COMPUTER VISION AND IMAGE UNDERSTANDING
Vol. 63, No. 2, March, pp. 232–250, 1996
ARTICLE NO. 0017

# Surface Correspondence and Motion Computation from a Pair of Range Images

BIKASH SABATA

*SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025*

AND

J. K. AGGARWAL

*Electrical and Computer Engineering Department, The University of Texas at Austin, Austin, Texas 78712*

Received June 4, 1995; accepted February 8, 1995

**The estimation of the motion transformation of a moving object from a sequence of images is of prime interest in computer vision. In this paper, the issues in estimating the motion parameters from a pair of range images are addressed. The motion estimation task, in the domain of range image sequences, has two components: (1) extract the surfaces and establish the correspondence of the surfaces over the frames in the sequence of range images, and (2) compute the motion transformation using these surface correspondences. A novel procedure based on a hypergraph representation is presented for finding surface correspondence. Two scenes are modeled as hypergraphs and the hyperedges are matched using a subgraph isomorphism algorithm. The hierarchical representation of hypergraphs not only reduces the search space significantly but also facilitates the encoding of the topological and geometrical information used to direct the search procedure. Results obtained from real range image pairs show that the algorithm is robust and performs well in presence of occlusions and incorrect segmentations. Motion transformation between image frames is computed using the planar and the quadric surface pairings. A least-squares minimization procedure is formulated that estimates the best motion transform, subject to the constraints of rigid motion. For the case of linear feature pairings, the motion computation becomes tractable because the rotation and the translation computations become independent of each other. However, for quadric surfaces this is not true. The equation to be minimized is highly nonlinear and the uniqueness of solution cannot be guaranteed. The solution obtained computes the motion by extracting unique linear features from the quadric surfaces and using them to compute the motion transformation. The main contribution of the work is a surface-based framework for motion estimation from a sequence of range images. The primary issues of correspondence and motion computation are formulated and solved in terms of the surface descriptions.** © 1996 Academic Press, Inc.

## 1. INTRODUCTION

The ability to discern the motion of objects is integral to computer vision as well as human vision. The relative motion between the camera and the objects in a scene gives rise to the apparent motion of the objects in a sequence of images. By an image, one means a function, $I(x, y)$, of the pixel locations $(x, y)$. The function values may represent 3-D coordinates, depth, intensity, or any other modality. The most widely studied modality for motion estimation is intensity, i.e., using a sequence of 2-D intensity images [1, 18, 25–27] to estimate motion. With new developments in 3-D range image acquisition techniques, it is now realistic to address the problem of motion estimation from a sequence of 3-D range images. The objective of this research is to focus on the important computer vision task of estimating the motion in the domain of 3-D range image sequences.

The use of 3-D range sensing allows for the direct extraction of depth information, i.e., the distance from the sensor to the objects. Compared to the use of sequence of 2-D images, this additional depth information greatly reduces the complexity of the motion estimation task. The detection of motion from a sequence of range images can be subdivided into three stages: (1) segmentation, (2) correspondence of features between frames, and (3) computation of motion using the feature correspondences. The properties of the features to be used directly affect the stability, reliability, and robustness with respect to noise and distortions in input data. Features can be classified into *local* and *global* features. Local features, such as corners and edges, are highly sensitive to noise and can easily be completely occluded. Global features, such as surfaces, are less sensitive to noise and are more likely to be only partially occluded. These observations prompt one to develop procedures using global features that are less sensitive to noise for the motion estimation task. But, using higher order features such as surfaces does not immediately solve the problem. The transformation equations now become nonlinear, and the issues of stability and convergence have to be addressed. There appears to be an interesting trade-off between the reliability of feature detection and the complexity of the motion estimation task. In this paper, surfaces are the features to be used in all the stages.

Segmentation of the input range image into homogeneous regions extracts the important features in the images. The segmentation task uses the procedure developed in [23]. The segmentation algorithm gives as the output homogeneous surface segments along with the surface parameters of each region.

Matching of object surfaces is the key to motion estimation and tracking an object over a sequence of images. A novel procedure for establishing correspondence is developed. The solution uses geometrical and topological information derived from the scenes to direct the search procedure. The performance of the matching depends greatly on the results of the segmentation algorithms. Incorrect segmentation causes poor estimation of the surface parameters and affects the performance of the matching algorithm. Occlusion of features and the appearance of new features further complicate the problem. These issues are addressed and a solution is obtained that is robust and able to handle occlusions of surfaces, noise in data, and incorrect segmentation from a segmentation algorithm. In the present implementation, it is assumed that the images have planar or quadric surfaces (which includes cylindrical and conical surfaces); however, the procedure is general enough to be extended to other surface classes.

Although the question of finding correspondences between features has been studied extensively [4, 8, 13–15], most of these approaches deal with matching a scene to a model of the object. The fundamental difference between model-to-scene matching and scene-to-scene matching is that in the former, the model description of the object is complete, and to that the incomplete description of the object obtained from the scene is matched. However, in the case of scene-to-scene matching, both object descriptions are incomplete and a match between two incomplete descriptions is obtained. By incomplete, it is meant that all the features are not present in the object description because of occlusions and sensor errors. This difference makes it impossible to use the strategies obtained for object recognition in the domain of object tracking. New strategies based on the constaints of the problem have to be designed.

The features extracted from the scene can be represented as an attributed graph, where the vertices of the graph are the features and the arcs are the relationships between the features. If the scene is represented as a graph, then the model-to-data matching of the features is the subgraph isomorphism problem in graph theory. Numerous algorithms have been proposed for computing the isomorphism between two graphs [21, 11], and it has been shown that the problem is *np*-complete and there is no polynomial time algorithm for the general case. However, if constraints are incorporated into the search procedure, then a good average case performance can be expected [20]. Grimson and Lozano-Perez [12, 14] incorporate a set of unary and binary constraints into the search procedure using a trees search strategy. The constraints prune the tree and reduce the search space.

In a related problem, Arman [3, 5] addresses the problem of locating an object in a scene. The approach adopted to identify the object from the input data is to form a search strategy by precompiling the object models in the database. The sequence of search steps is determined by measuring the goodness of the features of the model in terms of uniqueness, detectability, reliability, and computational cost. The precompiled sequence of model features has the effect of ordering the features such that the improbable pairings are rejected in an early stage. This results in pruning the search tree, since not all the branches of the interpretation tree are investigated for possible matches.

Fundamental to our strategy to match features over a sequence of range images is a *hypergraph representation* of the scenes. Wong *et al.* [29] first introduced the hypergraph representation to solve correspondence between the input data and the object models. Hypergraphs are generalizations of graphs where a *hyperedge* is the generalization of the edge in a graph. The hyperedge is a collection of vertices (in the graph, the edge is a set of two vertices) and the hyperedges and the vertices together from the hypergraph. The two scenes are modeled as hypergraphs, and the hyperedges are matched using a subgraph isomorphism algorithm. To reduce the complexity of the matching task, heuristics derived from the topological and geometrical information available from the scene are used to direct the search. The hierarchical representation of hypergraphs not only reduces the search space significantly, but also facilitates the encoding of the topological and geometrical information. Hyperedges are formed by grouping the surfaces. Using *a priori* knowledge arising out of the physical constraints of laser scanning, a fast matching algorithm is designed.

The *computation of the motion transformation parameters* from the established surface correspondences is addressed next. The motion transformation is not a general affine transformation but a constrained affine transformation where all the constraints of rigidity have to be satisfied by the computed solution. The constraints are nonlinear; therefore, the solution to the problem is not simple. Since, in real situations, noise and camera distortions cannot be ignored, the motion parameter computation is posed as a least-squares minimization problem, the solution yielding the best motion estimate in the least-squares sense. To make the motion estimation system reliable and robust, it is important to consider the issues of stability and convergence of the algorithms. Also, the uniqueness of the obtained solutions has to be assured.

Earlier work on motion computation from 3-D data assumed linear features in the data (planes, lines and points) [16, 19, 2, 6, 10, 22]. The methods cannot be extended to higher order features such as quadric surfaces because of the highly nonlinear nature of the problem. We address this issue and propose a solution method which reduces to the case of linear features. However, our method does not require the explicit detection of the linear features in the data, which is error prone.

The contribution of the presented work is twofold. A surface-based framework for motion estimation from a sequence of 3-D range data is developed that solves the two problems of correspondence and motion computation. More specifically, a solution to the problem of correspondence between surface patches over different frames is developed. Following correspondence, the task of motion estimation requires the computation of the motion transformation. Stable and convergent algorithms are developed which compute the motion.

## 2. PROBLEM FORMULATION

The goal of this research is to estimate the relative motion of the camera given a range image of a scene at two different time instances. By range, it is meant that the 3-D coordinates of each point sensed in the scene are known. The range may be obtained by using either a stereo algorithm or active ranging devices, such as lasers, sonar, or ultrasound. The three fundamental tasks in motion estimation are: (1) segmentation of the range image and feature extraction; (2) establishing correspondence between features over different frames; and (3) computing the motion transformation using the feature correspondences.

The range image $I(u, v)$, is defined as a vector function of the image coordinates $(u, v)$, where at each pixel location $(u, v)$ the range image gives a position vector of a point in the scene. Segmentation of the scene involves the partitioning of the range image into regions $\{R_i(u, v)\}$. Following the segmentation, each region represents a surface segment in the scene. The parameters of the surfaces are extracted to generate the set of parameters $\{\mathbf{P}_i\}$.

Each range image of the sequence is segmented and features are extracted, and a surface parameter set is obtained for each frame. Next, we pair the surface features in the two frames such that they correspond to each other in the scene. Let the feature be represented by the parameter vector $\mathbf{P}$ and let its corresponding parameter vector in the second frame be $\mathbf{P}'$. The ordered pair $(\mathbf{P}, \mathbf{P}')$ represents a pair of corresponding features in the two frames. A set $\mathbf{S} = \{(\mathbf{P}, \mathbf{P}')\}$ of all such ordered pairs is the output of the second task. Occlusions and errors in segmentation cause features to be hidden or missing in some frames, so the corresponding feature in the other frame cannot be paired; therefore the *null* element (represented by a $\bigstar$) is introduced into each set of surface parameters $\{\mathbf{P}_i\}$. The *null* element can be paired with any element. The function that computes the correspondence forms the set $\mathbf{S} = \{(\mathbf{P}, \mathbf{P}')_j\}$, such that $\mathbf{P} \in \{\mathbf{P}_i\} \cup \bigstar$ and $\mathbf{P}' \in \{\mathbf{P}'_i\} \cup \bigstar$. The matching task is restated as:

Given the set of surface parameters $\{\mathbf{P}_i\}$ and $\{\mathbf{P}'_i\}$ corresponding to the two frames of the range image sequence, compute the set of ordered pairings $\mathbf{S} = \{(\mathbf{P}, \mathbf{P}')\}$ such that $\mathbf{P} \in \{\{\mathbf{P}_i\} \cup \bigstar\}$ and $\mathbf{P}' \in \{\{\mathbf{P}'_i\} \cup \bigstar\}$, where $\bigstar$ is the *null* element that can match any element.

Motion computation *estimates the motion of the object, given the set* $\mathbf{S}$ *of feature correspondences.* If the transform parameters are represented as $\mathscr{T}$, then, in general,

$$\mathbf{P}' = \mathscr{F}(\mathbf{P}, \mathscr{T}), \tag{1}$$

where $\mathscr{F}$ is some function over $\mathbf{P}$ and $\mathscr{T}$. The nature of the function $\mathscr{F}$ depends on the features, their representation, and the representation of the transformation parameters $\mathscr{T}$. Since the object is rigid, the same transformation $\mathscr{T}$ holds for each ordered pair $(\mathbf{P}, \mathbf{P}')$ belonging to the object.

Let $\mathbf{n}$ represent the noise and the estimation error in $\mathbf{P}$ and let $\mathbf{n}'$ represent the same in $\mathbf{P}'$. The errors occur due to the noise in the data acquisition process and roundoff errors of the data. The error in estimating the final position of the feature after motion can be written as

$$E_{\text{est}}(\mathscr{T}) = \sum_{(\mathbf{P},\mathbf{P}')\in\mathbf{S}} \|\tilde{\mathbf{P}}' - \mathscr{F}(\tilde{\mathbf{P}}, \mathscr{T})\|^2, \tag{2.a}$$

where

$$\tilde{\mathbf{P}}' = \mathbf{P}' + \mathbf{n}' \quad \text{and} \quad \tilde{\mathbf{P}} = \mathbf{P} + \mathbf{n}. \tag{2.b}$$

The objective is to minimize the estimation error $\mathbf{E}_{\text{est}}(\mathscr{T})$. Hence, motion computation now becomes:

Given the set $\mathbf{S}$ of all detected feature correspondences, compute an optimal estimate of the motion transformation such that the error $E_{\text{est}}$ is minimized.

## 3. HYPERGRAPH REPRESENTATION

Representation of the available information in a suitable form is the key to the solution of matching. The output of the segmentation module is a low-level description of the geometrical and the topological information. Each pixel in the range image is assigned a label, and to each label a surface parameter vector is associated. Such representation is not conducive to reasoning about the topology of the scene. For example, a simple predicate such as "Is the region corresponding to label A neighbor to the region correspnding to the label B?" requires a significant amount of computation to be carried out before the query can be answered. To answer the query, first, a pixel with the label A has to be located in the image; then, the boundary of the region with the label A must be detected. The label of the neighbor pixels at the boundary are compared with B while the boundary of the region is tracked. Obviously, a representation that facilitates reasoning about the geometry and the topology of the scene is required.

The attributes computed by the low-level processing module are represented using a hypergraph data structure. The representation encodes the topological and geometrical information extracted from the range image of the scene. Hypergraphs are generalizations of graphs. The arc (or edge) is generalized as a hyperedge, where a set of vertices forms the hyperedge, instead of just two vertices forming the arc. The group of vertices forming the hyperedge may share some common property. Although hypergraphs have been used in vision and robotics applications [28, 29], a new definition of the hyperedge and a novel method for constructing the hypergraphs that makes them a powerful tool for vision applications is presented here.

Attributed hypergraphs are a concise way of representing objects such that both quantitative and qualitative information are encoded in the representation. The formal definition is as follows:

DEFINITION 3.1. The Hypergraph [7] is defined as an ordered pair $H = (X, E)$, where $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of attributed vertices and $E = \{e_1, e_2, \ldots, e_m\}$ are the hyperedges of the hypergraph. The set $E$ is a family of subsets of $X$ (i.e., each $e_i$ is a subset of $X$) such that

1. $e_i \neq \emptyset, i = 1, \ldots, m$
2. $\bigcup_{i=1}^{m} e_i = X$.

A graph is a hypergraph whose hyperedges have cardinality of two.

To arrive at the hypergraph representation, the scene is first represented as an attributed graph. Each surface patch in the range image forms an attributed vertex. The attribute values are the surface property values. For each pair of surfaces that are connected, an attributed arc is formed. The attributes of the arc describe the interfacing edge and the relative geometrical information between the two surfaces. Groups of the attributed vertices (surface patches) form a hyperedge, and with each hyperedge we associate an attributed graph that describes the topology of the component attributed vertices (surface patches).

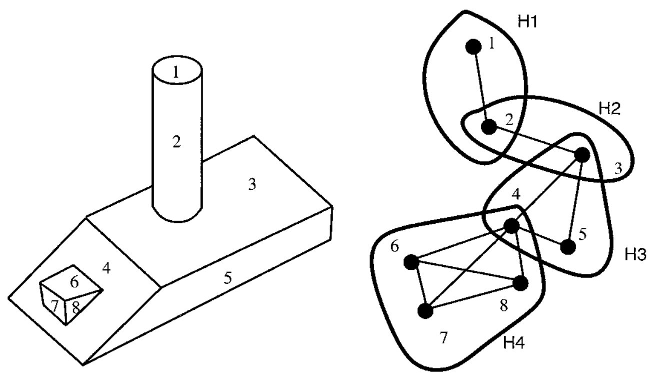


FIG. 1. An object and its corresponding hypergraph representation.

## 3.1. *Hyperedge Construction*

The set of vertices that form the hyperedge should represent a topologically significant feature in the graph so that the matching task is guided by the topology of the scene. Cliques in the graph are significant features that are rich in information.

DEFINITION 3.2. A clique of a graph $G$ is a maximal complete subgraph of $G$, i.e., a complete subgraph of $G$ not properly contained in any other subgraph of $G$.

Physically, the cliques represent groups of surfaces that are adjacent to each other. Since a clique provides a larger attribute set and many geometrical properties, the probability of a false positive match (between two cliques) is reduced significantly. Each clique forms a hyperedge in the hypergraph and the attributed graph describing the clique is the associated attribute of the hyperedge. Figure 1 illustrates the formation of a hypergraph from a scene.

The complexity of computing the cliques in a graph is exponential, so the advantage gained by matching using the hypergraph would be lost because of the cost of constructing the hypergraph. However, the physics of the range imaging process restricts the size of the cliques in the scenes that are observed. The laser scanning process results in a depth map which describes the depth $z$ in terms of the image plane coordinates $(u, v)$. The depth map description is a geographic map of the depth values on a plane, and for a planar map it has been proved that four colors are sufficient to color the map [9]. What the result signifies is that there cannot exist more than four regions that touch each other. If the map is represented as a graph, where the vertices are the regions and an arc exits between two vertices corresponding to adjacent regions, then there cannot exist a clique with more than four vertices in the graph. This result is stated as a lemma:

LEMMA 3.1. *For a range image described as a depth map, $z = f(u, v)$, there cannot exist a group of more than four surface segments that are adjacent to each other.*

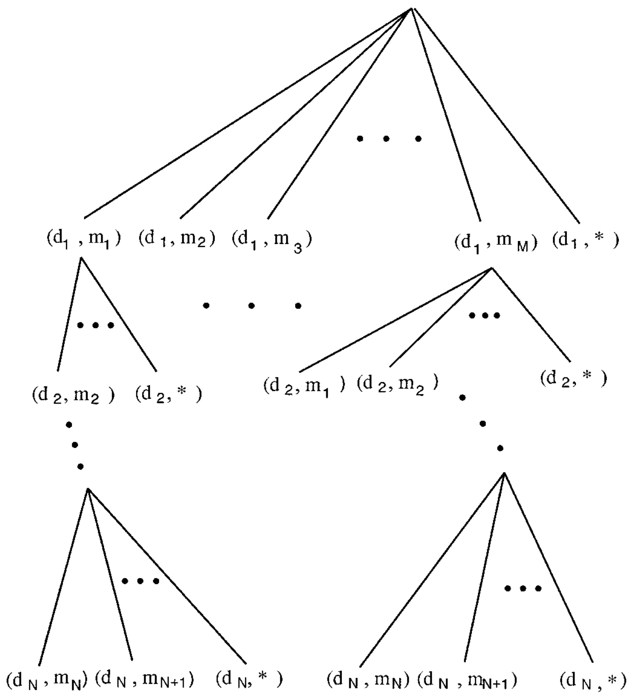
Once the upper bound on the size of the cliques is known, the complexity of computing the cliques becomes $O(n^3)$. (A tighter bound with a lower polynomial order can be obtained.) The strategy of the algorithm to detect the cliques is to consider one vertex at a time and find all the cliques that the vertex can form with its neighbors. Once all the cliques have been found, the vertex is removed from the graph and not considered again. Since the cardinality of the cliques can only be two, three, or four, the algorithm first creates all the groups of two vertices, which includes the current vertex. From the set of groups of two vertices, groups of three vertices are formed. The groups of four vertices are then formed from the group of three vertices. These groups are the cliques of the graph with the current vertex as a member of each clique.

The cliques are the hyperedges of the hypergraph representing the scene. Each clique forms a hyperedge and the attributed graph associated with the clique is associated with the corresponding hyperedge. The hyperedges in the scene hypergraph represent the regions of "high activity" in the scene. These regions are rich in geometrical information; therefore, the probability of making a false positive match between hyperedges is reduced significantly. The topological information is also available implicitly because of the nature of the representation. The topological information "steers" the matching algorithm in the proper direction so that the solution is obtained in the minimum number of steps.

## 4. MATCHING PROCEDURE

This section presents the matching procedure used to derive the surface correspondences in a sequence of range images. If the scene is represented as a graph, then matching of features, in general, is equivalent to subgraph isomorphism, where the first graph is matched to a subgraph of the second. However, scene-to-scene matching differs from this in that the first graph is not entirely a subgraph of the second graph, but a subgraph of the first graph matches a subgraph of the second graph. This additional variation makes the problem of scene-to-scene matching more complicated than the scene-to-model matching.

The matching procedure uses the information encoded in the hypergraph representation to result in the final pairings. The heart of the matching procedure is a *directed tree search* algorithm that tests various hypotheses and rejects the impossible ones. The interpretation that gives the largest match is selected as the solution. Constrained tree search algorithms have been used in many applications [14, 15, 20]. (Most of the constraint search algorithms are equivalent to the tree search algorithm [20].) Data pairings are formed by a depth-first search of an *interpretation tree* (Fig. 2). Each node of the tree represents a possible pairing. The first data (surface patch) is taken from the first scene and paired with each of the data in the second scene. These form the nodes in the first level of the tree. To account for missing surface segments due to occlusions, the data is also paired with a *wild card* ★. Subsequent levels of the tree correspond to pairings of other vertices. Each branch of the tree represents a partial matching of the scenes. The constraints are used to prune the search tree and thus reduce the search space.



$(d_1, m_1)$ $(d_1, m_2)$ $(d_1, m_3)$ $(d_1, m_M)$ $(d_1, *)$
$(d_2, m_2)$ $(d_2, *)$ $(d_2, m_1)$ $(d_2, m_2)$ $(d_2, *)$
$(d_N, m_N)$ $(d_N, m_{N+1})$ $(d_N, *)$ $(d_N, m_N)$ $(d_N, m_{N+1})$ $(d_N, *)$

FIG. 2. The interpretation tree used in the matching procedure. The tree has $N$ levels corresponding to the $N$ features in the first frame. Each feature in the first frame is matched to features in the second frame and the *null* feature.

A variation of the constrained tree search is presented, in which the search is directed based on the current hypothesis. The directed search, coupled with the termination conditions, reduces the search space. The key idea is to use the topological constraints of the scene to determine the next most likely match and to accept or reject the matches based on the geometrical constraints.

The features used in the matching process are surface segments. The segmentation module segments the range image into surface segments, and the surface parameters of each region are computed. The interfacing edges between the surface segments are detected and their properties are computed. The properties of the edge segments used are: (1) the edge type (straight line or curved), (2) the edge length, and (3) the depth discontinuity. The depth discontinuity across the edge implies that one surface may be occluding (partially or completely) another surface. The information about occlusion is incorporated in the attribute list of the surface patches.

The constraints used are similar to the unary and binary constraints developed by Grimson and Lozano-Perez [12]. The only unary constraint used is the surface type classification (planar, cylindrical, conical, etc.). Other properties used in model-based object recognition, such as area, perimeter, and compactness, are very sensitive to occlusion, and since occlusion may occur in either of the range images, these properties cannot be used as constraints. The binary constraints describe the relative properties between pairs of surface segments. The properties used are: (1) connectivity, (2) the angle between the surface patches, (3) the range of distances between the two surface patches, (4) the range of the components of the vector spanning the two surface patches, and (5) the properties of the interfacing edge.

Each constraint is measured and tested against a predetermined threshold. For surface segments that have an occluding edge, the neighbor information is not complete (a neighbor may be hidden) and the connectivity information may be inaccurate. Therefore, for such cases only a *weak arc* is formed in the attributed graph of the scene. A match based on a weak arc is subject to confirmation or rejection based on further evidence.

Matching between the two hypergraphs representing the scenes is achieved by computing the match between the component hyperedges. Hyperedges are matched by matching the attributed graphs representing the hyperedges. A vertex is selected from the hyperedge $H_1$ as the first vertex $n_1$ and it is matched with the corresponding vertex $n'_1$ in the hyperedge $H'_1$ in the second hypergraph. The unary and the binary constraints are checked to evaluate the match between the hyperedges. Once the hyperedge match has been established, the second set of hyperedges are selected. The next hyperedge $H_2$ is the hyperedge connected to $H_1$ at $n_1$. The matching hyperedge in the second hypergraph is selected from one of the hyperedges connected to $H'_1$ at $n'_1$. A match for each of the hyperedges connected to $H_1$ at $n_1$ is found. The search then proceeds to find matches of hyperedges connected to $H_1$ at other vertices belonging to $H_1$. For each hyperedge in the first hypergraph the corresponding hyperedge in the second hypergraph is considered from a topologically directed set. The procedure goes down the list of all the vertices in the hypergraphs in a predetermined order. Once a match for a vertex or hyperedge is found, that vertex or hyperedge is marked as *matched*. The marked vertices and hyperedges are not considered in the future hypotheses.

Algorithm 4.1 gives the hypergraph matching procedure.

ALGORITHM 4.1. Hypergraph matching algorithm.

```
begin
  1. ORDER ($hyperedge\text{–}list_1$, $vertex\text{–}list_1$, $Hypergraph_1$)
  2. ORDER ($hyperedge\text{–}list_2$, $vertex\text{–}list_2$, $Hypergraph_2$)
  3. INITIALIZE ($CH$, $CV$, $Hypergraph_1$, $Hypergraph_2$)
  4. $level = 0$
  5. FIND–MATCH ($CH$, $CV$)
end
```

Steps 1 and 2 of the algorithm order the hyperedges and the vertices in the scene hypergraphs. The order determines the branches taken in the interpretation tree. The ordering is done by selecting the first hyperedge $H_1$ in the hypergraph. The selection criterion can be (1) the size of the hyperedge (cardinality), or (2) the hyperedge with the vertex corresponding to the largest surface segment (area), or (3) the hyperedge containing the vertex with the highest degree. The ordered list of hyperedges is then formed by sorting the hyperedges in the order of the distance to the first selected hyperedge $H_1$. The distance is defined by the shortest distance between the component vertices of the hyperedges. If two hyperedges have vertices in common, then the distance is zero. When distances between two pairs of hyperedges are the same, then the selection criterion that was used to select the first hyperedge is used to break the tie. The order of the vertices is determined by ordering the vertices in the hyperedges in terms of the area or degree of the vertex.

The matching information is encoded in a *compatibility matrix*. The matching between the hyperedges is represented using the hyperedge compatibility matrix CH, where the $(i, j)$ entry represents the match (or possibility of a match) between the $i^{\text{th}}$ hyperedge in the first hypergraph with the $j^{\text{th}}$ hyperedge in the second hypergraph. Similarly the vertex compatibility matrix CV represents pairings of vertices. The $(i, j)$ entry gives the match (or possible match) between the $i^{\text{th}}$ vertex of the first hypergraph with the $j^{\text{th}}$ vertex in the second hypergraph. The entries of CV correspond to the nodes of the interpretation tree.

Step 3 of the matching procedure initializes the compatibility matrices (INITIALIZE). Each entry of the hyperedge compatibility matrix is initialized to 1 and the entries of the vertex compatibility matrix are initialized by considering the unary constraints. Entries corresponding to compatible pairs are set to 1 and the rest are set to 0. A depth first search of the interpretation tree is done in a recursive function (FIND–MATCH) and the level gives the depth of the recursion. The variable *level* is the level at which the current hypothesized match is in the interpretation tree. Algorithm 4.2 gives the recursive function FIND–MATCH.

ALGORITHM 4.2. FIND–MATCH.

```
begin
  1. while ($level \geq 0$) do
     1.1.  if ($\neg$Find–Level ($CH$, $level$, $row$, $col$)) then
           1.1.1. $level := level - 1$
           1.1.2. return
     1.2.  $CH_{bak} \leftarrow CH$;   $CV_{bak} \leftarrow CV$
     1.3.  Remove–Hyperedge–Incompatibilities
           ($CH$, $row$, $col$)
     1.4.  Extract–Submatrix ($CV_{sub}$, $CV$, $row$, $col$)
     1.5.  Remove–Submatrix–Incompatibilities
           ($CV_{sub}$, $CV$)
     1.6.  Get–Match–Set ($CV_{sub}$, $track–match$)
     1.7.  Remove–Vertex–Incompatibilities
           ($CV_{sub}$, $CV$)
     1.8.  if (Is–Solution ($CV$)) then
           1.8.1. Update–Best($CV$)
     1.9.  else
           1.9.1. $level := level + 1$
           1.9.2. Find–Match($CH$, $CV$)
     1.10. $CH \leftarrow CH_{bak}$;   $CV \leftarrow CV_{bak}$
     1.11. if (All–Sets–Matched($track–match$)) then
           1.11.1. Remove–Entry($CH$, $level$)
     od
end
```

Algorithm 4.2 recursively searches the tree in a depth-first manner. The function is called at the root of the interpretation tree ($level = 0$). The root of the tree corresponds to the first entry in the vertex compatibility matrix. The first nonzero entry in the hyperedge compatibility matrix provides with the first hypothesis, i.e., the hyperedges $H_1$ and $H'_1$ match. The hyperedge pairings are investigated by going down the interpretation tree. The *level* increases as the algorithm moves forward in the tree and decreases when the algorithm backtracks. When all the possible branches are exhausted, the *level* becomes less than 0. The search continues until there is a possibility of obtaining another match (the *while* loop in step 1). A hypothesis is generated by selecting an entry from the hyperedge compatibility matrix. Step 1.1 selects the *level*th nonzero entry from the *CH* matrix. If the nonzero *level*th entry exists in *CH* then the search has exhausted all the possible positive matches along the current branch, so the algorithm backtracks by decreasing the *level* (steps 1.1.1 and 1.1.2). If, however, the *level*th entry exists ((*row*, *col*) entry in the *CH* matrix) then the search progresses in the forward direction. The search involves updating the compatibility matrices; therefore, to enable backtracking, the matrices are backed up in step 1.2. At each stage, when a match is obtained, it is recorded in a *matched list* of all matches.

The hypothesis corresponding to the (*row*, *col*) entry of the *CH* matrix is "hyperedge $H_{row}$ in the first hypergraph corresponds to hyperedge $H'_{col}$ in the second hypergraph." Each hyperedge has an unique match; therefore, in step 1.3 the entries corresponding to matches between $H_{row}$ and other hyperedges, and those corresponding to matches between $H'_{col}$ and any other hyperedges, are set to 0. The test of hyperedge matching is done by testing the component attributed graphs. The submatrix in *CV*, representing the component vertices of the two hyperedges $H_{row}$ and $H'_{col}$, is extracted to compute the vertex matches (step 1.4). The entries is the submatrix $CV_{sub}$ are tested for compatibility with all the previous matches (in the *matched list*), i.e., all the pairings corresponding to the nodes of the interpretation tree in the same branch as the current hypothesis but above the current node are tested for compatibility with the pairings in the submatrix $CV_{sub}$. All incompatibilities in $CV_{sub}$ are removed.

The submatrix $CV_{sub}$ represents many sets of valid pairings. Each branch of the subtree rooted at the current hypothesis node is a valid pairing set. The algorithm investigates each set of pairings by keeping track of the sets it has already tested (*track–match*), i.e., the branches of the tree, fanning out of the current node, that have been tested are marked. Based on the sets already investigated, the next set for matching is extracted in step 1.6. The pairing set is a set of hypothesized vertex pairings. All entries in the *CV* matrix that are incompatible with the vertex pairings are removed in step 1.7.

At this point, if the algorithm obtains a solution (a pairing for each vertex in the graphs) or discovers that a match better than one obtained earlier is not feasible, then the algorithm backtracks to investigate other branches and obtain a better solution if possible. Step 1.8 checks for this condition and, if a solution is obtained, the current best match is updated. If it is determined that the algorithm does not have to backtrack, then the *level* is incremented and the search progresses in the forward direction by making a recursive call to FIND–MATCH. The algorithm investigates all the branches after the current node and returns to the current node on the way back. If all the possible pairings for the component vertices have been exhausted (step 1.11), then it implies that the best possible match with the current hypothesis has been obtained. The next hypothesis is investigated by removing the current entry in the restored *CH* matrix. The search continues and the next time step 1.1 is executed to determine the *level*th entry in the *CH* matrix, the new hypothesis is selected. If, however, it is determined in step 1.11 that all the possible pairings for the component vertices have not been exhausted, then the entry in the restored *CH* is not removed. This results in step 1.1 selecting the same hypothesis again, and step 1.6 ensures that the next pairing set is investigated by the algorithm.

Steps 1.5 and 1.7 are used to remove the incompatibilities of the entries of the vertex compatibility matrix. In step 1.7, a look-forward strategy can be implemented in which all possible future hypotheses are checked for compatibility with the current hypothesis. This feature significantly prunes the tree and thus reduces the search space. However, the look-forward strategy may turn out to be computationally expensive if the tree is large and if the probability of a false positive match is high. Therefore, the look-forward strategy is executed only when the total number of vertices in the graph is below a threshold number. Simple incompatibility tests, such as uniqueness of match, are used in either case to delete the entries from the matrix.

Occlusions can cause vertices in the hypergraph to disappear. This may cause hyperedges to split up and form new hyperedge sets. This phenomenon is accounted for in the matching algorithm, i.e., a provision has been made to dynamically change the hypergraph. By detecting the occluding edges, the locations in the hypergraph where occlusions may occur and modify the hypergraph are identified. For this reason, surfaces that are connected by an interfacing occluding edge form a weak arc in the graph representation. The algorithm will break the weak arc to form a new hypergraph when breaking the weak arc will give a better match.

When the algorithm determines that the best possible match that can be obtained by moving down the current branch is smaller than a threshold value, then it backtracks without going any further. Termination of the matching procedure occurs if the fraction of surface segments matched exceeds a threshold. Once a match has been determined (i.e., the search procedure has reached the leaf node of the tree), the number of positive pairings (i.e., non-wild card pairings) is computed. If this number is less than the threshold fraction, then the procedure backtracks and searches other branches. At every stage, the best possible match is compared with the current best match. If the best possible match is smaller than the current match, then the search along that branch is abandoned and the next branch is investigated.

## 4.1. *Complexity Issues*

In [14, 15] Grimson *et al.* made an extensive analysis of the complexity of the tree search procedure. Many procedures exist to compute the match between graphs, and most of the procedures are related to the constrained tree search in some form.

In the constrained tree search procedure, the fanout at each level is the total number of the vertices in the graph. However, in the hypergraph matching strategy introduced here, the current status of the match is tracked by using the compatibility matrices. Since the new hypothesis never considers a feature that already has been matched, this reduces the fanout in the tree nodes as the algorithm progresses deeper into the tree. The hypergraph representation affects the fanout in a more significant manner. In the matching procedure, the hypothesis about the hyperedge match is made first and then the match between the vertices in the hyperedge is hypothesized. Therefore, at the node of the tree where the hyperedge matching is carried out, the fanout is large (all the unmatched hyperedges are considered) but in the next $s$ ($s$ = the size of the hyperedge of the first hypergraph) levels of the tree, the fanout is bound by the size of the hyperedge of the second hypergraph.

Another reduction in the search space arises from the directed search strategy that estimates the best next branch to search. This reduces the average case complexity, although in the worst case there is no reduction in the search space. Finally, the use of termination criteria, and the criteria to abandon a particular branch and backtrack, makes a considerable reduction in the average case complexity.

The worst case complexity of the largest common subgraph isomorphism is exponential. The worst case will happen if all the hyperedges detected in the graph have only one vertex (i.e., the hypergraph and the graph representation are exactly the same). The hypergraph representation moves the computations in the matching procedure to the construction of the hypergraph. In general, the complexity of hypergraph construction (clique detection) is of exponential complexity, however, because of the bound on the size of the hyperedges, the hypergraph construction algorithm is of polynomial complexity. The heuristics derived from the topology and the geometry of the graphs only help to reduce the average case complexity and to prune the branches in the search tree. The goal has been to maximize the use of *a priori* information and the other sources of information in the search strategy.

## 4.2. *Illustrative Example*

The algorithm was tried successfully on different types of range image sequences. This section presents an example of a range image sequence and illustrates how the matching algorithm computes the surface correspondences.

Figures 3–5 demonstrate the different stages of the algorithm. Figure 3 shows the depth maps of two frames in the sequence of range images. The scenes consist of a jumble of three objects. One surface is cylindrical and the rest are planar. The camera is moved relative to the scene to obtain the second frame of the sequence. The segmentation algorithm was applied to the image data and the results were input to the matching algorithm. The segmented results are shown in Fig. 4. Notice that this example has only one

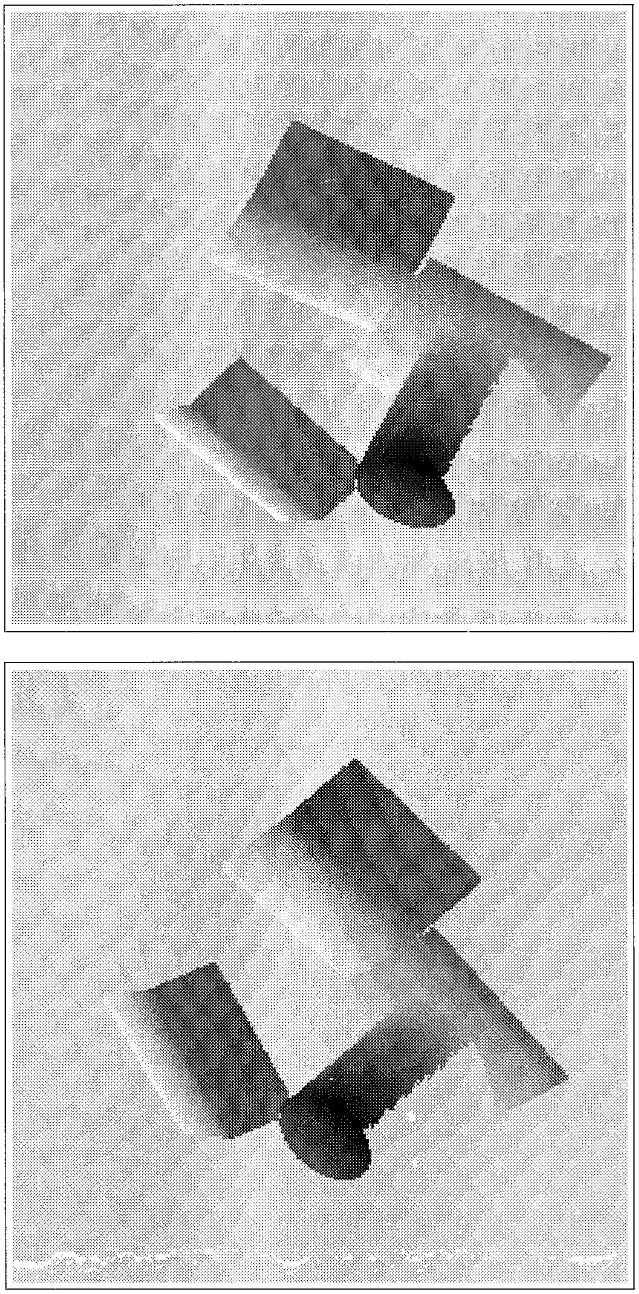


FIG. 3. The depth maps of a sequence of range images.

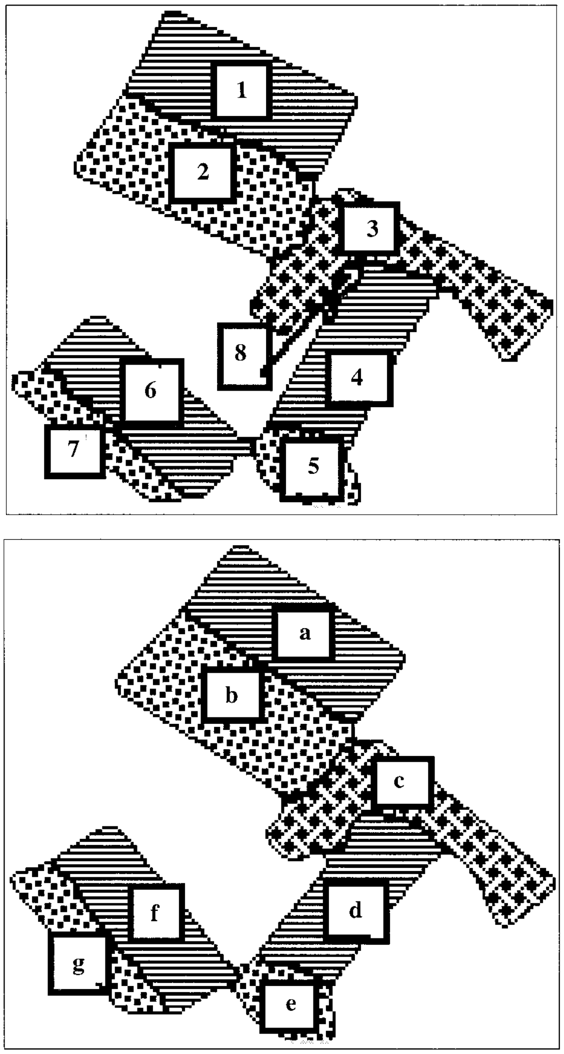


FIG. 4. The segmented results of the sequence of range images.

instance of incorrect segmentation. The first step of the algorithm generates the attributed graph of the scene and computes the cliques in the graph. Each clique forms a hyperedge in the generated hypergraph. The generated hypergraphs are shown in Fig. 5. For each hyperedge, the component vertices form an attributed graph. In the figure, the arcs of the attributed graph are shown in the hyperedges. Using the properties of the edge interfacing two surface segments, it is determined if two surfaces are connected. If an occluding edge occurs between two surfaces, then the arc in the attributed graph is *weak* (shown in Fig. 5 with dotted lines). A match based on a *weak* arc is a *weak* match and further evidence is required to confirm the hypothesis.

The algorithm starts by initializing the compatibility ma-

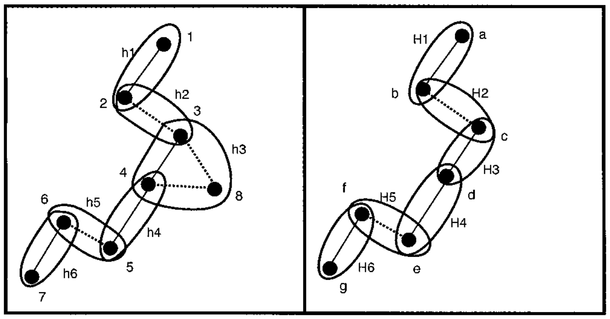


FIG. 5. The constructed hypergraphs and the component graphs.

trices. The vertex compatibility is initialized by considering the unary constraints. The matrix is given as:

| | *a* | *b* | *c* | *d* | *e* | *f* | *g* |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 | | 1 | 1 | 1 |
| 4 | | | | 1 | | | |
| 5 | 1 | 1 | 1 | | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | | 1 | 1 | 1 |
| 8 | | | | | | | |

The entries of 1 imply that the unary constrains are satisfied and a blank implies they are not satisfied. The first hyperedge pair hypothesized to match is {3, 4, 8}, in the first scene it matches {*a*, *b*}. The vertex with the highest degree, 3, is considered as the first vertex. The first vertex pair hypothesized to match is (3, *a*). The first set of possible pairings, that satisfy the binary constraints with (3, *a*), in the submatrix is chosen to be {(3, *a*), (4, ★), (8, ★)}. The vertex compatibility matrix, after removing all the incompatibilities with {(3, *a*), (4, ★), (8, ★)}, is:

| | *a* | *b* | *c* | *d* | *e* | *f* | *g* |
|---|---|---|---|---|---|---|---|
| 1 | | | 1 | | | | |
| 2 | | | | | | | |
| 3 | 1 | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | 1 | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |

Note that for this example the total number of vertices is below the threshold, so the look-forward strategy is invoked, which deletes most of the entries in the compatibility matrix. The incompatibilities occur because the binary constraints are not satisfied. It can be seen that the best possible match with the current hypothesis can match only three surfaces. This is lower than the threshold set for the minimum number of surfaces to be matched; therefore, the algorithm backtracks. Since the algorithm is at *level* 0, the first hypothesized hyperedge match is changed. After another search in which the algorithm backtracks after the second level of the tree, the hyperedge {3, 4, 8} is paired with the hyperedge {*c*, *d*}. The vertex pair hypothesized to match is (3, *c*). The first set of pairings of vertices of the two hyperedges that satisfies the binary constraints with (3, *c*), is {(3, *c*), (4, *d*), (8, ★)}. The next hyperedge to be considered for match is {2, 3} because it intersects with the first hyperedge at 3. Since 3 is paired with *c*, the hyperedge in the second scene considered for match is {*b*, *c*}. The binary constraints are satisfied for the set of pairings {(3, *c*), (2, *b*)}. This process continues and the final match is:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| II | *a* | *b* | *c* | *d* | *e* | *f* | *g* | ★ |

## 5. MOTION COMPUTATION

After the correspondence between surface segments has been established, the next task is to compute the motion transformation between the two successive frames. Ideally, the computation of the motion transform is equivalent to the computation of the affine transform, because the rigid motion transform is a special case of the affine transform. In practice, however, as a consequence of noise, parameter estimation, and measurement errors, it is impossible to realize the ideal situation. The motion transform space is a subspace in the space of all affine transforms. The distinguishing factor is the orthonormality of the rotation transform. If inaccurate data is used in computing the affine transform, the solution obtained need not be a rigid motion transform. In a vision system, it is important to ensure that the transform is a rigid motion transform because the output of the motion transform will be used to carry out some physical task which is likely to fail if the estimated transform is not a rigid motion. (For example, in navigation, the motion of the robot/target has to be rigid.)

Local features, such as points and lines, are sensitive to noise and occlusions; therefore, to have a reliable and robust procedure for motion computation, it is more appropriate to use global features, such as surfaces. Since the input data has inaccuracies due to roundoff errors and noise in the acquisition process, it is more appropriate to compute the optimal motion transformation that best fits the input data.

Formulating the motion computation task requires dealing with the issues of representation of the motion transformation, representation of the feature, and the use of these representations to compute the transformation. These issues are tightly linked, and the complexity and the solvability of the problem is completely dependent on them. Also, it is important to consider the uniqueness and the optimality of the solutions.

In [22] it was shown that some representations of the transformation $\mathscr{T}$ have singularities. At such singularities, the motion estimation task fails, so the motion computation algorithm has to account for such situations. The two components of motion, rotation and translation, are represented separately. The rotation transformation is represented as a $3 \times 3$ matrix whose row vectors are orthogonal and of unit magnitude.

$$\mathbf{R} = \begin{pmatrix} \vec{R}_1^T \\ \vec{R}_2^T \\ \vec{R}_3^T \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{3}$$

Rotation is computed by estimating these row vectors. Translation is represented by a vector $\mathbf{T} = (t_1 t_2 t_3)^T$.

## 5.1. *Motion Transformation for Planar Surfaces*

The motion parameters are computed from a sequence of range images, given the suface correspondences between frames. Before addressing the general case, where any general type of surface correspondence is used to compute the motion transformation, the problem is made more tractable by restricting the domain to planar surface segments. This assumption results in the set $\mathbf{S} = \{(P_i, P_i')\}$ of surface correspondence to represent pairs of plane surface parameters.

The plane is represented by two parameters, the normal $\hat{\mathbf{n}}$ and the distance to the origin $d$. Let the two planes $P_i(\hat{\mathbf{n}}_i, d_i)$ and $P_i'(\hat{\mathbf{n}}_i', d_i')$, be related by a motion transformation, $(\mathbf{R}, \mathbf{T})$, then

$$\hat{\mathbf{n}}_i' = \mathbf{R}\hat{\mathbf{n}}_i \quad \text{and} \quad d_i' = \hat{\mathbf{n}}_i' \cdot \mathbf{T} + d_i. \tag{4}$$

The correspondence computation stage gives the pairings between $(\hat{\mathbf{n}}_i, \hat{\mathbf{n}}_i')$ and $(d_i, d_i')$; therefore, Eq. (4) can be used to compute $\mathbf{R}$ and $\mathbf{T}$ independently.

Some assumptions about the problem and the domain have to be made before the problem can be solved. Given a set of planes $\{(\hat{\mathbf{n}}_i, d_i)\}$, in the first frame, and the corresponding set of planes $\{(\hat{\mathbf{n}}_i', d_i')\}$, in the second frame, the following assumptions are made:

1. The only motion observed is rigid motion. No deformation of objects occurs. This assumption restricts the domain of possible transformations of the objects in the scene.

2. The scene has at least three surface normals $\hat{\mathbf{n}}_i s$ in each frame that are linearly independent. The corresponding surface normals in the next frame $\hat{\mathbf{n}}_i'$ are also detected and are linearly independent. This assumption is necessary to ensure that the motion transformation obtained is unique.

In general, the number of planes in the scene is more than three; therefore, a least-squares estimate of the rotation is obtained from the overdetermined set of equations. The rotation is computed by minimizing equation (5.a), with the constraint that the rotation matrix is an orthonormal matrix.

$$e_R = \sum_{i=1}^{N} \|\hat{\mathbf{n}}_i' - \mathbf{R}\hat{\mathbf{n}}_i\|^2 \tag{5.a}$$

subjet to

$$\begin{aligned} &\vec{R}_1 \cdot \vec{R}_1 = 1, \quad \vec{R}_2 \cdot \vec{R}_2 = 1, \quad \vec{R}_3 \cdot \vec{R}_3 = 1, \\ &\vec{R}_1 \cdot \vec{R}_2 = 0, \quad \vec{R}_1 \cdot \vec{R}_3 = 0, \quad \text{and } \vec{R}_2 \cdot \vec{R}_3 = 0, \end{aligned} \tag{5.b}$$

where $\vec{R}_i s$ are the row vectors of the orthonormal matrix (Eq. 3). The error term to be minimized to compute the translation is

$$e_T = \sum_{i=1}^{N} (d_i' - d_i - \mathbf{T} \cdot \hat{\mathbf{n}}_i')^2. \tag{6}$$

The reflection transformation satisfies all the constraints of the rigid motion transformation, but it does not preserve the intrinsic properties of the features. The uniqueness of the solution can be ensured only if reflections are excluded from the solution space. This can be done by either imposing the constraint of *determinant*($\mathbf{R}$) = 1 on the rotation matrix, or ensuring that the intrinsic properties of the sets of normal vectors are similar. By similar, it is meant that the intrinsic properties are close to each other. Since the scalar triple product of the normals is an intrinsic property,

$$[\hat{\mathbf{n}}_i \hat{\mathbf{n}}_j \hat{\mathbf{n}}_k] \approx [\hat{\mathbf{n}}_i' \hat{\mathbf{n}}_j' \hat{\mathbf{n}}_k']. \tag{7}$$

Since the correspondence computation between surfaces ensures that the intrinsic properties are preserved, no new constraint has to be imposed in the motion computation stage.

The constraints are incorporated into the equation for $e_R$ using Lagrange multipliers. The resulting error functional is given by

$$\begin{aligned} e_{new} = N &- \vec{R}_1 \cdot \vec{S}_1 - \vec{R}_2 \cdot \vec{S}_2 - \vec{R}_3 \cdot \vec{S}_3 \\ &+ \lambda_1(1 - \vec{R}_1 \cdot \vec{R}_1) + \lambda_2(1 - \vec{R}_2 \cdot \vec{R}_2) \\ &+ \lambda_3(1 - \vec{R}_3 \cdot \vec{R}_3) + \mu_1 \vec{R}_1 \cdot \vec{R}_2 \\ &+ \mu_2 \vec{R}_1 \cdot \vec{R}_3 + \mu_3 \vec{R}_2 \cdot \vec{R}_3, \end{aligned} \tag{8}$$

where the $\lambda_i$ and $\mu_i$ are the Lagrange multipliers and $\vec{S}_i$ are constant vectors in terms of the normals. The solution of the minimization problem requires solving nonlinear equations; therefore, an iterative algorithm is used to arrive at the solution. The optimal estimate of translation is obtained by minimizing $e_T$ in Eq. (6), resulting in a set of linear equations in terms of the translation parameters.

From the assumption that at least three linearly independent planar surfaces exist, it can be shown that the rotation and the translation can be uniquely determined. Since the rotation matrix, $\mathbf{R}$, is orthogonal, the third row vector can be expressed as the cross product of the first two row vectors. Therefore, to determine the rotation, two row vectors have to be calculated. Two linearly independent vector equations in terms of the unknown row vectors is sufficient to compute the entire rotation matrix. Hence from Eq. (4), at least two pairs of corresponding normals that are linearly independent (i.e., nonparallel planes) are required to compute the rotation uniquely. Therefore, it can be seen that:

THEOREM 5.1. *Given two nonparallel vectors in the first frame of an image sequence and their corresponding vectors in the second frame of the sequence, then the rotation about the origin can be determined uniquely.*

The translation vector has three components $(t_1, t_2, t_3)$; therefore, from Eq. (4), to uniquely compute the translation at least three pairs of linearly independent surface normals are required. The net motion transformation can be uniquely computed by using a minimum of three planar surface patches in each frame whose normals are linearly independent. This is stated as a theorem.

THEOREM 5.2. *Given three plane surfaces whose normals are linearly independent* (*i.e., are not parallel to each other*), *then the translation can be uniquely determined.*

In general, more than three planar surface patches may be extracted from the object. To minimize the error over all observed data, the optimal motion in the least-mean-squares sense is estimated. It is important to assure that the minimization procedure results in a unique solution. The proof of uniqueness of the rotation transformation is based on well known results in linear algebra and has been shown by Horn [16, 17].

## 5.2. *Quadric Surfaces*

The computation of motion from a sequence of planar surfaces has the advantage that the rotation and translation estimation equations separate into two independent equations. Separating the translation estimation from the rotation estimation is the key to obtaining a simple solution for the motion transformation. This separation occurs for all linear features. However, in the case of quadric surfaces, the motion computation equations are not as well behaved as in the case of planar surfaces. The resulting equations are extremely nonlinear and there is no guarantee of a unique and optimal solution.

Quadric surfaces are in general represented by the second-order equation given by

$$\begin{aligned} a_1x^2 + a_2y^2 + a_3z^2 + 2a_4yz + 2a_5zx \\ + 2a_6xy + 2a_7x + 2a_8y + 2a_9z + a_{10} = 0, \end{aligned} \tag{9}$$

where $a_1$, $a_2$, $a_3$, $a_4$, $a_5$, $a_6$, $a_7$, $a_8$, $a_9$, and $a_{10}$ are the parameters of the quadric. Equation (9) is written as

$$\mathbf{X}^TD\mathbf{X} + \mathbf{X}\cdot E + J = 0, \tag{10}$$

where

$$\mathbf{X} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad D = \begin{pmatrix} a_1 & a_4 & a_5 \\ a_4 & a_2 & a_6 \\ a_5 & a_6 & a_3 \end{pmatrix} \quad E = \begin{pmatrix} a_7 \\ a_8 \\ a_9 \end{pmatrix} \quad J = a_{10}. \tag{11}$$

If the quadric surface is transformed by a rotation $\mathbf{R}$ and a translation $\mathbf{T}$ then the points on the quadric are related to the transformed points by the equation

$$\mathbf{X}_{new} = \mathbf{RX} + \mathbf{T}. \tag{12}$$

Also, the quadric before and after motion is given by

$$\mathbf{X}^TD\mathbf{X} + E^T\mathbf{X} + J = 0 \tag{13.a}$$

$$\mathbf{X}_{new}^TD_{new}\mathbf{X}_{new} + E_{new}^T\mathbf{X}_{new} + J_{new} = 0. \tag{13.b}$$

Substituting Eq. (12) in Equation (13.b), we obtain

$$\begin{aligned} (\mathbf{RX} + \mathbf{T})^TD_{new}(\mathbf{RX} + \mathbf{T}) \\ + E_{new}^T(\mathbf{RX} + \mathbf{T}) + \mathbf{J}_{new} = 0 \end{aligned} \tag{14.a}$$

$$\begin{aligned} \mathbf{X}^T\mathbf{R}^TD_{new}\mathbf{RX} + \mathbf{T}^TD_{new}\mathbf{T} \\ + X^T\mathbf{R}^TD_{new}\mathbf{T} + \mathbf{T}^TD_{new}\mathbf{RX} \\ + E_{new}^T\mathbf{RX} + E_{new}^T\mathbf{T} + J_{new} = 0 \end{aligned} \tag{14.b}$$

$$\begin{aligned} \mathbf{X}^T(\mathbf{R}^TD_{new}\mathbf{R})\mathbf{X} + (\mathbf{t}^TD_{new}\mathbf{R} + E_{new}^T\mathbf{R})\mathbf{X} \\ + \mathbf{X}^T\mathbf{R}^TD_{new}\mathbf{T} + (J_{new} + E_{new}^T\mathbf{T} \\ + \mathbf{T}^TD_{new}\mathbf{T}) = 0. \end{aligned} \tag{14.c}$$

But

$$\mathbf{X}^T\mathbf{R}^TD_{new}\mathbf{T} = (\mathbf{X}^T\mathbf{R}^TD_{new}\mathbf{T})^T, \tag{15}$$

as it is a scalar. Also as $D_{new}$ is a symmetric matrix,

$$D_{new} = D_{new}^T. \tag{16}$$

Therefore

$$\begin{aligned} \mathbf{X}^T(\mathbf{R}^TD_{new}\mathbf{R})\mathbf{X} + 2\mathbf{t}^TD_{new}\mathbf{R} + E_{new}^T\mathbf{R})\mathbf{X} \\ + (J_{new} + E_{new}^T\mathbf{T} + \mathbf{T}^TD_{new}\mathbf{T}) = 0. \end{aligned} \tag{17}$$

Comparing the coefficients with Eq. (13.a) we get

$$D = \mathbf{R}^TD_{new}\mathbf{R} \tag{18.a}$$

$$E^T = (2\mathbf{T}^TD_{new}\mathbf{R} + E_{new}^T\mathbf{R}) \tag{18.b}$$

$$J = (J_{new} + E_{new}^T\mathbf{T} + \mathbf{T}^TD_{new}\mathbf{T}), \tag{18.c}$$

or this can be rewritten as

$$D_{new} = \mathbf{R}D\mathbf{R}^T \tag{19.a}$$

$$E_{new} = \mathbf{R}E - 2\mathbf{R}D\mathbf{R}^T\mathbf{T} \tag{19.b}$$

$$J_{new} = J - E^T\mathbf{R}^T\mathbf{T} + \mathbf{T}^T\mathbf{R}D\mathbf{R}^T\mathbf{T}. \tag{19.c}$$

From Eqs. (18.a), (18.b), and (18.c), a least-squares minimization problem is formulated to compute the best estimate of the motion transformation. If the set of quadric surface correspondences is given as the set $\mathbf{S} = \{(Q_i, Q_i')\}$, where the surface parameter of the quadric is represented by $Q_i = (D_i, E_i, J_i)$, then the squared error for each quadric surface pair, $\varepsilon_i$, can be written as

$$\varepsilon_i = \|D_i - \mathbf{R}^T D_i'\mathbf{R}\|^2 + \|E_i - \mathbf{R}^T E_i' - 2\mathbf{R}^T D_i'\mathbf{T}\|^2 + \|J_i - J_i' - E_i'^T\mathbf{T} - \mathbf{T}^T D_i'\mathbf{T}\|^2. \tag{20.a}$$

The matrix norm $\|\cdot\|$ is taken as the Frobenius norm, i.e., $\|A\| = Tr(A^TA)$. Then

$$\begin{aligned}\varepsilon_i = {} & Tr((D_i - \mathbf{R}^T D_i'\mathbf{R})^T(D_i - \mathbf{R}^T D_i'\mathbf{R})) \\ & + (E_i - \mathbf{R}^T E_i' - 2\mathbf{R}^T D_i'\mathbf{T})^T \\ & (E_i - \mathbf{R}^T E_i' - 2\mathbf{R}^T D_i'\mathbf{T}) \\ & + (J_i - J_i' - E_i'^T\mathbf{T} - \mathbf{T}^T D_i'\mathbf{T})^T \\ & (J_i - J_i' - E_i'^T\mathbf{T} - \mathbf{T}^T D_i'\mathbf{T})\end{aligned} \tag{20.b}$$

$$\begin{aligned} = {} & Tr(D_i^T D_i) - 2Tr(D_i\mathbf{R}^T D_i'\mathbf{R}) + Tr(\mathbf{R}^T D_i'^T D_i'\mathbf{R}) \\ & + E_i^T E_i - 2E_i^T\mathbf{R}^T E_i' - 4E_i^T\mathbf{R}^T D_i'\mathbf{T} \\ & + E_i'^T E_i' + 4E_i'^T D_i'\mathbf{T} + 4\mathbf{T}^T D_i'^T D_i'\mathbf{T} \\ & + (J_i - J_i')^2 + 2(J_i' - J_i)E_i'^T\mathbf{T} \\ & + 2(J_i' - J_i)\mathbf{T}^T D_i'\mathbf{T} + (E_i'^T\mathbf{T})^2 \\ & + (\mathbf{T}^T D_i'\mathbf{T})^2 + 2(E_i'^T\mathbf{T}\mathbf{t}^T D_i'\mathbf{T})\end{aligned} \tag{20.c}$$

$$\begin{aligned} = {} & Tr(D_i^T D_i) + Tr(D_i'^T D_i') + E_i^T E_i + E_i'^T E_i' \\ & + (J_i - J_i')^2 - 2Tr(D_i\mathbf{R}^T D_i'\mathbf{R}) \\ & - 2E_i^T\mathbf{R}^T E_i' - 4E_i^T\mathbf{R}^T D_i'\mathbf{T} + 4E_i'^T D_i'\mathbf{T} \\ & + 2(J_i' - J_i)E_i'^T\mathbf{T} + 4T^T D_i'^T D_i'\mathbf{T} \\ & + 2(J_i' - J_i)\mathbf{T}^T D_i'\mathbf{T} + \mathbf{T}^T E_i' E_i'^T\mathbf{T} \\ & + 2E_i'^T\mathbf{T}\mathbf{T}^T D_i'\mathbf{T} + (\mathbf{T}^T D_i'\mathbf{T})^2.\end{aligned} \tag{20.d}$$

The final error to be minimized is $\varepsilon = \sum_i \varepsilon_i$, with the constraints of orthonormality on the rotation matrix. The expression (20.d) shows the highly nonlinear nature of the error to be minimized and, further, that the translation and the rotation are not separable. The minimization has to be carried over 12 parameters (9 rotation + 3 translation), and it is impossible to guarantee the uniqueness and global optimality of the solution.

5.2.1. *Linear features in quadric surfaces.* Since the coefficients of the quadric cannot be used directly to compute the motion transformation, the linear features extracted from the quadric are used in the motion computation. Quadrics are nonlinear surfaces that have intrinsic linear features. This section shows how unique linear features are extracted from the quadric parameters. These features are used in the computation of motion. In some instances, where there is an ambiguity in the extracted linear feature (e.g., where there are three principal planes and there is no way of distinguishing among them), the approximate estimate of the motion is used to disambiguate the feature correspondences. The motion transformation is refined using all the features.

The objective is to derive linear geometrical entities such as planes, lines, or points form the quadric surface which are unique. A few of such features are the diametral plane, principal planes, line of centers, line of vertices, center, and vertex. The following theorems provide the justifications to use the linear features. The theorems are presented with out the proofs. The proofs can be found in [24].

THEOREM 5.3. *The locus of the middle point of a system of parallel chords of a quadric is a plane* [24].

This plane is called the diametral plane. The existence of a diametral plane is important, but it cannot be used as a feature because it is not unique. The next result is more important and shows the existence of a unique linear feature.

THEOREM 5.4. *All the diametral planes of a quadric have at least one* (*finite or infinite*) *point in common* [24].

It can be shown that if

$$D = \begin{vmatrix} a_1 & a_4 & a_5 \\ a_4 & a_2 & a_6 \\ a_5 & a_6 & a_3 \end{vmatrix} \quad N = \begin{vmatrix} a_1 & a_4 & a_7 \\ a_4 & a_2 & a_8 \\ a_5 & a_6 & a_9 \end{vmatrix}$$
$$M = \begin{vmatrix} a_1 & a_5 & a_7 \\ a_4 & a_6 & a_8 \\ a_5 & a_3 & a_9 \end{vmatrix} \quad L = \begin{vmatrix} a_4 & a_5 & a_7 \\ a_2 & a_6 & a_8 \\ a_6 & a_3 & a_9 \end{vmatrix} \tag{21}$$

and $D \neq 0$, then the planes intersect in a single finite point given by

$$\mathbf{X}_0 = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} -\dfrac{L}{D} \\ \dfrac{M}{D} \\ -\dfrac{N}{D} \end{pmatrix}. \qquad (22)$$

If this point lines on the quadric then it is called the *vertex* of the quadric and if it does not lie on the quadric then it is called the *center* of the quadric.

If $D = 0$, but $L, M, N$ are not all zero, then the diametral planes intersect at a single point at infinity. If $D$ is of rank 2, then the planes intersect in a line. If the line is finite and does not lie on the quadric, then it is called the *line of centers*, and if the line lines on the quadric, then it is called the *line of vertices*. If $D$ is of rank 1, then the diametral planes coincide; and if the plane does not lie on the quadric, then it is called *the plane of centers*, and if the plane lies on the quadric, then it is called the *plane of vertices*.

A diametral plane that is perpendicular to the chords it bisects is called a *principal plane*.

THEOREM 5.5. *If the coefficients in the equation of a quadric are real, and if the quadric does not have the plane at infinity as a component, then the quadric has at least one real, finite, principal plane* [24].

As a consequence of theorems 5.4 and 5.5, unique linear features can be extracted from the quadrics.

### 5.3 *Summary of Motion Computation*

In summary, the procedure to compute the motion transformation from planar surface correspondences, $\mathbf{S} = \{(P_i, P_i')\}$, is divided into two minimization procedures. The first minimization gives the estimate for rotation, and the second minimization estimates the translation. The minimization for rotation is a constrained minimization, where the constraint is the orthonormality of the rotation matrix. The solution to the minimization is guaranteed to result in an optimal and unique estimate for the rotation transformation. Similarly translation is computed by solving Eq. (6), resulting in unique and optimal translation estimates.

The computation of motion from quadric surface correspondences does not simplify into separable expressions for rotation and translation. The minimization, in terms of the quadric parameters, is nonlinear, and there is no guarantee of a unique and optimal solution. Motion computation from quadric surface correspondences is simplified to the planar case by extracting linear features from the quadric surfaces and then using the linear features to compute the motion transformation.

**TABLE 1**
**Example 1: Surface Pairings Computed by the Hypergraph Isomorphism Algorithm**

| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| II | ★ | ★ | ★ | *c* | *d* | *f* | *h* | ★ | *i* | *j* |
| I | 11 | 12 | 13 | 14 | 15 | ★ | ★ | ★ | ★ | |
| II | *k* | *l* | *n* | *m* | *o* | *a* | *b* | *e* | *g* | |

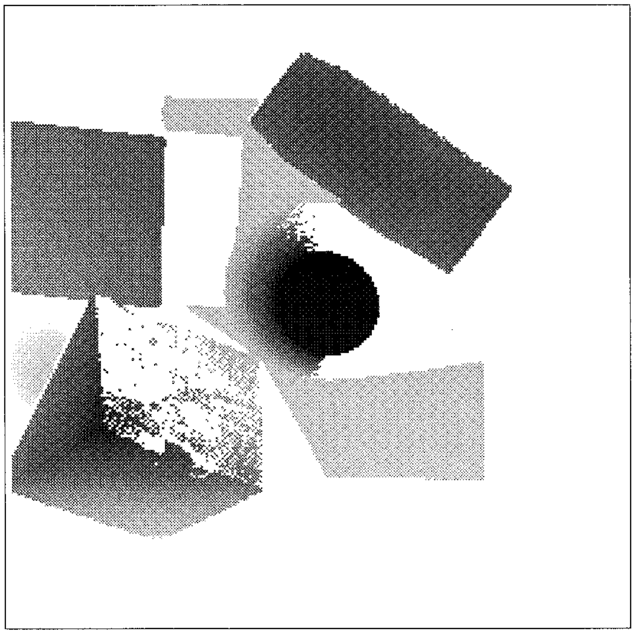
The segmentation stage gives the equation of the quadratic that fits each surface. Next, using Eq. (21), $D$ is computed. If $D$ is not zero then the center or vertex of the quadric $\mathbf{X}_0$ is computed from Eq. (22). $\mathbf{X}_0$ is used as the linear feature of the quadric surface. If $D$ is zero then the principal planes are computed. If the rank of $D$ is 2 then the intersection of the principal planes is computed to give the line of centers or vertices. This line of centers or vertices is used as the linear feature extracted from the quadric. Finally if the rank of $D$ is 1 then the principal planes coincide and this unique plane is used as the linear feature of the quadric.

Finally, the motion transformation is computed by solving the least-square Eqs. (5a) and (6). The error is minimized over all the planar surfaces and the computed linear features of the quadric surfaces.

## 6. RESULTS

Several experiments were performed with different types of image sequences. The scenes involved partially occluded surfaces and occlusion of surfaces over the image sequence. Here we present results of only two of the experiments (6 to 11). The first example (Figs. 6–8) shows a pair of range images that vary by a translation along the $x$ direction. In the second frame of the sequence, surfaces not visible in the first frame are detected. The segmentation results are shown in Fig. 7. There are some errors in segmentation, e.g., the regions 2 and 5 in the first frame are merged in the second frame into one region, $d$. The errors in segmentation and the appearance of new surfaces cause a difference in the two scene graphs shown in Fig. 8. The graph and the edge information is used to arrive at the hypergraph representations of the two frames (Fig. 8). The matching algorithm pairs the surfaces in the two frames and the results are presented in Table 1. The matches are used by the motion computation stage to compute the motion transformation between the two frames. The actual motion transformation and the computed motion transformation is given in Table 2.

The second example (Figs. 9–11, Tables 3 and 4), illustrates the procedure for a rotation about the $z$-axis and translations along the $x$ and the $y$ directions.



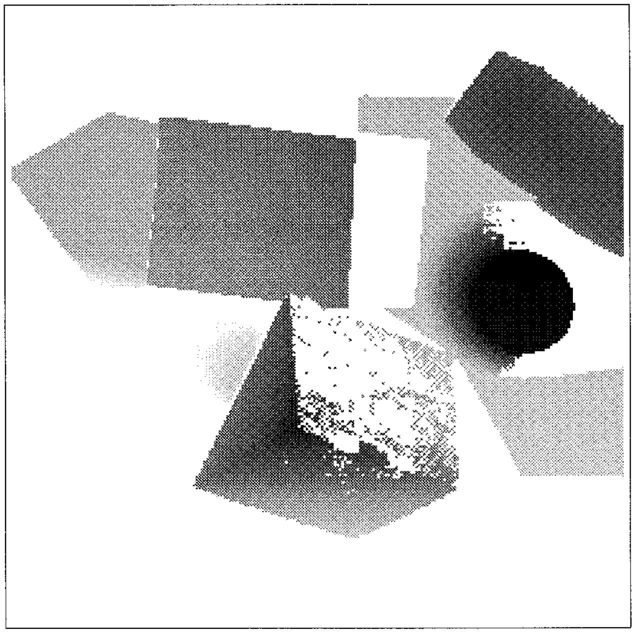


**FIG. 6.** Example 1, the depth maps of a sequence of range images.

**TABLE 2**
**Example 1: Rotation and Translation Estimates**

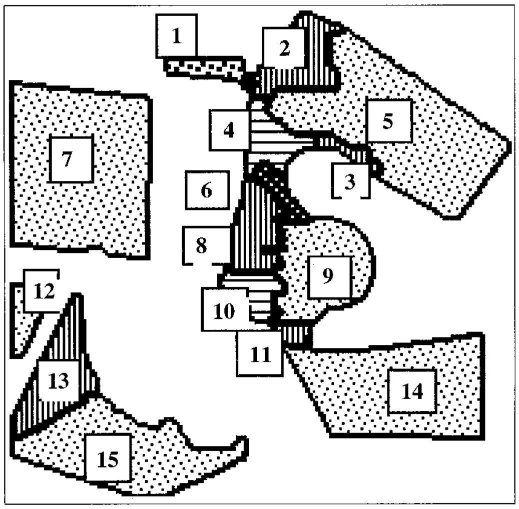
| Rotation | Actual | Estimated | Translation | Actual | Estimated |
|---|---|---|---|---|---|
| $\theta_x$ | 0° | 0.595° | $X$ | 10.0 cm | 12.447 cm |
| $\theta_y$ | 0° | 0.533° | $Y$ | 0.0 cm | 1.55 cm |
| $\theta_z$ | 0° | −0.215° | $Z$ | 0.0 cm | −0.602 cm |

*Note.* The images cover a range of 30.0 cm in the *x* and *y* directions.

## 7. SUMMARY AND CONCLUSIONS

Motion estimation from a sequence of images is a common task in many areas of computer vision research, and a large body of work has been published on the computation of motion from sequences of two-dimensional images. Little work has been done using sequences of three-dimensional images, such as dense range images, however, and the work presented aims to fill this crucial gap. Because 3-D images provide information on the structure of the scene and its objects, whereas 2-D images require that the structure be computed, 3-D (range) image sequences are clearly advantageous. This paper presents a general methodology for estimating rigid motion parameters from a sequence of range data using planar and quadric surfaces as the features.

Estimating the motion transformation over a sequence



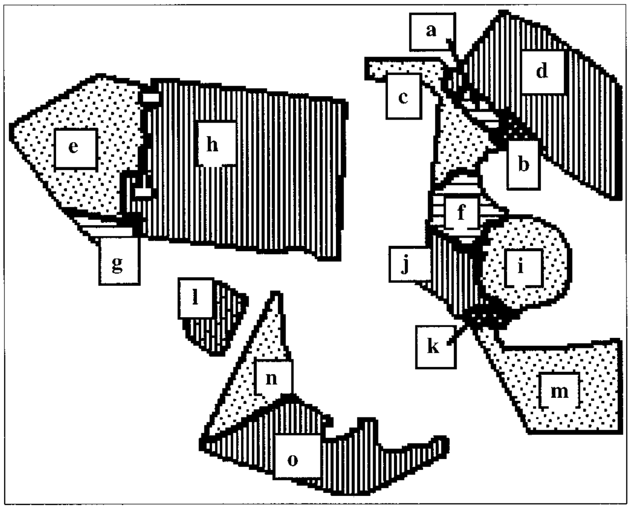


**FIG. 7.** Example 1, the segmented range images.

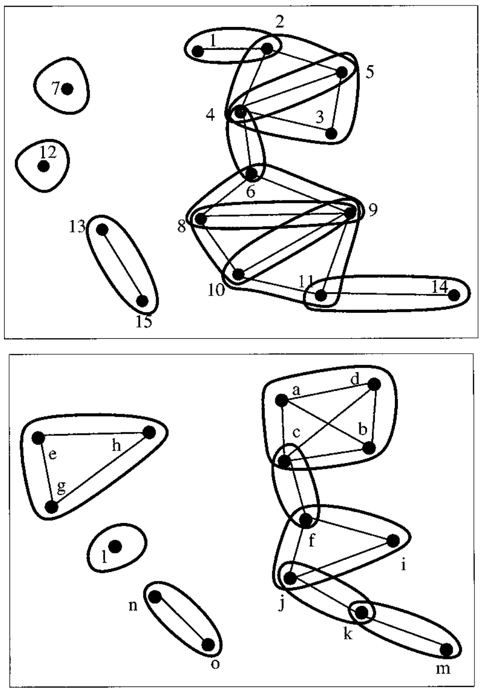


**FIG. 8.** Example 1, the hypergraphs generated from the graphs of the range images.

of range images involves solving the three subproblems of (1) segmenting the range images in each frame and extracting the key features, (2) establishing correspondence of the key features between frames (matching), and (3) computing motion using these feature correspondences. Each subproblem is solved using planar and quadric surfaces as the key features.

The use of surfaces as key features makes the methodology robust to data errors intrinsic in the acquisition and estimation of range information. Local features, such as lines and points, are quite sensitive to noise and thus will not yield accurate results for motion computation. Using global features like surfaces, on the other hand, tends to smooth out the errors in each individual point. Also, since the 3-D data acquisition systems usually sense the surfaces of the objects in the scene, they are a natural choice for the key features to be used in the motion estimation task. The tasks of feature extraction and feature correspondence

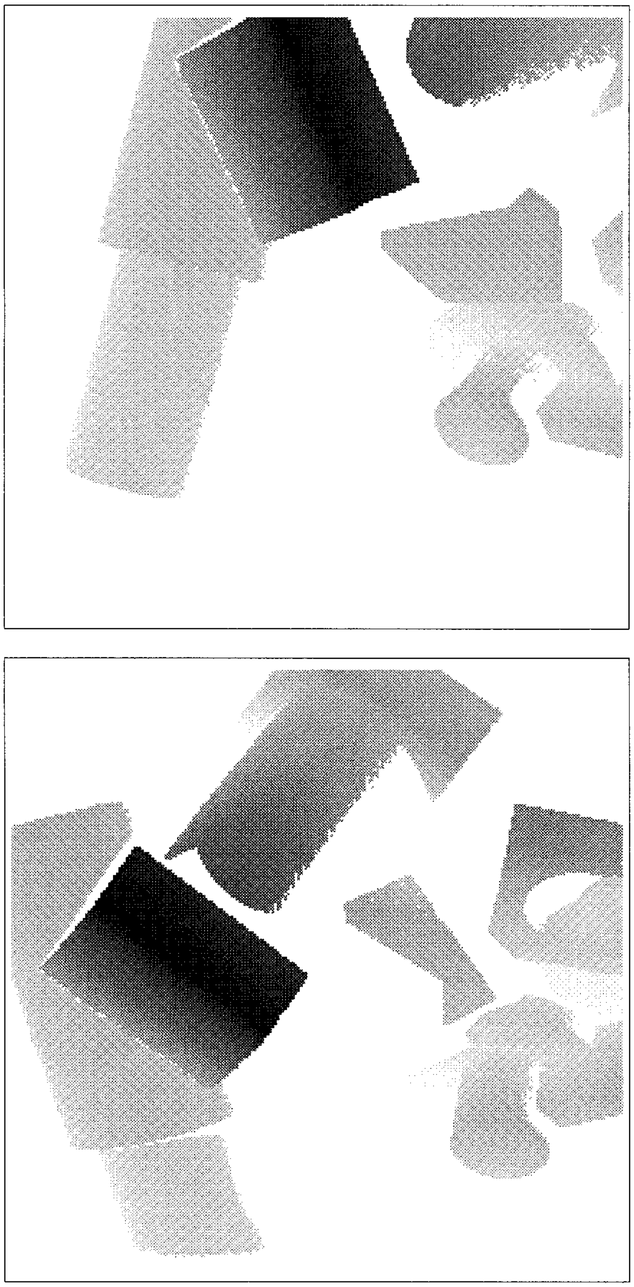


**FIG. 9.** Example 2, the depth maps of a sequence of range images.

**TABLE 3**
**Example 2: Surface Pairings Computed by the Hypergraph Isomorphism Algorithm**

| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| II | *h* | *o* | *q* | ★ | *v* | *n* | *m* | *p* | *u* | *r* | *t* | *w* | *x* |
| I | ★ | ★ | ★ | ★ | ★ | ★ | ★ | ★ | ★ | ★ | ★ | ★ | ★ |
| II | *a* | *b* | *c* | *d* | *e* | *f* | *g* | *i* | *j* | *k* | *l* | *s* | *v* |

**TABLE 4**
**Example 2: Rotation and Translation Estimates**

| Rotation | Actual | Estimated | Translation | Actual | Estimated |
|---|---|---|---|---|---|
| $\theta_x$ | 0° | 0.368° | $X$ | −5.0 cm | −6.485 cm |
| $\theta_y$ | 0° | −0.503° | $Y$ | 5.0 cm | 4.115 cm |
| $\theta_z$ | −30.0° | −30.125° | $Z$ | 0.0 cm | 0.069 cm |

*Note.* The images cover a range of 30.0 cm in the *x* and *y* directions.

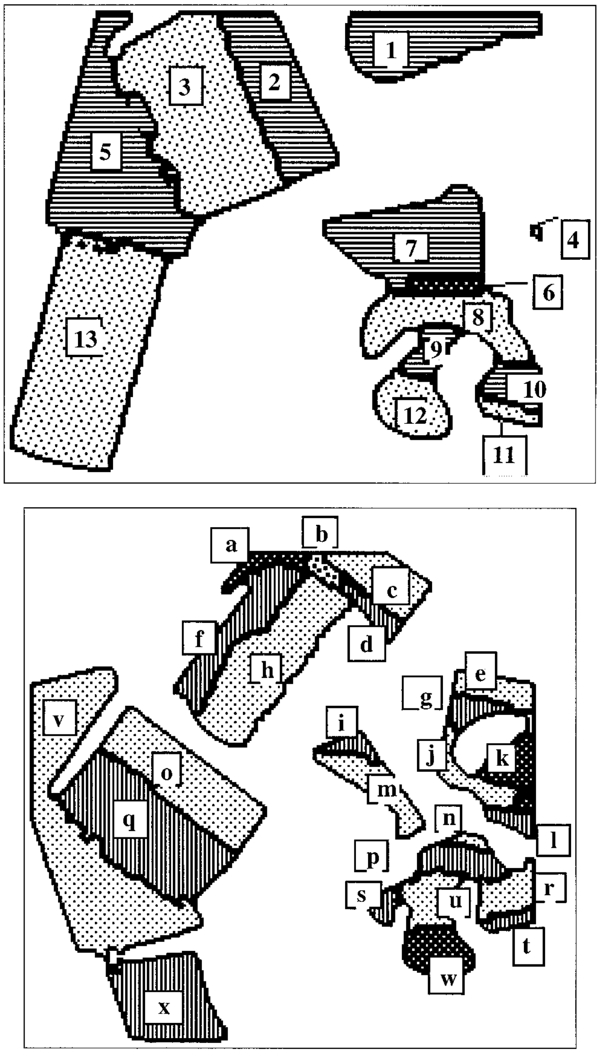


**FIG. 10.** Example 2, the segmented range images.

also become simpler when surfaces are used instead of local features.

In the first stage, segmentation, the range images are processed by using the pyramidal segmentation scheme developed in [23]. The partitioned images are used to compute the surface features in each range image. The next task, establishing correspondence between the image features in different frames of the sequence, is very different from finding correspondence between a model and an object description. In model-to-scene correspondence problems the model provides a complete description of the object, while in scene-to-scene correspondence problems, such as the case of a sequence of range images, both descriptions of the scene are incomplete.

This paper presents a new approach to compute the correspondences between surface segments in a sequence of range images. Fundamental to the approach is a hypergraph representation of the range images. The hierarchical representation of hypergraphs significantly reduces the search space and facilitates the encoding of topological and geometrical information. In addition to the topological and geometrical information obtained from the scene, the search process also uses *a priori* knowledge of the scene obtained from the physics of the laser scanning process used to produce the range images. Using this additional information reduces the compelxity of the matching procedure by pruning the search space. The solution is robust and accounts for errors in segmentation, occlusions of surfaces, and noise in the data. By using the topological information to guide the search procedure, the average case complexity of the algorithm is reduced significantly.

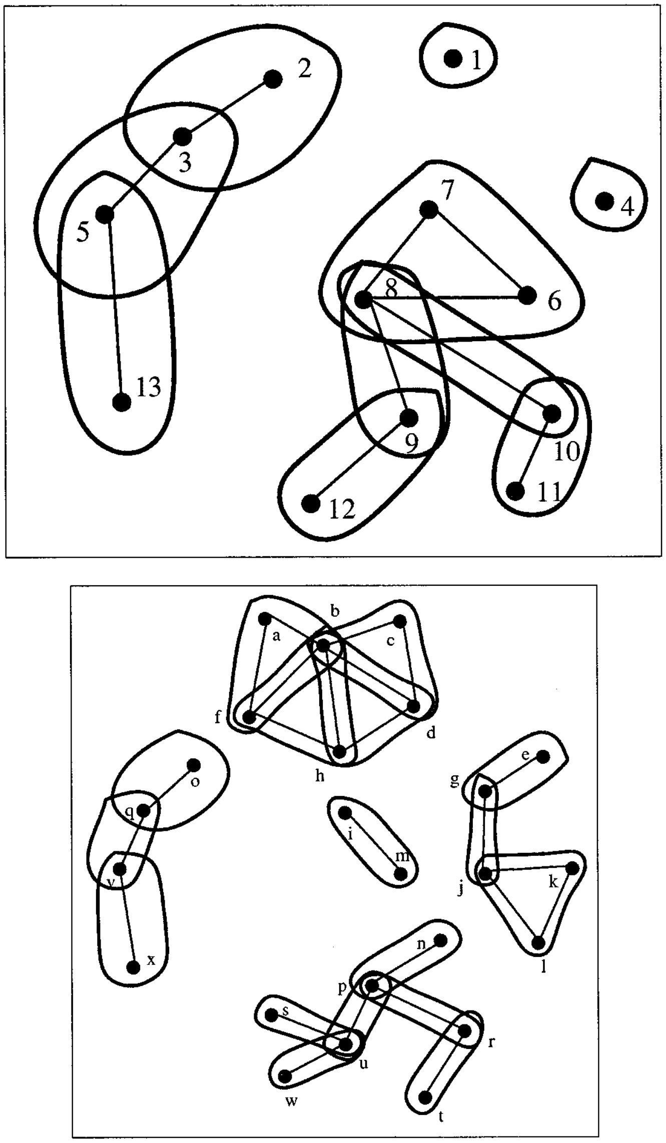


**FIG. 11.** Example 2, the hypergraphs generated from the graphs of the range images.

Using surfaces makes the task of computing the motion transform more difficult. By using a higher order feature, the constraint equations and the transformation equations become extremely complex. Many procedures that compute motion transformation omit the constraints of rigid body motion. However, if the transform does not satisfy these constraints, then the transform cannot be realized physically. This makes the incorporation of the constraints an important issue in the motion computation. Therefore, motion computation is formulated as a constrained minimization problem under the constraints of rigid body motion. A contrained minimization problem is formulated for the case of planar surface correspondences and is solved to give the rigid motion transformation. For the case of quadric surfaces, the minimization formulation is very complex, and there is no guarantee of a unique and optimal solution. The solution method developed extracts the linear features of the quadric from the quadric representation and uses the linear features to compute the motion transformation.

The contributions of the presented work are twofold. First, a method that solves the interframe correspondence problem using surface properties is presented. Second, a framework to compute the motion transformation using surface parameters is developed. Such methods are more reliable and robust with respect to noise and occlusions. It is to be noted that in our approach we do not extract the linear features from the data directly, so the procedure is less sensitive to noise in the individual points. The critical problem here is to fit the best quadric to the given data points. The surface fitting problem is an important problem that is currently under investigation. Least-square fitting of data to the model is not the best solution to the surface fitting problem. We are investigating minimum description length formulations that may lead to consistent fitting of the data to a model. Feedback from the motion estimates to the surface fitting algorithm will also help stabilize the fitting procedure. In the future work, we are also investigating alternative formulations for the best motion transform estimate. Least-square formulations are very suceptible to gross statistical outliers which are common occurences because of the nonuniqueness of the surface fitting procedure.

## ACKNOWLEDGMENTS

This research was supported in part by the Army Research Office under Contract DAAL 03 91-G-0050 and by Joint Services Electronic Program under Contract F49 620-92-C-0027. We thank Ms. Debi Paxton for proofreading the paper and her valuable comments on improving the understanding of the paper.

## REFERENCES

1. J. K. Aggarwal and N. Nandhakumar, On the computation of motion from sequences of images—A review, *Proc. IEEE* **76**(8), Aug. 1988, 917–935.
2. J. Aloimonos and I. Rigoutsos, Determining the 3-D motion of a rigid planar patch without correspondence, under perspective projection, in *Proceedings, IEEE Computer Society Workshop on Motion: Representation and Analysis, May 1986,* pp. 167–174.
3. F. Arman, *CAD-Based Object Recognition In 3-D Range Images Using Pre-Compiled Recognition Strategy Programs,* Ph.D. thesis, The University of Texas at Austin, Department of Electrical and Computer Engineering, 1992.
4. F. Arman and J. K. Aggarwal, Model-based object recognition in dense range images—A review, *ACM Comput. Surv.* **25**(1), Mar. 1993, 5–43.
5. F. Arman and J. K. Aggrawal, Cad-based vision: Object recognition in cluttered range images using recognition strategies, *Comput. Vision Graphics Image Process. Image Understanding* **58**(1), July 1993, 33–48.
6. K. S. Arun, T. S. Huang, and S. D. Blostein, Least squares fitting of two 3-D point sets, *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-9**(5), Sept. 1987, 698–700.
7. C. Berge, *Graphs and Hypergraphs,* North-Holland, Amsterdam, 1983.
8. P. J. Besl and R. C. Jain, Three-dimensional object recognition, *ACM Comput. Surv.* **17**(1), Mar. 1985, 75–145.
9. N. Deo, *Graph Theory with Applications to Engineering and Computer Science,* Prentice Hall, Englewood Cliffs, NJ, 1974.
10. O. D. Faugeras and M. Hebert, The representation, recognition, and locating of 3-D objects, *Int. J. Robotics Res.* **5**(3), Fall 1986.
11. G. Gati, Further annotated bibliography on the isomorphism disease, *J. Graph Theory* **3,** 1979, 95–109.
12. W. E. L. Grimson and T. Lozeno-Perez, Localizing overlapping parts by searching the interpretation tree, *IEEE Trans. Pattern Anal. Machine Intell.* **9**(4), Apr. 1987, 469–482.
13. W. E. L. Grimson, The combinatorics of object recognition in cluttered environments using constrained search, *Artif. Intell.* **44,** 1990, 121–165.
14. W. E. L. Grimson, The combinatorics of heuristic search termination for object recognition in cluttered environments, *IEEE Trans. Pattern Anal. Machine Intell.* **13**(9), Sept. 1991, 920–935.
15. W. E. L. Grimson and D. P. Huttenlocher, On the verification of hypothesized matches in model-based recognition, *IEEE Trans. Pattern Anal. Machine Intell.* **13**(12), Dec. 1991, 1201–1213.
16. B. K. P. Horn, Closed-form solution of absolute orientation using unit quaternions, *J. Optical Soc. Am.* **4**(4), Apr. 1987, 629–642.
17. B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, Closed-form solution of absolute orientation using orthonormal matrices, *J. Optical Soc. Am.* **5**(7), July 1988, 1127–1135.
18. T. S. Huang, Ed., *Image Sequence Analysis,* Springer Verlag, New York, 1981.
19. T. S. Huang and S. D. Blostein, Robust algorithms for motion estimation based on two sequential stereo image pairs, in *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, June 1985,* pp. 518–523.
20. V. Kumar, Algorithms for constraint satisfaction problems: A survey, *AI Magazine,* Spring 1992, 32–44.
21. R. C. Read and D. G. Corneil, The graph isomorphism disease, *J. Graph Theory* **1,** 1977, 339–363.
22. B. Sabata and J. K. Aggarwal, Estimation of motion from a pair of range images: A review, *Comput. Vision Graphics Image Process. Image Understanding* **54**(3), Nov. 1991, 309–324.
23. B. Sabata, F. Arman, and J. K. Aggarwal, Segmentation of range images using pyramidal data structures, in *Proceeding, International*

*Conference in Computer Vision, Osaka, Japan, Dec. 1990,* pp. 662–666.

24. V. Snyder and C. H. Sisam, *Analytic Geometry of Space,* Holt, New York, 1914.
25. S. Ullman, The interpretation of structure from motion, *Proc. R. Soc. London* **B203,** 1979, 405–426.
26. S. Ullman, *The Interpretation of Visual Motion,* MIT Press, Cambridge, MA, 1979.
27. W. E. Snyder, Guest Ed., Special issue on motion and time varying imagery, *Comput. Vision Graphics Image Process.* **21**(1), January 1983.
28. A. K. C. Wong, Knowledge representation for robot vision and path planning using attributed graphs and hypergraphs, in *Machine Intelligence and Knowledge Engineering for Robotic Applications* (A. K. C. Wong and A. Pugh, Eds.), 113–143, Springer-Verlag, Berlin/Heidelberg, 1987.
29. A. K. C. Wong, S. W. Lu, and M. Rioux, Recognition and shape synthesis of 3-D objects based on attributed hypergraphs, *IEEE Trans. Pattern Anal. Machine Intell.* **11**(3), Mar. 1989, 279–290.