Rigid, affine and locally affine registration of free-form surfaces

Jacques Feldmar, Nicholas Ayache

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

Rigid, Affine and Locally Affine Registration of Free-Form Surfaces

Jacques Feldmar, Nicholas Ayache

N˚ 2220

March 1994

PROGRAMME 4

Robotique,

image

et vision

ISSN 0249-6399

Rigid, Affine and Locally Affine Registration of Free-Form Surfaces

Jacques Feldmar, Nicholas Ayache

Programme 4 — Robotique, image et vision
Projet Epidaure

Rapport de recherche n ° 2220 — March 1994 — 36 pages

Abstract: In this paper, we propose a new framework to perform **nonrigid** surface registration. It is based on various extensions of an iterative algorithm recently presented by several researchers ([BM92], [Zha93], [CM92], [ML92], [CLSB92]) to **rigidly** register surfaces represented by a set of 3D points, when a prior estimate of the displacement is available. Our framework consists of three stages:

- First, we search for the best **rigid** displacement to superpose the two surfaces. We show how to efficiently use curvatures to superpose principal frames at possible corresponding points in order to find a prior rough estimate of the displacement and initialize the iterative algorithm.
- Second, we search for the best **affine** transformation. We introduce differential information in points coordinates: this allows us to match locally similar points. Then, we show how principal frames and curvatures are transformed by an affine transformation. Finally, we introduce this differential information in a global criterion minimized by extended Kalman filtering.
- Third, we locally deform the surface. Instead of computing a global affine transformation, we attach to each point a **local affine** transformation varying smoothly along the surface. We call this deformation a locally affine deformation.

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

All these stages are illustrated with experiments on various real biomedical surfaces (teeth, faces, skulls, brains and hearts), which demonstrate the validity of the approach.

Key-words: surface registration, surface matching, rigid displacement, affine transformation, curvature, principal frame, kd-tree, extended Kalman filter.

(Résumé : tsvp)

Inria

Recalage rigide, affine et localement affine de surfaces gauches

Résumé : Dans cet article, nous proposons une nouvelle approche pour réaliser le recalage **non rigide** de surfaces discrètes. Cette approche est fondée sur diverses extensions d'un algorithme itératif récemment présenté par plusieurs chercheurs ([BM92], [Zha93], [CM92], [ML92], [CLSB92]) pour recaler **rigidement** des surfaces représentées par des ensembles de points, lorsque l'on dispose d'une estimée du déplacement cherché. Elle consiste en trois étapes :

- Premièrement, nous cherchons la meilleure transformation **rigide** pour superposer les deux surfaces. Nous montrons comment utiliser efficacement les courbures pour superposer les repères principaux aux points qui peuvent se correspondre afin de trouver une estimée du déplacement et initialiser l'algorithme itératif.

- Deuxièmement, nous cherchons la meilleure transformation **affine**. Nous utilisons les informations différentielles sur les points. Cela nous permet de mettre en correspondance des points localement similaires. Puis nous montrons comment les courbures principales et le repère principal sont transformés par une transformation affine. Finalement, nous introduisons cette information différentielle dans un critère global que nous minimisons à l'aide d'un filtre de Kalman étendu.

- Troisièmement, nous déformons localement la surface. Au lieu de calculer une transformation affine globale, nous associons à chaque point de la surface une transformation **affine locale** telle que la variation de ces transformations sur la surface soit lisse. Nous appelons cette déformation une déformation localement affine.

Toutes ces étapes sont illustrées par des résultats sur diverses surfaces biomédicales réelles (des dents, des visages, des crânes, des cerveaux et des coeurs), qui démontrent la validité de cette nouvelle approche.

Mots-clé : recalage, mise en correspondance, déplacement rigide, transformation affine, courbure, repère principal, kd-tree, filtre de Kalman étendu.

1 Introduction

Recently, many studies in computer vision have been devoted to the analysis of curved objects. This is an extremely important task simply because a lot of objects are curved in our environment. These objects are often described by their surfaces. They are acquired using different techniques: extraction of contours or iso-surfaces in 3D medical images, range imaging sensors or stereo algorithms.

In this paper, we are concentrating on the surface matching problem. When two surfaces represent the same object, it is often useful to superpose them. For instance, this is important for the medical diagnosis to compare images acquired at different times or coming from different modalities. Another example is when surfaces are incomplete: fusing different acquisitions yields a complete description of the surface. One can refer to the article by Lisa Gottesfeld Brown [Bro92] for a review of the existing techniques.

Our work is an extension of an iterative algorithm described by [BM92], [Zha93], [ML92], [CM92] and [CLSB92] to rigidly register surfaces. This algorithm is detailed in section 2.1 and is called "the iterative algorithm" in this article. Cohen proposes in [Coh94] a rigorous mathematical formulation. The potential based method using principles of mechanics presented in [MR92] is also relatively close to this "iterative algorithm". Of course, classical techniques to register polyhedral objects have influenced our work. The book of Grimson ([Gri90]) provides a very good review of these techniques. Our method to find the initial estimate of the rigid displacement is a kind of hypothesis-verification scheme. The work of A. Gueziec [GA94b] to match crest lines has also influenced this part of our work. Indeed, A. Gueziec uses the curvature and the Frénet frame of points on curves. But it is a geometric hashing scheme and the goal is to match space curves.

When the two surfaces do not come from the same object, but from objects of the same class (for example two faces or two brains) it is also highly desirable to establish point to point correspondences. An example of such an application is to match a brain with an anatomical atlas in order to automatically find abnormalities or to label the brain with anatomical names. Another example is to track the deformation over time of a beating heart, or to register images of a deformable region of a patient (e.g. abdomen) taken at different times. But this nonrigid matching task is much harder. One can again refer to [Bro92] for a review of the existing (but not numerous) techniques.

Our approach to perform the nonrigid matching is quite related to the surface deformation methods because we deform the surfaces in order to bring the corresponding points nearer. For example, the deformation techniques presented in [BK89],

[MT93],[MT91], [PS91] or [NA93] are very interesting. These authors deform the surfaces using a **physical** model involving internal and external forces. Our deformations are quite different: they are the result of a **geometric** transformation, and the constraints used are based on geometric differential informations. Hence, unlike elastic surface models, the curvature of the surface tends to be preserved during the deformation process.

Our work is closer to the techniques which compute a global transformation of the 3D images to deform the surface. Indeed, we compute affine or locally affine transformations. See for example the 3D-spline deformations presented in [SL94] or [BCA94] which are very interesting. But these authors do not explicitly try to match points with similarity of shape which seems to be important to perform matching. Moreover, unlike locally affine deformations, the CPU time must be increased to obtain more local deformations with 3D-spline transformations.

The use of curvature to track points on deformable objects described by Cohen et al. in [CAS92] (and first introduced by Duncan et al. in [DOSA91]) has also connections with our work. Cohen uses the curvature to constraint the possible corresponding points on the two contours. But his definition of the closest point is different. Moreover, he is interested in matching plane curves and does not deform them. Finally, our approach is slightly close to the work of H. Delingette [Del94]. Even if the techniques are very different (the goal of Delingette is not to match surfaces), they share two common objectives: the deformation of the surface without any prior knowledge on its topology and the notion of preservation of the local shape during the deformation process.

In this paper, we propose several extensions to the iterative algorithm in order to develop a complete scheme to match surfaces. In section 2, we describe our contribution to find the best rigid displacement between two surfaces. We first present the original iterative algorithm (2.1). Then we present our method to efficiently find the requisite initial estimate, using curvatures and principal frames (2.2). Finally, we rapidly explain how we deal with partially occluded surfaces to find a very accurate rigid displacement (2.3).

In section 3, we explain how we have extended the iterative algorithm in order to find a good affine transformation. We first introduce differential information in point coordinates and we show how they are transformed by an affine transformation (3.2). Then we modify the definition of best affine transformation introducing curvatures in the criterion and we use extended Kalman filters to find it (3.3). Finally, we present results on real data to illustrate that the found affine transformation brings the two surfaces much nearer than the rigid one (3.4).

In section 4, we present a very simple and promising surface deformation technique to obtain the final match of the two surfaces. First we explain how we associate an affine transformation to each point on the surface (4.1). Then we present results on real data to show that the scheme described in this paper works quite well and allows us to match surfaces (4.2). Finally, we conclude in section 5, summarizing the contributions of this paper and presenting our future work.

2 Computing the rigid displacement

As explained in the introduction, finding the rigid displacement to superpose two identical surfaces described in two different frames is a very important task. For example, superposing brain surfaces enables to follow the evolution of a pathology (brain tumors, multiple sclerosis, etc...). Another application is to fuse together incomplete range data. For example, a laser technique is used to acquire teeth surfaces (Figure 1). Because of occlusion for each point of view, it is necessary to register together the common part of each view to obtain a complete surface.

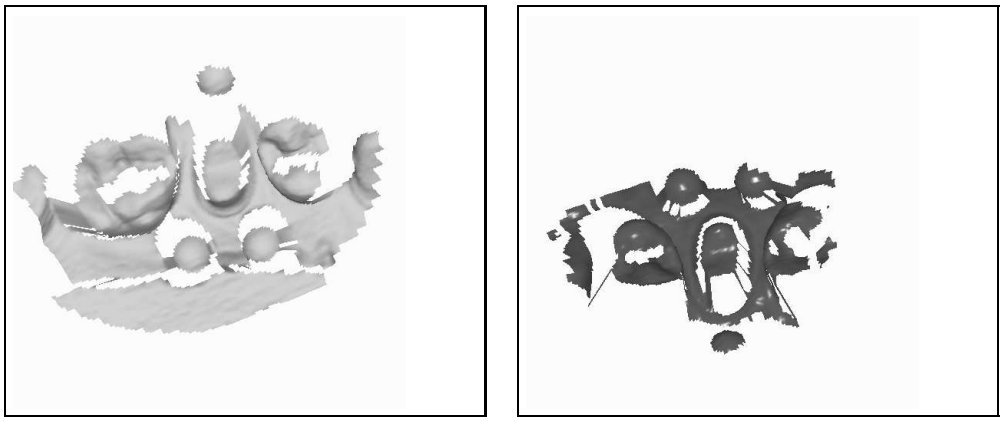

Figure 1: The teeth surfaces to superpose. They have been acquired by Sopha-Bioconcept using a laser technique. One can observe three teeth and three small spheres.

repeat

step 1: $Match_i := \{(M,N)/M \in S_1, N = ClosestPoint(\mathbf{R}_{i-1}M + \mathbf{t}_{i-1})\}$,
where $ClosestPoint(P) = N \in S_2 / \|\overrightarrow{PN}\| = \min_{L \in S_2}(\|\overrightarrow{PL}\|)$

step 2 : compute $(\mathbf{R}_i, \mathbf{t}_i) / \sum_{(M,N)\in Match_i} \|\mathbf{R}_i M + \mathbf{t}_i - N\|^2 =$
$\min_{(\mathbf{R},\mathbf{t})}(\sum_{(M,N)\in Match_i} \|\mathbf{R}M + \mathbf{t} - N\|^2)$

until the termination criterion is reached

Figure 2: The "iterative algorithm"

2.1 The iterative algorithm

We now rapidly describe the iterative algorithm (refer to the original papers for details). The goal is to find the rigid displacement $(\mathbf{R}, \mathbf{t})$[1] to superpose two surfaces, S_1 on S_2, given a very rough estimation $(\mathbf{R}_0, \mathbf{t}_0)$ of this rigid displacement. Each surface is described by a set of 3D-points. The algorithm consists of two iterated steps, each iteration i computing a new estimation $(\mathbf{R}_i, \mathbf{t}_i)$ of the rigid displacement.

1. The first step builds a set $Match_i$ of pairs of points. The construction is very simple: for each point M on S_1, a pair (M, N) is added to $Match_i$, where N is the closest point on S_2 to the point $\mathbf{R}_{i-1}M + \mathbf{t}_{i-1}$. To compute the closest point, different methods are proposed but one can use for example the distance map method [Dan80].

2. The second step is simple too. This is just the least square evaluation of the best rigid displacement $(\mathbf{R}_i, \mathbf{t}_i)$ to superpose the pairs of $Match_i$ (see for example [FH86] for the quaternion method).

The termination criterion depends on the authors: the algorithm stops either when a) the distance between the two surfaces is below a fixed threshold, b) the variation of the distance between the two surfaces at two successive iterations is below a fixed threshold or c) a maximum number of iterations is reached.

We call this algorithm "the iterative algorithm". It is summarized in Figure 2. It is very efficient and finds the right solution when the initial estimate $(\mathbf{R}_0, \mathbf{t}_0)$ of the

[1] A rigid displacement $(\mathbf{R}, \mathbf{t})$ maps each point M to $\mathbf{R}M + \mathbf{t}$ where $\mathbf{R}$ is a 3x3 rotation matrix and $\mathbf{t}$ a translation vector.

rigid displacement is "not too bad" and when each point on S_1 has a correspondent on S_2. But, in practice, this is often not the case. For example, we would like to be able to superpose the teeth of Figure 1, but we have no initial estimate of the rigid displacement and because of occlusions, surfaces are only partially described. The algorithm has to be improved. The two next subsections describe what we propose to avoid these two problems.

2.2 Finding the initial rigid displacement

In this section, we explain how we find the initial estimate we need to run the iterative algorithm of Figure 2. As explained in the various papers describing this algorithm, the estimate does not have to be very accurate : we just want a rough estimate. We use differential informations to get it. The surfaces we have to superpose come from techniques described in [TG92], [Gué93]. So, for each point M on a surface, we know the principal curvatures, and the principal frame.

In the ideal case, because principal curvatures are invariant under rigid displacement, given a point M on S_1 with principal curvatures (k_1, k_2), a point N on S_2 must have the same curvatures to be a possible correspondent. Moreover, if the pair (M, N) is a good match, then the rigid displacement which superposes S_1 on S_2 is also the one which superposes the principal frames attached to M and N respectively on S_1 and S_2. Hence, in the ideal case, the following algorithm would be very efficient: **(1)** choose a point M on S_1, **(2)** compute the set $SameCurvature(M)$ of points on S_2 which have the same curvatures as M, **(3)** for each point N in $SameCurvature(M)$, compute the rigid displacement corresponding to the superposition of the two principal frames and stop when this rigid displacement exactly superposes S_1 on S_2.

But in practice, the two surfaces cannot be exactly superposed, and there is an ambiguity in the orientation of the principal frame. To deal with imprecision on curvatures, we register the points of S_2 in a hash table or in a kd-tree (see [PS85]) indexed by the two principal curvatures. This way, given a point M on S_1, with curvatures (k_1, k_2), we can quickly find the set of points $CloseCurvature(M)$ on S_2 whose curvatures are close to (k_1, k_2). Then, we apply the following algorithm:

1. we randomly choose a point M on the surface S_1
2. we compute the set $CloseCurvature(M)$
3. for each point N in $CloseCurvature(M)$, we compute the rigid displacements corresponding to the superposition of the two principal frames. If $\mathcal{R}_1 =$

$(M, \vec{e}_{11}, \vec{e}_{21}, \vec{n}_1)$ is the principal frame at point M and $\mathcal{R}_2 = (N, \vec{e}_{12}, \vec{e}_{22}, \vec{n}_2)$ the principal frame at point N, we compute two rigid displacements $\mathcal{D}$ and $\mathcal{D}'$. $\mathcal{D}$ corresponds to the superposition of $\mathcal{R}_1$ on $\mathcal{R}_2$. $\mathcal{D}'$ corresponds to the superposition of $\mathcal{R}_1$ on $\mathcal{R}'_2$, where $\mathcal{R}'_2 = (N, -\vec{e}_{12}, -\vec{e}_{22}, \vec{n}_2)$. We have to compute these two rigid displacements because $\mathcal{R}_2$ and $\mathcal{R}'_2$ are both direct, and there is no way to choose between them[2]. The computation of $\mathcal{D}$ an $\mathcal{D}'$ is easy. If $\mathbf{A}$ is the 3×3 matrix whose columns are $(\vec{e}_{11}, \vec{e}_{21}, \vec{n}_1)$, if $\mathbf{B}$ is $(\vec{e}_{12}, \vec{e}_{22}, \vec{n}_2)$ and $\mathbf{B}'$ is $(-\vec{e}_{12}, -\vec{e}_{22}, \vec{n}_2)$, we simply have[3] $\mathcal{D} = (\mathbf{B}\mathbf{A}^t, N - \mathbf{B}\mathbf{A}^t M)$ and $\mathcal{D}' = (\mathbf{B}'\mathbf{A}^t, N - \mathbf{B}'\mathbf{A}^t M)$.

4. we now estimate the ratio of the number of points on the transformed surface $\mathbf{R}S_1 + \mathbf{t}$ which have their closest point on S_2 below a given distance, divided by the number of points of the surface S_1 to check whether $\mathcal{D}$ or $\mathcal{D}'$ reaches our termination criterion. First, we randomly choose a subset S'_1 of points on S_1. Then, for each point P in S'_1, we compute the closest point Q to $\mathbf{R}P + \mathbf{t}$ on S_2 and if
$$\|\mathbf{R}P + \mathbf{t} - Q\| < \delta \frac{\|PM\|}{Diameter},$$
where δ is a given distance threshold and $Diameter$ is the diameter of S_1, then we add the pair (P, Q) to a set $Pair\_ok$. Finally, if the ratio $|Pair\_ok|/|S'_1|$ is larger than ρ $(0 < \rho < 1)$, we decide that $(\mathbf{R}, \mathbf{t})$ is a good estimate of the rigid displacement which superposes S_1 on S_2 and we stop the algorithm. If neither $\mathcal{D}$ nor $\mathcal{D}'$ reaches the termination criterion, then we return to point 1.

The parameter δ is the estimation of the maximal distance between one point on the surface S_1 and its closest point on S_2 for a good estimate of the best rigid registration (it depends on the noise level). ρ is the estimation of the ratio of the number of points on S_1 which have a correspondent on S_2, divided by the number of points on S_1 (which depends on the occlusion). Assume that $(\mathbf{R}, \mathbf{t})$ is a rigid displacement which corresponds to the superposition of the two principal frames for a bad point matching hypothesis. If the points are randomly distributed within a volume and if for a point $M \in \mathbf{R}S_1 + \mathbf{t}$ the probability to have a point $N \in S_2$ such that $\|\overrightarrow{MN}\| < \delta$ is p, then the probability to stop the algorithm with this bad rigid

[2] Note that we are able to orient the normals because, in practice, we know the interior and the exterior of the objects.

[3] We keep the notation $\mathcal{D} = (\mathbf{R}, \mathbf{t})$ where $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ the translation vector of the rigid displacement as explained in section 2.1.

displacement is

$$\sum_{i=k}^{i=n} \binom{i}{n} p^i (1-p)^{n-i}$$

where $n = |S_1'|$ and $k = \rho * n$.

For instance, if $p = 0.4$, $\rho = 0.7$ and $n = 500$ this probability is approximatively 10^{-41}. Of course, it is not possible to set a value for p and the points are not randomly distributed (they belong to a smooth surface) but this can help to understand why, in practice, for our problems, it is not difficult to choose the two parameters δ and ρ and why a good solution is found after a very small number of iterations (two or three). Moreover, note that if after a given small number of iterations no solution is found, we can just stop the algorithm and rerun it choosing a lower ρ or a larger δ.

Typical surfaces we work with have around 10000 points. $CloseCurvature(M)$ is the set of points N on S_2 whose principal curvatures (k_1', k_2') are such that $((k_1' - k_1)^2 + (k_2' - k_2)^2)^{1/2} < Dim/20$ where $Dim = \max(Dim\_k_1, Dim\_k_2)$, assuming that Dim_k_1 (respectively Dim_k_2) is the difference between the maximum and the minimum value of k_1 (respectively k_2) in S_2. For S_1', we randomly choose 5% of points of S_1. For example, Figure 3 shows the initial estimate found for the teeth data. We have chosen $\rho = 0.8$ and $\delta = D/30$ where D is the largest surface diameter. The rigid displacement is found after two iterations in less than twenty seconds on a DEC 5000 workstation. It is not very accurate, but good enough to find the good solution, as described in the next section.

2.3 Working with incomplete surfaces

At step 1 of the iterative algorithm (Figure 2), a point on S_2 is associated to each point on S_1. But when the two surfaces are not complete because of occlusion for example , some points on S_1 do not have any correspondent on S_2. Thus, given a point M on S_1, $(\mathbf{R}_{i-1}, \mathbf{t}_{i-1})$, and $ClosestPoint(\mathbf{R}_{i-1}M + \mathbf{t}_{i-1})$, we have to decide whether (M, $ClosestPoint($ $\mathbf{R}_{i-1}M$ $+ \mathbf{t}_{i-1}$)) is a plausible match. This is very important because, if we accept incorrect matches, the found rigid displacement will not be accurate, and if we reject correct matches, the algorithm will not converge towards the good solution.

As proposed in [Aya91], we make use of the extended Kalman filter. This allows us to associate to the six parameters of $(\mathbf{R}_i, \mathbf{t}_i)$ a covariance matrix $\mathbf{S}_i$ and to use a generalized Mahalanobis distance to check if a match of two points is plausible or not. This implies a change in the second step of the iterative algorithm. Given $Match_i$, instead of computing the rigid displacement $(\mathbf{R}, \mathbf{t})$ which minimizes the

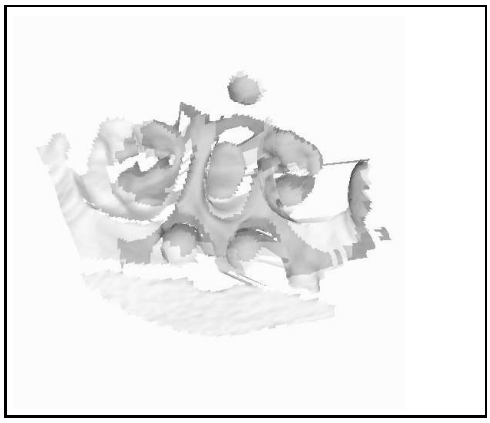

Figure 3: The rough estimate of the rigid displacement. 80% of points of the transformed surface have their closest point on the other surface at a distance smaller than 3.3% of the largest surface diameter.

least square criterion

$$\sum_{(M,N)\ \in\ Match_i} \|\mathbf{R}_i M + \mathbf{t}_i - N\|^2,$$

we recursively estimate the six parameters of $(\mathbf{R}, \mathbf{t})$, and the covariance matrix associated which minimizes the criterion

$$\sum_{(M,N)\ \in\ Match_i\ and\ (M,N)\ is\ plausible} \|\mathbf{R}_i M + \mathbf{t}_i - N\|^2.$$

We applied this technique to superpose the teeth data of Figure 1 using the initial estimate of Figure 3. The three spheres present in the images are artificial markers initially used to compute the rigid displacement. Of course, we do not use this information in our algorithm but we can use them to check the accuracy of the matching. Hence, we computed the set of points on S_1 which have a plausible correspondent on the surface S_2 in the sense described above. This represents 80% of the points on S_1. Then, we computed the average distance between the points of this set and their closest point on S_2 after rigid registration, using respectively the rigid displacement found by our algorithm and the one obtained superposing the center of the three spheres. The rigid displacement found by our algorithm yields an average distance of $0.0070u$, where u is the largest surface diameter. The average distance is

$0.0074u$ after superposition of the three spheres. In fact, because of the sphere center measurement error, our algorithm finds a better global registration than the method using the spheres and taking into account the fact that the surfaces are described by points, this values show that it is indeed very accurate even if the surfaces are partially occluded.

3 Nonrigid matching of two different surfaces

Given two surfaces representing the same object in two different frames, the algorithm of section 2.1 with the two improvements described in sections 2.2 and 2.3 allows us to find the best rigid displacement between the two surfaces. This rigid displacement is found even if we have no prior estimate and if the data are incomplete.

Because this algorithm is robust, we have extended it to tackle a harder problem : nonrigid matching of two different surfaces which present similarities. For example, the faces of two different persons are not identical up to a rigid displacement (Figure 4). But there exists a best rigid displacement which brings the two surfaces close to

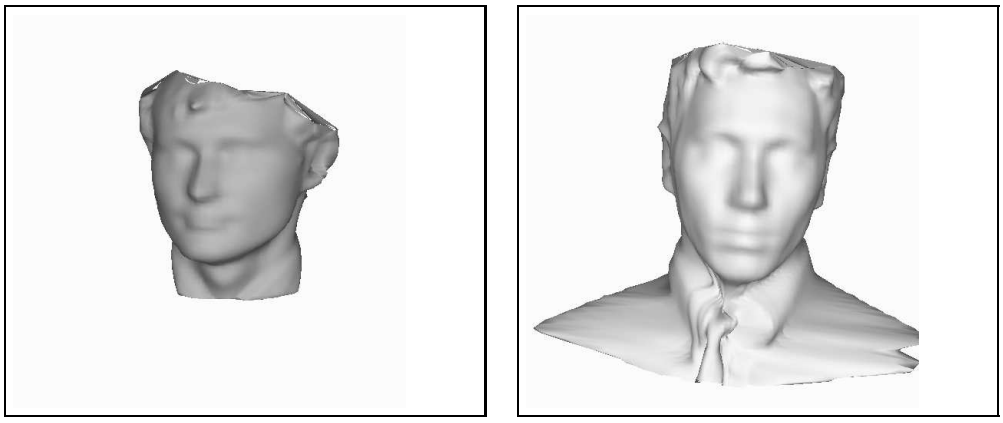

Figure 4: The two faces to superpose. They have been acquired using a Cyberware machine at the Harward Medical School and at NTT HILabs.

each other and the method described in section 2 allows us to find it. Figure 5 shows this best rigid displacement for the faces of Figure 4. The nose, the eyebrows and the chin are globally not very far. But we would like, in order to bring them nearer,

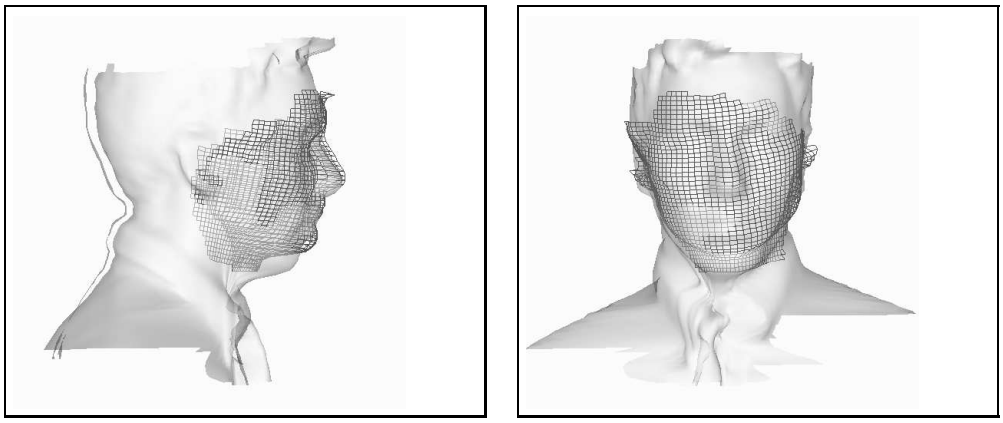

Figure 5: The best rigid displacement found between the two faces. One surface is transparent, the other is represented by lines. When the lines are dark, they are in front of the transparent surface, when they are not, they are behind.

to relax the rigidity constraint of the rigid displacement and extend the method to unconstrained affine transformations in a way which tends to match this similar regions.

Because points of high curvature seem to have a strong anatomical meaning [Aya93], we wish too to enhance their role as landmarks for matching. The three next subsections describe our work to find such affine transformations.

3.1 Finding an affine transformation

A natural extension from rigid transformations to nonrigid ones is the search for an unconstrained affine transformation $(\mathbf{A}, \mathbf{b})$[4]. We could use the rigid displacement found in the previous section as an initial estimate $(\mathbf{A}_0, \mathbf{b}_0)$ and just modify the second step of the iterative algorithm of Figure 2 to search for an affine transformation. It would just be the least square evaluation of the best affine transformation $(\mathbf{A}_i, \mathbf{b}_i)$ to superpose the pairs of $Match_i$. This evaluation is a classical least square problem.

But in practice, this very simple algorithm does not find a good solution: the similarities on the two surfaces do not tend to be brought nearer. Moreover, another

[4] An affine transformation $(\mathbf{A}, \mathbf{b})$ maps each point M to $\mathbf{A}M + \mathbf{b}$ where $\mathbf{A}$ is a 3x3 matrix and $\mathbf{b}$ a translation vector.

major problem occurs with it: it often does not tend to a stable solution. Indeed, when the transformed surface $\mathbf{A}S_1 + \mathbf{b}$ becomes very small or very flat, the criterion is minimized and nothing in the algorithm tends to avoid it. Especially, when $\mathbf{A}$ is the null matrix and $\mathbf{b}$ corresponds to a point on S_2, the criterion vanishes. Figure 6 illustrates this problem. It shows the shrinking of the transformed surface when we

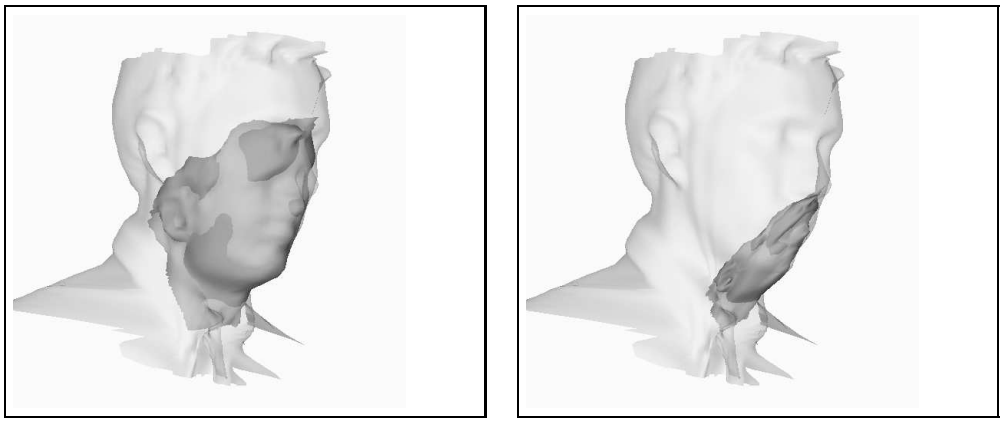

Figure 6: Affine least square matching without additional constraints: the two surfaces after 45 iterations (left) and 75 iterations (right). The transformed surface tends to shrink and becomes very small.

do not stop the algorithm using a termination criterion.

To avoid these two problems, we describe in sections 3.2 and 3.3 the modifications we bring to the original iterative algorithm (Figure 2), respectively to step 1 and step 2. Figure 7 shows the affine transformation found using this modifications for the faces of Figure 4.

3.2 Matching locally similar points

Because points belong to surfaces, we wish to match together points belonging to a local neighborhood of "similar shape" at step 1 of the iterative algorithm (Figure 2). In fact, a point on a surface can be locally described by the osculating quadric (order of contact 2) [Hos92]. When surfaces are quite smooth, this quadric approximate the shape of the surface "around" the point. Because the principal frame and the two curvatures define this quadric, this differential information can be added to the three spatial coordinates of each point to better account for the notion of

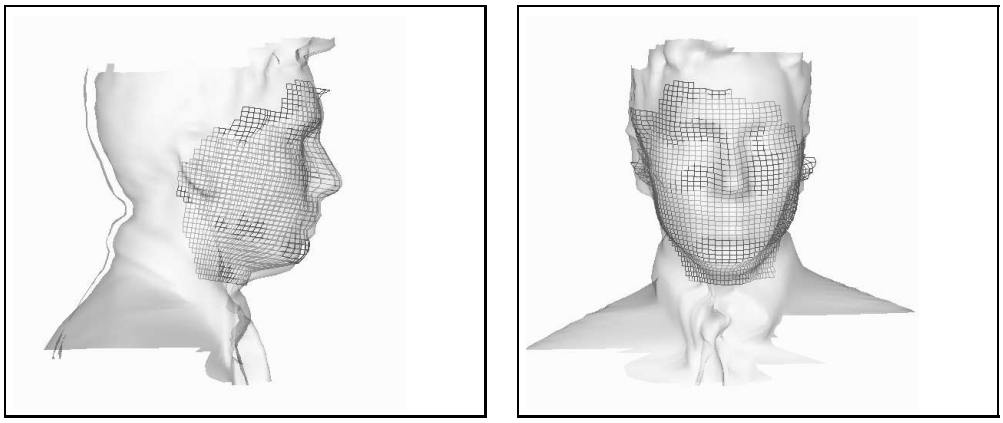

Figure 7: The final affine transformation between the two faces. This has to be compared with the rigid displacement of figure 5. One can see, especially in the left images, that the nose, the mouth and the chin are now much closer. Quantitatively, the average distance decreases from $0.0193u$ to $0.0152u$ (the largest surface diameter is the unit), i.e., a 22% improvement for the affine transformation.

"closest point of similar local shape". Hence, in our formulation, surface points are no longer 3D points: they become 8D points. Coordinates of a point M on the surface S are $(x, y, z, n_x, n_y, n_z, k_1, k_2)$ where (n_x, n_y, n_z) is the normal on S at M, and k_1, k_2 are the principal curvatures. For two points $M(x, y, z, n_x, n_y, n_z, k_1, k_2)$ and $N(x', y', z', n'_x, n'_y, n'_z, k'_1, k'_2)$ we now define the distance:

$$d(M,N) = (\alpha_1(x-x')^2 + \alpha_2(y-y')^2 + \alpha_3(z-z')^2 + \alpha_4(n_x-n'_x)^2 + \alpha_5(n_y-n'_y)^2 + \alpha_6(n_z-n'_z)^2 + \alpha_7(k_1-k'_1)^2 + \alpha_8(k_2-k'_2)^2)^{1/2}$$

where α_i is the inverse of the difference between the maximal and minimal value of the i^{th} coordinate of points in S_2. Using this new definition of the distance, the closest point to P on S_2 is a compromise between the 3D distance, the difference of normal orientation[5] and the difference of curvatures (Figure 8).

But this new definition of points coordinates introduces an interesting problem. At step 1 of the iterative algorithm (Figure 2), we have to compute

[5] Of course, only two parameters are necessary to describe the orientation of the normal (for example the two Euler angles). But we use (n_x, n_y, n_z) because, this way, the Euclidean distance really reflects the difference of orientation between the normals (that is not the case with the Euler angles) and we can use the kd-trees to find the closest point as explained later.

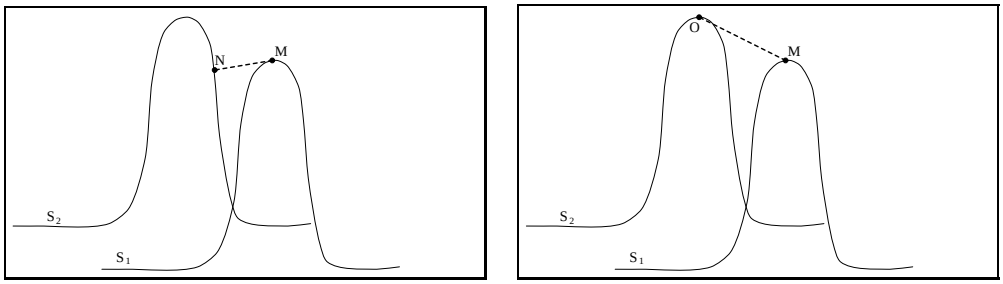

Figure 8: The illustration on two curves S_1 and S_2 of the new definition of the closest point. **Left**: the closest point on S_2 to M is N using the 3D definition of the closest point. **Right**: adding in points coordinates the informations on the normal and the principal curvatures, the points with local similarity of form tend to be matched. Hence, the closest point on S_2 to M is O.

$ClosestPoint(\mathbf{A}_i M + \mathbf{b}_i)$ where M is a point on S_1. Hence, we have to compute the new coordinates $(x', y', z', n'_x, n'_y, n'_z, k'_1, k'_2)$ of $\mathbf{A}_i M + \mathbf{b}_i$, where $(x', y', z')^t = \mathbf{A}_i(x, y, z)^t$, (n'_x, n'_y, n'_z) is the normal on the transformed surface $\mathbf{A}_i S_1 + \mathbf{b}_i$ at point (x', y', z'), and k'_1 and k'_2 are the principal curvatures. In fact, because we need this result in section 3.3 we show in appendix B that:

proposition 1:

> *When a surface S is transformed into a surface $\mathbf{A}S + \mathbf{b}$ by an affine transformation $(\mathbf{A}, \mathbf{b})$, the principal frame and the curvatures at point $\mathbf{A}M + \mathbf{b}$ on $\mathbf{A}S + \mathbf{b}$ depend only on the principal frame and the curvatures at point M on S.*

More precisely, there exists a parameterization of $\mathbf{A}S_1 + \mathbf{b}$ such that, denoting E', F', G' the coefficients of the first fundamental form at point $\mathbf{A}M + \mathbf{b}$ on $\mathbf{A}S_1 + \mathbf{b}$, e', f', g' the coefficients of the second fundamental form and $(M, \vec{e}_1, \vec{e}_2, \vec{n})$ the principal

frame at point M on S_1, we have:

$$\begin{cases} E' = \mathbf{A}\vec{e}_1.\mathbf{A}\vec{e}_1 \\ F' = \mathbf{A}\vec{e}_1.\mathbf{A}\vec{e}_2 \\ G' = \mathbf{A}\vec{e}_2.\mathbf{A}\vec{e}_2 \\ e' = \dfrac{det(\mathbf{A})k_1}{\|\mathbf{A}\vec{e}_1 \wedge \mathbf{A}\vec{e}_2\|} \\ f' = 0 \\ g' = \dfrac{det(\mathbf{A})k_2}{\|\mathbf{A}\vec{e}_1 \wedge \mathbf{A}\vec{e}_2\|} \end{cases} \quad (1)$$

Because we know the coefficients of the fundamental forms of the transformed surface $\mathbf{A}S_1+\mathbf{b}$ at point $\mathbf{A}M+\mathbf{b}$, we can compute the coordinates $(x', y', z', n'_x, n'_y, n'_z, k'_1, k'_2)$ (see appendix A).

Just a problem remains to introduce this new definition of points coordinates: computing $ClosestPoint(\mathbf{A}_i M + \mathbf{b}_i)$. In 8D, we cannot use the technique described in [Dan80] as in the 3D case. The distance map would be much too big. We use the kd-tree technique as proposed in [Zha93] for the 3D case. This takes much more time than before: the calculus time of the closest point has to be improved. But the use of kd-trees allows us to perform step 1 of figure 2 in a reasonable time. Each iteration now takes 45 seconds (CPU time) instead of 9 seconds (for surfaces of 7000 points). To improve the performances, it is possible to work on a subset of points of surfaces. For example, it is possible to extract crest lines points [TG92] or extremal points [Thi94]. Or simply, we can select a given percentage of points, choosing points which have the highest mean curvature. What is important is that most of the selected points on S_1 have a correspondent on S_2 and that the selected points describe relatively well the surfaces.

With this new definition of point coordinates, points with local similarity effectively tend to be matched. Of course, we could have done the extension also including in point coordinates the two principal direction $\vec{e}_1$ and $\vec{e}_2$. We have not done it for two reasons. First, the higher the dimension of the space, the less efficient the kd-tree technique. Second, as explained in section 2.2, it is very difficult to orient principal frames unambiguously.

3.3 Constraints on the affine transformation

We have now to focus on a major drawback in the previous search for the affine transformation which best superposes S_1 on S_2. The criterion we minimize at step 2 of the iterative algorithm (Figure 2) can always vanish. This is the case when $\mathbf{A}$ is

the null matrix and $\mathbf{b}$ corresponds to a point on S_2. In practice, the surface $\mathbf{A}_i S_1 + \mathbf{b}_i$ becomes sometimes very flat or very small as explained in section 3.1. To avoid this problem, we propose to define a new criterion. Let $(x, y, z, k_1, k_2, \vec{e}_1, \vec{e}_2)_k$ be the 3D coordinates, principal curvatures and principal directions of points M_k on S_1. Let $(x', y', z', k'_1, k'_2)_k$ be the 3D coordinates and principal curvatures of points $\mathbf{A}M_k + \mathbf{b}$ on the transformed surface. We call $\mathbf{g}$ the function which associates $(x', y', z', k'_1, k'_2)_k$ to $((x, y, z, k_1, k_2, \vec{e}_1, \vec{e}_2)_k, \mathbf{A}, \mathbf{b})$. The existence of this function is a consequence of the proposition 1 and we use the equations (1) to compute it. The new criterion we propose to minimize at step 2 of the algorithm is:

$$\sum_{(M_k, N_k) \in Match_i} p_k \| \mathbf{g}((x, y, z, k_1, k_2, \vec{e}_1, \vec{e}_2)_k, \mathbf{A}, \mathbf{b})) - N_k \|^2 \qquad (2)$$

where the coordinates of N_k are $(x'', y'', z'', k''_1, k''_2)_k$: the 3D coordinates and the two principal curvatures. The norm is the 5D norm with each coefficient possibly weighted. This new criterion measures both the 3D distance and the difference of curvature between S_2 and the transformed surface $\mathbf{A}S_1 + \mathbf{b}$. Moreover, the coefficients p_k allow to increase the importance in the criterion of the match for high curvature points. They can be, for example, the mean curvature or $max(|k_1|, |k_2|)$.

We use the extended Kalman filter formalism (EKF) to minimize this new criterion (2). The details are given in appendix C. The only difficulty is to derive $\mathbf{g}$ with respect to $(x, y, z, k_1, k_2, x'', y'', z'', k''_1, k''_2)$ and each coefficient of $\mathbf{A}$ and $\mathbf{b}$. Of course, the developed expressions are quite formidable. But, using the "Maple" software, they are computed without problem[6]. Even if the minimized criterion is nonlinear, the minimization works very well: we have made numerous experiments on synthetic data, and the global minimum was always found even with rather large data noise, and rather crude initial estimates.

Using the new definition of the closest point (section 3.2) at step 1 and this new criterion at step 2, the modified iterative algorithm find good and stable solutions. Figure 7 shows the affine transformation found for the faces of Figure 4. The solution is found after ten iterations in about 7.5 minutes (CPU time) on a DEC 5000 workstation[7]. It has to be compared with the rigid displacement of Figure 5. The chin, the mouth, the nose and the eyebrows of the two faces are now much closer.

[6] Just remark that this makes sense to compute these derivatives. Indeed the parameters are independent: given a set of points and given principal curvatures and principal frames at these points, there exists a surface which interpolates these points with these curvatures and principal frames.

[7] This time is relatively important. Most of it is spent in the search for the closest points with the kd-tree technique. But it could be drastically reduced selecting on the surfaces characteristic points (our surfaces have about 7000 points !). This is out of the scope of this paper.

To quantitatively evaluate the error, we have computed the average distance between a point of the transformed surface $\mathbf{A}S_1 + \mathbf{b}$ and its closest point on S_2, using the 3D norm to compute both the distance and the closest point. Setting the largest surface diameter to u, the rigid displacement (Figure 5) yields an average 3D distance of $0.0193u$ whereas the affine transformation (Figure 7) yields an average 3D distance of $0.0152u$, i.e., a 22% improvement. This shows that computing an affine transformation using our improvements brings the two surfaces much closer to each other, while preserving in a certain extent the local curvature information.

3.4 Results and discussion

In this section, we present results on skull data (Figure 10) and on brain data (Figure 9). In each figure, the two top images separately show the two surfaces to superpose. Of course, the two surfaces come from two different patients. We recall that the initial relative positions of the two surfaces do not matter for the search of the affine transformation: the process is initialized using the rigid displacement found as described in section 2 and this does not depend on the initial positions.

The two middle images of Figures 9 and 10 show the rigid displacement found between the two surfaces whereas the bottom images show the affine transformation. Quantitatively, the error[8] in the brain example is $0.011u$ for the rigid displacement and $0.0091u$ for the affine transformation. This is an 18% improvement. In the skull example, the error is $0.0120u$ for the rigid displacement and $0.0098u$ for the affine transformation. This is a 22% improvement. The affine transformation is found after 12 iterations in about 7 minutes (CPU time) on a DEC 5000 workstation for the brains and after 8 iterations in about 8 minutes for the skulls.

This examples illustrate that the best affine transformation is generally found using our improvements and that this affine transformation brings the surfaces much nearer than a rigid displacement. Of course, the superposition is not perfect: an affine transformation is a particular and simple type of **global** deformation (it can be decomposed into successively a rotation, a scaling in the direction of the axes and a rotation). We need a more **local** deformation to obtain a perfect superposition. In the next section, we present our method to locally deform the surfaces in a way which tends to correctly match them.

[8] We recall that the error is the average distance between a point of the transformed surface $\mathbf{A}S_1 + \mathbf{b}$ and its closest point on S_2, using the 3D norm to compute both the distance and the closest point, and setting the largest surface diameter to u.

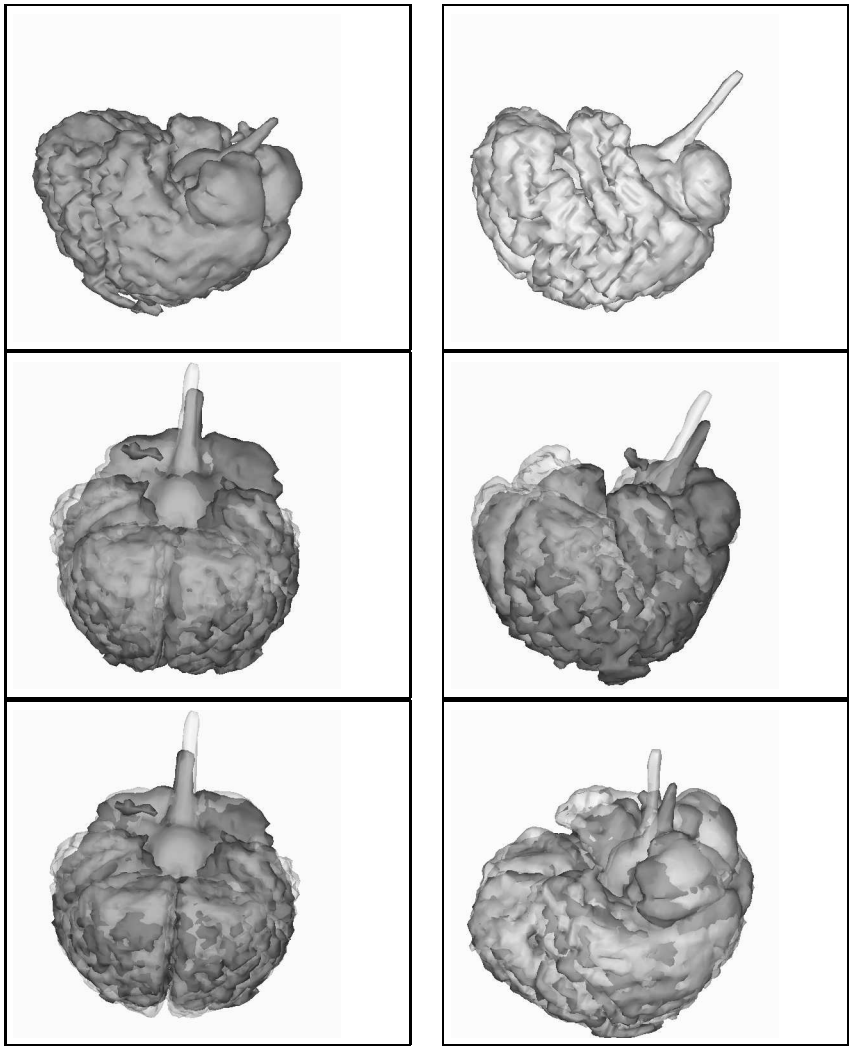

Figure 9: **Top:** the two brains to superpose. The surfaces have been extracted in MRI images (Brigham and Women's Hospital of Boston). **Middle:** result using a rigid transformation. **Bottom:** result using an affine transformation. The brightest surface is transparent. Hence, when the other surface is in front, it appears dark, and when it is behind, it appears clearer. One can observe, comparing the middle-left image and the bottom-left image, that the interhemispherical fissure is not well registered with the rigid displacement, whereas it is with the affine transformation.

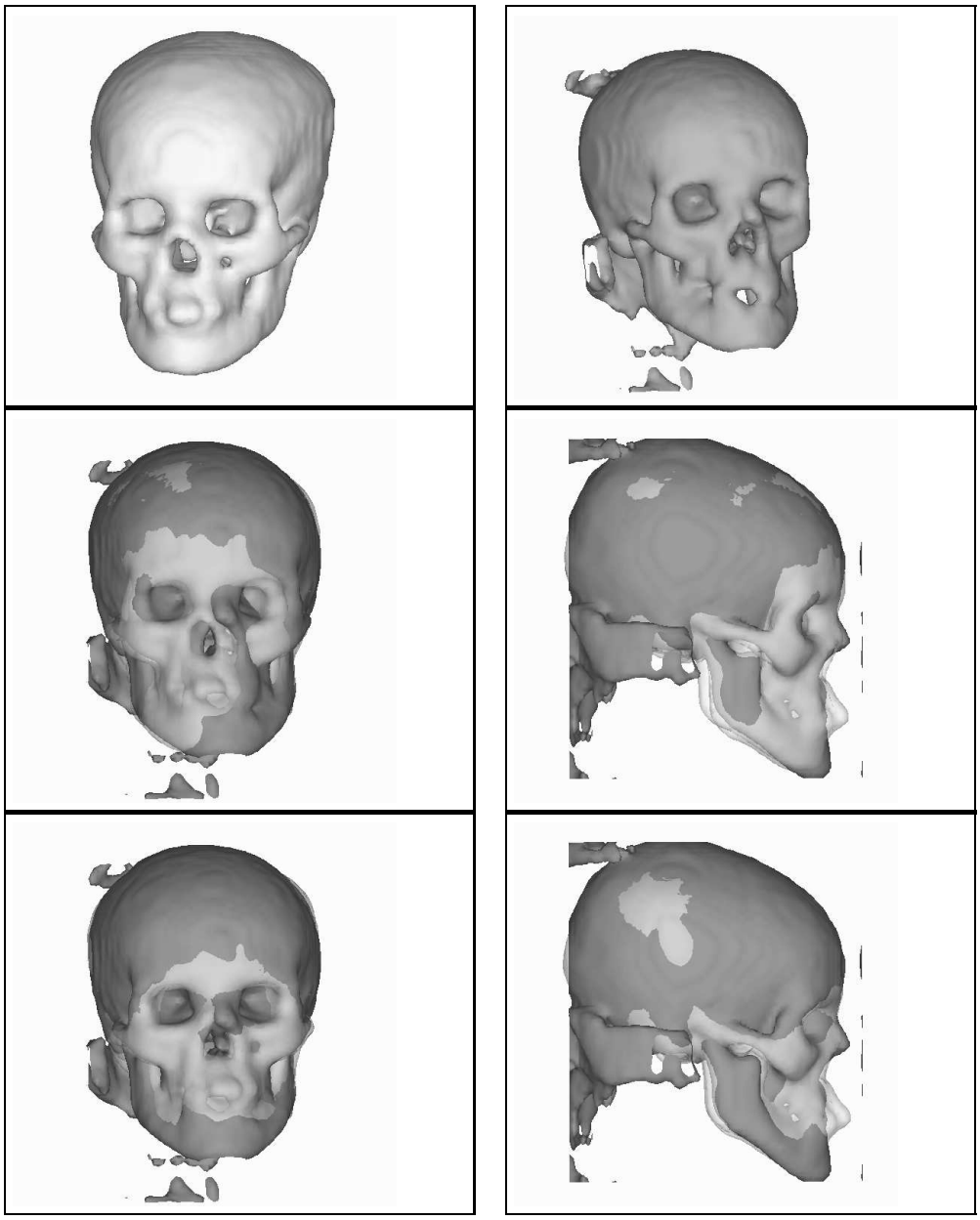

Figure 10: **Top:** the two skulls to superpose. The surfaces have been extracted in X-ray scanner images (New-York University Medical Center). **Middle:** result using a rigid transformation. **Bottom:** result using an affine transformation. One can observe, comparing the middle-left image and the bottom-left image, that the hole corresponding to the nose and the orbits are not well registered with the rigid displacement, whereas they are with the affine transformation.

4 Locally affine deformation

The extensions of the iterative algorithm described in the first two sections allow us to find the best affine transformation $(\mathbf{A}, \mathbf{b})$ to superpose the surface S_1 on the surface S_2. This transformation brings the two surfaces quite close but is too global to perform a perfect superposition. A natural extension is then to compute for each point of the transformed surface a local affine transformation. But, because our goal is to perform matching, it is very important to ensure that the global form of the deformed surface tends to be preserved during the deformation process. For example, the region corresponding to the nose on a face must not become flat. To ensure this, the attached local affine transformations must not vary too much from one point to its neighbor.

4.1 The locally affine deformation algorithm

4.1.1 Description

We now describe the algorithm to compute these local affine transformations with an iterative algorithm. Each iteration i consists of computing for each point M_k in S_1 an affine transformation $(\mathbf{A}_{k,i}, \mathbf{b}_{k,i})$. We call the set $\{(\mathbf{A}_{k,i}, \mathbf{b}_{k,i})\}$ the **locally affine deformation**. The corresponding transformed surface is the set of points $\mathbf{A}_{k,i}M_k + \mathbf{b}_{k,i}$ such that M_k belongs to S_1.

The algorithm is initialized with $\mathbf{A}_{k,0} = \mathbf{A}$ and $\mathbf{b}_{k,0} = \mathbf{b}$ for all M_k in S_1, where $(\mathbf{A}, \mathbf{b})$ is the best affine transformation to superpose S_1 on S_2. As a preprocessing, we first compute and store the set of points in S_1 which belong to the sphere of center M_k and of radius R, where R is the major parameter of the algorithm. We call this set $Sphere(M_k)$ and we use the kd-tree technique to build it (see [PS85]).

Then, in detail, each iteration i consists of two steps.

1. We first attach to each point M_k in S_1 a rigid displacement $(\mathbf{R}_{k,i}, \mathbf{t}_{k,i})$. As in the iterative algorithm (Figure 2), we build a set of pairs of points, associating to each point M_l in $Sphere(M_k)$ the closest point N_l on the surface S_2 to the point $\mathbf{A}_{k,i-1}M_k + \mathbf{b}_{k,i-1}$. The rigid displacement $(\mathbf{R}_{k,i}, \mathbf{t}_{k,i})$ is simply the one which minimizes the square distance between the associated points (M_l, N_l).

2. We now smooth the set of the rigid displacements $(\mathbf{R}_{k,i}, \mathbf{t}_{k,i})$ to compute the locally affine deformation $\{(\mathbf{A}_{k,i}, \mathbf{b}_{k,i})\}$. We simply let:

$$\mathbf{A}_{k,i} = \sum_{M_l \in Sphere(M_k)} \beta(M_l, M_k)\ \mathbf{R}_{l,i}, \qquad \mathbf{b}_{k,i} = \sum_{M_l \in Sphere(M_k)} \beta(M_l, M_k)\ \mathbf{t}_{l,i}$$

where

$$\beta(M, M_k) = \frac{\alpha(M, M_k)}{sum}, \quad \alpha(M, M_k) = 1 - \frac{\|\overrightarrow{MM_k}\|}{Diameter} \quad \text{and}$$

$$sum = \sum_{M \in Sphere(M_k)} \alpha(M, M_k)$$

($Diameter$ is the diameter of the surface S_1). Because of the linearity of the affine transformations, this simply means that the point M_k will be transformed into

$$\sum_{M_l \in Sphere(M_k)} \beta(M_l, M_k)\ (\mathbf{R}_{l,i} M_k + \mathbf{t}_{l,i}).$$

In order to understand where this transformed point is, imagine the set of points computed by applying to M_k all the rigid displacements attached to the points M_l belonging to $Sphere(M_k)$. The transformed point is the centroid of this set of points, each point being weighted by a value which is inversely proportional to the distance $\|\overrightarrow{M_l M_k}\|$. Hence, the non rigid motion finally computed for M_k is influenced by the computed rigid displacement attached to all its neighbors.

We stop the algorithm when the distance between the two surfaces is below a fixed threshold or after a given number of iterations. Of course, at step 1 of this deformation algorithm, the definition of the closest point we use is the definition given in section 3.2, i.e. the definition which tends to preserve the local curvature and orientation of the surface predicted by the current affine estimate of the local transformation.

To compute the principal curvatures and the principal frame at a point $P_k = \mathbf{A}_{k,i} M_k + \mathbf{b}_{k,i}$ of the deformed surface, we assume that the affine transformation $(\mathbf{A}_k, \mathbf{b}_k)$ is constant in the neighborhood of P_k. Thus, we can use the equations (1) to compute the new differential quantities. This is just an approximation but it seems to be well justified because the step 2 of the deformation algorithm tends to smooth the variation of the affine transformations along the surface[9]. Moreover, it appears to yield good results in practice.

[9] To compute the exact principal curvatures and principal frame at point P_k, we would need a continuous representation $(\mathbf{A}_i(u,v), \mathbf{b}_i(u,v))$ of the affine transformations in a neighborhood of order 2 of the point P_k. Especially, we would need a local parameterization of the surface around the point P_k. The algorithm would be much more complicate. Our approximation assumes that the terms involving the first and second derivatives of $\mathbf{A}_i(u,v)$ and $\mathbf{b}_i(u,v)$ are negligible with respect to the terms involving $\mathbf{A}_i(u,v)$ and $\mathbf{b}_i(u,v)$ in the computation of the differential properties.

4.1.2 Discussion

The locally affine deformation algorithm deforms the surface $\mathbf{A}S_1+\mathbf{b}$ in a way which tends to bring nearer the corresponding points. Indeed, for the same reason that the iterative algorithm of Figure 2 rigidly registers the surfaces, each transformation $(\mathbf{R}_{k,i}, \mathbf{t}_{k,i})$ tends to bring nearer the region of the surface S_1 described by the points belonging to $Sphere(M_k)$ to its corresponding region on S_2. The new definition of the closest point also tends to match the corresponding points.

Moreover, the global form of the deformed surface tends to be preserved because the variation of the transformations $(\mathbf{A}_{k,i}, \mathbf{b}_{k,i})$ is smooth along the surface and because a global affine transformation is just a composition of a rotation, a scaling in the direction of three orthogonal axes, and a rotation. The variation is smooth for two reasons. First, for two point M_k and M_l close on the surface, the sets $Sphere(M_k)$ and $Sphere(M_l)$ differ only from a few points and this implies that the computed rigid displacements $(\mathbf{R}_{k,i}, \mathbf{t}_{k,i})$ and $(\mathbf{R}_{l,i}, \mathbf{t}_{l,i})$ cannot be very different. Second, the affine transformation attached to M_k is a weighted average of the rigid displacements computed for the neighbors of M_k and this also tends to smooth the variation. If we need a smoother variation, it is possible to add a third step to the locally affine deformation algorithm. It is the repetition (typically two or three times) of:

$$\begin{cases} \forall k, \quad \mathbf{A'}_{k,i} = \displaystyle\sum_{M_l \in Sphere(M_k)} \beta(M_l, M_k)\ \mathbf{A}_{l,i}, \quad \mathbf{b'}_{k,i} = \sum_{M_l \in Sphere(M_k)} \beta(M_l, M_k)\ \mathbf{b}_{l,i} \\ \\ \forall k, \quad \mathbf{A}_{k,i} = \mathbf{A'}_{k,i}, \quad \mathbf{b}_{k,i} = \mathbf{b'}_{k,i} \end{cases}$$

Using this third step, the variation is smoother and the displacement of points M_k is influenced by the displacement of point which does not belong to $Sphere(M_k)$. This allows us to smooth over a large scale without increasing the radius R of $Sphere(M_k)$.

Note that the larger the radius R, the smoother the variation of the affine transformations and that if R is equal to $Diameter$, the locally affine deformation corresponds to a rigid displacement. This suggests a "multiscale" strategy. For the first iterations, the radius R is chosen to be quite high[10]. Hence, the biggest structures of the surfaces are first registered and we choose R smaller during the last iterations to have a more local registration. Finally, we want to emphasize a major advantage

[10] Of course, we never choose R equal to $Diameter$. Indeed, the complexity of the locally affine deformation algorithm is (both in time and memory space) $O(|Sphere(M_k)|*n)$, where n is the number of points in S_1. In practice, we typically choose for the first iterations R equal to $Diameter/20$ using, if necessary, step 3 of the algorithm to smooth over a larger scale.

of this algorithm with respect to other deformation techniques: we do not need to know either a parameterization or the topology of the surface to deform.

Figure 11 presents the result of the locally affine deformation algorithm for the faces of Figure 4, using as input the surface S_1 transformed by the affine transformation of Figure 7. This result is found after three iterations, choosing $Diameter/20$ for R during the two first iterations and $Diameter/50$ for the last iteration. We do not use step 3 of the algorithm. The CPU time is 2 minutes on a DEC 5000 workstation. Quantitatively, setting the largest surface diameter to u, the final error is $0.0066u$. This shows that the superposition is geometrically quasi-perfect.

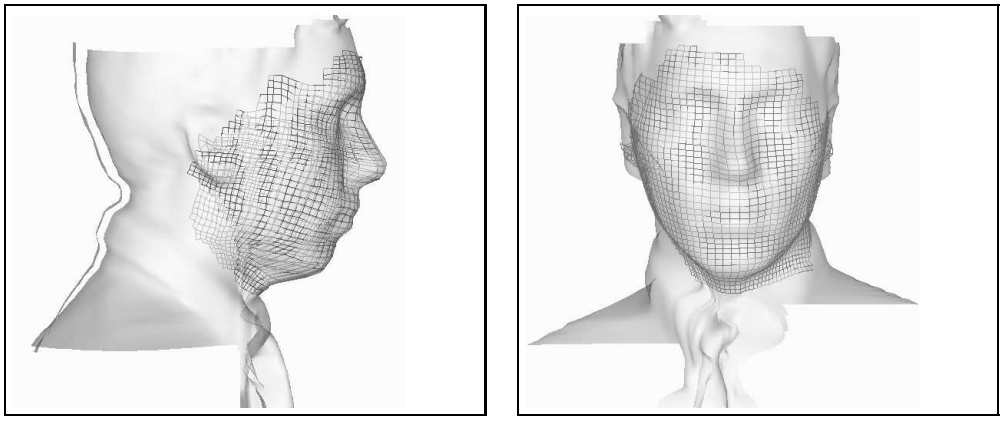

Figure 11: The result of the local affine deformation algorithm for the faces of figure 7. The superposition is quasi-perfect.

But one important question remains: does the iterative algorithm also performs a qualitatively correct matching ? To illustrate it, we have qualitatively colored the surface S_1. In the left image of Figure 12, the chin, the mouth, the nose and the eyebrows are now dark and the rest of the face is bright. This way, each point on the transformed surface S_1^* has an attached color. In order to visualize the match induced by the locally affine deformation algorithm, we have computed for each point M on the surface S_2 its closest point N on the deformed surface S_1^*, and we have attached to M the color of N. Hence, the surface S_2 is colored and the result of this coloration is shown in the right image of Figure 12. Like on the surface S_1, the chin, the mouth, the nose and the eyebrows are now dark and the rest of the

face is bright. This shows that the match of the two faces is both geometrically and qualitatively correct.

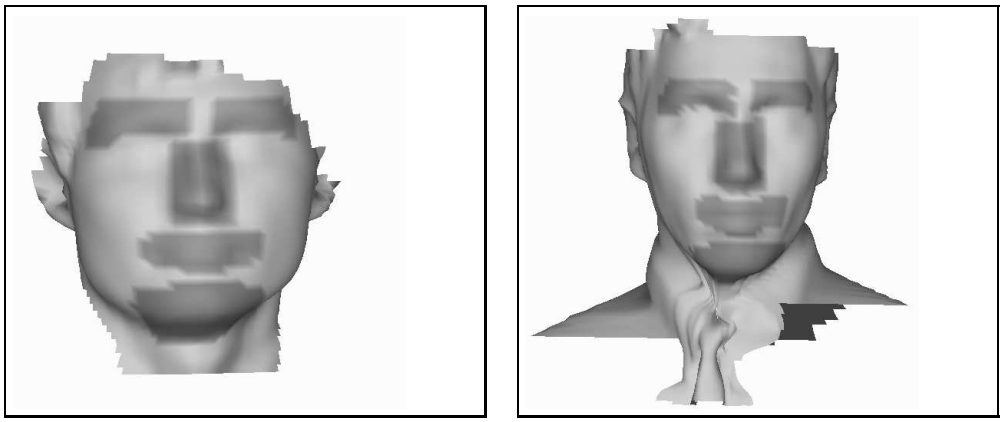

Figure 12: The visualization of the match induced by the deformation of figure 11. Left: the surface S_1 has been roughly colored. Right: the surface S_2 has been colored, computing for each point M its closest point N on the transformed surface S_1^* and attaching to M the same color than N.

4.2 Results

In this section, we present the results of the locally affine deformation for skull data and brain data of figures 10 and 9. We also present results on heart data.

For the skulls, the result using the affine transformation of Figure 10 (bottom) as initial estimate, is shown in Figure 13. The brightest surface is transparent. This explain why the other surface successively appears dark and clearer: it is because the two surfaces are so close that each one alternatively tends to appear in front of the other. This result is found after 3 iteration, and we have chosen R equal to $Diameter/80$. We do not use step 3 of the algorithm. The CPU time is about 4 minutes. Quantitatively, the error is $0.003u$, where u is the largest surface diameter. In fact, for this example too, the superposition is quasi-perfect.

For the brain, the result using the affine transformation of Figure 9 (bottom) as initial estimate, is shown in Figure 14. We have chosen R equal to $Diameter/20$ for the first four iterations and R equal to $Diameter/50$ for the last two iterations and we have repeated step 3 of the algorithm three times at each iteration. The

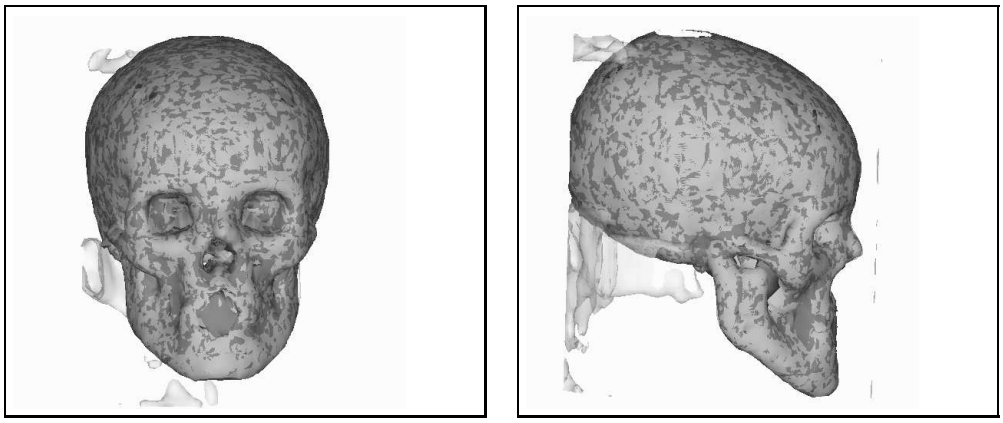

Figure 13: The result of the locally affine deformation algorithm for the skulls of figure 10. The brightest surface is transparent. Hence, when the other surface is in front, it appears dark, and when it is behind, it appears clearer. The alternation of dark and clearer regions shows that the superposition is quasi-perfect.

CPU time is about 15 minutes. It is quite important but the two surfaces are not very close using the best affine transformation (Figure 9). Quantitatively, setting the largest surface diameter to u, the error is $0.0061u$. Qualitatively, one can compare for example the positions of the spinal chord in the figures 9 and 14: they were quite far and the algorithm has correctly superposed them. Here, we want to point out that we have to compute an affine transformation to initialize the locally affine deformation algorithm. Otherwise, for this example, the deformation algorithm would not converge to a correct match when initialized from the best rigid registration.

Finally, the result for a beating heart is shown in Figure 15 (bottom). This example demonstrates that the locally affine deformation algorithm can be used to follow the points on a deformable object over time. Note in Figure 15 (middle) that we have chosen, to illustrate the robustness of the algorithm, the two times where the form of the myocardium is the most different. This result is found after 3 iteration chosing R equal to $Diameter/20$. We do not use step 3 of the algorithm. The CPU time is about 2 minutes. Quantitatively, the error is $0.0400u$ at the initialization of the algorithm and is $0.0041u$ at the end. This shows that the final registration is very good.

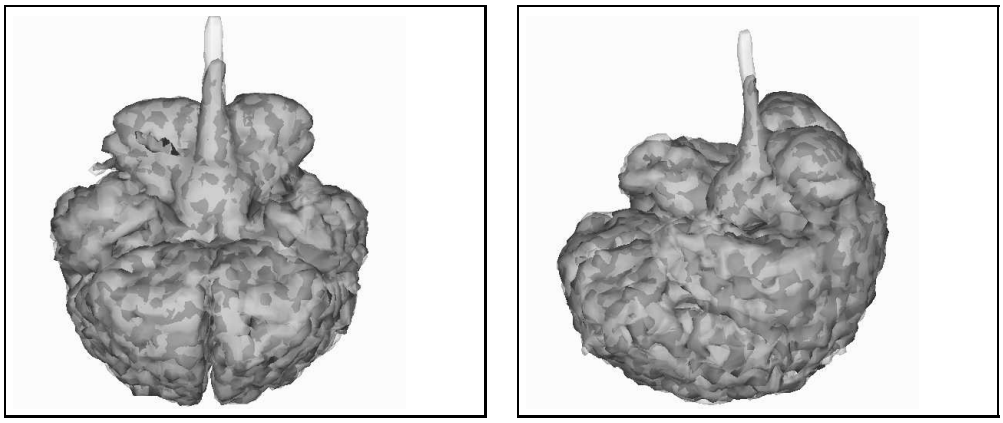

Figure 14: The result of the locally affine deformation algorithm for the brains of figure 9. The alternation of dark and clearer regions shows that the two surfaces are very close. One can also observe, for example, that the spinal chord is now correctly registered.

5 Conclusion

In this paper, we have introduced a complete framework to register curved surfaces nonrigidly, taking advantage first of the computation of the affine transformation of principal curvatures and directions and second of the definition of locally affine deformations. This framework does not require any prior information on the initial positions of the surfaces and applies even if we do not know either a parameterization or the topology of the surfaces to register. It leads to geometrically and qualitatively meaningful accurate registration as illustrated on various and complex real biomedical surfaces.

Future work will be oriented towards two directions. First, we will formalize the locally affine deformation algorithm in terms of continuous energy minimization to try to justify more theoretically why it works. Second, we will experiment the presented framework on new kind of data to estimate the limits of the validity of the approach (which seem to be reached with brain data) and we will concentrate on a biomedical validation at a larger scale of this framework in order to perform the registration between a 3D image and a 3D anatomical atlas.

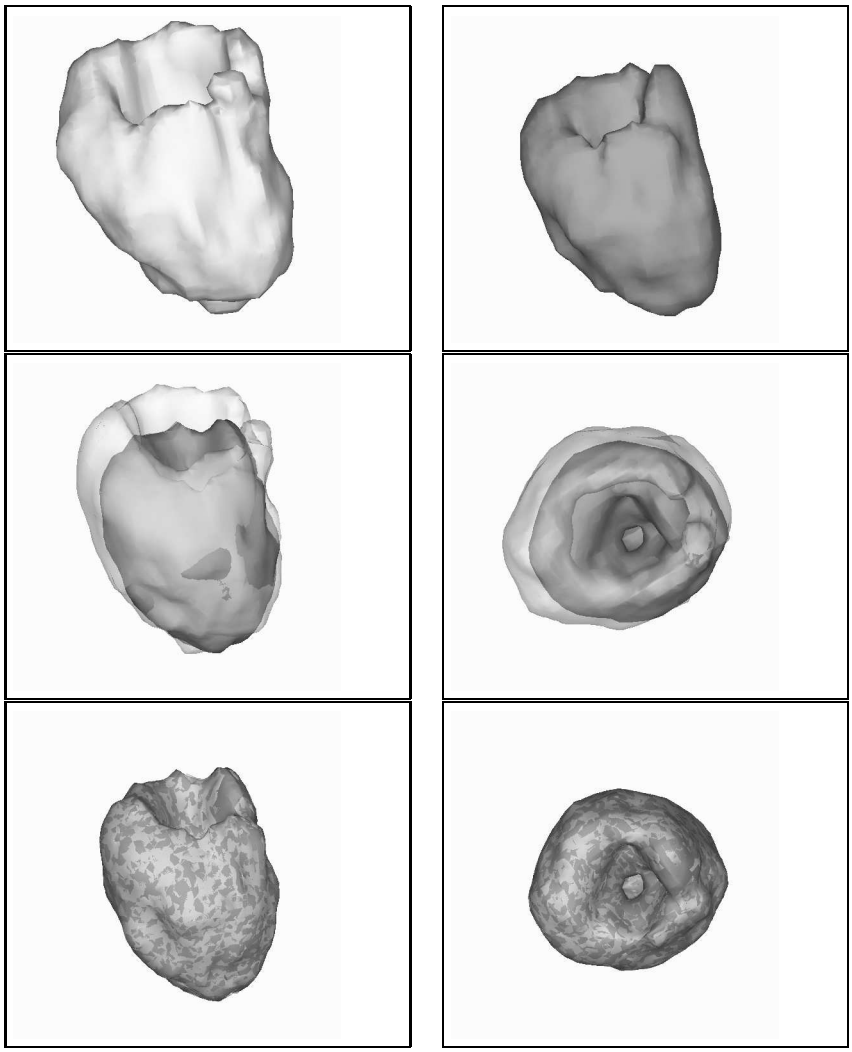

Figure 15: **Top**: The same heart (the myocardium) at two different times. The surfaces have been extracted in nuclear medicine images (Sopha Medical France). **Middle**: the two times shown in the same frame. The brightest surface is transparent. Hence, when the other surface is in front, it appears dark, and when it is behind, it appears less dark. Note that we have chosen the two times where the form of the myocardium is the most different, among all the images that we have of this beating heart. **Bottom**: the result of the local affine deformation algorithm for the beating myocardium. The initial relative positions are shown in the middle images (note that we did not computed the best affine transformation to initialize the algorithm). The alternation of dark and clearer regions shows that the superposition is very good.

Acknowledgements

We are very grateful to Alexis Gourdon for his help both for theoretical and practical problems. We wish to thank too Herve Delingette and Jean-Philippe Thirion for stimulating discussions. Thanks are also due to Sopha-Bioconcept for providing teeth data, Dr Altobelli (Harvard Medical School) and Yasuhiko Watanabe (NTT HILabs) for faces data, Ron Kikinis (Brigham and Women's Hospital of Boston) for brains data and Court Cutting (New-York University Medical Center) for skulls data. This work was supported in part by a grant from **Digital Equipment Corporation** and by the European Basic Research Action **VIVA** (Esprit project).

Appendices

A Computing the curvatures and the principal directions

We just give here the formulas to compute the curvatures and the principal directions of a parametric surface S. See [dC76] for a detailed presentation. Let $\mathbf{s}(u,v)$ be a parameterization of the surface S. E, F, G the three coefficients of the first fundamental form expressed in the basis $\{\mathbf{s}_u, \mathbf{s}_v\}$ of the tangent plane, and e, f, g the three coefficients of the second fundamental form are:

$$E = \mathbf{s}_u.\mathbf{s}_u, \quad F = \mathbf{s}_u.\mathbf{s}_v, \quad G = \mathbf{s}_v.\mathbf{s}_v, \quad e = \mathbf{N}.\mathbf{s}_{uu}, \quad f = \mathbf{N}.\mathbf{s}_{uv}, \quad g = \mathbf{N}.\mathbf{s}_{vv}$$

where

$$\mathbf{N} = \frac{\mathbf{s}_u \wedge \mathbf{s}_v}{\|\mathbf{s}_u \wedge \mathbf{s}_v\|}$$

[11] is the normal of the surface. The curvatures k_1 and k_2 are such that $-k1$ and $-k_2$ are the eigenvalues of the matrix

$$dN = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{3}$$

expressed in the basis $\{\mathbf{s}_u, \mathbf{s}_v\}$ where

$$a_{11} = \frac{fF - eG}{EG - F^2}, \quad a_{12} = \frac{gF - fG}{EG - F^2}, \quad a_{21} = \frac{eF - fE}{EG - F^2}, \quad a_{22} = \frac{fF - gE}{EG - F^2}.$$

[11] where "$\wedge$" denotes the usual cross product

In practice, we first compute K_g and K_m the Gaussian and the mean curvature

$$K_g = k_1 k_2 = \frac{eg - f^2}{EG - F^2}, \qquad K_m = \frac{1}{2}(k_1 + k_2) = \frac{1}{2}\frac{eG - 2fF + gE}{EG - F^2}$$

and then we compute

$$k_1 = K_m + (H_m^2 - K_g)^{1/2}, \qquad k_2 = K_m - (H_m^2 - K_g)^{1/2}$$

Let (α_1, β_1) (resp. (α_2, β_2)) be the coordinates in the basis $\{\mathbf{s}_u, \mathbf{s}_v\}$ of the eigenvector of dN corresponding to the eigenvalue k_1 (resp. k_2). The principal directions are

$$\vec{e}_1 = \alpha_1 s_u + \beta_1 s_v, \qquad \vec{e}_2 = \alpha_2 s_u + \beta_2 s_v$$

B Computing the curvatures and principal frames of the transformed surface

In this appendix, we want to show that:
proposition 1:
When a surface S is transformed into a surface $\mathbf{A}S + \mathbf{b}$ by an affine transformation $(\mathbf{A}, \mathbf{b})$, the principal frame and the curvatures at point $\mathbf{A}M + \mathbf{b}$ on $\mathbf{A}S + \mathbf{b}$ depend only on the principal frame and the curvatures at point M on S.

Given a point M on S with principal curvatures k_1, k_2 and principal frame $(M, \vec{e}_1, \vec{e}_2, \vec{n})$ and given an affine transformation $(\mathbf{A}, \mathbf{b})$, we want to compute the new curvatures k_1', k_2' and the principal frame $(M', \vec{e'}_1, \vec{e'}_2, \vec{n'})$ at point $M' = \mathbf{A}M + \mathbf{b}$ on $\mathbf{A}S + \mathbf{b}$. Assume that M is not an umbilic point. There exists a parameterization $\mathbf{s}(u, v)$ of S in a neighborhood V of M such that the coordinates curves $u = \text{const.}$, $v = \text{const.}$ are the lines of curvature of S (see [dC76]). We use this parameterization and we note $M = \mathbf{s}(u_0, v_0)$. We note too $\mathbf{s}_u = \partial\mathbf{s}/\partial u$ and $\mathbf{s}_v = \partial\mathbf{s}/\partial v$. E, F, G are the three coefficients of the first fundamental form and e, f, g the three coefficients of the second fundamental form. In this parameterization, we have [GA94a]:

$$\vec{e}_1 = \frac{\mathbf{s}_u}{\sqrt{E}}, \qquad k_1 = \frac{e}{E}, \qquad \vec{e}_2 = \frac{\mathbf{s}_v}{\sqrt{G}}, \qquad k_2 = \frac{g}{G}$$

and

$$\begin{cases} \mathbf{s}_{uu} = \frac{E_u}{2E}\,\mathbf{s}_u - \frac{E_v}{2G}\,\mathbf{s}_v + e\,\vec{n} \\ \mathbf{s}_{uv} = \frac{E_v}{2E}\,\mathbf{s}_u + \frac{G_u}{2G}\,\mathbf{s}_v \\ \mathbf{s}_{vv} = -\frac{G_u}{2E}\,\mathbf{s}_u + \frac{G_v}{2G}\,\mathbf{s}_v + g\,\vec{n} \end{cases}$$

To simplify the calculus, we first remark that it is possible to have a parameterization $\mathbf{s}_1(u,v)$ of S in a neighborhood V of M such that the coordinate curves are the lines of curvature and such that $\mathbf{s}_1(u_0,v_0) = M$, $E_1(u_0,v_0) = 1$ and $G_1(u_0,v_0) = 1$ where E_1, F_1, G_1 are the new coefficient of the first fundamental form. We just let $\mathbf{s}_1(u,v) = \mathbf{s}(\lambda u+\alpha, \gamma v+\beta)$ with $\lambda = 1/\|\mathbf{s}_u(u_0,v_0)\|$, $\gamma = 1/\|\mathbf{s}_v(u_0,v_0)\|$, $\alpha = (1-\lambda)u_0$ and $\beta = (1-\gamma)v_0$.

Now, letting $\mathbf{t}(u,v) = \mathbf{A}\mathbf{s}_1(u,v) + \mathbf{b}$, we obtain a parameterization $\mathbf{t}(u,v)$ of the transformed surface $\mathbf{A}S + \mathbf{b}$ in a neighborhood V of $\mathbf{A}M + \mathbf{b}$. In this parameterization, we have:

$$\begin{cases} \mathbf{t}_u(u_0,v_0) = \mathbf{A}\vec{e}_1, \qquad \mathbf{t}_v(u_0,v_0) = \mathbf{A}\vec{e}_2 \\ \mathbf{t}_{uu}(u_0,v_0) = \mathbf{A}\mathbf{s}_{1uu}(u_0,v_0), \qquad \mathbf{t}_{uv}(u_0,v_0) = \mathbf{A}\mathbf{s}_{1uv}(u_0,v_0), \qquad \mathbf{t}_{vv}(u_0,v_0) = \mathbf{A}\mathbf{s}_{1vv}(u_0,v_0) \end{cases}$$

Computing $e'(u_0,v_0)$, $f'(u_0,v_0)$, $g'(u_0,v_0)$ the three coefficients of the second fundamental form in the parameterization $\mathbf{t}(u,v)$, we obtain:

$$\begin{aligned} e'(u_0,v_0) &= \mathbf{A}\mathbf{s}_{1uu}(u_0,v_0).\frac{(\mathbf{A}\vec{e}_1 \wedge \mathbf{A}\vec{e}_2)}{\|\mathbf{A}\vec{e}_1 \wedge \mathbf{A}\vec{e}_2\|} \\ &= [\mathbf{A}\mathbf{s}_{1uu}(u_0,v_0), \mathbf{A}\vec{e}_1, \mathbf{A}\vec{e}_2]/\|\mathbf{A}\vec{e}_1 \wedge \mathbf{A}\vec{e}_2\| \\ &= det(\mathbf{A})[\mathbf{s}_{1uu}(u_0,v_0), \vec{e}_1, \vec{e}_2]/\|\mathbf{A}\vec{e}_1 \wedge \mathbf{A}\vec{e}_2\| \end{aligned}$$

Because we have $\mathbf{s}_{1uu}(u_0,v_0) = \dfrac{E_{1u}(u_0,v_0)}{2}\vec{e}_1 - \dfrac{E_{1v}(u_0,v_0)}{2}\vec{e}_2 + k_1(\vec{e}_1 \wedge \vec{e}_2)$, we have $[\mathbf{s}_{1uu}(u_0,v_0), \vec{e}_1, \vec{e}_2] = k_1$. Finally:

$$e'(u_0,v_0) = \frac{det(\mathbf{A})k_1}{\|\mathbf{A}\vec{e}_1 \wedge \mathbf{A}\vec{e}_2\|}.$$

In the same way, we can show that:

$$f'(u_0,v_0) = 0, \qquad g'(u_0,v_0) = \frac{det(\mathbf{A})k_2}{\|\mathbf{A}\vec{e}_1 \wedge \mathbf{A}\vec{e}_2\|}$$

E', F', G' the three coefficients of the first fundamental form are:

$$E' = \mathbf{A}\vec{e}_1.\mathbf{A}\vec{e}_1, \qquad F' = \mathbf{A}\vec{e}_1.\mathbf{A}\vec{e}_2, \qquad G' = \mathbf{A}\vec{e}_2.\mathbf{A}\vec{e}_2$$

Because the new principal frame $(\vec{e'}_1, \vec{e'}_2, \vec{n'})$ and k'_1, k'_2, the principal curvatures only depend on the coefficients of the two fundamental forms (see appendix A) and because these coefficients only depend on $\mathbf{A}$, k_1, k_2, $\vec{e}_1$ and $\vec{e}_2$, the new principal frame and the new curvatures only depend on $\mathbf{A}, k_1, k_2, \vec{e}_1$ and $\vec{e}_2$. We have developed the formulae, but we do not explicit them here, as they are quite straightforward.

C Minimization of the new criterion using EKF

We now explain how we minimize the new criterion 2. Assume that $Match_i$ is perfect and that there exists an affine transformation $(\mathbf{A}, \mathbf{b})$ which superposes the two surfaces. For each (M_k, N_k) in $Match_i$, we can write the following equation:

$$(x'', y'', z'', k_1'', k_2'')_k^t - \mathbf{g}((x, y, z, k_1, k_2, \vec{e}_1, \vec{e}_2)_k, \mathbf{A}, \mathbf{b})) = \mathbf{0}. \tag{4}$$

Each equation (4) can be rewritten in

$$\mathbf{f}_k(\mathbf{x}_k, \mathbf{a}) = \mathbf{0} \tag{5}$$

where $x_k = (x, y, z, k_1, k_2, x'', y'', z'', k_1'', k_2'')$ and $\mathbf{a}$ is the vector of coefficients of matrices $\mathbf{A}$ and $\mathbf{b}$. In reality, we cannot write equation (4) for all pairs (M_k, N_k) in $Match_i$. But we can assume that this is because each pair corresponds to a measure $\hat{\mathbf{x}}_k$ of $\mathbf{x}_k$ corrupted with a random error $\mathbf{v}_k$:

$$\hat{\mathbf{x}}_k = \mathbf{x}_k + \mathbf{v}_k, \quad E(\mathbf{v}_k) = \mathbf{0}, \quad E(\mathbf{v}_k \mathbf{v}_k^t) = \mathbf{\Lambda}_k \geq \mathbf{0}, \quad E(\mathbf{v}_i \mathbf{v}_j^t) = \mathbf{0} \quad \forall i \neq j \tag{6}$$

Given the measures $\hat{\mathbf{x}}_k$, the EKF allows us to recursively compute an estimate $\hat{\mathbf{a}}_k$ of $\mathbf{a}$ and the associated covariance matrix $\mathbf{S}_k$ ($\hat{\mathbf{a}}_k = \mathbf{a} + \mathbf{w}_k$, where $\mathbf{w}_k$ is a random error: $E(\mathbf{w}_k) = \mathbf{0}$ and $E(\mathbf{w}_k \mathbf{w}_k^t) = \mathbf{S}_k \geq \mathbf{0}$). Only an initial estimate $(\hat{\mathbf{a}}_0, \mathbf{S}_0)$ is required. See the book of Nicholas Ayache [Aya91] for details. In summary, each nonlinear equation (5) is linearized around $(\hat{\mathbf{x}}_k, \hat{\mathbf{a}}_{k-1})$:

$$\mathbf{f}_k(\mathbf{x}_k, \mathbf{a}) = \mathbf{0} \approx \mathbf{f}_k(\hat{\mathbf{x}}_k, \hat{\mathbf{a}}_{k-1}) + \widehat{\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}}(\mathbf{x}_k - \hat{\mathbf{x}}_k) + \widehat{\frac{\partial \mathbf{f}_k}{\partial \mathbf{a}}}(\mathbf{a}_k - \hat{\mathbf{a}}_k) \tag{7}$$

where $\widehat{\partial \mathbf{f}_k / \partial \mathbf{x}}$ and $\widehat{\partial \mathbf{f}_k / \partial \mathbf{a}}$ are estimated at $(\hat{\mathbf{x}}_k, \hat{\mathbf{a}}_{k-1})$. Equation (7) can be rewritten as:

$$\mathbf{y}_k = \mathbf{M}_k \mathbf{a} + \mathbf{u}_k \tag{8}$$

where:

$$\mathbf{y}_k = -\mathbf{f}_k(\hat{\mathbf{x}}_k, \hat{\mathbf{a}}_{k-1}) + \widehat{\frac{\partial \mathbf{f}_k}{\partial \mathbf{a}}} \hat{\mathbf{a}}_{k-1}, \quad \mathbf{M}_k = \widehat{\frac{\partial \mathbf{f}_k}{\partial \mathbf{a}}}, \quad \mathbf{u}_k = \widehat{\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}}(\mathbf{x}_k - \hat{\mathbf{x}}_k)$$

Equation (8) is now a linear measurement equation where $\mathbf{u}_k$ is a random error: $E(\mathbf{u}_k) = 0$ and $E(\mathbf{u}_k \mathbf{u}_k^t) = \mathbf{W}_k \geq \mathbf{0}$. $\mathbf{W}_k$ is the covariance matrix of the random measurement error and is easy to compute:

$$\mathbf{W}_k = \widehat{\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}} \mathbf{\Lambda}_k \widehat{\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}}^t$$

We can now use the linear Kalman filter formalism and the minimized criterion is :

$$C = (\mathbf{a} - \hat{\mathbf{a}}_0)^t \mathbf{S}_0^{-1} (\mathbf{a} - \mathbf{a}_0) + \sum_{(M,N) \in Match_i} (\mathbf{y}_k - \mathbf{M}_k \mathbf{a})^t \mathbf{W}_k^{-1} (\mathbf{y}_k - \mathbf{M}_k \mathbf{a}) \quad (9)$$

Taking $\mathbf{S}_0^{-1} \approx \mathbf{0}$, C is exactly the criterion we want to minimize. Hence in practice, we just have to determine $(\hat{\mathbf{a}}_0, \mathbf{S}_0)$ and $\mathbf{\Lambda}_k$ to minimize the criterion (2). The initial estimate $\hat{\mathbf{a}}_0$ simply corresponds to the rotation computed as explained in section 2. $\mathbf{S}_0$ is a diagonal matrix with a very large value. $\mathbf{\Lambda_k}$ depends on the noise we estimate on the data but is not critical: we just choose a diagonal matrix, each non-zero coefficient depending on the weight we desire for the associated coordinate in the criterion. Finally, to take the coefficients p_k into account, we just have to multiply each covariance matrix $\mathbf{\Lambda}_k$ by $1/\sqrt{p}_k$.

References

[Aya91] N. Ayache. *Artificial Vision for Mobile Robots — Stereo-Vision and Multisensory Perception.* Mit-Press, 1991.

[Aya93] N. Ayache. Analysis of three-dimensional medical images - results and challenges. Technical Report 2050, INRIA, 1993.

[BCA94] E. Bardinet, L. Cohen, and N. Ayache. Fitting 3-d data using superquadrics and free-form deformations. Technical report, INRIA, 1994. To appear. Submitted to ICPR'94.

[BK89] Ruzena Bajcsy and Stane Kovacic. Multiresolution elastic matching. *CVGIP*, 46:1–21, 1989.

[BM92] Paul Besl and Neil McKay. A method for registration of 3–D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[Bro92] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–375, December 1992.

[CAS92] I. Cohen, N. Ayache, and P. Sulger. Tracking points on deformable objects using curvature information. In *Proceedings of the Second European Conference on Computer Vision 1992*, Santa Margherita Ligure, Italy, May 1992.

[CLSB92] G. Champleboux, S. Lavallée, R. Szeliski, and L. Brunie. From accurate range imaging sensor calibration to accurate model-based 3–D object localization. In *Proceedings of the IEEE Conference on Vision and Pattern Recognition*, Urbana Champaign, June 1992.

[CM92] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

[Coh94] L. Cohen. Use of auxiliary variables in computer vision problems. *Cahiers de mathematiques de la décision*, (9409), February 1994.

[Dan80] P.E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[dC76] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, Englewood Cliffs, 1976.

[Del94] H. Delingette. Simplex meshes : a general representation for 3d shape reconstruction. In *IEEE Conference on Vision and Pattern Recognition*, Seattle, June 1994.

[DOSA91] J.S. Duncan, R.L. Owen, L.H. Staib, and P. Anandan. Measurement of non-rigid motion using contour shape descriptor. In *CVPR'91*, Hawaii, 1991.

[FH86] O. Faugeras and M. Hébert. The representation, recognition and locating of 3*d* objects. *Int. J. Robotics Res*, 5(3):27–52, 1986.

[GA94a] A. Gourdon and N. Ayache. Matching a curve on a surface using differential properties. In *Proceedings of the Third European Conference on Computer Vision 1994*, Stockholm, Sweden, May 1994.

[GA94b] A. Guéziec and N. Ayache. Smoothing and matching of 3−D-space curves. *Int. Journal of Computer Vision*, 12(1), February 1994.

[Gri90] W. Grimson. *Object Recognition by Computer: The role of geometric constraints.* MIT Press, 1990.

[Gué93] André Guéziec. Large deformable splines, crest lines and matching. In *Proceedings of the Fourth International Conference on Computer Vision (ICCV93)*, Berlin, June 1993.

[Hos92] M. Hosaka. *Modeling of Curves and Surfaces in CAD /CAM.* Springer Verlag, Berlin, 1992.

[ML92] Yau H.-T. Menq, C.-H. and G.-Y. Lai. Automated precision measurement of surface profile in cad-directed inspection. *IEEE Trans. RA*, 8(2):268–278, 1992.

[MR92] G. Malandain and J.M. Rocchisani. Registration of 3−D medical images using a mechanical based method. In *Proceedings of the Fourteenth Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS 92), Satellite Symposium on 3−D Advanced Image Processing in Medicine*, Rennes, France, November 1992.

[MT91] Dimitri Metaxas and Demetri Terzopoulos. Constrained deformable superquadrics and nonrigid motion tracking. In *IEEE Proceedings of Computer Vision and Pattern Recognition*, pages 337–343. IEEE Computer Society Conference, June 1991. Lahaina, Maui, Hawaii.

[MT93] D. McInerney and T. Terzopoulos. A finite element model for 3d shape reconstruction and non rigid motion tracking. In *Proceedings of the Fourth International Conference on Computer Vision (ICCV '93)*, Berlin, May 1993.

[NA93] Chahab Nastar and Nicholas Ayache. Fast segmentation, tracking, and analysis of deformable objects. In *Proceedings of the Fourth International Conference on Computer Vision (ICCV '93)*, Berlin, May 1993.

[PS85] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry, an Introduction*. Springer Verlag, 1985.

[PS91] Alex Pentland and Stan Sclaroff. Closed-form solutions for physically based shape modelling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(7):715–729, July 1991.

[SL94] Richard Szeliski and Stéphane Lavallée. Matching 3-d anatomical surfaces with non-rigid volumetric deformations. In *AAAI 1994 Spring Symposium Series. Application of Computer Vision in Medical Image Processing*, Stanford University, March 1994.

[TG92] J.P. Thirion and A. Gourdon. The 3−D marching lines algorithm and its application to crest lines extraction. Technical Report 1672, INRIA, May 1992.

[Thi94] J-P. Thirion. New feature points based on geometric invariants for 3D image registration. In *IEEE Conference on Vision and Pattern Recognition*, Seattle, June 1994.

[Zha93] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. Journal of Computer Vision*, 1993. to appear, research report available at Inria, Sophia-Antipolis.

Unité de recherche INRIA Lorraine, Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399