Ball Joints for Marker-less Human Motion Capture

Gerard Pons Moll and Bodo Rosenhahn
Institut für Informationsverarbeitung (TNT), Leibniz Universität Hannover,
Hannover, Germany
{pons,rosenhahn}@tnt.uni-hannover.de

Abstract

This work presents an approach for the modeling and numerical optimization of ball joints within a Marker-less Motion Capture (MoCap) framework. In skeleton based approaches, kinematic chains are commonly used to model 1 DoF revolute joints. A 3 DoF joint (e.g. a shoulder or hip) is consequently modeled by concatenating three consecutive 1 DoF revolute joints. Obviously such a representation is not optimal and singularities can occur. Therefore, we propose to model 3 DoF joints with spherical joints or ball joints using the representation of a twist and its exponential mapping (known from 1 DoF revolute joints). The exact modeling and numerical optimization of ball joints requires additionally the adjoint transform and the logarithm of the exponential mapping. Experiments with simulated and real data demonstrate that ball joints can better represent arbitrary rotations than the concatenation of 3 revolute joints. Moreover, we demonstrate that the 3 revolute joints representation is very similar to the Euler angles representation and has the same limitations in terms of singularities.

1. Introduction

In this paper, we deal with the task of human pose tracking, also known as motion capturing (MoCap) [11]. For this task, a 3D model of the person and at least one calibrated camera view is available. We are interested in finding the 3D rigid body motion (RBM) of the person, i.e. its pose relative to the camera and the joint angles of the limbs. The motion of the 3D model is usually parameterized by a global RBM that encodes the overall position of the body in the world coordinate system and by a set of joint angles forming a kinematic chain that encode the pose configuration of the body. The motion of one body segment can be described as the motion of the previous segment in the kinematic chain and an angular motion about a joint axis. Rotation about joint axis can be represented by Euler angles, quaternions [9, 10] or exponential coordinates [3, 4, 14], see [7] for a comparison of rotation parameterizations. Euler angles represent a rotation in 3D by 3 consecutive rotations about 3 different axis. Quaternions represent 3D rotations in much the same way as complex numbers represent planar rotations in the unit circle. Unlike Euler angles quaternions provide a global parameterization of $SO(3)$, at the expense of using 4 numbers. Bregler *et al.* [3] introduced the twist and the product of exponential formulation in the context of human pose estimation. The exponential coordinates parameterize the space of affine isometries, $SE(3)$ as a function of a rotation axis and a translation along this axis. Since the motion of the limbs consist of a rotation about a given axis, this formalism is very convenient to represent human motion. Twist based methods [3, 4, 14], usually parameterize RBM by a twist consisting of 6 degrees of freedom (DoF) and model each additional DoF in the kinematic chain with a revolute joint, i.e. a rotation with zero pitch about a constant axis location and orientation. For revolute joints the variable is given by an angle $\theta_i \in [0, 2\pi)$. Consequently, 3 DoF joints (e.g. a shoulder or hip) are modeled by the concatenation of 3 revolute joints that intersect at the same location, see Figure 1b. Obviously, this representation is not

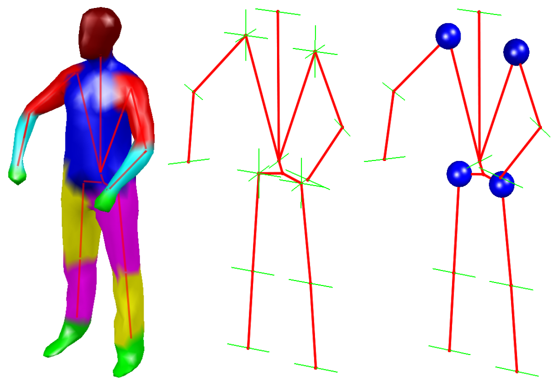

(a) (b) (c)

Figure 1: (a) Surface model mesh, vertices colored according to joint influence, (b) 3 Revolute joints skeleton model,(c) Ball Joints skeleton model

suitable for numerical optimization and like Euler angles it is not free from "gimbal lock". Therefore, we propose to use ball-and-socket joints parameterized in exponential coordinates for modeling body segments capable of arbitrary rotations, we will also refer to them as ball joints.

The modeling of ball joints is interesting for medical applications [8] and computer animation [6, 5], especially for the realistic interpolation of key poses as discussed in [1, 7]. A more anatomical exact model of the shoulder joint was proposed by [12]. Here the focus lies on the modeling of several redundantly involved shoulder parts which form the shoulder joint (and its deformation). In [10] they use real data taken from optical markers to learn the field boundaries of a quaternion based ball-and-socket joint. In [9] a hierarchical model for joint limits (e.g. a manifold for possible joint shoulder and elbow configurations) is proposed to penalize body intersections during tracking.

Researchers working in the area of computer vision usually prefer simplified human kinematic models that can be easily related to image features and allow for a simple control of the dynamics. To this end, in this work we focus on the integration of ball joints in a tracking system, specifically we concentrate on the linearization and parameter optimization from image features. We show that ball joints parameterized in exponential coordinates can be related linearly to image correspondences as the commonly used 3 revolute joints, and yet have a better performance. We present the improvements in a tracking system using ball joints in comparison to three coupled revolute joins on both simulated data and real sequences recorded in a lab setup. This paper is organized as follows, in Section 2 we recall the basics of twists, the exponential map and the logarithm, in Section 3 we introduce the mathematical foundations to incorporate ball joints in a human kinematic modeled by means of twists, in Section 4 we detail our experiments and finally we discuss the results in Section 5.

2. Twists and exponential maps

This section recalls the basics of twists, exponential maps and logarithm, for further details we refer to [13]. Every 3D rigid motion can be represented in an exponential form

$$M = \exp(\theta\widehat{\xi}) = \exp\begin{pmatrix} \widehat{\omega} & v \\ 0_{3\times 1} & 0 \end{pmatrix} \tag{1}$$

where $\theta\widehat{\xi}$ is the matrix representation of a twist $\xi \in se(3) = \{(v, \widehat{\omega}) | v \in \mathbb{R}^3, \widehat{\omega} \in so(3)\}$ with $so(3) = \{A \in \mathbb{R}^{3\times 3} | A = -A^T\}$. The Lie algebra $so(3)$ is the tangential space of the 3D rotations. Its elements are (scaled) rotation axes, which can be represented either as 3D vector:

$$\theta\omega = \theta \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}, \; with \; \|\omega\|_2 = 1 \tag{2}$$

or as a screw symmetric matrix:

$$\theta\widehat{\omega} = \theta \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{3}$$

Consider a 3D point p rotating about an axis ω intersecting the origin with an angular velocity θ. The matrix multiplication of $\theta\widehat{\omega}$ with the point p is equivalent to the cross product of the rotation axis and the point, $\theta\widehat{\omega}p = \theta\omega \times p$, and results in the tangential velocity of the rotating point.
A twist contains six parameters and can be scaled to θ with a unit vector ω. The parameter $\theta \in \mathbb{R}$ corresponds to the motion velocity (i.e., the rotation velocity and pitch). For varying θ, the motion can be identified as screw motion around an axis in space. The six twist components can be represented either as a 6D vector or as a 4×4 matrix:

$$\theta\xi = \theta(\omega_1, \omega_2, \omega_3, v_1, v_2, v_3) \tag{4}$$

$$\theta\widehat{\xi} = \theta \begin{pmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{5}$$

2.1. $\mathrm{se}(3)$ to $\mathrm{SE}(3)$: From Twist to Homogeneous matrix

To reconstruct a group action $g \in SE(3)$ from a given twist, the exponential function $\exp(\theta\widehat{\xi}) = g \in SE(3)$ must be computed. This can be done efficiently by using the Rodriguez formula,

$$\exp(\theta\widehat{\xi}) = \begin{pmatrix} \exp(\theta\widehat{\omega}) & (I - \exp(\theta\widehat{\omega}))(\omega \times v + \omega\omega^T v\theta) \\ 0_{1\times 3} & 1 \end{pmatrix} \tag{6}$$

with $\exp(\widehat{\omega})$ computed by calculating

$$\exp(\theta\widehat{\omega}) = I + \widehat{\omega}\sin(\theta) + \widehat{\omega}^2(1 - \cos(\theta)) \tag{7}$$

Note that only sine and cosine functions of real numbers need to be computed.

2.2. $\mathrm{SE}(3)$ to $\mathrm{se}(3)$: From Homogeneous matrix to Twist, the Logarithm

In [13], a constructive way is given to compute the twist which generates a given rigid body motion. Let $R \in SO(3)$ a rotation matrix and $t \in R^3$ a translation vector for the rigid body motion

$$g = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \tag{8}$$

For the case $R = I$, the twist is given by

$$\theta\xi = \theta(0, 0, 0, \frac{t}{\|t\|}), \qquad \theta = \|t\| \tag{9}$$

For the other cases, the motion velocity θ and the rotation axis ω are given by

$$\theta = \cos^{-1}\left(\frac{tr(R)-1}{2}\right), \quad \omega = \frac{1}{2\sin(\theta)}\begin{pmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{pmatrix} \tag{10}$$

To obtain v, the matrix,

$$A = (I - \exp(\theta\widehat{\omega}))\widehat{\omega} + \omega\omega^T\theta \tag{11}$$

obtained from the Rodriguez formula needs to be inverted and multiplied with the translation vector t,

$$v = A^{-1}t \tag{12}$$

We call the transformation from $SE(3)$ to $se(3)$ the logarithm, $\log(g)$.

3. The tracking system

This section is divided into 5 subsections. Subsections 3.1 and 3.2 describe our tracking system and numerical optimization approach. In Subsection 3.3 we describe the types of existing joints in the kinematic chain. In Subsection 3.4 we introduce the mathematical foundations needed for the linearization and integration of ball joints into the numerical optimization scheme. Finally, in Subsection 3.5 we explain how to update the twist parameters from frame to frame, specifically the adjoint transformation and the logarithm map need to be computed for the ball joint parameters.

3.1. Correspondences with ICP

In order to relate the surface model to the human's images we find correspondences between the 3D surface vertices and the 2D image contours obtained with background subtraction, Figure 2. We first collect 2D-2D correspondences by matching the projected surface silhouette with the background subtracted image contour. Thereby, we obtain a collection of 2D-3D correspondences since we know the 3D counterparts of the projected 2D points of the silhouette. These correspondences are used by the pose estimation module described in the subsequent section to determine the new pose parameters. The correspondences are iteratively updated according to the new pose parameters until the overall pose converges. This approach is commonly referred to as *Iterated Closest Point* (ICP) [2, 15].

3.2. Pose Estimation

For pose estimation we assume a set of point correspondences (X_i, x_i), with 4D (homogeneous) model points X_i and 3D (homogeneous) image points x_i. Each image point defines a 3D Plücker line $L_i = (n_i, m_i)$ (projective ray), with a (unit) direction n_i and moment m_i [13]. We combine

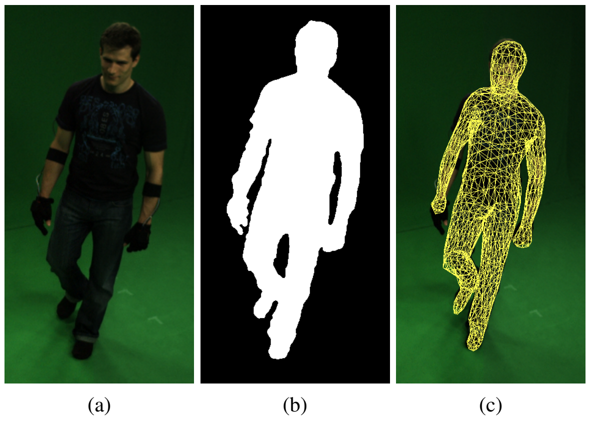


Figure 2: (a) Original Image, (b) Background subtracted image, (c) Projected surface mesh after convergence

the reconstructed Plücker lines with the screw representation for rigid motions and apply a gradient descent method: Incidence of the transformed 3D point X_i with the 3D ray $L_i = (n_i, m_i)$ can be expressed as

$$(\exp(\theta\widehat{\xi})X_i)_{3\times1} \times n_i - m_i = 0 \tag{13}$$

Indeed, X_i is an homogeneous 4D vector and after multiplication by the 4×4 matrix $\exp(\theta\widehat{\omega})$ we neglect the homogeneous component (which is 1) to evaluate the cross product with ni. Similar to Bregler *et al.* [3] we now linearize the Equation by using $\exp(\theta\widehat{\xi}) = \sum_{k=0}^{\infty}\frac{(\theta\widehat{\xi})^k}{k!}$, with I as identity matrix. This results in

$$((I + \theta\widehat{\xi})X_i)_{3\times1} \times n_i - m_i = 0 \tag{14}$$

and can be re-ordered into an equation of the form $A\xi = b$. Collecting a set of such equations (each is of rank two) leads to an over-determined system of equations, which can be solved using, for example, the Householder algorithm. The Rodriguez formula can be applied to reconstruct the group action g from the estimated twist ξ. Then the 3D points can be transformed and the process is iterated until convergence, see Figure 3 for a geometric interpretation.

3.3. Kinematic chains

There exist three different kinds of joints in our kinematic chain, see Table 1, the root joint, already explained in the previous section, ball joints capable of any rotation about a known joint location and simple revolute joints capable of only rotations about a fixed known axis. This last category is very convenient to constrain the motion of 1 DoF joints (e.g the knee). Revolute joints are expressed as special screws with no pitch of the form $\theta_j\widehat{\xi_j}$ with known ξ_j (the location of the rotation axes as part of the model representation) and unknown joint angle θ_j (saddle joints can be

Joint	**DoF**	**Unknown parameter**	**Example**
Root	6	$\xi = (\theta(v,\omega))$	All body
Ball	3	$\theta\omega$	Shoulders
Saddle	2	θ_1, θ_2	Wrist
Revolute	1	θ	Knee

Table 1: Table of existing joints to model human motion

modelled coupling two revolute joints since "gimbal lock" is not possible). Similar to revolute joints, ball joints are also expressed as special screws with no pitch with known joint location q but with unknown rotation axis $\theta\omega$. The scaled rotation axis $(\theta\omega)$ is then a free parameter we want to find. The constraint equation of a K_{th} joint has the form

$$(\prod_{j=0}^{j=K} \exp(\theta_j \xi_j) X_i)_{3\times 1} \times n_i - m_i = 0 \qquad (15)$$

which is linearized in the same way as the rigid body motion itself:

$$((I + \sum_{j=0}^{j=K} \theta_j \widehat{\xi_j})\ X_i)_{3\times 1} \times n_i - m_i = 0 \qquad (16)$$

For a kinematic model with 1 root joint, K revolute joints and L ball joints, Equation (16) leads to three linear equations with the six unknown pose parameters, K unknown joint angles and L unknown scaled screw axis. Collecting a sufficient number of equations leads to an over-determined system of equations of the form $A\Theta = b$ where $\Theta = (\underbrace{\xi_{6\times 1}}_{1\ root}, \underbrace{\ldots, \theta\omega^l_{3\times 1}, \ldots}_{L\ ball\ joints}, \underbrace{\ldots, \theta^k_{1\times 1}, \ldots}_{K\ revolute\ joints})$ is the vector of unknown parameters that determine the body configuration. The details of the error minimized in Equation (15) and its linearization in Equation (16) are explained in the following subsection, specifically we focus on the part of the linear system concerning the ball joint parameters.

3.4. Ball joints vs 3 Revolute joints

The orientation of a 3 DoF joint represented by the concatenation of 3 revolute joints is given by

$$R = e^{\theta_1\omega_1} e^{\theta_2\omega_2} e^{\theta_3\omega_3} \qquad (17)$$

where $\omega_1, \omega_2, \omega_3 \in \mathbb{R}^3$ represent the directions of each of the three axes that intersect at the joint location. This representation is very similar to an Euler angle representation and has the same limitations in terms of singularities. It is easy to see that the rotation representation in Equation (17) is singular for $\theta_2 =^+_- \frac{\pi}{2}$. In this singular configuration the first and third axis ω_1, ω_3 are parallel and one axis of rotation is lost. This is commonly known as a gimbal lock configuration. In particular we note that in such configuration,

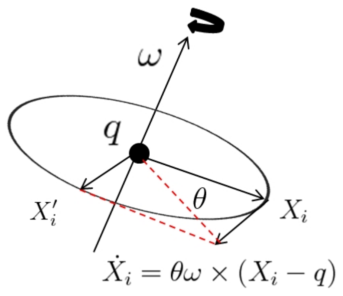


Figure 3: Optimization: $\dot{X}_i$ is the tangential velocity of the point X_i and X_i' is the 3D point that projects to the 2D image correspondence x_i. Given a correspondence (X_i, X_i'), the distance between the point X_i' and X_i translated in the tangential direction of the rotation is minimized. The action group is recovered by taking the exponential map on the estimated *twist* parameters. Then the point X_i is transformed and the process is iterated until convergence.

any rotation by $\theta_1 = \alpha$ degrees about the first axis followed by the same negative rotation about the third axis $\theta_3 = -\alpha$ leads to the same initial configuration. Actually, near singularities, the size of the joint velocities needed to maintain an end effector velocity can be extremely large and indeed lead to numerically unstable solutions during optimization as we show in Section 4. Furthermore, during tracking the mutual orthogonality between the first and third axes is lost, which results in a redundancy in the rotations that translates in a slower convergence. In order to avoid all these problems, we propose to model the 3 DoF joints with spherical joints or also known in the robotics community as ball joints. A segment connected to a ball joint is capable of any arbitrary rotation. The concatenation of 3 revolute joints is preferred in the robotics community because the mechanism works better to exert forces and torques. However, since for the task of human motion estimation this is not needed at all, ball joints provide a more elegant and natural representation of limb rotations that does not suffer from the shortcomings of the previous representation. Similarly to revolute joints, being q the location and ω the axis of rotation of the joint, the twist corresponding to a ball joint is given by:

$$\xi = \begin{bmatrix} -\omega \times q \\ \omega \end{bmatrix} \qquad (18)$$

The difference now is that the axis of rotation ω is not constant but rather a free parameter. Thereby, the three unknown angles, $\theta_1, \theta_2, \theta_3$, about 3 constant axis are replaced by a single instantaneous scaled axes of rotation $\theta\omega$. The tangential velocity of a point X_i, influenced by a ball joint, rotating about an unknown axis $\theta\omega$ is given by :

$$\dot{X}_i = \theta\omega \times (X_i - q) \qquad (19)$$

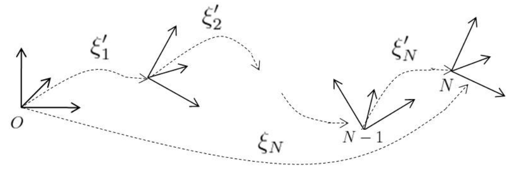


Figure 4: Absolute Twists: ξ'_N the relative twist from frame $N-1$ to frame N; ξ_N is the absolute twist from frame 0 to frame N

where X_i is a 3D point and q is the joint location where the axis ω intersects 3. Rewriting the cross product in matrix form and using the properties of the crossproduct [1] we can isolate ω,

$$\dot{X}_i = \theta\widehat{\omega}(X_i - q) = -(X_i - q)^\wedge \theta\omega \tag{20}$$

where the operator $^\wedge$ (wedge) maps a vector $a \in \mathbb{R}^3$ into a screw symmetric matrix $A \in so(3)$ which is the matrix representation of the cross product [2]. Thereby, distance to the projection rays can be written in matrix form as

$$(X_i + \dot{X}_i) \times n_i - m_i = -\widehat{n_i}(X_i + \dot{X}_i) - m_i = 0 \tag{21}$$

and substituting $\dot{X}_i$ by the expression in (20), we can isolate $\theta\omega$,

$$\widehat{n_i}(X_i - q)^\wedge \theta\omega = \widehat{n_i} X_i + m_i \tag{22}$$

to obtain three independent equations for each of the components of $\theta\omega$ that can be integrated into the big linear system. Note that expression (22) already has the form of $A\theta\omega = b$, with $A_{3\times3} = \widehat{n_i}(X_i - q)^\wedge$.

3.5. From differential to absolute twists

In the previous subsections we have shown the mathematical tools to track a body object from frame to frame. This implies that the solution of Equation (15) is the relative twist from consecutive pose configurations. During tracking the surface model is updated with this differential twists to keep track of the human pose. Although this is enough for tracking, it is also interesting to recover the absolute pose parameters, i.e. the parameters that bring the body from the initial configuration in the first frame to the current frame. The motivation is to obtain twist parameters that represent the total accumulated motion which makes the task of learning the probability densities of the parameters much easier. We denote ξ'_N the relative twist from frame $N-1$ to frame N, and ξ_N the absolute twist from frame 0 to frame N as depicted in Figure 4. For the sake of clarity, let us drop the magnitude of the twist θ. Just bear in mind that now all the twists ξ are scaled twists by their own magnitude θ. The absolute twist of the global RBM ξ_N (root joint) is recovered by updating the action group $\exp(\xi_{N-1})$ left multiplying it with the current relative motion and taking the logarithm.

$$\xi_N = \log\left(\prod_{n=0}^{n=N} \exp(\xi'_n)\right) = \log(\exp(\xi'_N)\exp(\xi_{N-1})) \tag{23}$$

The joint angles of simple revolute joints can be updated by just adding up all the relative angles,

$$\theta_N = \theta_{N-1} + \theta'_N \tag{24}$$

This is possible because the constant twists of revolute joints are updated with the rigid motion of upper segments in the kinematic chain. Like the root joint absolute twists of ball joints are recovered computing the logarithm map as in Equation (23), however these need to be taken more carefully because they are influenced by motion of other joints in the chain (e.g the arms are influenced by the global rigid motion and a spine joint). Hence, the twists need first to be transformed from a fixed spatial coordinate system to the body coordinate system. We use the same notation as in [13] with ξ^s denoting a twist in spatial coordinates (a fixed inertial reference frame) and ξ^b denoting the same twist in the body frame coordinates. This is achieved by adapting the twist by means of the so called adjoint transformation (Ad):

$$\begin{aligned} \widehat{\xi}^b_n &= g^{-1}_{n-1}\, \widehat{\xi}^s_n\, g_{n-1} && (25) \\ \xi^b_n &= Ad^{-1}_{g_{n-1}}\, \xi^s_n && (26) \end{aligned}$$

where g_{n-1} is the 4×4 RBM matrix that represents the accumulated motion of the parent joints of the ball joint in the chain. Hence, g_{n-1} is then the rigid body motion that brings the ball joint from the initial configuration in frame 0 to the frame $n-1$. After some calculation it is easy to derive the following expected expressions for the twist components ω^b, v^b in body coordinates,

$$\begin{aligned} \omega^b &= R^{-1}_{n-1}\, \omega^s && (27) \\ v^b &= -\omega^b \times q^b && (28) \end{aligned}$$

where R_{n-1} is the 3×3 rotation matrix corresponding to g_{n-1} and $q^b = g^{-1}_{n-1}\, q^s_{n-1}$ is the joint location in body coordinates. Note that q^b is constant during tracking.

4. Experiments

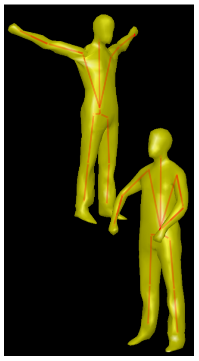
In this section we compare the performance of ball joints with three concatenated revolute joints. The experiments are divided into simulated data and real sequences in a controlled lab environment with 8 calibrated cameras. The experiments with simulated data can be further broken down into two subcategories; first, the convergence performance of a single motion is analyzed in detail and second, clouds

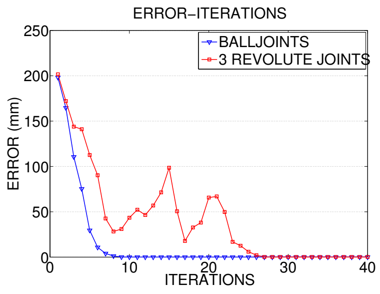
[1] $a \times b = -b \times a$
[2] $a \times b = (a)^\wedge b$

of motions are generated to test the behavior of both methods in a more general framework.

In the experiment shown in Figure 5b we analyze in detail the parameter convergence of both methods between the two different poses shown in Figure 5a. The two poses correspond to two distant frame poses of one of the real sequences. To obtain the correspondences we project all the vertices of the second pose mesh to the image plane and relate them to the 3D points in the first pose. Thereby, we are able to compare the two methods for large displacements. In Figure 5b, we plotted the error versus iterations, where the error minimized (Equation (15)) is the distance from the projection rays to the transformed $3D$ points of the first pose. We can see in Figure 5b how the ball joint representation, blue triangles graph, converges in less than 7 iterations while the coupled revolute joints take more than 25 iterations to converge. We also note that, convergence with ball joints is monotonically decreasing in contrast to revolute joints that fluctuates. To understand better the difference in performance, we show the evolution of the twist parameters of the right arm for both methods during optimization. On the one hand in Figure 6a we plotted the 3 unknown angles $\theta_1, \theta_2, \theta_3$ of the revolute joints representation and on the other hand in Figure 6b we plotted the the three components of the unknown scaled axis of rotation $\theta\omega$. We perfectly see in Figure 6a how the first and the third angles are completely mirrored. As explained in subsection 3.4, this redundancy is most remarkable for near singular pose configurations corresponding to $\theta_2 =^{+}_{-} \frac{\pi}{2}$. Actually, we can see in Figure 6a how θ_2, green dashed line, fluctuates near $\frac{\pi}{2}$. This generally results in a slower convergence, Figure 5b, and produces numerically unstable joint angle values much higher than 2π. In contrast, the non-constant twist axis of ball joints provides the shortest path, the geodesic, to travel from a point to its rotated version.

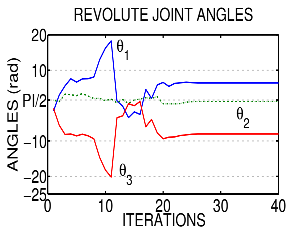
To test the performance of ball joints with respect to 3 revolute joints in a more general framework we have yet conducted another experiment. We first artificially generated 1000 poses with different arm orientations. To achieve that, we have sampled the space of pose configurations by varying the 3 angles defining the shoulder joint orientation using a 3 revolute joints representation $(\theta_1, \theta_2, \theta_3)$ of both arms. Specifically, we create 3D grid of $[10 \times 10 \times 10]$ samples allowing each angle to rotate 2π, $\theta_i(k) = \frac{2\pi}{10}k$. Thereafter, we obtained correspondences between each pair of poses as already explained early in the section in order to study the convergence performance. However, this time we are only interested in retrieving the number of iterations needed to convergence from an initial fixed pose to each of the 1000 generated poses. For this experiment the initial pose is selected such that the axis of the revolute joints are mutually orthogonal. We also note that the initial pose and each of the 1000 poses differ not only in the arm orientation


(a)

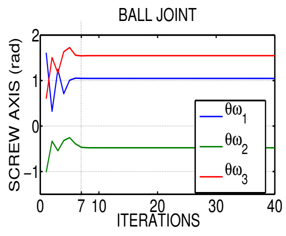


(b)

Figure 5: Error-Iterations: (a) Two poses used to compare the convergence, (b) Error versus iterations for ball joints, blue line with triangles, and coupling of revolute joints, red line with squares



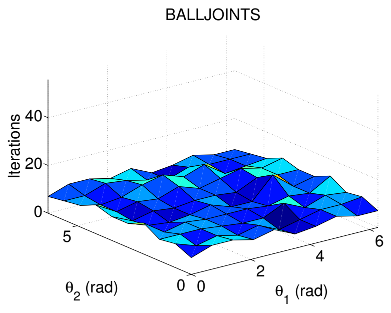
(a)



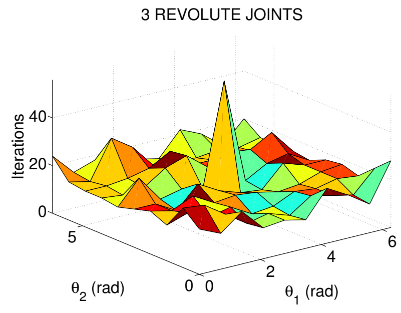
(b)

Figure 6: Parameter convergence for both methods: (a) The three angles θ_1, θ_3, in continuous lines and θ_2 in dashed line. (b) The three components of the Ball-Joint scaled axis $\theta\omega$

but also in several other parameters in the chain. Hence, each of the two pose pairs are related by more complex motions than a simple rotation about a joint. Thereby, for each triple $\theta_{1,2,3}$ we obtained the iterations needed to converge for both methods. We allow both methods to perform a maximum of 50 iterations because convergence above this threshold is highly unlikely. In Figure 7a we compare the methods by plotting one cross-section of the 3D grid for a constant $\theta_3 = 1.8$, obtaining a surface of iterations as a function of (θ_1, θ_2). We observe that the resulting surface using ball joints, Figure 7a, is much flatter and lower than the one using 3 revolute joints, Figure 7b. This confirms that ball joints are less dependent on the final configuration and represent general rotations more naturally. To get a better grasp of the overall performance we plotted the histogram of the iterations for both methods in Figure 8a; the lower mean and variance of the ball joints histogram is an indicative of faster convergence and independence of the final configuration. Additionally, we repeated the same

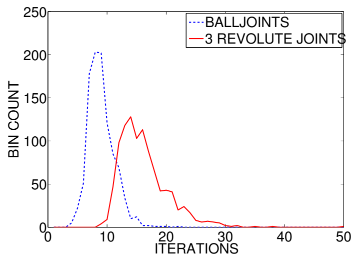


(a)

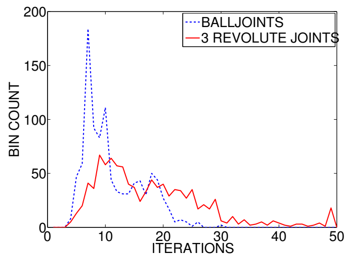


(b)

Figure 7: Iteration surfaces: Iterations for convergence from initial configuration to final poses, $\theta_3 = 1.8, (\theta_1, \theta_2) \in [0, 2\pi)$: (a) Ball Joints, (b) 3 Revolute Joints.

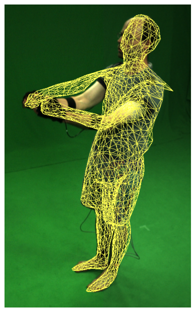


(a)

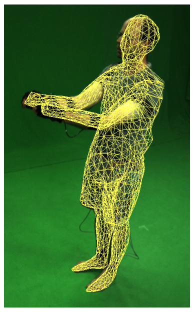


(b)

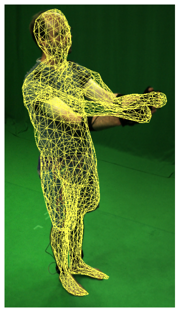
Figure 8: Histograms of iterations for both methods (Balljoints in dashed blue line, 3 revolute joints in continuous red line), 1000 poses computed varying the angles of the arm joint $(\theta_1, \theta_2, \theta_3) \in [0, 2\pi)$. (a) Histogram with orthogonal axis as starting pose, (b) Histogram with non-orthogonal axis as starting pose, (e.g. arms extended).



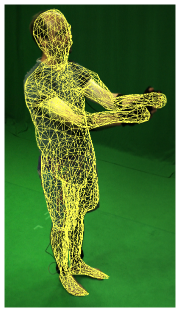
(a)



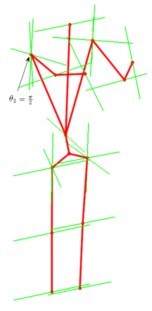
(b)



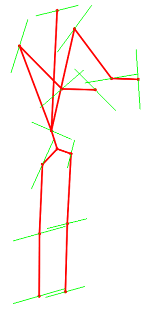
(c)



(d)



(e)



(f)

Figure 9: Mesh projection overlayed with different camera views: (a),(c) Tracking with revolute joints, (b),(d) Tracking with Ball Joints,(e) Skeleton with twist axis of revolute joints, (f) Skeleton with twist axis of ball joints

experiment but now starting from a non orthogonal configuration to show that the performance of 3 revolute joints is also very sensible on the current configuration of the 3 axis. See in Figure 8b how the 3 revolute joints histogram spreads towards more iterations and does not converge in 20 cases in comparison to the ball joints histogram that maintains the concentration around 9 iterations. In Figure 9 we show some examples of tracking results for both methods. In Figures 9a 9c we see problems that arise with the 3 revolute angles representation. Note that the right arm is not well estimated, Figures 9a9c, in contrast to ball joints, Figures 9b9d. The tracking error stems from a previous singular configuration in which the first and third axis almost align and one rotation is lost, see the right arm in the skeleton of

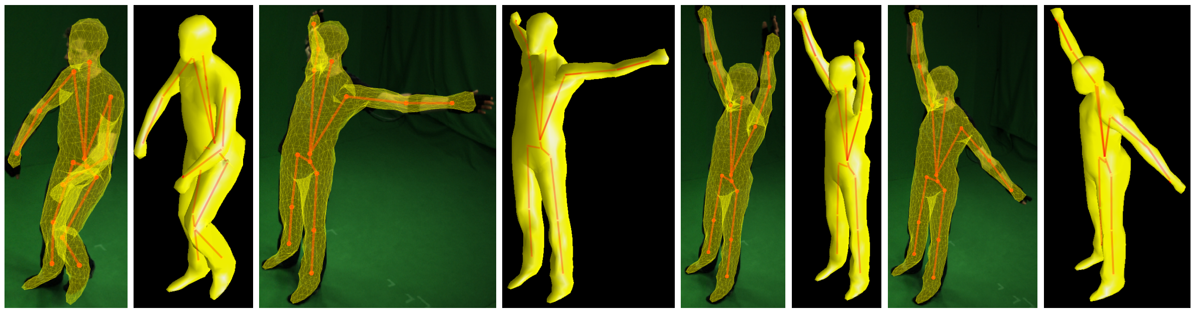


Figure 10: Tracking snapshots: 1025 frames sequence of an actor performing several fast arm rotations

Figure 9e, preventing the model to follow the image data. Local optimization methods might not recover from such situations and can easily degenerate. In contrast, tracking with ball joints poses no problem as the limbs rotate about an optimal single instantaneous axis, see Figure 9f. Finally, to conclude the experiments, we show several results of a tracked sequence using ball joints in Figure 10.

5. Conclusions

We proposed to use ball joints parameterized in exponential coordinates to model three DoF joints like the shoulders and hips. In this paper we focused on the integration of such joints in a tracking system. This requires linearization and numerical optimization. Additionally, the adjoint transformation and the logarithm of the exponential mapping must be computed to recover the absolute parameters. Experiments with simulated and real data demonstrate the advantages of using ball joints compared to using three revolute joints in a Marker-less Motion Capture system. First, ball joints provide a parameterization free of "gimbal lock". Second, their behavior in differential optimization methods is better for several reasons: convergence is less dependent on the initial and final pose parameters and redundancy between joints is no longer a problem thanks to the single instantaneous screw axis estimated at each iteration. Therefore, modeling with ball joints allow for a more stable and accurate capture of complex limb rotations without increasing the complexity of the system.

References

[1] P. Baerlocher and R. Boulic. Parametrization and range of motion of the ball-and-socket joint. In *Workshop on Deformable Avatars*, pages 180–190. Kluwer, B.V., 2001. 2

[2] P. Besl and N. McKay. A method for registration of 3D shapes. *tpami*, 12:239–256, 1992. 3

[3] C. Bregler and J. Malik. Tracking people with twists and exponential maps. *IEEE Conf. on Computer Vision and Patt. Recognition*, pages 8–15, 1998. 1, 3

[4] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. *International Journal of Computer Vision*, 56(3):179–194, 2004. 1

[5] M. da Silva, Y. Abe, and J. Popovic. Simulation of human motion data using short-horizon model-predictive control. *Comput. Graph. Forum*, 27(2):371–380, 2008. 2

[6] A. R. F. Remondino. Human motion reconstruction and animation from video sequences. In *17th International Conference on Computer Animation and Social Agents (CASA2004)*, pages 347–354, July 2004. 2

[7] F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3:29–48, 1998. 1, 2

[8] G. Guillard and N. Magnenat-Thalmann. Ball-and-socket joint motion description using spherical medial representation. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 4293–4296, Aug. 2007. 2

[9] L. Herda, R. Urtasun, and P. Fua. Implicit surface joint limits to constrain video-based motion capture. In T. Pajdla and J. Matas, editors, *Proc. 8th European Conference on Computer Vision*, volume 3022 of *lncs*, pages 405–418, Prague, May 2004. Springer. 1, 2

[10] L. Herda, R. Urtasun, A. Hanson, and P. Fua. Automatic determination of shoulder joint limits using experimentally determined quaternion field boundaries. *International Journal of Robotics Research*, 22(6), June 2003. 1, 2

[11] T. Moeslund and E. Granum. A survey of computer vision based human motion capture. 81(3):231–268, 2001. 1

[12] T. B. Moeslund, C. B. Madsen, and E. Granum. Modelling the 3d pose of a human arm and the shoulder complex utilising only two parameters. *Integrated Computer-Aided Engineering*, 12, 2005. 2

[13] R. Murray, Z. Li, and S. Sastry. *Mathematical Introduction to Robotic Manipulation.* CRC Press, Baton Rouge, 1994. 2, 3, 5

[14] B. Rosenhahn, T. Brox, and H.-P. Seidel. Scaled motion dynamics for markerless motion capture. In *IEEE Conf. on Computer Vision and Patt. Recognition*, pages 1–8. 1

[15] Z. Zhang. Iterative points matching for registration of free form curves and surfaces. *ijcv*, 13(2):119–152, 1994. 3