On Local Matching of Free-Form Curves

Zhengyou Zhang
INRIA Sophia-Antipolis, 2004 route des Lucioles
BP 93, F-06902 Sophia-Antipolis Cedex (France)
E-Mail: zzhang@sophia.inria.fr

Abstract

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 either small or approximately known, but a more precise registration is required for environment modeling. The algorithm described in this paper meets this need. Objects are represented by free-form curves, i.e., arbitrary space curves of the type found in practice. A curve is available in the form of a set of chained points. The proposed algorithm is based on iteratively matching points on one curve to the closest points on the other. A least-squares technique is used to estimate 3-D motion from the point correspondences, which reduces the average distance between curves 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.

Keywords: Free-Form Curve Matching, 3-D registration, Motion Estimation, 3-D Vision

1 Introduction

Most of the previous work on geometric matching 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 curves, particularly free-form curves, i.e., arbitrary space curves of the type found in practice.

A free-form curve is 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 [1]. However, it is difficult to localize precisely curvature extrema [2, 3], 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 [4, 5, 6]. 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.

BMVC 1992 doi:10.5244/C.6.36

The methods cited above exploit global matching criteria in the sense that they can deal with two sets of free-form curves 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 given 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. In this paper we propose a new method for the registration of curves undergoing small motion.

The key idea underlying our approach is the following. Given that the motion between two successive frames is small, a curve in the first frame is close to the corresponding curve in the second frame. By matching points on the curves in the first frame to their closest points on the curves in the second, we can find a motion that brings the curves in the two frames closer (i.e., the distance between the two curves becomes smaller). Iteratively applying this procedure, the algorithm yields a better and better motion estimate. Interestingly enough, during the preparation of this paper Besl and McKay published a paper in PAMI (issue February 1992) which exploited the same idea [7]. Our work is an independent and much improved treatment. See [8] for a more detailed comparison.

2 Problem Statement

A 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 available in the form of a set of chained 3-D points from either a stereo algorithm [9] or a range imaging sensor [10]. If we know the type of the curve, we can obtain its description $\mathbf{x}$ by fitting, say, conics to the point data [11, 12]. In this work, we shall use directly the chained points, i.e., we are interested in free-form space curves without regard to particular curve primitives.

The use of chained points is equivalent to a piecewise linear approximation to a curve. Let $\mathbf{x}_{i,j}$ $(j = 1, \ldots, N_i)$ be the N_i chained points on the curve $\mathcal{C}_i$. The approximation error can be made arbitrarily small by increasing N_i and decreasing the distances $\|\mathbf{x}_{i,j} - \mathbf{x}_{i,j+1}\|$. At every point $\mathbf{x}_{i,j}$, we compute the tangent direction $\mathbf{u}_{i,j}$ which will be used in the matching procedure. It is not necessary in our algorithm to know precisely the tangent directions. We use the simple estimate

$$\mathbf{u}_{i,j} = (\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j-1}) / \|\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j-1}\| \,,$$

except at the beginning and end points where

$\mathbf{u}_{i,1} = (\mathbf{x}_{i,2} - \mathbf{x}_{i,1}) / \|\mathbf{x}_{i,2} - \mathbf{x}_{i,1}\|$, $\mathbf{u}_{i,N_i} = (\mathbf{x}_{i,N_i} - \mathbf{x}_{i,N_i-1}) / \|\mathbf{x}_{i,N_i} - \mathbf{x}_{i,N_i-1}\|$.

Given two 3-D frames of a scene observed at two different positions, each containing a set of curves. Let $\mathcal{C}_i$ $(i = 1, \ldots, m)$ and $\mathcal{C}'_k$ $(k = 1, \ldots, n)$ be the curves observed in the first and second frames, respectively. Let $\mathbf{x}_{i,j}$ $(j = 1, \ldots, N_i)$ and $\mathbf{x}'_{k,l}$ $(l = 1, \ldots, N_k)$ be the points on the curves $\mathcal{C}_i$ and $\mathcal{C}'_j$,

respectively. The objective 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

$$\mathcal{F}(\mathbf{R},\mathbf{t}) = \frac{1}{\sum_{i=1}^{m}\sum_{j=1}^{N_i} p_{i,j}} \sum_{i=1}^{m}\sum_{j=1}^{N_i} p_{i,j}\, d^2(\mathbf{R}\mathbf{x}_{i,j}+\mathbf{t}, \mathcal{C}'_k) \,. \tag{1}$$

is minimized, where $d(\mathbf{x},\mathcal{C})$ denotes the distance of the point $\mathbf{x}$ to the curve $\mathcal{C}$ (to be defined below), $p_{i,j}$ takes value 1 if the point $\mathbf{x}_{i,j}$ can be matched to a point on the curve $\mathcal{C}'_k$ in the second frame and takes value 0 otherwise.

Furthermore, we assume 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.

3 Iterative Pseudo Point Matching Algorithm

We describe in this section an iterative algorithm for curve 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 curves 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 $d(\mathbf{x},\mathcal{C}'_k)$ between point $\mathbf{x}$ and curve $\mathcal{C}'_k$, which is used in Eq. (1), the criterion defined in the last section. If $\mathcal{C}'_k$ is a parametric curve $(\mathbf{x}'_k : [a,b] \rightarrow \mathbb{R}^3)$, then $d(\mathbf{x},\mathcal{C}'_k) = \min_{u\in[a,b]} d(\mathbf{x},\mathbf{x}'_k(u))$, 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, $\mathcal{C}'_k$ is given as a set of chained points $\mathbf{x}'_{k,l}$ $(l=1,\ldots,N_k)$. We simply define $d(\mathbf{x},\mathcal{C}'_k) = \min_{l\in\{1,\ldots,N_k\}} d(\mathbf{x},\mathbf{x}'_{k,l})$. See [8] 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

$$d(\mathbf{x},\mathbf{y}) = \min_{k\in\{1,\ldots,n\}} d(\mathbf{x},\mathcal{C}'_k) = \min_{k\in\{1,\ldots,n\}} \min_{l\in\{1,\ldots,N_k\}} d(\mathbf{x},\mathbf{x}'_{k,l}) \,.$$

The worst case cost of finding the closest point is $O(N_k^n)$, where N_k^n is the total number of points in the second frame. The total cost while performing the above computation for each point in the first frame is $O(N_i^m N_k^n)$, where N_i^m is the total number of points in the first frame. The use of k-D trees can considerably speed up this process, see the next section.

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 capability, or because some points visible in one frame are not in the other due to sensor/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 along a curve, which is similar to the figural continuity in stereo matching [13, 14], 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 impose the following two simple constraints, which are all unary.

The first is the maximum tolerance for distance. If the distance between a point $\mathbf{x}_{i,j}$ and its closest one $\mathbf{y}_{i,j}$, denoted by $d(\mathbf{x}_{i,j}, \mathbf{y}_{i,j})$, is bigger than the maximum tolerable distance $D_{\max}$, then we set $p_{i,j} = 0$ in Eq. (1), i.e., we cannot pair a reasonable point in the second frame with the point $\mathbf{x}_{i,j}$. This constraint is easily justified for 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, as described below.

The second is the orientation consistency. It can be easily shown that the angle between the tangent of point $\mathbf{x}$ and that of its closest point $\mathbf{y}$ can not go beyond the rotation angle between the two frames [15]. Therefore, we can impose that the angle between the tangents of two paired points should not be bigger than a prefixed value Θ, which is the maximum of the rotation angle expected between the two frames. In our implementation, we set $\Theta = 60°$ to take into account noise effect in the tangent computation. If the tangents can be precisely computed, Θ 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. To this end, one parameter, denoted by $\mathfrak{D}$, needs to be set by the user, which indicates when he considers the registration between two frames is good. See the next section for the choice of the value $\mathfrak{D}$.

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 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 μ and the sample deviation σ of the distances, which are given by

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

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

if $\mu < \mathfrak{D}$ **then** $D^{\mathrm{I}}_{\max} = \mu + 3\sigma$ `/* the regist. is quite good */`

*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 Fig. 1.

else if $\mu < 3\mathfrak{D}$ **then** $D^{\mathrm{I}}_{\max} = \mu + 2\sigma$ /* the regist. is still good */
else if $\mu < 6\mathfrak{D}$ **then** $D^{\mathrm{I}}_{\max} = \mu + \sigma$ /* the regist. is not too bad */
else $D^{\mathrm{I}}_{\max} = \xi$ /* the regist. is really bad */
endif

Here, ξ is the median of all the distances. That is, the number of d_i's less than ξ is approximately equal to the number of d_i's larger than ξ.

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.

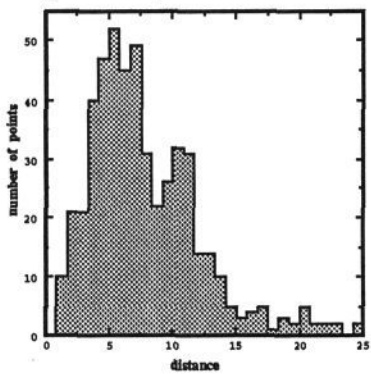


Fig. 1. A histogram of distances

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). For example, when the registration is really bad, only half of the originally recovered matches are retained. Even if there remain several false matches in the retained set, the use of least-squares technique yields still 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

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

which is the direct result of Eq. (1) with the definition of distance given above. Any optimization method, such as steepest descent, conjugate gradient, or complex, 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 [16], singular value decomposition [17] and dual number quaternion method [18]. We have implemented both the quaternion method and the dual number quaternion one. They yield exactly the same motion estimate. See [8] for more details.

3.5 Summary

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

- **input:** Two 3D frames containing m curves $\mathcal{C}_i$ and n curves $\mathcal{C}'_k$, respectively. Each curve $\mathcal{C}$ is a set of chained 3D points $\mathbf{x}_j$.
- **output:** The optimal motion between the two frames.
- **procedure:**
  a) **initialization**
  $D^0_{\max}$ is set to $20\mathfrak{D}$, 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 consideration during the first iteration. The number *20* is not crucial in the algorithm, and can be replaced by a larger one.
  b) **preprocessing**
    (i) Compute the tangent at each point of the first frame.
    (ii) Compute the tangent at each point of the second frame.
    (iii) Build the k-D tree representation of the second frame (see the next section).
  c) **iteration** until convergence of the computed motion
    (i) Finding the closest points satisfying the distance and orientation constraints.
    (ii) Update the matching through statistical analysis of distances.
    (iii) Compute the motion between the two frames from the updated matches.
    (iv) Apply the motion to all points and their tangents in the first frame.

Several remarks should be made here. The construction and the use of k-D trees for finding closest points will be described in the next section. 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. 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 $\mathbf{q} = \begin{bmatrix} \sin(\theta/2)\mathbf{n} \\ \cos(\theta/2) \end{bmatrix}$. 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%.

4 Practical Considerations

In this section, we consider several important aspects in practice, including search for closest points, choice of the parameter $\mathfrak{D}$, and coarse-to-fine search strategy.

4.1 Search for Closest Points

As can be observed in the last section, the search for the closest point to a given point is $O(N)$ in time, where $N = N^n_k$ is the total number of points in the second frame. Several methods [19] exist to speed up the search process, including bucketing techniques and k-D trees (abbreviation for *k-dimensional binary search tree*). We have chosen k-D trees, because the curves we have

form chained points which 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. The reader is referred to [8] for more details of the implementation.

4.2 Choice of the Parameter $\mathfrak{D}$

The only parameter needed to be supplied by the user is $\mathfrak{D}$, which indicates when the registration between two frames can be considered to be good. In other words, the value of $\mathfrak{D}$ should correspond to the expected average distance when the registration is good. When the motion is big, $\mathfrak{D}$ should not be very small. Because we set $D^0_{\max} = 20\mathfrak{D}$, if $\mathfrak{D}$ 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 $30\mathfrak{D}$).

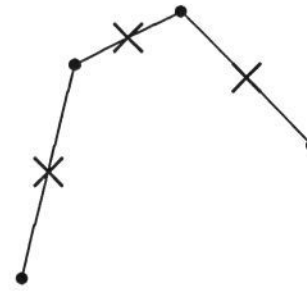


Fig. 2. Illustration of a perfect registration to show how to choose $\mathfrak{D}$

The value of $\mathfrak{D}$ has an impact on the convergence of the algorithm. If $\mathfrak{D}$ 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 $\mathfrak{D}$ 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 $\mathfrak{D}$.

We have worked out a better solution to $\mathfrak{D}$ instead of an ad hoc choice. Let $\bar{D}$ be the average distance between successive points in the second frame, that is

$$\bar{D} = \frac{\sum_{k=1}^{n} \sum_{l=1}^{N_k - 1} \|\mathbf{x}_{k,l} - \mathbf{x}_{k,l+1}\|}{\sum_{k=1}^{n} (N_k - 1)} .$$

Consider a perfect registration shown in Fig. 2. 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 μ of the distances between two sets of points is equal to $\bar{D}/2$. Therefore, we can expect $\mu > \bar{D}/2$ when the registration is not perfect. In our implementation, we set $\mathfrak{D} = \bar{D}$ which gives us satisfactory results.

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, as described above. 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 estimation.

5 Experimental Results

The proposed algorithm has been implemented in C. In order to maintain the modularity, the code is not optimized. In all the experiments described below, the same parameters are used. The program is run on a SUN 4/60 workstation, and any quoted times are given for execution on that machine.

We have tested our algorithm using computer-generated data with different levels of noise. The results show that it is quite robust to noise. Due to space limitation, we only provide in this section the experimental results with real data. The reader is referred to [8] for more examples.

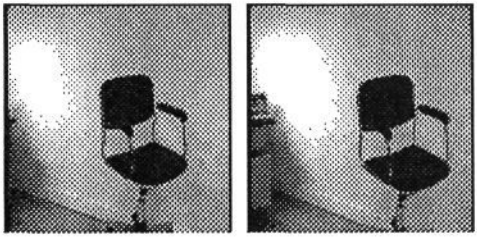


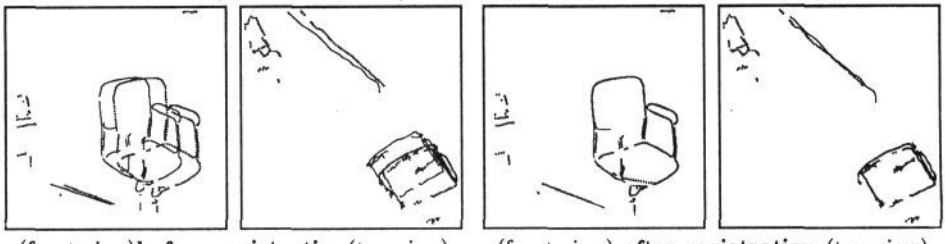
Fig. 3. Images of a chair scene taken by the first camera from two different positions

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 Fig. 3 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 [9] 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 two left-hand pictures of Fig. 4 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

$$\hat{\mathbf{r}} = [-1.527 \times 10^{-3},\ 6.639 \times 10^{-2},\ 2.894 \times 10^{-3}]^T\ ,$$
$$\hat{\mathbf{t}} = [-4.266 \times 10^{0},\ -1.586 \times 10^{0},\ -1.009 \times 10^{2}]^T\ .$$

The motion change is: $\delta\mathbf{r} = 0.78\%$ and $\mathbf{t} = 0.53\%$. The result is shown in the two right-hand pictures of Fig. 4 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 is introduced during the 3-D reconstruction, for it is far away from the cameras.



(front view)**before registration**(top view) (front view) **after registration** (top view)

Fig. 4. Superposition of two 3-D frames before and after registration: front and top views

Now we exploit the coarse-to-fine strategy. 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 estimation as when only doing fine

matching. The execution time, however, decreases from 32.5 seconds to 10.5 seconds, about three times faster. If now we 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 Conclusions

We have described an algorithm for the registration of free-form curves, i.e., arbitrary space curves of the type found in practice. 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 is general for representing arbitrary space curves 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 and 3-D surface 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 considerably speeds up the algorithm with only a small change in the precision of the final estimation.
- It is robust to gross errors and can deal with appearance, disappearance and occlusion of objects. This is achieved by analyzing dynamically the statistics of the distances.
- 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. That is, there is no approximation error.
- It does not require any derivative estimation (which is sensitive to noise), in contrast with many other feature-based or string-based matching methods.

Our algorithm converges 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 [8]: coupling with a global method or sampling the motion space.

In our algorithm, one parameter, the parameter $\mathfrak{D}$, needs to be set by the user. It indicates when the registration can be considered to be good. It has an impact on the convergence rate. In our implementation, $\mathfrak{D}$ is automatically computed using the intervals of chained points. This method works well for all experiments we have carried out. However, a better method probably exists. Intuitively, the parameter $\mathfrak{D}$ is related not only to the intervals of chained points but also to the shape of the curves. $\mathfrak{D}$ should be smaller for rough curves than for smooth ones. We are currently investigating this issue.

We are currently extending the algorithm to solve surface matching problems arising in navigation. When a mobile vehicle navigates in a natural environment, a

correlation-based stereo algorithm or a range finder provides a sequence of dense 3-D maps. Only minor modifications are needed in order to produce an algorithm for registering successive 3-D maps.

Acknowledgment: The author would like to thank Olivier Faugeras for stimulating discussions during the work, and Steve Maybank for carefully reading the draft version.

References

[1] R. Bolles and R. Cain, "Recognizing and locating partially visible objects, the local-feature-focus method," *Int'l J. Robotics Res.*, vol. 1, no. 3, pp. 57–82, 1982.

[2] D. Walters, "Selection of image primitives for general-purpose visual processing," *Comput. Vision, Graphics Image Process.*, vol. 37, no. 3, pp. 261–298, 1987.

[3] E. E. Milios, "Shape matching using curvature processes," *Comput. Vision, Graphics Image Process.*, vol. 47, pp. 203–226, 1989.

[4] T. Pavlidis, "Algorithms for shape analysis of contours and waveforms," *IEEE Trans. PAMI*, vol. 2, no. 4, pp. 301–312, 1980.

[5] J. T. Schwartz and M. Sharir, "Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves," *Int'l J. Robotics Res.*, vol. 6, no. 2, pp. 29–44, 1987.

[6] H. Wolfson, "On curve matching," *IEEE Trans. PAMI*, vol. 12, no. 5, pp. 483–489, 1990.

[7] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. PAMI*, vol. 14, pp. 239–256, February 1992.

[8] Z. Zhang, "Iterative point matching for registration of free-form curves," Research Report 1658, INRIA Sophia-Antipolis, March 1992.

[9] L. Robert and O. Faugeras, "Curve-based stereo: Figural continuity and curvature," in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, (Maui, Hawaii), pp. 57–62, June 1991.

[10] R. E. Sampson, "3D range sensor-phase shift detection," *Computer*, no. 20, pp. 23–24, 1987.

[11] R. Safaee-Rad, I. Tchoukanov, B. Benhabib, and K. C. Smith, "Accurate parameter estimation of quadratic curves from grey-level images," *CVGIP: Image Understanding*, vol. 54, pp. 259–274, September 1991.

[12] G. Taubin, "Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation," *IEEE Trans. PAMI*, vol. 13, pp. 1115–1138, November 1991.

[13] J. E. W. Mayhew and J. P. Frisby, "Psychophysical and computational studies towards a theory of human stereopsis," *Artif. Intell.*, vol. 17, pp. 349–385, 1981.

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

[15] Z. Zhang, O. Faugeras, and N. Ayache, "Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints," in *Proc. Second Int'l Conf. Comput. Vision*, (Tampa, FL), pp. 177–186, December 1988.

[16] O. Faugeras and M. Hebert, "The representation, recognition, and locating of 3D shapes from range data," *Int'l J. Robotics Res.*, vol. 5, no. 3, pp. 27–52, 1986.

[17] K. Arun, T. Huang, and S. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. PAMI*, vol. 9, pp. 698–700, September 1987.

[18] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-D location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 54, pp. 358–367, November 1991.

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