Appeared in the 2006 IEEE ICRA, Orlando, Florida, USA

# Towards 3D Motion Estimation From Deformable Surfaces

Adrien Bartoli
CNRS / LASMEA – Adrien.Bartoli@gmail.com
24, avenue des Landais – 63177 Aubière cedex, France

***Abstract*— Estimating the pose of an imaging sensor is a central research problem. Many solutions have been proposed for the case of a rigid environment. In contrast, we tackle the case of a non-rigid environment observed by a 3D sensor, which has been neglected in the literature. We represent the environment as sets of time-varying 3D points explained by a low-rank shape model, that we derive in its implicit and explicit forms. The parameters of this model are learnt from data gathered by the 3D sensor. We propose a learning algorithm based on minimal 3D non-rigid tensors that we introduce. This is followed by a Maximum Likelihood nonlinear refinement performed in a bundle adjustment manner. Given the learnt environment model, we compute the pose of the 3D sensor, as well as the deformations of the environment, that is, the non-rigid counterpart of pose, from new sets of 3D points. We validate our environment learning and pose estimation modules on simulated and real data.**

## I. Introduction

Aligning 3D views – sets of 3D points – gathered by a 3D sensor, such as a calibrated stereo rig, is important for constructing comprehensive 3D models of the environment or updating the position of a mobile imaging sensor. When the environment is rigid, the 3D views are related by rigid Euclidean transformations. Many approaches have been proposed to compute these transformations, *e.g.* [1]. Aligning 3D views is one of the building blocks of hierarchical approaches to Structure-From-Motion. However, the assumption of rigidity is violated in many cases of interest, for instance a garment deforming as a person moves. The alignment problem is then particularly challenging because a different shape is observed in each 3D view.

A large body of work has been done in the medical imaging community but with the aim of estimating dense deformation fields from dense, often voxel-based, reconstructions. Dealing with non-rigid scenes coming from single-camera footage has received an increasing attention over the last few years. The problem is highly challenging since both the cameras and the non-rigid shape have to be recovered. A major step forwards for such cases was made by Bregler *et al.* [2] and Brand [3]. Building on the work of [4], they developed and demonstrated factorization of images of non-rigid scenes, where the non-rigidity was represented as a linear combination of basis shapes. It is shown in [5] how the constraints coming from two synchronized cameras can be incorporated into non-rigid factorization.

We tackle the problem of computing the pose of a 3D sensor with respect to a non-rigid scene, that we represent using the low-rank shape model used in non-rigid factorization methods. Most previous work, *e.g.* [3], [2], [5], [6] use the weak perspective camera model. In contrast, we do not specify a camera model, since we directly consider 3D views. We assume that spatial and temporal point correspondences are established. Pose estimation in a non-rigid environment raises two main problems. First, one has to define the meaning of non-rigid pose. One benefit of using the low-rank shape model is that the 'true' camera pose is recovered. Second, contrarily to classical model-based pose estimation in a rigid environment, a prior model of the non-rigid environment is not available in many cases. We propose to learn this model from a collection of unregistered 3D views gathered by the 3D sensor. Once this learning stage has been passed, our non-rigid pose estimator can be launched.

We bring the following contributions. First, §III, we state the implicit and explicit low-rank shape models, and state the notion of pose in this context. Second, §IV, we propose algorithms to learn the non-rigid environment. The implicit model parameters are learnt using a factorization technique, while for the explicit model, we use what we call *minimal 3D non-rigid tensors*. Third, §V, we show how the pose of the 3D sensor can be computed with respect to the learnt model while the environment is moving and deforming. Experimental results on simulated and real data are reported in §VI. We give our conclusions in §VII.

## II. Notation

Matrices are written in sans-serif fonts, *e.g.* $\mathsf{R}$, and vectors using bold fonts, *e.g.* $\mathbf{x}$. The $n$ 3D views are sets of $m$ points denoted $\mathbf{Q}_{tj}$, where $t$ is the time index and $j$ the point index. We do not use homogeneous coordinates, *e.g.* $\mathbf{Q}_{tj}$ is a 3-vector. The identity matrix of size $(s \times s)$ is written $\mathsf{I}_{(s)}$, the zero matrix $\mathsf{0}$ and the zero vector $\mathbf{0}$. We use $\mathsf{I}$ for the $(3 \times 3)$ identity matrix. The Kronecker product is written $\otimes$, matrix Frobenius norm as $\|\cdot\|$ and the Moore-Penrose pseudo-inverse as $^\dagger$.

## III. Non-Rigid Shape and Pose

### A. Non-Rigid Shape

We describe the low-rank non-rigid shape model. The pose of the 3D sensor is modeled by 3D Euclidean transformations $\{(\mathsf{R}_t, \mathbf{y}_t)\}$ with $\mathsf{R}_t$ an orthonormal matrix and $\mathbf{y}_t \in \mathbb{R}^3$ such that $\hat{\mathbf{Q}}_{tj} = \mathsf{R}_t \tilde{\mathbf{Q}}_{tj} + \mathbf{y}_t$. The $\{\tilde{\mathbf{Q}}_{tj}\}$ form a motionless version of the 3D views, *i.e.* that do not undergo any 'global motion',

but are deforming through time. The low-rank shape model represents the $\{\tilde{\mathbf{Q}}_{tj}\}$ as linear combinations of $l$ *basis shapes* $\{\mathbf{B}_{kj}\}$: $\tilde{\mathbf{Q}}_{tj} = \sum_{k=1}^{l} \xi_{tk}\mathbf{B}_{kj}$. The time-varying $\{\xi_{tk}\}$ are the *configuration weights*. Introducing the $\{(\mathsf{R}_t, \mathbf{y}_t)\}$, we obtain the *explicit model*:

$$\hat{\mathbf{Q}}_{tj} = \mathsf{R}_t \left( \sum_{k=1}^{l} \xi_{tk}\mathbf{B}_{kj} \right) + \mathbf{y}_t \quad (1)$$

$$= \mathsf{M}_t\mathbf{B}_j + \mathbf{y}_t \qquad \text{with} \quad (2)$$

$$\mathsf{M}_t = \mathsf{R}_t \begin{pmatrix} \xi_{t1}\mathrm{I} & \cdots & \xi_{tl}\mathrm{I} \end{pmatrix}. \quad (3)$$

We call $\mathsf{M}_t$ a $(3 \times r)$ *explicit non-rigid motion matrix* and $\mathbf{B}_j = \begin{pmatrix} \mathbf{B}_{1j}^\mathsf{T} & \cdots & \mathbf{B}_{lj}^\mathsf{T} \end{pmatrix}$ a $(r \times 1)$ *non-rigid basis shape vector*. Parameter $r = 3l$ is the *rank* of the model. For reasons that are made clearer below, we derive a bilinear *implicit model*. Let $\mathcal{A}$ be a $(3l \times 3l)$ rank-$3l$ matrix. It is seen that $\hat{\mathbf{Q}}_{tj} = \mathsf{M}_t\mathbf{B}_j + \mathbf{y}_t = (\mathsf{M}_t\mathcal{A}^{-1})(\mathcal{A}\mathbf{B}_j) + \mathbf{y}_t$, yielding:

$$\hat{\mathbf{Q}}_{tj} = \mathsf{N}_t\mathbf{S}_j + \mathbf{y}_t, \quad (4)$$

with $\mathsf{N}_t = \mathsf{M}_t\mathcal{A}^{-1}$ and $\mathbf{S}_j = \mathcal{A}\mathbf{B}_j$. We call $\mathsf{N}_t$ and $\mathbf{S}_j$ the *implicit* non-rigid motion matrix and shape vector, and $\mathcal{A}$ a *corrective transformation matrix*.

### B. Non-Rigid Pose

Pose in a non-rigid environment has a rigid and a non-rigid counterpart. The rigid part $\{(\mathsf{R}_t, \mathbf{y}_t)\}$ represents the 'global' motion of the environment relative to the sensor. It gives the 'true' *relative* sensor displacement. In contrast, the non-rigid part only concerns the environment, and not the imaging sensor. In the above-described model, it is represented by the configuration weights $\{\xi_{tk}\}$, giving the intrinsic, *i.e.* motionless, deformations of the environment at each time instant. The motionless and deformationless environment is modeled by the basis shapes $\{\mathbf{B}_{kj}\}$.

The implicit model is useless for pose estimation: it can be seen as an 'uncalibrated' model of the environment. However, its ML (Maximum Likelihood) Estimate can be computed very reliably, as will be seen in the next section.

## IV. Learning the Environment

Given a collection of 3D views, we learn the environment by estimating the parameters of the low-rank shape model. Note that only the basis shapes $\{\mathbf{B}_{kj}\}$ are subsequently used for pose estimation, see §V. However, to get an ML Estimate, all parameters of the model must be computed.

We state the ML residual error and show how to compute the translations. We first tackle the case of the implicit model and then the explicit one. We assume all points to be visible in all 3D views.

### A. Maximum Likelihood residual error

Assuming that the error on the 3D points is Gaussian, centred and i.i.d., the ML residual error is:

$$\mathcal{D}^2 = \frac{1}{nm} \sum_{t=1}^{n} \sum_{j=1}^{m} d^2(\hat{\mathbf{Q}}_{tj}, \mathbf{Q}_{tj}), \quad (5)$$

where $d^2(\mathbf{X}, \mathbf{Y}) = \|\mathbf{X} - \mathbf{Y}\|^2$ is the Euclidean distance measure and $\{\hat{\mathbf{Q}}_{tj}\}$ are corrected points, exactly explained by the non-rigid shape model.

### B. Computing the Translations

We show that the translations $\mathbf{y}_t$ can be eliminated prior to estimating the other parameters. By substituting equation (1) or equation (4) in the residual error (5) and nullifying its partial derivatives with respect to $\mathbf{y}_t$, we obtain $\mathbf{y}_t = \frac{1}{m}\left(\sum_{j=1}^{m} \mathbf{Q}_{tj} - \hat{\mathbf{Q}}_{tj}\right)$. The origin of the $r$-dimensional space containing the non-rigid shape vectors is arbitrary and is chosen such that $\sum_{j=1}^{m} \mathbf{S}_j = \mathbf{0}$ in the implicit case and $\sum_{j=1}^{m} \mathbf{B}_j = \mathbf{0}$ in the explicit case, giving for the translation $\mathbf{y}_t$ the centroid $\mathbf{y}_t = \frac{1}{m}\sum_{j=1}^{m} \mathbf{Q}_{tj} = \bar{\mathbf{Q}}_t$ of the $t$-th 3D view. This means that one cancels the translations out by centring each set of 3D points on its centroid: $\mathbf{Q}_{tj} \leftarrow \mathbf{Q}_{tj} - \bar{\mathbf{Q}}_t$. Henceforth, we assume that this has been done.

### C. Shape Learning With the Implicit Model

We consider the implicit non-rigid shape model of equation (4). We factorize the 3D views $\{\mathbf{Q}_{tj}\}$ into implicit non-rigid motion matrices $\{\mathsf{N}_t\}$ and shape vectors $\{\mathbf{S}_j\}$. The problem is to minimize the ML residual error (5) over the $\{\hat{\mathbf{Q}}_{tj}\}$ such that $\hat{\mathbf{Q}}_{tj} = \mathsf{N}_t\mathbf{S}_j$. Rewrite (5) as:

$$\mathcal{D}^2 \propto \|\hat{\mathcal{Q}} - \mathcal{Q}\|^2,$$

where $\mathcal{Q}$ is the $(3n \times m)$ *measurement matrix*:

$$\mathcal{Q} = \begin{pmatrix} \mathbf{Q}_{11} & \cdots & \mathbf{Q}_{1m} \\ \vdots & \ddots & \vdots \\ \mathbf{Q}_{n1} & \cdots & \mathbf{Q}_{nm} \end{pmatrix},$$

and $\hat{\mathcal{Q}}$ is defined by the implicit $(3n \times 3l)$ 'non-rigid joint motion matrix' $\mathcal{N}$ and the $(3l \times m)$ 'non-rigid joint structure matrix' $\mathcal{S}$ as $\hat{\mathcal{Q}} = \mathcal{N}\mathcal{S}$ with $\mathcal{N}^\mathsf{T} = \begin{pmatrix} \mathsf{N}_1^\mathsf{T} & \cdots & \mathsf{N}_n^\mathsf{T} \end{pmatrix}$ and $\mathcal{S} = \begin{pmatrix} \mathbf{S}_1 & \cdots & \mathbf{S}_m \end{pmatrix}$. Since $\mathcal{N}$ has $3l$ columns and $\mathcal{S}$ has $3l$ rows, $\hat{\mathcal{Q}}$ has maximum rank $3l$. The problem is to find the closest rank-$3l$ matrix $\hat{\mathcal{Q}}$ to $\mathcal{Q}$. Let $\mathcal{Q} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$ be a Singular Value Decomposition (SVD) of matrix $\mathcal{Q}$, see *e.g.* [7], where $\mathsf{U}$ and $\mathsf{V}$ are orthonormal matrices and $\Sigma$ is diagonal and contains the singular values of $\mathcal{Q}$. Let $\Sigma = \Sigma_u\Sigma_v$ be any decomposition of $\Sigma$, *e.g.* $\Sigma_u = \Sigma_v = \sqrt{\Sigma}$. The non-rigid joint motion and structure matrices are obtained by, loosely speaking, 'truncating' the decomposition by nullifying all but the $3l$ largest singular values, which leads, assuming the singular values in decreasing order in $\Sigma$, to $\mathcal{N} = \psi_{3l}(\mathsf{U}\Sigma_u)$ and $\mathcal{S} = \psi_{3l}^\mathsf{T}(\mathsf{V}\Sigma_v^\mathsf{T})$, where $\psi_c(\mathsf{W})$ is formed with the $c$ leading columns of matrix $\mathsf{W}$.

### D. Shape Learning With the Explicit Model

The aim is to compute the ML Estimate of the configuration weights, rotation matrices and non-rigid structure in equation (1) by minimizing the residual error (5). This is a nonlinear problem for which two approaches have been followed in the non-rigid factorization litterature. On the one hand Bregler *et al.* [2], Brand [3], Aanaes *et al.* [8], Del Bue *et al.* [5] and

Xiao *et al.* [6] compute a matrix $\mathcal{A}$ that upgrades the implicit motion matrix $\mathcal{N}$ so that the metric constraints of the explicit model are enforced. Xiao *et al.* show that in order to get the correct solution, two types of metric constraints must be taken into account: the *rotation constraints* and the *basis constraints*, from which they derive a closed-form solution for matrix $\mathcal{A}$.

On the other hand, Torresani *et al.* [9] directly learn the parameters of the explicit model. They propose a comprehensive system based on a generalized EM (Expectation Maximization) algorithm. An important, still unsolved problem is to find a suitable initialization, since EM performs local optimization only.

Our solution lies in the second category: a suboptimal initialization is computed and subsequently refined in a bundle adjustment manner. These two steps are presented below, followed by an analysis of the ambiguities of the solution.

*1) Initializing:*

*a) The rotations:* Brand proposes a solution based on upgrading the implicit motion matrices [3], which requires at least $n \geq \frac{l(9l+3)}{4}$ 3D views to compute a corrective transformation and is thus not feasible for many practical cases. For example, at least 39 views giving independent constraints are necessary to use this method with the sequence presented in §VI-B. In [5], the authors compute a block-diagonal corrective transformation matrix. Another solution used in [8] is to assume that the environment has a sufficiently strong rigid component, and to estimate the rotation using a standard procedure such as [1]. This approach is not feasible for highly deforming environments.

In contrast, we propose an approach taking the non-rigid nature of the environment into account. Our algorithm is presented below in the occlusion-free case for simplicity, but can be easily extended to the missing data case. Consider the explicit non-rigid joint motion equation $\mathcal{Q} = \mathcal{M}\mathcal{B}$ with:

$$\mathcal{M} = \begin{pmatrix} \xi_{11}\mathsf{R}_1 & \cdots & \xi_{1l}\mathsf{R}_1 \\ \vdots & \ddots & \vdots \\ \xi_{n1}\mathsf{R}_n & \cdots & \xi_{nl}\mathsf{R}_n \end{pmatrix} \quad \text{and} \quad \mathcal{B} = \begin{pmatrix} \mathbf{B}_{11} & \cdots & \mathbf{B}_{1m} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{l1} & \cdots & \mathbf{B}_{lm} \end{pmatrix}.$$

Define two subsets $\mathbb{A}$ and $\mathbb{B}$ of $n_a$ and $n_b$ 3D views respectively, $\mathcal{Q}_a = \mathcal{M}_a\mathcal{B}$ and $\mathcal{Q}_b = \mathcal{M}_b\mathcal{B}$. Our goal is to eliminate the structure $\mathcal{B}$ from the equations. We assume without loss of generality $\mathrm{rank}(\mathcal{Q}_b) \geq 3l$. This implies $n_b \geq l$. We express $\mathcal{B}$ in terms of $\mathcal{Q}_b$ and $\mathcal{M}_b$ using the equation subset $\mathbb{B}$ as $\mathcal{B} = \mathcal{M}_b^\dagger\mathcal{Q}_b$. Plugging this into the equation subset $\mathbb{A}$ yields $\mathcal{Q}_a = \mathcal{M}_a\mathcal{B} = \mathcal{M}_a\mathcal{M}_b^\dagger\mathcal{Q}_b$ that we rewrite:

$$\underbrace{\begin{pmatrix} \mathrm{I}_{(3n_a)} & -(\mathcal{M}_a\mathcal{M}_b^\dagger) \end{pmatrix}}_{\mathcal{Z}} \mathcal{Q}_{ab} \quad = \quad \mathsf{0}_{(3n_a \times m)} \tag{6}$$

where $n_{ab} = n_a + n_b$ and $\mathcal{Q}_{ab}^\mathsf{T} = (\mathcal{Q}_a^\mathsf{T} \;\; \mathcal{Q}_b^\mathsf{T})$. We call matrix $\mathcal{Z}_{(3n_a \times 3n_{ab})}$ a *3D non-rigid tensor*. Let us examine more closely the expression of $\mathcal{M}_a\mathcal{M}_b^\dagger$. The joint motion matrix can be rewritten as $\mathcal{M} = \mathcal{R}(\Xi \otimes \mathrm{I})$ where $\mathcal{R} = \mathrm{diag}(\mathsf{R}_1, \ldots, \mathsf{R}_n)$ is an orthonormal matrix and $\Xi$ is an $(n \times l)$ matrix containing the $\{\xi_{ik}\}$. Similarly, $\mathcal{M}_a = \mathcal{R}_a(\Xi_a \otimes \mathrm{I})$ and $\mathcal{M}_b = \mathcal{R}_b(\Xi_b \otimes \mathrm{I})$, yielding:

$$\begin{aligned} \mathcal{M}_a\mathcal{M}_b^\dagger &= \mathcal{R}_a(\Xi_a \otimes \mathrm{I})(\mathcal{R}_b(\Xi_b \otimes \mathrm{I}))^\dagger \\ &= \mathcal{R}_a(\Xi_a \otimes \mathrm{I})((\Xi_b \otimes \mathrm{I}))^\dagger\mathcal{R}_b^\mathsf{T}, \end{aligned}$$

since $\mathcal{R}_b$ is an orthonormal matrix. We make use of the following properties: $(\mathsf{S} \otimes \mathrm{I})^\dagger = \mathsf{S}^\dagger \otimes \mathrm{I}$ and $(\mathsf{S} \otimes \mathrm{I})(\mathsf{S}' \otimes \mathrm{I}) = (\mathsf{S}\mathsf{S}') \otimes \mathrm{I}$ to get:

$$\mathcal{M}_a\mathcal{M}_b^\dagger \quad = \quad \mathcal{R}_a\left((\Xi_a\Xi_b^\dagger) \otimes \mathrm{I}\right)\mathcal{R}_b^\mathsf{T}.$$

Substituting in equation (6) and multiplying on the left by the orthonormal $\mathcal{R}_a^\mathsf{T}$ yields:

$$\left(\mathcal{R}_a^\mathsf{T} \quad -\left((\Xi_a\Xi_b^\dagger) \otimes \mathrm{I}\right)\mathcal{R}_b^\mathsf{T}\right)\mathcal{Q}_{ab} \quad = \quad \mathsf{0}_{(3n_a \times m)}. \tag{7}$$

From this equation, knowing $\mathcal{R}_a$ and using the orthonormality constraints on $\mathcal{R}_b$ to eliminate the weights $\Xi_a\Xi_b^\dagger$ should allow to compute $\mathcal{R}_b$. We use the fact that the coordinate frame can be aligned on a reference view $t$, *i.e.* such that $\mathsf{R}_t = \mathrm{I}$ and choose one view in the initial set of 3D views $\mathbb{A}$ to be the reference one.

The first idea that comes to mind to solve this problem is to consider the left nullspace of $\mathcal{Q}_{ab}$. Define a $(3n_{ab} \times (3n_{ab} - 3l))$ matrix $\mathsf{U}$ whose columns span the left nullspace of $\mathcal{Q}_{ab}$: $\mathsf{U}^\mathsf{T}\mathcal{Q}_{ab} = \mathsf{0}$. Using equation (7), we obtain:

$$\left(\mathcal{R}_a^\mathsf{T} \quad -\left((\Xi_a\Xi_b^\dagger) \otimes \mathrm{I}\right)\mathcal{R}_b^\mathsf{T}\right) \quad = \quad \mathsf{H}\mathsf{U}^\mathsf{T},$$

where $\mathsf{H}$ accounts for the fact that any linear combination of the columns of $\mathsf{U}$ are in the left nullspace of $\mathcal{Q}_{ab}$. While this approach works fine in the absence of noise contaminating the data, it is however very unstable and useless when even very slight noise is present in the data. Indeed, if one employs *e.g.* SVD to compute matrix $\mathsf{U}$, then the singular vectors corresponding to the lowest singular values will be selected, and will not in general allow to recover the sought-after rotations, since the SVD mixes the singular vectors to obtain the lowest residual error as possible.

The second idea that comes to mind is to estimate each rotation in $\mathbb{B}$ and the corresponding weight at a time. Consider a 3D view $g \in \mathbb{A}$. Equation (7) induces the following residual error:

$$\sum_{j=1}^{m} \|\mathsf{R}_g^\mathsf{T}\mathbf{Q}_{gj} - \sum_{t \in \mathbb{B}} \zeta_t \mathsf{R}_t^\mathsf{T}\mathbf{Q}_{tj}\|^2, \tag{8}$$

where $\{\zeta_t\}$ are unknown weights. Initialize all rotations in $\mathcal{R}_b$ to the identity: $\mathsf{R}_t^0 = \mathrm{I}$, $t \in \mathbb{B}$. Let $p \leftarrow 0$ be the iteration counter. The idea is to iteratively compute the $t$-th rotation for $t \in \mathbb{B}$ while holding the other $n_b - 1$ rotations in $\mathbb{B}$ until convergence, by minimizing the residual error (8) that we rewrite:

$$\sum_{j=1}^{m} \|\mathbf{E}_j^p - \zeta_t^{p+1}\left(\mathsf{R}_t^{p+1}\right)^\mathsf{T}\mathbf{Q}_{tj}\|^2 \tag{9}$$

with:

$$\mathbf{E}_j^p \quad = \quad \mathsf{R}_g^\mathsf{T}\mathbf{Q}_{gj} - \left(\sum_{t \in \mathbb{B}} (\mathsf{S}_t^p)^\mathsf{T}\mathbf{Q}_{tj}\right), \tag{10}$$

where $\mathsf{S}_t^p$ is the latest estimate, *i.e.* :

$$\mathsf{S}_t^p = \begin{cases} \zeta_t^{p+1}\mathsf{R}_t^{p+1} & \text{if it is computed} \\ \zeta_t^p \mathsf{R}_t^p & \text{otherwise.} \end{cases}$$

We use a standard procedure for computing the 3D rotation and scale from 3D point correspondences – here $\{\mathbf{E}_j^p \leftrightarrow \mathbf{Q}_{ij}\}$ – due to [1] to solve this problem. Our algorithm is summarized in table I. Note that at most $l$ rotations in $\mathcal{R}_b$ can be computed at each iteration which implies that the number of rotations in $\mathcal{R}_b$ must be $l$. This is why only the smallest, *i.e.* minimal 3D non-rigid tensors can be used by our algorithm. Also, the unknown $\mathcal{M}_b$ must be full-rank. We use the corresponding implicit $\mathcal{N}_b$ to check that this is the case, since there exists a full-rank corrective transformation matrix $\mathcal{A}$ such that $\mathcal{N}_a\mathcal{A} = \mathcal{M}_a$. In the case of missing data, the sum in equations (8) and (9) is simply replaced by a sum over the points seen in subsets $\mathbb{A}$ and $\mathbb{B}$.

OBJECTIVE

Given $n$ 3D views $\{\mathbf{Q}_{tj}\}$ of $m$ corresponding points and the rank $3l$ of the non-rigid model, compute the relative pose $\{(\mathsf{R}_t, \mathbf{y}_t)\}$ of the 3D sensor, the non-rigid pose of the environment, *i.e.* the configuration weights $\{\xi_{tk}\}$, while learning the low-rank non-rigid shape model $\{\mathbf{B}_{kj}\}$.

ALGORITHM

1) **Set initial equation sets.** $\mathbb{A}$ is any 3D view $t$, $\mathbb{B}$ is any $l$ 3D views at least one of them not in $\mathbb{A}$ and such that $\mathcal{N}_b$ is full-rank, $\mathsf{R}_t \leftarrow \mathrm{I}$ and $\mathcal{R}_a \leftarrow \mathrm{I}$.
2) **Compute the rotations:**
   a) Set initial rotations $\mathsf{R}_t^0 \leftarrow \mathrm{I}$, $t \in \mathbb{B}$ and the iteration counter $p \leftarrow 0$.
   b) For $t \in \mathbb{B}$: form the $\{\mathbf{E}_j^p\}$, equation (10). Compute $\mathsf{R}_t^{p+1}$ by minimizing (9), see Horn *et al.* [1].
   c) $p \leftarrow p + 1$.
   d) If the decrease in the residual error is smaller than $\varepsilon$, go to step 3 else go to step b.
3) **Test convergence.** If all rotations are computed, stop.
4) **Update equation sets.** $\mathbb{A} \leftarrow \mathbb{A} \cup \mathbb{B}$ and $\mathbb{B}$ is any $l$ 3D views, at least one of them not in $\mathbb{A}$ and such that $\mathcal{N}_b$ is full-rank.
5) **Iterate.** go to step 2.

TABLE I

THE PROPOSED INITIALIZATION ALGORITHM FOR THE EXPLICIT MODEL PARAMETERS.

*b) The configuration weights and non-rigid structure:* Consider the ML residual error (5) that we rewrite below for convenience:

$$\mathcal{D}^2 = \frac{1}{nm}\sum_{t=1}^{n}\sum_{j=1}^{m} \|\mathbf{Q}_{tj} - \mathsf{R}_t\left(\sum_{k=1}^{l}\xi_{tk}\mathbf{B}_{kj}\right)\|^2.$$

Let $\tilde{\mathbf{Q}}_{tj} = \mathsf{R}_t^\mathsf{T}\mathbf{Q}_{tj}$ be a motionless version of the 3D points, the residual error transforms in:

$$\mathcal{D}^2 = \frac{1}{nm}\sum_{t=1}^{n}\sum_{j=1}^{m} \|\tilde{\mathbf{Q}}_{tj} - \left(\sum_{k=1}^{l}\xi_{tk}\mathbf{B}_{kj}\right)\|^2.$$

Introduce matrices $\mathcal{L}_{(n\times 3m)}$ and $\mathcal{T}_{(l\times 3m)}$ which are obtained by reorganizing $\tilde{\mathcal{Q}}$ and $\mathcal{B}$, respectively:

$$\tilde{\mathcal{L}} = \begin{pmatrix} \tilde{\mathbf{Q}}_{11}^\mathsf{T} & \cdots & \tilde{\mathbf{Q}}_{1m}^\mathsf{T} \\ \vdots & \ddots & \vdots \\ \tilde{\mathbf{Q}}_{n1}^\mathsf{T} & \cdots & \tilde{\mathbf{Q}}_{nm}^\mathsf{T} \end{pmatrix} \quad \text{and} \quad \mathcal{T} = \begin{pmatrix} \mathbf{B}_{11}^\mathsf{T} & \cdots & \mathbf{B}_{1m}^\mathsf{T} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{l1}^\mathsf{T} & \cdots & \mathbf{B}_{lm}^\mathsf{T} \end{pmatrix}.$$

The residual error is rewritten $\mathcal{D}^2 = \frac{1}{nm}\|\tilde{\mathcal{L}} - \Xi\mathcal{T}\|^2$. This means that matrix $\tilde{\mathcal{L}}$ has rank $l$ at most. Similarly to §IV-C, let $\tilde{\mathcal{L}} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$ be an SVD of matrix $\tilde{\mathcal{L}}$, we get $\Xi = \psi_l(\mathsf{U}\Sigma_u)$ and $\mathcal{T} = \psi_l^\mathsf{T}(\mathsf{V}\Sigma_v^\mathsf{T})$.

*2) Bundle Adjustment:* Starting from the above-derived initial solution, we minimize the ML residual error (5) using nonlinear least-squares in a bundle adjustment manner, see *e.g.* [10]. We use the Levenberg-Marquardt algorithm, implemented to exploit the sparse block structure of the Jacobian and (Gauss-Newton approximation of) the Hessian matrices. Bundle adjustment in the non-rigid case is developed in [8], [5], where the authors show that compared to the rigid case, additional 'gauge freedoms' in the recovered structure and motion must be handled. However, the Levenberg-Marquardt optimization engine deals with those by damping the approximated Hessian matrix which makes it full rank. We found that the regularization term employed in [5] does not have a significant effect on the results we obtained. This is mainly due to the fact that we directly use 3D data, while [5] use image points.

*3) Ambiguities of the Solution:* The ambiguity of the solution demonstrated by Xiao *et al.* [6] in the 2D case when only the rotation constraints are used does not hold for our algorithm. The reason is that it enforces the replicated block structure of the joint motion matrix $\mathcal{M}$, which provides stronger constraints than the rotation constraints only. The ambiguity matrix $\mathcal{E}$ on the learnt model is $\mathcal{E} = \mathrm{diag}_l(\mathsf{S})(\Lambda_{(l\times l)} \otimes \mathrm{I})$, where $\mathrm{diag}_l(\mathsf{S})$ is a $l$ block diagonal matrix for some 3D orthonormal matrix $\mathsf{S}$, representing the indeterminateness of the orientation for the global coordinate frame. Matrix $\Lambda_{(l\times l)} \otimes \mathrm{I}$ models linear combinations of the basis shapes. This shows that it is not possible to recover the 'true' basis shapes and configuration weights, but that 'true' camera pose can still be computed.

## V. COMPUTING POSE

Given the non-rigid model of the environment – the basis shapes $\{\mathbf{B}_{kj}\}$ – and a 3D view $\{\mathbf{Q}_j\}$, we want to estimate the pose of the 3D sensor, namely the Euclidean transformation $(\mathsf{R}, \mathbf{y})$, jointly with the non-rigid counterpart of the pose, *i.e.* the configuration weights $\{\xi_k\}$. Note that we drop index $t$ since only one 3D view is considered in this section. It is not necessary to observe all points used in the learning phase to compute pose. The ML residual error is:

$$\mathcal{C}^2 = \frac{1}{m}\sum_{j=1}^{m} d^2(\mathsf{M}\mathbf{B}_j + \mathbf{y}, \mathbf{Q}_j). \tag{11}$$

It must be minimized over $(\mathsf{R}, \mathbf{y})$ and $\{\xi_k\}$. Matrix $\mathsf{M}$ is defined by equation (3).

We propose to nonlinearly minimize the ML residual error (11) using the Levenberg-Marquardt algorithm. It is not possible to use a direct estimator as [1] due to the configuration weights. Note that, as shown below, the translation $\mathbf{y}$ can be eliminated from the equation. The minimization is thus performed over $\mathsf{R}$ and $\{\xi_k\}$. Such an algorithm as Levenberg-Marquardt requires one to provide an initial solution. Our algorithm for finding it is described below.

*a) Eliminating the translation:* The derivatives of the ML residual error (11) with respect to $\mathbf{y}$ must vanish: $\frac{\partial \mathcal{C}^2}{\partial \mathbf{y}} = \mathbf{0}$, which leads to $\mathbf{y} = \frac{1}{m}\sum_{j=1}^{m}(\mathbf{Q}_j - \mathsf{M}\mathbf{B}_j)$. This result means that $\mathbf{y}$ is given by the difference between the centroid of the points $\{\mathbf{Q}_j\}$ and the centroid predicted by the points from the shape model $\{\mathsf{M}\mathbf{B}_j\}$, which vanishes if the set of points used for computing the pose is exactly the same as the one used in the learning phase. In any case, by centring the points on their centroid, the translation vanishes. Henceforth, we assume this has been done and rewrite the ML residual error (11) as:

$$\mathcal{C}^2 \quad = \quad \frac{1}{m}\sum_{j=1}^{m} d^2(\mathsf{M}\mathbf{B}_j, \mathbf{Q}_j). \tag{12}$$

*b) Initializing the rotation and configuration weights:* We linearly compute a motion matrix $\tilde{\mathsf{M}}$ without enforcing the correct replicated structure by $\min_{\tilde{\mathsf{M}}} \sum_{j=1}^{m} d^2(\tilde{\mathsf{M}}\mathbf{B}_j, \mathbf{Q}_j)$, which yield:

$$\tilde{\mathsf{M}} \quad = \quad \begin{pmatrix}\mathbf{Q}_1 & \cdots & \mathbf{Q}_m\end{pmatrix}\begin{pmatrix}\mathbf{B}_1 & \cdots & \mathbf{B}_m\end{pmatrix}^{\dagger}.$$

We extract the $\{\xi_k\}$ and $\mathsf{R}$ from $\tilde{\mathsf{M}}$ by solving $\min_{\mathsf{R},\{\xi_k\}} \sum_{k=1}^{l} \|\tilde{\mathsf{M}}_k - \xi_k \mathsf{R}\|^2$, where the $\tilde{\mathsf{M}}_k$ are $(3 \times 3)$ blocks from $\tilde{\mathsf{M}}$. By vectorizing and reorganizing the residual error, we obtain:

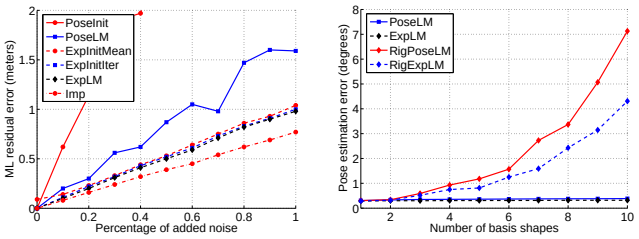
$$\| \underbrace{\begin{pmatrix}\mathrm{vect}^{\mathsf{T}}(\tilde{\mathsf{M}}_1)\\ \vdots \\ \mathrm{vect}^{\mathsf{T}}(\tilde{\mathsf{M}}_l)\end{pmatrix}}_{\Lambda} - \underbrace{\begin{pmatrix}\xi_1\\ \vdots \\ \xi_l\end{pmatrix}}_{\boldsymbol{\xi}} \underbrace{\mathrm{vect}^{\mathsf{T}}(\tilde{\mathsf{R}})}_{\tilde{\mathbf{r}}} \|^2,$$

which is a rank-1 approximation problem that we solve by 'truncating' the SVD $\Lambda = \mathsf{U}\Sigma\mathsf{V}^{\mathsf{T}}$, as in §IV-C: $\boldsymbol{\xi} = \psi_1(\mathsf{U}\Sigma)$ and $\tilde{\mathbf{r}} = \psi_1^{\mathsf{T}}(\mathsf{V})$. Note that $\|\tilde{\mathbf{r}}\| = \|\tilde{\mathsf{R}}\| = 1$. Matrix $\tilde{\mathsf{R}}$ must be subsequently corrected to give $\mathsf{R}$ by enforcing the orthonormality constraints. This is done by finding the closest orthonormal matrix to $\tilde{\mathsf{R}}$ using SVD, see [1]: $\tilde{\mathsf{R}} = \mathsf{U}\Sigma\mathsf{V}^{\mathsf{T}}$ gives $\mathsf{R} = \frac{1}{3}\mathrm{tr}(\Sigma)\det(\mathsf{U})\det(\mathsf{V})\mathsf{U}\mathsf{V}^{\mathsf{T}}$, while compensating the possible sign change by $\boldsymbol{\xi} \leftarrow \det(\mathsf{U})\det(\mathsf{V})\boldsymbol{\xi}$.

## VI. Experimental Evaluation

### A. Simulated Data

We report experimental results on simulated data. The default simulation setup consists of $n = 15$ time-varying 3D views, each containing $m = 35$ points. They are generated by randomly drawn linear combinations of $l = 3$ basis shapes, all of them lying in a sphere with unit radius. An additive, zero-mean Gaussian noise with variance $\sigma = 0.01$ (*i.e.* 1% of the scene scale) is added to the 3D points. We vary each of these parameters in turn. We average the error measures over 100 trials. The true number of basis shapes is used by the algorithms.




(a) (b)

Fig. 1. (a) ML residual error against the level of added noise and (b) pose error against the number of basis shapes.

*c) Environment learning:* We observe on figure 1 (a) that the ML residual error is very close to $\sigma$. The implicit learning Imp consistently gives a significantly lower residual error than the explicit learning algorithms Exp⋆. This means that, despite the fact that the data were generated using the explicit low-rank shape model, the extra degrees of freedom of the implicit model represent quite well the added Gaussian noise.

We observe that the difference between the three explicit learning methods is small compared to the difference with Imp. ExpLM (from §IV-D.2, 'LM' stands for Levenberg-Marquardt) always performs better than ExpInitIter (from table I), which always performs better than ExpInitMean (based on [1] to get the rotations). This means that the residual error (8), which is minimized by ExpInitIter while estimating the minimal 3D non-rigid tensors, is well-adapted to our problem.

Figure 1 (b) compares the error raised by the rotation part of the pose, in degrees, between our non-rigid algorithms and rigid SFM and pose algorithms, respectively dubbed RigExpLM and RigPoseLM. We observe that the proposed ExpLM gives errors independent of the number of basis shapes, while, as could have been expected, RigExpLM rapidly degrades as the number of basis shapes grows.

*d) Pose computation:* Figure 1 (a) shows that all pose algorithms Pose⋆ consistently give a higher residual error than the explicit learning algorithms. This is explained by the fact that pose estimation suffers from the errors in the learnt model *and* in the 3D view. PoseInit gives quite high errors, roughly $5\sigma$, while PoseLM converges to roughly $1.5\sigma$ which is reasonable. The same remarks as for the learning algorithms can be made in the case of pose, for figure 1 (b).

Another experiment was intended to assess to which extent, reliable pose estimate can be obtained when the environment is deforming in a very different way compared to the learning stage. Let $\nu$ be the mean value of the configuration weights. We alter them by adding randomly drawn perturbations with increasing magnitude $\mu$, and generate a 3D view with these parameters, from which pose is estimated. Obviously, the results depend on the simulation setup, the number of points, views, basis shapes and the level of noise. However, we

observe that for $\mu \leq 1.3\nu$, the residual error indicates that the pose estimate is reasonable for most configurations. For $\mu > 1.3\nu$, the pose estimate rapidly degrades.

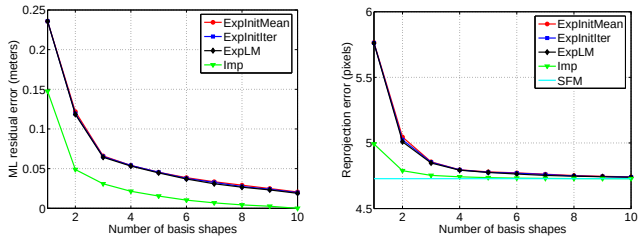
### B. Real Data

We tested our algorithms on sets of 3D points reconstructed from a calibrated stereo rig. The sequence consists of $n = 650$ pairs of views. The $m = 30$ point tracks were obtained semi-automatically and reconstruction was performed using ML triangulation, *i.e.* by minimizing the reprojection error. The reprojection error we obtained is 4.7276 pixels, which is rather large and explained by the low quality of the manually entered point tracks.

Fig. 2. One out of the 650 stereo pairs used in the experiments, overlaid with the 30 point tracks.

We used a subset of the full sequence, made of 1 3D view over 25 from 1 to 551, that is 23 3D views, for learning the environment. The remaining 3D views are registered by computing pose. For views $1 < i < 551$, this can be viewed as 'interpolation' since the surrounding 3D views are used for learning the environment, while for views $551 < i < 650$ this can be viewed as an 'extrapolation' of the model since new pose and deformations are seen in these views.

An important aspect is the choice of the number $l$ of basis shapes. If $l$ is too low, the model is not able to represent all the possible deformations, while if $l$ is too high, the noise is modeled, resulting in unreliable pose estimates in both cases. We propose to manually choose $l$ by examining the graphs shown on figure 3. It shows the ML residual errors and the reprojection errors, *i.e.* the Sum of Squared Differences between measured and predicted image points, resulting of the learning algorithms for different numbers of basis shapes. We




(a) (b)

Fig. 3. (a) ML residual error and (b) 2D reprojection error versus the number of basis shapes.

observe that the 3D ML residual error and the 2D reprojection error decrease while $l$ increases, the former towards 0 and the latter towards the reconstruction error, shown by an horizontal line 'SFM' on the graph, which was expected. Based on this graph, we choose $l = 4$, for which the ExpLM ML residual error is 5.32 centimeters and the 2D reprojection error is 4.7922 pixels. For comparison, a rigid environment model gives a 23.58 centimeters ML residual error and a 5.7605 pixels 2D reprojection error. It is important to note that for $l = 5$ and $l = 6$ basis shapes, very similar pose estimates are subsequently obtained.

The learnt path appears visually satisfying, However, the mean difference in the rotations is 2.81 degrees, which is significant, but difficult to illustrate visually. The mean ML residual errors are 8.66 and 12.83 centimeters for the 'interpolated' and the 'extrapolated' poses respectively.

The computation time for the learning phase is of the order of a minute while pose estimation is roughly a tenth of a second.

## VII. Conclusions

One weakness of the approach is to rely on 3D point correspondences. We are currently working on using more robust types of inputs, such as contours or image patches, that can be reliably tracked through sequences of stereo pairs using *e.g.* particule filtering techniques. This is intended to be part of an iterative deforming environment learning system. Essential issues that will be dealt with are assessing what kind of deformations can be represented by the low-rank shape model and choosing the number of basis shapes, which will be examined in the framework of model selection.

## References


[1] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A*, vol. 5, no. 7, pp. 1127–1135, July 1988.

[2] C. Bregler, A. Hertzmann, and H. Biermann, "Recovering non-rigid 3D shape from image streams," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2000.

[3] M. Brand, "Morphable 3D models from video," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2001.

[4] M. Irani, "Multi-frame optical flow estimation using subspace constraints," in *Proceedings of the International Conference on Computer Vision*, 1999.

[5] A. D. Bue and L. Agapito, "Non-rigid 3D shape recovery using stereo factorization," in *Proceedings of the Asian Conference on Computer Vision*, 2004.

[6] J. Xiao, J.-X. Chai, and T. Kanade, "A closed-form solution to non-rigid shape and motion recovery," in *Proceedings of the European Conference on Computer Vision*, 2004.

[7] G. H. Golub and C. F. van Loan, *Matrix Computations*. Baltimore: The Johns Hopkins University Press, 1989.

[8] H. Aanæs and F. Kahl, "Estimation of deformable structure and motion," in *Proceedings of the Vision and Modelling of Dynamic Scenes Workshop*, 2002.

[9] L. Torresani and A. Hertzmann, "Automatic non-rigid 3D modeling from video," in *Proceedings of the European Conference on Computer Vision*, 2004.

[10] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle ajustment — a modern synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, 2000.