Available online at www.sciencedirect.com

SCIENCE DIRECT®

Robotics and Autonomous Systems 47 (2004) 93–108

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Programming full-body movements for humanoid robots by observation

Aleš Ude [a,b,*], Christopher G. Atkeson [a,c], Marcia Riley [d,e]

[a] *ATR Computational Neuroscience Laboratories, Department of Humanoid Robotics and Computational Neuroscience, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan*

[b] *Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia*

[c] *Carnegie Mellon University, Robotics Institute, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA*

[d] *ATR Human Information Science Laboratories, Department 2, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan*

[e] *College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, GA 30332-0280, USA*

Abstract

The formulation and optimization of joint trajectories for humanoid robots is quite different from this same task for standard robots because of the complexity of humanoid robots' kinematics and dynamics. In this paper we exploit the similarity between human motion and humanoid robot motion to generate joint trajectories for humanoids. In particular, we show how to transform human motion information captured by an optical tracking device into a high dimensional trajectory for a humanoid robot. We propose an automatic approach to relate humanoid robot kinematic parameters to the kinematic parameters of a human performer. Based on this relationship we infer the desired trajectories in robot joint space. B-spline wavelets are utilized to efficiently represent the trajectories. The density of the basis functions on the time axis is selected automatically. Large-scale optimization techniques are employed to solve the underlying computational problems efficiently. We applied our method to the task of teaching a humanoid robot how to make various naturally looking movements.

Keywords: Humanoid robots; Joint trajectories; Optimization

1. Introduction

Generation of motion for humanoid robots is quite different from that of standard robots because of the large numbers of joints, coupling between joints, redundancies, and people's expectations that humanoid robots move like humans. A possible solution is to develop a special motion creator with a rich graphical user interface that enables programmers to tackle these problems in an interactive manner [8]. In this paper we take another approach and investigate the application of motion capture techniques for the generation of humanoid movements. This is motivated by the fact that motion capture has become a premier technique for animation of human like characters

* Corresponding author.
E-mail address: ales.ude@ijs.si (A. Ude).

in computer graphics. Unlike other robots, humanoids can perform human like movements and we can exploit this to generate appropriate trajectories by observing human motion.

Our basic assumption is that humanoid robot kinematics can be embedded into human body kinematics (see Fig. 1). The humanoid's kinematics do not need to reproduce every aspect of human kinematics (indeed, this would be impossible), but should reproduce some properties of the human body. The best known humanoids developed up to now, e.g. ASIMO, QRIO, SDR, HRP-2, HOAP-2, etc., fulfill this condition and images like Fig. 1 are commonly distributed with humanoids to demonstrate that they can perform human like movements. Only such robots can truly be called humanoid and an increasing number of researchers think that humanoids are the most appropriate robots to bring robotics away from factory floor because of their similarity to people.

Our approach involves capturing full-body motions of a human performer using an optical tracking device, which provides 3D locations of identified active markers that are currently in view. We break the problem up into three parts: (1) identifying a kinematic model of the person being observed, (2) estimating the joint angle trajectories of the motion to be imitated, and (3) transforming the motion so that it is appropriate for the kinematics of the robot. This paper presents a theoretically well founded and experimentally tested solution to problems (1) and (2), while the technique used to solve problem (3) is relatively simple and is only briefly presented in the discussion.

1.1. Related research

The automatic construction of kinematic models for robot manipulators is a well-established field in robotics. A lot of research has been done to identify the most suitable kinematic parameter system according to criteria such as completeness, proportionality and equivalence (see [7] for a review). The unknown kinematic parameters are most commonly identified from end-effector pose measurements and robot joint position readings. However, joint position data is not available when constructing a kinematic model of a human body. Techniques that were developed by the computer graphics community to estimate human body kinematics are therefore more relevant to our problem [2,10,13]. These approaches have shown that it is possible to identify human body kinematics from motion capture data.

Computer animators have the liberty to use any kinematic model they want when generating movements for virtual characters. This is not the case when working with a humanoid robot because we must also take into account the properties of the robot which is supposed to learn the observed movements. Our kinematic model should not only model human motion well, but should also be related to the kinematics of the available robot. Obviously, this aspect does not need to be considered when generating animations.

Finite elements such as B-splines are often used to represent joint space trajectories when a parametric form of the trajectory is not known. The main problem with splines is that an optimal set of basis functions cannot easily be determined. If there are not enough basis functions, the generated motion may be far from the desired motion. If there are too many of them, the computational complexity is increased unnecessarily due to the larger number of variables as well as the resulting ill-conditioning of the linear subproblems that arise in the optimization process. Among approaches proposed to resolve this problem, a wavelet representation, in which the trajectory is represented hierarchically [5], seems to be one of the most promising.

An alternative approach to splines was recently proposed in [6]. Instead of splines these authors use a set of differential equations that forms a control policy to encode the information about joint space motion. They show that their representation has various advantages such as robustness against perturbations, ease of re-use, etc. Our work concentrates on the transformation of the demonstrated motion acquired in Cartesian space into humanoid robot joint space. For this task, a spline-based representation seems to be ideal because it allows us to adapt local and global properties of the trajectory.

A few approaches have been proposed in the past to convert human movements into humanoid robot movements [3,12,15]. Unlike these authors, who primarily deal with the conversion of human motion into humanoid robot motion in real-time based on images acquired from on-board cameras and who match their kinematic models manually, we concentrate here on the accurate reconstruction of complex, articulated human movements, even if the

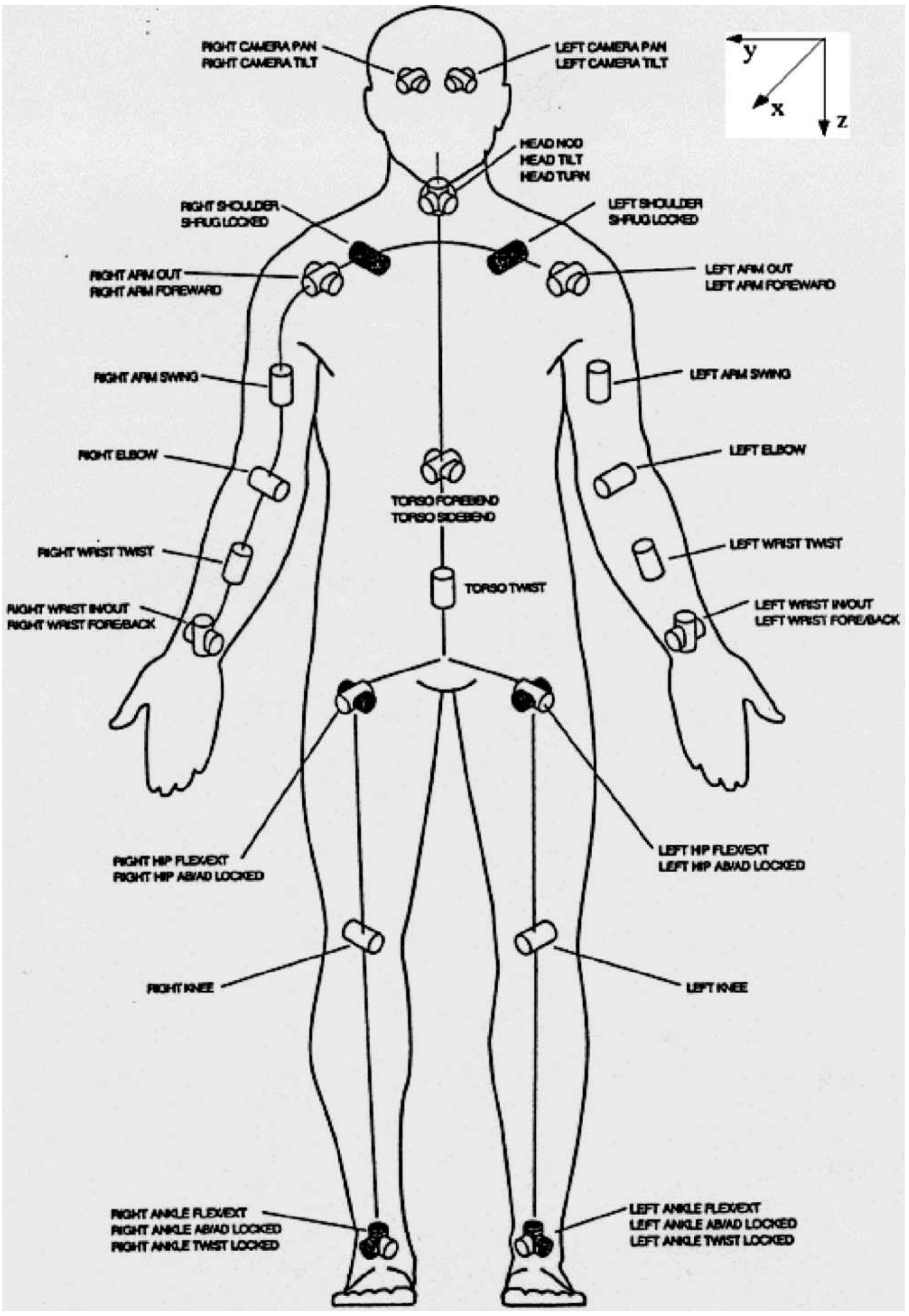

Fig. 1. Kinematic structure of our humanoid DB (see also Fig. 4). In the upright position with extended arms and legs, all joint axes are parallel to one of the three main axes of the body (forward/backward: x-axis, left/right: y-axis, up/down: z-axis).

resulting system does not work in real-time. In this way we can generate high degree of freedom naturally looking motions and also provide a high-quality input to systems like [6], which can generalize robot joint space trajectories.

2. Kinematic modeling

To a certain degree of accuracy, human motion can be modeled as an articulated motion of rigid body parts. The first decision to be made is the selection of an adequate kinematic parameter convention to model motions generated by such systems. Since Denavit–Hartenberg parameterization is inappropriate because it lacks proportionality, other notations distinguished by various desirable properties have been proposed for the calibration of kinematic models [7]. Our goal is to model human motion by a specific kinematic structure of the humanoid robot, therefore it is very natural to specify such a parametric model using twists [9]. A twist is defined by a point on the joint axis and by the direction of the axis. Twists allow us to work directly in a global body coordinate system instead of specifying transformations between consecutive local coordinate frames in a kinematic chain. Such a representation is not minimal (it involves six instead of four parameters per transformation), but we shall show later how to derive a set of independent parameters for calibration based on twists.

To describe the kinematic body model by twists, we need to determine the location and direction of joint axes on the human body. For a revolute joint,[1] let $\boldsymbol{n}_i$ be the unit vector in the direction of the joint axis and let $\boldsymbol{q}_i$ be any point on the axis, both given in a global body coordinate system at zero configuration, i.e. with all joint angles θ_i equal to 0. In this case, the twist $\boldsymbol{\xi}_i$ describing motion about ith joint axis has the form $\boldsymbol{\xi}_i = [\,(-\boldsymbol{n}_i \times \boldsymbol{q}_i)^{\mathrm{T}} \quad \boldsymbol{n}_i^{\mathrm{T}}\,]^{\mathrm{T}}$. The displacement of a body part caused by motion about such a joint by an angle θ_i can be calculated by an exponential map $\exp(\theta_i \boldsymbol{\xi}_i)$ [9].

To generate humanoid robot motion from human motion capture data, we need to relate the human joint motion to the 3D marker motion. If the coordinates of a marker in a local coordinate system of a rigid body part to which it is attached are given by $\boldsymbol{y}_j$, then its 3D position at body configuration $(\boldsymbol{r}, \boldsymbol{d}, \theta_1, \ldots, \theta_n)$ can be calculated as follows:

$$\tilde{\boldsymbol{y}}_j = \boldsymbol{g}(\boldsymbol{r}, \boldsymbol{d}) \exp(\theta_{i_1} \boldsymbol{\xi}_{i_1}) \cdots \exp(\theta_{i_{n_x}} \boldsymbol{\xi}_{i_{n_x}}) \cdot \boldsymbol{G}_x \cdot \boldsymbol{y}_j. \tag{1}$$

Here $\boldsymbol{\xi}_{i_1}, \ldots, \boldsymbol{\xi}_{i_{n_x}}$ are the twists that describe the kinematic chain generating motion of the marker $\boldsymbol{y}_j$, $\boldsymbol{G}_x$ is the homogeneous matrix combining the position and orientation of the local body part coordinate system to which the marker is attached with respect to the global body coordinate system at zero configuration (for DB, x = torso, pelvis, head, left upper arm, left lower arm, left hand, right upper arm, right lower arm, right hand, left upper leg, left lower leg, left foot, right upper leg, right lower leg, right foot), $\boldsymbol{r}$ and $\boldsymbol{d}$ are the orientation (represented by a rotation vector) and position of a global body coordinate system with respect to the world coordinate system, and $\boldsymbol{g}(\boldsymbol{r}, \boldsymbol{d})$ denotes the homogeneous matrix corresponding to $\boldsymbol{r}$ and $\boldsymbol{d}$. Note that the set of twists affecting motion of the marker varies with the identity of the body part to which the marker is attached.

In this setting, the kinematic modeling involves the specification of joint axes positions and directions and the placement of local body part coordinate systems, all these parameters being specified in the global body coordinate system when the body is at zero configuration. The location of the global body coordinate system should also be determined.

3. Automatic model generation

The relationship between the kinematics of a human body and humanoid robot kinematics is established by modeling the performer's kinematics by a model standard for humanoid robots, yet scaled to the physical size of

[1] All current humanoid robots possess only revolute joints, therefore we limit our attention to this type of joints. It is, however, straightforward to also consider prismatic joints.

the performer. In biomechanics and computer graphics, neck, shoulder, hip and ankle joints are normally modeled as spherical joints. This is not the case with current humanoid robots, which rather have a number of consecutive revolute joints (typically three) that are orthogonal but do not necessarily intersect at all joints. In addition, at the zero configuration the directions of the corresponding joint axes are normally aligned with the three main body axes (up/down, left/right, and forward/backward). We consider the body to be in zero configuration when a subject or a robot stands in the upright position with extended arms and legs.

Regardless of the kinematic parameter system in use, the actual values of the parameters (joint angles) depend on the choice of local coordinate systems because they specify transformations between them. We orient the global and the local body part coordinate systems on the human performer in such a way that they are all aligned when the performer stands in the upright position with extended arms and legs. The axes are chosen to be parallel to the main body axes in this configuration. Directions of joint axes at zero configuration therefore coincide with the coordinate axes, which gives us a reference posture for the mapping of human joint angles onto robot joint angles.

3.1. Practical determination of coordinate systems and marker positions on the body

To be able to relate the marker motion to the joint motion of the observed actor, we must know the positions of markers on the body in local body part coordinate systems. This data can be acquired by measuring the position of markers while the actor stands in a posture for which the joint angles are known. The zero configuration is suitable for this purpose. In our experiments we asked the performers to stand in such a position and recorded the markers. For each body part, the origin of a local coordinate system is taken to be at the centroid of all markers attached to it.

To determine the directions of the main body axes, the optical tracking system is calibrated so that we know where up and down is[2]. We can therefore assume one of the body axis at zero configuration (z-axis on our robot) to be parallel to the z-axis of the world coordinate system. The axis from the left to the right shoulder can be estimated by attaching two markers at the opposite positions on the left and right shoulder. The line between the two opposite positions defines the second body axis (y-axis in our system). Due to inaccuracies in the measurements and/or marker placement, the derived axes are not exactly orthogonal, therefore we enforce the orthogonality by calculating a new z-axis direction

$$\boldsymbol{z} = \left(\frac{\tilde{\boldsymbol{y}}_\mathrm{r} - \tilde{\boldsymbol{y}}_\mathrm{l}}{\|\tilde{\boldsymbol{y}}_\mathrm{r} - \tilde{\boldsymbol{y}}_\mathrm{l}\|} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) \times \frac{\tilde{\boldsymbol{y}}_\mathrm{r} - \tilde{\boldsymbol{y}}_\mathrm{l}}{\|\tilde{\boldsymbol{y}}_\mathrm{r} - \tilde{\boldsymbol{y}}_\mathrm{l}\|}, \tag{2}$$

where $\tilde{\boldsymbol{y}}_\mathrm{l}$ and $\tilde{\boldsymbol{y}}_\mathrm{r}$ are the coordinates of the two special markers attached to the left and right shoulder, respectively. This new direction is orthogonal to the y-axis and deviates from the original z-axis the least among all directions orthogonal to y. Finally, the x-axis can be calculated using the cross product.

The origin of the global body coordinate system is assumed to be in the middle of a line connecting the two specially placed markers. It is rigidly attached to the torso. At zero configuration, the global body coordinate system and all local body part coordinate systems are assumed to be aligned, which makes it possible to calculate the position and orientation of each body part at zero configuration in the global body coordinate system ($\boldsymbol{G}_x$ in Eq. (1)). We can also use this assumption to transform the world coordinates of the markers into the local coordinates ($\tilde{\boldsymbol{y}}_j$ and $\boldsymbol{y}_j$ in Eq. (1)).

Because all joint angles are equal to 0, Eq. (1) simplifies to

$$\tilde{\boldsymbol{y}}_j = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{d} \\ 0 & 1 \end{bmatrix} \cdot \boldsymbol{G}_x \cdot \boldsymbol{y}_j. \tag{3}$$

[2] This is done by aligning the z-coordinate of the calibration object with the up/down direction in the real world.

The position of the global body coordinate system is given by $\boldsymbol{d} = (\tilde{\boldsymbol{y}}_l + \tilde{\boldsymbol{y}}_r)/2$, while the orientation is given by the rotation matrix $\boldsymbol{R}$ having the three coordinate axes in its columns

$$\boldsymbol{R} = \left[\frac{\tilde{\boldsymbol{y}}_r - \tilde{\boldsymbol{y}}_l}{\|\tilde{\boldsymbol{y}}_r - \tilde{\boldsymbol{y}}_l\|} \times \boldsymbol{z} \quad \frac{\tilde{\boldsymbol{y}}_r - \tilde{\boldsymbol{y}}_l}{\|\tilde{\boldsymbol{y}}_r - \tilde{\boldsymbol{y}}_l\|} \quad \boldsymbol{z} \right]. \tag{4}$$

Due to the alignment by construction, the relative orientation of each body part coordinate system with respect to the global body coordinate system at zero configuration is given by the identity matrix. The relative position can be calculated by subtracting the position of the global body coordinate system from the position of local coordinate systems. Thus for the body part x we have

$$\boldsymbol{G}_x = \begin{bmatrix} \boldsymbol{I} & \frac{1}{M_x} \sum_{\forall j \in x} \tilde{\boldsymbol{y}}_j - \boldsymbol{d} \\ 0 & 1 \end{bmatrix}, \tag{5}$$

where M_x is the number of markers attached to the body part. Finally, the local coordinates of markers $\boldsymbol{y}_j$ can be obtained by inverting Eq. (3).

3.2. Estimation of joint positions

At this point, the joint axis locations are the only parameters that still need to be estimated. To estimate these parameters, the subject is asked to perform a set of movements which are measured by the motion capture system. He or she should exercise motions around all relevant degrees of freedom if the method is to return an unambiguous answer. Instead of trying to estimate all joint locations in one big optimization process, we decided to split the estimation in 10 separate smaller optimization problems: neck, waist, left and right shoulder + elbow, left and right wrist, left and right hip + knee, and left and right ankle.

The position of a joint axis in 3D space has only two degrees of freedom because the position along the axis is arbitrary. Recall from Section 3.1 that directions of all joint axes at zero configuration are parallel to the coordinate axes of the global body coordinate system and are thus all given by one of the vectors $[\,1 \quad 0 \quad 0\,]^{\mathrm{T}}$, $[\,0 \quad 1 \quad 0\,]^{\mathrm{T}}$, $[\,0 \quad 0 \quad 1\,]^{\mathrm{T}}$, $[\,-1 \quad 0 \quad 0\,]^{\mathrm{T}}$, $[\,0 \quad -1 \quad 0\,]^{\mathrm{T}}$ and $[\,0 \quad 0 \quad -1\,]^{\mathrm{T}}$. It is easy to see that the coordinate corresponding to the non-zero coordinate of the joint axis direction vector does not influence the twist. For example, for a joint axis located at $[\,a \quad b \quad c\,]^{\mathrm{T}}$ and parallel to $[\,0 \quad 1 \quad 0\,]^{\mathrm{T}}$, the corresponding twist is equal to

$$[(-[\,0 \quad 1 \quad 0\,]^{\mathrm{T}} \times [\,a \quad b \quad c\,]^{\mathrm{T}})^{\mathrm{T}} [\,0 \quad 1 \quad 0\,]]^{\mathrm{T}} = [\,-c \quad 0 \quad a \quad 0 \quad 1 \quad 0\,]^{\mathrm{T}} \tag{6}$$

and is thus independent of the second coordinate, i.e. b. It follows that we can parameterize the location of each joint axis by two coordinates different from the non-zero coordinate of the direction vector of the axis.

Now that we have the independent coordinates for the joint axis locations, we can estimate them by minimizing a suitable optimization criterion. Apart from joint axis locations, we also need to estimate the position and orientation of the body in space as well as the joint angles in order to match the model markers with the measured marker positions. To make the optimization process smaller, we estimate all the degrees of freedoms prior to the joints under consideration in a separate optimization process using the sequential method from Section 4.1. The optimization process then involves only the joint angle locations that need to be estimated and the corresponding joint angles. Still, this results in a very large optimization problem. For instance, to estimate the neck joint locations, we need to minimize the following optimization criterion:

$$\begin{aligned} &h(a, b, c, d, e, f, \{\theta_1(t_k), \theta_2(t_k), \theta_3(t_k)\}_{k=1}^{N}) \\ &\quad = \sum_{k=1}^{N} \sum_{j \in \mathrm{head}} \|\boldsymbol{g}(\boldsymbol{r}(t_k), \boldsymbol{d}(t_k)) \exp(\theta_1(t_k)\boldsymbol{\xi}_1(a, b)) \exp(\theta_2(t_k)\boldsymbol{\xi}_2(c, d)) \\ &\qquad \times \exp(\theta_3(t_k)\boldsymbol{\xi}_3(e, f)) \cdot \boldsymbol{G}_{\mathrm{head}} \cdot \boldsymbol{y}_j - \tilde{\boldsymbol{y}}_j(t_k)\|^2. \end{aligned} \tag{7}$$

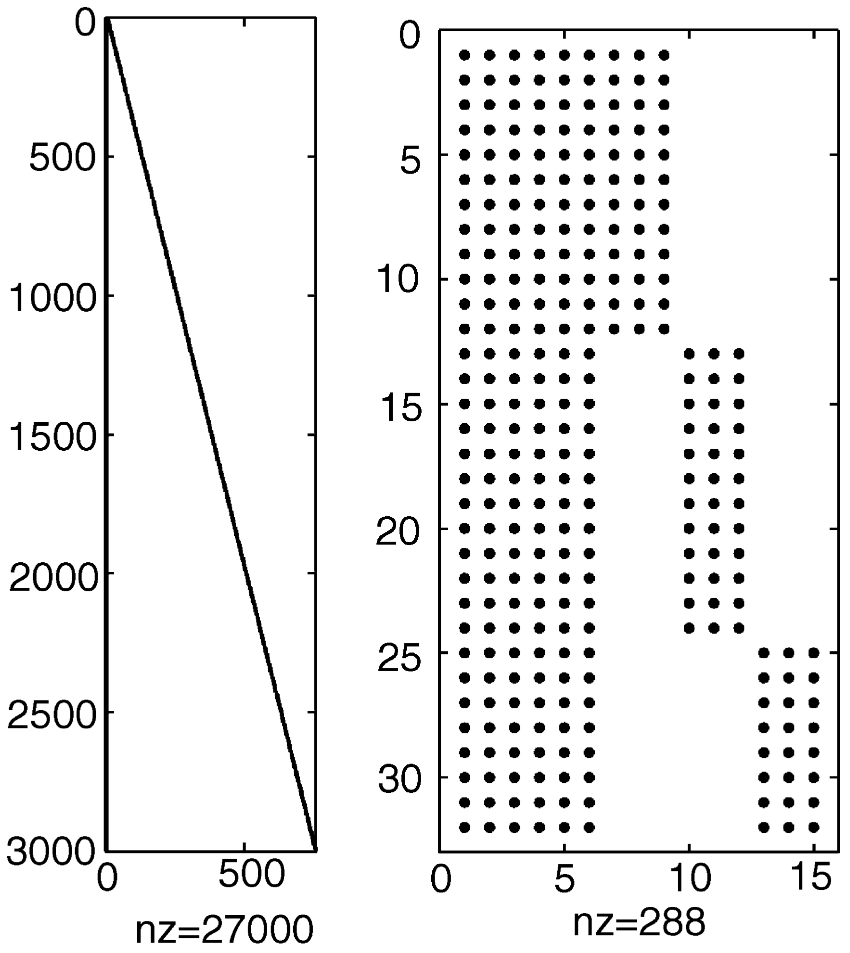

Fig. 2. Sparsity pattern of the Jacobian of the vectorized optimization criterion (7). The full Jacobian is shown on the left. On the right, the upper left corner of the Jacobian corresponding to the first three measurement sets and variables is depicted.

Note that the number of parameters increases with the number of measurement times. We typically used 300 measurements in each of the 10 optimization problems and therefore needed to estimate 906 or 1208 variables per optimization problem. To solve such large optimization problems we utilized a subspace trust region approach and sparse matrix algebra [4]. The trust region approach requires the calculation of first derivatives of the criterion function h. Fortunately, the Jacobian of h (shown in Fig. 2) is sparse and our optimization method uses sparse matrix algebra to reduce the computation time.

3.3. Discussion of the technique

We first note that our system does not directly estimate body part positions and orientations. Indeed, we did not always have enough marker data to do this in our experiments (we typically had 2, 3, or 4 markers per body part when performing calibration). Thus the parameters describing joint motion preceding the axes under consideration are either determined in a previous step or are part of the current estimation process.

Some of the constraints imposed on the humanoid robot kinematics can be relaxed. For example, it is not necessary that the robot joint axes are parallel to the main body axes or even that they are orthogonal. However, to map human motion onto a humanoid robot body, we need to know the orientation of these axes with respect to the main body axes (or any other coordinate system chosen as a global body coordinate system). We could then parameterize the kinematic model using these *known* joint axis directions. However, looking at current humanoid robots, the practical value of this is rather limited and since this greatly complicates the notation while clouding important issues, we do not present the necessary mathematics in this section.

Special placement of the two shoulder markers could be avoided by estimating the left and right shoulder joint position as in [10] or [13]. These two methods are based on the estimation of neighboring body part positions and orientations, but this is problematic with systems that provide only sparse marker data such as our Optotrak. In addition, the estimation of joint locations is a noisy process and the estimates vary depending on the movements that were actually performed because the human shoulder is not a perfect spherical joint and has some translation capability. Hence there is no guarantee that the parameters generated in this way would be more accurate than the parameters acquired from special marker placement. For these reasons we decided to stick with our original approach which is simpler to implement.

4. Motion generation from marker data

Our trajectory planning method should generate motions which are perceptually similar to the motion of the performer. To attain this, we minimize the difference between the measured marker positions and marker positions generated by the recovered joint angles for each frame of motion over the set of body configurations $(\boldsymbol{r}(t_k), \boldsymbol{d}(t_k), \theta_i(t_k))$

$$\begin{aligned}&\sum_{j=1}^{M} \|\boldsymbol{g}(\boldsymbol{r}(t_k), \boldsymbol{d}(t_k)) \exp(\theta_{i_1}(t_k)\boldsymbol{\xi}_{i_1}) \cdots \exp(\theta_{i_{n_x}}(t_k)\boldsymbol{\xi}_{i_{n_x}}) \cdot \boldsymbol{G}_x \cdot \boldsymbol{y}_j - \tilde{\boldsymbol{y}}_j(t_k)\|^2 \\ &\quad = \sum_{j=1}^{M} \|\boldsymbol{h}_j(\boldsymbol{r}(t_k), \boldsymbol{d}(t_k), \theta_1(t_k), \dots, \theta_n(t_k)) - \tilde{\boldsymbol{y}}_j(t_k)\|^2 \\ &\quad = \|\boldsymbol{h}(\boldsymbol{r}(t_k), \boldsymbol{d}(t_k), \theta_1(t_k), \dots, \theta_n(t_k)) - \tilde{\boldsymbol{y}}(t_k)\|^2, \end{aligned} \tag{8}$$

where $\boldsymbol{h} = [\boldsymbol{h}_1^{\mathrm{T}}, \dots, \boldsymbol{h}_M^{\mathrm{T}}]^{\mathrm{T}}$ and $\boldsymbol{y}(t_k) = [\boldsymbol{y}_1(t_k)^{\mathrm{T}}, \dots, \boldsymbol{y}_M(t_k)^{\mathrm{T}}]^{\mathrm{T}}$. $\tilde{\boldsymbol{y}}_j(t_k)$ denotes the measured markers at time t_k, M is the number of markers, and n the number of joints.

4.1. Sequential motion estimation

The kinematics of humanoid robots are much too complicated to allow for analytical solutions, although partial solutions are sometimes feasible [1]. The optimization of criterion (8) is a standard non-linear least-squares optimization problem and there are several good iterative methods that can be applied to solve it [4]. Our experiments have shown that the incorporation of constraints into the above optimization criterion can increase the quality of the reconstructed motion. For example, the elbow joint angle is equal to 0 when the performer fully extends his arm. It is visually very disturbing if the estimated elbow angle turns out to be negative because of measurement noise and/or inaccuracies in the kinematic model. Thus it is sensible to add simple bounds which limit the angles to the area that can actually be reached by a human to the above optimization criterion

$$l_i \le \theta_i \le u_i. \tag{9}$$

Efficient algorithms are available for the minimization of the non-linear least-squares criterion (8) subject to box constraints (9). We utilized a method based on the Gauss–Newton iteration available in the MATLAB Optimization Toolbox to solve this problem.

4.2. Estimation of global trajectory

The straightforward approach of sequentially minimizing criterion (8) at each measurement time has several deficiencies. First of all, optical motion capture systems often cannot recover the positions of all markers due to occlusions. This can result in underconstrained linear systems causing the optimization process to break down. Moreover, experiments showed that even when the positions of all markers can be recovered, the optimization process can still break down because of the singularities in the kinematic model. Finally, besides making our robot move like a human, we also need to generate feasible robot motions.

The optimization process can be made more reliable by recovering a global trajectory instead of single configurations and by adding a regularization term to the objective function (8). Writing the full-body trajectory as $\boldsymbol{f}(t) = (\boldsymbol{r}(t), \boldsymbol{d}(t), \theta_1(t), \dots, \theta_n(t))$, we look for a function that minimizes

$$s(\boldsymbol{f}) = \frac{1}{2} \sum_{k=1}^{N} \sum_{j=1}^{M} \|\boldsymbol{h}_j(\boldsymbol{f}(t_k)) - \tilde{\boldsymbol{y}}_j(t_k)\|^2, \tag{10}$$

over all possible trajectories. Regularization can be achieved by minimizing the amplitude of physical quantities such as acceleration (second derivative) or jerk (third derivative)

$$r(\boldsymbol{f}) = \frac{1}{2} \int_0 1 \|\boldsymbol{f}^{(m)}(t)\|^2 \, \mathrm{d}t, \quad m = 2 \text{ or } 3. \tag{11}$$

Note that we normalize the time of our trajectory. In general we could penalize any weighted combination of kinematic variables such as acceleration, jerk, and violation of joint space or Cartesian soft limits.

The trajectory planning problem thus becomes

$$\min_{\boldsymbol{f}} \{s(\boldsymbol{f}) + \lambda r(\boldsymbol{f})\}, \tag{12}$$

where λ is the parameter governing the tradeoff between the two objective functions. Apart from the above-mentioned computational issues, it is advantageous to generate smooth trajectories also in order to reduce the wear and tear of the mechanical system, avoid exciting higher order dynamics, and because real actuators often have limits on their torque output or on the rate of change of output. Furthermore, we find that jerky motions do not look natural.

Since parametric forms of complex body movements are normally not known, we used a finite element method to represent the trajectory. As pointed out in Section 1, wavelets are suitable for the representation of general motions because they allow us to automatically determine the density of basis functions on the time axis.

4.3. B-spline wavelets

Let $V^j(m)$ be the set of $2^j + m$ endpoint-interpolating B-splines of degree m constructed from knot sequence

$$\frac{1}{2^j} \left[\underbrace{0, \dots, 0}_{m+1 \text{ times}}, 1, 2, \dots, 2^j - 2, 2^j - 1, \underbrace{2^j, \dots, 2^j}_{m+1 \text{ times}} \right]. \tag{13}$$

Linear spaces spanned by these splines are nested, i.e.

$$\mathcal{L}(V^0(m)) \subset \mathcal{L}(V^1(m)) \subset \mathcal{L}(V^2(m)) \subset \cdots, \tag{14}$$

where $\mathcal{L}(X)$ denotes the linear space spanned by members of X. Each orthogonal complement of $\mathcal{L}(V^j(m))$ in $\mathcal{L}(V^{j+1}(m))$ is called a wavelet space and its basis is denoted by $W^j(m)$. Members of $W^j(m)$ are called semiorthogonal wavelets because they are orthogonal to B-splines but not to each other. By definition $\mathcal{L}(W^j(m)) \subset \mathcal{L}(V^{j+1}(m))$, thus wavelets are piecewise polynomials of the same degree as the underlying B-splines. B-splines have small support, i.e. they are different from zero only on a short interval, and a construction for semiorthogonal wavelets with the smallest support, which are called B-spline wavelets, is given in [14]. We omit the details because of lack of space.

Let ϕ_i^j and ψ_i^j be members of $V^j(m)$ and $W^j(m)$, respectively. The multiresolution finite element approach assumes that the optimal trajectory can be written as a linear combination of B-spline wavelets:

$$\boldsymbol{f} = \sum_i \boldsymbol{C}_i \phi_i^L + \sum_{L \le j \le K} \sum_i \boldsymbol{D}_i^j \psi_i^j. \tag{15}$$

Here B-splines ϕ_i^L are fixed at the lowest possible resolution L, while the optimal set of wavelets ψ_i^j, $L \le j \le K$, should be determined automatically by the optimization procedure.

4.4. Large-scale optimization

By replacing $\boldsymbol{f}$ in the optimization problem (12) with the above linear combination of B-splines and wavelets, which are fixed in this section, we obtain a classic unconstrained optimization problem. Instead of minimizing over all functions from some function space, we can minimize over parameters $\boldsymbol{C}_i$, $\boldsymbol{D}_i^j$. Note, however, that the number of unknown parameters is very high. The full-body motion of DB involves 26 joints plus six degrees of freedom for the displacement in space. If we fix the highest resolution space of piecewise polynomials to $V^6(3)$, there are altogether 67 basis functions. Thus in this case the highest possible number of variables is $67 \times 32 = 2144$.

Trust region methods are suitable for solving large-scale optimization problems [4]. Let $\boldsymbol{H}$ and $\boldsymbol{g}$, respectively, be the Hessian and the gradient of the objective function (12) at the current estimate for unknown variables $\boldsymbol{x} = (\boldsymbol{C}_i, \boldsymbol{D}_i^j)$. The main idea of the trust region approach is to approximate the criterion function with a quadratic function in the neighborhood around the current estimate. The next approximation is thus computed by minimizing

$$\min_{\boldsymbol{x}} \{\tfrac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{g} \text{ such that } \|\boldsymbol{S}\boldsymbol{x}\| \le \Delta\}, \tag{16}$$

where $\boldsymbol{S}$ is a diagonal scaling matrix and Δ a positive scalar setting the size of the neighborhood.

The trust region approach needs the gradient and the Hessian of the criterion function. Let us define

$$\boldsymbol{c}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{h}_1\left(\sum_i \boldsymbol{C}_i\phi_i^L(t_1) + \sum_j\sum_i \boldsymbol{D}_i^j\psi_i^j(t_1)\right) - \tilde{\boldsymbol{y}}_1(t_1) \\ \vdots \\ \boldsymbol{h}_M\left(\sum_i \boldsymbol{C}_i\phi_i^L(t_N) + \sum_j\sum_i \boldsymbol{D}_i^j\psi_i^j(t_N)\right) - \tilde{\boldsymbol{y}}_M(t_N) \end{bmatrix}. \tag{17}$$

Comparing (17) with the objective function (10) we note that $s(\boldsymbol{f}) = (1/2)\|\boldsymbol{c}(\boldsymbol{x})\|^2$. Let $\boldsymbol{J}$ be the Jacobian of $\boldsymbol{c}$ at the current estimate. It is easy to verify that the gradient of s is equal to $\boldsymbol{J}^{\mathrm{T}}\boldsymbol{c}(\boldsymbol{x})$ while the Hessian of s is given by $\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}+$ second order terms. It is a common practice in non-linear least-squares problems to neglect the second order terms, which are expensive to calculate, and to approximate the Hessian by $\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$.

Calculation of the gradient and Hessian of the criterion function (11) involves the calculation of inner products of derivatives of basis functions

$$\int_0^1 \phi_i^{(m)}\phi_j^{(m)}\mathrm{d}t, \qquad \int_0^1 \phi_i^{(m)}\psi_j^{(m)}\mathrm{d}t, \qquad \int_0^1 \psi_i^{(m)}\psi_j^{(m)}\mathrm{d}t.$$

Let $\boldsymbol{\Omega}$ be the matrix of these inner products. It is easy to see that (11) can be rewritten as $r(\boldsymbol{f}) = (1/2)\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Omega}\boldsymbol{x}$. Thus the gradient of the objective function r is given by $\boldsymbol{\Omega x}$ and its Hessian is simply equal to $\boldsymbol{\Omega}$. We employed the Gaussian quadrature formulae to evaluate these integrals exactly.

The bulk of the computing time when calculating $\boldsymbol{J}$ is spent on the calculation of the body kinematics Jacobian at all measurement times. Due to the structure of our model, the kinematics Jacobian is sparse. Moreover, due to the minimal support property of B-splines and wavelets, $\boldsymbol{J}$, $\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$, $\boldsymbol{\Omega}$ and the resulting combined Hessian $\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda\boldsymbol{\Omega}$ are also sparse.

We showed that the gradient and the Hessian of the combined criterion function (12) can be estimated as

$$\boldsymbol{g} = \boldsymbol{J}^{\mathrm{T}}\boldsymbol{c}(\boldsymbol{x}) + \lambda\boldsymbol{\Omega x}, \qquad \boldsymbol{H} \approx \boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda\boldsymbol{\Omega}. \tag{18}$$

Thus the next estimate for our trajectory can be computed by solving the trust region subproblem (16). To deal with the high dimensionality of the solution space, the solution of (16) is restricted to a two-dimensional subspace spanned by the direction of the gradient and direction of the negative curvature. The calculation of the next approximation in this 2D space is trivial. We were able to use the MATLAB Optimization Toolbox implementation of a trust region method for large-scale optimization problems and the sparse matrix capabilities of MATLAB for the resolution of sparse linear systems.

4.5. *Selecting the optimal resolution*

Criteria similar to the ones proposed in [5] for variational geometric modeling were applied to choose the optimal resolution: add more wavelets to better approximate the trajectory and remove the unneeded wavelets to obtain a solution with lower energy. While the first principle can be realized using hierarchical B-splines, the second criterion is much easier to realize in a wavelet basis because the necessary density is reflected in the magnitude of wavelet coefficients.

After calculating an estimate for the trajectory at a given resolution, we check the magnitude of wavelet coefficients. If they are below a manually tuned threshold, we remove the corresponding wavelets. The next step is the addition of higher resolution wavelets on time intervals where marker positions generated by the recovered trajectory poorly match the measured markers. Wavelets centered on such intervals are added to the solution and their initial coefficients are set to zero. Experimentally we set the threshold to 4 cm, which is about double the average marker error we typically had in our sequential tests. This procedure is repeated until a stable solution is found. To prevent cycling between adding and removing the wavelets, each wavelet can be removed only once; once it is removed it cannot be switched on again. While this is clearly a heuristic, such an approach allows us to quickly arrive at a stable solution.

5. Results and discussion

We carried out numerous experiments to test our method for the automatic generation of kinematic models. We also captured several motion trajectories involving full-body motion of a human performer to evaluate our trajectory generation approach. A marker-based measurement system Optotrak (see the web page http://www.ndigital.com/opto.html) was used for this purpose. Optotrak uses identifiable active markers which is advantageous because full-body movements often cause some of the markers to be occluded. Systems using passive markers are more prone to matching errors in such cases.

In our first set of experiments we tested the accuracy of the model construction process. The subjects exercised each joint complex in isolation trying to use all the degrees of freedom. Similar movements were performed in every motion capture session and the recovered kinematic parameters were very stable. The statistics for one subject (mean values and average errors), who exercised the upper body degrees of freedom 10 times, is shown in

Table 1
Locations of neck axes (in mm)

	x-Axis location			*y*-Axis location			*z*-Axis location		
Mean	0	1.7	−100.4	−78.1	0	−76.6	−66.1	−7.6	0
RMSE	22.0			24.6			5.5		

Table 2
Locations of elbow axes (in mm)

	Left elbow, *y*-axis location			Right elbow, *y*-axis location		
Mean	−51.9	0	325.0	−79.5	0	313.5
RMSE	1.4			1.3		

Table 3
Locations of left shoulder axes (in mm)

	x-Axis location			*y*-Axis location			*z*-Axis location		
Mean	0	−167.0	27.4	−72.3	0	6.3	−52.0	−215.5	0
RMSE	3.0			3.0			1.4		

Tables 1–6 (RMSE stands for root mean square error of the calculated joint locations with respect to the average joint location). A revolute joint is a rather poor approximation for the true neck kinematics, so it is not surprising that the determination of the joint axes on the head is much less repeatable than the determination of the joint axes on the arms. Note also that the results for both arms are reasonably symmetrical. The processing time needed to generate a 30 degrees of freedom kinematic model (DB has 26 DOFs excluding the eyes, but we added two DOFs

Table 4
Locations of right shoulder axes (in mm)

	x-Axis location			*y*-Axis location			*z*-Axis location		
Mean	0	175.8	31.0	−63.9	0	19.3	−51.6	213.6	0
RMSE	5.3			4.0			2.7		

Table 5
Locations of left wrist axes (in mm)

	x-Axis location			*y*-Axis location			*z*-Axis location		
Mean	0	−205.1	589.0	18.2	0	602.6	25.2	−224.3	0
RMSE	2.2			2.3			1.1		

Table 6
Locations of right wrist axes (in mm)

	x-Axis location			*y*-Axis location			*z*-Axis location		
Mean	0	214.3	575.0	−14.8	0	583.3	−15.8	241.2	0
RMSE	2.3			2.4			0.7		

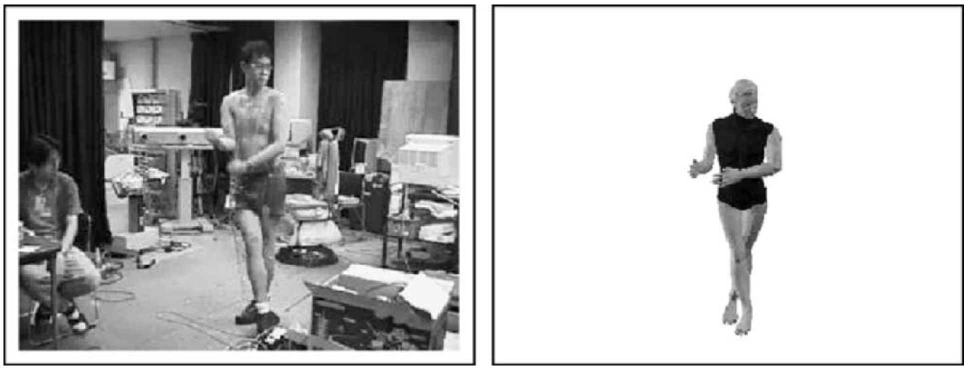
Fig. 3. Human demonstration and snapshot from the karate kata animation generated by wavelets.

at each hip to allow for more complex animations as in Fig. 3) was between 9.5 and 10.5 min on a 2 GHz Pentium 4 using 15–30 s long motion sequences captured at 60 Hz for each joint complex and starting from a generic kinematic model.

Figs. 3 and 4 compare the generated computer animations and humanoid robot motions with the captured data. The Okinawan dance movements were reproduced from motion capture sessions shorter than 10 s. Longer motions were generated by concatenation. Our experiments have shown that the rather limited kinematic structure of DB is sufficient to reproduce quite complex human motions. The karate kata animation was generated from a data sequence over 1.5 min long, although this sequence was divided into 10 s long subsequences to make the trajectory optimization problem manageable. The data were captured at 60 Hz. The generation of a 10 s long trajectory took about 6 min on a 2 GHz Pentium 4 with the lowest resolution fixed at $V^3(3)$ and the highest resolution fixed at $V^6(3)$. Most of the computation time was spent on the last few iterations when more wavelets are activated. One iteration step with more than half of the wavelets active takes about 1.5 min. On the other hand, a sequential approach can be implemented in real-time [12].

The experiments have also shown that our global trajectory generation approach offers some advantages over the straightforward approach of sequentially minimizing the criterion function (8). Firstly, the sequential minimization requires that a sufficient number of markers be visible at each measurement time. The optimization process can fail when too many of them are occluded. In the sequential approach, we were able to recover part of the Okinawan

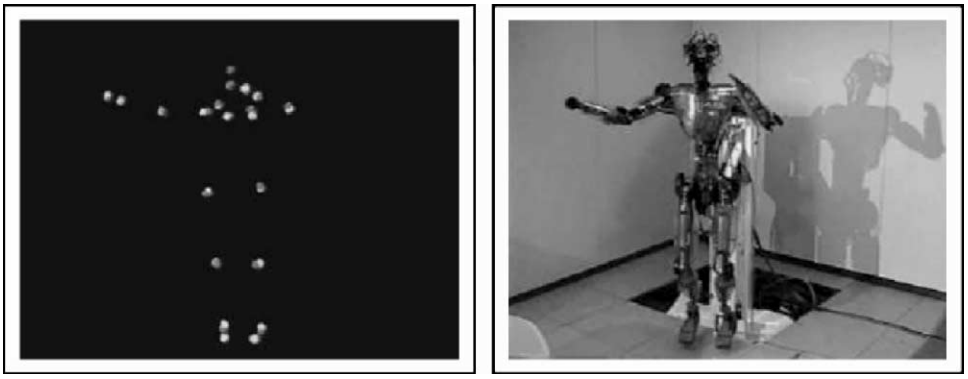
Fig. 4. Left image shows the captured markers (light gray) and the reconstructed marker motion (dark gray). Right image shows DB performing Okinawan dance generated by the sequential approach.

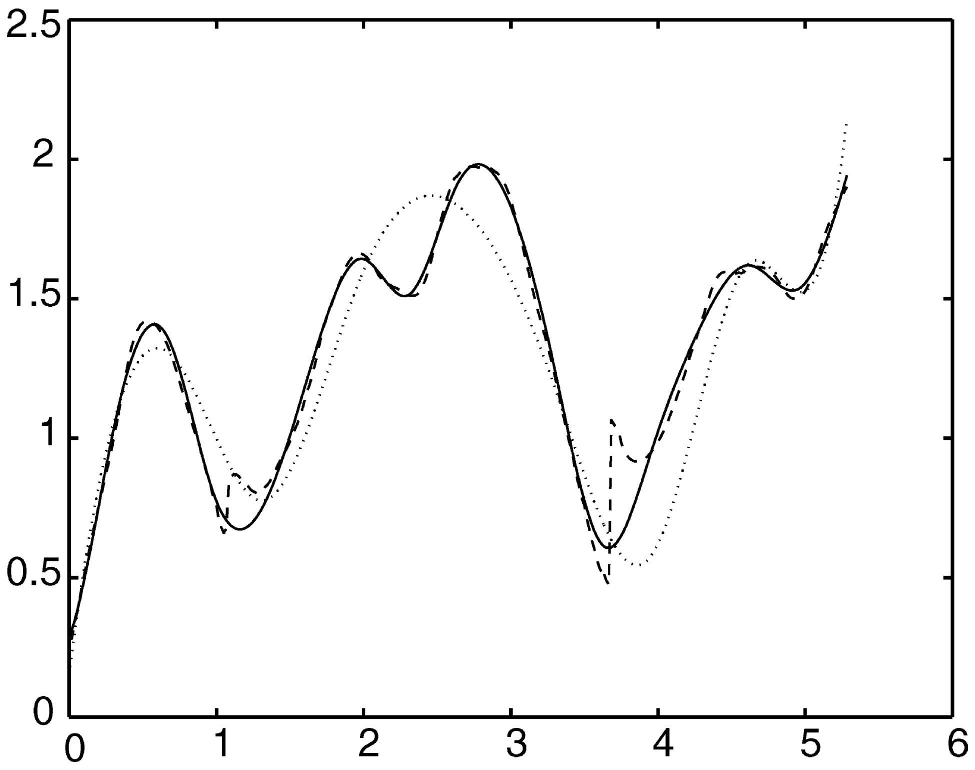

Fig. 5. Trajectory of the right arm flexion/extension.

trajectory only by interpolating the marker positions from the times when they were not occluded over the intervals of occlusion. But when there are too many consecutive missing markers, the interpolation process does not do a good job because it has no knowledge about the human motion. The missing markers problem was alleviated by the batch recovery of a complete trajectory. Secondly, it turned out that even when the positions of all markers can be measured, the recovery of some of the joint angles becomes sensitive at some configurations due to the kinematic singularities. This issue is resolved in the global approach with the introduction of the regularization factor (11). Thirdly, the recovered trajectory is smooth and can avoid discontinuities that can arise in the sequential approach because of the switching between local minima of the objective function.

The reconstructed trajectories are high dimensional (over 30 degrees of freedom), so due to space limitations we are unable to present all of their components. Some of the above-mentioned findings are nicely demonstrated on the recovery of the right shoulder flexion/extension degree of freedom (see Fig. 5). The dashed trajectory shows the trajectory recovered by the sequential approach. It is more noisy than the other two trajectories and it contains discontinuities. The dotted trajectory shows the recovered trajectory using an initial cubic B-spline basis (third resolution level, 11 basis functions). While it is less noisy than the trajectory generated by the sequential approach and it does not contain discontinuities, it only poorly follows the measured markers. The solid trajectory shows our final result after adding some wavelet functions. It is both smooth and continuous and it follows the measured markers better.

When the target character has a different size and proportion from the actual performer, we cannot simultaneously preserve the joint angles and end-effector positions. This leads to various artifacts such as foot sliding, which are well known to the computer animation community. We have indeed observed these artifacts in our animation experiments, but this was only a minor problem when mapping the reconstructed motion onto DB. It turned out that the mismatch between the joint limits of an actor and DB is a much more serious problem. We dealt with it by globally scaling and translating the reconstructed joint trajectories into the range of DBs joints. A local approach to scaling was proposed in [11]. It is also possible to replace the human joint limits with DBs joint limits in (9). This approach trades off joint errors and Cartesian target errors in a straightforward way and makes the robot see only feasible postures when interpreting or reasoning about the captured motion. Unfortunately, this approach also prevents the joints from moving when reaching joint limits, which is sometimes unnatural.

6. Summary and conclusions

We presented a fully automatic technique for the generation and synchronization of kinematic models describing human and humanoid robot motion. Our technique involves measuring marker positions at zero configuration and over a repertoire of motions exercising all relevant degrees of freedom. No manual measurement of the performer's limb lengths is necessary. We exploited the sparseness of the Jacobian matrices to solve the resulting optimization problems efficiently.

The second important contribution of this paper is a new approach to the formulation and optimization of joint trajectories for humanoid robots using B-spline wavelets. We demonstrated that B-spline wavelets are suitable for the formulation and optimization of humanoid robots' trajectories at different resolution levels and showed how to resolve the resulting large-scale optimization problems to compute such trajectories. The ability to treat large-scale optimization problems that need to be solved to generate optimal full-body motions and to automatically infer the appropriate resolution level draws a distinction between our approach and other approaches proposed for human motion capture in the literature.

The difference between motion capabilities of a human performer and of a humanoid robot is a serious problem when converting motion capture data. Constraining the joint limits in the optimization criterion does result in feasible motions, but the 'style' or 'essence' of motion might suffer. A possible field for further research is the development of additional criterion functions that improve the style of motion and that can be added to the objective function.

Acknowledgements

Support for all authors was provided by the Communications Research Laboratory (CRL) of Japan. C.A. is also supported by the US National Science Foundation under award ECS-0325383.

References

[1] T. Asfour, R. Dillmann, Human-like motion of a humanoid robot arm based on a closed form solution of the inverse kinematics problem, in: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems, Las Vegas, NV, October 2003, pp. 1407–1412.

[2] B. Bodenheimer, C. Rose, The process of motion capture: Dealing with the data, in: Proceedings of the Eurographics Workshop on Computer Animation and Simulation'97, Springer, 1997, pp. 3–18.

[3] G. Cheng, A. Nagakubo, Y. Kuniyoshi, Continuous humanoid interaction: An integrated perspective—gaining adaptivity, redundancy, flexibility—in one, Robotics and Autonomous Systems 37 (2001) 161–183.

[4] T. Coleman, M.A. Branch, A. Grace, Optimization Toolbox User's Guide, The MathWorks, Natick, MA, 1999.

[5] S.J. Gortler, M.F. Cohen, Hierarchical and variational geometric modeling with wavelets, in: Proceedings of the 1995 Symposium on Interactive 3D Graphics, Monterey, CA, April 1995, pp. 35–42.

[6] A.J. Ijspeert, J. Nakanishi, S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, 2002, pp. 1398–1403.

[7] B. Karan, M. Vukobratović, Calibration and accuracy of manipulation robot models—An overview, Mechanism and Machine Theory 29 (3) (1994) 479–500.

[8] Y. Kuroki, B. Blank, T. Mikami, P. Mayeux, A. Miyamoto, R. Playter, K. Nagasaka, M. Raibert, M. Nagano, J. Yamaguchi, Motion creating system for a small biped entertainment robot, in: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems, Las Vegas, NV, October 2003, pp. 1394–1399.
[9] R.M. Murray, Z. Li, S.S. Sastry, A Mathematical Introduction to Robotic Manipulation, CRC Press, Boca Raton, FL, 1994.
[10] J.F. O'Brien, R.E. Bodenheimer, G.J. Brostow, J.K. Hodgins, Automatic joint parameter estimation from magnetic motion capture data, in: Proceedings of the Graphics Interface 2000 Conference, Montreal, Canada, 2000.
[11] N.S. Pollard, J.K. Hodgins, M. Riley, C.G. Atkeson, Adapting human motion for the control of a humanoid robot, in: Proceedings of the IEEE International Conference on Robotics and Automation, Washington, May 2002, pp. 1390–1397.
[12] M. Riley, A. Ude, K. Wade, C.G. Atkeson, Enabling real-time full body imitation: A natural way of transferring human movements to humanoids, in: Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, Taiwan, September 2003, pp. 2368–2374.
[13] M.-C. Silaghi, R. Plänkers, R. Boulic, P. Fua, D. Thalmann, Local and global skeleton fitting techniques for optical motion capture, in: Modeling and Motion Capture Techniques for Virtual Environments, Lecture Notes in Computer Science, vol. 1537, Springer, Berlin, 1998, pp. 26–40.
[14] E.J. Stollnitz, T.D. DeRose, D.H. Salesin, Wavelets for Computer Graphics: Theory and Applications, Morgan Kaufmann, San Francisco, CA, 1996.
[15] A. Ude, C.G. Atkeson, On line tracking and mimicking of human movements by a humanoid robot, Advanced Robotics 17 (2) (2003) 165–178.

Aleš Ude studied applied mathematics at the University of Ljubljana, Slovenia, and computer science at the University of Karlsruhe, Germany, where he received his Doctoral degree in 1996. From 1998 to 2000 he was an STA fellow in the Kawato Dynamic Brain Project, ERATO, JST. Currently he holds a research position at the Jožef Stefan Institute, Ljubljana, Slovenia, and is also associated with the ATR Computational Neuroscience Laboratories, Kyoto, Japan. His research interests include humanoid robot vision and visual perception of human activity.

Christopher G. Atkeson is an Associate Professor at the Robotics Institute and the Human–Computer Interaction Institute at Carnegie Mellon University. His research focuses on numerical approaches to machine learning, and uses robotics and intelligent environments as domains in which to explore the behavior of learning algorithms. He is a recipient of a National Science Foundation Presidential Young Investigator award.

Marcia Riley is a researcher at the ATR Human Information Science Laboratories in Kyoto, Japan, and a Ph.D. candidate at the Georgia Institute of Technology in the College of Computing. Her research focuses on creating biologically-inspired robust behaviors for humanoids using a combination of approaches from intelligent systems, computer animation and computational neuroscience. She holds her M.S. in computer science from the University of Geneva in Switzerland, and B.S. in mathematics from the Johns Hopkins University.