frame F has its origin at the first vertex.

b. The z axis is the (normalized) orientation vector.

c. The x axis is perpendicular to z and lies in the same plane as the first segment. If the first segment and the orientation vector are aligned, take the next segment.

d. The y axis is perpendicular to x and z in a right-handed coordinate system.

- In our implementation, we use $\varepsilon_\gamma = 25°$.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their suggestions and helpful remarks for improving this paper.

REFERENCES

[1] P. J. Besl, "The Free-Form Surface Matching Problem," in *Machine Vision for Three Dimensional Scenes* (H. Freeman, Ed.). New York: Academic, 1990, pp. 25–71.

[2] B. Bhanu, "Representation and shape matching of 3-D objects," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 6, no. 3, pp. 340–350, 1984.

[3] R. C. Bolles and P. Horaud, "DPO: A three-dimensional part orientation system," *Int. J. Robotics Res.*, vol. 5no. 3, pp. 3–26, 1986.

[4] T. M. Breuel, "Adaptive model base indexing," in *Proc. DARPA Image Understanding Workshop*, 1989, pp. 805–814.

[5] C. H. Chen and A. C. Kak, "A robot vision system for recognizing 3-D objects in low-order polynomial time," *IEEE Trans. Syst. Man Cybern.*, vol. 19, no. 6, pp. 1535–1563, 1989.

[6] T. J. Fan, *Describing and Recognizing 3-D Objects Using Surface Properties*. New York: Springer Verlag, 1990.

[7] T. J. Fan, G. Medioni, and R. Nevatia, "Recognizing 3-D objects using surface descriptions," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 11, no. 11, pp. 1140–1157, 1989.

[8] O. D. Faugeras and M. Hebert, "The representation, recognition, and locating of 3-D objects," *Int. J. Robotics Res.*, vol. 5, no. 3, pp. 27–52, 1986.

[9] P. J. Flynn and A. K. Jain, "On reliable curvature estimation," in *Proc. IEEE Comput. Vision Patt. Recogn.* (San Diego, CA), June 1989, pp. 110–116.

[10] W. E. L. Grimson, "The combinatorics of object recognition in cluttered environments using constrained search," in *Proc. IEEE Int. Conf. Comput. Vision* (Tampa, FL), Dec. 1988, pp. 218–227.

[11] W. E. L. Grimson and T. Lozano-Pérez, "Model-based recognition and localization from sparse range or tactile data," *Int. J. Robotics Res.*, vol. 3, no. 3, pp. 3–35, 1984.

[12] ——, "Localizing overlapping parts by searching the interpretation tree," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 9, no. 4, pp. 469–482, 1987.

[13] K. Ikeuchi, "Precompiling a geometrical model into an interpretation for object recognition in bin-picking tasks," in *Proc. DARPA Image Understanding Workshop* (Los Angeles, CA), Feb. 1987, pp. 321–339.

[14] A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir, "Two-dimensional, model-based, boundary matching using footprints," *Int. J. Robotics Res.*, vol. 5, no. 4, pp. 38–55, 1986.

[15] E. Kishon and T. Hastie, "3-D Curve Matching Using Splines," in *Proc. European Conf. Comput. Vision* (Antibes, France), Apr. 1990, pp. 589–591.

[16] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "On recognition of 3-D objects from 2-D images," in *Proc. IEEE Int. Conf. Robotics Automat.*, Apr. 1988.

[17] Y. Lamdan and H. J. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," in *Proc. IEEE Int. Conf. Comput. Vision* (Tampa, FL), Dec. 1988, pp. 218–249.

[18] P. Horaud and R. C. Bolles, "3DPO's strategy for matching three-dimensional objects in range data," in *Proc. Int. Conf. Robotics* (Atlanta, GA), Mar. 1984, pp. 78–85.

[19] B. Parvin and G. Medioni, "A constraint satisfaction network for matching 3d objects," in *Proc. Int. Conf. Neural Networks* (Washington, DC), June 1989, pp. 281–286, volume II.

[20] G. M. Radack and N. I. Badler, "Local matching of surfaces using a boundary-centered radial decomposition," *J. Comput. Vision Graphics Image Processing*, vol. 45, pp. 380–396, 1989.

[21] P. Saint-Marc, J. - S. Chen, and G. Medioni, "Adaptive smoothing: A general tool for early vision," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 13, no. 6, pp. 514–530, June 1991.

[22] K. Sato and S. Inokuchi, "Range-imaging system utilizing nematic liquid crystal mask," in *Proc. IEEE Int. Conf. Comput. Vision*, June 1987, pp. 657–661.

[23] F. Stein and G. Medioni, "Efficient two dimensional object recognition," in *Proc. Int. Conf. Patt. Recogn.* (Atlantic City, NJ), June 1990.

[24] ——, "TOSS—A system for efficient three dimensional object recognition, in *Proc. DARPA Image Understanding Workshop* (Pittsburgh, PA), Sept. 1990.
[25] J. S. Vitter and W. -C. Chen, *Design and Analysis of Coalesced Hashing*. Oxford: Oxford University Press, 1987.

Fridtjof J. Stein (S'90) was born in Stuttgart, Germany, in 1962. He received the Vordiplom in 1984 and the Diplom in informatics in 1988 from the University of Karlsruhe, Germany. He is currently working towards the Ph.D. degree in computer science at the University of Southern California, Los Angeles.

He is a Research Assistant with the Institute for Robotics and Intelligent Systems, and his current research interests include object recognition, robot navigation, and machine learning.

Gérard Medioni (M'83) received the Diplôme d'Ingénieur Civil from the Ecole Supérieure des Télécommunications, Paris, France, in 1977 and the M. S. and Ph. D. degrees in computer science from the University of Southern California (USC), Los Angeles, in 1980 and 1983, respectively.

Since then, he has been an Assisstant Professor of Electrical Engineering and Computer Science at USC. He is the author of many articles in the field of computer vision.

Dr. Medioni has served as Program co-chair of the 1991 IEEE Conference on Computer Vision and Pattern Recognition in Hawaii. He is a member of the American Association for the Advancement of Science and the Association for Computing Machinery.