The Alignment Between 3-D Data and Articulated Shapes with Bending Surfaces

Guillaume Dewaele, Frédéric Devernay, Radu Horaud, Florence Forbes

The Alignment Between 3-D Data and Articulated Shapes with Bending Surfaces

Guillaume Dewaele, Frédéric Devernay, Radu Horaud, and Florence Forbes

INRIA Rhône-Alpes, 655, avenue de l'Europe,
38330 Montbonnot Saint-Martin, France

Abstract. In this paper we address the problem of aligning 3-D data with articulated shapes. This problem resides at the core of many motion tracking methods with applications in human motion capture, action recognition, medical-image analysis, etc. We describe an articulated and bending surface representation well suited for this task as well as a method which aligns (or registers) such a surface to 3-D data. Articulated objects, e.g., humans and animals, are covered with clothes and skin which may be seen as textured surfaces. These surfaces are both articulated and deformable and one realistic way to model them is to assume that they bend in the neighborhood of the shape's joints. We will introduce a surface-bending model as a function of the articulated-motion parameters. This combined articulated-motion and surface-bending model better predicts the observed phenomena in the data and therefore is well suited for surface registration. Given a set of sparse 3-D data (gathered with a stereo camera pair) and a textured, articulated, and bending surface, we describe a register-and-fit method that proceeds as follows. First, the data-to-surface registration problem is formalized as a classifier and is carried out using an EM algorithm. Second, the data-to-surface fitting problem is carried out by minimizing the distance from the registered data points to the surface over the joint variables. In order to illustrate the method we applied it to the problem of hand tracking. A hand model with 27 degrees of freedom is successfully registered and fitted to a sequence of 3-D data points gathered with a stereo camera pair.

1 Introduction

In this paper we address the problem of aligning 3-D data to articulated shapes. This problem resides at the core of a variety of methods, including object localization, tracking, e.g., human-body motion capture, model-to-data registration, etc. The problem is difficult for a number of reasons. First of all, there is a lack of a general framework for representing large varieties of objects, such as humans and their body parts, animals, etc. Second it is difficult to predict the appearance of such objects such that the tasks of identifying them in images and of locating them become tractable. Third, since they have a large number of degrees of freedom, the problem of estimating their pose is confronted with a difficult optimization problem that can be trapped in local minima.

A. Leonardis, H. Bischof, and A. Pinz (Eds.): ECCV 2006, Part III, LNCS 3953, pp. 578–591, 2006.

A first class of methods addresses the problem of articulated object tracking [1]. A human motion tracker, for example, uses a previously estimated pose as a prior to predict the current pose and to update the model's parameters [2], [3], [4], [5], [6]. Objects may have large motion amplitudes between two video frames and their aspect may drastically change as one body part is occluded by another one or when it turns away from the camera's field of view [7], [8]. Image data are often ambiguous and it is not easy to separate the tracked object from the background or from other moving objects.

A second class of methods addresses the problem of aligning (or registering) a point data set to a rigid or a deformable object. The data may lie at some distance from the object and the shape of the object may change over time. There are two classes of techniques available for solving this problem. The first class describes the object as a point data set and estimates the motion parameters using point-to-point assignments [9]. This type of methods works well provided that point-assignments (that may well be viewed as *hidden variables*) are properly established. The second class of techniques describes the object as a parameterized surface (or a curve) and fits the latter to the data [10], [2], [3]. This type of methods works well provided that the data are not too far from the object, that the data are evenly distributed around the object and that they are not corrupted by large-amplitude noise or outliers. Indeed, the distance from a datum to a surface (whether algebraic or Euclidean) is a non-linear function of the model's parameters and the associated non-linear optimization problem does not have a trivial solution.

In this paper we address both object tracking and object registration. One one side we consider a 3-D point data set, e.g., data gathered with a stereo camera pair. On the other side we consider articulated objects with their associated surface and we assume that this surface is textured. The surface itself is parameterized by the joint parameters associated with an underlying kinematic chain. The texture points are loosely attached to the kinematic chain such that when the surface bends, the texture points slightly *slide* along the surface. The amount of sliding is controlled by a set of *bending parameters* such that the texture sliding is proportional to the amount of surface bending.

We introduce an align-and-fit method that proceeds as follows. *Alignment*: The data points are associated with texture points. This point-to-point assignment is modelled as a classification problem. The texture points are viewed as classes and each data point is assigned to one of these classes. Data classification is carried out by an EM algorithm that performs three tasks: it classifies the points, it rejects outliers, and it estimates the joint parameters. *Fit*: The surface is fitted to the registered data points by minimizing a distance function over the joint parameters. An example is shown on Figure 1.

The methodology described in this paper has several contributions. We introduce an object representation framework that is well suited for describing articulated shapes with bending surfaces. We cast the data-to-model association problem into a classification problem. We show that it is more judicious, both from theoretical and practical points of view, to allow for one-to-many data-to-model

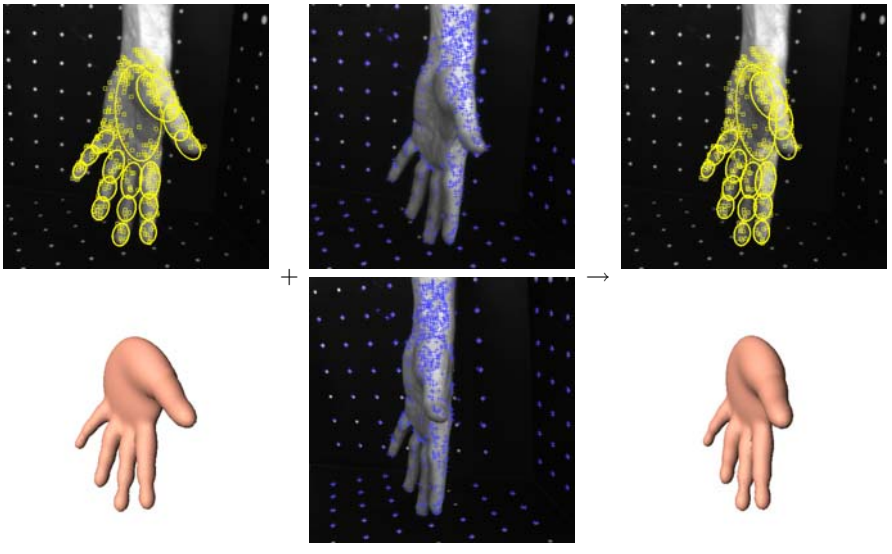

Fig. 1. This figure illustrates the alignment method described in this paper. *Left:* The articulated model (texture and surface) shown in its previously estimated pose, *Middle:* a set of 3-D data points obtained by stereo, and *Right:* the result of aligning the data (middle) with the model (left).

assignments, rather than pair-wise assignments, as is usually the case with most existing methods. We combine the advantages of point registration and of parameterized surface fitting. We emphasize that the *point-to-surface alignment* problem has primarily been addressed in the case of rigid and deformable objects, and has barely been considered in the case of articulated objects.

The idea of approximatively matching sets of points stems from [11]. [12]. These same authors propose a point matching algorithm able to deal with outliers [13]. However, they constrain the points to be assigned pair-wise which leads to some difficulties, as explained in section 4. The idea of using the EM algorithm [14] for solving the point matching problem was used by others [15], [16], [17]. However, previous work did not attempt to with articulated shapes. Moreover, the prolem of outlier rejection is not handled by their EM algorithms.

The remainder of this paper is organized as follows. Section 2 describes the articulated and bending surface model. Section 3 is a detailed account of the point-to-surface alignment method. In section 4, we compare our method with other similar methods and section 5 describes experiments performed with a 3-D hand tracker.

2 Articulated Shapes with Bending Surfaces

The object model that will be used throughout the paper has three main components:

- It has one or several kinematic chains linked to a common part, i.e., a base part. The kinematic joints are the model's parameters;
- A volumetric representation that describes the shape of each part of the kinematic chain as well as a surface embedding the whole object, and
- a texture that is described by a set of points loosely attached to the underlying surface, i.e., the texture is allowed to slide in the neighbourhood of the kinematic joints where the surface bends.

The kinematic model. A kinematic chain consists in P elementary parts which are referred by $\mathcal{Q}_1 ... \mathcal{Q}_P$ in this paper. These parts are linked by rotational joints. The motion of this chain can be described by a set of K parameters, one for each degree of freedom, $\mathbf{\Theta} = \{\theta_1, ..., \theta_K\}$. The first six parameters will correspond to the 3-D position and orientation of the chain's base-part, and the remaining parameters correspond to the joints angles. The position and orientation of each part $\mathcal{Q}_p$ is therefore determined by by $\mathbf{\Theta}$. The kinematically-constrained motion of such a multi-chain articulated structure is conveniently described by the rigid motions, $T_p(\mathbf{\Theta})$, of its parts.

A hand, for example, can be described with 5 such kinematic chains, and a total of 16 parts and 27 degrees of freedom. The base-part (the palm) has six degrees of freedom which correspond to the free motion of the hand. There are five degrees of freedom for the thumb and four degrees of freedom for the other fingers. The thumb has two joints with two rotational degrees of freedom and one joint with one degree of freedom, while the other fingers have one joint with two degrees of freedom and two joints with one degree of freedom.

The 3-D shape model. We will associate a rigid shape to each elementary part $\mathcal{Q}_p$ of the kinematic chain. In principle, it is possible to use a large variety of shapes, such as truncated cylinders or cones, quadrics, superquadrics, and so forth. One needs to have in mind that it is important to efficiently compute the distance $d(\mathbf{X}, \mathcal{Q}_p)$ from a point $\mathbf{X}$ to such a part.

In order to obtain a single and continuous surface describing the whole object, we will fuse these parts into a single one using isosurfaces. Isosurfaces are a very useful tool to create complex shapes, such as human-body parts, and have already been used in computer vision [10, 2]. To build this isosurface we will first associate a field function ϕ_p to each part $\mathcal{Q}_p$ of the model:

$$\phi_p(\mathbf{X}) = e^{-\frac{d(\mathbf{X}, \mathcal{Q}_p)}{\nu}} \tag{1}$$

The surface associated to the part $\mathcal{Q}_p$ can be described by the implicit equation $\phi_p(\mathbf{X}) = 1$. The fusion of the parts $\mathcal{Q}_p$ is obtained simply by summing up the field functions in one single field function $\phi(\mathbf{X}) = \sum_p \phi_p(\mathbf{X})$, and by considering the surface $\mathcal{S}$ defined by $\phi(\mathbf{X}) = 1$. The coefficient ν allows some amount of control of the fusion. In practice the summation will not be carried out over all the elements $\mathcal{Q}_p$. Indeed, a blind summation over all the elementary parts will have the tendency to fuse them whenever they are close to each other, even if they are not adjacent within the kinematic chain. For example, one doesn't want

two touching fingers to be glued together. To prevent this effect, whenever we determine the field at a surface point $\mathbf{X}$, e.g., $\phi(\mathbf{X})$, we look for the part $\mathcal{Q}_p$ that corresponds to the most important contribution $\phi_p(\mathbf{X})$, and we only consider the contribution of this part and the adjacent parts in the kinematic chain.

The distance between a point $\mathbf{X}$ and a part $\mathcal{Q}_p$ is evaluated through the formula $d(\mathbf{X}, \mathcal{Q}_p) = -\nu\ log(\phi_p(\mathbf{X}))$ Similarly, the distance from a point $\mathbf{X}$ to the isosurface $\mathcal{S}$ will be evaluated through the formula:

$$d(\mathbf{X}, \mathbf{\Theta}) = -\nu\ log(\phi(\mathbf{X})) \tag{2}$$

This distance depends on the position of the point $\mathbf{X}$ and on the parameters $\mathbf{\Theta} = \{\theta_k\}_{1 \leq k \leq K}$. For points close to the surface one may take the first order Taylor expansion of the above expression, so that the quantity $-\nu(\phi_p(\mathbf{X}) - 1)$ is a good approximation of $d(\mathbf{X}, \mathbf{\Theta})$.

The hand model uses ellipsoids as basic volumes to describe each part. Ellipsoids are relatively simple objects and correspond quite precisely to the shape of the fingers' phalanxes. To estimate the distance from a point to an ellipsoid, one may use the algebraic distance, i.e., the quadratic form $\mathbf{X}^\top \mathcal{Q}_p \mathbf{X}$. Estimation of the Euclidean distance from a point to an ellipsoid requires to solve for a six degree polynomial. In practice we will use a pseudo-Euclidean distance which is more efficiently computed than the Euclidean distance, [7]. Finally, by fusing these ellipsoids together, we are able to build a realistic model of the hand.

Modelling bending surfaces. The description detailed so far yields a surface model that is parameterized by the kinematic-joint parameters. Nevertheless, this model contains no information about how to control the non-rigid behaviour of the texture lying onto the object's surface, when the joint parameters vary. Consider again the case of the hand. In between two phalanxes the skin will stretch on one side of the finger and will be compressed on the opposite side. In other terms, the skin will slide along the underlying bones whenever the finger is bent. This is a non-rigid motion and therefore this phenomenon will not be properly modelled if surface points are rigidly attached to a part $\mathcal{Q}_p$. We will use an approach similar to skinning techniques in graphics.

The motion of a point $\mathbf{X}$, that lies nearby the surface of the object, is modelled as a linear combination of the rigid motions T_p and T_r of two adjacent parts $\mathcal{Q}_p$ and $\mathcal{Q}_r$; We will use the field function $\phi_p(\mathbf{X_i})$ for weighting the contributions of these two motions:

$$\mathcal{T}(\mathbf{X}, \mathbf{\Theta}) = \frac{(\phi_p(\mathbf{X}, \mathbf{\Theta}))^a T_p(\mathbf{\Theta})\mathbf{X} + (\phi_r(\mathbf{X}, \mathbf{\Theta}))^a T_r(\mathbf{\Theta})\mathbf{X}}{(\phi_p(\mathbf{X}, \mathbf{\Theta}))^a + (\phi_r(\mathbf{X}, \mathbf{\Theta}))^a} \tag{3}$$

The parameter a allows to adjust the sliding when the distances to both $\mathcal{Q}_p$ and $\mathcal{Q}_r$ are of comparable value. A small value for a yields a nicely interpolated motion while the points remain attached to either $\mathcal{Q}_p$ or to $\mathcal{Q}_r$ for a large a. A few examples of how the sliding is controlled in this way are shown on figure 2. For the hand, we use $a = 2$.

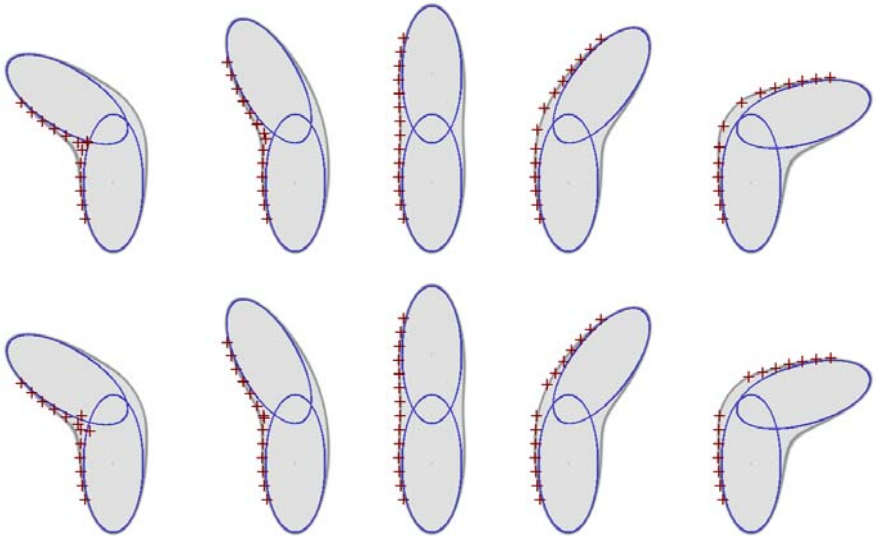

Fig. 2. The sliding of the texture onto a bending surface is controlled by the parameter a. For $a = 2$ (top) the texture nicely compresses and streches when the surface bends. For $a = 10$ (bottom) the texture is rigidly attached to the nearest rigid part.

3 3-D Point-to-Surface Alignment

The input data of our method consist in 3-D points extracted at time t and we denote these points by $\{\mathbf{Y_j}\}_{1\leq j\leq m}$. The input model consists in an articulated surface. We want to estimate the articulated motion of this surface from its pose at time $t - dt$ to the its pose at time t. Moreover, the surface has a texture associated with it and the latter is desribed as a set of 3-D points that we denote by $\{\mathbf{X_i}\}_{1\leq i\leq n}$. These points correspond to data points that were correctly fitted to the surface at $t - dt$, i.e., the distance from eachone of these points to the surface falls below a threshold ε. The latter depends on the accuracy with which one wants to estimate the motion parameters, the accuracy of the data, and the accuracy of the model. For the hand-tracking experiments we use $\varepsilon = 5mm$.

3.1 3-D Point Matching Via Classification

Given a set of *model* points $\{\mathbf{X_i}\}_{1\leq i\leq n}$ and a set of *data* points $\{\mathbf{Y_j}\}_{1\leq j\leq m}$, the goal is to estimate the transformation $\mathcal{T}$. Therefore, one needs to find correspondences between the priors and the candidates. This is known in statistics as a *missing data problem* that can be formulated as a classification problem. Each point $\mathbf{Y_j}$ should be either associated to one (or several) of the points $\mathbf{X_i}$, or be rejected as an outlier. Therefore, the $\mathbf{X_i}$'s can be seen as classes into which the $\mathbf{Y_j}$'s have to be classified or thrown out.

To recast the problem in a statistical framework, the $\{\mathbf{Y_j}\}_{1\leq j\leq m}$ are the observed values of a set of random variables $\{\mathcal{Y}_j\}_{1\leq j\leq m}$. Each $\mathcal{Y}_j$ is associated

to another random variable $\mathcal{Z}_j$, whose specific value is unknown, which represents the point associated to $\mathcal{Y}_j$ when $\mathcal{Y}_j = \mathbf{Y_j}$. More specifically, $\mathcal{Z}_j$ is equal to one of the points $\mathbf{X_i}$, or fall into a special additionnal "outlier" class when the observed point $\mathbf{Y_j}$ does not correspond to any point among the $\mathbf{X_i}$'s.

Let us specify some additionnal entities that will be needed. The prior probabilities $\{\Pi_i\}_{1 \leq i \leq n+1}$ of each of the classes $\{\mathbf{X_1}, ..., \mathbf{X_n}, outlier\}$, and probability distribution $f_j(\mathbf{Y_j}|\mathcal{Z}_j = \mathbf{X_i})$ of a point $\mathbf{Y_j}$ when its class $\mathcal{Z}_j$ is known and equal to $\mathbf{X_i}$, or $f_j(\mathbf{Y_j}|\mathcal{Z}_j = outlier)$ when $\mathbf{Y_j}$ cannot be associated to any of the $\mathbf{X_i}$'s.

The prior probability, for a point in a working space of volume V, to be associated to a given point $\mathbf{X_i}$ corresponds to the probability of finding it in a volume $v < V$ around the new position of the point, $\mathcal{T}(\mathbf{X_i})$. Therefore, we have

$$\Pi_i = \frac{v}{V}. \tag{4}$$

The prior probability for a point to be an outlier (meaning that the point is not in any of the small volumes v) will then be:

$$\Pi_{outlier} = \frac{V - nv}{V}. \tag{5}$$

The probability distribution for $\mathcal{Y}_j$ can also be written quite easily. If this point $\mathcal{Y}_j$ corresponds to a point $\mathbf{X_i}$, its position $\mathbf{Y_j}$ should be close to the new position of this point (after the motion), $\mathcal{T}(\mathbf{X_i})$. We will choose a Gaussian distribution $\mathcal{N}$, centered in $\mathcal{T}(\mathbf{X_i})$ with variance σ^2:

$$f_j(\mathbf{Y_j}|\mathcal{Z}_j = \mathbf{X_i}) = \mathcal{N}(\mathbf{Y_j}; \mathcal{T}(\mathbf{X_i}), \sigma^2) \tag{6}$$

If the point $\mathcal{Y}_j$ is an outlier, this probability distribution should be uniform over the working space. So, if V is the volume of the working space,

$$\forall\ \mathbf{Y_j} \in V,\ \ f_j(\mathbf{Y_j}|\mathcal{Z}_j = outlier) = \frac{1}{V} \tag{7}$$

Using Bayes' formula, we can then write the distribution of the random variable $\mathcal{Y}_j$, which is :

$$f_j(\mathbf{Y_j}) = \sum_{i=1}^{n} \Pi_i\ f_j(\mathbf{Y_j}|\mathcal{Z_j} = \mathbf{X_i}) + \Pi_{outlier}\ f_j(\mathbf{Y_j}|\mathcal{Z_j} = outlier) \tag{8}$$

or, by specifying the different terms,

$$f_j(\mathbf{Y_j}) = \sum_{i=1}^{n} \frac{v}{V}\, \mathcal{N}(\mathbf{Y_j}; \mathcal{T}(\mathbf{X_i}), \sigma^2)\ +\ \frac{V - nv}{V^2} \tag{9}$$

For the volume v, we will use a sphere of radius σ, so that we have $v = 4\pi\sigma^3/3$. We can choose a working space large enough so that the quantity nv is negligeable when compared to V. We will use this remark later for simplification.

3.2 A Robust EM Algorithm

To simplify the notations, we will denote by Ψ the set of unknown parameters, i.e., the joint variables $\mathbf{\Theta}$ defining the transformation $\mathcal{T}$ and the variance σ. We seek the parameters Ψ which maximize the likelihood $\mathcal{P}(\mathbf{Y}|\Psi)$, where the $\mathbf{Y_j}$'s are the measured positions of the points $\{\mathcal{Y}_j\}_{1\leq j\leq m}$.

Due to the unknowns $\mathcal{Z}_j$, this is a *missing data problem* and the EM algorithm will be used. In addition of providing estimates of Ψ, the algorithm will provide values for the unknown $\mathcal{Z}_j$'s. We will use the following notation:

$$\alpha_{i,j} = \mathcal{P}(\mathcal{Z}_\mathbf{j} = \mathbf{X_i}|\mathbf{Y}, \Psi) \tag{10}$$

Therefore, $\alpha_{i,j}$ denotes the probability that the class $\mathcal{Z}_j$ of the observed point $\mathbf{Y_j}$ corresponds to the prior point $\mathbf{X_i}$.

Starting with an initial guess Ψ^0, the EM algorithm proceeds iteratively and the iteration q consists in searching for the parameters Ψ that maximize the following term:

$$Q(\Psi|\Psi^q) = E\,[\,log(\mathcal{P}(\mathbf{Y}, \mathcal{Z}|\Psi))\mid \mathbf{Y}, \Psi^q\,] \tag{11}$$

where Ψ^q is the prior estimation of Ψ available at the current iteration q. The expectation is taken over all the values of $\mathcal{Z}$, thus taking into account all possible values of the $\mathcal{Z}_j$'s. This process will be iterated until convergence. We will first develop $Q(\Psi|\Psi^q)$ for the distributions (6) and (7) that we defined in the previous section. First, one can write that:

$$\begin{aligned}\mathcal{P}(\mathbf{Y}, \mathcal{Z}|\Psi) = \prod_{j=1}^{m}\Bigg(\prod_{i=1}^{n}(\Pi_i f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = \mathbf{X_i}, \mathcal{T}, \sigma))^{\delta_{\{\mathcal{Z}_\mathbf{j}=\mathbf{X_i}\}}} \\ \times\ (\Pi_{outlier} f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = outlier))^{\delta_{\{\mathcal{Z}_\mathbf{j}=outlier\}}}\Bigg)\end{aligned} \tag{12}$$

where $\delta_{\{\mathcal{Z}_\mathbf{j}=\mathbf{X_i}\}}$ (resp. $\delta_{\{\mathcal{Z}_\mathbf{j}=outlier\}}$) equals 1 when the class $\mathcal{Z}_j$ of the point $\mathcal{Y}_j$ is $\mathbf{X_i}$ (resp. is an outlier), and 0 otherwise. Note that in the second product, all terms but one are equal to 1. Therefore, we have:

$$\begin{aligned}log\,(\mathcal{P}(\mathbf{Y}, \mathcal{Z}|\Psi)) = \sum_{j=1}^{m}\Bigg(\sum_{i=1}^{n}(log\ \Pi_i + log\ f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = \mathbf{X_i}, \mathcal{T}, \sigma))\,\delta_{\{\mathcal{Z}_\mathbf{j}=\mathbf{X_i}\}} \\ +\ (log\ \Pi_{outlier} + log\ f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = outlier, \mathcal{T}, \sigma))\,\delta_{\{\mathcal{Z}_\mathbf{j}=outlier\}}\Bigg).\end{aligned} \tag{13}$$

By reporting the expression (13) in (11), we obtain :

$$\begin{aligned}Q(\Psi|\Psi^q) = \sum_{j=1}^{n}\Bigg(\sum_{i=1}^{m}(log\ \Pi_i + log\ f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = \mathbf{X_i}, \mathcal{T}, \sigma))\,\mathcal{P}(\mathcal{Z}_\mathbf{j} = \mathbf{X_i}|\mathbf{Y}, \Psi^q) \\ +\ (log\ \Pi_{outlier} + log\ f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = outlier, \mathcal{T}, \sigma))\,\mathcal{P}(\mathcal{Z}_\mathbf{j} = outlier|\mathbf{Y}, \Psi^q)\Bigg).\end{aligned} \tag{14}$$

This expression involves the probabilities $\alpha_{i,j}^q = \mathcal{P}(\mathcal{Z}_\mathbf{j} = \mathbf{X_i}|\mathbf{Y}, \Psi^q)$ which, using the Bayes formula, can be written as:

$$\alpha_{i,j}^q = \frac{\Pi_i f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = \mathbf{X_i}, \mathcal{T}^q, \sigma_q)}{\sum_{k=1}^n \Pi_k f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = \mathbf{X_k}, \mathcal{T}^q, \sigma_q) + \Pi_{outlier} f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = outlier)} \quad (15)$$

where

$$f_j(\mathbf{Y_j}|\mathcal{Z}_\mathbf{j} = \mathbf{X_i}, \mathcal{T}^q, \sigma_q^2) = \frac{1}{(2\pi\sigma_q^2)^{\frac{3}{2}}} e^{-\frac{\|\mathbf{Y_j} - \mathcal{T}^q(\mathbf{X_i})\|^2}{2\sigma_q^2}}. \quad (16)$$

The expression of $\alpha_{i,j}^q$, i.e., the probability that a point $\mathbf{Y_j}$ is matched with the point $\mathbf{X_i}$, can be further developed using expressions (4), (5) and (7):

$$\alpha_{i,j}^q = \frac{e^{-\frac{\|\mathbf{Y_j} - \mathcal{T}^q(\mathbf{X_i})\|^2}{2\sigma_q^2}}}{\sum_{k=1}^m e^{-\frac{\|\mathbf{Y_j} - \mathcal{T}^q(\mathbf{X_k})\|^2}{2\sigma_q^2}} + c} \quad (17)$$

where

$$c = 3\sqrt{\frac{\pi}{2}}\left(1 - \frac{nv^q}{V}\right) \simeq 3\sqrt{\frac{\pi}{2}} \text{ since we assumed } nv^q \ll V. \quad (18)$$

This expression for the $\alpha_{i,j}^q$ is at the core of the robust method (outlier rejection):

- If the transformation $\mathcal{T}^q$ does not take any point $\mathbf{X_i}$, i.e., the action $\mathcal{T}^q(\mathbf{X_i})$, in a small volume of radius σ_q around $\mathbf{Y_j}$ then all the terms in the denominator's sum will be small compared to c. Therefore, all the coefficients $\alpha_{i,j}^q$ corresponding to the point $\mathbf{Y_j}$ will lean towards zero. We will see in the following section that it will mean that the point $\mathbf{Y_j}$ will not be taken into account for the estimation of the transformation $\mathcal{T}$.
- On the contrary, a point $\mathbf{X_i}$ taken by $\mathcal{T}^q$ within the neighborhood of $\mathbf{Y_j}$ will yield a value for $\alpha_{i,j}^q$ that is close to 1 (if only one point $\mathbf{X_i}$ is present in this neighborhood), and the association between $\mathbf{X_i}$ and $\mathbf{Y_j}$ will be used to determine $\mathcal{T}$.

3.3 Estimating the Transformation $\mathcal{T}^q$

It is now possible to derive a simpler expression of equation (14). Neither the coefficients Π_i, nor the second row of eq. (14) depend on the transformation $\mathcal{T}$. By substituting f_i in (14) by its expression (16), we obtain a new expression for $Q(\Psi|\Psi^q)$:

$$Q(\Psi|\Psi^q) = -\frac{1}{2\sigma_q^2} \sum_{i=1}^n \sum_{j=1}^m \alpha_{i,j}^q \|\mathbf{Y_j} - \mathcal{T}(\mathbf{X_i}, \mathbf{\Theta})\|^2 + C^{st} \quad (19)$$

where the constant term does not depend on the transformation $\mathcal{T}$. To maximize Q, the EM algorithm iterates a two-step procedure:

– *Expectation:* The probabilities $\alpha_{i,j}^q$ are evaluated using equation (17).
– *Maximization:* We seek the parameters $\mathbf{\Theta}$ that maximize the criterion Q, given the $\alpha_{i,j}^q$'s previously evaluated. As it will be explained below, σ_q will not be evaluated through the maximization process. Its value will be evaluated through an annealing schedule.

These two steps are carried out until changes between $\mathcal{T}^q$ and $\mathcal{T}$ fall under a given threshold.

After developing the sum over j in equation (19) and after a few algebraic manipulation, one obtains that the maximization of (19) is equivalent to the minimization of:

$$E(\mathbf{\Theta}) = \sum_{i=1}^{n} \lambda_i^q \|\mathcal{T}(\mathbf{X_i}, \mathbf{\Theta}) - \mathbf{G_i^q}\|^2 \tag{20}$$

with:

$$\lambda_i^q = \sum_{j=1}^{m} \alpha_{i,j}^q \text{ and } \mathbf{G_i^q} = \frac{1}{\lambda_i^q} \sum_{j=1}^{m} \alpha_{i,j}^q \mathbf{Y_j} \ .$$

It is less complex and more efficient to minimize equation (20) rather than to maximize eq. x(19). Indeed, the summation is restricted to the priors $\mathbf{X_i}$. Moreover, it is straightforward to evaluate the Jacobian matrix associated with the transformation $\mathcal{T}$ given by equation (3). As the value of $E(\mathbf{\Theta})$ decreases, the model points $\mathbf{X_i}$ which do not have a data-point assignment will rapidly disappear from the summation since their associated coefficient λ_i rapidly converges to 0 as the value of σ_q decreases. On the contrary, for those priors $\mathbf{X_i}$ which do have *one or several* data-point assignments, their center of gravity, $\mathbf{G_i}$, will get closer to the data point $\mathbf{Y_j}$ which is the closest to $\mathcal{T}(\mathbf{X_i})$.

Conventionally, EM algorithms also include the evaluation of σ_q during the maximisation step. In fact, it has been observed by us and by other authors that this does not lead to good alignment results. Very often, the values of σ^q evaluated by the maximization step, decrease too quickly, and the algorithm tends to get trapped in a local minimum. To prevent this, we will simply specify an annealing schedule for σ_q, in the spirit of simulated annealing techniques. The initial value, σ_0, will be set at the threshold previously defined, σ_m, which corresponds to the largest allowed motion of a point. At each step of the algorithm, σ_q will decrease geometrically, according to $\sigma_{q+1} = \kappa\sigma_q$ with $\kappa < 1$. The value σ should not fall below the variance σ_r of the noise associatd with the 3-D locations of the points (a few millimeters in our case), and the decrease of σ_q is stopped when it reaches this threshold σ_r. The decrease rate κ is chosen so that it takes about five steps to go from σ_m downto σ_r.

3.4 Surface Fitting

Now that outliers were rejected and that the articulated object moved such that the inliers are in the neighbourhood of the object's surface, it is worthile to perform surface fitting. This introduces small corrections and it is useful to prevent drift over time. We will simply refine the position of the model (through

the parameters $\mathbf{\Theta}$) so that the 3-D points lie on the surface or as close as possible to this surface. The minimization criterium can be written as, [2], [7]:

$$E_s(\mathbf{\Theta}) = \sum_{j=1}^{m} \beta_j d(\mathbf{Y_j}, \mathbf{\Theta})^2 \tag{21}$$

where d is given by equation (2), and

$$\beta_j = e^{-\frac{d(\mathbf{Y_j})^2}{\sigma^2}} \tag{22}$$

4 Comparison with Other Methods

The main and fundamental difference between our method and other methods such as [11], [12], [13], [15], and [16] resides both in the expression of the probabilities $\alpha_{i,j}$ provided by equation (17) and in the fact that these probabilites are subject to the constraint $\sum_j \alpha_{i,j} = 1$ i.e., *a summation over the data points*. This constraint imposes that to each model point $\mathbf{X_i}$ corresponds a unique data point $\mathbf{Y_j}$ but a data point may correspond to several model points.

The SoftAssign method described in [11] treats the data- and the model-points symmetrically, and hence it needs the additional constraint $\sum_i \alpha_{i,j} = 1$ i.e., *a summation over the model points*. From a statistical point of view, it means that one needs to consider two non-independent classifiers. A formal derivation for the $\alpha_{i,j}$'s is more delicate. In [11] [12] it is suggested to normalize the $\alpha_{i,j}$'s, alternating over i and over j, until convergence, the latter being insured by the Sinkhorn theorem. However this is computationally expensive. In addition, since the data and the model are treated symmetrically, it is crucial to reject outliers both from the data and from the model. This is done by adding one row and one column to the matrix $\alpha_{i,j}$. For example, a data point $\mathbf{Y_j}$ which is an outlier will have a "1" entry in the extra row and a model point $\mathbf{X_i}$ which is an outlier will have a "1" entry in the extra column. The normalization along the rows and along the columns should not affect, however, these extra row and column. Indeed, there should be several outliers and therefore there should be several 1's in each one of these extra row and column. We implemented this method and noticed that when the normalization is not applied to these extra row and column, the final solution depends a lot upon the initial entries of these row and column. If the initial probabilities to have outliers are too high, all the points will be classified as outliers. If these initial probabilities are too low, there will be no outliers.

5 Experimental Results

The method described in this paper was applied to the problem of tracking a hand model with 3-D data. We used a calibrated stereo camera pair together with a stereo algorithm. This algorithm provides a dense disparity map. Texture

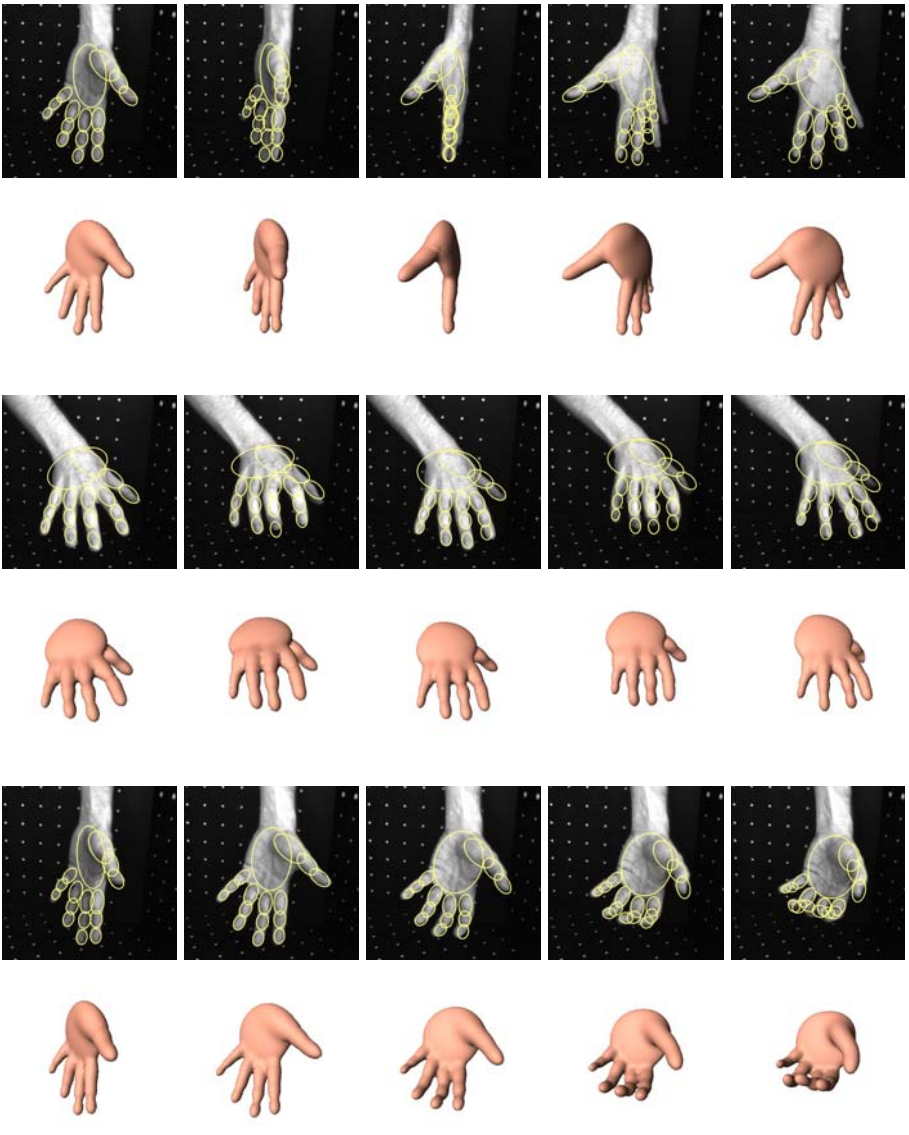

Fig. 3. Three different stereo video sequences of a hand. When the hand turns, the tracker has difficulties because the texture of the hidden side of the hand has not yet been included in the model. When the fingers bend, the hand may be tracked from both its sides.

points are extracted from both the left and right images using the Harris interest point detector. Since a dense disparity map is available, the texture points are easily matched and their 3-D positions estimated.

We gathered several stereoscopic video sequences at 20 frames per second. Each sequence has approximatively 100 frames. There are in between 500 and

1000 3-D data points associated with each stereo pair in the sequence. Notice that while some background points may be easily thrown out, there still remain many unrelevant points, such as those on the forearm (which is not modelled), e.g., Figure 1.

The results of applying the alignment method are shown on three different image sequences on Figure 3. In the first example, the hand rotates around an axis parallel to the image plane. In the next two examples the fingers bend and the hand is viewed from two different viewpoints.

The tracker maintains approximatively 250 inliers. Notice that the number of data points varies a lot as a function of the position of the hand with respect to the cameras. In particular, when the hand flips from one side to another, the tracker has to start the alignment from scratch because there are no model points with the side of the hand that has never been seen. Due to the fact that the hand-model has 27 degrees of freedom, it cannot capture all the hand's deformations.

6 Conclusion

In this paper we described a new method for aligning a set of 3-D data points to an articulated shape. We introduced a shape model that includes both an articulated kinematic representation and a surface-bending model. We described in detail an alignment method that robustly classifies data points as model points, i.e., points that lie onto the model's surface. The alignment problem was formalized as an EM algorithm that is able to reject outliers. The EM procedure that we described finds data-point-to-model-point assignments (expectation) and estimates the best transformation that maps the model points onto the data points (maximization). Our EM algorithm differs from previous attempts to solve for the point-registration problem: it has a built-in outlier rejection mechanism and it allows, one-to-many data classsifications (or data assignments). Relaxing the pair-wise assignment constraint results in a more efficient and more reliable alignment method. Also, we appear to be the first ones to apply EM-based point registration to complex articulated shapes.

In the near future we plan to address the problem of articulated shape matching, where there may be a large discrepancy between the pose parameters of the model and the unknown pose parameters to be estimated from the data. We plan to add a relational-graph representation of the model and to address the matching problem as the problem of both matching the model-graph to the data-graph and of finding the kinematic pose.

References

1. Gavrila, D.M.: The visual analysis of human movement: A survey. Computer Vision and Image Understanding **73** (1999) 82–98
2. Plaenkers, R., Fua, P.: Articulated soft objects for multi-view shape and motion capture. IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003)

3. Herda, L., Urtasun, R., Fua, P.: Hierarchical implicit surface joint limits for human body tracking. Computer Vision and Image Understanding **99** (2005) 189–209
4. Deutscher, J., Blake, A., Reid, I.: Articulated body motion capture by annealed particle filtering. In: Computer Vision and Pattern Recognition. (2000) 2126–2133
5. Sminchisescu, C., Triggs, B.: Estimating articulated human motion with covariance scaled sampling. International Journal of Robotics Research **22** (2003) 371–379
6. Delamarre, Q., Faugeras, O.: 3d articulated models and multi-view tracking with physical forces. Computer Vision and Image Understanding **81** (2001) 328–357
7. Dewaele, G., Devernay, F., Horaud, R.: Hand motion from 3d point trajectories and a smooth surface model. In Pajdla, T., Matas, J., eds.: 8th European Conference on Computer Vision. Volume I of LNCS 3021., Springer (2004) 495–507
8. Athitsos, V., Sclaroff, S.: Estimating 3D hand pose from a cluttered image. In: CVPR '03: Proceedings of Conference in Computer Vision and Pattern Recognition. (2003)
9. Zhang, Z.: Iterative point matching for registration of free-form curves and surfaces. International Journal on Computer Vision **13** (1994) 119–152
10. Ferrie, F.P., Lagarde, J., Whaite, P.: Darboux Frames, Snakes, and Super-Quadrics: Geometry from the Bottom Up. IEEE Transactions on Pattern Analysis and Machine Intelligence **15** (1993) 771–784
11. Gold, S., Rangarajan, A.: A Graduated Assignment Algorithm for Graph Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence **18** (1996) 377–388
12. Rangarajan, A., Chui, H., Bookstein, F.L.: The softassign procrustes matching algorithm. In: Information Processing in Medical Imaging (IPMI). (1997) 29–42
13. Chui, H., Rangarajan, A.: A new point matching algorithm for non-rigid registration. Computer Vision and Image Understanding **89** (2003) 114–141
14. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood estimation from incomplete data via the EM algorithm (with discussion). Journal of the Royal Statistical Society, Series B **39** (1977) 1–38
15. Cross, A.D.J., Hancock, E.R.: Graph Matching With a Dual-Step EM Algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 1236–1253
16. Luo, B., Hancock, E.: A unified framework for alignment and correspondence. CVIU **92** (2003) 26–55
17. Granger, S., Pennec, X.: Multi-scale em-icp: A fast and robust approach for surface registration. In: ECCV02. (2002) IV: 418 ff.