International Journal of Computer Vision, 13:2, 119–152 (1994)

# Iterative Point Matching for Registration of Free-Form Curves and Surfaces

ZHENGYOU ZHANG zzhang@sophia.inria.fr
*INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, F-06902 Sophia-Antipolis Cedex-FRANCE*

Received June 23, 1992. Revised March 4, 1993.

**Abstract**
A heuristic method has been developed for registering two sets of 3-D curves obtained by using an edge-based stereo system, or two dense 3-D maps obtained by using a correlation-based stereo system. Geometric matching in general is a difficult unsolved problem in computer vision. Fortunately, in many practical applications, some a priori knowledge exists which considerably simplifies the problem. In visual navigation, for example, the motion between successive positions is usually approximately known. From this initial estimate, our algorithm computes observer motion with very good precision, which is required for environment modeling (e.g., building a Digital Elevation Map). Objects are represented by a set of 3-D points, which are considered as the samples of a surface. No constraint is imposed on the form of the objects. The proposed algorithm is based on iteratively matching points in one set to the closest points in the other. A statistical method based on the distance distribution is used to deal with outliers, occlusion, appearance and disappearance, which allows us to do subset-subset matching. A least-squares technique is used to estimate 3-D motion from the point correspondences, which reduces the average distance between points in the two sets. Both synthetic and real data have been used to test the algorithm, and the results show that it is efficient and robust, and yields an accurate motion estimate.

## 1 Introduction

The work described in this paper was carried out in the context of autonomous vehicle navigation in rugged terrain based on vision. A single view is usually not sufficient for path planning and manipulation, and it is preferable to combine several views to produce a more credible interpretation. The objective of this work is to compute precisely the displacement of the vehicle between successive views in order to register different 3-D visual maps. A 3-D visual map can be a set of curves obtained by using either an edge-based stereovision system (Pollard et al. 1985; Robert and Faugeras 1991) or a range imaging sensor (Sampson 1987). It can also be a dense 3-D map either acquired by an active sensor (e.g., ERIM (Sampson 1987)), or reconstructed by a correlation-based stereovision system (Fua 1992), or obtained by fusing the two. The reader is referred to (Faugeras et al. 1992) for a quantitative and qualitative comparison of some area and feature-based stereo algorithms. The registration step is indispensable for the following reasons:

- better localize the mobile vehicle,
- eliminate errors introduced in stereo matching and reconstruction,
- build a more global Digital Elevation Map (DEM) of the environment.

Geometric matching remains one of the bottlenecks in computer and robot vision, although progress has been made in recent years for some particular applications. There are two main applications: object recognition and visual navigation. The problem in object recognition is to match observed data to a prestored model representing different objects of interest. The problem in visual navigation is to match data observed in a dynamic scene at different instants in order to recover object motions and to interpret the scene. Registration for inspection/validation is also an important application of geometric matching (Menq et al. 1992). Besl and Jain (1985), and Chin and Dyer (1986) have made two excellent surveys of pre-1985 work on matching in object recognition. Besl (1988) surveys the current methods for geometric matching and geometric representations while emphasizing the latter. Most of the previous work focused on polyhedral objects; geometric primitives such as points, lines and planar patches were usually used. This is of course very limited compared with the real world we live in. Recently, curved objects have attracted the attention of many researchers in computer vision. This paper deals with objects represented by free-form curves and surfaces, i.e., arbitrary space shapes of the type found in practice.

A free-form curve can be represented by a set of chained points. Several matching techniques for free-form curves have been proposed in the literature. In the first category of techniques, curvature extrema are detected and then used in matching (Bolles and Cain 1982). However, it is difficult to localize precisely curvature extrema (Walters 1987; Milios 1989), especially when the curves are smooth. Very small variations in the curves can change the number of curvature extrema and their positions on the curves. Thus, matching based on curvature extrema is highly sensitive to noise. In the second category, a curve is transformed into a sequence of local, rotationally and translationally invariant features (e.g., curvature and torsion). The curve matching problem is then reduced to a 1-D string matching problem (Pavlidis 1980; Schwartz and Sharir 1987; Wolfson 1990; Gueziec and Ayache 1992). As more information is used, the methods in this category tend to be more robust than those in the first category. However, these methods are still subject to noise disturbance because they use arclength sampling of the curves to obtain point sets. The arclength itself is sensitive to noise.

A dense 3-D map is a set of 3-D points. We can divide the methods proposed in the literature for registering two dense 3-D maps in two categories (the reader is referred to Zhang (1991) for a more detailed review):

- **Primitive-based approach.** A set of primitives are first extracted. A dense 3-D map can then be described by a graph with primitives defining the nodes and geometric relations defining the links. The registration of two maps becomes the mapping of the two graphs: *subgraph isomorphism.* Some heuristics are usually introduced to reduce the complexity.
- **Surface-based approach.** A 3-D map is considered as a surface, having the form (a Monge patch)

$$\mathbf{x}(x, y) = [x,\ y,\ z(x, y)]^T \quad \text{with } (x, y) \in \mathbb{R}^2.$$

The idea is to find the transformation by minimizing a criterion relating the distance between the two surfaces.

In the primitive-based approach, one often uses some differential properties invariant to rigid transformation such as Gaussian curvature. The primitives often used are

1. special points (Goldgof et al. 1988; Hebert et al. 1989; Kweon and Kanade 1992), whose curvature is locally maximal and is bigger than a threshold.
2. contours. A contour can indicate where the elevation changes significantly, which is called a *cliff* in Rodríguez and Aggarwal (1989). It can also be a *distance profile* (Radack and Badler 1989), each point on which has the same distance to a common point. In certain specific cases, a contour can be a curve of a constant depth (Kamgar-Parsi et al. 1991).
3. surface patches (Kehtarnavaz and Mohan 1989; Liang and Todhunter 1990). Each surface patch is classified into different categories according to the sign of the Gaussian and mean curvatures. This type of primitives is usually used in a limited scene, for example, a scene containing several objects to be recognized. In a natural scene, there will be many surface patches such that the mapping becomes impractical.

Among the surface-based methods, we find

1. a technique similar to the correlation (Gennery 1989), applicable when the number of degrees of freedom of the transformation between two maps is reduced (2, for example).
2. a differential technique (Horn and Harris 1991), applicable when the motion between two views is very small or when we have a very good initial estimate of the motion, and when the data are not very noisy.
3. a technique based on the coherence and compatibility between two maps (Hebert et al. 1989; Kweon and Kanade 1992) (quantified by the distance between two surfaces). Szeliski (1988) proposed a similar technique by adding a smoothness constraint.

The main difference between the above two approaches resides in the information to be processed during the registration. The information used in the primitive-based approach is much more concise than in the surface-based approach, and is in general preferable. But in a natural environment, with the state of the art of the current methods, we cannot detect robustly and localize precisely primitives (Walters 1987; Milios 1989). The surface-based approach uses all available information. The large redundancy allows for a precise computation of the transformation between the two maps, but this approach usually requires some a priori knowledge of the transformation.

The primitive-based approach and the curve matching methods cited above exploit global matching criteria in the sense that they can deal with two sets of free-form curves and surfaces which differ by a large motion/transformation. This ability to deal with large motions is usually essential for applications to object recognition. In many other applications, for example, visual navigation, the motion between curves in successive frames is in general either small (because the maximum velocity of an object is limited and the sample frequency is high) or known within a reasonable precision (because a mobile vehicle is usually equipped with several instruments such as odometric and inertial systems which can provide such information). In the latter case, we can first apply the rough estimate of the motion to the first frame to produce an intermediate frame; then the motion between the intermediate frame and the second frame can be considered to be small. A surface-based method is attractive for such applications.

This paper describes a method, similar to the third technique of the surface-based approach but much faster, to register two 3-D maps differing by a small motion. The key idea underlying our approach is the following. Given that the motion between two successive frames is small, a point in the first frame is close to the corresponding point in the second frame. By matching points in the first frame to their closest points in the second, we can find a motion that brings the two sets of points closer. Iteratively applying this procedure, the algorithm yields a better and better motion estimate.

Recently, several pieces of independent work exploiting the similar ideas have been published. They are Besl and McKay (1992); Chen and Medioni (1992); Menq et al. (1992); Champleboux et al. (1992). A detailed comparison between these methods and ours will be given in Section 8.

## 2 Problem Statement

A parametric 3-D (space) curve segment $\mathcal{C}$ is a vector function $\mathbf{x} : [a, b] \rightarrow \mathbb{R}^3$, where $a$ and $b$ are scalar. In computer vision applications, the data of a space curve are usually available in the form of a set of chained 3-D points. If we know the type of the curve, we can obtain its description $\mathbf{x}$ by fitting, say, conics to the point data (Safaee-Rad et al. 1991; Taubin 1991).

A parametric surface $\mathcal{S}$ is a vector function $\mathbf{x} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$. In computer vision applications, the data of a surface are usually available in the form of a set of 3-D points. If we know the type of the surface, we can obtain its description $\mathbf{x}$ by fitting, say, planes or quadratic surfaces to the point data (Faugeras and Hebert 1986; Taubin 1991).

In this work, we shall use directly the chained points for curves and point sets for surfaces, i.e., we are interested in free-form shapes without regard to particular primitives. This is very appropriate for a non-structured environment. In the following, if not explicitly stated, the property that a curve is a set of *chained* points is not used, i.e., we shall treat curve data in the same way as surface data (a set of points). The word *shape* ($S$) will refer to either curves or surfaces. The points in the first 3-D map are noted by $\mathbf{x}_i$ $(i = 1, \ldots, m)$, and those in the second map are noted by $\mathbf{x}'_j$ $(j = 1, \ldots, n)$. These points are sampled from $S$ and $S'$, where $S = \mathcal{C}$ when curves are in consideration and $S = \mathcal{S}$ when surfaces are in consideration.

In the noise-free case, if $S$ and $S'$ are registered by a transformation $T$, then the distance of a point on $S$, after applying $T$, to $S'$ is zero, and the distance of a point on $S'$, after applying the inverse of $T$, to $S$ should be zero, too. The objective of registration is to find the motion between the two frames, i.e., $\mathbf{R}$ for rotation and $\mathbf{t}$ for translation, such that the following criterion

$$\mathcal{F}(\mathbf{R}, \mathbf{t}) = \frac{1}{\sum_{i=1}^{m} p_i} \sum_{i=1}^{m} p_i \, d^2(\mathbf{R}\mathbf{x}_i + \mathbf{t}, S') + \frac{1}{\sum_{j=1}^{n} q_j} \sum_{j=1}^{n} q_j \, d^2(\mathbf{R}^T\mathbf{x}'_j - \mathbf{R}^T\mathbf{t}, S) \quad (1)$$

is minimized, where $d(\mathbf{x}, S)$ denotes the distance of the point $\mathbf{x}$ to $S$ (to be defined below), and $p_i$ (resp. $q_j$) takes value 1 if the point $\mathbf{x}_i$ (resp. $\mathbf{x}'_j$) can be matched to a point on $S'$ in the second frame (resp. $S$ in the first frame) and takes value 0 otherwise. The minimum of $\mathcal{F}(\mathbf{R}, \mathbf{t})$ will be zero in the noise-free case. It is necessary to have the parameters $p_i$ and $q_j$ because some points are only visible from one point of view and some are outliers, as to be described in Section 8.

The above criteria are symmetric in the sense that neither of the two frames prevails over the other. To economize computation, we shall only use the first part of the right hand side of Equation 1. In other words, the objective function to be minimized is

$$\mathcal{F}(\mathbf{R}, \mathbf{t}) = \frac{1}{\sum_{i=1}^{m} p_i} \sum_{i=1}^{m} p_i \, d^2(\mathbf{R}\mathbf{x}_i + \mathbf{t}, S'). \tag{2}$$

The effect of this simplification is described in Section 7.3.

However, the minimization of $\mathcal{F}(\mathbf{R}, \mathbf{t})$ is very difficult not only because $d(\mathbf{R}\mathbf{x}_i + \mathbf{t}, S')$ is highly nonlinear (the corresponding point of $\mathbf{x}_i$ on $S'$ is not known beforehand) but also because $p_i$ can take either 0 or 1 (an *Integer Programming Problem*). As said in the introduction, we follow a heuristic approach by assuming the motion between the two frames is small or approximately known. In the latter case, we can first apply the approximate estimate of the motion between the two frames to the first one to produce an intermediate frame; then the motion between the intermediate frame and the second frame can be considered to be small. *Small* depends essentially on the scene of interest. If the scene is dominated by a repetitive pattern, the motion should not be bigger than half of the pattern distance. For example, in the situation illustrated in Figure 1, our algorithm will converge to a local minimum. In this case, other methods based on more global criteria, such as those cited in the introduction section, could be used to recover a rough estimate of the motion. The algorithm described in this paper can then be used to obtain a precise motion estimate.

## 3 Iterative Pseudo Point Matching Algorithm

We describe in this section an iterative algorithm for 3-D shape registration by matching points in the first frame, after applying the previously recovered motion estimate $(\mathbf{R}, \mathbf{t})$, with their closest points in the second. A least-squares estimation reduces the average distance between the matched points in the two frames. As a point in one frame and its closest point in the other do not necessarily correspond to a single point in space, several iterations are indispensable. Hence the name of the algorithm.

### *3.1 Finding Closest Points*

Let us first define the distance $d(\mathbf{x}, S')$ between point $\mathbf{x}$ and shape $S'$, which is used in Equation 2. By definition, we have

$$d(\mathbf{x}, S') = \min_{\mathbf{x}' \in S'} d(\mathbf{x}, \mathbf{x}'), \tag{3}$$

where $d(\mathbf{x}_1, \mathbf{x}_2)$ is the Euclidean distance between the two points $\mathbf{x}_1$ and $\mathbf{x}_2$, i.e., $d(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|$. In our case, $S'$ is available as a set of points $\mathbf{x}'_j$ ($j = 1, \ldots, n$). We use the following simplification:

$$d(\mathbf{x}, S') = \min_{j \in \{1, \ldots, n\}} d(\mathbf{x}, \mathbf{x}'_j). \tag{4}$$

See Section 7.4 for more discussions on the distance.

The closest point $\mathbf{y}$ in the second frame to a given point $\mathbf{x}$ is the one satisfying

$$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) \,, \quad \forall \mathbf{z} \in S'.$$

The worst case cost of finding the closest point is $O(n)$, where $n$ is the number of points in the second frame. The total cost while performing the above computation for each point in the first frame is $O(mn)$, where $m$ is the number of points in the first frame. There are several methods which can considerably speed up the search process, including bucketing techniques and $k$-D trees (abbreviation for *k-dimensional binary search tree*) (Preparata and Shamos 1986). $k$-D trees are implemented in our algorithm, see Appendix A of this article for the details.

### *3.2 Pseudo Point Matching*

For each point $\mathbf{x}$ we can always find a closest point $\mathbf{y}$. However, because there are some spurious points in both frames due to sensor error, or because some points visible in one frame are not in the other due to sensor or object motion, it probably does not make any sense to pair $\mathbf{x}$ with $\mathbf{y}$. Many constraints can be imposed to remove such spurious pairings. For example, distance continuity in a neighborhood, which is similar to the figural continuity in stereo

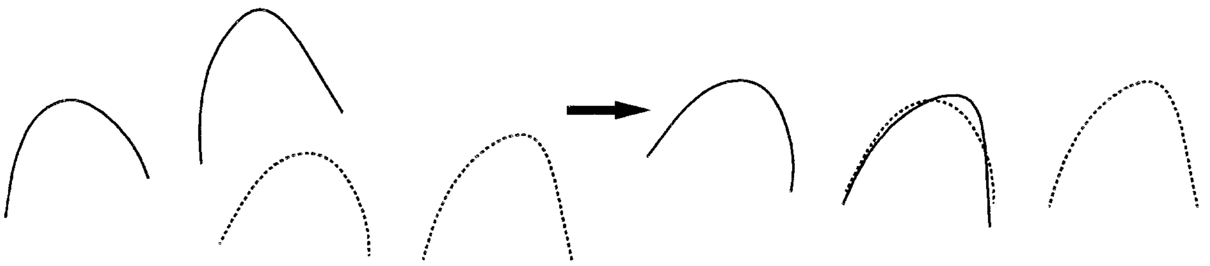


*Fig. 1.* Our algorithm exploits a local matching technique, and converges to the closest local minimum, which is not necessarily the optimal one

matching (Mayhew and Frisby 1981; Pollard et al. 1985; Grimson 1985), should be very useful to discard the false matches. These constraints are not incorporated in our algorithm in order to maintain the algorithm in its simplest form. Instead, we can exploit the following two simple heuristics, which are all unary.

The first is the maximum tolerance for distance. If the distance between a point $\mathbf{x}_i$ and its closest one $\mathbf{y}_i$, denoted by $d(\mathbf{x}_i, \mathbf{y}_i)$, is bigger than the maximum tolerable distance $D_{\max}$, then we set $p_i = 0$ in Equation 2, i.e., we cannot pair a reasonable point in the second frame with the point $\mathbf{x}_i$. This constraint is easily justified since we know that the motion between the two frames is small and hence the distance between two points reasonably paired cannot be very big. In our algorithm, $D_{\max}$ is set adaptively and in a robust manner during each iteration by analyzing distances statistics. See Section 3.3.

The second is the orientation consistency. We can estimate the surface normal or the curve tangent (both referred below as orientation vector) at each point. It can be easily shown that the angle between the orientation vector at point $\mathbf{x}$ and that at its corresponding point $\mathbf{y}$ in the second frame can not go beyond the rotation angle between the two frames (Zhang et al. 1988). Therefore, we can impose that the angle between the orientation vectors at two paired points should not be bigger than a prefixed value $\Theta$, which is the maximum of the rotation angle expected between the two frames. This constraint is not implemented for surface registration, because the computation of the surface normals from 3-D scattered points is relatively expensive. For curves, we compute an approximate tangent for each point from the vector linking its neighboring points (Zhang 1992b; Zhang 1992a). It is the only place where the property that points of a curve are chained is used. In our implementation, we set $\Theta = 60^\circ$ to take into account noise effect in the tangent computation. If the tangents can be precisely computed, $\Theta$ can be set to a smaller value. This constraint is especially useful when the motion is relatively big.

### *3.3 Updating the Matching*

Instead of using all matches recovered so far, we exploit a robust technique to discard several of them by analyzing the statistics of the distances. The basic idea is that the distances between reasonably paired points should not be very different from each other. To this end, one parameter, denoted by $\mathcal{D}$, needs to be set by the user, which indicates when the registration between two frames is good. See Section 4.1 for the choice of the value $\mathcal{D}$.

Let $D^{\mathrm{I}}_{\max}$ denote the maximum tolerable distance in iteration I. At this point, each point in the first frame (after applying the previously recovered motion) whose distance to its closest point is less than $D^{\mathrm{I}-1}_{\max}$ is retained, together with its closest point and their distance. Let $\{\mathbf{x}_i\}$, $\{\mathbf{y}_i\}$, and $\{d_i\}$ be, respectively, the resulting sets of original points, closest points, and their distances after the pseudo point matching, and let $N$ be the cardinal of the sets. Now compute the mean $\mu$ and the sample deviation $\sigma$ of the distances, which are given by

$$\mu = \frac{1}{N}\sum_{i=1}^{N} d_i ,$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - \mu)^2} .$$

Depending on the value of $\mu$, we adaptively set the maximum tolerable distance $D^{\mathrm{I}}_{\max}$ as shown below:[1]

```
if = $\mu < \mathcal{D}$
     /* the registration is quite good */
$D^{\mathrm{I}}_{\max} = \mu + 3\sigma$;
elseif $\mu < 3\mathcal{D}$
     /* the registration is still good */
$D^{\mathrm{I}}_{\max} = \mu + 2\sigma$;
elseif $\mu < 6\mathcal{D}$
    /* the registration is not too bad */
$D^{\mathrm{I}}_{\max} = \mu + \sigma$;
else
     /* the registration is really bad */
$D^{\mathrm{I}}_{\max} = \xi$;
endif
```

The explanation of $\xi$ is deferred to Section 4.2.

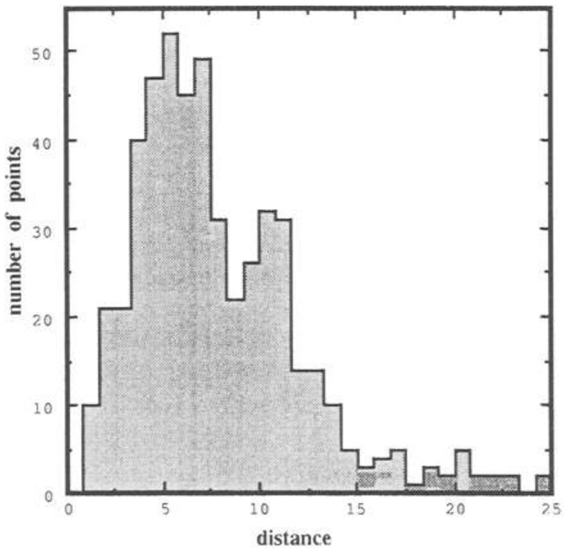


*Fig. 2.* A histogram of distances

At this point, we use the newly set $D^{\mathrm{I}}_{\max}$ to update the matching previously recovered: a paring between $\mathbf{x}_i$ and $\mathbf{y}_i$ is removed if their distance $d_i$ is bigger than $D^{\mathrm{I}}_{\max}$. The remaining pairings are used to compute the motion between the two frames, as to be described below.

Because $D_{\max}$ is adaptively set based on the statistics of the distances, our algorithm is rather robust to relatively big motion and to gross outliers (as to be shown in the experiment section). Even if there remain several false matches in the retained set after update, the use of least-squares technique still yields a reasonable motion estimate, which is sufficient for the algorithm to converge to the correct solution.

### 3.4 Computing Motion

At this point, we have a set of 3-D points which have been reasonably paired with a set of closest points, denoted respectively by $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_i\}$. Let $N$ be the number of pairs. Because $N$ is usually much greater than 3 (three points are the minimum for the computed rigid motion to be unique), it is necessary to devise a procedure for computing the motion by minimizing the following mean-squares objective function

$$\mathcal{F}(\mathbf{R}, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2, \qquad (5)$$

which is the direct result of Equation 2 with the definition of distance given by Equation 4. Any optimization method, such as steepest descent, conjugate gradient, or simplex, can be used to find the least-squares rotation and translation. Fortunately, several much more efficient algorithms exist for solving this particular problem. They include quaternion method (Faugeras and Hebert 1986; Horn 1987), singular value decomposition (Arun et al. 1987), dual number quaternion method (Walker et al. 1991), and the method proposed by Brockett (1989). We have implemented both the quaternion method and the dual number quaternion one. They yield exactly the same motion estimate. For completeness, the dual quaternion method (Walker et al. 1991) is summarized in Appendix B.

### 3.5 Summary

We can now summarize the iterative pseudo point matching algorithm as follows:

- **input:** Two 3-D frames containing $m$ and $n$ 3-D points, respectively.
- **output:** The optimal motion between the two frames.
- **procedure:**
  1. **initialization**
     $D^{0}_{\max}$ is set to $20\mathcal{D}$, which implies that every point in the first frame whose distance to its closest point in the second frame is bigger than $D^{0}_{\max}$ is discarded from considera-

tion during the first iteration. The number *20* is not crucial in the algorithm, and can be replaced by a larger one.

2. **preprocessing**
   (a) Compute the tangent at each point of the two frames (only for curves).
   (b) Build the $k$-D tree representation of the second frame.
3. **iteration** until convergence of the computed motion
   (a) Find the closest points satisfying the distance and orientation constraints, as described in Section 3.2.
   (b) Update the recovered matches through statistical analysis of distances, as described in Section 3.3.
   (c) Compute the motion between the two frames from the updated matches, as described in Section 3.4.
   (d) Apply the motion to all points (and their tangents for curves) in the first frame.

Several remarks should be made here. First, the construction and the use of $k$-D trees for finding closest points are explained in Appendix A. Second, the motion is computed between the original points in the first frame and the points in the second frame. Therefore, the final motion given by the algorithm represents the transformation between the *original* first frame and the second frame. Last, the iteration-termination condition is defined as the change in the motion estimate between two successive iterations. The change in translation at iteration I is defined as $\delta \mathbf{t} = \|\mathbf{t}_I - \mathbf{t}_{I-1}\| / \|\mathbf{t}_I\|$. To measure the change in rotation, we use the rotation axis representation, which is a 3-D vector, denoted by $\mathbf{r}$. Let $\theta = \|\mathbf{r}\|$ and $\mathbf{n} = \mathbf{r}/\|\mathbf{r}\|$, the relation between $\mathbf{r}$ and the quaternion $\mathbf{q}$ is $\mathbf{q} = [\sin(\theta/2)\mathbf{n}^T, \ \cos(\theta/2)]^T$. We do not use the quaternions because their difference does not make much sense. We then define the change in rotation at iteration I as $\delta \mathbf{r} = \|\mathbf{r}_I - \mathbf{r}_{I-1}\| / \|\mathbf{r}_I\|$. We terminate the iteration when both $\delta \mathbf{r}$ and $\delta \mathbf{t}$ are less than 1%, or when the number of iterations achieves a prefixed threshold (20 for curves and 40 for surfaces). One could also define the termination condition as the *absolute* change, i.e., $\delta \mathbf{r} = \|\mathbf{r}_I - \mathbf{r}_{I-1}\|$ and $\delta \mathbf{t} = \|\mathbf{t}_I - \mathbf{t}_{I-1}\|$. We stop the iteration if $\delta \mathbf{r}$ is less than a threshold, say 0.5 degrees, and $\delta \mathbf{t}$ is less than a threshold, say 0.5 centimeters.

## 4 Practical Considerations

In this section, we consider several important aspects in practice, including choice of the parameters $\mathcal{D}$ and $\xi$, and coarse-to-fine strategy.

### *4.1 Choice of the Parameter $\mathcal{D}$*

The only parameter needed to be supplied by the user is $\mathcal{D}$, which indicates when the registration between two frames can be considered to be good. In other words, the value of $\mathcal{D}$ should correspond to the expected average distance when the registration is good. When the motion is big, $\mathcal{D}$ should not be very small. Because we set $D^0_{\max} = 20\mathcal{D}$, if $\mathcal{D}$ is very small we cannot find any matches in the first iteration and of course we cannot improve the motion estimate. (A solution to this is to set $D^0_{\max}$ bigger, say $30\mathcal{D}$). In practice, if we know the precision of the initial estimate, say, within 20 centimeters, we can set $D^0_{\max}$ to that value.

The value of $\mathcal{D}$ has an impact on the convergence of the algorithm. If $\mathcal{D}$ is smaller than necessary, then more iterations are required for the algorithm to converge because many good matches will be discarded at the step of matching update. On the other hand, if $\mathcal{D}$ is much bigger than necessary, it is possible for the algorithm not to converge to the correct solution because possibly many false matches will not be discarded. Thus, to be prudent, it is better to choose a small value for $\mathcal{D}$.

In our implementation, we relate $\mathcal{D}$ to the resolution of the data. Let $\bar{D}$ be the average distance between neighboring points in the second frame. Consider a perfect registration shown in Figure 3. Points from the first frame are marked by a cross and those from the second, by a dot. Assume that a cross is located in the middle of two dots. Then in this case, the mean $\mu$ of the distances between two sets of points is equal to $\bar{D}/2$. In general, we can expect $\mu > \bar{D}/2$. So, if $\bar{D}$ is computed, we can set $\mathcal{D} = \bar{D}$. For curves, we do compute $\bar{D}$ for each run. For surfaces, $\mathcal{D}$ is set to 10 centimeters, which corresponds roughly twice the resolution of a 3-D map reconstructed by a correlation-based stereo for a depth range of about 10 meters. This gives us satisfactory results.

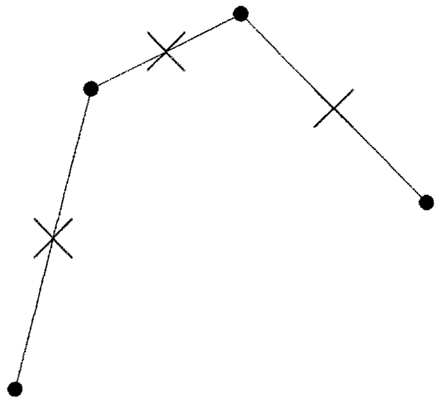


*Fig. 3.* Illustration of a perfect registration to show how to choose $\mathcal{D}$

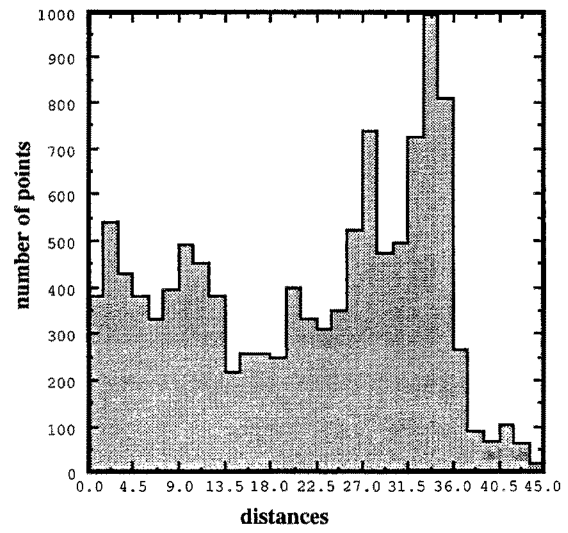


*Fig. 4.* Histogram of distances when the initial estimate of the motion is very bad

### 4.2 Choice of the Parameter $\xi$

In Section 3.3, we described how to update matches through a statistical analysis of distances, and we have assumed that the distribution of distances is approximately Gaussian when the registration between two frames is good. Because of the local property of the matching criterion used, our algorithm converges to the closest minimum. It is thus best applied in situations where the motion is small or approximately known and a precise estimate of the motion is required. In the case of a very bad initial estimate of the motion between two frames, one observes that the form of the distribution of distances is in general very complex. We show in Figure 4 one such typical histogram.

As can be observed, the form of the histogram in Figure 4 is irregular. There are several peaks. Furthermore, many points are found near zero. This shows the difficulty of our approach. When the initial estimate is very bad, we probably find matches having small distances due to occasionally bad alignments, that is, these matches are in fact not reasonable. We cannot guarantee that our algorithm yields the correct estimate of the motion. One possible method is to generate a hypothesis for each peak. And then evaluate each hypothesis in parallel. The criterion for measuring the quality of a hypothesis can be a function of the number of matches and of the final average distance. In the end, the hypothesis which gives the best score is retained as the transformation between the two frames.

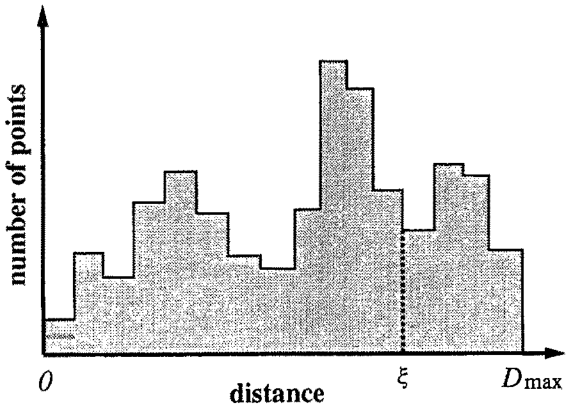


*Fig. 5.* How to choose the value of $\xi$

We have adopted a simpler method. The maximal peak gives in general, at least we expect, a hint of a reasonable correspondence between the two frames. We have chosen in our implementation the valley after the maximal peak as the value of $\xi$ (see Figure 5). That is, all matches after the valley are discarded from consideration. To avoid the noise perturbation, we impose that the number of points at the valley must not go beyond 60% of the number of points at the peak. In all our experiments, this method provides us with sufficient results, as to be shown below.

### *4.3 Coarse-to-Fine Strategy*

As to be shown in the next section, we find fast convergence of the algorithm during the first few iterations that slows down as it approaches the local minimum. We find also that more search time is required during the first few iterations because the search space is larger at the beginning. Since the total search time is linear in the number of points in the first frame, it is natural to exploit a coarse-to-fine strategy. During the first few iterations, we can use coarser samples (e.g., every five) instead of all sample points on the curve. When the algorithm almost converges, we use all available points in order to obtain a precise estimate.

## 5 Experimental Results with Curves

The proposed algorithm has been implemented in C. In order to maintain the modularity, the code is not optimized. The program is run on a SUN 4/60 workstation,[2] and any quoted times are given for execution on that machine.

This section is divided into three subsections. In the first the algorithm is applied to synthetic data. The results show clearly the typical behavior of the algorithm to be expected in practice. The second describes the robustness and efficiency of the algorithm using synthetic data with different levels of noise and different samplings. The third describes the experimental results with real data.

### *5.1 A Case Study*

In this experiment, the parametric curve described by $\mathbf{x}(u) = [u^2, 5u\sin(u) + 10u\cos(1.5u), 0]^T$ is used. The curve is sampled twice in *different* ways. Each sample set contains 200 points. The second set is then rotated and translated with $\mathbf{r} = [0.02, 0.25, -0.15]^T$ and $\mathbf{t} = [40.0, 120.0, -50.0]^T$. We thus get two noise-free frames. (The same noise-free data are used in the experiments described in the next section.)

For each point, zero-mean Gaussian noise with a standard deviation equal to 2 is added to its $x$, $y$ and $z$ components. We show in Figure 6 the front and top views of the noisy data. For visual convenience, points are linked. The solid curve is the one in the first frame, and the dashed one, in the second frame. The data are used as is; no smoothing is performed.

The first step is then to find matches for the points in the first frame. As $D^0_{\max}$ is big, each point has a match. We find 200 matches in total, which are shown in Figure 7, where matched points are linked. Many false matches are observed. We then update these matches using the technique described in Section 3.3, and 100 matches survive, which are shown in Figure 8.

Even after the updating, there are still some false matches. Because there are more good matches then false matches, the motion estimation algorithm still yields a reasonable estimate. This can be observed in Figure 9, where the motion estimated has been applied to the points in the first frame. We can observe the improvement of the registration of the two curves, especially in the top view.

Now we enter the second iteration. We find at this time 176 matches, which are shown in Figure 10a. (Top view is not shown, because the two curves are very close.) Several false matches are observable. After updating, 146 matches remain, as shown in Figure 10b. Almost all these matches are correct. Motion is then computed from these matches.

We iterate the process in the same manner. The motion result after 10 iterations is shown in Figure 11. The registration between the two curves is already quite good.

The algorithm yields after 15 iterations the following motion estimate:

$$\hat{\mathbf{r}} = [2.442 \times 10^{-2}, 2.503 \times 10^{-1}, -1.484 \times 10^{-1}]^T,$$

$$\hat{\mathbf{T}} = [3.879 \times 10^1, 1.139 \times 10^2, -4.967 \times 10^1]^T.$$

To measure the precision in the motion estimate, we define the rotation error as

$$e_r = \|\mathbf{r} - \hat{\mathbf{r}}\|/\|\mathbf{r}\| \times 100\%, \tag{6}$$

where $\mathbf{r}$ and $\hat{\mathbf{r}}$ are respectively the real and estimated rotation parameters, and the translation error as

$$e_t = \|\mathbf{t} - \hat{\mathbf{t}}\|/\|\mathbf{t}\| \times 100\%, \tag{7}$$

where $\mathbf{t}$ is the real translation parameter and $\hat{\mathbf{t}}$ is the estimated one. In Figure 12, we show the evolution of the rotation and translation errors versus the number of iterations. Fast convergence is observed during the first few iterations and relatively slower later. After 15 iterations, the rotation error is 1.6% and the translation error is 4.6%.

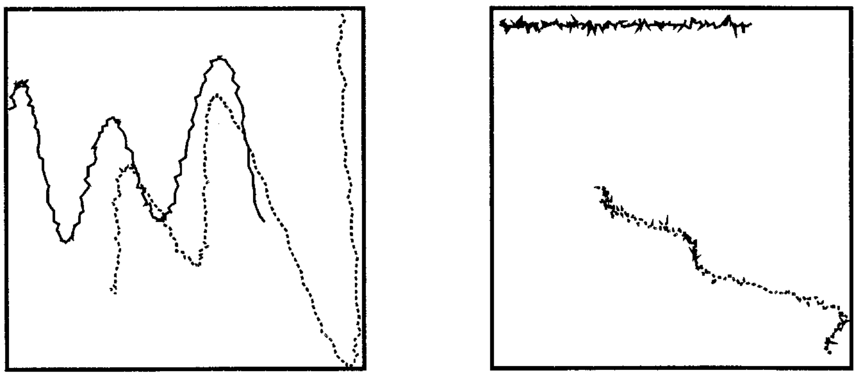


*Fig. 6.* Front and top views of the data

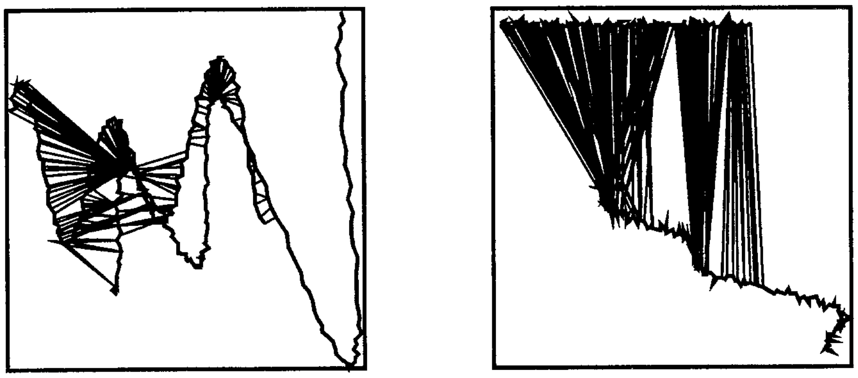


*Fig. 7.* Matched points in the first iteration before updating (front and top views)

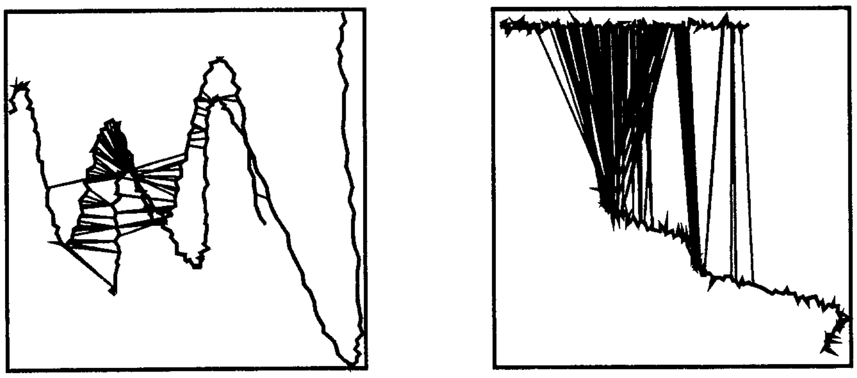


*Fig. 8.* Matched points in the first iteration after updating (front and top views)

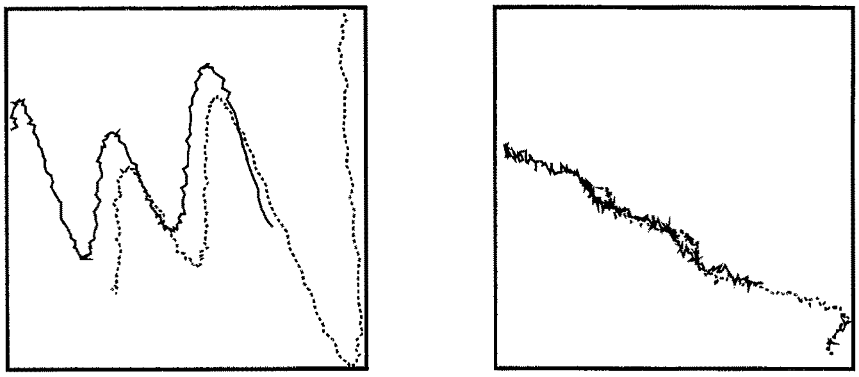
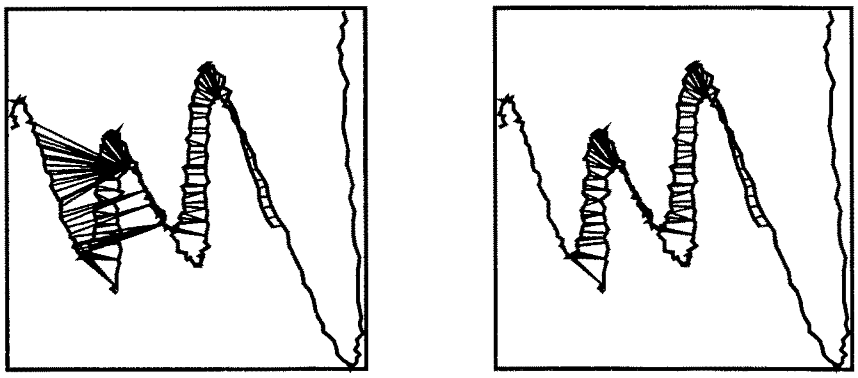


*Fig. 9.* Front and top views of the motion result after the first iteration



(a) (b)

*Fig. 10.* Matched points before and after updating in the second iteration (only the front view)

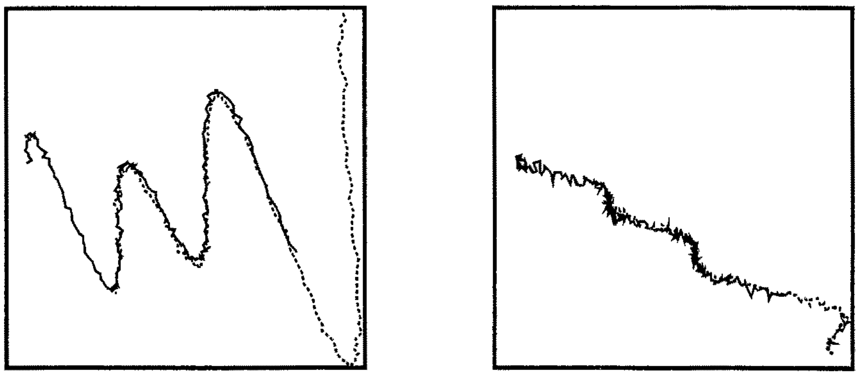


*Fig. 11.* Front and top views of the motion result after ten iterations

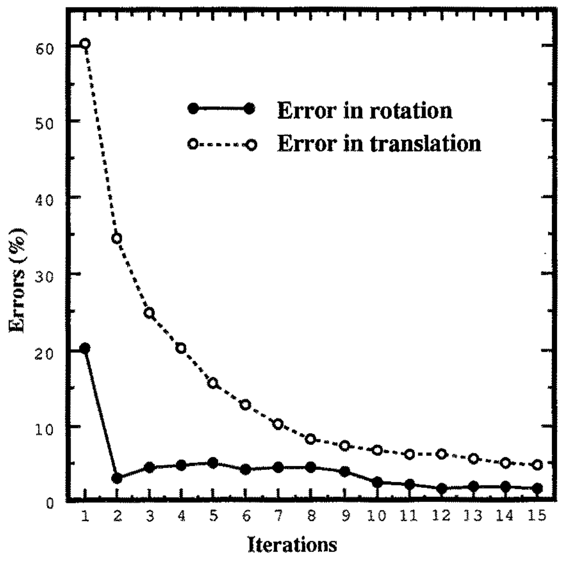


*Fig. 12.* Evolution of the rotation and translation errors versus the number of iterations

We show in Table 1 several intermediate results during different iterations. The results are divided into three parts. The second to fourth rows indicate the execution time (in seconds) required for finding matches, updating the matching, and computing the motion, respectively. The fifth row shows the values of $D_{\max}$ used in different iterations. The last row shows the comparison of the numbers of matches found in different iterations before and after updating. We have the following remarks:

- $D_{\max}$ almost decreases monotonically with the number of iterations. This is because the registration becomes better and better, and $D_{\max}$ is computed dynamically through the statistical analysis of distances.
- The time required for finding matches almost decreases monotonically, too. This is because of the almost monotonic decrease of $D_{\max}$. Less search in $k$-D tree is required when the search region becomes smaller.
- The time required for updating the matching is negligible.
- The time required for computing the motion is almost constant, as it is related to the number of matches (here almost constant). Furthermore, the motion algorithm is very efficient: about 0.05 seconds for 145 matches.
- The numbers of matches before and after updating do not vary much after the first few iterations. This also implies that the Gaussian assumption of the distance distribution is reasonable.

The total execution time is 6.5 seconds in this experiment.

### 5.2 Synthetic Data

In this section, we describe the robustness and efficiency of the algorithm using the same synthetic data as in the last section, but with different levels of noise and different samplings. All results given below are the average of ten tries.

The first series of experiments are carried out with respect to different levels of noise. The standard deviation of the noise added to each point varies from 0 to 20. Similar to Figure 12, we show, as a sample, in Figure 13 and Figure 14 the evolutions of the rotation and translation errors versus the number of iterations with a standard deviation equal to 2 and 8. From these results, we observe that

- The translation error decreases almost monotonically, while the behavior of the rotation error is more complex.
- Noise has a stronger impact on the rotation parameters than on the translation parameters. When noise is small, there is in general a smaller error in rotation than in translation. When noise is significant, the inverse is observed.

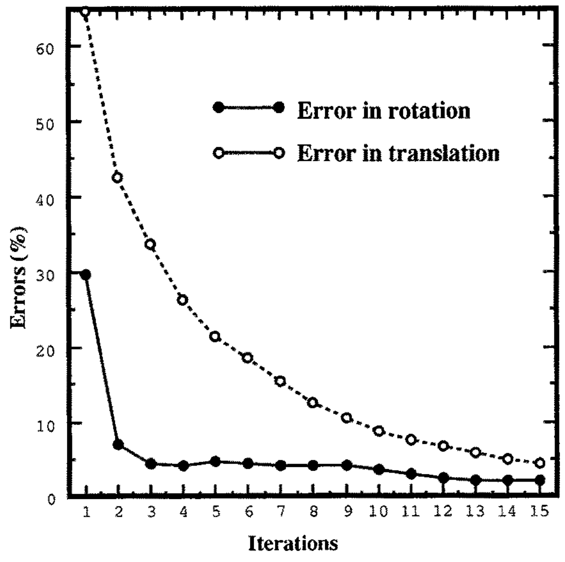


*Fig. 13.* Evolution of the rotation and translation errors versus the number of iterations with a standard deviation equal to 2

*Table 1.* Several detailed results in different iterations

| iteration | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| matching time | 2.20 | 1.30 | 0.62 | 0.33 | 0.25 | 0.28 | 0.22 | 0.17 | 0.15 | 0.17 | 0.15 | 0.12 | 0.13 | 0.13 | 0.12 |
| update time | 0.03 | 0.02 | 0.02 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.02 | 0.00 | 0.00 | 0.02 | 0.00 | 0.02 |
| motion time | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.03 | 0.05 | 0.03 | 0.02 | 0.07 | 0.05 | 0.03 | 0.03 | 0.02 | 0.02 |
| $D_{\max}$ | 235 | 140 | 78.5 | 46.1 | 32.8 | 34.7 | 28.0 | 22.4 | 18.9 | 16.7 | 15.3 | 13.6 | 12.3 | 10.7 | 9.89 |
| nb. $\frac{\text{before}}{\text{after}}$ | $\frac{200}{100}$ | $\frac{176}{146}$ | $\frac{150}{143}$ | $\frac{148}{137}$ | $\frac{147}{147}$ | $\frac{148}{148}$ | $\frac{148}{147}$ | $\frac{148}{147}$ | $\frac{148}{146}$ | $\frac{148}{146}$ | $\frac{147}{147}$ | $\frac{146}{145}$ | $\frac{143}{143}$ | $\frac{143}{143}$ | $\frac{143}{143}$ |

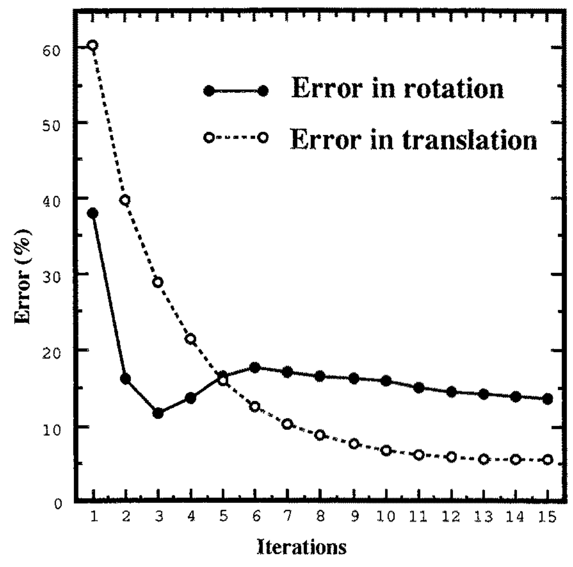


*Fig. 14.* Evolution of the rotation and translation errors versus the number of iterations with a standard deviation equal to 8

We think the above phenomena are due to the fact that the relation between the measurements and the rotation parameters is nonlinear while that between the measurements and the translation parameters is linear.

To visually demonstrate the effect of the noise added and the ability of the algorithm, we show in Figure 15 and Figure 16 two sample results. In each figure, the upper row displays the front and top views of the two noisy curves before registration; the lower row displays the front and top views of the two noisy curves after registration. In Figure 15 and Figure 16, we have added, to each $x$, $y$, and $z$ components of each point of the two curves, zero-mean Gaussian noise with a standard deviation equal to 8 and 16, respectively. Even though the curves are so noisy, the registration between them is surprisingly good.

We now summarize more results in Table 2. The rotation and translation errors are measured in percents, and the execution time, in seconds. Each number shown is the average of 10 tries. 15 iterations have been applied. We have the following conclusions:

- The errors in rotation and in translation increase with the increase in the noise added to the data, as expected.
- Noise in the measurements has more effect in the rotation than in translation.
- The algorithm is robust to noise. It yields a reasonable motion estimate even when the data are significantly corrupted.
- The execution time increases also with the increase in the noise added to the data. This is because when the data are very noisy the value of $D_{\max}$ stays big, and the search has to be performed in a large space.

We now investigate the ability of the algorithm with respect to different samplings of curves. The same data are used. Zero-mean Gaussian noise with a standard deviation equal to 2 is added to each $x$, $y$, and $z$ components of each point of the two curves. We will describe in Section 7.4 the effect of different samplings of the curves in the second frames. Here we vary the sampling of the curve in the first frame from 1 (i.e., all points) to 10 (i.e., one out of every ten points). Ten tries are carried out for each sampling. The errors in rotation and in translation (in percents), and the execution time (in seconds) versus different samplings are shown in Table 3. Two remarks can be made:

- Generally speaking, the more samples there are in a curve, the less the error in the estimation of the rotation and translation. However, the exact relation is not very clear. Consider sampling = 1 and sampling = 10. The latter has only 20 points

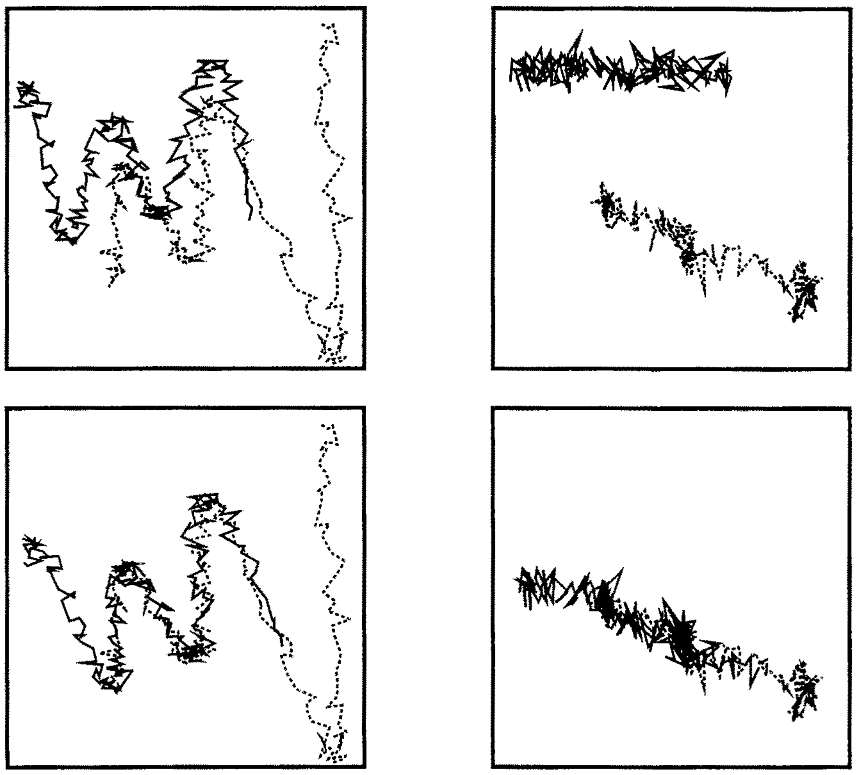


*Fig. 15.* Front and top views of two noisy curves with a standard deviation equal to 8 before and after registration

*Table 2.* A summary of the experimental results with synthetic data

| standard deviation | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| rotation error | 2.25 | 2.12 | 4.63 | 9.62 | 13.73 | 14.31 | 20.47 | 18.07 | 23.87 | 37.04 | 33.20 |
| translation error | 1.77 | 4.36 | 4.55 | 4.84 | 5.70 | 7.81 | 8.93 | 9.89 | 17.15 | 22.00 | 27.17 |
| execution time | 6.27 | 6.82 | 8.58 | 9.26 | 11.12 | 11.86 | 12.59 | 13.35 | 16.40 | 16.56 | 17.32 |

while the former has 200 points. The motion error, however, is only twice as large.

- The execution time decreases monotonically as the number of sample points decreases. If disregarding the preprocessing time, the execution time is linear in the number of points in the first frame.

In the foregoing discussions we have observed that using coarsely sampled points in the curves in the first frame does not affect too much the accuracy of the final motion estimate, but it considerably speeds up the whole process. It is natural to think about using a coarse-to-fine strategy such as that described in Section 4.3. The finding of fast convergence of the algorithm during the first few iterations (see Figure 13 and Figure 14) and the finding of relatively expensive search (see Table 1) justify the following strategy. During the first few iterations, we use coarser, instead of all, sample points, which allows for finding an estimate close to the optimal. We then use all sample points to refine this estimate. We have conducted ten experiments using the same data as before by adding zero-mean Gaussian noise with a standard deviation equal to 3. During the first five iterations, only 40 points (one out of every five points) are used. These are followed by ten iterations using all points. The average results of the ten experiments are: rotation

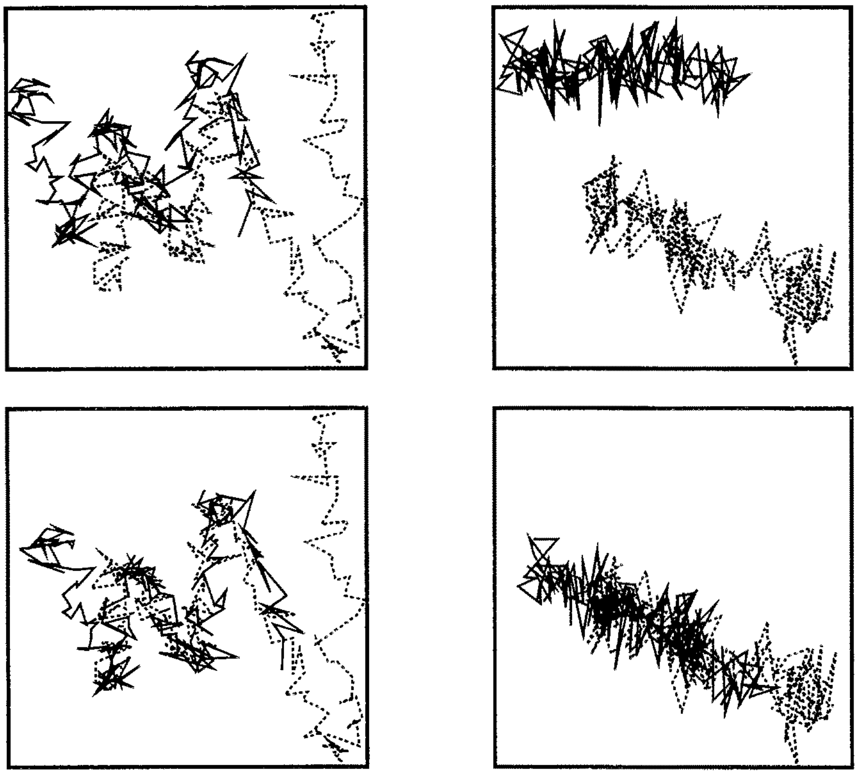


*Fig. 16.* Front and top views of two noisy curves with a standard deviation equal to 16 before and after registration

*Table 3.* Results with respect to different samplings

| fraction of points | 1 | 1/2 | 1/3 | 1/4 | 1/5 | 1/6 | 1/7 | 1/8 | 1/9 | 1/10 |
|---|---|---|---|---|---|---|---|---|---|---|
| rotation error | 2.12 | 3.44 | 4.19 | 4.88 | 4.09 | 7.52 | 4.75 | 6.09 | 5.98 | 4.90 |
| translation error | 4.36 | 5.14 | 4.27 | 4.75 | 4.11 | 6.67 | 8.54 | 7.45 | 8.52 | 7.34 |
| execution time | 6.82 | 3.53 | 2.41 | 1.85 | 1.52 | 1.28 | 1.11 | 1.01 | 0.89 | 0.83 |

error = 4.56%, translation error = 4.29%, and execution time = 3.39 s. For comparison, we performed 15 iterations using all points. The average results of the ten tries are: rotation error = 4.68%, translation error = 4.14%, and execution time = 7.49 s. Only little difference between the final motion estimates is observed, but the algorithm is more than twice faster by exploiting the coarse-to-fine strategy.

### 5.3 Real Data

In this section, we provide an example with real data. A trinocular stereo system mounted on our mobile vehicle is used to take images of a chair scene (the scene is static but the robot moves). We show in Figure 17 two images taken by the first camera from two different positions. The displacement between the two positions is about 4 degrees in rotation and 100 millimeters in translation. The chair is about 3 meters from the mobile vehicle.

The curve-based trinocular stereo algorithm developed in our laboratory (Robert and Faugeras 1991) is used to reconstruct the 3-D frames corresponding to the two positions. There are 36 curves and 588 points in the first frame, and 48 curves and 763 points in the second frame. We show in the upper row of Figure 18

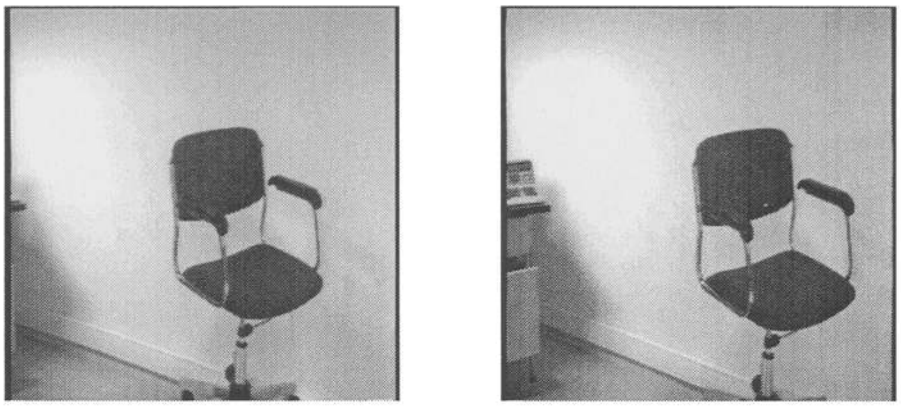


*Fig. 17.* Images of a chair scene taken by the first camera from two different positions

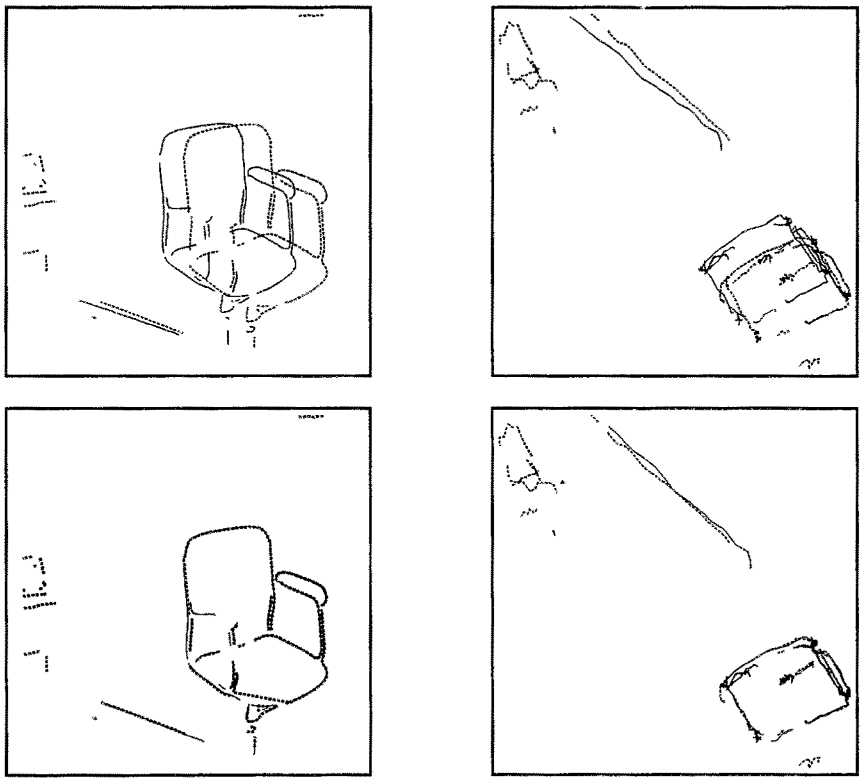


*Fig. 18.* Superposition of two 3-D frames before and after registration: front and top views

the front view and the top view of the superposition of the two 3-D frames. The curves in the first frame is displayed in solid lines while those in the second frames, in dashed lines. We apply the algorithm to the two frames. The algorithm converges after 12 iterations. It takes in total 32.5 seconds on a SUN 4/60 workstation and half of the time is spent in the first iteration (so we could speed up the process by setting $D^0_{\max}$ to a smaller value). The final motion estimate is

$$\hat{\mathbf{r}} = [-1.527 \times 10^{-3},\ 6.639 \times 10^{-2},\ 2.894 \times 10^{-3}]^T,$$

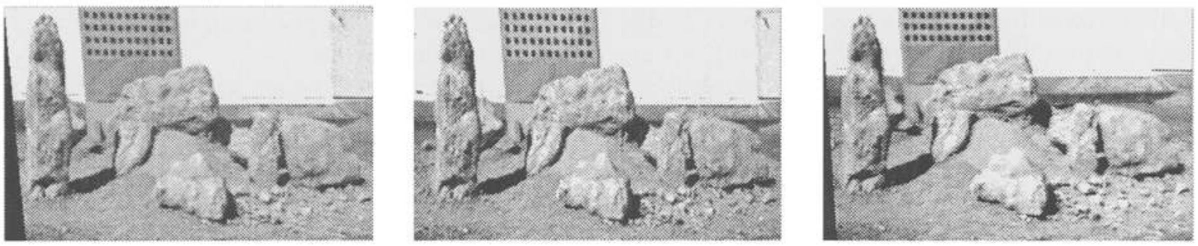


*Fig. 19.* The first triplet of images of a rock scene

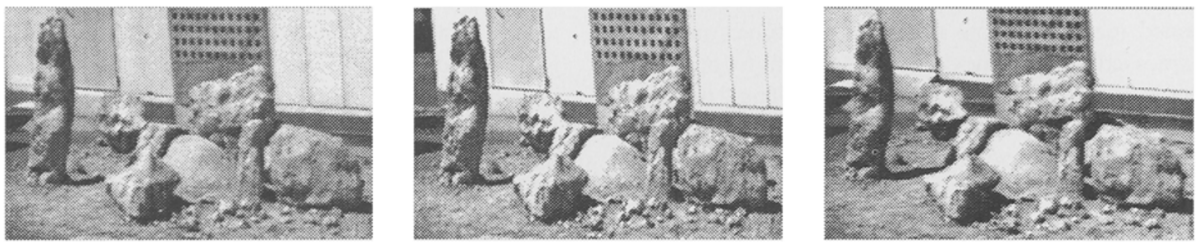


*Fig. 20.* The second triplet of images of a rock scene

$$\hat{\mathbf{t}} = [-4.266 \times 10^0,\ -1.586 \times 10^0,\ -1.009 \times 10^2]^T,$$

where $\mathbf{r}$ is in radians and $\mathbf{t}$ is in centimeters. The motion change is: $\delta\mathbf{r} = 0.78\%$ and $\delta\mathbf{t} = 0.53\%$. The result is shown in the lower row of Figure 18 where we have applied the estimated motion to the first frame. Excellent registration is observed for the chair. The registration of the border of the wall is a little bit worse because more error has been introduced during the 3-D reconstruction, for it is far away from the cameras.

Now we exploit the coarse-to-fine strategy. As before, we do coarse matching in the first five iterations by sampling evenly one out of every five points on the curves in the first frame, followed by fine matching using all points. The algorithm converges after 12 iterations and yields exactly the same motion estimate as when only doing fine matching. The execution time, however, decreases from 32.5 seconds to 10.5 seconds, about three times faster. If we now sample evenly one out of every *ten* points on the curves in the first frame, and do coarse matching in the first five iterations and fine matching in the subsequent ones, the algorithm converges after 13 iterations (one iteration more), and the final motion estimate is

$$\hat{\mathbf{r}} = [-1.438 \times 10^{-3},\ 6.653 \times 10^{-2},\ 2.995 \times 10^{-3}]^T,$$

$$\hat{\mathbf{t}} = [-4.282 \times 10^0,\ -1.637 \times 10^0,\ -1.007 \times 10^2]^T,$$

which is almost the same as the one estimated using directly all points. The motion change is: $\delta\mathbf{r} = 0.71\%$ and $\mathbf{t} = 0.50\%$. The execution time is now 8.8 seconds.

## 6 Experimental Results with Surfaces

We provide in this section two examples. In the first example, two 3-D frames of a rock scene are reconstructed by a correlation-based stereovision system. They are first registered manually. Then we want to see the limit of our algorithm by using different initial estimates. The second example shows the registration of two range images of a head figure.

### *6.1 A Rock Scene*

We show in Figure 19 and Figure 20 two triplets of images of a rock scene. The stereo rig is about 6 meters from the scene. The two positions differ by 30 degrees in rotation and 3.75 meters in translation.

The correlation-based stereo system reconstructs 71505 points for the first position and 51503 points for the second position. For experimental purpose, we have taken two similar triplets of images having marks put on the rocks. From these marks, we are able to manually compute the displacement between the two positions. This result is shown in Figure 21 (see color figure section), where the first map is drawn in quadrangles, and the second in grayed surface. The registration is reasonable. One can observe that many points are only visible from one position. In the sequel, we vary the initial motion estimate, run our algorithm on the two frames, and then compare the results obtained by our algorithm with the result obtained manually. Note that the two frames are now expressed in the same coordinate system by applying the manual estimate to the first frame. Thus, the final estimate of the displacement between the two frames is expected to be zero, and the estimate given by the algorithm is directly the motion error with respect to the manual estimate. We have done several tests, and three of them are shown in the following. The initial estimate will be represented by a 6-vector: the first three elements constitute the r vector, and the last three, the t vector.

If we set the initial estimate to $[0.0, 0.0, 0.35, 0.5, -2.0, 0.2]^T$ (i.e., a rotation of 20 degrees and a translation of 2.07 meters). The difference between the two frames corresponding to this estimate is shown in Figure 22 (see color figure section). After 40 iterations, the motion estimate given by our algorithm is $[7.885261 \times 10^{-3}, -1.208467 \times 10^{-2}, 4.158183\times 10^{-3}, 2.661822\times 10^{-2}, -1.230099\times 10^{-2}, -4.845936 \times 10^{-2}]^T$. Thus, there is a difference of 0.86 degrees in rotation and 5.66 centimeters in translation from the manual estimate. The result after the registration is shown in Figure 23 (see color figure section). We see that even when the initial estimate is very different from the real one, we still obtain satisfactory results. After several more iterations, we obtain a better result.

What happens if we increase further the difference between the initial and final estimates? The initial estimate in this test is $[0.0, 0.0, 0.35, -0.5, -2.5, 0.2]^T$ (i.e., a rotation of 20 degrees and a translation of 2.56 meters). The difference between the two frames corresponding to this estimate is shown in Figure 24 (see color figure section). After 40 iterations, the motion estimate given by our algorithm is $[-3.825570 \times 10^{-2}, 6.001669 \times 10^{-2}, 3.071064\times 10^{-1}, 3.449387\times 10^{-1}, -1.795149\times 10^{0}, 3.954597 \times 10^{-1}]^T$. The result is mediocre, as shown in Figure 25 (see color figure section), but it is better than the initial estimate.

If we continue, the result shows some further improvement. After 80 iterations, the motion estimate is $[1.012586 \times 10^{-2}, -9.563972 \times 10^{-3}, 8.020001 \times 10^{-3}, 3.110009 \times 10^{-2}, -3.634908 \times 10^{-2}, -3.905361 \times 10^{-2}]^T$. The difference with the manual estimate is of 0.92 degrees in rotation and 6.18 centimeters in translation, which is reasonably small, as shown in Figure 26 (see color figure section).

Up to now, the tests we have carried out are all with a rotation around an axis perpendicular to the ground plane. What happens if the vehicle is found in two different slopes (e.g., the vehicle scrambles on a pile of rocks)? Here is an example. The initial estimate is $[0.35, 0.0, 0.17, -0.5, -2.5, 0.2]^T$. Thus, there is a rotation of 20 degrees with respect to the ground plane, and a rotation of 10 degrees around an axis perpendicular to the ground plane. The translation between the two views is 2.56 meters. The difference between the two frames corresponding to this estimate is shown in Figure 27 (see color figure section). After 40 iterations, the motion estimate is $[1.125988 \times 10^{-2}, 4.676589 \times 10^{-4}, 1.893187\times 10^{-3}, 4.918958\times 10^{-3}, -2.754730\times 10^{-2}, 6.532630 \times 10^{-3}]^T$. The difference with the manual estimate is of 0.65 degrees in rotation and 2.87 centimeters in translation. The registration result is quite good, as shown in Figure 28 (see color figure section).

### *6.2 A Head Figure*

The proposed algorithm is used by Chen for range image registration (Chen 1992). One modification he has made is the closest point search procedure. He uses the technique proposed by Chen and Medioni (1992) (see Section 8). One example is shown in this section. Figure 29a and Figure 29d show two range images of a head figure. For display purpose, they are shown in shaded intensity image form. The coordinate system is defined as follows: The origin is in the center of the image, the $x$-axis is parallel to the columns of the images (unit in pixels), the $y$-axis is parallel to the rows (unit in pixels), and the $z$-axis points out of the paper (unit in grey levels). The two images differ by $[-0.1745, 0, 0, 0, 0, 0]^T$.

Instead of using all points, about 150 points on a regular grid are chosen from the first image, as shown

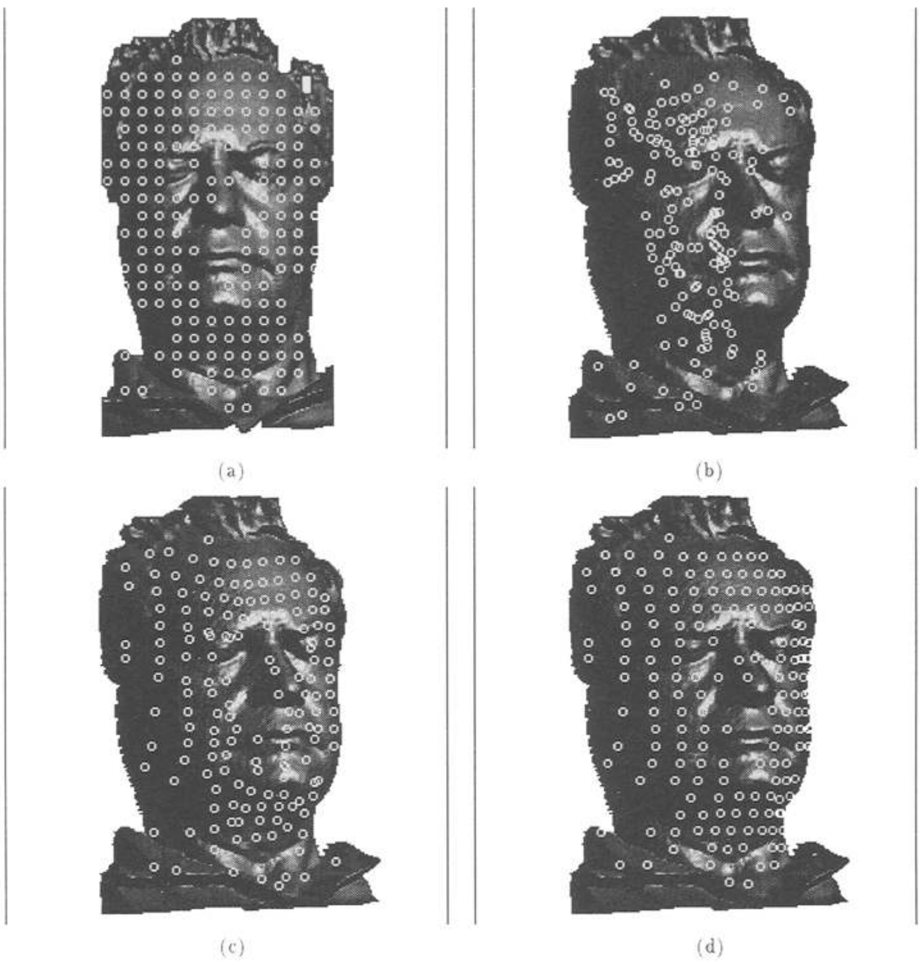


*Fig. 29.* A head figure. (a) First view; (b) Result after one iteration; (c) Result after five iterations; (d) Result after 39 iterations

in Figure 29a as small circles overlaid on the original image. The initial motion estimate is [0.0873, 0, 0, 0, 0, 0]$^T$, i.e., a difference of 15 degrees in rotation from the real transformation. The result after one iteration is shown in Figure 29b, where the points from the first image after transformation (in circles) are projected to the second image. The result is very bad. After five iterations, the estimated transformation is already reasonable, as shown in Figure 29c. The algorithm converges after 39 iterations, yielding a result as shown in Figure 29d. The corresponding motion estimate is [−0.1702, 0.0016, .0008, −0.0461, 0.1529, −0.1244]$^T$. The error is less than 0.3 degrees in rotation and 0.2 pixels in translation.

## 7 Discussions

### *7.1 About Complexity*

As described earlier, each iteration of our algorithm consists of four main steps. The first is to find closest points, at an expected cost of $O(m \log n)$, where $m$ and $n$ are the numbers of points in the first and second frames, respectively. The second is to update the matches recovered in the first step, at a cost of $O(m)$. The third step is to compute the 3-D motion, also at a cost of $O(m)$. The last step is to apply the estimated motion to all points in the first frame, at a cost of $O(m)$. Thus the total complexity of our algorithm is $O(m \log n)$. In Zhang (1992a), we show that our algorithm has a lower bound of computational cost than the string-based curve matching algorithms, e.g., Schwartz and Sharir (1987).

## *7.2 About Convergence*

Convergence is always an important issue of an iterative procedure. Our algorithm cannot be guaranteed to reach the global minimum. Although we have observed, given a reasonable start point, a good convergence of our algorithm in the experimental sections, we are not able to show that it converges monotonically to a local minimum. This is not like the algorithm of Besl and McKay (1992) (see Section 8) which converges always monotonically to a local minimum. The difference is that $p_i$ in our objective function (2) can take a value of either one or zero depending on situations. As will be clear, however, our algorithm is well-behaved.

Let us make a thorough investigation during each iteration. As described in Section 3.2, only points whose distances to their closest points in the second frame are less than $D^{\mathrm{I}-1}_{\max}$ are retained as potential matches. The mean squared error $d^{\mathrm{I}}_{\text{closest}}$ of these matches, given by

$$d^{\mathrm{I}}_{\text{closest}} = \frac{1}{\sum_{i=1}^{m} p_i} \sum_{i=1}^{m} p_i \, \|\mathbf{R}^{\mathrm{I}-1}\mathbf{x}_i + \mathbf{t}^{\mathrm{I}-1} - \mathbf{y}_i\|^2,$$

is upper-bounded by $D^{\mathrm{I}-1}_{\max}$, i.e., $d^{\mathrm{I}}_{\text{closest}} \le D^{\mathrm{I}-1}_{\max}$. Then a statistical analysis of distances is carried out, and a new distance threshold $D^{\mathrm{I}}_{\max}$ is computed. The pairings whose distances are greater than $D^{\mathrm{I}}_{\max}$ are discarded, i.e., their $p_i$'s are turned to be zero. Thus the mean squared error $d^{\mathrm{I}}_{\text{update}}$ of the updated matches, given by

$$d^{\mathrm{I}}_{\text{update}} = \frac{1}{\sum_{i=1}^{m} p'_i} \sum_{i=1}^{m} p'_i \, \|\mathbf{R}^{\mathrm{I}-1}\mathbf{x}_i + \mathbf{t}^{\mathrm{I}-1} - \mathbf{y}_i\|^2,$$

is less than $d^{\mathrm{I}}_{\text{closest}}$, i.e., $d^{\mathrm{I}}_{\text{update}} \le d^{\mathrm{I}}_{\text{closest}}$. We have of course $d^{\mathrm{I}}_{\text{update}} \le D^{\mathrm{I}}_{\max}$. The least-squares technique described in Section 3.4 is applied to the remaining matches, and a new motion estimate $(\mathbf{R}^{\mathrm{I}}, \mathbf{t}^{\mathrm{I}})$ is available. Let

$$d^{\mathrm{I}}_{\text{lsq}} = \frac{1}{\sum_{i=1}^{m} p'_i} \sum_{i=1}^{m} p'_i \, \|\mathbf{R}^{\mathrm{I}}\mathbf{x}_i + \mathbf{t}^{\mathrm{I}} - \mathbf{y}_i\|^2.$$

We have always $d^{\mathrm{I}}_{\text{lsq}} \le d^{\mathrm{I}}_{\text{update}}$, because if $d^{\mathrm{I}}_{\text{lsq}} > d^{\mathrm{I}}_{\text{update}}$, then the zero motion ($\mathbf{R}^{\mathrm{I}} = \mathbf{I}, \mathbf{t}^{\mathrm{I}} = 0$) would yield a smaller mean-squared error, which contradicts the hypothesis. Thus we have

$$d^{\mathrm{I}}_{\text{lsq}} \le d^{\mathrm{I}}_{\text{update}} \le \min(d^{\mathrm{I}}_{\text{closest}}, D^{\mathrm{I}}_{\max}), \quad \text{and} \quad d^{\mathrm{I}}_{\text{closest}} \le D^{\mathrm{I}-1}_{\max}.$$

Unfortunately, we do not have the inequality: $d^{\mathrm{I}+1}_{\text{closest}} \le d^{\mathrm{I}}_{\text{lsq}}$. Indeed, $d^{\mathrm{I}+1}_{\text{closest}}$ contains two parts. The first consists of $\mathbf{x}_i$'s which are also contained in $d^{\mathrm{I}}_{\text{lsq}}$. We can easily show that this part is always decreasing. The second part consists of $\mathbf{x}_i$'s which are not contained in $d^{\mathrm{I}}_{\text{lsq}}$, but whose distances to their closest points are less than $D^{\mathrm{I}}_{\max}$. The combination of the two parts is not necessarily less than $d^{\mathrm{I}}_{\text{lsq}}$.

As is clear from the above discussions, the objective function is upper-bounded by $D_{\max}$. As the registration becomes better and better, $D_{\max}$ in general becomes smaller and smaller, but it may be occasionally bigger. In order to ensure a monotonic decrease of $D_{\max}$, we must impose that $D^{\mathrm{I}}_{\max} \le D^{\mathrm{I}-1}_{\max}$ after computing $D^{\mathrm{I}}_{\max}$ as described in Section 3.3. We have done this and rerun the algorithm with the synthetic and real data presented in Section 5, and *exactly the same results* have been obtained. We have also rerun the algorithm with the data presented in Section 6. For the test 1, the estimate after 40 iterations is $[6.752922 \times 10^{-3}, -1.142817 \times 10^{-2}, 5.471850 \times 10^{-3}, 2.741838 \times 10^{-2}, -1.966461 \times 10^{-2}, -4.403366 \times 10^{-2}]^T$. There is a difference of 0.11 degrees in rotation and 0.86 centimeters in translation. For the test 2, we obtained the same motion estimate after 40 iterations. The estimate after 80 iterations is $[7.06317 \times 10^{-3}, -7.186778 \times 10^{-3}, 1.308734 \times 10^{-2}, 3.428337 \times 10^{-2}, -6.574115 \times 10^{-2}, -2.475497 \times 10^{-2}]^T$. There is a difference of 0.37 degrees in rotation and 3.28 centimeters in translation. For the test 3, the estimate after 40 iterations is $[1.390749 \times 10^{-2}, -2.727281 \times 10^{-3}, 1.089010 \times 10^{-3}, 3.807930 \times 10^{-3}, -2.346437 \times 10^{-2}, -1.019406 \times 10^{-2}]^T$. There is a difference of 0.24 degrees in rotation and 1.73 centimeters in translation. These differences are sufficiently small compared with the resolution of the data (about 5 centimeters).

In Figure 30, two graphs are shown. The first plots the evolution of the mean distance (i.e., the objective function) versus iteration number. The second plots the evolution of the number of matches after update versus iteration number (one example has already been given in the last row of Table 1). The data presented in Section 6 have been used. Note that there are two curves for test 2: "Test 2" for iterations from 1 to 40, and "Test 2bis" for iterations from 41 to 80. About one fourth (17749 points) of the points in the first view have been used. From Figure 30a, we see that the mean distance decreases towards 0.04 meters in all three tests. These curves confirm that our algorithm is well-behaved. As shown in Figure 30b, the number of matches varies continuously through the iterations and finally steadies towards 15800.

### *7.3 About Simplifications*

For computation consideration, we have made two simplifications. The first is that the non-symmetric matching criterion (2) is used, instead of the symmetric one (1). The second is that the approximate distance metric (4) is used, which will be discussed in Section 7.4.

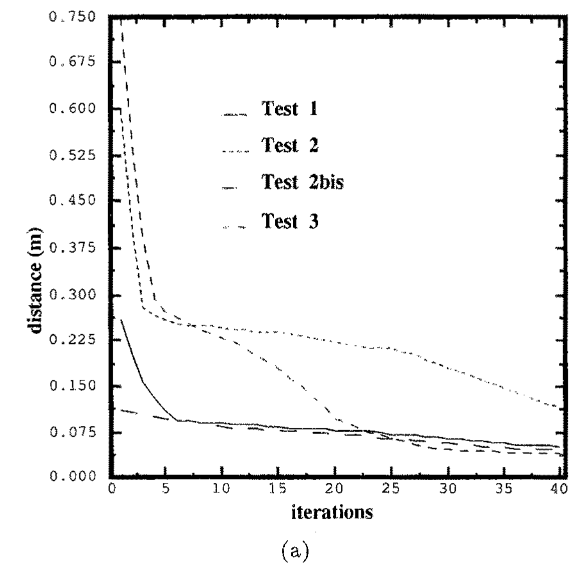
The symmetric matching criterion (1) is in fact also implemented for curves. Table 4 gives a comparison of the results using the two criteria. The synthetic data in Section 5 are used. Different levels of Gaussian noise are added. Ten iterations are applied in each case. Rotation errors, translation errors, and execution times are shown, each being the average of ten tries. The algorithm using the symmetric criterion yields better motion estimates than that using the non-symmetric one. This is expected because the data in the two frames both contribute to the motion estimation and neither of the frames prevails over the other. On the other hand, the execution time using the symmetric criterion is twice as long.

We have also carried out an experiment with the real data described in Section 5. The algorithm using the symmetric criterion converges after 12 iterations, yielding a motion estimate

$$\hat{\mathbf{r}} = [-1.706 \times 10^{-3},\ 6.673 \times 10^{-2},\ 2.774 \times 10^{-3}]^T,$$

$$\hat{\mathbf{t}} = [-4.294 \times 10^{0},\ -1.613 \times 10^{0},\ -1.003 \times 10^{2}]^T.$$

The difference compared with that using the non-symmetric criterion is 0.6% in both rotation and translation. The execution time is 70.2 seconds on a SUN 4/60 workstation, about twice as long. Thus in time critical applications the non-symmetric matching criterion is preferred.



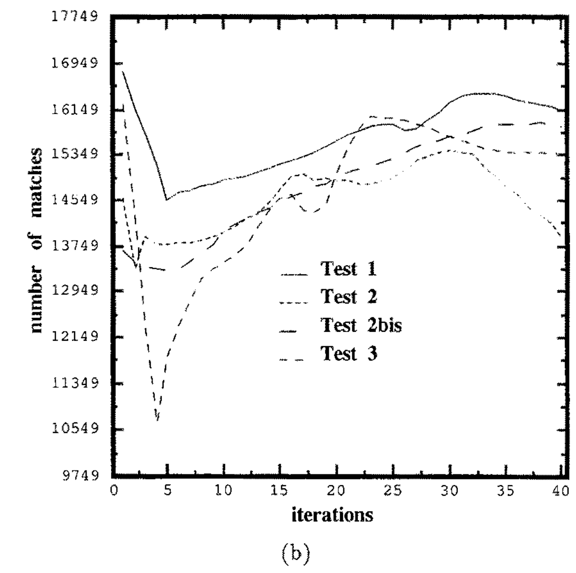


*Fig. 30.* Evolution of (a) the mean distance and (b) the number of matches versus iteration number

### *7.4 About Sampling*

As described earlier, the algorithm developed is based on the use of a simplified, instead of real, definition of the distance between a point and a shape (see Equation 4). That is, we use the minimum of all distances from a given point to each *sample* point of the shape. Different sampling of a shape (even the approxima-

*Table 4.* Comparison between the matching criteria (1) and (2)

| standard deviation | 0 | | 2 | | 4 | | 6 | | 8 | | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| criterion | (2) | (1) | (2) | (1) | (2) | (1) | (2) | (1) | (2) | (1) | (2) | (1) |
| rotation error | 1.81 | 0.12 | 4.36 | 2.68 | 4.60 | 3.63 | 7.56 | 6.40 | 11.35 | 8.52 | 11.94 | 8.36 |
| translation error | 8.22 | 5.83 | 7.38 | 6.35 | 8.56 | 7.32 | 7.61 | 6.42 | 7.36 | 7.08 | 8.96 | 7.92 |
| execution time | 5.04 | 10.04 | 5.32 | 10.80 | 6.10 | 12.54 | 7.64 | 15.88 | 8.18 | 16.74 | 9.98 | 20.48 |

tion error is negligible) does affect the final estimate of the motion. Take a simple example of curves as shown in Figure 31. The curve consists of two line segments (Figure 31a). The sampling in the first frame consists of three points as indicated by the crosses in Figure 31a. We have two samplings in the second frame. The first sampling consists of three points as indicated by dark dots, and the second sampling consists of five points by adding two additional ones (indicated by empty dots) to the first sampling, as shown in Figure 31a. The motion result between the two frames with the first sampling is shown in Figure 31b, and that with the second sampling, in Figure 31c. Clearly, more samples, better results. To solve the problem resulted from sampling, we should ideally use the real distance definition (Equation 3), and use the *real* closest points instead of the closest *sample* points. However, we lose the efficiency achieved with sample points. One possible improvement, as proposed in Besl and McKay (1992), could be the following: First, create a piecewise-simplex (line segments or triangles) approximation of the shape in the second frame (e.g., the Delaunay triangulation from the sample points (Faugeras et al. 1990)). Then, given a point in the first frame, a pure Newton's minimization procedure can be used to find the real closest point, starting with the closest sample point.

There is an easy way to overcome the sampling problem while maintaining the efficiency of the algorithm. It consists in simply increasing the number of sample points through interpolation. The more the number of sample points, the less the sampling will affect the final motion estimate. However, this causes two problems: the increase in the memory required and the increase in the search time (because we increase also the size of the $k$-D tree). Thus a tradeoff must be found.

From the experiments we have carried out, we have obtained satisfactory results using directly closest sample points because the sample points are sufficiently dense.

## *7.5 Uncertainty*

The importance of explicitly estimating and manipulating uncertainty is now well recognized by the computer vision and robotics community (Blostein and Huang 1987; Matthies and Shafer 1987; Kriegman et al. 1989; Ayache and Faugeras 1989; Szeliski 1990). This is extremely important when the data available have different uncertainty distribution for example in stereo where uncertainty increases significantly with depth. We have shown in Zhang and Faugeras (1991) that accounting for uncertainty in motion estimation (via, e.g., a Kalman filter) yields much better results.

For computational tractability and as a reasonable approximation, the uncertainty in a 3-D point reconstructed from stereo is usually modeled as Gaussian; that is, it is characterized by a 3-D position vector and a $3 \times 3$ covariance matrix. The algorithm for motion computation described in Section 3.4 is very efficient. However, it assumes each point has equal uncertainty. And unfortunately it is difficult to extend it to fully take uncertainty into account. To fully take uncertainty into account, we can use for example Kalman filtering techniques which have been widely and successfully applied to solve quite a number of vision problems (Zhang and Faugeras 1992a). However, there will be a significant increase in computation.

The method described below can *partially* take uncertainty into account. Indeed, we can associate, to each pairing between the two frames, a scalar weighting factor $w_i$. Instead of minimizing Equation 5, we compute $\mathbf{R}$ and $\mathbf{t}$ by minimizing the following weighted objective function:

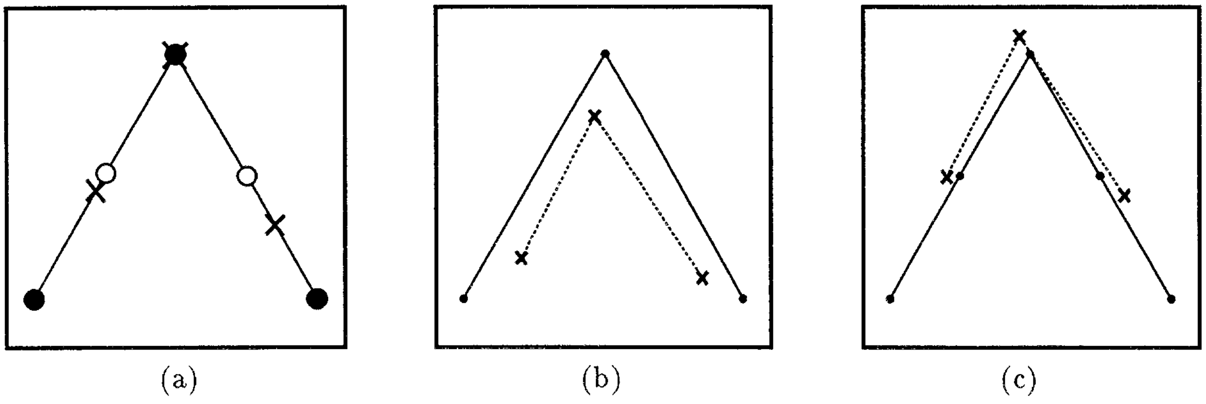


*Fig. 31.* Influence of curve sampling on motion estimation

$$\mathcal{F}(\mathbf{R}, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^{N} w_i \|\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2. \quad (8)$$

The quaternion or dual quaternion method can still be used to compute efficiently $\mathbf{R}$ and $\mathbf{t}$. The weighting factor $w_i$ should be related to the uncertainty of $\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i$. Let $\Lambda\mathbf{x}_i$, $\Lambda\mathbf{y}_i$, and $\Lambda i$ be the covariance matrices of $\mathbf{x}_i$, $\mathbf{y}_i$, and $\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i$. $\Lambda\mathbf{x}_i$ and $\Lambda\mathbf{y}_i$ are given by the sensing system. $\Lambda i$ is then computed as

$$\Lambda i = \mathbf{R}\Lambda\mathbf{x}_i\mathbf{R}^T + \Lambda\mathbf{y}_i,$$

where $\mathbf{R}$ takes the rotation matrix computed during a previous iteration as an approximation. The trace of $\Lambda i$ roughly indicates the magnitude of the uncertainty of $\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i$. Therefore, we choose $w_i$ as

$$w_i = \frac{1}{\mathrm{tr}(\Lambda i)} = \frac{1}{\mathrm{tr}(\Lambda\mathbf{x}_i) + \mathrm{tr}(\Lambda\mathbf{y}_i)}.$$

Thus, the weighting factor is independent of the motion. We have not implemented this method in the current version.

The mechanism for updating the matching, described in Section 3.3, has been designed without considering the different uncertainties in the data points. The same threshold $D_{\max}$ has been used for all points. If the uncertainties in the data points and that in the motion are modeled, one would like to use a pruning criterion that better reflects the sources uncertainty.[3] The idea is the following, similar to that used in Zhang and Faugeras (1992a) for matching 3-D line segments. Let the point under consideration in the first view be $\mathbf{x}$ with covariance matrix $\Lambda\mathbf{x}$. Let the points in the second view be $\{\mathbf{y}_i\}$ with covariance matrix $\{\Lambda\mathbf{y}_i\}$. Let the motion relating the two views be $\mathbf{d}$ with covariance matrix $\Lambda\mathbf{d}$. The vector $\mathbf{d}$ could be $[\mathbf{r}^T, \mathbf{t}^T]^T$. To be general, we define two functions relating $\mathbf{d}$ to the rotation matrix $\mathbf{R}$ and the translation $\mathbf{t}$:

$$\mathbf{R} = \mathbf{f}(\mathbf{d}) \quad \text{and} \quad \mathbf{t} = \mathbf{g}(\mathbf{d}).$$

The (squared) Mahalanobis distance can be used to take into account the uncertainty. It is defined by

$$d_i^{\mathrm{M}} = (\mathbf{f}(\mathbf{d})\mathbf{x} + \mathbf{g}(\mathbf{d}) - \mathbf{y}_i)^T \Lambda i (\mathbf{f}(\mathbf{d})\mathbf{x} + \mathbf{g}(\mathbf{d}) - \mathbf{y}_i),$$

which can be interpreted as the squared Euclidean distance weighted by the uncertainty measure. $\Lambda i$ is the covariance matrix of $\mathbf{f}(\mathbf{d})\mathbf{x} + \mathbf{g}(\mathbf{d}) - \mathbf{y}_i$, and is given, up to the first order, by

$$\Lambda i = \mathbf{f}(\mathbf{d})\Lambda\mathbf{x}\mathbf{f}(\mathbf{d})^T + \Lambda\mathbf{y}_i + J_{\mathrm{d}}\Lambda\mathbf{d}J_{\mathrm{d}}^T,$$

where $J_{\mathrm{d}}$ is the Jacobian $\partial\left[\mathbf{f}(\mathbf{d})\mathbf{x} + \mathbf{g}(\mathbf{d})\right]/\partial\mathbf{d}$. Now the closest point to $\mathbf{x}$ is the point $\mathbf{y}_i$ having the smallest distance $d_i^{\mathrm{M}}$. The reader is referred to Zhang and Faugeras (1992a) for more details on the Mahalanobis distance.

As described in Section 3.3, we do not want to simply match the closest point $\mathbf{y}_i$ with $\mathbf{x}$. In order for $\mathbf{y}_i$ and $\mathbf{x}$ to be matched, the Mahalanobis distance $d_i^{\mathrm{M}}$ must be less than some threshold $\varepsilon$. As $d_i^{\mathrm{M}}$ follows a $\mathcal{X}^2$ distribution with three degrees of freedom, we can choose an appropriate $\varepsilon$, for example, 7.81 for a probability of 89%. In summary, we can replace, if uncertainty is considered, the first two steps of the algorithm described in Section 3.5 by the following:

1. Find, for each point $\mathbf{x}$ in the first view, the point $\mathbf{y}_i$ having the smallest Mahalanobis distance $d_i^{\mathrm{M}}$.

2. Discard the pairings $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ whose $d_i^{\mathrm{M}}$'s are larger than the threshold $\varepsilon$.

### *7.6 About Large Motion*

Because of the local property of the matching criterion used, our algorithm converges to the closest minimum. It is thus best applied in situations where the motion is small or approximately known. In the case of large motion, the algorithm can be adapted in two different ways. The first way is to apply first the global methods as cited in the introductory section to obtain an estimate, which can then be refined by applying the algorithm described in this paper. The second way is to obtain a set of initial registrations by sampling the 6-D motion space, and then apply our algorithm to each initial registration. The final estimate corresponding to the global minimum error is retained as the optimal one. A similar method has been used in Besl and McKay (1992) to solve the object recognition problem.

### *7.7 Multiple Object Motions*

In a dynamic environment, there is usually more than one moving object. It is important to have a reliable algorithm for segmenting the scene into objects using motion information. However, little work has been done so far in this direction.

We have proposed in Zhang and Faugeras (1992b) a framework to deal with multiple object motions. It consists of two levels. The first level deals with the tracking of 3-D tokens from frame to frame and the estimation of their motions. The processing is completely parallel for each token. The second level groups tokens into objects based on the similarity of motion parameters. Tokens coming from a single object should have the same motion parameters. In Zhang and Faugeras (1992b) the tokens used are 3-D line segments, and the experiments have shown that the framework is flexible and powerful. This framework is used in Navab and Zhang (1992) to solve multiple object motions through motion and stereo cooperation. Now if we replace 3-D line segments by 3-D curves and estimate 3-D motion for each curve, the general framework is still applicable. For surfaces, we need to over-segment them into patches such that each patch belongs only to one object. We can then compute motion for each patch and finally group these patches into objects according to motion similarity.

## 8 Highlights With Respect to Previous Work

As mentioned in the introduction, several pieces of similar but independent work have recently been published. They include Besl and McKay (1992); Chen and Medioni (1992); Menq et al. (1992); Champleboux et al. (1992). The same idea is: iteratively matching points in one set to the closest points in another set, given the transformation between the two sets is small. However, as each algorithm is developed in its own context, different techniques have been used.

One of the main differences lies in the matching criterion. Refer to Equation 2. In our algorithm, $p_i$ can take value either 1 or 0 depending on whether the point in the first set has a reasonable match in the second set or not. This is determined by the maximum tolerable distance $D_{\max}$, which, in turn, is set in a dynamic way by analyzing the statistics of the distances as described in Section 3.3. Therefore, our algorithm is capable of dealing with the following situations:

- Gross outliers in the data. The outliers are automatically discarded in the matching and thus have no effect on the final motion estimate.
- Appearance and disappearance in which curves in one set do not appear in the other set. This is usually the case in navigation where objects may enter or leave the field of view.
- Occlusion. An object may occlude other objects, and it may itself be occluded. This is common in both object recognition and navigation.

Besl and McKay (1992) have developed an algorithm for object recognition and location, where a portion of a given model shape is assumed to be observed. In their algorithm, $p_i$ takes always value 1. Thus, their algorithm can only deal with the case in which the first set is a *subset* of the second set. It is powerless in the situations described above. The quaternion algorithm is used to estimate the transformation between the two sets. The singular-value-decomposition algorithm by Haralick et al. (1989) is suggested to replace it in order to identify outliers.

Chen and Medioni (1992) have developed an algorithm for registering multiple range images in order to create a complete model of an object. About one hundred of points on a regular grid in the first range image, called *control points*, are used in order to save computation time. Only points in smooth areas are selected as the control points in order to find a reliable closest point by their method (see below). The method for motion estimation is not specified. Occlusion and outliers issues are not addressed.

Menq et al. (1992) have developed an algorithm for registering range data points with a CAD model for the inspection purpose. In their algorithm, $p_i$ always takes the value 1, too. Occlusion and outliers issues are not addressed. The transformation is estimated by solving a set of nonlinear equations.

Champleboux et al. (1992) have developed an algorithm for the registration of two sets of 3-D points obtained with a laser range finder. Assume that most (about 99%) of points in one set match surfaces in the other, an iterative nonlinear least-squares technique (the Levenberg-Marquardt algorithm) is applied to find the rigid transformation between the two sets. When the iterative process converges, the points whose distances to the other set are larger than a prefixed threshold are considered as outliers and are rejected. Some more iterations are then applied to the retained points.

Another main difference is in the procedure for closest-point computation. In our applications, dense point sets are available, which are directly sorted in a $k$-D tree for efficient closest-point search. In Besl and McKay (1992), several methods are proposed to compute the closest point on a geometric entity (point set, curve, or surface) to a given point. In Chen and Medioni (1992), the surface normal for each control point in the first set is computed. The closest point is found, through an iterative process, at the intersection of the surface normal to the digital surface in the second frame. In Menq et al. (1992), as the model is represented by a set of parametric surface patches, the closest point is determined by solving two nonlinear equations. In Champleboux et al. (1992), the first set of 3-D points is converted into an octree-spline, which is a classical octree decomposition of the work volume, followed by a further division near surface points. The Euclidean distance from nodes to the surface are computed in an exhaustive manner, and saved in the octree-spline. This allows them to quickly compute *approximate* Euclidean distances from points to surface.

## 9 Conclusions

We have described an algorithm for registering two sets of 3-D curves obtained by using an edge-based stereo system, or two dense 3-D maps obtained by using a correlation-based stereo system. We have used the assumption that the motion between two frames is small or approximately known, a realistic assumption in many practical applications including visual navigation. A number of experiments have been carried out and good results have been obtained.

Our algorithm has the following features:

- It is simple. The reader can easily reproduce the algorithm.
- It is extensible. More sophisticated strategies such as figural continuity can be easily integrated in the algorithm.
- It is general. First, the representation used, i.e., point sets, is general for representing arbitrary shapes of the type found in practice. Second, the ideas behind the algorithm are applicable to (many) other matching problems. The algorithm can easily be adapted to solve for example 2-D curve matching.
- It is efficient. The most expensive computation is the process of finding closest points, which has a complexity $O(N \log N)$. Exploiting the coarse-to-fine strategy described in Section 4.3 considerably speeds up the algorithm with only a small change in the precision of the final estimate.
- It is robust to gross errors and can deal with appearance, disappearance and occlusion of objects, as described in Section 8. This is achieved by analyzing dynamically the statistics of the distances, as described in Section 3.3.
- It yields an accurate estimation because all available information is used in the algorithm.
- It does not require any preprocessing of 3-D point data such as for example smoothing. The data are used as is in our algorithm. That is, there is no approximation error.[4]
- The registration results do not depend on any derivative estimation (which is sensitive to noise), in contrast with many other feature-based or string-based matching methods. However, imposing the orientation consistency in matching (Section 3.2) increases the convergence range of the algorithm.

Our algorithm can only partially take the uncertainty of measurements into account. To fully take into account the uncertainty, we should replace the quaternion or dual quaternion algorithm by other methods such as Kalman filtering techniques. This would cause a significant increase in the computational cost of the algorithm.

In our algorithm, one parameter, the parameter $\mathcal{D}$, needs to be set. It indicates when the registration can be considered to be good. It has an impact on the convergence rate, as described in Section 4.1. This is a limitation of our algorithm. In our implementation, $\mathcal{D}$ is related to the resolution of the data available. This method works well for all experiments we have carried out. The result of our algorithm is not sensitive to the value of $\mathcal{D}$. Instead of 10 centimeters, we can set $\mathcal{D}$ to 8 or 12 centimeters, and the same results are obtained. However, a better method probably exists. One question raised is: *can the parameter $\mathcal{D}$ be eliminated?* The parameter $\mathcal{D}$ has been introduced with the concern that the initial estimate could be mediocre. If we have faith in the result provided by the instruments such as odometric and inertial systems on the mobile vehicle, then we can directly use $3\sigma$ (the first case in Section 3.3) to update the matching. The parameter $\mathcal{D}$ is thus not necessary.

Our algorithm converges (not necessarily monotonically) to the closest local minimum, and thus is not appropriate for solving large motion problems. Two possible extensions of the algorithm to deal with large motions have been described in Section 7.6: coupling with a global method or sampling the motion space.

The proposed algorithm works better in a rugged terrain than on a flat ground. This is because there are many local minima for a flat ground which are very close to each other. Due to the local technique we exploit, the final motion estimate will depend essentially on the initial one in this case. By the way, primitive-based methods will not work, either. No salient features can be extracted.

## A Search for Closest Points With $k$-D Trees

Several methods exist to speed up the search process for closest points, including bucketing techniques and $k$-D trees (abbreviation for *$k$-dimensional binary search tree*). We have chosen $k$-D trees, because the data points we have are sparse in space. It is not efficient enough to use bucketing techniques because only a few buckets would contain many points, and many others nothing.[5]

The $k$-D tree is a generalization of bisection in one dimension to $k$ dimensions (Preparata and Shamos 1986). In our case, $k = 3$. A 3-D tree is constructed as follows. First choose a plane parallel to $yz$-plane passing through a data point $P$ to cut the whole space into two (generalized) rectangular parallelepipeds[6] such that there are approximately equal numbers of points on either side of the cut. We obtain then a left son and a right son. Next, each son is further split by a plane parallel to $xz$-plane such that there are approximately equal numbers of points on either side of the cut, and we obtain a left grandson and a right one. We continue splitting each grandson by choosing a plane parallel to $xy$-plane, and so on, letting at each step the direction of the cutting plane alternate between $yz$-, $xz$- and $xy$-plane. This splitting process stops when we reach a rectangular parallelepiped not containing any point; the corresponding node is a leaf of the tree. A $k$-D tree can be constructed in $O(n \log n)$ time with $O(n)$ storage, which are both optimal (Preparata and Shamos 1986).

We now investigate the use of the 3-D tree in searching for closest points. The standard way of using $k$-D trees is to find all points whose distances to $\mathbf{x}$ are within a given value. In our case, we want to find the closest point. One possibility is to use the standard technique to find all points within a given distance, and then to find the point having the smallest distance. We have developed a recursive algorithm which allows the given distance to vary. The algorithm is thus more efficient. More formally, a node $v$ of the 3-D tree $T$ is characterized by two items $(P(v), t(v))$. Point $P(v)$ is the point through which the space is cut into two. The parameter $t(v)$, taking the value 0, 1, or 2, indicates whether the cutting plane is parallel to $yz$-, $xz$-, or $xy$-plane. Two global variables $P$ and $D$ are used to save the point found and the corresponding distance. They are initialized to $-1$ and $D_{\max}$, respectively. At the output, if $P$ is still $-1$, it implies that we cannot find any point with distance less than $D_{\max}$. The search for the closest points to $\mathbf{x}$ is conducted by calling SEARCH(root($T$), $\mathbf{x}$) of the following procedure:

- **input:** a point $\mathbf{x}$, a 3-D tree $T$; two global variables $P$ and $D$ initialized to $-1$ and $D_{\max}$, respectively.
- **output:** the closest point $P$ and the corresponding distance $D$.

```
• procedure: SEARCH($v$, x)
    — if ($v$ == leaf) return ;
    — $c_1 = \mathrm{x}[t(v)]$ ;
    — $c_2 = P(v)[t(v)]$ ; /* $c_2$ has been used to cut the space */
    — if ($|c_1 - c_2| \leq D$) and ($\|\mathbf{x} - \mathbf{P(v)}\| \leq \mathrm{D}$)
      then $P \Leftarrow P(v)$, $D \Leftarrow \|\mathbf{x} - \mathbf{P(v)}\|$ ;
    — if ($c_1 - D < c_2$) then SEARCH(leftson($v$), x);
    — if ($c_2 - D < c_1$) then SEARCH(rightson($v$), x);
```

Unfortunately, the worst-case search time is $O(n^{2/3})$ with the 3-D tree method (see (Preparata and Shamos 1986; pp.77)). Other more efficient algorithms exist, such as a direct access method, but they require much more storage. In practice, we observed good performance with 3-D trees. We found that the search time depends on $D_{\max}$. When $D_{\max}$ is small, the search can be performed very fast. As we update $D_{\max}$ during each iteration, it becomes quite small after a few iterations.

## B Motion Computation using Dual Number Quaternions

For completeness, we summarize in this appendix the dual number quaternion method described in Walker et al. (1991), which can solve a weighted least-squares problem. We can compute $\mathbf{R}$ and t by minimizing the following function

$$\mathcal{F}(\mathbf{R}, \mathrm{t}) = \frac{1}{N}\sum_{i=1}^{N} w_i \|\mathbf{R}\mathrm{x}_i + \mathrm{t} - \mathrm{y}_i\|^2, \tag{9}$$

where $w_i$ is the positive weighting factor associated with the pairing between $\mathrm{x}_i$ and $\mathrm{y}_i$. We can relate $w_i$ to the uncertainty in $\mathrm{x}_i$ and $\mathrm{y}_i$ as shown in Section 7.5.

A quaternion q can be considered as being either a 4-D vector $[q_1, q_2, q_3, q_4]^T$ or a pair $(\check{\mathrm{q}}, q_4)$ where $\check{\mathrm{q}} = [q_1, q_2, q_3]^T$. A dual quaternion $\hat{\mathrm{q}}$ consists of two quaternions q and s, i.e.,

$$\hat{\mathrm{q}} = \mathrm{q} + \varepsilon \mathrm{s}, \tag{10}$$

where a special multiplication rule for $\varepsilon$ is defined by $\varepsilon^2 = 0$. Two important matrix functions of quaternions are defined as

$$\mathrm{Q(q)} = \begin{bmatrix} q_4\mathbf{I} + \mathbf{K}(\check{\mathrm{q}}) & \check{\mathrm{q}} \\ -\check{\mathrm{q}}^T & q_4 \end{bmatrix}, \tag{11}$$

$$\mathbf{W}(\mathrm{q}) = \begin{bmatrix} q_4\mathbf{I} - \mathbf{K}(\check{\mathrm{q}}) & \check{\mathrm{q}} \\ -\check{\mathrm{q}}^T & q_4 \end{bmatrix}, \tag{12}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{K}(\check{\mathrm{q}})$ is the skew-symmetric matrix defined as

$$\mathbf{K}(\check{\mathrm{q}}) = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}.$$

A 3-D rigid motion can be represented by a dual quaternion $\hat{\mathrm{q}}$ satisfying the following two constraints:

$$\mathrm{q}^T\mathrm{q} = 1 \qquad \text{and} \qquad \mathrm{q}^T\mathrm{s} = 0. \tag{13}$$

Thus, we have still six independent parameters for representating a 3-D motion. The rotation matrix $\mathbf{R}$ can be expressed as

$$\mathbf{R} = (q_4^2 - \check{\mathrm{q}}^T\check{\mathrm{q}})\mathbf{I} + 2\check{\mathrm{q}}\check{\mathrm{q}}^T + 2q_4\mathbf{K}(\check{\mathrm{q}}), \tag{14}$$

and the translation vector $\mathbf{T} = \check{\mathrm{p}}$, where $\check{\mathrm{p}}$ is the vector part of the quaternion p given by

$$\mathrm{p} = \mathbf{W}(\mathrm{q})^{\mathbf{T}}\mathrm{s}. \tag{15}$$

The scalar part $p_4$ of p is always zero.

A 3-D vector x is identified with the quaternion $(\mathrm{x}, 0)^7$, and we shall also use x to represent its corresponding quaternion if there is no ambiguity in the context. It can then be easily shown that

$$\mathbf{R}\mathrm{x} + \mathrm{t} = \mathbf{W}(\mathrm{q})^{\mathbf{T}}\mathrm{s} + \mathbf{W}(\mathrm{q})^{\mathbf{T}}\mathrm{Q}(\mathrm{q})\mathrm{x}.$$

Thus the objective function Equation 5 can be written as a quadratic function of q and s

$$\mathcal{F} = \frac{1}{N}[\mathrm{q}^T C_1 \mathrm{q} + W\mathrm{s}^T\mathrm{s} + \mathrm{s}^T C_2 \mathrm{q} + \text{const.}], \tag{16}$$

where

$$\begin{aligned} C_1 &= -2\sum_{i=1}^{N} w_i \mathrm{Q}(\mathrm{y}_i)^T \mathbf{W}(\mathrm{x_i}) \\ &= -2\sum_{i=1}^{N} w_i \end{aligned}$$

$$\begin{bmatrix} \mathbf{K}(\mathrm{y})\mathbf{K}(\mathrm{x}) + \mathrm{y}\mathrm{x}^{\mathbf{T}} & -\mathbf{K}(\mathrm{y})\mathrm{x} \\ -\mathrm{y}^{\mathbf{T}}\mathbf{K}(\mathrm{x}) & \mathrm{y}^{\mathbf{T}}\mathrm{x} \end{bmatrix}, \tag{17}$$

$$\begin{aligned} C_2 &= 2\sum_{i=1}^{N} w_i[\mathbf{W}(\mathrm{x}_i) - \mathbf{Q}(\mathrm{y}_i)] \\ &= 2\sum_{i=1}^{N} w_i \begin{bmatrix} -\mathbf{K}(\mathrm{x}) - \mathbf{K}(\mathrm{y}) & \mathrm{x}-\mathrm{y} \\ -(\mathrm{x}-\mathrm{y})^{\mathrm{T}} & 0 \end{bmatrix}, \quad (18) \\ W &= \sum_{i=1}^{N} w_i, \quad (19) \\ \text{const.} &= \sum_{i=1}^{N} w_i(\mathrm{x}_i^T\mathrm{x}_i + \mathrm{y}_i^T\mathrm{y}_i). \quad (20) \end{aligned}$$

By adjoining the constraints (Equation 13), the optimal dual quaternion is obtained by minimizing

$$\begin{aligned} \mathcal{F}' = \frac{1}{N}[&\mathrm{q}^T C_1 \mathrm{q} + W\mathrm{s}^T\mathrm{s} + \mathrm{s}^T C_2 \mathrm{q} + \text{const.} \\ &+ \lambda_1(\mathrm{q}^T\mathrm{q} - 1) + \lambda_2(\mathrm{s}^T\mathrm{q})], \end{aligned} \quad (21)$$

where $\lambda_1$ and $\lambda_2$ are Lagrange multipliers. Taking the partial derivatives gives

$$\begin{aligned} \frac{\partial\mathcal{F}'}{\partial\mathrm{q}} &= \frac{1}{N}\left[(C_1 + C_1^T)\mathrm{q} + C_2^T\mathrm{s} + 2\lambda_1\mathrm{q} + \lambda_2\mathrm{s}\right] \\ &= 0, \quad (22) \\ \frac{\partial\mathcal{F}'}{\partial\mathrm{s}} &= \frac{1}{N}[2W\mathrm{s} + C_2\mathrm{q} + \lambda_2\mathrm{q}] = 0. \quad (23) \end{aligned}$$

Multiplying Equation 23 by $\mathrm{q}^T$ gives $\lambda_2 = -\mathrm{q}^T C_2 \mathrm{q} = 0$, because $C_2$ is skew-symmetric. Thus s is given by

$$\mathrm{s} = -\frac{1}{2W}C_2\mathrm{q}. \quad (24)$$

Substituting these into Equation 22 yields

$$A\mathrm{q} = \lambda_1\mathrm{q}, \quad (25)$$

where

$$A = \frac{1}{2}\left[\frac{1}{2W}C_2^T C_2 - C_1 - C_1^T\right]. \quad (26)$$

Thus, the quaternion q is an eigenvector of the matrix $A$ and $\lambda_1$ is the corresponding eigenvalue. Substituting the above result back into Equation 21 gives

$$\mathcal{F}' = \frac{1}{N}(\text{const.} - \lambda_1). \quad (27)$$

The error is thus minimized if we select the eigenvector corresponding to the largest eigenvalue.

Having computed q, the rotation matrix **R** is computed from Equation 14. The dual part s is computed from Equation 24 and the translation vector t can then be solved from Equation 15.

## Acknowledgments

This work was carried out in part in the French CNES program VAP. The author would like to thank Olivier Faugeras for stimulating discussions during the work, Steve Maybank for carefully reading the draft version, and Xin Chen for providing the result described in Section 6.2. The author would also like to thank the anonymous reviewers for their suggestions and comments which helped me improve this paper.

## Notes

1. Here we assume the distribution of distances is approximately Gaussian when the registration is good. This has been confirmed by experiments. A typical histogram is shown in Figure 2. More strictly, the $\chi$ distribution is a better approximation if we use the sum of squared distances. As is well known in statistics (central limit theorem), the distribution can be well approximated by a Gaussian if a large number of samples are available. Indeed, we have more than one hundred point matches.
2. The double precision LINPACK rating for the SUN 4/60 is 1.05 Mflops.
3. I thank one of the reviewers for having raised this discussion.
4. It is certain that errors have been introduced during the reconstruction of 3-D points, and that they have been propagated in the motion estimate.
5. Another possibility is to apply bucketing techniques in 2-D, for example, by projecting all points on the ground or on the image plane of the sensors. We have not compared this technique with the $k$-D trees.
6. A generalized rectangular parallelepiped is possibly an infinite volume.
7. Note that in Walker et al. (1991) a 3-D vector x is identified with the quaternion (x/2, 0).

## References

Arun, K., Huang, T. and Blostein, S.: 1987, Least-squares fitting of two 3-D point sets, *IEEE Trans. PAMI* **9**(5), 698–700.

Ayache, N. and Faugeras, O. D.: 1989, Maintaining Representations of the Environment of a Mobile Robot, *IEEE Trans. RA* **5**(6), 804–819.

Besl, P. and Jain, R.: 1985, Three-dimensional object recognition, *ACM Computing Surveys* **17**(1), 75–145.

Besl, P. J.: 1988, Geometric modeling and computer vision, *Proc. IEEE* **76**(8), 936–958.

Besl, P. J. and McKay, N. D.: 1992, A method for registration of 3-D shapes, *IEEE Trans. PAMI* **14**(2), 239–256.

Blostein, S. and Huang, T.: 1987, Error analysis in stereo determination of a 3-D point position, *IEEE Trans. PAMI* **9**(6), 752–765.

Bolles, R. and Cain, R.: 1982, Recognizing and locating partially visible objects, the local-feature-focus method, *Int'l J. Robotics Res.* **1**(3), 57–82.

Brockett, R.: 1989, Least squares matching problems, *Linear Algebra and Its Applications* **122/123/124**, 761–777.

Champleboux, G., Lavallée, S., Szeliski, R. and Brunie, L.: 1992, From accurate range imaging sensor calibration to accurate model-based 3-D object localization, *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Champaign, Illinois, pp. 83–89.

Chen, X.: 1992, *Vision-Based Geometric Modeling*, Ph.D. dissertation, Ecole Nationale Supérieure des Télécommunications, Paris, France.

Chen, Y. and Medioni, G.: 1992, Object modelling by registration of multiple range images, *Image and Vision Computing* **10**(3), 145–155.

Chin, R. and Dyer, C.: 1986, Model-based recognition in robot vision, *ACM Computing Surveys* **18**(1), 67–108.

Faugeras, O. and Hebert, M.: 1986, The representation, recognition, and locating of 3D shapes from range data, *Int'l J. Robotics Res.* **5**(3), 27–52.

Faugeras, O. D., Lebras-Mehlman, E. and Boissonnat, J.: 1990, Representing Stereo data with the Delaunay Triangulation, *Artif. Intell.*

Faugeras, O., Fua, P., Hotz, B., Ma, R., Robert, L., Thonnat, M. and Zhang, Z.: 1992, Quantitative and qualitative comparison of some area and feature-based stereo algorithms, *in* W. Föstner and S. Ruwiedel (eds), *Robust Computer Vision: Quality of Vision Algorithms*, Wichmann, Karlsruhe, Germany, pp. 1–26.

Fua, P.: 1992, A parallel stereo algorithm that produces dense depth maps and preserves image features, *Machine Vision and Applications*. Accepted for publication.

Gennery, D. B.: 1989, Visual terrain matching for a Mars rover, *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, San Diego, CA, pp. 483–491.

Goldgof, D. B., Huang, T. S. and Lee, H.: 1988, Feature extraction and terrain matching, *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Ann Arbor, Michigan, pp. 899–904.

Grimson, W.: 1985, Computational experiments with a feature based stereo algorithm, *IEEE Trans. PAMI* **7**(1), 17–34.

Gueziec, A. and Ayache, N.: 1992, Smoothing and matching of 3-D space curves, *Proc. Second European Conf. Comput. Vision*, Santa Margharita Ligure, Italy, pp. 620–629.

Haralick, R. et al.: 1989, Pose estimation from corresponding point data, *IEEE Trans. SMC* **19**(6), 1426–1446.

Hebert, M., Caillas, C., Krotkov, E., Kweon, I. S. and Kanade, T.: 1989, Terrain mapping for a roving planetary explorer, *Proc. Int'l Conf. Robotics Automation*, pp. 997–1002.

Horn, B.: 1987, Closed-form solution of absolute orientation using unit quaternions, *Journal of the Optical Society of America A* **7**, 629–642.

Horn, B. and Harris, J.: 1991, Rigid body motion from range image sequences, *CVGIP: Image Understanding* **53**(1), 1–13.

Kamgar-Parsi, B., Jones, J. L. and Rosenfeld, A.: 1991, Registration of multiple overlapping range images: Scenes without distinctive features, *IEEE Trans. PAMI* **13**(9), 857–871.

Kehtarnavaz, N. and Mohan, S.: 1989, A framework for estimation of motion parameters from range images, *Comput. Vision, Graphics Image Process.* **45**, 88–105.

Kriegman, D., Triendl, E. and Binford, T.: 1989, Stereo vision and navigation in buildings for mobile robots, *IEEE Trans. RA* **5**(6), 792–803.

Kweon, I. and Kanade, T.: 1992, High-resolution terrain map from multiple sensor data, *IEEE Trans. PAMI* **14**(2), 278–292.

Liang, P. and Todhunter, J. S.: 1990, Representation and recognition of surface shapes in range images: A differential geometry approach, *Comput. Vision, Graphics Image Process.* **52**, 78–109.

Matthies, L. and Shafer, S. A.: 1987, Error modeling in stereo navigation, *IEEE J. RA* **3**(3), 239–248.

Mayhew, J. E. W. and Frisby, J. P.: 1981, Psychophysical and computational studies towards a theory of human stereopsis, *Artif. Intell.* **17**, 349–385.

Menq, C.-H., Yau, H.-T. and Lai, G.-Y.: 1992, Automated precision measurement of surface profile in CAD-directed inspection, *IEEE Trans. RA* **8**(2), 268–278.

Milios, E. E.: 1989, Shape matching using curvature processes, *Comput. Vision, Graphics Image Process.* **47**, 203–226.

Navab, N. and Zhang, Z.: 1992, From multiple objects motion analysis to behavior-based object recognition, *Proc. ECAI 92*, Vienna, Austria, pp. 790–794.

Pavlidis, T.: 1980, Algorithms for shape analysis of contours and waveforms, *IEEE Trans. PAMI* **2**(4), 301–312.

Pollard, S., Mayhew, J. and Frisby, J.: 1985, PMF: A stereo correspondence algorithm using a disparity gradient limit, *Perception* **14**, 449–470.

Preparata, F. and Shamos, M.: 1986, *Computational Geometry, An Introduction*, Springer, Berlin, Heidelberg, New-York.

Radack, G. M. and Badler, N. I.: 1989, Local matching of surfaces using a boundary-centered radial decomposition, *Comput. Vision, Graphics Image Process.* **45**, 380–396.

Robert, L. and Faugeras, O.: 1991, Curve-based stereo: Figural continuity and curvature, *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Maui, Hawaii, pp. 57–62.

Rodríguez, J. J. and Aggarwal, J. K.: 1989, Navigation using image sequence analysis and 3-D terrain matching, *Proc. Workshop on Interpretation of 3D Scenes*, Austin, TX, pp. 200–207.

Safaee-Rad, R., Tchoukanov, I., Benhabib, B. and Smith, K. C.: 1991, Accurate parameter estimation of quadratic curves from grey-level images, *CVGIP: Image Understanding* **54**(2), 259–274.

Sampson, R. E.: 1987, 3D range sensor-phase shift detection, *Computer* **20**, 23–24.

Schwartz, J. T. and Sharir, M.: 1987, Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves, *Int'l J. Robotics Res.* **6**(2), 29–44.

Szeliski, R.: 1988, Estimating motion from sparse range data without correspondence, *Proc. Second Int'l Conf. Comput. Vision*, IEEE, Tampa, FL, pp. 207–216.

Szeliski, R.: 1990, Bayesian modeling of uncertainty in low-level vision, *Int'l J. Comput. Vision* **5**(3), 271–301.

Taubin, G.: 1991, Estimation of planar curves, surfaces, and non-planar space curves defined by implicit equations with applications to edge and range image segmentation, *IEEE Trans. PAMI* **13**(11), 1115–1138.

Walker, M. W., Shao, L. and Volz, R. A.: 1991, Estimating 3-D location parameters using dual number quaternions, *CVGIP: Image Understanding* **54**(3), 358–367.

Walters, D.: 1987, Selection of image primitives for general-purpose visual processing, *Comput. Vision, Graphics Image Process.* **37**(3), 261–298.

Wolfson, H.: 1990, On curve matching, *IEEE Trans. PAMI* **12**(5), 483–489.

Zhang, Z.: 1991, Recalage de deux cartes de profondeur denses: L'état de l'art, *Rapport VAP de la phase 4*, CNES, Toulouse, France.

Zhang, Z.: 1992a, Iterative point matching for registration of free-form curves, *Research Report 1658*, INRIA Sophia-Antipolis.

Zhang, Z.: 1992b, On local matching of free-form curves, *Proc. British Machine Vision Conf.*, University of Leeds, UK, pp. 347–356.

Zhang, Z. and Faugeras, O.: 1991, Determining motion from 3D line segments: A comparative study, *Image and Vision Computing* **9**(1), 10–19.

Zhang, Z. and Faugeras, O.: 1992a, *3D Dynamic Scene Analysis: A Stereo Based Approach*, Springer, Berlin, Heidelberg.

Zhang, Z. and Faugeras, O.: 1992b, Three-dimensional motion computation and object segmentation in a long sequence of stereo frames, *Int'l J. Comput. Vision* **7**(3), 211–241.

Zhang, Z., Faugeras, O. and Ayache, N.: 1988, Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints, *Proc. Second Int'l Conf. Comput. Vision*, Tampa, FL, pp. 177–186. Also as a chapter in R. Kasturi and R.C. Jain (eds), *Computer Vision: Principles*, IEEE computer society press, 1991.

## Color Figures

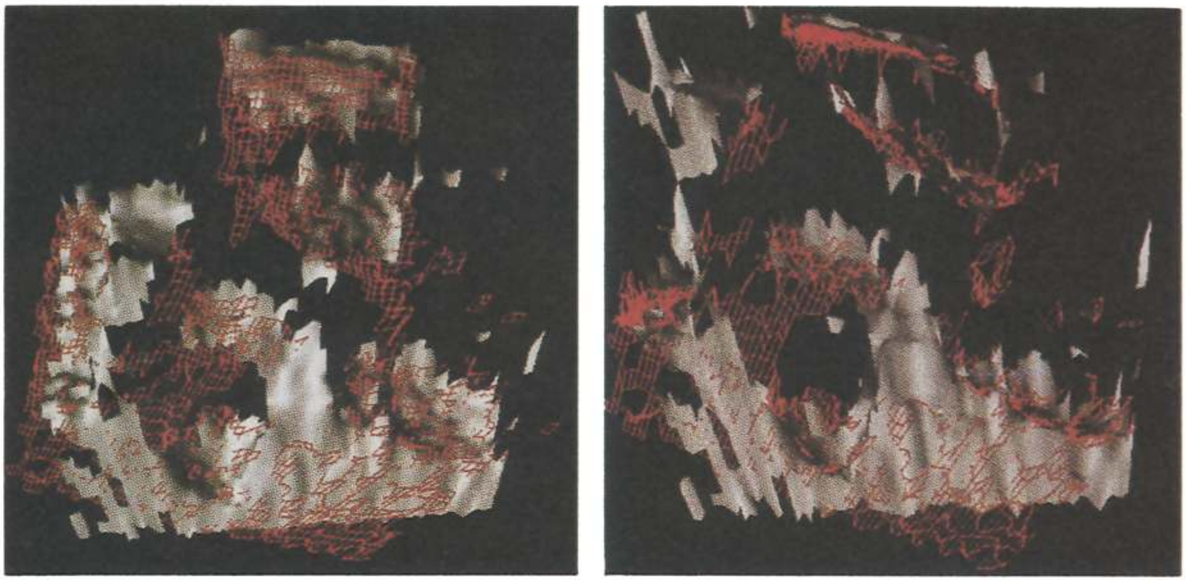


*Fig. 21.* Superposition of the two 3-D maps of a rock scene after a manual registration: front and top views

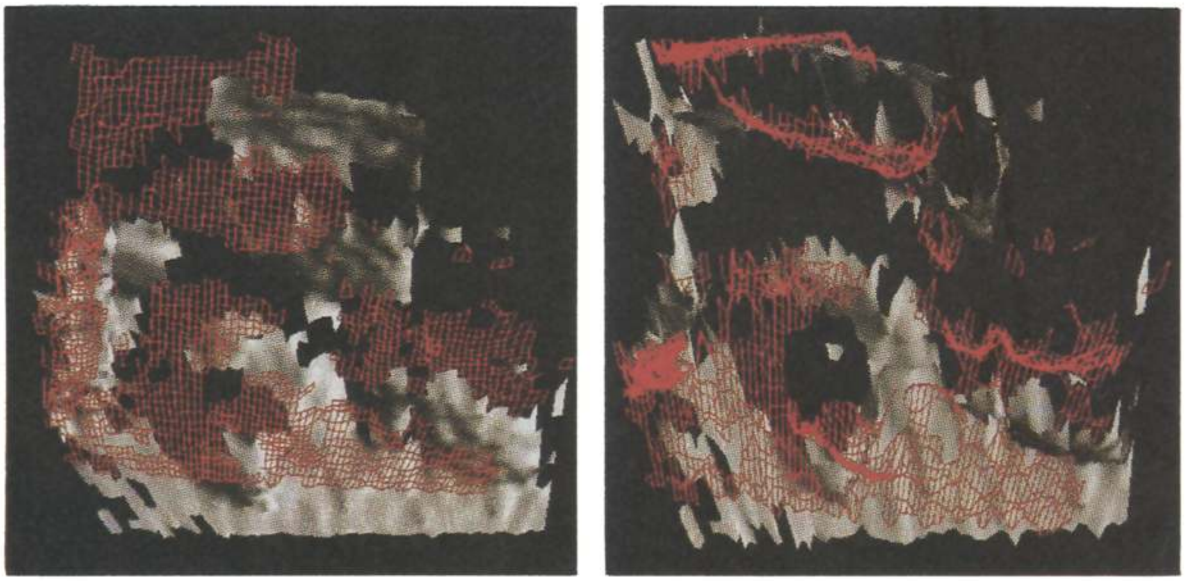


*Fig. 22.* Test 1: Superposition of two 3-D maps before registration: front and top views

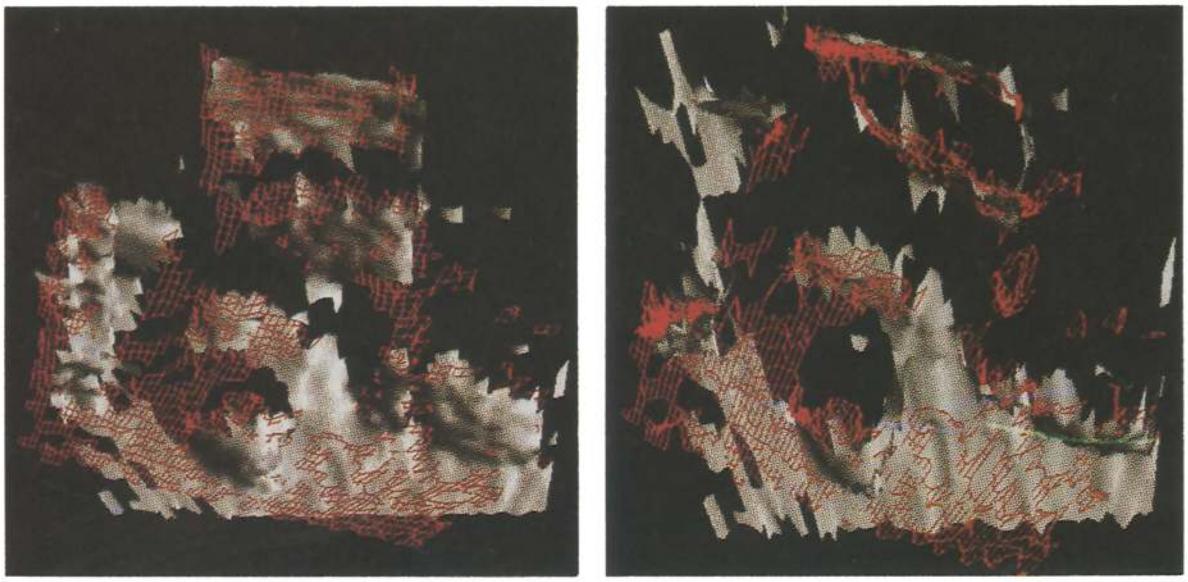


*Fig. 23.* Test 1: Superposition of two 3-D maps after registration: front and top views

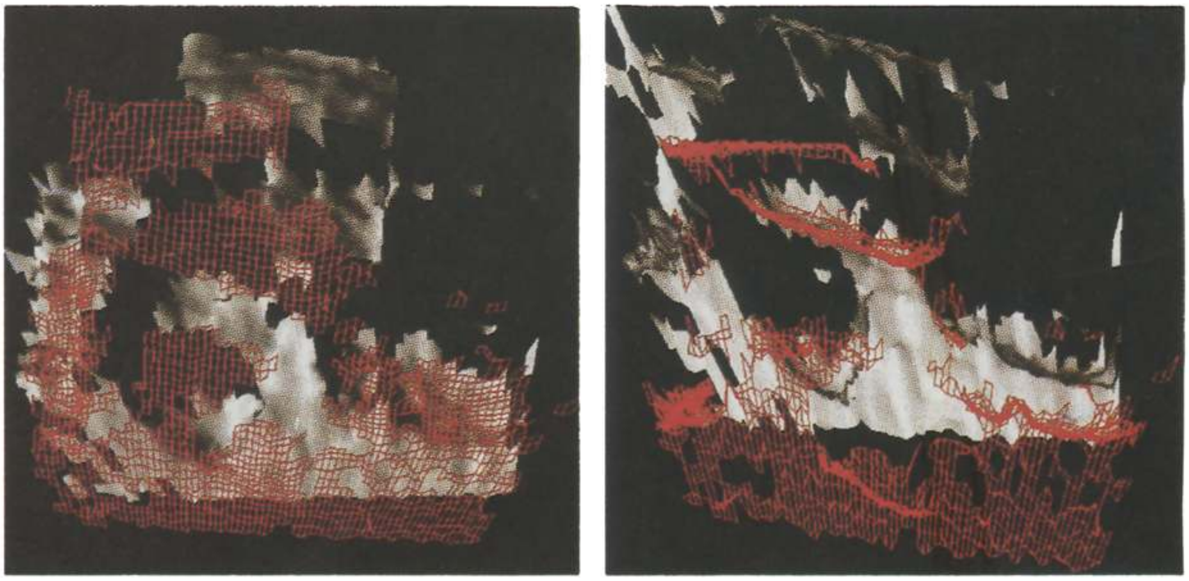


*Fig. 24.* Test 2: Superposition of two 3-D maps before registration: front and top views

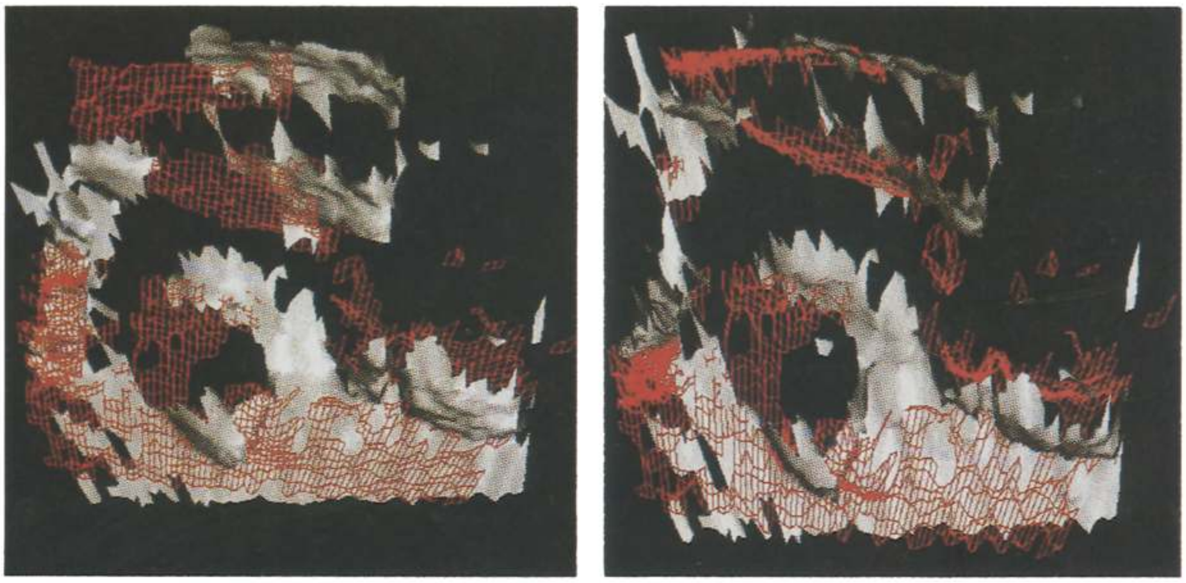


*Fig. 25.* Test 2: Superposition of two 3-D maps after 40 iterations: front and top views

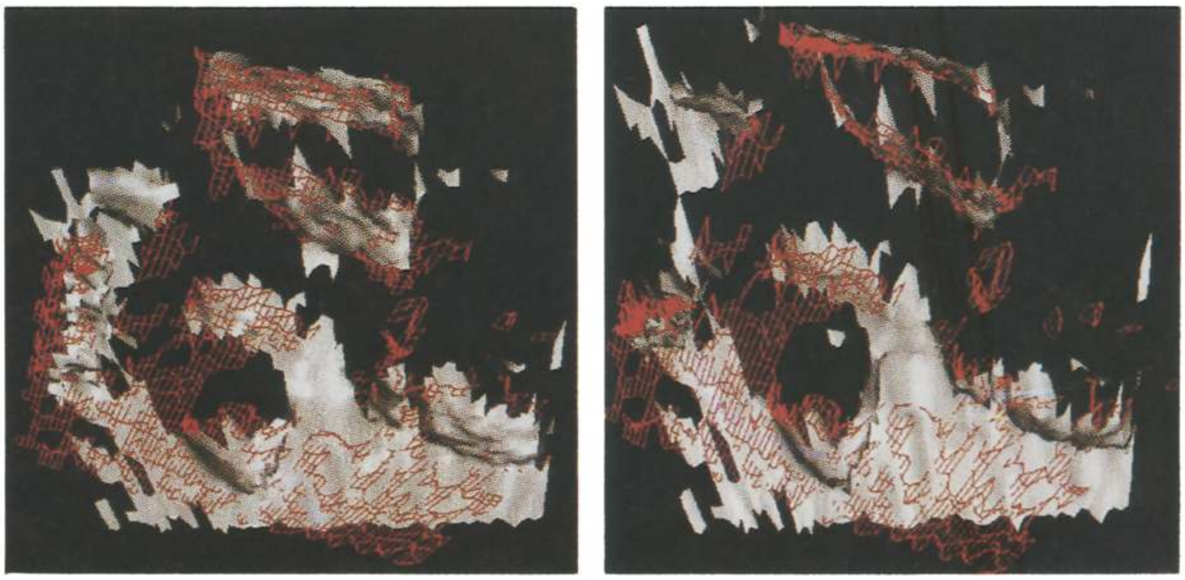


*Fig. 26.* Test 2: Superposition of two 3-D maps after 80 iterations: front and top views

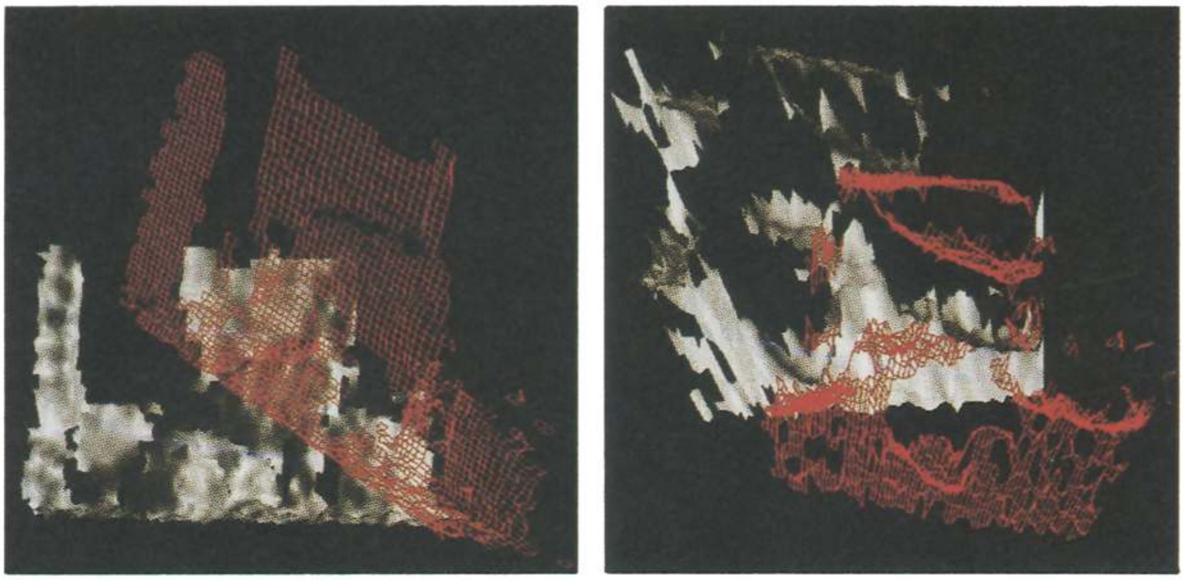


*Fig. 27.* Test 3: Superposition of two 3-D maps before registration: front and top views

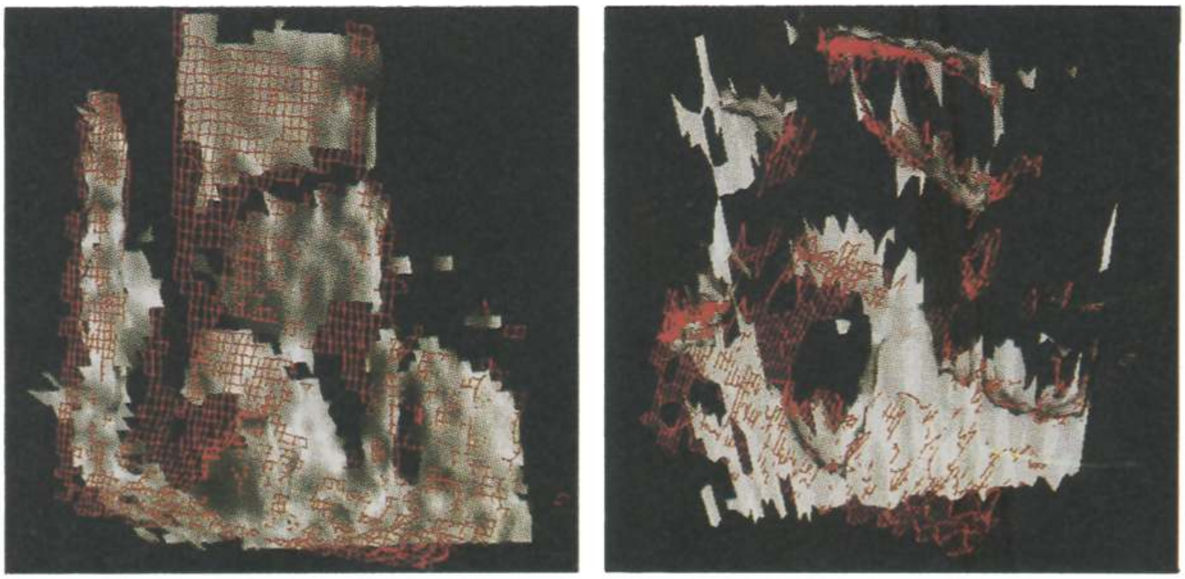


*Fig. 28.* Test 3: Superposition of two 3-D maps after registration: front and top views