International Journal of Advanced Robotic Systems

OPEN ACCESS
ARTICLE

Estimation of Cartesian Space Robot Trajectories Using Unit Quaternion Space

Regular Paper

Aleš Ude[1,*]

1 Jožef Stefan Institute, Department of Automatics, Biocybernetics and Robotics, Ljubljana, Slovenia
* Corresponding author E-mail: ales.ude@ijs.si

Received 25 Jun 2013; Accepted 16 Jun 2014

DOI: 10.5772/58871

Abstract The ability to estimate Cartesian space trajectories that include orientation is of great importance for many practical applications. While it is becoming easier to acquire trajectory data by computer vision methods, data measured by general-purpose vision or depth sensors are often rather noisy. Appropriate smoothing methods are thus needed in order to reconstruct smooth Cartesian space trajectories given noisy measurements. In this paper, we propose an optimality criterion for the problem of the smooth estimation of Cartesian space trajectories that include the end-effector orientation. Based on this criterion, we develop an optimization method for trajectory estimation which takes into account the special properties of the orientation space, which we represent by unit quaternions. The efficiency of the developed approach is discussed and experimental results are presented.

Keywords Unit Quaternions, Nonlinear Optimization, Robot Programming by Demonstration

1. Introduction

The estimation of human hand motion is an important problem for many applications. Our interest stems from programming by demonstration [1] (also called 'imitation learning') in robotics. Figure 1 shows an example of a programming by demonstration system where a robot is guided to perform a classic peg-in-hole task [2]. The goal of imitation learning is to provide robotic systems with the ability to relate perceived human actions to their own embodiment in order to learn - and later perform - the demonstrated actions [3]. In imitation learning, knowledge about the demonstrated human hand motion is often essential for the understanding of the demonstrated behaviour.

Ignoring the fingers, we normally encode hand motion as a rigid body motion. It is well known that rigid body motion consists of a translational and a rotational part [4]. The reconstruction of a pure translational motion can be accomplished by standard optimization methods, because the set of all translations forms a three-dimensional (3D) vector space. On the other hand, the set of all rotations in the 3D Cartesian space, which we denote by $SO(3)$, forms only a group and not a vector space. The special Euclidean group $SE(3)$ is defined as a semi-direct product of $\mathbb{R}^3$ and $SO(3)$. It represents the Euclidean transformation of rotation followed by translation. Unfortunately, there exists no representational scheme for rotations that would be simultaneously non-redundant, continuous and free of singularities [4]. This causes problems when solving optimization problems for $SO(3)$ because representations free of singularities (e.g., rotation matrices and quaternions) contain more than the minimal number of parameters. The resulting parameters are therefore not independent of each other.

One possibility for measuring human arm and hand motion is to use RGB-D sensors, e.g., Kinect. The Kinect sensor uses the principle of structured light and captures depth and colour images simultaneously at a frame rate of about 30 Hz [5]. Together with the appropriate software, Kinect enables the tracking of several joints on the human body, including hand position and orientation [6, 7]. No special markers are needed. This results in a sequence of noisy measurements of the form:

$$(t_k, \boldsymbol{p}_k^*, \boldsymbol{q}_k^*, \boldsymbol{\Sigma}_k),\ k = 1, \ldots, n, \tag{1}$$

where t_k are the measurement times, $\boldsymbol{p}_k^* \in \mathbb{R}^3$ the measured positions, $\boldsymbol{q}_k^* \in \mathrm{S}^3$ the measured orientations represented by unit quaternions, and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{6\times 6}$ the covariance matrices describing the uncertainties in the measured position and orientations (see Section 3). Alternatively, such data can be obtained from marker-based systems, e.g., Optotrak. While the unit quaternion space S^3 does not uniquely represent SO(3) (i.e., for every orientation there are two equivalent unit quaternions), this duality does not represent a practical problem because the two unit quaternions representing the same orientation are well separated and there are no singularities in this representation. Our goal is to find a sequence of positions and orientations $(\boldsymbol{p}_k, \boldsymbol{q}_k)$ that approximate measurements (1) well and at the same time encode a smooth rigid body motion. Note that, in this paper, we consider the problem of batch (offline) processing, i.e., all of the data are available at the time of estimation. This is different from online filtering, which can only use past measurements to smooth the incoming data. Online filters on SE(3) and SO(3) were considered, e.g., in [8–12]. Methods that take into account the properties of the special Euclidean group have also been considered in the context of pose estimation in computer vision [13–16] and in control [17].

The problem of smoothing on general Riemannian manifolds has been considered in general mathematical texts [18–20]. The approach proposed in this paper is a special case of smoothing on Riemannian manifolds. Unlike these more general papers, in this paper we focus on a specific problem of smoothing in $\mathbb{R}^3 \times \mathrm{S}^3$ and address many particular issues relevant to this problem, e.g., the definition of tangential space and metrics using an exponential and logarithmic map and how to use them within the framework of the Gauss-Newton and Levenberg-Marquardt methods on $\mathbb{R}^3 \times \mathrm{S}^3$.

As mentioned above, SO(3) cannot be globally embedded in the 3D Euclidean space. This means that if the rotation group is represented by three real parameters (e.g., as in the case of Euler angles), the Euclidean metric topology in $\mathbb{R}^3$ does not induce a global topology or metric structure in SO(3). This is the main motivation for selecting unit quaternions to represent rotations - the spherical metric of S^3 corresponds to the angular metric of SO(3) [21]. We would, however, obtain similar results if we used a different representation free of singularities, e.g., rotation matrices. In the following, we first formulate the problem of estimating motion trajectories on $\mathbb{R}^3 \times \mathrm{S}^3$ and then propose an optimization method that can be applied to solve it. The main feature of our approach is that we exploit the properties of the exponential map to calculate new estimates at each step of the iterative optimization process, which enables us to formulate the optimization process directly on $\mathbb{R}^3 \times \mathrm{S}^3$.

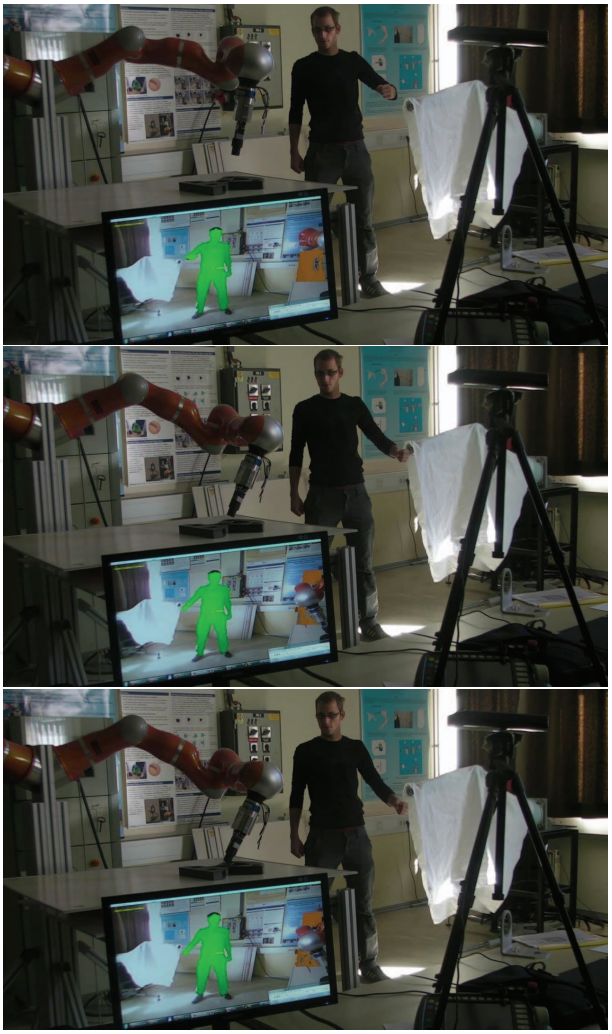

Figure 1. Demonstration of a peg-in-hole operation. The human demonstrator observes the performance of the robot and adapts his hand motion so that the robot successfully executes the task. The demonstrator's hand motion is measured by Kinect.

2. Preliminaries

Formally, a quaternion $\boldsymbol{q} = (w, u_1, u_2, u_3)$ is a vector quantity, where w is the scalar component of $\boldsymbol{q}$ and $\boldsymbol{u} = [u_1, u_2, u_3]^T$ is the vector component. The quaternion multiplication is defined by:

$$\boldsymbol{q} * \boldsymbol{q}' = (ww' - \boldsymbol{u}^T\boldsymbol{u}', w\boldsymbol{u}' + w'\boldsymbol{u} + \boldsymbol{u} \times \boldsymbol{u}'). \tag{2}$$

Quaternions form a non-commutative group with respect to the above multiplication. The magnitude of a quaternion is defined as:

$$|\boldsymbol{q}| = \sqrt{\boldsymbol{q} * \overline{\boldsymbol{q}}} = \sqrt{w^2 + \boldsymbol{u}^T\boldsymbol{u}}, \tag{3}$$

$$\overline{\boldsymbol{q}} = (w, -\boldsymbol{u}), \tag{4}$$

where $\overline{\boldsymbol{q}}$ is the conjugate of $\boldsymbol{q}$. Given a rotation by ϑ about a unit axis vector $\boldsymbol{n}$, we define the associated unit quaternion as:

$$\boldsymbol{q}(\vartheta, \boldsymbol{n}) = \left(\cos\left(\frac{\vartheta}{2}\right), \sin\left(\frac{\vartheta}{2}\right)\boldsymbol{n}\right). \tag{5}$$

There is a 2-1 mapping between unit quaternions and the rotation group [21]. Each rotation from SO(3) can thus be represented by two quaternions belonging to the unit sphere $\mathrm{S}^3 \subset \mathbb{R}^4$. However, the two unit quaternions representing the same rotation are well separated because they lie on different sides of the unit sphere. It can be shown that a vector $\boldsymbol{v}' \in \mathbb{R}^3$, rotated from a vector $\boldsymbol{v} \in \mathbb{R}^3$ by a rotation represented by a unit quaternion $\boldsymbol{q}$, can be calculated by a simple quaternion multiplication:

$$\boldsymbol{v}' = \boldsymbol{q} * \boldsymbol{v} * \overline{\boldsymbol{q}}. \tag{6}$$

In this multiplication, the 3D vector $\boldsymbol{v}$ is treated as a quaternion with a zero scalar component. It is easy to see that the resulting quaternion $\boldsymbol{v}'$ has a zero scalar component as well.

In the following, we will need the exponential map $\exp : \mathbb{R}^3 \to \mathrm{S}^3$, which is given by:

$$\exp(\boldsymbol{r}) = \begin{cases} \left(\cos(\|\boldsymbol{r}\|), \sin(\|\boldsymbol{r}\|)\dfrac{\boldsymbol{r}}{\|\boldsymbol{r}\|}\right), & \|\boldsymbol{r}\| \neq 0 \\ (1,0,0,0), & \boldsymbol{r} = 0 \end{cases}. \tag{7}$$

We denote by $\mathrm{T}_q(\mathrm{S}^3) \subset \mathbb{R}^4$ the tangent space of S^3 at unit quaternion $\boldsymbol{q}$. It can be shown that the exponential map transforms a tangent vector $\boldsymbol{r} \in \mathrm{T}_1(\mathrm{S}^3) \equiv \mathbb{R}^3$ into $\boldsymbol{q} \in \mathrm{S}^3$, where $\boldsymbol{q}$ is a quaternion at distance $\|\boldsymbol{r}\|$ from the identity quaternion 1 (a unit quaternion with a zero vector component) along the geodesic curve, which is given by $\boldsymbol{q}(t) = \exp(t\log(\boldsymbol{q}))$, starting from quaternion 1 in the direction of $\boldsymbol{r}$. The geodesic curve represents the shortest path from 1 to $\boldsymbol{q}$ on S^3. The logarithmic map $\log : \mathrm{S}^3 \to \mathbb{R}^3$ is defined as:

$$\log(\boldsymbol{q}) = \log(w, \boldsymbol{u}) = \begin{cases} \arccos(w)\dfrac{\boldsymbol{u}}{\|\boldsymbol{u}\|}, & \boldsymbol{u} \neq 0 \\ (0,0,0), & \boldsymbol{q} = (1,0,0,0) \end{cases}. \tag{8}$$

If we limit the domain of the exponential map to $\|\boldsymbol{r}\| < \pi$ and the domain of the logarithmic map to $\mathrm{S}^3/(-1,0,0,0)$, then both mappings become one-to-one, continuously differentiable and inverse to each other. It can be shown that the expression:

$$\mathrm{d}(\boldsymbol{q_1}, \boldsymbol{q_2}) = \begin{cases} 2\|\log(\boldsymbol{q}_1 * \overline{\boldsymbol{q}_2})\|, & \boldsymbol{q}_1 * \overline{\boldsymbol{q}_2} \neq (-1,0,0,0) \\ 2\pi, & \text{otherwise} \end{cases} \tag{9}$$

is a metric on S^3. This metric is usually called the 'angular metric' of S^3.

3. Formulation of the Problem

The estimation of noisy vector-valued measurements with non-diagonal covariance matrices has been considered by [22, 23], who developed an iterative algorithm for the nonlinear estimation of a smooth vector-valued function based on a non-parametric optimality criterion. In our case, the problem is more complicated because the space of all orientations is not a vector space. While the difference between the measured and the true position can be modelled as additive, namely:

$$\boldsymbol{p}_k = \boldsymbol{p}'_k + \boldsymbol{e}^p_k, \tag{10}$$

this is not the case for the difference between the measured orientation $\boldsymbol{q}_k$ and the true orientation $\boldsymbol{q}'_k$. This error can be modelled as [8]:

$$\boldsymbol{q}_k = \exp(\boldsymbol{e}^q_k) * \boldsymbol{q}'_k, \tag{11}$$

where $\boldsymbol{e}^q_k \in \mathbb{R}^3$ is an error vector.

We assume that the error in the position and orientation is Gaussian with zero mean and a covariance matrix $\boldsymbol{\Sigma}_k$. The changing of the current position $\boldsymbol{p}'_k$ and orientation $\boldsymbol{q}'_k$ by a deterministic displacement $(\Delta\boldsymbol{p}_k, \Delta\boldsymbol{q}_k)$ results in:

$$(\boldsymbol{p}''_k, \boldsymbol{q}''_k) = (\boldsymbol{p}'_k + \Delta\boldsymbol{p}_k, \Delta\boldsymbol{q}_k * \boldsymbol{q}'_k). \tag{12}$$

We note that the position error vector remains unchanged under transformation (12). To find the relationship between the old and the transformed rotation error vector, we make the following observation:

$$\begin{aligned} \boldsymbol{q}^{\text{new}}_k &= \Delta\boldsymbol{q}_k * \boldsymbol{q}_k = \Delta\boldsymbol{q}_k * \exp(\boldsymbol{e}^q_k) * \boldsymbol{q}'_k \\ &= \Delta\boldsymbol{q}_k * \exp(\boldsymbol{e}^q_k) * \overline{\Delta\boldsymbol{q}_k} * \Delta\boldsymbol{q}_k * \boldsymbol{q}'_k \\ &= \exp\left(\Delta\boldsymbol{q}_k * \boldsymbol{e}^q_k * \overline{\Delta\boldsymbol{q}_k}\right) * \boldsymbol{q}''_k. \end{aligned} \tag{13}$$

Hence, there exists the following relationship between the two error vectors:

$$\boldsymbol{e}^{q\ \text{new}}_k = \Delta\boldsymbol{q}_k * \boldsymbol{e}^q_k * \overline{\Delta\boldsymbol{q}_k} = \boldsymbol{R}(\Delta\boldsymbol{q}_k)\boldsymbol{e}^q_k, \tag{14}$$

where the rotation matrix $\boldsymbol{R}$ is given by the formula:

$$\boldsymbol{R}(\boldsymbol{q}) = \boldsymbol{R}(w, \boldsymbol{u}) = \begin{bmatrix} w^2 + u_1^2 - u_2^2 - u_3^2 & 2(u_1u_2 - wu_3) & 2(u_1u_3 + wu_2) \\ 2(u_1u_2 + wu_3) & w^2 - u_1^2 + u_2^2 - u_3^2 & 2(u_2u_3 - wu_1) \\ 2(u_1u_3 - wu_2) & 2(u_2u_3 + wu_1) & w^2 - u_1^2 - u_2^2 + u_3^2 \end{bmatrix}.$$

The new error vector is obtained by rotating the old error vector into a new orientation. Writing the covariance matrix $\boldsymbol{\Sigma}_k$ as:

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}^p_k & \boldsymbol{\Sigma}^{pq}_k \\ \boldsymbol{\Sigma}^{pq}_k & \boldsymbol{\Sigma}^q_k \end{bmatrix}, \tag{15}$$

the covariance matrix describing the uncertainties in the new pose can be calculated by:

$$\boldsymbol{\Sigma}^{\text{new}}_k = \begin{bmatrix} \boldsymbol{\Sigma}^p_k & \boldsymbol{\Sigma}^{pq}_k \boldsymbol{R}(\Delta\boldsymbol{q}_k)^T \\ \boldsymbol{R}(\Delta\boldsymbol{q}_k)(\boldsymbol{\Sigma}^{pq}_k)^T & \boldsymbol{R}(\Delta\boldsymbol{q}_k)\boldsymbol{\Sigma}^q_k\boldsymbol{R}(\Delta\boldsymbol{q}_k)^T \end{bmatrix}. \tag{16}$$

The aim of reconstruction is to find a trajectory that not only approximates the measurements well but is also smooth. If the measurements were simply interpolated, the reconstructed trajectory would not be smooth enough. Hence, we must search for a compromise between smoothness and goodness of fit. Writing $\boldsymbol{p} = \left[\boldsymbol{p}_1^T, \ldots, \boldsymbol{p}_n^T\right]^T$ and $\boldsymbol{q} = \left[\boldsymbol{q}_1^T, \ldots, \boldsymbol{q}_n^T\right]^T$, the goodness of fit can be evaluated by:

$$\mathrm{F}_0(\boldsymbol{p}, \boldsymbol{q}) = \sum_{k=1}^{n} \begin{bmatrix} \boldsymbol{p}_k - \boldsymbol{p}^*_k \\ \log(\boldsymbol{q}_k * \overline{\boldsymbol{q}^*_k}) \end{bmatrix}^T \boldsymbol{\Sigma}_k^{-1} \begin{bmatrix} \boldsymbol{p}_k - \boldsymbol{p}^*_k \\ \log(\boldsymbol{q}_k * \overline{\boldsymbol{q}^*_k}) \end{bmatrix}, \tag{17}$$

where $(\boldsymbol{p}_k^*, \boldsymbol{q}_k^*)$, $k = 1, \ldots, n$, are the measured poses and $\boldsymbol{\Sigma}_k$ are the positive definite covariance matrices with respect to the zero mean error vector $[\boldsymbol{e}_k^{p\,T}, \boldsymbol{e}_k^{q\,T}]^T$ defined in (10) and (11). It is easy to show that each of the summed terms in (17) is a metric on $\mathbb{R}^3 \times \mathrm{S}^3$ if $\boldsymbol{\Sigma}_k$ are positive definite matrices.

A good measure of the smoothness of trajectories is given by the amount of linear and angular acceleration. The linear acceleration $\boldsymbol{a}_k$, $k = 2, \ldots, n-1$, can be estimated by:

$$\begin{aligned} \boldsymbol{v}_k(\boldsymbol{p}) &= \frac{\boldsymbol{p}_{k+1} - \boldsymbol{p}_k}{\Delta t_k}, \\ \boldsymbol{a}_k(\boldsymbol{p}) &= \frac{\boldsymbol{v}_k(\boldsymbol{p}) - \boldsymbol{v}_{k-1}(\boldsymbol{p})}{\Delta t_{k-1}} \\ &= \frac{\boldsymbol{p}_{k+1} - \boldsymbol{p}_k}{\Delta t_{k-1}\Delta t_k} - \frac{\boldsymbol{p}_k - \boldsymbol{p}_{k-1}}{\Delta t_{k-1}\Delta t_{k-1}}, \end{aligned}$$

where $\Delta t_k = t_{k+1} - t_k$. Similarly, the angular acceleration $\boldsymbol{\alpha}_k$, $k = 2, \ldots, n-1$, can be estimated by:

$$\begin{aligned} \boldsymbol{\omega}_k(\boldsymbol{q}) &= \frac{2}{\Delta t_k}\log(\boldsymbol{q}_{k+1} * \overline{\boldsymbol{q}_k}) \\ \boldsymbol{\alpha}_k(\boldsymbol{q}) &= \frac{\boldsymbol{\omega}_k(\boldsymbol{q}) - \boldsymbol{\omega}_{k-1}(\boldsymbol{q})}{\Delta t_{k-1}} \\ &= \frac{2\log(\boldsymbol{q}_{k+1} * \overline{\boldsymbol{q}_k})}{\Delta t_{k-1}\Delta t_k} - \frac{2\log(\boldsymbol{q}_k * \overline{\boldsymbol{q}_{k-1}})}{\Delta t_{k-1}\Delta t_{k-1}} \end{aligned}$$

Writing:

$$\mathrm{g}_1(\boldsymbol{p}) = \sum_{k=2}^{n-1} \|\boldsymbol{a}_k(\boldsymbol{p})\|^2, \tag{18}$$

$$\mathrm{g}_2(\boldsymbol{q}) = \sum_{k=2}^{n-1} \|\boldsymbol{\alpha}_k(\boldsymbol{q})\|^2, \tag{19}$$

we can formulate the following criterion, which should be minimized by a rigid body motion that exhibits good balance between smoothness and goodness of fit:

$$\mathrm{F}(\boldsymbol{p}, \boldsymbol{q}) = \frac{1}{2}(\mathrm{F}_0(\boldsymbol{p}, \boldsymbol{q}) + \lambda_1 \mathrm{g}_1(\boldsymbol{p}) + \lambda_2 \mathrm{g}_2(\boldsymbol{q})). \tag{20}$$

The parameters λ_1 and λ_2 govern the trade-off between the two criteria.

Since the criterion function (20) is nonlinear, we must apply nonlinear optimization techniques to find the optimal sequence of poses $(\boldsymbol{p}_k, \boldsymbol{q}_k)$. The minimization of (20) over $\boldsymbol{p}_k$, $\boldsymbol{q}_k$ would be a classic nonlinear least squares optimization problem if we could treat unit quaternions $\boldsymbol{q}_k$ as elements of $\mathbb{R}^4$ and not of S^3. Since this is not the case, the classic approach would be to add the constraints $|\boldsymbol{q}_k| = 1$ to the optimization criterion. However, such constraints make the optimization problem significantly harder. In the following, we propose a technique that can be used to optimize the criterion (20) without specifying additional constraints.

4. Optimization in $\mathbb{R}^3 \times \mathrm{S}^3 \times \ldots \times \mathbb{R}^3 \times \mathrm{S}^3$

The tangent space $\mathrm{T}_q(\mathrm{S}^3) \subset \mathbb{R}^4$ is defined as a space that contains the directions of all paths on S^3 passing through the quaternion $\boldsymbol{q}$. As mentioned in Section 2, the exponential map exp transforms a tangent vector $\boldsymbol{r} \in \mathrm{T}_1(\mathrm{S}^3) \equiv \mathbb{R}^3$ into a point $\tilde{\boldsymbol{q}} \in \mathrm{S}^3$ that lies on the geodesic curve corresponding to the tangent vector $\boldsymbol{r}$. It turns out that for any $\boldsymbol{q} \in \mathrm{S}^3$, the exponential map $\exp_q$ that transforms each tangent vector $\boldsymbol{x} \in \mathrm{T}_q(\mathrm{S}^3)$ into a point $\tilde{\boldsymbol{q}}$ that lies on S^3 along the geodesic curve starting at $\boldsymbol{q}$ in the direction of $\boldsymbol{x}$ at the distance $\|\boldsymbol{x}\|$ is given by:

$$\exp_q(\boldsymbol{x}) = \exp(\boldsymbol{x} * \overline{\boldsymbol{q}}) * \boldsymbol{q}. \tag{21}$$

It can be shown [24] that $\boldsymbol{x} * \overline{\boldsymbol{q}} \in \mathrm{T}_1(\mathrm{S}^3)$ or - in other words - $\boldsymbol{x} * \overline{\boldsymbol{q}}$ is a quaternion with a zero scalar component, for any $\boldsymbol{x} \in \mathrm{T}_q(\mathrm{S}^3)$, $\boldsymbol{q} \in \mathrm{S}^3$. Thus, the above mapping is well-defined for all $\boldsymbol{x} \in \mathrm{T}_q(\mathrm{S}^3)$. As the mapping $\boldsymbol{x} * \overline{\boldsymbol{q}}$ is an isomorphism from $\mathrm{T}_q(\mathrm{S}^3)$ to $\mathbb{R}^3$, all the unit quaternions in the neighbourhood of $\boldsymbol{q}$ can be represented by $\exp(\boldsymbol{r}) * \boldsymbol{q}$, $\boldsymbol{r} \in \mathbb{R}^3$.

Taking $(\boldsymbol{p}_k^i, \boldsymbol{q}_k^i) \in \mathbb{R}^3 \times \mathrm{S}^3$, $k = 1, \ldots n$, to be the i-th estimate for the optimal sequence of positions and orientations, it is appropriate to calculate the next sequence of positions and orientations as follows:

$$\boldsymbol{p}_k^{i+1} = \boldsymbol{p}_k^i + \boldsymbol{d}_k^i, \tag{22}$$

$$\boldsymbol{q}_k^{i+1} = \exp(\boldsymbol{r}_k^i) * \boldsymbol{q}_k^i. \tag{23}$$

where $\boldsymbol{d}^i = \left[\boldsymbol{d}_1^{i\,T}, \ldots, \boldsymbol{d}_n^{i\,T}\right]^T$ and $\boldsymbol{r}^i = \left[\boldsymbol{r}_1^{i\,T}, \ldots, \boldsymbol{r}_n^{i\,T}\right]^T$ should be obtained by approximating the minimum of the objective function:

$$\mathrm{F}_i(\boldsymbol{d}, \boldsymbol{r}) = \frac{1}{2}\mathrm{F}(\boldsymbol{p}^i + \boldsymbol{d}, \exp(\boldsymbol{r}) * \boldsymbol{q}^i). \tag{24}$$

where:

$$\exp(\boldsymbol{r}) * \boldsymbol{q}^i = \begin{bmatrix} \exp(\boldsymbol{r}_1) * \boldsymbol{q}_1^i \\ \vdots \\ \exp(\boldsymbol{r}_n) * \boldsymbol{q}_n^i \end{bmatrix}.$$

The above criterion can be rewritten as:

$$\mathrm{F}_i(\boldsymbol{d}, \boldsymbol{r}) = \frac{1}{2}\boldsymbol{f}_i(\boldsymbol{d}, \boldsymbol{r})^T \boldsymbol{f}_i(\boldsymbol{d}, \boldsymbol{r}), \tag{25}$$

where:

$$\boldsymbol{f}_i(\boldsymbol{d}, \boldsymbol{r}) = \begin{bmatrix} \boldsymbol{\Sigma}_1^{-1/2} \begin{bmatrix} \boldsymbol{d}_1 + \boldsymbol{p}_1^i - \boldsymbol{p}_1^* \\ \log(\exp(\boldsymbol{r}_1) * \boldsymbol{q}_1^i * \overline{\boldsymbol{q}_1^*}) \end{bmatrix} \\ \sqrt{\lambda_1}\,\boldsymbol{a}_2(\boldsymbol{d} + \boldsymbol{p}^i) \\ \sqrt{\lambda_2}\,\boldsymbol{\alpha}_2(\exp(\boldsymbol{r}) * \boldsymbol{q}^i) \\ \boldsymbol{\Sigma}_2^{-1/2} \begin{bmatrix} \boldsymbol{d}_2 + \boldsymbol{p}_2^i - \boldsymbol{p}_2^* \\ \log(\exp(\boldsymbol{r}_2) * \boldsymbol{q}_2^i * \overline{\boldsymbol{q}_2^*}) \end{bmatrix} \\ \vdots \\ \sqrt{\lambda_1}\,\boldsymbol{a}_{n-1}(\boldsymbol{d} + \boldsymbol{p}^i) \\ \sqrt{\lambda_2}\,\boldsymbol{\alpha}_{n-1}(\exp(\boldsymbol{r}) * \boldsymbol{q}^i) \\ \boldsymbol{\Sigma}_{n-1}^{-1/2} \begin{bmatrix} \boldsymbol{d}_{n-1} + \boldsymbol{p}_{n-1}^i - \boldsymbol{p}_{n-1}^* \\ \log(\exp(\boldsymbol{r}_{n-1}) * \boldsymbol{q}_{n-1}^i * \overline{\boldsymbol{q}_{n-1}^*}) \end{bmatrix} \\ \boldsymbol{\Sigma}_n^{-1/2} \begin{bmatrix} \boldsymbol{d}_n + \boldsymbol{p}_n^i - \boldsymbol{p}_n^* \\ \log(\exp(\boldsymbol{r}_n) * \boldsymbol{q}_n^i * \overline{\boldsymbol{q}_n^*}) \end{bmatrix} \end{bmatrix}$$

is a vector function from $\mathbb{R}^{6n}$ to $\mathbb{R}^{12n-12}$. The gradient and the Hessian of F_i are given by:

$$\nabla \mathrm{F}_i(\boldsymbol{d},\boldsymbol{r}) = \boldsymbol{J}_i(\boldsymbol{d},\boldsymbol{r})^T \boldsymbol{f}_i(\boldsymbol{d},\boldsymbol{r}), \tag{26}$$

$$\nabla^2 \mathrm{F}_i(\boldsymbol{d},\boldsymbol{r}) = \boldsymbol{J}_i(\boldsymbol{d},\boldsymbol{r})^T \boldsymbol{J}_i(\boldsymbol{d},\boldsymbol{r}) + \sum_{k=1}^{n} f_i^k(\boldsymbol{d},\boldsymbol{r}) \nabla^2 f_i^k(\boldsymbol{d},\boldsymbol{r}), \tag{27}$$

where $\boldsymbol{J}_i(\boldsymbol{d},\boldsymbol{r})$ is the Jacobian of $\boldsymbol{f}_i$ at $(\boldsymbol{d},\boldsymbol{r})$ and f_i^k are the component functions of $\boldsymbol{f}_i$.

The Taylor series expansion for the vector function $\nabla \mathrm{F}_i$ around $\nabla \mathrm{F}_i(0,0)$ is given by:

$$\nabla \mathrm{F}_i(\boldsymbol{d},\boldsymbol{r}) \approx \nabla \mathrm{F}_i(0,0) + \nabla^2 \mathrm{F}_i(0,0) \left[\boldsymbol{d}^T, \boldsymbol{r}^T\right]^T \tag{28}$$

If we assume that the value of F_i is small for all $\boldsymbol{q}$ belonging to the neighbourhood of the solution (i.e., $f_i^k(\boldsymbol{d},\boldsymbol{r}) \approx 0$ for all k), we obtain from Eq. (27) the following approximation for the Hessian in the neighbourhood of the solution:

$$\nabla^2 \mathrm{F}_i(0,0) \approx \boldsymbol{J}_i^T \boldsymbol{J}_i, \tag{29}$$

where $\boldsymbol{J}_i \in \mathbb{R}^{(12n-12)\times(6n)}$ is the Jacobian of $\boldsymbol{f}_i$ at $\boldsymbol{d} = \boldsymbol{r} = 0$. Using the Taylor series expansion (28) and the fact that $\nabla \mathrm{F}_i(\boldsymbol{d},\boldsymbol{r}) = 0$ at the minimum of F_i, we can calculate the appropriate modification $(\boldsymbol{d}^i, \boldsymbol{r}^i)$ as follows:

$$\begin{bmatrix} \boldsymbol{d}^i \\ \boldsymbol{r}^i \end{bmatrix} = -(\boldsymbol{J}_i^T \boldsymbol{J}_i)^{-1} \boldsymbol{J}_i^T \boldsymbol{f}_i(0,0), \tag{30}$$

The next sequence of poses can then be calculated using Eq. (22) and (23). The iteration is stopped when:

$$\|\nabla \mathrm{F}_i(0,0)\| = \|\boldsymbol{J}_i^T \boldsymbol{f}_i(0,0)\| < \varepsilon. \tag{31}$$

Note that $\boldsymbol{J}_i^T \boldsymbol{J}_i \in \mathbb{R}^{6n\times 6n}$ is a symmetric band matrix with bandwidth 35 (17+1+17). Thus, the number of arithmetic operations needed to solve the resulting linear system of equations is linear with respect to the number of measurements. Note that this is by far the most computationally expensive part of our system. Since we have a good initial approximation for our optimization problem (the measurements themselves are used to initialize optimization), there are not too many iterations that need to be performed in order to find the optimal solution (see also Table 1). Hence, our approach can easily deal with thousands of measurements, which is the order of magnitude for the number of data points we normally acquire when measuring demonstrated trajectories. Demonstrated trajectories are usually acquired at 30 Hz, and up to 120 Hz with marker-based systems, and take from a few seconds to tens of seconds. Note also that batch optimization is by definition offline; hence, real-time operation is not an issue.

The counterpart of the derived iteration in real spaces is the Gauss-Newton iteration. Actually, we have shown above how to carry out the Gauss-Newton iteration on $\mathbb{R}^3 \times \mathrm{S}^3 \times \ldots \times \mathbb{R}^3 \times \mathrm{S}^3$. However, the classic Gauss-Newton method can encounter problems when the

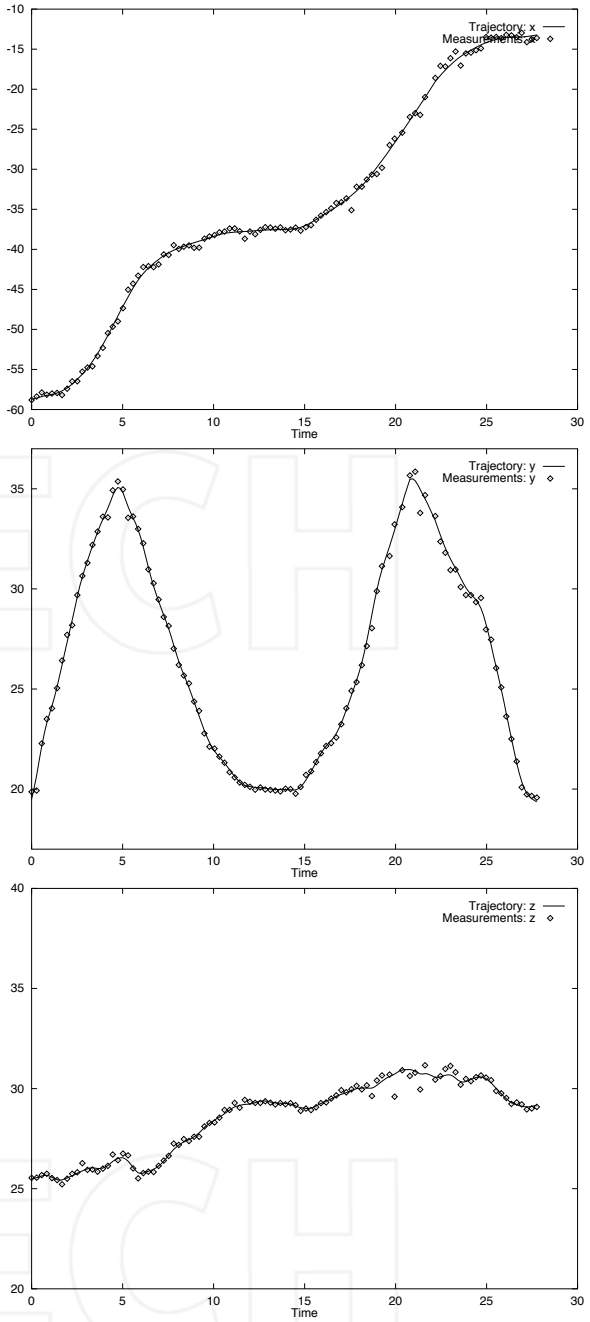

Figure 2. The translational part (x, y and z components) of the reconstructed trajectory (in centimetres) and a sample of measurements. Not all measurements are shown for better visualization.

second-order term in Eq. (27) is significant. While for small smoothing parameters λ_1, λ_2 the criterion functions f_i^k are also small in the neighbourhood of the minimum, this is not the case for larger smoothing parameters. Therefore, we can expect slower convergence when the smoothing parameter becomes large.

We can overcome this problem by applying the *Levenberg-Marquardt* method, in which the search direction is calculated as follows:

$$\begin{bmatrix} \boldsymbol{d}^i \\ \boldsymbol{r}^i \end{bmatrix} = -(\boldsymbol{J}_i^T \boldsymbol{J}_i + \mu_i \boldsymbol{I})^{-1} \boldsymbol{J}_i^T \boldsymbol{f}_i(0,0). \tag{32}$$

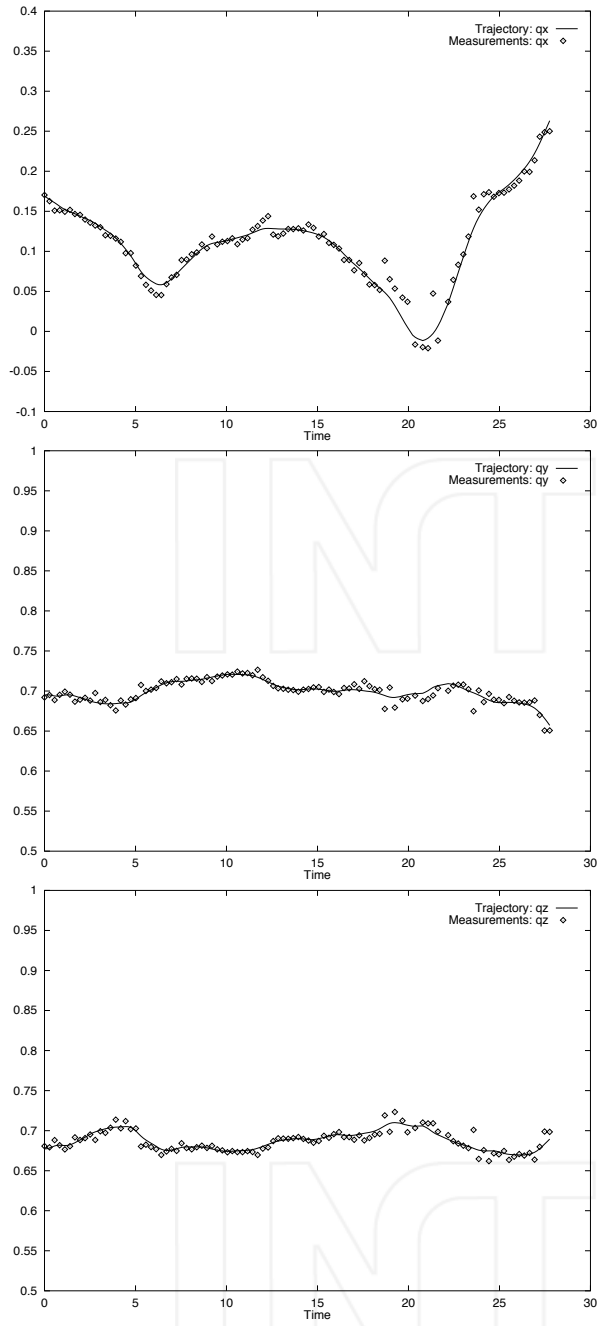

Figure 3. The orientational part (x, y and z components of $\boldsymbol{u}(t)$, $\boldsymbol{q}(t) = (w(t), \boldsymbol{u}(t))$) of the reconstructed trajectory and a sample of measurements. Not all measurements are shown for better visualization.

Note that the system matrix $(\boldsymbol{J}_i^T \boldsymbol{J}_i + \mu_i \boldsymbol{I})$ is positive definite for every $\mu_i > 0$. When μ_i is equal to zero, the search direction becomes identical to that of the Gauss-Newton method. As μ_i tends towards infinity, $(\boldsymbol{d}^i, \boldsymbol{r}^i)$ tends towards a vector of zeros and a steepest descent direction. This implies that, for some sufficiently large μ_i, the value $\mathrm{F}_i(\boldsymbol{d}^i, \boldsymbol{r}^i)$ is smaller than $\mathrm{F}_i(0,0) = \mathrm{F}(\boldsymbol{p}^i, \boldsymbol{q}^i)$. Thus, the Levenberg-Marquardt method uses a search direction that is a cross between the Gauss-Newton direction and the steepest descent.

It remains to show how to determine the smoothing parameters λ_1 and λ_2. It is important to properly select

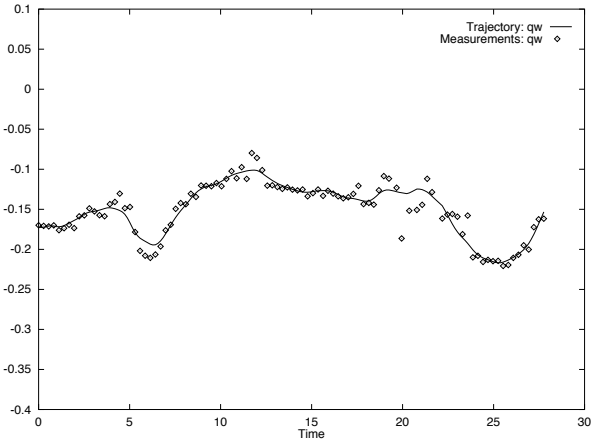

Figure 4. The orientational part (w component of $\boldsymbol{q}(t) = (w(t), \boldsymbol{u}(t))$) of the reconstructed trajectory and a sample of measurements. Not all measurements are shown for better visualization.

the degree of smoothing to find a proper balance between smoothness and goodness of fit. Often, methods like cross-validation are used, but in general cross-validation is computationally expensive because it requires that the data be partitioned into two sets: one used to learn or train a model, and the other used to validate the model. These sets need to be changed many times so that all the data can be validated. For this reason, we prefer to use an approach proposed in [25] for the case when the amount of noise associated with the data is known. With this method, we can determine the optimal values for λ_1 and λ_2 by solving the following nonlinear systems of equations:

$$\sum_{k=1}^{n} (\boldsymbol{p}_k(\lambda_1) - \boldsymbol{p}_k^*)^T (\boldsymbol{\Sigma}_k^p)^{-1} (\boldsymbol{p}_k(\lambda_1) - \boldsymbol{p}_k^*) = \mathrm{S}_1, \quad (33)$$

$$\sum_{k=1}^{n} \log(\boldsymbol{q}_k(\lambda_2) * \overline{\boldsymbol{q}_k^*})^T (\boldsymbol{\Sigma}_k^q)^{-1} \log(\boldsymbol{q}_k(\lambda_2) * \overline{\boldsymbol{q}_k^*}) = \mathrm{S}_2. \quad (34)$$

Unlike in (20), where the calculation of $\{\boldsymbol{p}_k, \boldsymbol{q}_k\}$ is coupled through the covariance matrices, here, the smoothed positions and orientations $\{\boldsymbol{p}_k(\lambda_1), \boldsymbol{q}_k(\lambda_2)\}$ are calculated in a decoupled way by solving:

$$\frac{1}{2} \sum_{k=1}^{n} (\boldsymbol{p}_k - \boldsymbol{p}_k^*)^T (\boldsymbol{\Sigma}_k^p)^{-1} (\boldsymbol{p}_k - \boldsymbol{p}_k^*) + \lambda_1 g_1(\boldsymbol{p}), \quad (35)$$

$$\frac{1}{2} \sum_{k=1}^{n} \log(\boldsymbol{q}_k * \overline{\boldsymbol{q}_k^*})^T (\boldsymbol{\Sigma}_k^q)^{-1} \log(\boldsymbol{q}_k * \overline{\boldsymbol{q}_k^*}) + \lambda_2 g_2(\boldsymbol{q}). \quad (36)$$

Assuming that $\boldsymbol{\Sigma}_k^p$ and $\boldsymbol{\Sigma}_k^q$ are the covariance matrices of the measurements, the acceptable values for S_1 and S_2 are within the range $N - \sqrt{2N} \leq \mathrm{S}_1, \mathrm{S}_2 \leq N + \sqrt{2N}$, $N = n + 1$. This approach requires that we solve two additional nonlinear zero finding problems (33) and (34), but these are single variable, scalar, nonlinear equations, and can therefore easily be solved with standard solvers for scalar functions, such as, e.g., `fzero` available in MATLAB. Every time we need to compute the value of (33) or (34), we must first solve the decoupled systems (35) and (36) to obtain $\boldsymbol{p}_k(\lambda_1)$ and $\boldsymbol{q}_k(\lambda_2)$, which are needed for the calculation of (33) and (34) at the given λ_1 and λ_2, respectively. While the problems of determining λ_1 and λ_2 are not totally independent of each other, the approach of

$\lambda_1 = \lambda_2 = 1$		$\lambda_1 = \lambda_2 = 10^5$	
$F(\boldsymbol{p}^i, \boldsymbol{q}^i)$	$\|\nabla F_i(0,0)\|$	$F(\boldsymbol{p}^i, \boldsymbol{q}^i)$	$\|\nabla F_i(0,0)\|$
1.8854645e+04	7.6323626e+03	1.8799571e+09	7.6100687e+08
1.4852889e+03	2.1105371e+01	5.2411567e+05	1.4645490e+07
1.4852889e+03	1.2726109e-04	3.4902288e+05	7.6035127e+04
1.4852889e+03	2.2911019e-06	3.4901791e+05	8.8733884e+01
1.4852889e+03	1.1957504e-07	3.4901790e+05	2.3231791e+00
1.4852889e+03	1.8406582e-09	3.4901790e+05	2.7915866e-01
1.4852889e+03	3.6851655e-10	3.4901790e+05	1.6016155e-02
		3.4901790e+05	2.1152017e-03
		3.4901790e+05	1.2825353e-04
		3.4901790e+05	1.7447888e-05
		3.4901790e+05	4.6314701e-06

Table 1. Convergence of the Gauss-Newton method for different smoothing parameters

decoupling the calculation of both smoothing parameters worked well in practice.

The described method finds a smooth sequence of hand postures that approximate the measurements well. To generate a continuous motion trajectory that can be used for robot control, one must interpolate the smoothed postures. While standard techniques for interpolation in $\mathbb{R}^n$ [26] can be utilized for the interpolation of position vectors, more specialized methods are needed for smooth interpolation on SO(3). The most commonly used quaternion interpolation method is spherical linear interpolation (Slerp), proposed by [21], but more advanced methods that can ensure higher-order smoothness also exist [27].

5. Experimental Results and Conclusions

We applied the developed method for the reconstruction of 15 real hand motions. In these experiments, the Gauss-Newton method always converged. As expected, the convergence was slower for larger values of smoothing parameters (see Tab. 1). The measured poses were used as a starting point in iteration (30) or (32). One example smoothed trajectory, which was calculated at the optimal values of the smoothing parameters, is depicted in Figures 2, 3 and 4. In this way, smoother trajectories were estimated that resulted in less jerky robot movements.

To show the benefit of the proposed approach, which considers full covariance matrices $\boldsymbol{\Sigma}_k$ and computes positions and orientations simultaneously, we compared it to scalar spline smoothing, where each component is evaluated separately. Decoupled smoothing of separate components of unit quaternions has an additional disadvantage that the smoothed quaternions are not unit quaternions. They should, therefore, be normalized after smoothing, which introduces additional errors. Note that scalar spline smoothing also requires the determination of an optimal smoothing parameter λ. The comparison was done in a simulation experiment where the correct position $\boldsymbol{p}(t) \in \mathbb{R}^3$ and orientation trajectories $\boldsymbol{q}(t) \in S^3$ were known. To the simulated trajectories, we added Gaussian noise using preselected covariance matrices $\boldsymbol{\Sigma}_k$. The quality of approximation was evaluated using the mean error, namely:

	ME_p (mm)	ME_q (deg)
Proposed approach	1.04	1.39
Scalar spline smoothing	1.81	2.07

Table 2. Comparison of smoothing with the proposed approach and with scalar spline smoothing, where each position and quaternion dimension is smoothed separately.

$$ME_p = \frac{1}{n}\sum_{k=1}^{n} \|\boldsymbol{p}_k - \boldsymbol{p}(t_k)\|, \tag{37}$$

$$ME_q = \frac{1}{n}\sum_{k=1}^{n} d(\boldsymbol{q}_k, \boldsymbol{q}(t_k)), \tag{38}$$

where d is the metric on S^3 defined in Eq. (9). Results for a typical trajectory are shown in Tab. 2. Significant improvement could be achieved both in position and the orientation trajectory.

In summary, in this paper we developed two trajectory estimation methods on $\mathbb{R}^3 \times S^3$ - one based on Gauss-Newton iteration and the other on Levenberg-Marquardt iteration. We have also shown how to treat the measurement errors and suggested an approach for the automatic calculation of smoothing parameters. The method based on Gauss-Newton iteration turned out to be sufficient in our experiments and converged faster. However, it might become necessary to apply the method based on Levenberg-Marquardt iteration for the data containing more noise. With our approach, we were able to reconstruct smooth trajectories on $\mathbb{R}^3 \times S^3$ using real data obtained by measurement systems with or without markers. In this way, we can provide a high quality input for imitation learning systems.

6. Acknowledgments

This research has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 600578, ACAT.

7. References

[1] Ruediger Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2-3):109–116, 2004.

[2] H. Asada and J. J. Slotine. *Robot Analysis and Control*. John Wiley and Sons, 1968.

[3] Volker Krüger, Danica Kragic, Aleš Ude, and Christopher Geib. The meaning of action: a review on action recognition and mapping. *Advanced Robotics*, 21(3):1473–1501, 2007.

[4] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL, 1994.

[5] Kourosh Khoshelham. Accuracy analysis of Kinect depth data. In *ISPRS Workshop on Laser Scanning 2011*, Calgary, Canada, 2011.

[6] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition with Kinect sensor. In *ACM International Conference on Multimedia*, pages 759–760, Scottsdale, Arizona, 2011.

[7] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1297–1304, Providence, RI, 2011.

[8] Aleš Ude. Filtering in a unit quaternion space for model-based object tracking. *Robotics and Autonomous Systems*, 28(2-3):163–172, 1999.

[9] John L. Crassidis, F. Landis Markley, and Yang Cheng. A survey of nonlinear attitude estimation methods. *Journal of Guidance, Control, and Dynamics*, 30(1):12–28, 2007.

[10] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 52(2):1203–1218, 2008.

[11] L. Armesto, J. Tornero, and M. Vincze. On multi-rate fusion for non-linear sampled-data systems: Application to a 6d tracking system. *Robotics and Autonomous Systems*, 56:706–715, 2008.

[12] Henry Himberg and Yuichi Motai. Head orientation prediction: Delta quaternions versus quaternions. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 39(6):1382–1392, 2009.

[13] Aleš Ude. Nonlinear least squares optimisation of unit quaternion functions for pose estimation from corresponding features. In *14th International Conference on Pattern Recognition (ICPR)*, pages 425–427, Brisbane, Australia, 1998.

[14] Venu Madhav Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 684–691, Washington, DC, 2004.

[15] Raghav Subbarao, Yakup Genc, and Peter Meer. Nonlinear mean shift for robust pose estimation. In *IEEE Workshop on Applications of Computer Vision*, Austin, Texas, 2007.

[16] Henry Himberg and Yuichi Motai. Pose estimation using linearized rotations and quaternion algebra. *Acta Astronautica*, 68(1-2):101–112, 2011.

[17] John L. Crassidis, F. Landis Markley, and Yang Cheng. Optimal control of a rigid body using geometrically exact computations on se(3). *45th IEEE Conference on Decision & Control*, pages 2710–2715, 2006.

[18] P. A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2009.

[19] L. Machado, F. Silva Leite, and K. Krakowski. Higher-order smoothing splines versus least-squares problems on Riemannian manifolds. *Journal of Dynamical and Control Systems*, 16(1):121–148, 2010.

[20] Chafik Samir, P.-A. Absil, Anuj Srivastava, and Eric Klassen. A gradient-descent method for curve fitting on riemannian manifolds. *Foundations of Computational Mathematics*, 12:49–73, 2012.

[21] Ken Shoemake. Animating rotation with quaternion curves. *Computer Graphics (Proc. SIGGRAPH '85)*, 19(3):245–254, 1985.

[22] Jeffrey A. Fessler. Nonparametric fixed-interval smoothing of nonlinear vector-valued measurements. *IEEE Transactions on Signal Processing*, 39(4):907–913, April 1991.

[23] Jeffrey A. Fessler. Nonparametric fixed-interval smoothing with vector splines. *IEEE Transactions on Signal Processing*, 39(4):852–859, April 1991.

[24] Min-Ho Kyung, Myung-Soo Kim, and Sung-Je Hong. A new approach to through-the-lens camera control. *Graphical Models and Image Processing*, 58(3):262–285, May 1996.

[25] Christian H. Reinsch. Smoothing by spline functions. *Numerische Mathematik*, 10:177–183, 1967.

[26] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, New York, 1978.

[27] Myung-Soo Kim and Kee-Won Nam. Interpolating solid orientations with circular blending quaternion curves. *Computer-Aided Design*, 27(5):385–398, 1995.