Probabilistic inference of multi-joint movements, skeletal parameters and marker attachments from diverse motion capture data

Emanuel Todorov

Abstract— **We describe a comprehensive solution to the problem of reconstructing the multi-joint movement trajectories of the human body from diverse motion capture data. The problem is formulated in a probabilistic framework so as to handle multiple and unavoidable sources of uncertainty: sensor noise, soft tissue deformation and marker slip, inaccurate marker placement and limb measurement, and missing data due to occlusions. All unknown quantities are treated as state variables even though some of them are constant. In this way state estimation and system identification can be performed simultaneously, obtaining not only the most likely values but also the confidence intervals of the joint angles, skeletal parameters, and marker positions and orientations relative to the limb segments. The inference method is a Gauss-Newton generalization of the extended Kalman filter. It is adapted to the kinematic domain by expressing spatial rotations via quaternions and computing the sensor residuals and their Jacobians analytically. The ultimate goal of this project is to provide a reliable data analysis tool used in practice. The software implementation is available online.**

Index Terms— **motion capture, probabilistic inference, self-calibration, extended Kalman filter, kinematics, quaternion**

I. Introduction

THE need to reconstruct the multi-joint movements of the human body arises in a number of fields including motor control and biomechanics, sports, ergonomics, physical medicine and rehabilitation, and computer animation. Motion capture devices that measure the position and orientation of markers attached to the body have become widely available. However, the data analysis methods and software tools lag behind these hardware advances. While a lot has been accomplished [1-10], estimating the configuration of a multi-articulate body to which markers are non-rigidly and non-accurately attached remains a challenge. In particular:

- existing methods do not provide error estimates;
- the requirements for accurate marker placement and limb measurement are rarely met in practice, despite the prolonged setup sessions they lead to;
- reliance on hardware-specific methods makes it difficult to utilize new sensor modalities and to use multiple devices simultaneously;
- most existing methods are tailored to the animation industry and may not have the accuracy needed for research and clinical applications.

E. Todorov is with the Department of Cognitive Science, University of California San Diego, http://www.cogsci.ucsd.edu/~todorov

Supported by NIH grant R01–NS045915. Thanks to Xiuchuan Pan and Weiwei Li for their contributions to the initial phase of this project, and to Francisco Valero-Cuevas for providing hand movement data.

The aim of this project is to formulate the general motion estimation problem within an appropriate mathematical framework, develop a systematic approach to solving it numerically, and provide a software implementation which avoids the above limitations. The problem of computing joint angles given marker positions and orientations is sometimes considered to be one of inverse kinematics. However this is appropriate only in an ideal world without uncertainty. In real-world applications uncertainty must be dealt with, ideally from the very beginning and not as an afterthought. Thus it is more appropriate to cast the problem in the general framework of probabilistic inference. In this framework, all unknown quantities are treated as latent variables and any prior knowledge about them is encoded probabilistically. Their relation to the sensor data is captured by a generative model which defines the conditional probability of the data given the latent variables. Probabilistic inference is essentially an inversion of the generative model, but unlike inverse kinematics it handles uncertainty and makes optimal use of data.

Two types of latent variables will be distinguished: those that vary over time (joint angles, global position and orientation of the body) and those that remain constant (limb sizes, axes of joint rotation, marker placements). Inferring time-varying parameters from data is called estimation while inferring constant parameters is called system identification. Despite this traditional distinction, however, both can be performed simultaneously by recursively computing the joint posterior distribution of all latent variables given the data. An example of such computation is the Kalman filter which has previously been used to perform estimation and system identification simultaneously [11], [12]. Here the system is nonlinear, requiring at least an extended Kalman filter. The method we develop is more accurate than the extended Kalman filter and involves iterative Gauss-Newton maximization of the posterior at each time step. Another extension is our use of an implicit sensor noise model. This is necessary because the natural representations of 3D orientation (unit quaternions and 3x3 orthonormal matrices) are redundant and thus a straightforward noise model would be degenerate.

A substantial part of the paper is devoted to constructing a kinematic generative model. This involves computing the expected values of the sensor measurements conditional on the values of the latent variables. Adding Gaussian noise around the expected values yields the necessary generative model. The inference method also requires the derivatives of the expected sensor measurements with respect to the latent

variables. The computation of the corresponding Jacobian matrix is somewhat elaborate and is described in detail. The kinematic structure is a tree of rigid segments connected with joints. The joints are spatial transformations representing either time-varying joint rotations or fixed relations such as offsets between coordinate frames. Sensors are identified with virtual segments attached to the body using joints. In this way every latent variable is a degree of freedom of some joint, which simplifies the derivation and implementation. 3D orientations are represented and accumulated using quaternions in order to improve numerical efficiency and accuracy.

The probabilistic framework for motion estimation is developed in Section II. The kinematic computations underlying the generative model are presented in Section III. Numerical results on both synthetic and real data are given in Section IV. Preliminary results were reported in [13].

II. Probabilistic inference

Let $\mathbf{y}_k$ denote the vector of all sensor measurements available at sampling time k. Our general formulation applies to any combination of sensors as long as they output real numbers that can be stacked into $\mathbf{y}_k$. It accommodates missing data – which can be missing due to occlusion of optical markers, or because some sensors have lower sampling rates and do not provide data at all times. Calibration data, collected by asking subjects to assume specified postures, is also accommodated. The joint angles of the specified postures are treated as measurements of virtual goniometers and are included in $\mathbf{y}_k$.

Let $\mathbf{x}_k$ denote the vector of all latent variables that affect the sensor data: the global position and orientation of the body, the joint angles, the joint centers and rotation axes, the sizes of the limb segments, and the marker placements relative to the limb segments. Some of these variables vary over time while others remain constant but nevertheless need to be estimated. Derivatives of time-varying quantities can also be included, which makes it possible to introduce smoothness priors. Known parameters are implicit in the kinematic model and are not included in $\mathbf{x}_k$; everything included in $\mathbf{x}_k$ is automatically estimated.

Given measurements $\mathbf{y}_k$ as well as prior information, our objective is to estimate the most likely states $\widehat{\mathbf{x}}_k$ and their confidence intervals. Key to such inference problems is a generative model specifying the probability distribution of the measurement conditional on the state, as well as a dynamics model specifying the probability distribution of the next state conditional on the current state.

A. Generative models

The most commonly used generative model is of the form

$$\mathbf{y}_k = \mathbf{f}(\mathbf{x}_k) + \varepsilon_k, \quad \varepsilon_k \sim \mathcal{N}_\varepsilon(0; V) \tag{1}$$

where V is the covariance of the sensor noise ε. The noise is assumed Gaussian, which tends to be a reasonable assumption. The function $\mathbf{f}(\mathbf{x})$ is the expected value of $\mathbf{y}$ given $\mathbf{x}$, thus $p(\mathbf{y}|\mathbf{x}) = \mathcal{N}_{\mathbf{y}}(\mathbf{f}(\mathbf{x}); V)$. In motion capture applications $\mathbf{f}(\mathbf{x})$ corresponds to the forward kinematics.

The standard model (1) turns out to be overly restrictive, because it implies that the subtraction $\mathbf{y} - \mathbf{f}(\mathbf{x})$ is a sensible operation. However 3D rotations cannot be represented in a coordinate system where vector addition and subtraction make sense geometrically (see below). Thus we introduce a more general model of the form

$$\mathbf{r}(\mathbf{y}_k, \mathbf{x}_k) = \varepsilon_k, \quad \varepsilon_k \sim \mathcal{N}_\varepsilon(0; V) \tag{2}$$

where $\mathbf{r}(\mathbf{y}, \mathbf{x})$ is some function playing the role of a residual. Here $p(\mathbf{y}|\mathbf{x})$ is defined implicitly: it is such that $\mathbf{r}(\mathbf{y}, \mathbf{x})$ has conditional probability density $p(\mathbf{r}|\mathbf{x}) = \mathcal{N}_{\mathbf{r}}(0; V)$. Even though $p(\mathbf{r}|\mathbf{x})$ is still Gaussian, a function $\mathbf{r}(\mathbf{y}, \mathbf{x})$ which is non-linear in $\mathbf{y}$ will induce a non-Gaussian $p(\mathbf{y}|\mathbf{x})$, making (2) more general than (1). The two models become equivalent in the special case $\mathbf{r}(\mathbf{y}, \mathbf{x}) = \mathbf{y} - \mathbf{f}(\mathbf{x})$.

In addition to a generative model of the observations, we need a model of dynamics to describe how the states vary over time. In a sense this is another generative model because it specifies how the next state arises as a function of the current state. A complete model of dynamics would incorporate musculo-skeletal mechanics, muscle force production and statistics of muscle activations. However such models have so many unknown parameters that estimation performance is likely to be poor, in addition to being computationally inefficient. One can save a lot of effort and obtain better results by assuming a simplified linear model of the form

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + \omega_k, \quad \omega_k \sim \mathcal{N}_\omega(0; R) \tag{3}$$

A linear model affords more efficient computation and is sufficient to capture movement continuity. Continuity allows us to propagate information through time, and in particular estimate constant parameters by pooling information from multiple time frames. It is also useful in estimating time-varying parameters: given estimate $\widehat{\mathbf{x}}_k$ and estimation error covariance S_k at time k, the most likely next state (in the absence of new data) is $A\widehat{\mathbf{x}}_k$ and its covariance is $AS_kA^{\mathsf{T}} + R$. This induces a prior over the unknown state and enables estimation even if some of the sensor data is missing. Since the estimation algorithm is a form of iterative numerical optimization (see below), the prediction $A\widehat{\mathbf{x}}_k$ is further useful in initializing the iteration.

The structure of the dynamics model (3) can be adapted to the estimation problem at hand without modifying the rest of the framework. In the simplest model the state vector is partitioned as $\mathbf{x} = [\mathbf{x}^{\mathbf{w}}; \mathbf{x}^{\mathbf{p}}]$ where $\mathbf{x}^{\mathbf{w}}$ contains all constant parameters while $\mathbf{x}^{\mathbf{p}}$ contains all time-varying parameters. The dynamics of $\mathbf{x}^{\mathbf{p}}$ are modeled as a random walk. Thus

$$\mathbf{x}^{\mathbf{w}}_{k+1} = \mathbf{x}^{\mathbf{w}}_k, \quad \mathbf{x}^{\mathbf{p}}_{k+1} = \mathbf{x}^{\mathbf{p}}_k + \omega^{\mathbf{p}}_k \tag{4}$$

The matrices A, R needed to represent (4) in the form (3) are

$$A = I, \quad R = \begin{bmatrix} 0 & 0 \\ 0 & R^{\mathbf{p}} \end{bmatrix} \tag{5}$$

where $R^{\mathbf{p}}$ is the covariance matrix specifying how rapidly the time-varying parameters are expected to change.

A more sophisticated model, which captures not only the continuity but also the smoothness of natural movements, can be obtained by including time-derivatives of $\mathbf{x}^{\mathbf{p}}$. Consider the

state vector $\mathbf{x} = [\mathbf{x}^{\mathbf{w}}; \mathbf{x}^{\mathbf{p}}; \mathbf{x}^{\mathbf{v}}; \mathbf{x}^{\mathbf{a}}]$ where $\mathbf{x}^{\mathbf{v}} = d\mathbf{x}^{\mathbf{p}}/dt$ and $\mathbf{x}^{\mathbf{a}} = d\mathbf{x}^{\mathbf{v}}/dt$. Then the dynamics are

$$\begin{aligned} \mathbf{x}^{\mathbf{w}}_{k+1} &= \mathbf{x}^{\mathbf{w}}_k, \qquad \mathbf{x}^{\mathbf{p}}_{k+1} = \mathbf{x}^{\mathbf{p}}_k + \Delta \mathbf{x}^{\mathbf{v}}_k \\ & \qquad\quad \mathbf{x}^{\mathbf{v}}_{k+1} = \mathbf{x}^{\mathbf{v}}_k + \Delta \mathbf{x}^{\mathbf{a}}_k \\ & \qquad\quad \mathbf{x}^{\mathbf{a}}_{k+1} = \mathbf{x}^{\mathbf{a}}_k + \omega^{\mathbf{a}}_k \end{aligned} \tag{6}$$

where Δ is the time step. Here the movement is driven by the acceleration change $\omega^{\mathbf{a}}$. Such an assumption is equivalent to imposing a minimum-jerk smoothness prior and is motivated by the empirical success of the minimum-jerk model of human movement [14], [15].

We have experimented with (6) and found that although the resulting estimator can be more accurate, it also diverges more often. This is because errors in the velocity and acceleration estimates can accumulate, which can cause the next-state predictions of the sensor positions and orientations to be far from the correct values, which in turn can cause the optimization process to get trapped in local minima. Incorporating accelerometers and gyroscopes is likely to eliminate this failure mode (see Discussion). However typical motion capture applications rely on position and orientation sensors, in which case the simpler model (4) is recommended.

B. Recursive estimation

The estimation procedure is a generalization of the extended Kalman filter. At each time step k the probability density of $\mathbf{x}_k$ is approximated with the Gaussians

$$\begin{aligned} p(\mathbf{x}_k | \mathbf{y}_1 \cdots \mathbf{y}_{k-1}) &= \mathcal{N}_{\mathbf{x}}(\widetilde{\mathbf{x}}_k; P_k) \\ p(\mathbf{x}_k | \mathbf{y}_1 \cdots \mathbf{y}_{k-1}, \mathbf{y}_k) &= \mathcal{N}_{\mathbf{x}}(\widehat{\mathbf{x}}_k; S_k) \end{aligned} \tag{7}$$

$\mathcal{N}_{\mathbf{x}}(\widetilde{\mathbf{x}}_k; P_k)$ is the prior probability before $\mathbf{y}_k$ is observed while $\mathcal{N}_{\mathbf{x}}(\widehat{\mathbf{x}}_k; S_k)$ is the posterior probability after $\mathbf{y}_k$ is observed. Note that "prior" and "posterior" are used here in a recursive sense: the posterior at one time step yields the prior at the next time step.

The estimation procedure is initialized by specifying $\widehat{\mathbf{x}}_0; S_0$. The time-varying components of $\widehat{\mathbf{x}}_0$ should be set to some intermediate values, and their covariance should be large because the initial configuration of the body is typically unknown. Proper initialization of the constant components of $\widehat{\mathbf{x}}_0$ is more important and should incorporate any available information about skeletal parameters and marker placements. Setting the prior covariance for the constant components is also important. A small covariance implies that we are very confident in our guess. If that guess turns out to be inaccurate, the inaccuracies will persist for a long time even though the sensor data may contain enough information to correct them. If on the other hand the initial covariance is large, the estimator will make large changes on the basis of little data. Problem-dependent tuning of the initial covariance is likely to be needed. In an ideal Bayesian world prior knowledge would be quantified precisely, but in practice the shape of the prior distribution is often used as a control knob for fine-tuning the Bayesian inference machinery.

The recursive estimation procedure is now as follows. At time step k we are given $\widehat{\mathbf{x}}_{k-1}, S_{k-1}, \mathbf{y}_k$. First we "predict" the prior $\widetilde{\mathbf{x}}_k; P_k$ using the dynamics model, and then "correct" it in light of the new data $\mathbf{y}_k$ and compute the posterior $\widehat{\mathbf{x}}_k; S_k$. This is known as a predictor-corrector method. For linear-Gaussian dynamics of the form (3) the prediction is simple:

$$\widetilde{\mathbf{x}}_k = A\widehat{\mathbf{x}}_{k-1}, \quad P_k = AS_{k-1}A^{\mathsf{T}} + R \tag{8}$$

The correction involves two steps: computing the most likely state $\widehat{\mathbf{x}}_k$ via numerical optimization, and linearizing the observation model (2) around $\widehat{\mathbf{x}}_k$ in order to compute S_k. These two steps are discussed in turn. Suppressing the time index k, the residual $\mathbf{r}$ has conditional density $\mathcal{N}_{\mathbf{r}}(0; V)$ and the state $\mathbf{x}$ has prior density $\mathcal{N}_{\mathbf{x}}(\widetilde{\mathbf{x}}; P)$. Thus the joint density is

$$p(\mathbf{r}, \mathbf{x}) = p(\mathbf{r}|\mathbf{x})\, p(\mathbf{x}) = \mathcal{N}_{\mathbf{r}}(0; V)\, \mathcal{N}_{\mathbf{x}}(\widetilde{\mathbf{x}}; P) \tag{9}$$

The maximum aposteriori (MAP) estimate is the value of $\mathbf{x}$ which maximizes the above expression for given $\mathbf{y}, \widetilde{\mathbf{x}}$. Maximizing $p(\mathbf{r}, \mathbf{x})$ is equivalent to minimizing $-\log(p(\mathbf{r}, \mathbf{x}))$, which, up to an additive constant, is equal to

$$\ell(\mathbf{x}) = \frac{1}{2}\mathbf{r}(\mathbf{y}, \mathbf{x})^{\mathsf{T}} V^{-1} \mathbf{r}(\mathbf{y}, \mathbf{x}) + \frac{1}{2}(\mathbf{x} - \widetilde{\mathbf{x}})^{\mathsf{T}} P^{-1}(\mathbf{x} - \widetilde{\mathbf{x}})$$

The function $\ell(\mathbf{x})$ plays the role of a cost function. The first term encourages small residual (explaining the sensor data) while the second term encourages staying close to the prior (avoiding motion discontinuities). This is a nonlinear least squares problem. The most efficient method for solving such problems is usually the Gauss-Newton method. It requires computation of the Jacobian matrix

$$J(\mathbf{y}, \mathbf{x}) = \frac{\partial \mathbf{r}(\mathbf{y}, \mathbf{x})}{\partial \mathbf{x}} \tag{10}$$

The gradient of the cost function is

$$\nabla \ell(\mathbf{x}) = J(\mathbf{y}, \mathbf{x})^{\mathsf{T}} V^{-1} \mathbf{r}(\mathbf{y}, \mathbf{x}) + P^{-1}(\mathbf{x} - \widetilde{\mathbf{x}}) \tag{11}$$

Taking another derivative, the Hessian is

$$J(\mathbf{y}, \mathbf{x})^{\mathsf{T}} V^{-1} J(\mathbf{y}, \mathbf{x}) + P^{-1} + \frac{\partial J}{\partial \mathbf{x}} \times V^{-1}\mathbf{r} \tag{12}$$

The last term involves a tensor product because $\frac{\partial J}{\partial \mathbf{x}}$ is not a matrix but rather a tensor with rank 3. The essence of the Gauss-Newton is to ignore this term, and define the approximate Hessian

$$H(\mathbf{x}) = J(\mathbf{y}, \mathbf{x})^{\mathsf{T}} V^{-1} J(\mathbf{y}, \mathbf{x}) + P^{-1} \tag{13}$$

This avoids computation of second-order derivatives while still yielding second-order convergence. Note that if $\mathbf{r}$ is a linear function of $\mathbf{x}$, the second-order derivative $\frac{\partial J}{\partial \mathbf{x}}$ is zero and (13) becomes the exact Hessian. Now initialize $\widehat{\mathbf{x}} = \widetilde{\mathbf{x}}$ and minimize ℓ via the quasi-Newton iteration

$$\widehat{\mathbf{x}} \leftarrow \widehat{\mathbf{x}} - H(\widehat{\mathbf{x}})^{-1} \nabla \ell(\widehat{\mathbf{x}}) \tag{14}$$

The term P^{-1} in (13) makes H invertible even for $J = 0$. In general one should combine quasi-Newton iteration with a trust-region or a line-search method to ensure stability. We have implemented the latter, however it does not appear to be necessary in our application.

Note that both the Kalman filter and the extended Kalman filter are special cases corresponding to the first iteration of (14). When $\mathbf{r}$ is linear in $\mathbf{x}$, the cost function $\ell(\mathbf{x})$ is

quadratic and (14) converges in one iteration which coincides with the Kalman filter. In nonlinear problems, the extended Kalman filter still uses a single iteration even though additional iterations can improve accuracy. Our method iterates for as long as time permits or until convergence. This yields an adaptive procedure, which is as efficient as the extended Kalman filter when $\widetilde{\mathbf{x}}$ is close to the minimum of $\ell(\mathbf{x})$ but is more accurate (albeit slower) in less favorable conditions.

Once ℓ has been optimized and $\widehat{\mathbf{x}}$ has been found, the covariance S can be approximated by linearizing $\mathbf{r}(\mathbf{y},\mathbf{x})$:

$$\mathbf{r}(\mathbf{y},\mathbf{x}) \approx \mathbf{r}(\mathbf{y},\widehat{\mathbf{x}}) + J(\mathbf{y},\widehat{\mathbf{x}})(\mathbf{x}-\widehat{\mathbf{x}}) \tag{15}$$

Then the posterior covariance matrix is

$$S = H(\widehat{\mathbf{x}})^{-1} \tag{16}$$

The sensor noise covariance V can also be estimated. The standard method is the expectation-maximization (EM) algorithm [16], [17] which iterates between computing the sequence $\widehat{\mathbf{x}}_k$ and setting V to the covariance of the residuals:

$$V = \frac{1}{n}\sum_{k=1}^{n} \mathbf{r}(\mathbf{y}_k,\widehat{\mathbf{x}}_k)\,\mathbf{r}(\mathbf{y}_k,\widehat{\mathbf{x}}_k)^{\mathsf{T}} \tag{17}$$

The mean of $\mathbf{r}$ is not subtracted because the noise model is zero-mean. EM is an offline algorithm, however it can also be implemented online via the following recursion:

$$V_k = (1-\eta_k)\,V_{k-1} + \eta_k \mathbf{r}(\mathbf{y}_k,\widehat{\mathbf{x}}_k)\,\mathbf{r}(\mathbf{y}_k,\widehat{\mathbf{x}}_k)^{\mathsf{T}} \tag{18}$$

If the computation of $\widehat{\mathbf{x}}_k$ and $\mathbf{r}$ were not affected by V and the learning rate η decayed as $\eta_k = k^{-1}$, the online version would yield exactly the same result as the batch version. In online applications it may be advantageous to use learning rates which decay slower than k^{-1}, so as to avoid premature convergence to local minima. Note that estimation of V is different from estimation of $\mathbf{x}$ because the uncertainty about V is ignored. One could represent the latter via a Wishart distribution, however it is not clear if the resulting improvement would justify the added complexity.

C. Observability

A system is observable if the sensor data contains enough information to infer the state variables. Observability is usually analyzed in the limit of zero sensor noise. Even then, a system can be unobservable for a number of reasons. The most obvious reason is having fewer measurements than state variables. However counting is not always sufficient to determine observability. Consider the example in **Fig 1a** with 4 joint angles and one 3D position-and-orientation sensor. Such a sensor would normally provide 6 independent measurements, but here only 3 of the measurements are independent due to the planar geometry. Instead of counting, one should ask a more general question: Can the state variables be modified without affecting the sensor data? If so, the system is unobservable from the viewpoint of estimation (system identification is considered later).

The mathematical answer to the above question is well-known for linear systems of the form

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + \omega_k, \quad \mathbf{y}_k = C\mathbf{x}_k + \upsilon_k \tag{19}$$

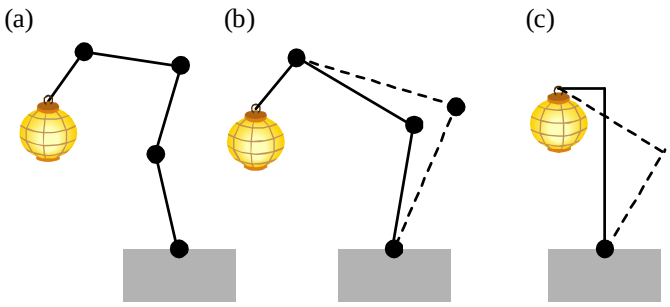

Fig. 1. Examples of unobservable systems. Suppose the room is dark and the kinematic mechanisms cannot be seen. Only the lanterns (denoting 3D position-and-orientation sensors) are visible. In these examples the motion of the lantern is insufficient to infer the motion (a) or structure (b,c) of the underlying mechanism. **(a)**– The joints (dots) can rotate without affecting the position and orientation of the last link where the sensor is rigidly attached. **(b,c)**– Pairs of mechanisms (solid vs. dashed lines) such that any feasible motion of the sensor could have resulted from either mechanism.

Suppose the noise terms ω_k, υ_k are negligibly small and let the state vector $\mathbf{x}_k$ have dimensionality d. The system is observable if and only if the observability matrix

$$O = \left[C;\ CA;\ \cdots\ CA^{d-1}\right] \tag{20}$$

has rank d, that is, it has full column-rank. This is because in the small-noise limit the observations are

$$\left[\mathbf{y}_k;\ \mathbf{y}_{k+1};\ \cdots\ \mathbf{y}_{k+d-1}\right] = O\mathbf{x}_k \tag{21}$$

The rank condition is equivalent to saying that every change in the state causes some change in future observations. Only the first d observations are relevant because for all n the matrix A^n is a linear combination of $A, \cdots A^{d-1}$.

This classic result is not applicable here because the measurements are nonlinear. Also, our dynamics model $A = I$ effectively reduces the observability matrix to $O = C$ and makes the above condition trivial. Nevertheless the main ideas behind the classic result are useful in developing an observability test for our problem. In the remainder of this section we ignore the observation noise, the dynamics model, and the issue of local minima. Suppose we are given a collection of noiseless sensor measurements and our state estimates obtained from those measurements happen to be correct – in which case all residuals are zero. If the system is observable, it should be impossible to perturb the estimates while keeping the residuals equal to zero. Thus the issue of observability comes down to sensitivity of the residuals with respect to changes in the state variables. This suggests looking at the Jacobian matrix, which will now be partitioned as

$$J_k = \left[J_k^{\mathbf{W}}\ J_k^{\mathbf{P}}\right] = \left[\frac{\partial \mathbf{r}(\mathbf{y}_k,\widehat{\mathbf{x}}_k)}{\partial \widehat{\mathbf{x}}_k^{\mathbf{W}}}\ \frac{\partial \mathbf{r}(\mathbf{y}_k,\widehat{\mathbf{x}}_k)}{\partial \widehat{\mathbf{x}}_k^{\mathbf{P}}}\right] \tag{22}$$

The system is observable for estimation purposes if and only if $J_k^{\mathbf{P}}$ has full column-rank for all k. This resembles the classic result with $A = I$ and C corresponding to $J_k^{\mathbf{P}}$, except that here the measurements are non-linear and so the Jacobian is state-dependent. Once a generative model is constructed, this observability condition can be checked numerically by generating a database of random states, predicting the sensor

measurements, evaluating the matrices $J_k^{\mathbf{p}}$ and computing their rank. The size n of this database should be larger than the number of measurements or state variables.

Next consider observability from the viewpoint of system identification. **Fig 1b,c** gives examples where the system identification problem cannot be solved even though the state estimation problem is solvable. Counting is again insufficient and one needs sensitivity analysis. However the situation here is more complex. Looking at $J_k^{\mathbf{w}}$ alone can be misleading, because the constant parameters $\mathbf{x}^{\mathbf{w}}$ are estimated using data from all time steps and so the system may be observable even if all $J_k^{\mathbf{w}}$ are singular. Suppose that the filter yielding time-varying estimates $\widehat{\mathbf{x}}_k^{\mathbf{w}}$ is replaced with a batch method yielding a single estimate $\widehat{\mathbf{x}}^{\mathbf{w}}$. Let $\delta\widehat{\mathbf{x}}^{\mathbf{w}}$ denote an arbitrary small perturbation to $\widehat{\mathbf{x}}^{\mathbf{w}}$. Can such a perturbation leave all residuals at all time steps unchanged? Although the perturbation is likely to change at least some of the predicted sensor measurements, it may be possible to cancel those changes by appropriate perturbations $\delta\widehat{\mathbf{x}}_k^{\mathbf{p}}$ to the estimates of the time-varying parameters. This reasoning yields the following condition. The model is observable for system identification purposes if and only if vectors $\delta\widehat{\mathbf{x}}^{\mathbf{w}} \neq 0$ and $\delta\widehat{\mathbf{x}}_k^{\mathbf{p}}$ satisfying

$$J_k^{\mathbf{w}}\delta\widehat{\mathbf{x}}^{\mathbf{w}} + J_k^{\mathbf{p}}\delta\widehat{\mathbf{x}}_k^{\mathbf{p}} = 0, \quad k = 1 \cdots n \tag{23}$$

cannot be found. This reduces to a matrix rank condition as follows. The n equations (23) are equivalent to

$$M\left[\delta\widehat{\mathbf{x}}_1^{\mathbf{p}};\ \delta\widehat{\mathbf{x}}_2^{\mathbf{p}};\ \cdots\ \delta\widehat{\mathbf{x}}_n^{\mathbf{p}};\ \delta\widehat{\mathbf{x}}^{\mathbf{w}}\right] = 0 \tag{24}$$

where the extended observability matrix M is

$$M = \begin{bmatrix} J_1^{\mathbf{p}} & & & J_1^{\mathbf{w}} \\ & J_2^{\mathbf{p}} & & J_2^{\mathbf{w}} \\ & & \ddots & \vdots \\ & & J_n^{\mathbf{p}} & J_n^{\mathbf{w}} \end{bmatrix} \tag{25}$$

Observability as defined above is equivalent to the matrix M having full column-rank. That matrix is normally large, but it is also sparse which can be taken advantage of, as follows. Construct the symmetric positive semi-definite matrix $M^{\mathsf{T}}M$, which is also sparse, and look for its smallest eigenvalue using a sparse eigensolver. The smallest eigenvalue of $M^{\mathsf{T}}M$ is zero if and only if M is column-rank-defficient. The number of zero eigenvalues is equal to the dimensionality of the null-space. This yields our numerical test for observability.

III. Kinematics

The above estimation procedure is straightforward as long as the residuals $\mathbf{r}(\mathbf{y}_k, \mathbf{x}_k)$ and Jacobians $J(\mathbf{y}_k, \mathbf{x}_k)$ can be computed. Here we describe a kinematic modeling approach which is sufficiently general and yet allows efficient computation of $\mathbf{r}$ and J. In the remainder of this section we focus on a single time step and drop the time index k.

The simplest sensor, which does not require forward kinematics computation, is a joint angle sensor or goniometer. Its predicted output is the linear function $a + b\theta$ where θ is the joint angle being measured, a is the offset and b is the gain. The scalars a, b, θ are part of the state vector $\mathbf{x}$, in particular a, b are part of $\mathbf{x}^{\mathbf{w}}$ and θ is part of $\mathbf{x}^{\mathbf{p}}$. The goniometer's output y and residual r are part of the measurement vector $\mathbf{y}$ and residual vector $\mathbf{r}(\mathbf{y}, \mathbf{x})$ respectively. Then

$$r = y - a - b\theta \tag{26}$$

The goniometer's Jacobian, which is part of the Jacobian matrix $J(\mathbf{y}, \mathbf{x})$, has elements

$$\frac{\partial r}{\partial a} = -1, \quad \frac{\partial r}{\partial b} = -\theta, \quad \frac{\partial r}{\partial \theta} = -b \tag{27}$$

If the parameters a, b are known and do not need to be estimated, they become implicit in the kinematic model and are omitted from $\mathbf{x}$. This is the case when a virtual goniometer is used to specify calibration data: since the instructed joint angles which subjects were attempting to produce are known, they can be entered directly in the vector $\mathbf{y}$ and the parameters can be set to $a = 0$, $b = 1$.

The other two sensor types, discussed in detail below, are 3D position and orientation sensors. These are the sensors most commonly used in applications. Some motion capture devices (usually optical) measure only position, while others (usually electromagnetic) measure both position and orientation. One could also use magnetometers which only measure orientation. When both position and orientation data are available, they will be treated as being generated by independent sensors. Our framework allows any combination of such sensors to be used simultaneously. In case of missing sensor data, the corresponding elements of $\mathbf{y}, \mathbf{r}, J$ are removed and the estimation procedure remains formally identical.

A. Kinematic modeling

The body is assumed to have the topology of a tree – which is almost always the case in practice. The tree is composed of segments connected with joints. Each segment is associated with a 3D coordinate frame. The joints are spatial transformations which determine the position and orientation of each frame relative to the frame of the parent segment. Here "segments" and "joints" refer not only to limb segments and anatomical joints but also to other entities with the same kinematic properties. Any spatial transformation between a parent and a child frame is called "joint" even if it is constant. For example, forearm length is the parameter of a sliding joint which translates the frame from the elbow to the wrist. Segments can also have non-anatomical meaning: a virtual segment is created for every sensor attached to the body. The sensor's position and orientation relative to the underlying limb segment are described by the joint parameters, while the sensor's expected measurement corresponds to the position (respectively orientation) of the virtual segment's frame.

Kinematic trees are constructed by connecting segments with joints. The allowed types of joints are chosen so as to achieve a balance between expressive power and ease of derivation as well as implementation. We define the following four types of joints:

joint type	**description**	**d.o.f.**
T (translation)	translation in any direction	3
R (rotation)	rotation around any axis	3
S (sliding)	translation in predefined direction	1
H (hinge)	rotation around predefined axis	1

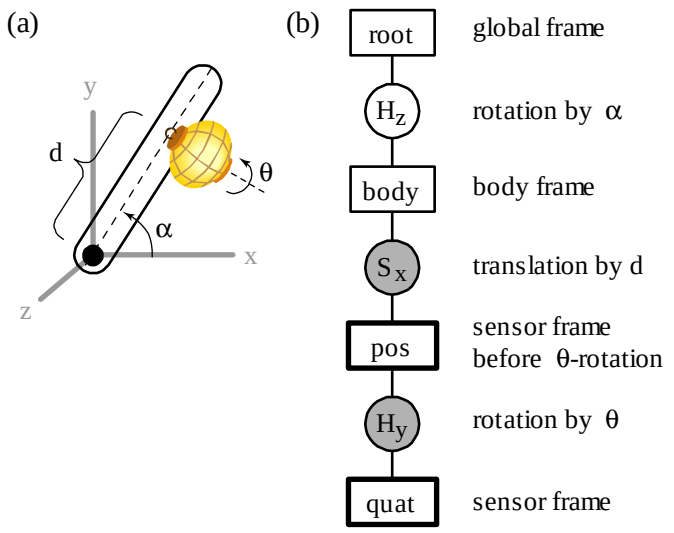

Fig. 2. A simple example of a kinematic mechanism **(a)** and its corresponding kinematic tree **(b)**. See text for details.

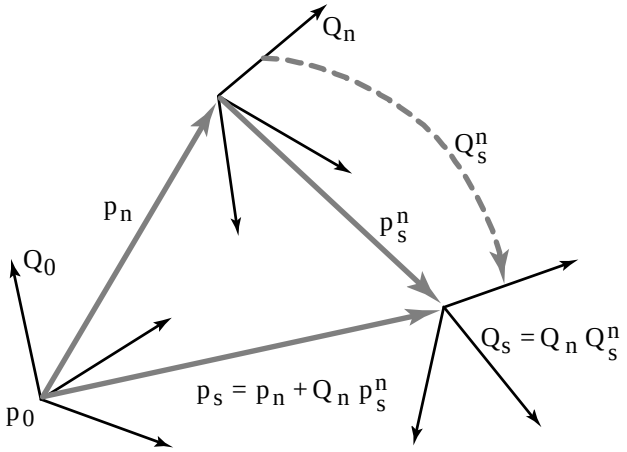

Fig. 3. Schematic illustration of the relations between spatial frames. Superscripts indicate the frame with respect to which the quantity is measured. Subscripts indicate the frame being described by the quantity.

Joints of **type T** correspond to arbitrary 3D translation of the child frame relative to the parent frame. They have three parameters (included in the state vector $\mathbf{x}$) which specify the translation vector. Joints of **type R** correspond to arbitrary 3D rotation. They have three parameters specifying an "angle-axis" vector. The length of that vector is the rotation angle while its direction is the rotation axis. Joints of **type S** correspond to sliding in a predefined direction. They have one parameter specifying the sliding distance. The direction is given by a unit vector which is implicit in the kinematic model and is not part of the state vector. Finally, joints of **type H** correspond to rotation around a predefined axis. They have one parameter specifying the rotation angle. The rotation axis is given by a unit vector implicit in the kinematic model.

A simple example illustrating the use of these joint types is given in **Fig 2**. A rigid body rotates within the x-y plane around the z-axis. The rotation angle is α. A 3D position-and-orientation sensor is mounted on the body, at a distance d from the pivot point on the x-axis of the body (dashed). The sensor is rotated by an angle θ around the y-axis of the body. Thus the state vector is 3-dimensional: $\mathbf{x} = [d; \theta; \alpha]$. The measurement vector $\mathbf{y}$ is 7-dimensional and contains the 3D position followed by a quaternion specifying the 3D orientation. The residual vector $\mathbf{r}$ is only 6-dimensional because orientation errors are expressed in the same format as the R-type joint. Thus the Jacobian J is a 6x3 matrix.

The corresponding kinematic tree is shown in **Fig 2b**. Rectangles denote segments/frames while circles denote joints/transformations. Each child segment is connected to its parent segment with a joint specifying the spatial transformation between the parent and child frames. Joints are labeled with their type. Sliding and hinge joints (S and H) are indexed with the coordinate axis serving as their predefined direction. In general this direction does not have to coincide with one of the coordinate axes of the parent frame, however such generality is not necessary here. Shaded circles correspond to constant transformations (part of $\mathbf{x}^{\mathbf{w}}$) while empty circles corresponds to time-varying transformations (part of $\mathbf{x}^{\mathbf{p}}$). Rectangles with heavy outlines denote frames whose position or orientation are being measured by 3D sensors. Note that a single sensor is represented with two virtual segments – one for the position measurement and another for the orientation measurement. The position measurement depends on d but not on θ, thus it can be taken before the rotation by θ. In this way we avoid dummy segments and keep the number of sensor types to a minimum.

B. Forward kinematics

The segments in the kinematic tree are enumerated in depth-first order. The root segment is 0 and the remaining segments satisfy $pa\,(s) < s$ where $pa\,(s)$ is the parent of s. Each segment s has an associated frame with position $\mathbf{p}_s \in \mathbb{R}^3$ and orientation $Q_s \in SO\,(3)$ measured relative to the global (root) frame $\mathbf{p}_0 = 0$, $Q_0 = I$. Recall that $SO\,(3)$ denotes the set of 3x3 orthonormal matrices which parameterize the space of 3D rotations. Composition of 3D rotations corresponds to matrix multiplication in $SO\,(3)$. The inverse rotation is $Q_s^{-1} = Q_s^{\mathsf{T}}$. The columns of Q_s are the unit vectors specifying the axes of frame s relative to the global frame. The relations between frames are illustrated in **Fig 3**.

Throughout this section we use the following indexing notation. $\mathbf{p}_s^n$ refers to a quantity (position in this case) which is measured relative to frame n and is a feature of frame s. When the superscript is zero it can be omitted, thus $\mathbf{p}_s$ is a shortcut for $\mathbf{p}_s^0$. The vector composed of all elements of $\mathbf{x}$ specific to frame s is denoted $\mathbf{x}_{\{s\}}$. This notation is defined only for the "composite" vectors $\mathbf{x}, \mathbf{y}, \mathbf{r}$. For example, if s is the virtual segment corresponding to a goniometer then $\mathbf{x}_{\{s\}}$ is the vector $[a; b; \theta]$ described earlier. $J_{\{s,n\}}$ denotes the corresponding sub-matrix of the Jacobian. $[\mathbf{z}]_{2:4}$ denotes the vector composed of elements $2, 3, 4$ of $\mathbf{z}$. If the argument is a matrix, as in $[Z]_{2:4}$, this notation refers to the matrix composed of rows $2, 3, 4$ of Z.

We now return to the construction of the kinematic model. The joint connecting segments s and $pa\,(s)$ specifies where

frame s is relative to frame $pa\,(s)$. The parameters $\mathbf{x}_{\{s\}}$ define a translation $\mathbf{t}\left(\mathbf{x}_{\{s\}}\right) \in \mathbb{R}^3$ and a rotation $U\left(\mathbf{x}_{\{s\}}\right) \in SO\,(3)$ of frame s relative to its parent frame $pa\,(s)$:

$$\mathbf{p}_s^{pa(s)} = \mathbf{t}\left(\mathbf{x}_{\{s\}}\right), \quad Q_s^{pa(s)} = U\left(\mathbf{x}_{\{s\}}\right) \tag{28}$$

Therefore frames s and $pa\,(s)$ are related as

$$\begin{aligned} \mathbf{p}_s &= \mathbf{p}_{pa(s)} + Q_{pa(s)}\mathbf{t}\left(\mathbf{x}_{\{s\}}\right) \\ Q_s &= Q_{pa(s)}U\left(\mathbf{x}_{\{s\}}\right) \end{aligned} \tag{29}$$

Given vectors $\mathbf{x}_{\{s\}}$ and functions $\mathbf{t}, U$ we can apply (29) for all s in increasing order and obtain all $\mathbf{p}_s, Q_s$. At the end of this computation we have the global positions and orientations of all frames. The functions $\mathbf{t}, U$ and their parameters $\mathbf{x}_{\{s\}}$ depend on the joint types and will be defined later.

Although (29) is mathematically correct, it is not the best way to implement forward kinematics. For a long sequence of matrix multiplications numerical errors tend to accumulate and make the result non-orthonormal. Instead it is better to replace the matrices Q, U with the corresponding quaternions $\mathbf{q}, \mathbf{u}$ and to accumulate 3D rotations via quaternion multiplication. To this end, let us review the relevant properties of quaternions. A unit quaternion $\mathbf{q} \in \mathbb{R}^4$ with elements

$$\begin{aligned} \mathbf{q} &= \left[\cos\frac{\alpha}{2};\ v_1 \sin\frac{\alpha}{2};\ v_2 \sin\frac{\alpha}{2};\ v_3 \sin\frac{\alpha}{2}\right] \\ &= \left[\cos\frac{\alpha}{2};\ \mathbf{v}\sin\frac{\alpha}{2}\right] \end{aligned} \tag{30}$$

represents 3D rotation by an angle α around the axis given by the unit vector $\mathbf{v} = [v_1; v_2; v_3]$. The zero rotation corresponds to $\mathbf{q} = [1; 0; 0; 0]$. Rotation opposite to $\mathbf{q}$ is given by the conjugate quaternion $\overline{\mathbf{q}}$ with elements

$$\overline{\mathbf{q}} = [q_1; -q_2; -q_3; -q_4] \tag{31}$$

The vector $\mathbf{q}$ defined in (30) is a unit vector which lies on the surface of the unit sphere $S^3 \subset \mathbb{R}^4$. Every 3D rotation can be represented in this way, so we can transform between S^3 and $SO\,(3)$. The matrix Q corresponding to the quaternion $\mathbf{q}$ is

$$\begin{matrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2\,(q_2 q_3 - q_1 q_4) & 2\,(q_2 q_4 + q_1 q_3) \\ 2\,(q_2 q_3 + q_1 q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2\,(q_3 q_4 - q_1 q_2) \\ 2\,(q_2 q_4 - q_1 q_3) & 2\,(q_3 q_4 + q_1 q_2) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{matrix}$$

The inverse transformation is

$$q_1 = \frac{1}{2}\sqrt{\mathrm{trace}\,(Q) + 1} \tag{32}$$

$$\begin{bmatrix} 0 & -q_4 & q_3 \\ q_4 & 0 & -q_2 \\ -q_3 & q_2 & 0 \end{bmatrix} = \frac{1}{2q_1}\left(Q - Q^{\mathsf{T}}\right)$$

Composition of 3D rotations corresponds to quaternion multiplication. If $\mathbf{q}$ and $\mathbf{u}$ are two quaternions, their product $\mathbf{q} * \mathbf{u}$ in component form is

$$\mathbf{q} * \mathbf{u} = \begin{bmatrix} q_1 u_1 - q_2 u_2 - q_3 u_3 - q_4 u_4 \\ q_1 u_2 + q_2 u_1 + q_3 u_4 - q_4 u_3 \\ q_1 u_3 - q_2 u_4 + q_3 u_1 + q_4 u_2 \\ q_1 u_4 + q_2 u_3 - q_3 u_2 + q_4 u_1 \end{bmatrix} \tag{33}$$

The conjugate quaternion plays the role of an inverse under quaternion multiplication. Similarly to the matrix identity $(QU)^{-1} = U^{-1}Q^{-1}$, here we have

$$\overline{\mathbf{q} * \mathbf{u}} = \overline{\mathbf{u}} * \overline{\mathbf{q}} \tag{34}$$

A vector $\mathbf{t} \in \mathbb{R}^3$ can be rotated via matrix multiplication, or alternatively via quaternion multiplication as follows. If $\mathbf{q}$ is the quaternion corresponding to Q, then

$$Q\mathbf{t} = \left[\mathbf{q} * \widetilde{\mathbf{t}} * \overline{\mathbf{q}}\right]_{2:4} \tag{35}$$

where the augmented vector $\widetilde{\mathbf{t}}$ is defined as

$$\widetilde{\mathbf{t}} = [0; \mathbf{t}] \tag{36}$$

This completes our review of quaternions. The forward kinematics (29) can now be expressed in quaternion form:

$$\begin{aligned} \mathbf{p}_s &= \mathbf{p}_{pa(s)} + \left[\mathbf{q}_{pa(s)} * \widetilde{\mathbf{t}}\left(\mathbf{x}_{\{s\}}\right) * \overline{\mathbf{q}}_{pa(s)}\right]_{2:4} \\ \mathbf{q}_s &= \mathbf{q}_{pa(s)} * \mathbf{u}\left(\mathbf{x}_{\{s\}}\right) \end{aligned} \tag{37}$$

This form not only allows more efficient and accurate numerical computation, but also facilitates the derivation of formulas for the Jacobians as shown below.

C. Position and orientation sensors

Once the forward kinematics are computed, it is straightforward to compute the residuals for 3D position and orientation sensors. Let s be a virtual segment for a position sensor whose measurement is $\mathbf{y}_{\{s\}} \in \mathbb{R}^3$. The measurement predicted by the generative model is the frame's position $\mathbf{p}_s$. Thus the residual vector $\mathbf{r}_{\{s\}} \in \mathbb{R}^3$ is defined as

$$\mathbf{r}_{\{s\,:\,\mathrm{pos}\}} = \mathbf{y}_{\{s\}} - \mathbf{p}_s \tag{38}$$

The notation $s : \mathrm{pos}$ serves to remind us that s is a position sensor and disambiguates from (39) below.

Next, let s be a virtual segment for an orientation sensor whose measurement is $\mathbf{y}_{\{s\}} \in \mathbb{R}^4$. The predicted measurement is the frame's orientation $\mathbf{q}_s$. Since both $\mathbf{y}_{\{s\}}$ and $\mathbf{q}_s$ are real-valued vectors with the same dimensionality, one may be tempted to subtract them and define the residual as $\mathbf{y}_{\{s\}} - \mathbf{q}_s$. However this does not make sense geometrically, and in practice leads to poor estimation performance. The problem is that the length of the above vector depends not only on the magnitude of the orientation error but also on the coordinate frame in which $\mathbf{y}_{\{s\}}$ and $\mathbf{q}_s$ are expressed. Ideally the length of the residual vector would be fully determined by the amount of rotation α needed to align the 3D orientations $\mathbf{y}_{\{s\}}$ and $\mathbf{q}_s$. This alignment operation can be expressed as a quaternion $\mathbf{z}$ which satisfies $\mathbf{q}_s * \mathbf{z} = \mathbf{y}_{\{s\}}$. The solution is $\mathbf{z} = \overline{\mathbf{q}}_s * \mathbf{y}_{\{s\}}$. From (30) we see that the last three elements of a unit quaternion form a vector with length $\sin\frac{\alpha}{2}$ independent of the coordinate system. Thus we define the orientation residual $\mathbf{r}_{\{s\}} \in \mathbb{R}^3$ as the last three elements of the vector $\mathbf{z}$:

$$\mathbf{r}_{\{s\,:\,\mathrm{orient}\}} = 2\left[\overline{\mathbf{q}}_s * \mathbf{y}_{\{s\}}\right]_{2:4} \tag{39}$$

For small orientation error α this yields

$$\left\|\mathbf{r}_{\{s\}}\right\| = 2\sin\frac{\alpha}{2} \approx \alpha \tag{40}$$

For large α the function $2\sin\frac{\alpha}{2}$ saturates, making the estimator robust to orientation outliers. Robustness with respect to position outliers can also be incorporated. For example, one can apply the nonlinear transformation $c\tanh\left(\frac{r_i}{c}\right)$ to the elements

r_i of the position residual defined in (38). The constant $c > 0$ sets the amount of saturation.

Given the position and orientation residuals (38, 39), the Jacobians can be computed by differentiating with respect to all the state variables. Let s be a virtual segment for either a position or an orientation sensor, and n be any other segment. As before, let $\mathbf{x}_{\{n\}}$ denote the part of the state vector $\mathbf{x}$ which parameterizes the 3D translation $\mathbf{t}\left(\mathbf{x}_{\{n\}}\right)$ and quaternion rotation $\mathbf{u}\left(\mathbf{x}_{\{n\}}\right)$ of frame n relative to its parent $pa\left(n\right)$. Our goal is to compute the matrix

$$J_{\{s,n\}} = \frac{\partial \mathbf{r}_{\{s\}}}{\partial \mathbf{x}_{\{n\}}} = \frac{\partial \mathbf{r}_{\{s\}}}{\partial \mathbf{t}\left(\mathbf{x}_{\{n\}}\right)} \frac{\partial \mathbf{t}\left(\mathbf{x}_{\{n\}}\right)}{\partial \mathbf{x}_{\{n\}}} + \frac{\partial \mathbf{r}_{\{s\}}}{\partial \mathbf{u}\left(\mathbf{x}_{\{n\}}\right)} \frac{\partial \mathbf{u}\left(\mathbf{x}_{\{n\}}\right)}{\partial \mathbf{x}_{\{n\}}} \tag{41}$$

The terms $\frac{\partial \mathbf{t}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ depend on the joint type and will be described in the next section. Here we only describe the computation of $\frac{\partial \mathbf{r}}{\partial \mathbf{t}}$ and $\frac{\partial \mathbf{r}}{\partial \mathbf{u}}$. If n is not an ancestor of s, that is, n is not in the path from s to the root of the tree, then $\mathbf{r}_{\{s\}}$ does not depend on $\mathbf{x}_{\{n\}}$ and we have $J_{\{s,n\}} = 0$. Thus we only need to compute $J_{\{s,n\}}$ when n is an ancestor of s. Algorithmically, all ancestors of s can be found by initializing $n = s$ and iterating $n \leftarrow pa\left(n\right)$ until $n = 0$.

Computing the partial derivatives of $\mathbf{r}$ with respect to $\mathbf{t}$ is straightforward. Translation $\mathbf{t}$ of frame n affects the position of frame s but not its orientation. Therefore, when s is an orientation sensor, we have

$$\frac{\partial \mathbf{r}_{\{s\,:\,\mathrm{orient}\}}}{\partial \mathbf{t}\left(\mathbf{x}_{\{n\}}\right)} = 0 \tag{42}$$

When s is a position sensor, its predicted measurement $\mathbf{p}_s$ is affected by $\mathbf{x}_{\{n\}}$ via translation by $Q_{pa(n)}\mathbf{t}\left(\mathbf{x}_{\{n\}}\right)$. Since $\mathbf{p}_s$ appears with a negative sign in the residual (38), we have

$$\frac{\partial \mathbf{r}_{\{s\,:\,\mathrm{pos}\}}}{\partial \mathbf{t}\left(\mathbf{x}_{\{n\}}\right)} = -Q_{pa(n)} \tag{43}$$

Computing the partial derivatives of $\mathbf{r}$ with respect to $\mathbf{u}$ is more complicated. First, let s be an orientation sensor. From (37) the predicted measurement is

$$\mathbf{q}_s = \mathbf{q}_n * \mathbf{q}_s^n = \mathbf{q}_{pa(n)} * \mathbf{u}\left(\mathbf{x}_{\{n\}}\right) * \mathbf{q}_s^n \tag{44}$$

and so the orientation residual (39) can be written as

$$\mathbf{r}_{\{s\}} = 2\left[\overline{\mathbf{q}}_s^n * \overline{\mathbf{u}}\left(\mathbf{x}_{\{n\}}\right) * \overline{\mathbf{q}}_{pa(n)} * \mathbf{y}_{\{s\}}\right]_{2:4} \tag{45}$$

Then the derivative is

$$\frac{\partial \mathbf{r}_{\{s\,:\,\mathrm{orient}\}}}{\partial \mathbf{u}\left(\mathbf{x}_{\{n\}}\right)} = 2\left[\overline{\mathbf{q}}_s^n * \frac{\partial \overline{\mathbf{u}}}{\partial \mathbf{u}} * \overline{\mathbf{q}}_{pa(n)} * \mathbf{y}_{\{s\}}\right]_{2:4} \tag{46}$$

This notation means that we treat each column of the 4x4 matrix $\frac{\partial \overline{\mathbf{u}}}{\partial \mathbf{u}}$ as a quaternion, apply the rules of quaternion multiplication, assemble the four resulting quaternions into another 4x4 matrix, and keep the last three rows. From the definition of the quaternion conjugate it follows that $\frac{\partial \overline{\mathbf{u}}}{\partial \mathbf{u}}$ is a diagonal matrix with elements $(1, -1, -1, -1)$.

Finally, let s be a position sensor. Its predicted measurement can be written as

$$\mathbf{p}_s = \mathbf{p}_n + \left[\mathbf{q}_n * \widetilde{\mathbf{p}}_s^n * \overline{\mathbf{q}}_n\right]_{2:4} \tag{47}$$

$\mathbf{q}_n$ depends on $\mathbf{u}\left(\mathbf{x}_{\{n\}}\right)$ as in (37), thus the derivative is

$$\frac{\partial \mathbf{r}_{\{s\,:\,\mathrm{pos}\}}}{\partial \mathbf{u}\left(\mathbf{x}_{\{n\}}\right)} = -\left[\mathbf{q}_{pa(n)} * \frac{\partial \mathbf{u}}{\partial \mathbf{u}} * \widetilde{\mathbf{p}}_s^n * \overline{\mathbf{q}}_n + \mathbf{q}_n * \widetilde{\mathbf{p}}_s^n * \frac{\partial \overline{\mathbf{u}}}{\partial \mathbf{u}} * \overline{\mathbf{q}}_{pa(n)}\right]_{2:4} \tag{48}$$

This notation is the same as in (46), the summation is performed element-wise, and $\frac{\partial \mathbf{u}}{\partial \mathbf{u}} = I$.

D. Joint types

Here we define the translation and rotation functions $\mathbf{t}\left(\mathbf{x}_{\{n\}}\right)$, $\mathbf{u}\left(\mathbf{x}_{\{n\}}\right)$ as well as their parameterizations $\mathbf{x}_{\{n\}}$, and compute the derivatives $\frac{\partial \mathbf{t}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$. Note that for joints of type T or S there is no rotation, so $\mathbf{u} = [1; 0; 0; 0]$. For joints of type R or H there is no translation, so $\mathbf{t} = [0; 0; 0]$.

For joints of type T, the parameterization $\mathbf{x}_{\{n\}} \in \mathbb{R}^3$ is a vector specifying the translation directly:

$$\mathbf{t}\left(\mathbf{x}_{\{n\,:\,\mathrm{T}\}}\right) = \mathbf{x}_{\{n\}}, \quad \frac{\partial \mathbf{t}\left(\mathbf{x}_{\{n\}}\right)}{\partial \mathbf{x}_{\{n\}}} = I \tag{49}$$

For joints of type S, $\mathbf{x}_{\{n\}} \in \mathbb{R}$ is a scalar specifying the sliding distance. The sliding direction is given by a joint-specific unit vector $\mathbf{v}$ measured relative to frame $pa\left(n\right)$. It is an implicit parameter of the kinematic model and is not included in the state vector. Thus the translation and its derivative are

$$\mathbf{t}\left(\mathbf{x}_{\{n\,:\,\mathrm{S}\}}\right) = \mathbf{x}_{\{n\}}\mathbf{v}, \quad \frac{\partial \mathbf{t}\left(\mathbf{x}_{\{n\}}\right)}{\partial \mathbf{x}_{\{n\}}} = \mathbf{v} \tag{50}$$

For joints of type H, $\mathbf{x}_{\{n\}} \in \mathbb{R}$ is a scalar specifying the rotation angle. The rotation axis is given by a joint-specific unit vector $\mathbf{v}$. From the quaternion definition (30) we have

$$\mathbf{u}\left(\mathbf{x}_{\{n\,:\,\mathrm{H}\}}\right) = \left[\cos\frac{\mathbf{x}_{\{n\}}}{2};\ \mathbf{v}\sin\frac{\mathbf{x}_{\{n\}}}{2}\right] \qquad \frac{\partial \mathbf{u}\left(\mathbf{x}_{\{n\}}\right)}{\partial \mathbf{x}_{\{n\}}} = \left[-\frac{1}{2}\sin\frac{\mathbf{x}_{\{n\}}}{2};\ \frac{1}{2}\mathbf{v}\cos\frac{\mathbf{x}_{\{n\}}}{2}\right] \tag{51}$$

For joints of type R, $\mathbf{x}_{\{n\}} \in \mathbb{R}^3$ is a vector specifying both the rotation angle and the rotation axis. In particular, the rotation angle is $\alpha = \left\|\mathbf{x}_{\{n\}}\right\|$ and the rotation axis is given by the unit vector $\mathbf{x}_{\{n\}}/\alpha$. The transformation from this angle-axis vector $\mathbf{x}_{\{n\}}$ to the corresponding quaternion $\mathbf{u}\left(\mathbf{x}_{\{n\}}\right)$ is called the *exponential map*. It has a number of appealing properties [18], which is why we have chosen it to parameterize arbitrary 3D rotations. One can verify that the exponential map and its derivative are

$$\mathbf{u}\left(\mathbf{x}_{\{n\,:\,\mathrm{R}\}}\right) = \left[\cos\frac{\alpha}{2};\ \mathbf{x}_{\{n\}}s\left(\alpha\right)\right] \qquad \frac{\partial \mathbf{u}\left(\mathbf{x}_{\{n\}}\right)}{\partial \mathbf{x}_{\{n\}}} = \left[-\frac{s\left(\alpha\right)}{2}\mathbf{x}_{\{n\}}^{\mathsf{T}};\ c\left(\alpha\right)\mathbf{x}_{\{n\}}\mathbf{x}_{\{n\}}^{\mathsf{T}} + s\left(\alpha\right)I\right] \tag{52}$$

where the shortcuts $s\left(\alpha\right), c\left(\alpha\right)$ are defined as

$$s\left(\alpha\right) = \frac{1}{\alpha}\sin\frac{\alpha}{2}, \quad c\left(\alpha\right) = \frac{1}{\alpha^2}\left(\frac{1}{2}\cos\frac{\alpha}{2} - s\left(\alpha\right)\right) \tag{53}$$

The functions $s\left(\alpha\right)$ and $c\left(\alpha\right)$ involve division by α and cannot be computed directly when α is very small. Nevertheless they

are well-behaved in the limit $\alpha \to 0$ and can be approximated via a Taylor series expansion. Using the fact that

$$\sin\alpha = \alpha - \frac{\alpha^3}{6} + \frac{\alpha^5}{120} + o\left(\alpha^7\right) \tag{54}$$
$$\cos\alpha = 1 - \frac{\alpha^2}{2} + \frac{\alpha^4}{24} + o\left(\alpha^6\right)$$

the Taylor series expansions of $s(\alpha)$ and $c(\alpha)$ are

$$s(\alpha) = \frac{1}{2} - \frac{\alpha^2}{48} + o\left(\alpha^4\right) \tag{55}$$
$$c(\alpha) = -\frac{1}{24} + \frac{\alpha^2}{960} + o\left(\alpha^4\right)$$

This completes the computation of the residual vector $\mathbf{r}(\mathbf{y}, \mathbf{x})$ and the Jacobian matrix $J(\mathbf{y}, \mathbf{x})$ given the state vector $\mathbf{x}$ and the measurement vector $\mathbf{y}$.

IV. Simulation results

We now illustrate the numerical accuracy of the above estimation method with three examples. The first example uses synthetic data, obtained by simulating the noisy outputs of 3D position-and-orientation sensors attached to a simulated kinematic chain. The second example uses real data from two electromagnetic sensors (Polhemus Liberty) which are attached to the forearm and hand, and measure 3D position and orientation. The third example uses real data from 14 optical sensors (Vicon) which are attached to the hand and fingers and only measure 3D position.

A. Synthetic movement

The simulated body (**Fig 4a**) has 3 links connected with 2 hinge joints with orthogonal axes. Simulated 3D position-and-orientation sensors are attached to links 1 and 3. The kinematic tree is shown in **Fig 4b**. The convention is the same as in **Fig 2**. There are 10 segments and 9 joints in the tree even though the body only has 3 physical links and 2 physical joints. The extra segments correspond to the sensors (denoted with thick outlines), the root (which is the stationary global frame), and to dummy segments needed to construct more complex joints from the primitive joint types defined above. As before, position and orientation measurements are modeled with separate virtual segments. Note that the sensor data contains no information about the lengths of links 1 and 3 and so these lengths are not estimated (otherwise the model would be unobservable). Instead we estimate the distances from the sensors to the axes of the hinge joints, assuming the sensors are attached on the long (x) axis of the corresponding links. We use a T and an R joint to connect the root to the rest of the tree. This yields a universal joint allowing the body to have any global position and orientation. There are 4 time-varying joints (empty circles) and 5 fixed joints (gray circles) with a total of 8 and 9 degrees of freedom respectively. Thus the state vector is 17-dimensional.

Synthetic sensor data were generated as follows. The time-varying degrees of freedom were set to low-pass-filtered white noise sequences with smoothness similar to human movement. The constant degrees of freedom were fixed to prespecified

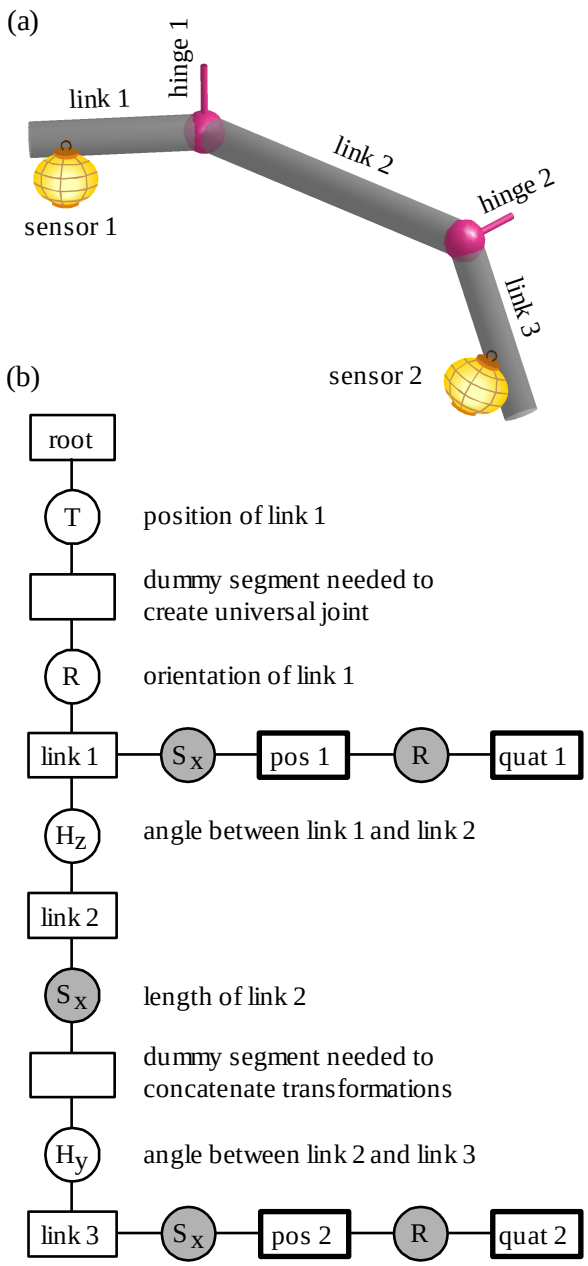

Fig. 4. Illustration of the model used in the synthetic example. **(a)**– The kinematic mechanism. **(b)**– The corresponding kinematic tree. Same convention as in **Fig 2**.

values. The expected values of the sensor measurements given the state variables at each point in time were computed using the generative model. White Gaussian noise was then added, with standard deviation similar to the estimates obtained from real data in the next example (2mm position and 2deg orientation). Quaternion data was renormalized to unit length. Fifty sequences of sensor measurements were simulated, each lasting 20 seconds at a sampling rate of 20Hz, and fed into the estimation method. The initial estimates of the constant parameters were sampled from Gaussians centered around the correct values. The initial estimates of the time-varying parameters were set to 0. The sensor noise covariance V was set to the correct value and was not estimated.

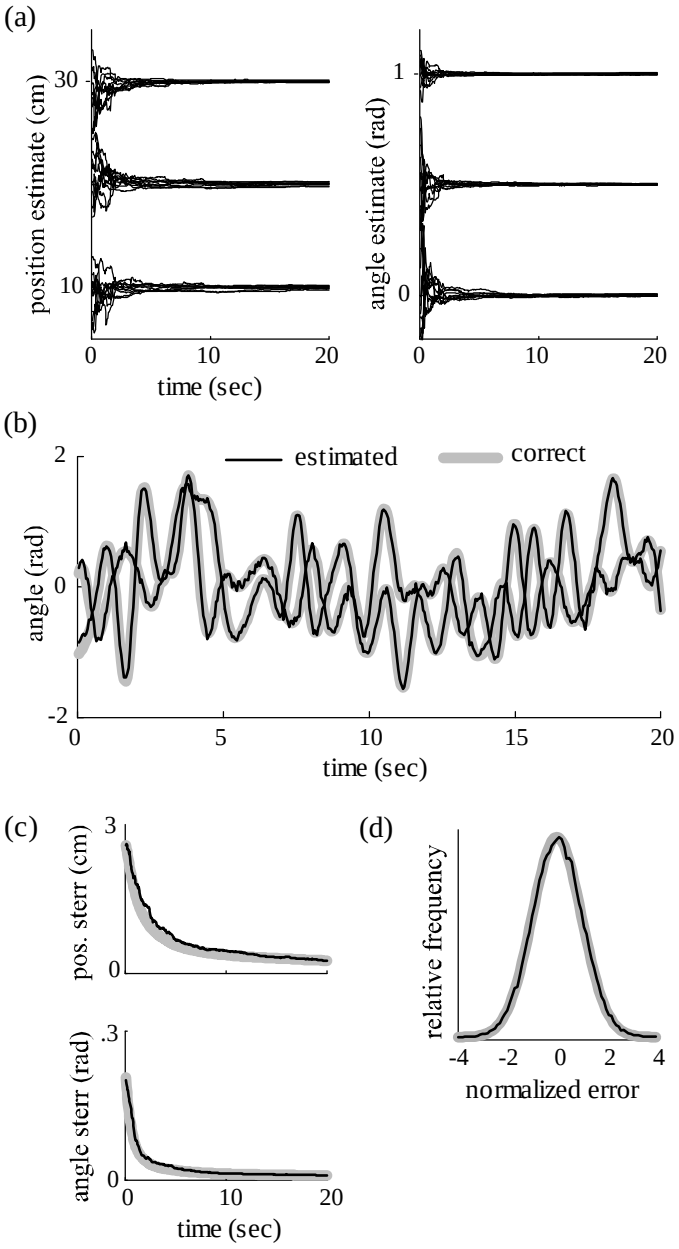

Fig. 5. Results on synthetic data. **(a)–** The estimates of the constant parameters converge to the correct values on multiple runs. **(b)–** The estimated joint angles are close to the correct angles. **(c)–** The standard errors computed by the method, for the constant parameters, agree with the actual estimation errors. **(d)–** The same holds for the time-varying parameters.

Fig 5a (left) shows how the estimates of the three constant position parameters (gray S joints) evolved over time. Despite the initial discrepancy, the estimates converged rapidly to the correct values. Similar behavior was seen in the estimates of the angle parameters. The three components of the constant R joint corresponding to sensor 2 are shown in **Fig 5a** (right).

In assessing the accuracy of the time-varying parameter estimates as well as the magnitude of the residuals, it is important to keep in mind that our method has the freedom to make all residuals zero – by rapidly adjusting the constant parameters. The latter is of course discouraged by the dynamics model prior, and indeed we see in **Fig 5a** that the estimates of the constant parameters do not change once they converge. Nevertheless this is a serious concern, because if the method gets trapped in a local minimum it may choose to violate the prior in order to explain the sensor data. Thus a conservative assessment would use fixed estimates of the constant parameters. Here and in the remaining examples, we run the method once to obtain the fixed estimates and then run it again to assess performance on time-varying quantities. This is how **Fig 5b, 6b, 7a** are generated. In **Fig 5b** we see accurate estimation of the two hinge joint angles.

Next we assess the accuracy of the confidence intervals or standard errors, defined as the square roots of the diagonal terms in the posterior covariance matrix. The thin lines in **Fig 5c** show the standard errors computed by our method for the constant parameters. Results are averaged over all positional (top) and angular (bottom) parameters. The thick lines show the correct standard errors. They are obtained by subtracting the correct from the estimated values of the constant parameters at each point in time, and computing the standard deviation over the 50 repetitions. The close correspondence seen in the figure would be guaranteed for a linear-Gaussian system, but here the nonlinear relation between latent variables and sensor measurements makes it difficult to derive any theoretical guarantees. Yet the standard errors computed by our method turn out to be accurate, at least in this example.

Similar accuracy is demonstrated in **Fig 5d** (albeit in a different format) for the estimates of the time-varying quantities. Here we divided the actual estimation error at each time step by the standard error computed by our method, and plotted a histogram of the resulting values (thin line). If the calculation of standard errors was correct, this histogram should be a Gaussian with mean 0 and variance 1. Such a Gaussian is plotted with a thick line and indeed matches the histogram.

The results shown in **Fig 5** exclude 8% outlier trials on which the method diverged. Divergence is easily diagnosed: both the standard errors and residuals increase over time. This occurs because of the local minima problem: at each time step we initialize the minimization process at the previous solution, and if that solution is poor the next solution also tends to be poor. The obvious way to alleviate the problem is to restart the minimization process multiple times when divergence is detected. We have not yet implemented this, partly because we did not encounter such problems on real data.

B. Wrist movement

Two Polhemus Liberty sensors were attached to the forearm and hand of one subject, on the dorsal side near the wrist. Each sensor measured 3D position and orientation (expressed as a quaternion) at 60Hz. The subject performed random arm and wrist movements. Five data sequences were recorded, each lasting 20 seconds. The structure of the kinematic model here is similar to the synthetic example, except that we now have two physical links (forearm and hand) connected with two hinge joints. The joint axes can be non-orthogonal and non-intersecting, therefore the distance and angle between them are estimated. The other estimated parameters are the distances of the sensors from the wrist and the sensor orientations relative

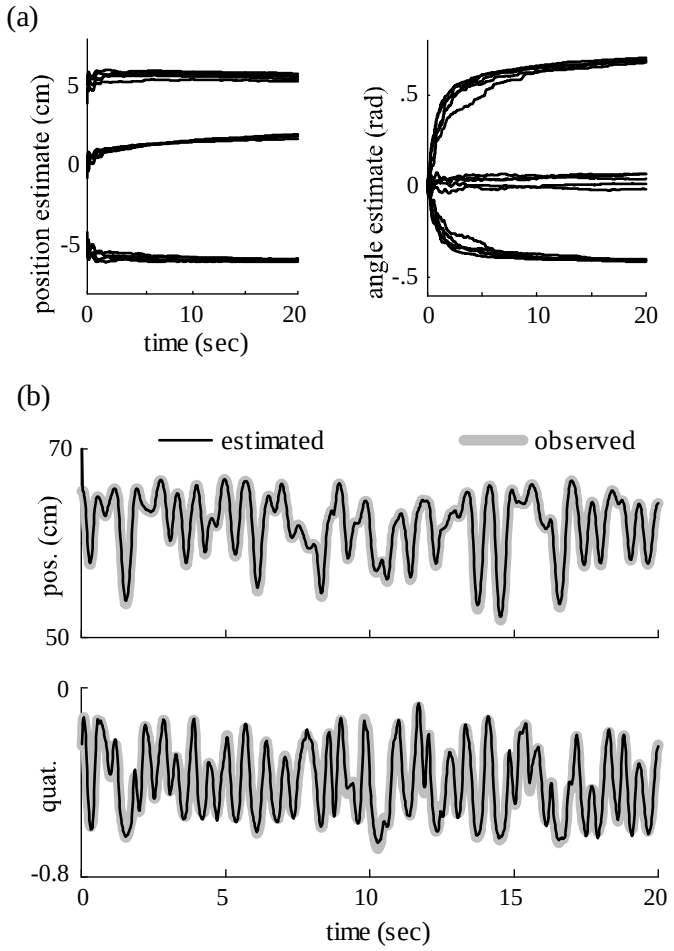

Fig. 6. Results on wrist movement data. **(a)–** Estimates of the constant parameters on 5 runs with different motion sequences. **(b)–** Comparison of estimated and observed sensor data; one position reading and one orientation reading are shown.

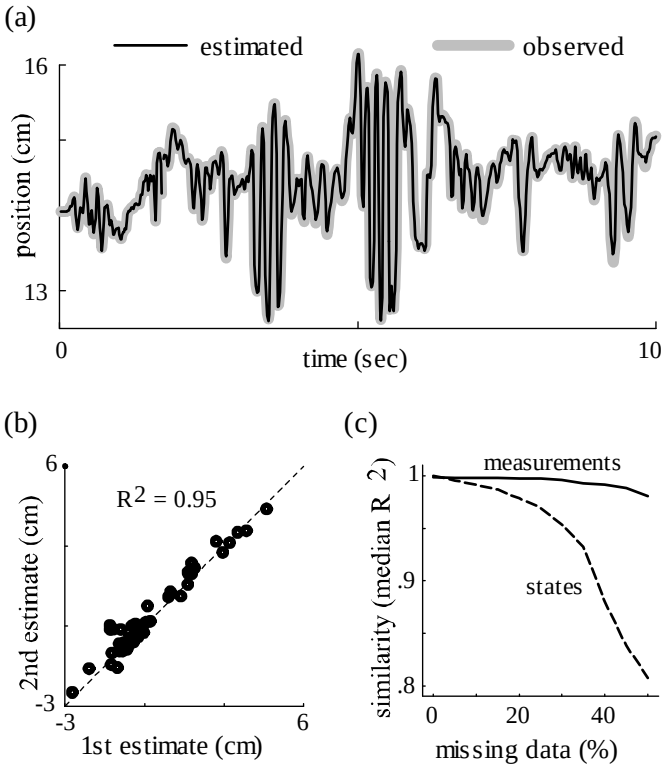

Fig. 7. Results from hand movement. **(a)–** Comparison of estimated and observed position of one sensor. **(b)–** Comparison of the estimates of constant parameters obtained from two runs on different movement sequences. **(c)–** Graceful degradation in the face of missing data.

to the limb segments. This model has 8 time-varying parameters and 10 constant parameters, yielding an 18-dimensional state vector.

The estimates of the position parameters from all five runs are shown in **Fig 6a** (left). They converge to the same values even though the motion sequences are very different. The middle curve corresponds to the distance between the two axes of wrist rotation. Note that the estimate converges to a positive value, inducating non-intersecting axes. **Fig 6a** (right) shows convergences for 3 of the 7 angle parameters.

In a real application there is no way to independently measure the correct joint angles (which is why synthetic examples are very useful in assessing the accuracy of such estimation methods). However one can compare the observed sensor data to the predicted sensor data. The difference between the two is the residual which we are trying to minimize. **Fig 6b** shows one position component and one orientation component of the data generated by the hand sensor on a typical run. Note the excellent agreement. As in the previous example, the results in **Fig 6b** are computed by first obtaining asymptotic estimates of the constant parameters and then repeating the procedure while keeping those estimates fixed.

C. Hand movement

Fourteen Vicon markers were attached to the dorsal surface of the hand and fingers of one subject. The markers were distributed as follows: 4 on the back of the hand over the index and middle finger metacarpals, 2 on each of the proximal and medial phalanges of the index and middle fingers, and 1 on each of the distal phalanges. The kinematic model had 7 physical segments including the hand and the 3 phalanges of the index and middle fingers. There were 8 hinge joints at the fingers and 1 universal joint "connecting" the free-floating hand to the fixed global frame. The kinematic tree had 30 segments. The state vector was 61-dimensional and incorporated 14 time-varying parameters and 47 constant parameters. The latter corresponded to the lengths of the proximal and medial phalanges, the distance between the proximal joints of the index and middle fingers, and the 3D positions of the 14 markers relative to the underlying limb segments. The motion capture data consisted of two sequences of random hand and finger movements, sampled at 66Hz and lasting 1 minute each.

Despite the increased model complexity, estimation accuracy was as high as in the previous examples. **Fig 7a** compares observed and estimated data from the position sensor attached to the distal phalange of the index finger. Note the almost perfect agreement. As before, the constant parameters are fixed. **Fig 7b** demonstrates that the estimates of the constant parameters converged to similar values on the two runs of the algorithm, even though the two movement sequences were

unrelated. Each circle in the figure corresponds to one of the 47 parameters; the horizontal axis is the estimate on the first run, the vertical axis is the estimate on the second run.

Finally we assessed the impact of missing data, by running the algorithm on 10 additional datasets obtained from the original by randomly removing a certain amount of data (5, 10, ... 50 percent). The data from each sensor at each point in time had equal probability of being removed. For each new dataset we found the asymptotic estimates of the constant parameters, fixed them, and repeated the estimation process to obtain the time-varying parameters. The latter were used to compute the results in **Fig 7c**. The dashed line shows the similarity between the state estimates obtained from the new datasets and those obtained from the original dataset. To measure similarity we took the pair of time series corresponding to each parameter, found the R^2 between the two, and then plotted the median of the R^2 values over the 14 time-varying parameters. The same procedure was repeated for the observed and estimated sensor data (42 pairs of time series in each dataset), and included the data which were removed when forming the new datasets. This is possible because our method yields estimates for all state variables regardless of how much data are missing, and these estimates can be used to predict the missing data. The result is shown with the solid line in **Fig 7c**.

V. Discussion

This paper described a probabilistic inference method for computing movement trajectories, skeletal parameters and marker attachments from motion capture data. Accurate results were demonstrated on both synthetic and real data. Direct comparison to alternative methods is difficult due to the lack of publicly-available implementations and benchmark datasets. However, because of the multiple sources of noise inherent in motion capture applications, we expect our method to outperform alternatives which ignore uncertainty.

Although we focused on position and orientations sensors and goniometers, other modalities can be incorporated in a similar fashion. Of particular interest are gyroscopes and accelerometers – which are becoming more compact and less expensive due to advances in MEMS technology. Such an extension requires the more elaborate dynamics model (6), as well as differentiation of the segments' velocities and accelerations with respect to the joint parameters. The latter may require a finite difference approximation.

Another extension worth considering is automatic discovery of correlations between sensor residuals and joint angles. Such correlations can arise due to soft-tissue deformation. For example, consider a sensor attached to the forearm. The rotation of the skin surface is significantly smaller than the rotation of the humerus, which results in correlation between the humeral rotation and the forearm sensor residual. These effects can be captured by augmenting the definition of sensor residuals with terms linear in the relevant joint angles. The coefficients of the extra terms would be inferred in the same way as the other constant parameters. Note that such correlations become explicit only if the residuals are expressed in the coordinate frame of the underlying limb segment. Currently all residuals are expressed in the global coordinate frame, however it is not difficult to transform them into local coordinates.

A reader familiar with the robotics literature may have noticed the similarity between our kinematic modeling approach and the Denavit-Hartenberg convention. Indeed our convention was chosen not because it is the most intuitive one, but because it facilitates the construction of minimal kinematic models. Minimal models are more likely to be observable (although observability should always be checked). The drawback of this approach is the non-intuitive nature of the kinematic tree. To aid the user, we need a high-level description language as well as a tool for automatic generation of kinematic trees.

All these extensions are topics for future work. As we continue to make progress, the software implementation available online will be updated.

VI. Literature cited

[1] Badler, N., Hollick, M. and Graneri, J. (1993) Real-time control of a virtual human using minimal sensors. Presence, 2: 82-86

[2] Capozzo, A., Catani, F., Leardini, A., Benedetti, M. and Della Croce, U. (1996) Position and orientation in space of bones during movement: Experimental Artefacts. Clinical Biomechanics, 11: 90-100

[3] Foxlin, E. (2002) Motion tracking requirements and technologies. In: Handbook of virtual environments (Stanney K, ed), pp 163-210. Mahwah, NJ: Lawrence Elbraum Associates

[4] Herna, L., Fua, P., Plaenkers, R., Boulic, R. and Thalmann, D. (2001) Using skeleton-based tracking to increase the reliability of optical motion capture. Human Movement Science, 20: 313-341

[5] Lee, J. and Ha, I. (2001) Real-time motion capture for a human body using accelerometers. Robotica 19: 601-610

[6] Meyer, K., Apllewhite, H. and Biocca, F. (1992) A survey of position trackers. Presence, 1: 173-200

[7] Molet, T., Boulic, R. and Thalmann, D. (1999) Human motion capture driven by orientation measurements. Presence, 8: 187-203

[8] O'Brien, J., Bodenheimer, B., Brostow, G. and Hodgins, J. (2000) Automatic joint parameter estimation from magnetic motion capture data. Proceeding of Graphics Interface 2000: 53-60

[9] Park, C., Jeong, I., Kim, H. and Wohn, K. (2000) Sensor fusion for motion capture system based on system identification. Proceedings of IEEE Computer Animation 2000: 71-76

[10] Ude, A. (1998) Nonlinear least squares optimization of unit quaternion functions for pose estimation from corresponding features. Proceedings of the International Conference on Pattern Recognition, 14: 425-427

[11] Cox, H. (1964) On the estimation of state variables and parameters for noisy dynamic systems. IEEE Trans Automatic Control, 9: 5-12

[12] Wan, E. and Nelson, A. (2001) Dual extended Kalman filter methods. In Kalman Filtering and Neural Networks, Haykin, S. (ed), pp 123-174, John Wiley and Sons, New York

[13] Pan, X., Todorov, E. and Li, W. (2005) Towards an integrated system for estimating multi-joint movement from diverse sensor data. Proceedings of the IEEE Engineering in Medicine and Biology, 27: 4982-4985

[14] Flash, T. and Hogan, N. (1985) The coordination of arm movements: An experimentally confirmed mathematical model. J Neuroscience, 5: 1688-1703

[15] Todorov, E. and Jordan, M. (1998) Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. J Neurophysiology, 80: 696-714

[16] Dempster, A., Laird, N. and Rubin, D. (1977) Maximum likelihood from incomplete data via the EM algorithm. J Royal Stat Soc B, 39: 1-38

[17] Roweis, S. and Ghahramani, Z. (1999) A unifying review of linear-Gaussian models. Neural Comp, 11: 305-345

[18] Sebastian Grassia, F. (1998) Practical parameterization of rotations using the exponential map. Journal of Graphics Tools, 3: 29-48